

libstdc++

Generated by Doxygen 1.8.11

Contents

1	Todo List	1
2	Module Documentation	2
2.1	Adaptors for pointers to functions	2
2.1.1	Detailed Description	2
2.1.2	Function Documentation	2
2.2	Adaptors for pointers to members	4
2.2.1	Detailed Description	4
2.3	Algorithms	5
2.3.1	Detailed Description	5
2.4	Allocators	6
2.4.1	Detailed Description	6
2.4.2	Typedef Documentation	6
2.5	Arithmetic Classes	8
2.5.1	Detailed Description	8
2.6	Associative	9
2.6.1	Detailed Description	9
2.7	Atomics	10
2.7.1	Detailed Description	11
2.7.2	Macro Definition Documentation	11
2.7.3	Typedef Documentation	12
2.7.4	Enumeration Type Documentation	16
2.7.5	Function Documentation	16
2.8	Base and Implementation Classes	17
2.8.1	Detailed Description	19
2.9	Base and Implementation Classes	20
2.9.1	Detailed Description	21

2.9.2 Enumeration Type Documentation	21
2.10 Base and Policy Classes	22
2.10.1 Detailed Description	22
2.11 Base and Policy Classes	23
2.11.1 Detailed Description	23
2.12 Base and Policy Classes	24
2.12.1 Detailed Description	24
2.13 Bernoulli Distributions	25
2.13.1 Detailed Description	26
2.13.2 Function Documentation	26
2.14 Binary Search	29
2.14.1 Detailed Description	29
2.14.2 Function Documentation	30
2.15 Binder Classes	34
2.15.1 Detailed Description	34
2.15.2 Function Documentation	35
2.16 Boolean Operations Classes	36
2.16.1 Detailed Description	36
2.17 Branch-Based	37
2.17.1 Detailed Description	37
2.18 Comparison Classes	38
2.18.1 Detailed Description	38
2.19 Concurrency	39
2.20 Containers	40
2.20.1 Detailed Description	40
2.21 Containers	41
2.21.1 Detailed Description	41
2.22 Data Structure Type	42

2.22.1 Detailed Description	42
2.23 Diagnostics	43
2.24 Exceptions	44
2.24.1 Detailed Description	44
2.25 Exceptions	45
2.25.1 Detailed Description	46
2.25.2 Function Documentation	46
2.26 Extensions	47
2.26.1 Detailed Description	47
2.27 Function Objects	48
2.27.1 Detailed Description	49
2.28 Hash-Based	50
2.28.1 Detailed Description	50
2.29 Hashes	51
2.29.1 Detailed Description	51
2.30 Heap	52
2.30.1 Detailed Description	52
2.30.2 Function Documentation	52
2.31 Heap-Based	58
2.31.1 Detailed Description	59
2.31.2 Function Documentation	59
2.32 Invalidation Guarantees	60
2.32.1 Detailed Description	60
2.33 Iterator Tags	61
2.33.1 Detailed Description	61
2.34 Iterators	62
2.34.1 Detailed Description	64
2.34.2 Function Documentation	65

2.35 List-Based	68
2.35.1 Detailed Description	68
2.36 Locales	69
2.36.1 Detailed Description	69
2.37 Mutating	70
2.37.1 Detailed Description	72
2.37.2 Function Documentation	72
2.38 Negators	90
2.38.1 Detailed Description	90
2.38.2 Function Documentation	91
2.39 Non-Mutating	92
2.39.1 Detailed Description	93
2.39.2 Function Documentation	93
2.40 Normal Distributions	105
2.40.1 Detailed Description	106
2.40.2 Function Documentation	106
2.41 Numeric_arrays	108
2.41.1 Detailed Description	111
2.41.2 Function Documentation	111
2.42 Pointer_abstractions	122
2.42.1 Detailed Description	124
2.42.2 Function Documentation	124
2.43 Poisson Distributions	126
2.43.1 Detailed Description	127
2.43.2 Function Documentation	127
2.44 Policy-Based Data Structures	131
2.44.1 Detailed Description	131
2.45 Random Number Distributions	132

2.45.1 Detailed Description	132
2.46 Random Number Generation	133
2.46.1 Detailed Description	133
2.46.2 Function Documentation	133
2.47 Random Number Generators	134
2.47.1 Detailed Description	135
2.47.2 Typedef Documentation	135
2.47.3 Function Documentation	136
2.48 Random Number Utilities	139
2.48.1 Detailed Description	139
2.49 Regular Expressions	140
2.49.1 Detailed Description	145
2.49.2 Typedef Documentation	145
2.49.3 Function Documentation	146
2.50 SGI	176
2.51 Sequences	177
2.51.1 Detailed Description	177
2.52 Set Operation	178
2.52.1 Detailed Description	178
2.52.2 Function Documentation	179
2.53 Sorting	184
2.53.1 Detailed Description	186
2.53.2 Function Documentation	186
2.54 Strings	203
2.54.1 Detailed Description	203
2.54.2 Typedef Documentation	203
2.55 Tags	204
2.55.1 Detailed Description	204
2.55.2 Typedef Documentation	204
2.56 Traits	205
2.56.1 Detailed Description	206
2.57 Uniform Distributions	207
2.57.1 Detailed Description	207
2.57.2 Function Documentation	207
2.58 Unordered Associative	210
2.58.1 Detailed Description	210
2.59 Utilities	211
2.59.1 Detailed Description	212
2.59.2 Function Documentation	212
2.59.3 Variable Documentation	217

3	Namespace Documentation	218
3.1	__gnu_cxx Namespace Reference	218
3.1.1	Detailed Description	225
3.1.2	Function Documentation	226
3.2	__gnu_cxx::__detail Namespace Reference	235
3.2.1	Detailed Description	236
3.2.2	Function Documentation	236
3.3	__gnu_cxx::typelist Namespace Reference	237
3.3.1	Detailed Description	237
3.3.2	Function Documentation	237
3.4	__gnu_debug Namespace Reference	237
3.4.1	Detailed Description	241
3.4.2	Enumeration Type Documentation	241
3.4.3	Function Documentation	242
3.5	__gnu_internal Namespace Reference	245
3.5.1	Detailed Description	245
3.6	__gnu_parallel Namespace Reference	245
3.6.1	Detailed Description	253
3.6.2	Typedef Documentation	253
3.6.3	Enumeration Type Documentation	254
3.6.4	Function Documentation	255
3.6.5	Variable Documentation	293
3.7	__gnu_pbds Namespace Reference	293
3.7.1	Detailed Description	295
3.8	__gnu_profile Namespace Reference	295
3.8.1	Detailed Description	299
3.8.2	Typedef Documentation	299
3.8.3	Function Documentation	299

3.9	__gnu_sequential Namespace Reference	300
3.9.1	Detailed Description	300
3.10	abi Namespace Reference	300
3.10.1	Detailed Description	300
3.11	std Namespace Reference	300
3.11.1	Detailed Description	368
3.11.2	Typedef Documentation	368
3.11.3	Enumeration Type Documentation	370
3.11.4	Function Documentation	370
3.12	std::__debug Namespace Reference	425
3.12.1	Detailed Description	426
3.13	std::__detail Namespace Reference	427
3.13.1	Detailed Description	429
3.14	std::__parallel Namespace Reference	429
3.14.1	Detailed Description	444
3.15	std::__profile Namespace Reference	444
3.15.1	Detailed Description	447
3.16	std::regex_constants Namespace Reference	447
3.16.1	Detailed Description	448
3.16.2	Enumeration Type Documentation	449
3.16.3	Function Documentation	451
3.17	std::rel_ops Namespace Reference	455
3.17.1	Detailed Description	456
3.17.2	Function Documentation	456
3.18	std::tr1 Namespace Reference	457
3.18.1	Detailed Description	457
3.19	std::tr1::__detail Namespace Reference	458
3.19.1	Detailed Description	458
3.20	std::tr2 Namespace Reference	458
3.20.1	Detailed Description	458
3.21	std::tr2::__detail Namespace Reference	458
3.21.1	Detailed Description	458

4	Class Documentation	458
4.1	__cxxabiv1::__forced_unwind Class Reference	458
4.1.1	Detailed Description	458
4.2	__gnu_cxx::__alloc_traits<_Alloc> Struct Template Reference	459
4.2.1	Detailed Description	460
4.2.2	Member Typedef Documentation	460
4.2.3	Member Function Documentation	461
4.3	__gnu_cxx::__common_pool_policy<_PoolTp, _Thread> Struct Template Reference	464
4.3.1	Detailed Description	464
4.4	__gnu_cxx::__detail::__mini_vector<_Tp> Class Template Reference	464
4.4.1	Detailed Description	465
4.5	__gnu_cxx::__detail::__Bitmap_counter<_Tp> Class Template Reference	465
4.5.1	Detailed Description	465
4.6	__gnu_cxx::__detail::__Ffit_finder<_Tp> Class Template Reference	466
4.6.1	Detailed Description	466
4.6.2	Member Typedef Documentation	466
4.7	__gnu_cxx::__mt_alloc<_Tp, _Poolp> Class Template Reference	467
4.7.1	Detailed Description	468
4.8	__gnu_cxx::__mt_alloc_base<_Tp> Class Template Reference	468
4.8.1	Detailed Description	469
4.9	__gnu_cxx::__per_type_pool_policy<_Tp, _PoolTp, _Thread> Struct Template Reference	469
4.9.1	Detailed Description	469
4.10	__gnu_cxx::__pool<_Thread> Class Template Reference	470
4.10.1	Detailed Description	470
4.11	__gnu_cxx::__pool<false> Class Template Reference	470
4.11.1	Detailed Description	471
4.12	__gnu_cxx::__pool<true> Class Template Reference	471
4.12.1	Detailed Description	472

4.13	__gnu_cxx::__pool_alloc< _Tp > Class Template Reference	472
4.13.1	Detailed Description	474
4.14	__gnu_cxx::__pool_alloc_base Class Reference	474
4.14.1	Detailed Description	475
4.15	__gnu_cxx::__pool_base Struct Reference	475
4.15.1	Detailed Description	476
4.16	__gnu_cxx::__rc_string_base< _CharT, _Traits, _Alloc > Class Template Reference	476
4.16.1	Detailed Description	478
4.17	__gnu_cxx::__scoped_lock Class Reference	479
4.17.1	Detailed Description	479
4.18	__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > Class Template Reference	479
4.18.1	Detailed Description	482
4.18.2	Constructor & Destructor Documentation	483
4.18.3	Member Function Documentation	486
4.18.4	Member Data Documentation	536
4.19	__gnu_cxx::__Caster< _ToType > Struct Template Reference	536
4.19.1	Detailed Description	536
4.20	__gnu_cxx::__Char_types< _CharT > Struct Template Reference	537
4.20.1	Detailed Description	537
4.21	__gnu_cxx::__ExtPtr_allocator< _Tp > Class Template Reference	537
4.21.1	Detailed Description	538
4.22	__gnu_cxx::__Invalid_type Struct Reference	539
4.22.1	Detailed Description	539
4.23	__gnu_cxx::__Pointer_adapter< _Storage_policy > Class Template Reference	539
4.23.1	Detailed Description	541
4.24	__gnu_cxx::__Relative_pointer_impl< _Tp > Class Template Reference	541
4.24.1	Detailed Description	542
4.25	__gnu_cxx::__Relative_pointer_impl< const _Tp > Class Template Reference	542

4.25.1 Detailed Description	542
4.26 __gnu_cxx::Std_pointer_impl<_Tp> Class Template Reference	543
4.26.1 Detailed Description	543
4.27 __gnu_cxx::Unqualified_type<_Tp> Struct Template Reference	543
4.27.1 Detailed Description	543
4.28 __gnu_cxx::annotate_base Struct Reference	544
4.28.1 Detailed Description	544
4.29 __gnu_cxx::array_allocator<_Tp, _Array> Class Template Reference	545
4.29.1 Detailed Description	546
4.30 __gnu_cxx::array_allocator_base<_Tp> Class Template Reference	546
4.30.1 Detailed Description	547
4.31 __gnu_cxx::bitmap_allocator<_Tp> Class Template Reference	547
4.31.1 Detailed Description	548
4.31.2 Member Function Documentation	549
4.32 __gnu_cxx::char_traits<_CharT> Struct Template Reference	550
4.32.1 Detailed Description	551
4.33 __gnu_cxx::character<_Value, _Int, _St> Struct Template Reference	551
4.33.1 Detailed Description	552
4.34 __gnu_cxx::condition_base Struct Reference	552
4.34.1 Detailed Description	552
4.35 __gnu_cxx::debug_allocator<_Alloc> Class Template Reference	552
4.35.1 Detailed Description	553
4.36 __gnu_cxx::enc_filebuf<_CharT> Class Template Reference	553
4.36.1 Detailed Description	554
4.37 __gnu_cxx::encoding_char_traits<_CharT> Struct Template Reference	554
4.37.1 Detailed Description	555
4.38 __gnu_cxx::encoding_state Class Reference	555
4.38.1 Detailed Description	556

4.39	__gnu_cxx::forced_error Struct Reference	556
4.39.1	Detailed Description	556
4.40	__gnu_cxx::free_list Class Reference	557
4.40.1	Detailed Description	557
4.40.2	Member Function Documentation	557
4.41	__gnu_cxx::limit_condition Struct Reference	558
4.41.1	Detailed Description	559
4.42	__gnu_cxx::limit_condition::always_adjustor Struct Reference	559
4.42.1	Detailed Description	559
4.43	__gnu_cxx::limit_condition::limit_adjustor Struct Reference	559
4.43.1	Detailed Description	560
4.44	__gnu_cxx::limit_condition::never_adjustor Struct Reference	560
4.44.1	Detailed Description	560
4.45	__gnu_cxx::malloc_allocator< _Tp > Class Template Reference	560
4.45.1	Detailed Description	561
4.46	__gnu_cxx::new_allocator< _Tp > Class Template Reference	561
4.46.1	Detailed Description	562
4.47	__gnu_cxx::random_condition Struct Reference	563
4.47.1	Detailed Description	563
4.48	__gnu_cxx::random_condition::always_adjustor Struct Reference	563
4.48.1	Detailed Description	564
4.49	__gnu_cxx::random_condition::group_adjustor Struct Reference	564
4.49.1	Detailed Description	564
4.50	__gnu_cxx::random_condition::never_adjustor Struct Reference	564
4.50.1	Detailed Description	564
4.51	__gnu_cxx::recursive_init_error Class Reference	565
4.51.1	Detailed Description	565
4.52	__gnu_cxx::stdio_filebuf< _CharT, _Traits > Class Template Reference	565

4.52.1 Detailed Description	566
4.52.2 Constructor & Destructor Documentation	566
4.52.3 Member Function Documentation	567
4.53 <code>__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits></code> Class Template Reference	568
4.53.1 Detailed Description	569
4.53.2 Member Function Documentation	569
4.54 <code>__gnu_cxx::throw_allocator_base<_Tp, _Cond></code> Class Template Reference	569
4.54.1 Detailed Description	570
4.55 <code>__gnu_cxx::throw_allocator_limit<_Tp></code> Struct Template Reference	571
4.55.1 Detailed Description	572
4.56 <code>__gnu_cxx::throw_allocator_random<_Tp></code> Struct Template Reference	573
4.56.1 Detailed Description	574
4.57 <code>__gnu_cxx::throw_value_base<_Cond></code> Struct Template Reference	574
4.57.1 Detailed Description	575
4.58 <code>__gnu_cxx::throw_value_limit</code> Struct Reference	576
4.58.1 Detailed Description	577
4.59 <code>__gnu_cxx::throw_value_random</code> Struct Reference	577
4.59.1 Detailed Description	578
4.60 <code>__gnu_debug::_After_nth_from<_Iterator></code> Class Template Reference	578
4.60.1 Detailed Description	578
4.61 <code>__gnu_debug::_BeforeBeginHelper<_Sequence></code> Struct Template Reference	579
4.61.1 Detailed Description	579
4.62 <code>__gnu_debug::_Equal_to<_Type></code> Class Template Reference	579
4.62.1 Detailed Description	579
4.63 <code>__gnu_debug::_Not_equal_to<_Type></code> Class Template Reference	580
4.63.1 Detailed Description	580
4.64 <code>__gnu_debug::_Safe_iterator<_Iterator, _Sequence></code> Class Template Reference	580
4.64.1 Detailed Description	582

4.64.2	Constructor & Destructor Documentation	582
4.64.3	Member Function Documentation	583
4.64.4	Member Data Documentation	589
4.65	<code>__gnu_debug::_Safe_iterator_base</code> Class Reference	589
4.65.1	Detailed Description	590
4.65.2	Constructor & Destructor Documentation	590
4.65.3	Member Function Documentation	591
4.65.4	Member Data Documentation	593
4.66	<code>__gnu_debug::_Safe_local_iterator<_Iterator, _Sequence></code> Class Template Reference	594
4.66.1	Detailed Description	595
4.66.2	Constructor & Destructor Documentation	596
4.66.3	Member Function Documentation	597
4.66.4	Member Data Documentation	602
4.67	<code>__gnu_debug::_Safe_local_iterator_base</code> Class Reference	603
4.67.1	Detailed Description	604
4.67.2	Constructor & Destructor Documentation	604
4.67.3	Member Function Documentation	605
4.67.4	Member Data Documentation	607
4.68	<code>__gnu_debug::_Safe_sequence<_Sequence></code> Class Template Reference	607
4.68.1	Detailed Description	608
4.68.2	Member Function Documentation	608
4.68.3	Member Data Documentation	610
4.69	<code>__gnu_debug::_Safe_sequence_base</code> Class Reference	611
4.69.1	Detailed Description	612
4.69.2	Constructor & Destructor Documentation	612
4.69.3	Member Function Documentation	612
4.69.4	Member Data Documentation	613
4.70	<code>__gnu_debug::_Safe_unordered_container<_Container></code> Class Template Reference	614

4.70.1	Detailed Description	615
4.70.2	Member Function Documentation	615
4.70.3	Member Data Documentation	617
4.71	<code>__gnu_debug::__Safe_unordered_container_base</code> Class Reference	618
4.71.1	Detailed Description	619
4.71.2	Constructor & Destructor Documentation	620
4.71.3	Member Function Documentation	620
4.71.4	Member Data Documentation	622
4.72	<code>__gnu_parallel::__accumulate_binop_reduct<_BinOp></code> Struct Template Reference	622
4.72.1	Detailed Description	623
4.73	<code>__gnu_parallel::__accumulate_selector<_It></code> Struct Template Reference	623
4.73.1	Detailed Description	623
4.73.2	Member Function Documentation	624
4.73.3	Member Data Documentation	624
4.74	<code>__gnu_parallel::__adjacent_difference_selector<_It></code> Struct Template Reference	624
4.74.1	Detailed Description	625
4.74.2	Member Data Documentation	625
4.75	<code>__gnu_parallel::__adjacent_find_selector</code> Struct Reference	625
4.75.1	Detailed Description	626
4.75.2	Member Function Documentation	626
4.76	<code>__gnu_parallel::__binder1st<_Operation, _FirstArgumentType, _SecondArgumentType, _ResultType></code> Class Template Reference	627
4.76.1	Detailed Description	628
4.76.2	Member Typedef Documentation	628
4.77	<code>__gnu_parallel::__binder2nd<_Operation, _FirstArgumentType, _SecondArgumentType, _ResultType></code> Class Template Reference	628
4.77.1	Detailed Description	629
4.77.2	Member Typedef Documentation	629
4.78	<code>__gnu_parallel::__count_if_selector<_It, _Diff></code> Struct Template Reference	630

4.78.1 Detailed Description	630
4.78.2 Member Function Documentation	630
4.78.3 Member Data Documentation	631
4.79 <code>__gnu_parallel::__count_selector<_It, _Diff></code> Struct Template Reference	631
4.79.1 Detailed Description	632
4.79.2 Member Function Documentation	632
4.79.3 Member Data Documentation	632
4.80 <code>__gnu_parallel::__fill_selector<_It></code> Struct Template Reference	633
4.80.1 Detailed Description	633
4.80.2 Member Function Documentation	633
4.80.3 Member Data Documentation	634
4.81 <code>__gnu_parallel::__find_first_of_selector<_FIterator></code> Struct Template Reference	634
4.81.1 Detailed Description	635
4.81.2 Member Function Documentation	635
4.82 <code>__gnu_parallel::__find_if_selector</code> Struct Reference	636
4.82.1 Detailed Description	636
4.82.2 Member Function Documentation	636
4.83 <code>__gnu_parallel::__for_each_selector<_It></code> Struct Template Reference	637
4.83.1 Detailed Description	638
4.83.2 Member Function Documentation	638
4.83.3 Member Data Documentation	638
4.84 <code>__gnu_parallel::__generate_selector<_It></code> Struct Template Reference	639
4.84.1 Detailed Description	639
4.84.2 Member Function Documentation	639
4.84.3 Member Data Documentation	640
4.85 <code>__gnu_parallel::__generic_find_selector</code> Struct Reference	640
4.85.1 Detailed Description	641
4.86 <code>__gnu_parallel::__generic_for_each_selector<_It></code> Struct Template Reference	642

4.86.1 Detailed Description	643
4.86.2 Member Data Documentation	643
4.87 <code>__gnu_parallel::__identity_selector<_It></code> Struct Template Reference	643
4.87.1 Detailed Description	644
4.87.2 Member Function Documentation	644
4.87.3 Member Data Documentation	644
4.88 <code>__gnu_parallel::__inner_product_selector<_It, _It2, _Tp></code> Struct Template Reference	645
4.88.1 Detailed Description	645
4.88.2 Constructor & Destructor Documentation	645
4.88.3 Member Function Documentation	646
4.88.4 Member Data Documentation	646
4.89 <code>__gnu_parallel::__max_element_reduct<_Compare, _It></code> Struct Template Reference	647
4.89.1 Detailed Description	647
4.90 <code>__gnu_parallel::__min_element_reduct<_Compare, _It></code> Struct Template Reference	647
4.90.1 Detailed Description	648
4.91 <code>__gnu_parallel::__mismatch_selector</code> Struct Reference	648
4.91.1 Detailed Description	649
4.91.2 Member Function Documentation	649
4.92 <code>__gnu_parallel::__multiway_merge_3_variant_sentinel_switch<__sentinels, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare></code> Struct Template Reference	649
4.92.1 Detailed Description	650
4.93 <code>__gnu_parallel::__multiway_merge_3_variant_sentinel_switch<true, _RAIterIterator, _RAIter3, _↵ DifferenceTp, _Compare></code> Struct Template Reference	650
4.93.1 Detailed Description	650
4.94 <code>__gnu_parallel::__multiway_merge_4_variant_sentinel_switch<__sentinels, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare></code> Struct Template Reference	650
4.94.1 Detailed Description	651
4.95 <code>__gnu_parallel::__multiway_merge_4_variant_sentinel_switch<true, _RAIterIterator, _RAIter3, _↵ DifferenceTp, _Compare></code> Struct Template Reference	651
4.95.1 Detailed Description	651

4.96	<code>__gnu_parallel::__multiway_merge_k_variant_sentinel_switch< __sentinels, __stable, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare ></code> Struct Template Reference	651
4.96.1	Detailed Description	652
4.97	<code>__gnu_parallel::__multiway_merge_k_variant_sentinel_switch< false, __stable, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare ></code> Struct Template Reference	652
4.97.1	Detailed Description	652
4.98	<code>__gnu_parallel::__replace_if_selector< _It, _Op, _Tp ></code> Struct Template Reference	653
4.98.1	Detailed Description	653
4.98.2	Constructor & Destructor Documentation	653
4.98.3	Member Function Documentation	654
4.98.4	Member Data Documentation	654
4.99	<code>__gnu_parallel::__replace_selector< _It, _Tp ></code> Struct Template Reference	655
4.99.1	Detailed Description	655
4.99.2	Constructor & Destructor Documentation	655
4.99.3	Member Function Documentation	656
4.99.4	Member Data Documentation	656
4.100	<code>__gnu_parallel::__transform1_selector< _It ></code> Struct Template Reference	657
4.100.1	Detailed Description	657
4.100.2	Member Function Documentation	657
4.100.3	Member Data Documentation	658
4.101	<code>__gnu_parallel::__transform2_selector< _It ></code> Struct Template Reference	658
4.101.1	Detailed Description	659
4.101.2	Member Function Documentation	659
4.101.3	Member Data Documentation	659
4.102	<code>__gnu_parallel::__unary_negate< _Predicate, argument_type ></code> Class Template Reference	660
4.102.1	Detailed Description	660
4.102.2	Member Typedef Documentation	661
4.103	<code>__gnu_parallel::_DRandomShufflingGlobalData< _RAIter ></code> Struct Template Reference	661
4.103.1	Detailed Description	661

4.103.2 Constructor & Destructor Documentation	662
4.103.3 Member Data Documentation	662
4.104 __gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator > Struct Template Reference	663
4.104.1 Detailed Description	663
4.104.2 Member Data Documentation	664
4.105 __gnu_parallel::_DummyReduct Struct Reference	665
4.105.1 Detailed Description	665
4.106 __gnu_parallel::_EqualFromLess< _T1, _T2, _Compare > Class Template Reference	665
4.106.1 Detailed Description	666
4.106.2 Member Typedef Documentation	666
4.107 __gnu_parallel::_EqualTo< _T1, _T2 > Struct Template Reference	666
4.107.1 Detailed Description	667
4.107.2 Member Typedef Documentation	667
4.108 __gnu_parallel::_GuardedIterator< _RAIter, _Compare > Class Template Reference	668
4.108.1 Detailed Description	668
4.108.2 Constructor & Destructor Documentation	668
4.108.3 Member Function Documentation	669
4.108.4 Friends And Related Function Documentation	669
4.109 __gnu_parallel::_IteratorPair< _Iterator1, _Iterator2, _IteratorCategory > Class Template Reference	670
4.109.1 Detailed Description	671
4.109.2 Member Typedef Documentation	671
4.109.3 Member Data Documentation	672
4.110 __gnu_parallel::_IteratorTriple< _Iterator1, _Iterator2, _Iterator3, _IteratorCategory > Class Template Reference	672
4.110.1 Detailed Description	673
4.111 __gnu_parallel::_Job< _DifferenceTp > Struct Template Reference	673
4.111.1 Detailed Description	673
4.111.2 Member Data Documentation	673

4.112	__gnu_parallel::_Less<_T1, _T2 > Struct Template Reference	674
4.112.1	Detailed Description	675
4.112.2	Member Typedef Documentation	675
4.113	__gnu_parallel::_Lexicographic<_T1, _T2, _Compare > Class Template Reference	676
4.113.1	Detailed Description	676
4.113.2	Member Typedef Documentation	676
4.114	__gnu_parallel::_LexicographicReverse<_T1, _T2, _Compare > Class Template Reference	677
4.114.1	Detailed Description	678
4.114.2	Member Typedef Documentation	678
4.115	__gnu_parallel::_LoserTree<__stable, _Tp, _Compare > Class Template Reference	678
4.115.1	Detailed Description	679
4.115.2	Member Function Documentation	679
4.115.3	Member Data Documentation	680
4.116	__gnu_parallel::_LoserTree<false, _Tp, _Compare > Class Template Reference	681
4.116.1	Detailed Description	682
4.116.2	Member Function Documentation	682
4.116.3	Member Data Documentation	683
4.117	__gnu_parallel::_LoserTreeBase<_Tp, _Compare > Class Template Reference	684
4.117.1	Detailed Description	685
4.117.2	Constructor & Destructor Documentation	685
4.117.3	Member Function Documentation	686
4.117.4	Member Data Documentation	687
4.118	__gnu_parallel::_LoserTreeBase<_Tp, _Compare >::_Loser Struct Reference	688
4.118.1	Detailed Description	688
4.118.2	Member Data Documentation	688
4.119	__gnu_parallel::_LoserTreePointer<__stable, _Tp, _Compare > Class Template Reference	689
4.119.1	Detailed Description	690
4.120	__gnu_parallel::_LoserTreePointer<false, _Tp, _Compare > Class Template Reference	690

4.120.1 Detailed Description	691
4.121 <code>__gnu_parallel::_LoserTreePointerBase< _Tp, _Compare ></code> Class Template Reference	691
4.121.1 Detailed Description	692
4.122 <code>__gnu_parallel::_LoserTreePointerBase< _Tp, _Compare >::_Loser</code> Struct Reference	692
4.122.1 Detailed Description	692
4.123 <code>__gnu_parallel::_LoserTreePointerUnguarded< __stable, _Tp, _Compare ></code> Class Template Reference	693
4.123.1 Detailed Description	693
4.124 <code>__gnu_parallel::_LoserTreePointerUnguarded< false, _Tp, _Compare ></code> Class Template Reference	694
4.124.1 Detailed Description	694
4.125 <code>__gnu_parallel::_LoserTreePointerUnguardedBase< _Tp, _Compare ></code> Class Template Reference	695
4.125.1 Detailed Description	695
4.126 <code>__gnu_parallel::_LoserTreeTraits< _Tp ></code> Struct Template Reference	696
4.126.1 Detailed Description	696
4.126.2 Member Data Documentation	696
4.127 <code>__gnu_parallel::_LoserTreeUnguarded< __stable, _Tp, _Compare ></code> Class Template Reference	697
4.127.1 Detailed Description	697
4.128 <code>__gnu_parallel::_LoserTreeUnguarded< false, _Tp, _Compare ></code> Class Template Reference	698
4.128.1 Detailed Description	698
4.129 <code>__gnu_parallel::_LoserTreeUnguardedBase< _Tp, _Compare ></code> Class Template Reference	699
4.129.1 Detailed Description	699
4.130 <code>__gnu_parallel::_Multiplies< _Tp1, _Tp2, _Result ></code> Struct Template Reference	700
4.130.1 Detailed Description	700
4.130.2 Member Typedef Documentation	700
4.131 <code>__gnu_parallel::_Nothing</code> Struct Reference	701
4.131.1 Detailed Description	701
4.131.2 Member Function Documentation	701
4.132 <code>__gnu_parallel::_Piece< _DifferenceTp ></code> Struct Template Reference	702
4.132.1 Detailed Description	702

4.132.2 Member Data Documentation	702
4.133 <code>__gnu_parallel::Plus<_Tp1, _Tp2, _Result></code> Struct Template Reference	703
4.133.1 Detailed Description	703
4.133.2 Member Typedef Documentation	703
4.134 <code>__gnu_parallel::PMWMSSortingData<_RAIter></code> Struct Template Reference	704
4.134.1 Detailed Description	704
4.134.2 Member Data Documentation	705
4.135 <code>__gnu_parallel::PseudoSequence<_Tp, _DifferenceTp></code> Class Template Reference	706
4.135.1 Detailed Description	706
4.135.2 Constructor & Destructor Documentation	706
4.135.3 Member Function Documentation	707
4.136 <code>__gnu_parallel::PseudoSequenceIterator<_Tp, _DifferenceTp></code> Class Template Reference	707
4.136.1 Detailed Description	708
4.137 <code>__gnu_parallel::QSBThreadLocal<_RAIter></code> Struct Template Reference	709
4.137.1 Detailed Description	709
4.137.2 Member Typedef Documentation	710
4.137.3 Constructor & Destructor Documentation	710
4.137.4 Member Data Documentation	711
4.138 <code>__gnu_parallel::RandomNumber</code> Class Reference	712
4.138.1 Detailed Description	712
4.138.2 Constructor & Destructor Documentation	712
4.138.3 Member Function Documentation	712
4.139 <code>__gnu_parallel::RestrictedBoundedConcurrentQueue<_Tp></code> Class Template Reference	713
4.139.1 Detailed Description	713
4.139.2 Constructor & Destructor Documentation	714
4.139.3 Member Function Documentation	714
4.140 <code>__gnu_parallel::SamplingSorter<__stable, _RAIter, _StrictWeakOrdering></code> Struct Template Reference	715
4.140.1 Detailed Description	715

4.141 <code>__gnu_parallel::_SamplingSorter< false, _RAIter, _StrictWeakOrdering ></code> Struct Template Reference . . .	715
4.141.1 Detailed Description	715
4.142 <code>__gnu_parallel::_Settings</code> Struct Reference	716
4.142.1 Detailed Description	717
4.142.2 Member Function Documentation	717
4.142.3 Member Data Documentation	717
4.143 <code>__gnu_parallel::_SplitConsistently< __exact, _RAIter, _Compare, _SortingPlacesIter ></code> Struct Template Reference	723
4.143.1 Detailed Description	723
4.144 <code>__gnu_parallel::_SplitConsistently< false, _RAIter, _Compare, _SortingPlacesIter ></code> Struct Template Reference	724
4.144.1 Detailed Description	724
4.145 <code>__gnu_parallel::_SplitConsistently< true, _RAIter, _Compare, _SortingPlacesIter ></code> Struct Template Reference	724
4.145.1 Detailed Description	724
4.146 <code>__gnu_parallel::balanced_quicksort_tag</code> Struct Reference	725
4.146.1 Detailed Description	725
4.146.2 Member Function Documentation	725
4.147 <code>__gnu_parallel::balanced_tag</code> Struct Reference	726
4.147.1 Detailed Description	726
4.147.2 Member Function Documentation	726
4.148 <code>__gnu_parallel::constant_size_blocks_tag</code> Struct Reference	727
4.148.1 Detailed Description	727
4.149 <code>__gnu_parallel::default_parallel_tag</code> Struct Reference	728
4.149.1 Detailed Description	728
4.149.2 Member Function Documentation	728
4.150 <code>__gnu_parallel::equal_split_tag</code> Struct Reference	729
4.150.1 Detailed Description	729
4.151 <code>__gnu_parallel::exact_tag</code> Struct Reference	730

4.151.1 Detailed Description	730
4.151.2 Member Function Documentation	730
4.152 __gnu_parallel::find_tag Struct Reference	731
4.152.1 Detailed Description	731
4.153 __gnu_parallel::growing_blocks_tag Struct Reference	732
4.153.1 Detailed Description	732
4.154 __gnu_parallel::multiway_mergesort_exact_tag Struct Reference	732
4.154.1 Detailed Description	733
4.154.2 Member Function Documentation	733
4.155 __gnu_parallel::multiway_mergesort_sampling_tag Struct Reference	734
4.155.1 Detailed Description	734
4.155.2 Member Function Documentation	734
4.156 __gnu_parallel::multiway_mergesort_tag Struct Reference	735
4.156.1 Detailed Description	735
4.156.2 Member Function Documentation	736
4.157 __gnu_parallel::omp_loop_static_tag Struct Reference	736
4.157.1 Detailed Description	737
4.157.2 Member Function Documentation	737
4.158 __gnu_parallel::omp_loop_tag Struct Reference	738
4.158.1 Detailed Description	738
4.158.2 Member Function Documentation	738
4.159 __gnu_parallel::parallel_tag Struct Reference	740
4.159.1 Detailed Description	741
4.159.2 Constructor & Destructor Documentation	741
4.159.3 Member Function Documentation	741
4.160 __gnu_parallel::quicksort_tag Struct Reference	742
4.160.1 Detailed Description	742
4.160.2 Member Function Documentation	743

4.161 <code>__gnu_parallel::sampling_tag</code> Struct Reference	743
4.161.1 Detailed Description	744
4.161.2 Member Function Documentation	744
4.162 <code>__gnu_parallel::sequential_tag</code> Struct Reference	744
4.162.1 Detailed Description	745
4.163 <code>__gnu_parallel::unbalanced_tag</code> Struct Reference	745
4.163.1 Detailed Description	745
4.163.2 Member Function Documentation	745
4.164 <code>__gnu_pbds::associative_tag</code> Struct Reference	746
4.164.1 Detailed Description	746
4.165 <code>__gnu_pbds::basic_branch< Key, Mapped, Tag, Node_Update, Policy_Tl, _Alloc ></code> Class Template Reference	746
4.165.1 Detailed Description	747
4.166 <code>__gnu_pbds::basic_branch_tag</code> Struct Reference	748
4.166.1 Detailed Description	748
4.167 <code>__gnu_pbds::basic_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Resize_Policy, Store_Hash, Tag, Policy_Tl, _Alloc ></code> Class Template Reference	748
4.167.1 Detailed Description	749
4.168 <code>__gnu_pbds::basic_hash_tag</code> Struct Reference	750
4.168.1 Detailed Description	750
4.169 <code>__gnu_pbds::basic_invalidation_guarantee</code> Struct Reference	751
4.169.1 Detailed Description	751
4.170 <code>__gnu_pbds::binary_heap_tag</code> Struct Reference	752
4.170.1 Detailed Description	752
4.171 <code>__gnu_pbds::binomial_heap_tag</code> Struct Reference	753
4.171.1 Detailed Description	753
4.172 <code>__gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type ></code> Class Template Reference	753
4.172.1 Detailed Description	754

4.172.2 Member Enumeration Documentation	754
4.172.3 Constructor & Destructor Documentation	754
4.172.4 Member Function Documentation	755
4.173 <code>__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store↵ _Hash, _Alloc ></code> Class Template Reference	758
4.173.1 Detailed Description	759
4.173.2 Constructor & Destructor Documentation	759
4.174 <code>__gnu_pbds::cc_hash_tag</code> Struct Reference	762
4.174.1 Detailed Description	762
4.175 <code>__gnu_pbds::container_error</code> Struct Reference	763
4.175.1 Detailed Description	763
4.176 <code>__gnu_pbds::container_tag</code> Struct Reference	763
4.176.1 Detailed Description	764
4.177 <code>__gnu_pbds::container_traits< Cntnr ></code> Struct Template Reference	764
4.177.1 Detailed Description	764
4.177.2 Member Enumeration Documentation	765
4.178 <code>__gnu_pbds::container_traits_base< _Tag ></code> Struct Template Reference	765
4.178.1 Detailed Description	765
4.179 <code>__gnu_pbds::container_traits_base< binary_heap_tag ></code> Struct Template Reference	765
4.179.1 Detailed Description	766
4.180 <code>__gnu_pbds::container_traits_base< binomial_heap_tag ></code> Struct Template Reference	766
4.180.1 Detailed Description	766
4.181 <code>__gnu_pbds::container_traits_base< cc_hash_tag ></code> Struct Template Reference	766
4.181.1 Detailed Description	767
4.182 <code>__gnu_pbds::container_traits_base< gp_hash_tag ></code> Struct Template Reference	767
4.182.1 Detailed Description	767
4.183 <code>__gnu_pbds::container_traits_base< list_update_tag ></code> Struct Template Reference	767
4.183.1 Detailed Description	768

4.184	__gnu_pbds::container_traits_base< ov_tree_tag > Struct Template Reference	768
4.184.1	Detailed Description	768
4.185	__gnu_pbds::container_traits_base< pairing_heap_tag > Struct Template Reference	768
4.185.1	Detailed Description	769
4.186	__gnu_pbds::container_traits_base< pat_trie_tag > Struct Template Reference	769
4.186.1	Detailed Description	769
4.187	__gnu_pbds::container_traits_base< rb_tree_tag > Struct Template Reference	769
4.187.1	Detailed Description	770
4.188	__gnu_pbds::container_traits_base< rc_binomial_heap_tag > Struct Template Reference	770
4.188.1	Detailed Description	770
4.189	__gnu_pbds::container_traits_base< splay_tree_tag > Struct Template Reference	770
4.189.1	Detailed Description	771
4.190	__gnu_pbds::container_traits_base< thin_heap_tag > Struct Template Reference	771
4.190.1	Detailed Description	771
4.191	__gnu_pbds::detail::bin_search_tree_const_it< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > Class Template Reference	771
4.191.1	Detailed Description	773
4.192	__gnu_pbds::detail::bin_search_tree_const_node_it< Node, Const_Iterator, Iterator, _Alloc > Class Template Reference	773
4.192.1	Detailed Description	774
4.192.2	Member Typedef Documentation	774
4.192.3	Member Function Documentation	775
4.193	__gnu_pbds::detail::bin_search_tree_it< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > Class Template Reference	776
4.193.1	Detailed Description	778
4.194	__gnu_pbds::detail::bin_search_tree_node_it< Node, Const_Iterator, Iterator, _Alloc > Class Template Reference	778
4.194.1	Detailed Description	779
4.194.2	Member Typedef Documentation	779
4.194.3	Member Function Documentation	780

4.195 __gnu_pbds::detail::bin_search_tree_traits< Key, Mapped, Cmp_Fn, Node_Update, Node, _Alloc > Struct Template Reference	.781
4.195.1 Detailed Description	.782
4.195.2 Member Typedef Documentation	.782
4.196 __gnu_pbds::detail::bin_search_tree_traits< Key, null_type, Cmp_Fn, Node_Update, Node, _Alloc > Struct Template Reference	.783
4.196.1 Detailed Description	.783
4.196.2 Member Typedef Documentation	.783
4.197 __gnu_pbds::detail::binary_heap< Value_Type, Cmp_Fn, _Alloc > Class Template Reference	.784
4.197.1 Detailed Description	.786
4.198 __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc > Class Template Reference	.786
4.198.1 Detailed Description	.787
4.198.2 Member Typedef Documentation	.787
4.198.3 Constructor & Destructor Documentation	.788
4.198.4 Member Function Documentation	.789
4.199 __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc > Class Template Reference	.790
4.199.1 Detailed Description	.791
4.199.2 Member Typedef Documentation	.791
4.199.3 Constructor & Destructor Documentation	.792
4.199.4 Member Function Documentation	.793
4.200 __gnu_pbds::detail::binomial_heap< Value_Type, Cmp_Fn, _Alloc > Class Template Reference	.793
4.200.1 Detailed Description	.795
4.201 __gnu_pbds::detail::binomial_heap_base< Value_Type, Cmp_Fn, _Alloc > Class Template Reference	.796
4.201.1 Detailed Description	.798
4.202 __gnu_pbds::detail::branch_policy< Node_Cltr, Node_Itr, _Alloc > Struct Template Reference	.798
4.202.1 Detailed Description	.799
4.203 __gnu_pbds::detail::branch_policy< Node_Cltr, Node_Cltr, _Alloc > Struct Template Reference	.799
4.203.1 Detailed Description	.800

4.204 __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy > Class Template Reference	800
4.204.1 Detailed Description	802
4.204.2 Member Enumeration Documentation	803
4.204.3 Member Function Documentation	803
4.205 __gnu_pbds::detail::cond_dealtor< Entry, _Alloc > Class Template Reference	804
4.205.1 Detailed Description	805
4.206 __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, Tag, Policy_TI > Struct Template Reference	805
4.206.1 Detailed Description	805
4.207 __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, binary_heap_tag, null_type > Struct Template Reference	806
4.207.1 Detailed Description	806
4.207.2 Member Typedef Documentation	806
4.208 __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, binomial_heap_tag, null_type > Struct Template Reference	806
4.208.1 Detailed Description	806
4.208.2 Member Typedef Documentation	807
4.209 __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, pairing_heap_tag, null_type > Struct Template Reference	807
4.209.1 Detailed Description	807
4.209.2 Member Typedef Documentation	807
4.210 __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, rc_binomial_heap_tag, null_type > Struct Template Reference	808
4.210.1 Detailed Description	808
4.210.2 Member Typedef Documentation	808
4.211 __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, thin_heap_tag, null_type > Struct Template Reference	808
4.211.1 Detailed Description	808
4.211.2 Member Typedef Documentation	809
4.212 __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, cc_hash_tag, Policy_TI > Struct Template Reference	809

4.212.1 Detailed Description	809
4.212.2 Member Typedef Documentation	809
4.213__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, gp_hash_tag, Policy_TI > Struct Template Reference	810
4.213.1 Detailed Description	810
4.213.2 Member Typedef Documentation	810
4.214__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, list_update_tag, Policy_TI > Struct Template Reference	810
4.214.1 Detailed Description	810
4.214.2 Member Typedef Documentation	811
4.215__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, ov_tree_tag, Policy_TI > Struct Template Reference	811
4.215.1 Detailed Description	811
4.215.2 Member Typedef Documentation	811
4.216__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, pat_trie_tag, Policy_TI > Struct Template Reference	812
4.216.1 Detailed Description	812
4.217__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, rb_tree_tag, Policy_TI > Struct Template Reference	812
4.217.1 Detailed Description	812
4.217.2 Member Typedef Documentation	812
4.218__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, splay_tree_tag, Policy_TI > Struct Template Reference	813
4.218.1 Detailed Description	813
4.218.2 Member Typedef Documentation	813
4.219__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, cc_hash_tag, Policy_TI > Struct Template Reference	813
4.219.1 Detailed Description	813
4.219.2 Member Typedef Documentation	814
4.220__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, gp_hash_tag, Policy_TI > Struct Template Reference	814
4.220.1 Detailed Description	814

4.220.2 Member Typedef Documentation	814
4.221__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, list_update_tag, Policy_TI > Struct Template Reference	815
4.221.1 Detailed Description	815
4.221.2 Member Typedef Documentation	815
4.222__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, ov_tree_tag, Policy_TI > Struct Template Reference	815
4.222.1 Detailed Description	815
4.222.2 Member Typedef Documentation	816
4.223__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, pat_trie_tag, Policy_TI > Struct Template Reference	816
4.223.1 Detailed Description	816
4.223.2 Member Typedef Documentation	816
4.224__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, rb_tree_tag, Policy_TI > Struct Template Reference	817
4.224.1 Detailed Description	817
4.225__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, splay_tree_tag, Policy_TI > Struct Template Reference	817
4.225.1 Detailed Description	817
4.225.2 Member Typedef Documentation	817
4.226__gnu_pbds::detail::default_comb_hash_fn Struct Reference	818
4.226.1 Detailed Description	818
4.226.2 Member Typedef Documentation	818
4.227__gnu_pbds::detail::default_eq_fn< Key > Struct Template Reference	818
4.227.1 Detailed Description	818
4.227.2 Member Typedef Documentation	819
4.228__gnu_pbds::detail::default_hash_fn< Key > Struct Template Reference	819
4.228.1 Detailed Description	819
4.228.2 Member Typedef Documentation	819
4.229__gnu_pbds::detail::default_probe_fn< Comb_Probe_Fn > Struct Template Reference	819

4.229.1 Detailed Description	820
4.229.2 Member Typedef Documentation	820
4.230 <code>__gnu_pbds::detail::default_resize_policy< Comb_Hash_Fn ></code> Struct Template Reference	820
4.230.1 Detailed Description	820
4.230.2 Member Typedef Documentation	820
4.231 <code>__gnu_pbds::detail::default_trie_access_traits< Key ></code> Struct Template Reference	821
4.231.1 Detailed Description	821
4.232 <code>__gnu_pbds::detail::default_trie_access_traits< std::basic_string< Char, Char_Traits, std::allocator< char > > ></code> Struct Template Reference	821
4.232.1 Detailed Description	821
4.232.2 Member Typedef Documentation	821
4.233 <code>__gnu_pbds::detail::default_update_policy</code> Struct Reference	822
4.233.1 Detailed Description	822
4.233.2 Member Typedef Documentation	822
4.234 <code>__gnu_pbds::detail::dumnode_const_iterator< Key, Data, _Alloc ></code> Struct Template Reference	822
4.234.1 Detailed Description	822
4.235 <code>__gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, No_Throw ></code> Struct Template Reference	823
4.235.1 Detailed Description	823
4.236 <code>__gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, false ></code> Struct Template Reference	823
4.236.1 Detailed Description	823
4.237 <code>__gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, false >::type</code> Struct Reference	823
4.237.1 Detailed Description	824
4.238 <code>__gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, true ></code> Struct Template Reference	824
4.238.1 Detailed Description	824
4.238.2 Member Typedef Documentation	824
4.239 <code>__gnu_pbds::detail::entry_pred< _VTp, Pred, _Alloc, No_Throw ></code> Struct Template Reference	825
4.239.1 Detailed Description	825
4.240 <code>__gnu_pbds::detail::entry_pred< _VTp, Pred, _Alloc, false ></code> Struct Template Reference	825

4.240.1 Detailed Description	825
4.241 <code>__gnu_pbds::detail::entry_pred<_VTp, Pred, _Alloc, true></code> Struct Template Reference	825
4.241.1 Detailed Description	826
4.242 <code>__gnu_pbds::detail::eq_by_less<Key, Cmp_Fn></code> Struct Template Reference	826
4.242.1 Detailed Description	826
4.243 <code>__gnu_pbds::detail::gp_ht_map<Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy></code> Class Template Reference	826
4.243.1 Detailed Description	828
4.243.2 Member Enumeration Documentation	829
4.243.3 Member Function Documentation	829
4.244 <code>__gnu_pbds::detail::hash_eq_fn<Key, Eq_Fn, _Alloc, Store_Hash></code> Struct Template Reference	831
4.244.1 Detailed Description	832
4.245 <code>__gnu_pbds::detail::hash_eq_fn<Key, Eq_Fn, _Alloc, false></code> Struct Template Reference	832
4.245.1 Detailed Description	832
4.246 <code>__gnu_pbds::detail::hash_eq_fn<Key, Eq_Fn, _Alloc, true></code> Struct Template Reference	833
4.246.1 Detailed Description	833
4.247 <code>__gnu_pbds::detail::hash_load_check_resize_trigger_size_base<Size_Type, Hold_Size></code> Class Template Reference	833
4.247.1 Detailed Description	833
4.248 <code>__gnu_pbds::detail::hash_load_check_resize_trigger_size_base<Size_Type, true></code> Class Template Reference	834
4.248.1 Detailed Description	834
4.249 <code>__gnu_pbds::detail::left_child_next_sibling_heap<Value_Type, Cmp_Fn, Node_Metadata, _Alloc></code> Class Template Reference	834
4.249.1 Detailed Description	836
4.250 <code>__gnu_pbds::detail::left_child_next_sibling_heap_const_iterator<Node, _Alloc></code> Class Template Reference	836
4.250.1 Detailed Description	837
4.250.2 Member Typedef Documentation	837
4.250.3 Constructor & Destructor Documentation	838

4.250.4 Member Function Documentation	838
4.251 <code>__gnu_pbds::detail::left_child_next_sibling_heap_node_< _Value, _Metadata, _Alloc > Struct Template Reference</code>	839
4.251.1 Detailed Description	840
4.252 <code>__gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > Class Template Reference</code>	840
4.252.1 Detailed Description	841
4.252.2 Member Typedef Documentation	841
4.252.3 Constructor & Destructor Documentation	843
4.252.4 Member Function Documentation	843
4.253 <code>__gnu_pbds::detail::lu_counter_metadata< Size_Type > Class Template Reference</code>	844
4.253.1 Detailed Description	844
4.254 <code>__gnu_pbds::detail::lu_counter_policy_base< Size_Type > Class Template Reference</code>	844
4.254.1 Detailed Description	845
4.255 <code>__gnu_pbds::detail::lu_map< Key, Mapped, Eq_Fn, _Alloc, Update_Policy > Class Template Reference</code>	845
4.255.1 Detailed Description	847
4.256 <code>__gnu_pbds::detail::mask_based_range_hashing< Size_Type > Class Template Reference</code>	847
4.256.1 Detailed Description	848
4.257 <code>__gnu_pbds::detail::mod_based_range_hashing< Size_Type > Class Template Reference</code>	848
4.257.1 Detailed Description	849
4.258 <code>__gnu_pbds::detail::no_throw_copies< Key, Mapped > Struct Template Reference</code>	849
4.258.1 Detailed Description	849
4.259 <code>__gnu_pbds::detail::no_throw_copies< Key, null_type > Struct Template Reference</code>	850
4.259.1 Detailed Description	850
4.260 <code>__gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > Class Template Reference</code>	850
4.260.1 Detailed Description	852
4.260.2 Member Function Documentation	852
4.261 <code>__gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::cond_dtor< Size_Type > Class Template Reference</code>	853

4.261.1 Detailed Description	854
4.262 <code>__gnu_pbds::detail::ov_tree_node_const_it_< Value_Type, Metadata_Type, _Alloc > Class Template Reference</code>	854
4.262.1 Detailed Description	855
4.262.2 Member Function Documentation	855
4.263 <code>__gnu_pbds::detail::ov_tree_node_it_< Value_Type, Metadata_Type, _Alloc > Class Template Reference</code>	856
4.263.1 Detailed Description	857
4.263.2 Member Function Documentation	857
4.264 <code>__gnu_pbds::detail::pairing_heap< Value_Type, Cmp_Fn, _Alloc > Class Template Reference</code>	858
4.264.1 Detailed Description	860
4.265 <code>__gnu_pbds::detail::pat_trie_base Struct Reference</code>	860
4.265.1 Detailed Description	861
4.265.2 Member Enumeration Documentation	861
4.266 <code>__gnu_pbds::detail::pat_trie_base::_Clter< Node, Leaf, Head, Inode, Is_Forward_Iterator > Class Template Reference</code>	861
4.266.1 Detailed Description	863
4.267 <code>__gnu_pbds::detail::pat_trie_base::_Head< _ATraits, Metadata > Struct Template Reference</code>	863
4.267.1 Detailed Description	864
4.268 <code>__gnu_pbds::detail::pat_trie_base::_Inode< _ATraits, Metadata > Struct Template Reference</code>	864
4.268.1 Detailed Description	866
4.269 <code>__gnu_pbds::detail::pat_trie_base::_Inode< _ATraits, Metadata >::const_iterator Struct Reference</code>	866
4.269.1 Detailed Description	867
4.270 <code>__gnu_pbds::detail::pat_trie_base::_Inode< _ATraits, Metadata >::iterator Struct Reference</code>	867
4.270.1 Detailed Description	868
4.271 <code>__gnu_pbds::detail::pat_trie_base::_Iter< Node, Leaf, Head, Inode, Is_Forward_Iterator > Class Template Reference</code>	869
4.271.1 Detailed Description	870
4.272 <code>__gnu_pbds::detail::pat_trie_base::_Leaf< _ATraits, Metadata > Struct Template Reference</code>	871
4.272.1 Detailed Description	872

4.273__gnu_pbds::detail::pat_trie_base::_Metadata< Metadata, _Alloc > Struct Template Reference	872
4.273.1 Detailed Description	872
4.274__gnu_pbds::detail::pat_trie_base::_Metadata< null_type, _Alloc > Struct Template Reference	873
4.274.1 Detailed Description	873
4.275__gnu_pbds::detail::pat_trie_base::_Node_base< _ATraits, Metadata > Struct Template Reference	873
4.275.1 Detailed Description	874
4.276__gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc > Class Template Reference	874
4.276.1 Detailed Description	875
4.276.2 Member Typedef Documentation	876
4.276.3 Member Function Documentation	876
4.277__gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc > Class Template Reference	877
4.277.1 Detailed Description	878
4.277.2 Member Typedef Documentation	879
4.277.3 Member Function Documentation	879
4.278__gnu_pbds::detail::pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc > Class Template Refer- ence	880
4.278.1 Detailed Description	882
4.278.2 Member Enumeration Documentation	883
4.278.3 Member Function Documentation	883
4.279__gnu_pbds::detail::probe_fn_base< _Alloc > Class Template Reference	884
4.279.1 Detailed Description	884
4.280__gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, Store_Hash > Class Template Reference	884
4.280.1 Detailed Description	884
4.281__gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, false > Class Template Reference	885
4.281.1 Detailed Description	885
4.282__gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, true > Class Template Reference	885

4.282.1 Detailed Description	. 886
4.283 <code>_gnu_pbds::detail::ranged_hash_fn< Key, null_type, _Alloc, Comb_Hash_Fn, false ></code> Class Template Reference	. 886
4.283.1 Detailed Description	. 887
4.284 <code>_gnu_pbds::detail::ranged_hash_fn< Key, null_type, _Alloc, Comb_Hash_Fn, true ></code> Class Template Reference	. 887
4.284.1 Detailed Description	. 887
4.285 <code>_gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, Store_↵ Hash ></code> Class Template Reference	. 888
4.285.1 Detailed Description	. 888
4.286 <code>_gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, false ></code> Class Template Reference	. 888
4.286.1 Detailed Description	. 889
4.287 <code>_gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, true ></code> Class Template Reference	. 889
4.287.1 Detailed Description	. 890
4.288 <code>_gnu_pbds::detail::ranged_probe_fn< Key, null_type, _Alloc, Comb_Probe_Fn, null_type, false ></code> Class Template Reference	. 890
4.288.1 Detailed Description	. 890
4.289 <code>_gnu_pbds::detail::rb_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc ></code> Class Template Reference	. 891
4.289.1 Detailed Description	. 893
4.289.2 Member Function Documentation	. 894
4.290 <code>_gnu_pbds::detail::rb_tree_node_< Value_Type, Metadata, _Alloc ></code> Struct Template Reference	. 894
4.290.1 Detailed Description	. 895
4.291 <code>_gnu_pbds::detail::rc< _Node, _Alloc ></code> Class Template Reference	. 895
4.291.1 Detailed Description	. 896
4.292 <code>_gnu_pbds::detail::rc_binomial_heap< Value_Type, Cmp_Fn, _Alloc ></code> Class Template Reference	. 896
4.292.1 Detailed Description	. 898
4.293 <code>_gnu_pbds::detail::resize_policy< _Tp ></code> Class Template Reference	. 898
4.293.1 Detailed Description	. 899

4.294 __gnu_pbds::detail::splay_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > Class Template Reference	899
4.294.1 Detailed Description	902
4.294.2 Member Function Documentation	902
4.295 __gnu_pbds::detail::splay_tree_node_< Value_Type, Metadata, _Alloc > Struct Template Reference	903
4.295.1 Detailed Description	903
4.296 __gnu_pbds::detail::stored_data< _Tv, _Th > Struct Template Reference	904
4.296.1 Detailed Description	904
4.297 __gnu_pbds::detail::stored_data< _Tv, null_type > Struct Template Reference	905
4.297.1 Detailed Description	905
4.298 __gnu_pbds::detail::stored_hash< _Th > Struct Template Reference	906
4.298.1 Detailed Description	906
4.299 __gnu_pbds::detail::stored_value< _Tv > Struct Template Reference	907
4.299.1 Detailed Description	907
4.300 __gnu_pbds::detail::synth_access_traits< Type_Traits, Set, _ATraits > Struct Template Reference	907
4.300.1 Detailed Description	908
4.301 __gnu_pbds::detail::thin_heap< Value_Type, Cmp_Fn, _Alloc > Class Template Reference	908
4.301.1 Detailed Description	910
4.302 __gnu_pbds::detail::tree_metadata_helper< Node_Update, _BTp > Struct Template Reference	911
4.302.1 Detailed Description	911
4.303 __gnu_pbds::detail::tree_metadata_helper< Node_Update, false > Struct Template Reference	911
4.303.1 Detailed Description	911
4.304 __gnu_pbds::detail::tree_metadata_helper< Node_Update, true > Struct Template Reference	911
4.304.1 Detailed Description	912
4.305 __gnu_pbds::detail::tree_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc > Struct Template Reference	912
4.305.1 Detailed Description	912
4.306 __gnu_pbds::detail::tree_traits< Key, Data, Cmp_Fn, Node_Update, Tag, _Alloc > Struct Template Reference	912

4.306.1 Detailed Description	912
4.307__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc > Struct Template Reference	913
4.307.1 Detailed Description	913
4.307.2 Member Typedef Documentation	913
4.308__gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc > Struct Template Reference	913
4.308.1 Detailed Description	914
4.308.2 Member Typedef Documentation	914
4.309__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc > Struct Template Reference	914
4.309.1 Detailed Description	915
4.309.2 Member Typedef Documentation	915
4.310__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc > Struct Template Reference	916
4.310.1 Detailed Description	917
4.310.2 Member Typedef Documentation	917
4.311__gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc > Struct Template Reference	917
4.311.1 Detailed Description	918
4.311.2 Member Typedef Documentation	918
4.312__gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc > Struct Template Reference	919
4.312.1 Detailed Description	919
4.312.2 Member Typedef Documentation	920
4.313__gnu_pbds::detail::trie_metadata_helper< Node_Update, _BTp > Struct Template Reference	920
4.313.1 Detailed Description	920
4.314__gnu_pbds::detail::trie_metadata_helper< Node_Update, false > Struct Template Reference	920
4.314.1 Detailed Description	921
4.315__gnu_pbds::detail::trie_metadata_helper< Node_Update, true > Struct Template Reference	921
4.315.1 Detailed Description	921

4.316 __gnu_pbds::detail::trie_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc > Struct Template Reference	921
4.316.1 Detailed Description	922
4.317 __gnu_pbds::detail::trie_policy_base< Node_Cltr, Node_Itr, _ATraits, _Alloc > Class Template Reference	922
4.317.1 Detailed Description	923
4.318 __gnu_pbds::detail::trie_traits< Key, Data, _ATraits, Node_Update, Tag, _Alloc > Struct Template Reference	924
4.318.1 Detailed Description	924
4.319 __gnu_pbds::detail::trie_traits< Key, Mapped, _ATraits, Node_Update, pat_trie_tag, _Alloc > Struct Template Reference	924
4.319.1 Detailed Description	924
4.319.2 Member Typedef Documentation	925
4.320 __gnu_pbds::detail::trie_traits< Key, null_type, _ATraits, Node_Update, pat_trie_tag, _Alloc > Struct Template Reference	925
4.320.1 Detailed Description	926
4.320.2 Member Typedef Documentation	926
4.321 __gnu_pbds::detail::type_base< Key, Mapped, _Alloc, Store_Hash > Struct Template Reference	927
4.321.1 Detailed Description	927
4.322 __gnu_pbds::detail::type_base< Key, Mapped, _Alloc, false > Struct Template Reference	927
4.322.1 Detailed Description	928
4.323 __gnu_pbds::detail::type_base< Key, Mapped, _Alloc, true > Struct Template Reference	928
4.323.1 Detailed Description	928
4.324 __gnu_pbds::detail::type_base< Key, null_type, _Alloc, false > Struct Template Reference	929
4.324.1 Detailed Description	929
4.325 __gnu_pbds::detail::type_base< Key, null_type, _Alloc, true > Struct Template Reference	929
4.325.1 Detailed Description	930
4.326 __gnu_pbds::detail::type_dispatch< Key, Mapped, _Alloc, Store_Hash > Struct Template Reference	930
4.326.1 Detailed Description	930
4.327 __gnu_pbds::detail::types_traits< Key, Mapped, _Alloc, Store_Hash > Struct Template Reference	930
4.327.1 Detailed Description	931

4.328 __gnu_pbds::direct_mask_range_hashing< Size_Type > Class Template Reference	931
4.328.1 Detailed Description	932
4.328.2 Member Function Documentation	932
4.329 __gnu_pbds::direct_mod_range_hashing< Size_Type > Class Template Reference	933
4.329.1 Detailed Description	933
4.329.2 Member Function Documentation	934
4.330 __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc > Class Template Reference	934
4.330.1 Detailed Description	935
4.330.2 Constructor & Destructor Documentation	936
4.331 __gnu_pbds::gp_hash_tag Struct Reference	939
4.331.1 Detailed Description	940
4.332 __gnu_pbds::hash_exponential_size_policy< Size_Type > Class Template Reference	940
4.332.1 Detailed Description	940
4.332.2 Constructor & Destructor Documentation	940
4.333 __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type > Class Template Reference	941
4.333.1 Detailed Description	942
4.333.2 Member Enumeration Documentation	942
4.333.3 Constructor & Destructor Documentation	942
4.333.4 Member Function Documentation	942
4.334 __gnu_pbds::hash_prime_size_policy Class Reference	943
4.334.1 Detailed Description	944
4.334.2 Member Typedef Documentation	944
4.334.3 Constructor & Destructor Documentation	944
4.335 __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type > Class Template Reference	944
4.335.1 Detailed Description	945
4.335.2 Constructor & Destructor Documentation	945

4.335.3 Member Function Documentation	946
4.336 __gnu_pbds::insert_error Struct Reference	947
4.336.1 Detailed Description	948
4.337 __gnu_pbds::join_error Struct Reference	948
4.337.1 Detailed Description	948
4.338 __gnu_pbds::linear_probe_fn< Size_Type > Class Template Reference	948
4.338.1 Detailed Description	949
4.338.2 Member Function Documentation	949
4.339 __gnu_pbds::list_update< Key, Mapped, Eq_Fn, Update_Policy, _Alloc > Class Template Reference	949
4.339.1 Detailed Description	949
4.339.2 Constructor & Destructor Documentation	950
4.340 __gnu_pbds::list_update_tag Struct Reference	950
4.340.1 Detailed Description	951
4.341 __gnu_pbds::lu_counter_policy< Max_Count, _Alloc > Class Template Reference	951
4.341.1 Detailed Description	952
4.341.2 Member Typedef Documentation	952
4.341.3 Member Enumeration Documentation	952
4.341.4 Member Function Documentation	952
4.342 __gnu_pbds::lu_move_to_front_policy< _Alloc > Class Template Reference	953
4.342.1 Detailed Description	953
4.342.2 Member Typedef Documentation	953
4.342.3 Member Function Documentation	954
4.343 __gnu_pbds::null_node_update< _Tp1, _Tp2, _Tp3, _Tp4 > Struct Template Reference	954
4.343.1 Detailed Description	955
4.344 __gnu_pbds::null_type Struct Reference	955
4.344.1 Detailed Description	955
4.345 __gnu_pbds::ov_tree_tag Struct Reference	956
4.345.1 Detailed Description	956

4.346__gnu_pbds::pairing_heap_tag Struct Reference	957
4.346.1 Detailed Description	957
4.347__gnu_pbds::pat_trie_tag Struct Reference	958
4.347.1 Detailed Description	958
4.348__gnu_pbds::point_invalidation_guarantee Struct Reference	959
4.348.1 Detailed Description	959
4.349__gnu_pbds::priority_queue<_Tv, Cmp_Fn, Tag, _Alloc > Class Template Reference	959
4.349.1 Detailed Description	960
4.350__gnu_pbds::priority_queue_tag Struct Reference	961
4.350.1 Detailed Description	961
4.351__gnu_pbds::quadratic_probe_fn<Size_Type > Class Template Reference	961
4.351.1 Detailed Description	962
4.351.2 Member Function Documentation	962
4.352__gnu_pbds::range_invalidation_guarantee Struct Reference	962
4.352.1 Detailed Description	963
4.353__gnu_pbds::rb_tree_tag Struct Reference	963
4.353.1 Detailed Description	963
4.354__gnu_pbds::rc_binomial_heap_tag Struct Reference	964
4.354.1 Detailed Description	964
4.355__gnu_pbds::resize_error Struct Reference	964
4.355.1 Detailed Description	965
4.356__gnu_pbds::sample_probe_fn Class Reference	965
4.356.1 Detailed Description	965
4.356.2 Constructor & Destructor Documentation	965
4.356.3 Member Function Documentation	966
4.357__gnu_pbds::sample_range_hashing Class Reference	966
4.357.1 Detailed Description	966
4.357.2 Member Typedef Documentation	966

4.357.3 Constructor & Destructor Documentation	967
4.357.4 Member Function Documentation	967
4.358__gnu_pbds::sample_ranged_hash_fn Class Reference	967
4.358.1 Detailed Description	968
4.358.2 Constructor & Destructor Documentation	968
4.358.3 Member Function Documentation	968
4.359__gnu_pbds::sample_ranged_probe_fn Class Reference	968
4.359.1 Detailed Description	969
4.360__gnu_pbds::sample_resize_policy Class Reference	969
4.360.1 Detailed Description	970
4.360.2 Member Typedef Documentation	970
4.360.3 Constructor & Destructor Documentation	970
4.360.4 Member Function Documentation	970
4.361__gnu_pbds::sample_resize_trigger Class Reference	972
4.361.1 Detailed Description	972
4.361.2 Member Typedef Documentation	973
4.361.3 Constructor & Destructor Documentation	973
4.361.4 Member Function Documentation	973
4.362__gnu_pbds::sample_size_policy Class Reference	975
4.362.1 Detailed Description	975
4.362.2 Member Typedef Documentation	975
4.362.3 Constructor & Destructor Documentation	975
4.362.4 Member Function Documentation	975
4.363__gnu_pbds::sample_tree_node_update< Const_Node_Iter, Node_Iter, Cmp_Fn, _Alloc > Class Template Reference	976
4.363.1 Detailed Description	976
4.364__gnu_pbds::sample_trie_access_traits Struct Reference	976
4.364.1 Detailed Description	977

4.364.2 Member Typedef Documentation	977
4.364.3 Member Function Documentation	977
4.365 __gnu_pbds::sample_trie_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc > Class Template Reference	977
4.365.1 Detailed Description	978
4.365.2 Constructor & Destructor Documentation	978
4.365.3 Member Function Documentation	978
4.366 __gnu_pbds::sample_update_policy Struct Reference	978
4.366.1 Detailed Description	979
4.366.2 Member Typedef Documentation	979
4.366.3 Constructor & Destructor Documentation	979
4.366.4 Member Function Documentation	979
4.367 __gnu_pbds::sequence_tag Struct Reference	980
4.367.1 Detailed Description	980
4.368 __gnu_pbds::splay_tree_tag Struct Reference	981
4.368.1 Detailed Description	981
4.369 __gnu_pbds::string_tag Struct Reference	982
4.369.1 Detailed Description	982
4.370 __gnu_pbds::thin_heap_tag Struct Reference	983
4.370.1 Detailed Description	983
4.371 __gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc > Class Template Reference	983
4.371.1 Detailed Description	984
4.371.2 Member Typedef Documentation	984
4.371.3 Constructor & Destructor Documentation	985
4.372 __gnu_pbds::tree_order_statistics_node_update< Node_Cltr, Node_Itr, Cmp_Fn, _Alloc > Class Template Reference	986
4.372.1 Detailed Description	987
4.372.2 Member Function Documentation	987
4.373 __gnu_pbds::tree_tag Struct Reference	988

4.373.1 Detailed Description	989
4.374 <code>__gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc ></code> Class Template Reference	989
4.374.1 Detailed Description	989
4.374.2 Member Typedef Documentation	990
4.374.3 Constructor & Destructor Documentation	990
4.375 <code>__gnu_pbds::trie_order_statistics_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc ></code> Class Template Reference	991
4.375.1 Detailed Description	993
4.375.2 Member Function Documentation	993
4.376 <code>__gnu_pbds::trie_prefix_search_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc ></code> Class Template Reference	994
4.376.1 Detailed Description	996
4.376.2 Member Typedef Documentation	996
4.376.3 Member Function Documentation	996
4.377 <code>__gnu_pbds::trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc ></code> Struct Template Reference	998
4.377.1 Detailed Description	998
4.377.2 Member Typedef Documentation	998
4.377.3 Member Function Documentation	999
4.378 <code>__gnu_pbds::trie_tag</code> Struct Reference	1000
4.378.1 Detailed Description	1001
4.379 <code>__gnu_pbds::trivial_iterator_tag</code> Struct Reference	1001
4.379.1 Detailed Description	1001
4.380 <code>__gnu_profile::__container_size_info</code> Class Reference	1001
4.380.1 Detailed Description	1002
4.381 <code>__gnu_profile::__container_size_stack_info</code> Class Reference	1002
4.381.1 Detailed Description	1003
4.382 <code>__gnu_profile::__hashfunc_info</code> Class Reference	1003
4.382.1 Detailed Description	1004

4.383 __gnu_profile::__hashfunc_stack_info Class Reference	1004
4.383.1 Detailed Description	1005
4.384 __gnu_profile::__list2vector_info Class Reference	1005
4.384.1 Detailed Description	1006
4.385 __gnu_profile::__map2umap_info Class Reference	1007
4.385.1 Detailed Description	1007
4.386 __gnu_profile::__map2umap_stack_info Class Reference	1008
4.386.1 Detailed Description	1008
4.387 __gnu_profile::__object_info_base Class Reference	1009
4.387.1 Detailed Description	1009
4.388 __gnu_profile::__reentrance_guard Struct Reference	1009
4.388.1 Detailed Description	1010
4.389 __gnu_profile::__stack_hash Class Reference	1010
4.389.1 Detailed Description	1010
4.390 __gnu_profile::__stack_info_base< __object_info > Class Template Reference	1010
4.390.1 Detailed Description	1010
4.391 __gnu_profile::__trace_base< __object_info, __stack_info > Class Template Reference	1011
4.391.1 Detailed Description	1011
4.392 __gnu_profile::__trace_container_size Class Reference	1011
4.392.1 Detailed Description	1012
4.393 __gnu_profile::__trace_hash_func Class Reference	1012
4.393.1 Detailed Description	1013
4.394 __gnu_profile::__trace_hashtable_size Class Reference	1013
4.394.1 Detailed Description	1014
4.395 __gnu_profile::__trace_map2umap Class Reference	1014
4.395.1 Detailed Description	1015
4.396 __gnu_profile::__trace_vector_size Class Reference	1015
4.396.1 Detailed Description	1015

4.397 __gnu_profile::__trace_vector_to_list Class Reference	1016
4.397.1 Detailed Description	1016
4.398 __gnu_profile::__vector2list_info Class Reference	1017
4.398.1 Detailed Description	1018
4.399 __gnu_profile::__vector2list_stack_info Class Reference	1018
4.399.1 Detailed Description	1019
4.400 __gnu_profile::__warning_data Struct Reference	1019
4.400.1 Detailed Description	1019
4.401 const_iterator_ Class Reference	1020
4.401.1 Detailed Description	1021
4.401.2 Member Typedef Documentation	1021
4.401.3 Constructor & Destructor Documentation	1022
4.401.4 Member Function Documentation	1022
4.401.5 Member Data Documentation	1023
4.402 iterator_ Class Reference	1024
4.402.1 Detailed Description	1025
4.402.2 Member Typedef Documentation	1025
4.402.3 Constructor & Destructor Documentation	1026
4.402.4 Member Function Documentation	1026
4.402.5 Member Data Documentation	1028
4.403 point_const_iterator_ Class Reference	1028
4.403.1 Detailed Description	1029
4.403.2 Member Typedef Documentation	1029
4.403.3 Constructor & Destructor Documentation	1030
4.403.4 Member Function Documentation	1031
4.404 point_iterator_ Class Reference	1032
4.404.1 Detailed Description	1032
4.404.2 Member Typedef Documentation	1032

4.404.3 Constructor & Destructor Documentation	1033
4.404.4 Member Function Documentation	1034
4.405std::__atomic_base<_ITp> Struct Template Reference	1035
4.405.1 Detailed Description	1035
4.406std::__atomic_base<_PTp*> Struct Template Reference	1036
4.406.1 Detailed Description	1036
4.407std::__atomic_flag_base Struct Reference	1037
4.407.1 Detailed Description	1037
4.408std::__codecvt_abstract_base<_InternT, _ExternT, _StateT> Class Template Reference	1038
4.408.1 Detailed Description	1039
4.408.2 Member Function Documentation	1039
4.409std::__ctype_abstract_base<_CharT> Class Template Reference	1042
4.409.1 Detailed Description	1044
4.409.2 Member Typedef Documentation	1044
4.409.3 Member Function Documentation	1044
4.410std::__debug::map<_Key, _Tp, _Compare, _Allocator> Class Template Reference	1056
4.410.1 Detailed Description	1059
4.410.2 Member Function Documentation	1059
4.410.3 Member Data Documentation	1060
4.411std::__debug::multimap<_Key, _Tp, _Compare, _Allocator> Class Template Reference	1061
4.411.1 Detailed Description	1063
4.411.2 Member Function Documentation	1063
4.411.3 Member Data Documentation	1065
4.412std::__debug::multiset<_Key, _Compare, _Allocator> Class Template Reference	1066
4.412.1 Detailed Description	1068
4.412.2 Member Function Documentation	1068
4.412.3 Member Data Documentation	1070
4.413std::__debug::set<_Key, _Compare, _Allocator> Class Template Reference	1071

4.413.1 Detailed Description	1073
4.413.2 Member Function Documentation	1073
4.413.3 Member Data Documentation	1075
4.414std::__detail::__BracketMatcher< _TraitsT, __icase, __collate > Struct Template Reference	1075
4.414.1 Detailed Description	1076
4.415std::__detail::__Compiler< _TraitsT > Class Template Reference	1076
4.415.1 Detailed Description	1076
4.416std::__detail::__Default_ranged_hash Struct Reference	1077
4.416.1 Detailed Description	1077
4.417std::__detail::__Equal_helper< _Key, _Value, _ExtractKey, _Equal, _HashCodeType, __cache_hash↵ code > Struct Template Reference	1077
4.417.1 Detailed Description	1077
4.418std::__detail::__Equal_helper< _Key, _Value, _ExtractKey, _Equal, _HashCodeType, false > Struct Tem- plate Reference	1077
4.418.1 Detailed Description	1077
4.419std::__detail::__Equal_helper< _Key, _Value, _ExtractKey, _Equal, _HashCodeType, true > Struct Tem- plate Reference	1078
4.419.1 Detailed Description	1078
4.420std::__detail::__Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, _Unique_keys > Struct Template Reference	1078
4.420.1 Detailed Description	1079
4.421std::__detail::__Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false > Struct Template Reference	1079
4.421.1 Detailed Description	1080
4.422std::__detail::__Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true > Struct Template Reference	1080
4.422.1 Detailed Description	1080
4.423std::__detail::__Equality_base Struct Reference	1081
4.423.1 Detailed Description	1081
4.424std::__detail::__Executor< _Bilter, _Alloc, _TraitsT, __dfs_mode > Class Template Reference	1081
4.424.1 Detailed Description	1082

4.425std::__detail::Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache_hash_code > Struct Template Reference	1083
4.425.1 Detailed Description	1083
4.426std::__detail::Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Default_ranged_hash, false > Struct Template Reference	1083
4.426.1 Detailed Description	1084
4.427std::__detail::Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Default_ranged_hash, true > Struct Template Reference	1085
4.427.1 Detailed Description	1086
4.428std::__detail::Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, false > Struct Template Reference	1086
4.428.1 Detailed Description	1087
4.429std::__detail::Hash_node< _Value, _Cache_hash_code > Struct Template Reference	1087
4.429.1 Detailed Description	1087
4.430std::__detail::Hash_node< _Value, false > Struct Template Reference	1088
4.430.1 Detailed Description	1089
4.431std::__detail::Hash_node< _Value, true > Struct Template Reference	1089
4.431.1 Detailed Description	1090
4.432std::__detail::Hash_node_base Struct Reference	1090
4.432.1 Detailed Description	1091
4.433std::__detail::Hash_node_value_base< _Value > Struct Template Reference	1091
4.433.1 Detailed Description	1092
4.434std::__detail::Hashtable_alloc< _NodeAlloc > Struct Template Reference	1092
4.434.1 Detailed Description	1093
4.435std::__detail::Hashtable_base< _Key, _Value, _ExtractKey, _Equal, _H1, _H2, _Hash, _Traits > Struct Template Reference	1093
4.435.1 Detailed Description	1094
4.436std::__detail::Hashtable_ebo_helper< _Nm, _Tp, __use_ebo > Struct Template Reference	1095
4.436.1 Detailed Description	1095
4.437std::__detail::Hashtable_ebo_helper< _Nm, _Tp, false > Struct Template Reference	1095

4.437.1 Detailed Description	1095
4.438std::__detail::__Hashtable_ebo_helper< _Nm, _Tp, true > Struct Template Reference	1095
4.438.1 Detailed Description	1096
4.439std::__detail::__Hashtable_traits< _Cache_hash_code, _Constant_iterators, _Unique_keys > Struct Template Reference	1096
4.439.1 Detailed Description	1096
4.440std::__detail::__Insert< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, _Constant_iterators, _Unique_keys > Struct Template Reference	1097
4.440.1 Detailed Description	1097
4.441std::__detail::__Insert< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false, _Unique_keys > Struct Template Reference	1098
4.441.1 Detailed Description	1099
4.442std::__detail::__Insert< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true, false > Struct Template Reference	1099
4.442.1 Detailed Description	1100
4.443std::__detail::__Insert< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true, true > Struct Template Reference	1101
4.443.1 Detailed Description	1102
4.444std::__detail::__Insert_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits > Struct Template Reference	1102
4.444.1 Detailed Description	1103
4.445std::__detail::__List_node_base Struct Reference	1104
4.445.1 Detailed Description	1104
4.446std::__detail::__Local_const_iterator< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __constant_↵ iterators, __cache > Struct Template Reference	1105
4.446.1 Detailed Description	1105
4.447std::__detail::__Local_iterator< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __constant_iterators, __↵ cache > Struct Template Reference	1106
4.447.1 Detailed Description	1106
4.448std::__detail::__Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache_hash_code > Struct Template Reference	1107
4.448.1 Detailed Description	1107

4.449std::__detail::Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, true > Struct Template Reference	1107
4.449.1 Detailed Description	1108
4.450std::__detail::Map_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, _Unique_keys > Struct Template Reference	1108
4.450.1 Detailed Description	1109
4.451std::__detail::Map_base< _Key, _Pair, _Alloc, _Select1st, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false > Struct Template Reference	1109
4.451.1 Detailed Description	1109
4.452std::__detail::Map_base< _Key, _Pair, _Alloc, _Select1st, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true > Struct Template Reference	1109
4.452.1 Detailed Description	1110
4.453std::__detail::Mod_range_hashing Struct Reference	1110
4.453.1 Detailed Description	1110
4.454std::__detail::Node_const_iterator< _Value, __constant_iterators, __cache > Struct Template Reference	1111
4.454.1 Detailed Description	1112
4.455std::__detail::Node_iterator< _Value, __constant_iterators, __cache > Struct Template Reference	1112
4.455.1 Detailed Description	1113
4.456std::__detail::Node_iterator_base< _Value, _Cache_hash_code > Struct Template Reference	1113
4.456.1 Detailed Description	1113
4.457std::__detail::Prime_rehash_policy Struct Reference	1114
4.457.1 Detailed Description	1114
4.458std::__detail::Rehash_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits > Struct Template Reference	1115
4.458.1 Detailed Description	1115
4.459std::__detail::Rehash_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _Prime_rehash_policy, _Traits > Struct Template Reference	1115
4.459.1 Detailed Description	1116
4.460std::__detail::Scanner< _CharT > Class Template Reference	1116
4.460.1 Detailed Description	1117
4.460.2 Member Enumeration Documentation	1118

4.461	std::__detail::__StateSeq< _TraitsT > Class Template Reference	1118
4.461.1	Detailed Description	1118
4.462	std::__exception_ptr::exception_ptr Class Reference	1119
4.462.1	Detailed Description	1119
4.463	std::__has_iterator_category_helper< _Tp > Class Template Reference	1119
4.463.1	Detailed Description	1119
4.464	std::__parallel::__CRandNumber< _MustBeInt > Struct Template Reference	1120
4.464.1	Detailed Description	1120
4.465	std::__profile::map< _Key, _Tp, _Compare, _Allocator > Class Template Reference	1120
4.465.1	Detailed Description	1122
4.466	std::__profile::multimap< _Key, _Tp, _Compare, _Allocator > Class Template Reference	1122
4.466.1	Detailed Description	1124
4.467	std::__profile::multiset< _Key, _Compare, _Allocator > Class Template Reference	1124
4.467.1	Detailed Description	1126
4.468	std::__profile::set< _Key, _Compare, _Allocator > Class Template Reference	1126
4.468.1	Detailed Description	1128
4.469	std::__Deque_base< _Tp, _Alloc > Class Template Reference	1128
4.469.1	Detailed Description	1129
4.469.2	Member Function Documentation	1129
4.470	std::__Deque_iterator< _Tp, _Ref, _Ptr > Struct Template Reference	1130
4.470.1	Detailed Description	1131
4.470.2	Member Function Documentation	1131
4.471	std::__Enable_copy_move< _Copy, _CopyAssignment, _Move, _MoveAssignment, _Tag > Struct Template Reference	1132
4.471.1	Detailed Description	1132
4.472	std::__Enable_default_constructor< _Switch, _Tag > Struct Template Reference	1132
4.472.1	Detailed Description	1132
4.473	std::__Enable_destructor< _Switch, _Tag > Struct Template Reference	1133

4.473.1 Detailed Description	1133
4.474std::Enable_special_members< _Default, _Destructor, _Copy, _CopyAssignment, _Move, _Move← Assignment, _Tag > Struct Template Reference	1133
4.474.1 Detailed Description	1134
4.475std::_Fwd_list_base< _Tp, _Alloc > Struct Template Reference	1134
4.475.1 Detailed Description	1135
4.476std::_Fwd_list_const_iterator< _Tp > Struct Template Reference	1135
4.476.1 Detailed Description	1136
4.477std::_Fwd_list_iterator< _Tp > Struct Template Reference	1136
4.477.1 Detailed Description	1137
4.478std::_Fwd_list_node< _Tp > Struct Template Reference	1137
4.478.1 Detailed Description	1138
4.479std::_Fwd_list_node_base Struct Reference	1138
4.479.1 Detailed Description	1139
4.480std::_Hashtable< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits > Class Template Reference	1139
4.480.1 Detailed Description	1143
4.481std::_List_base< _Tp, _Alloc > Class Template Reference	1145
4.481.1 Detailed Description	1146
4.482std::_List_const_iterator< _Tp > Struct Template Reference	1146
4.482.1 Detailed Description	1147
4.483std::_List_iterator< _Tp > Struct Template Reference	1147
4.483.1 Detailed Description	1148
4.484std::_List_node< _Tp > Struct Template Reference	1148
4.484.1 Detailed Description	1149
4.484.2 Member Data Documentation	1149
4.485std::_Sp_ebo_helper< _Nm, _Tp, false > Struct Template Reference	1149
4.485.1 Detailed Description	1150
4.486std::_Sp_ebo_helper< _Nm, _Tp, true > Struct Template Reference	1150

4.486.1 Detailed Description	1150
4.487std::_Temporary_buffer<_ForwardIterator, _Tp > Class Template Reference	1150
4.487.1 Detailed Description	1151
4.487.2 Constructor & Destructor Documentation	1151
4.487.3 Member Function Documentation	1151
4.488std::_Vector_base<_Tp, _Alloc > Struct Template Reference	1152
4.488.1 Detailed Description	1153
4.489std::allocator<_Tp > Class Template Reference	1153
4.489.1 Detailed Description	1154
4.490std::allocator<void > Class Template Reference	1155
4.490.1 Detailed Description	1155
4.491std::allocator_arg_t Struct Reference	1155
4.491.1 Detailed Description	1155
4.492std::allocator_traits<_Alloc > Struct Template Reference	1156
4.492.1 Detailed Description	1157
4.492.2 Member Typedef Documentation	1157
4.492.3 Member Function Documentation	1158
4.493std::allocator_traits<allocator<_Tp > > Struct Template Reference	1161
4.493.1 Detailed Description	1162
4.493.2 Member Typedef Documentation	1162
4.493.3 Member Function Documentation	1163
4.494std::atomic_flag Struct Reference	1166
4.494.1 Detailed Description	1167
4.495std::auto_ptr<_Tp > Class Template Reference	1167
4.495.1 Detailed Description	1168
4.495.2 Member Typedef Documentation	1168
4.495.3 Constructor & Destructor Documentation	1168
4.495.4 Member Function Documentation	1170

4.496std::auto_ptr_ref< _Tp1 > Struct Template Reference	1172
4.496.1 Detailed Description	1172
4.497std::back_insert_iterator< _Container > Class Template Reference	1173
4.497.1 Detailed Description	1174
4.497.2 Member Typedef Documentation	1174
4.497.3 Constructor & Destructor Documentation	1175
4.497.4 Member Function Documentation	1175
4.498std::bad_weak_ptr Class Reference	1176
4.498.1 Detailed Description	1176
4.499std::basic_ios< _CharT, _Traits > Class Template Reference	1176
4.499.1 Detailed Description	1179
4.499.2 Member Typedef Documentation	1180
4.499.3 Member Enumeration Documentation	1183
4.499.4 Constructor & Destructor Documentation	1183
4.499.5 Member Function Documentation	1184
4.499.6 Member Data Documentation	1197
4.500std::basic_regex< _Ch_type, _Rx_traits > Class Template Reference	1202
4.500.1 Detailed Description	1204
4.500.2 Constructor & Destructor Documentation	1204
4.500.3 Member Function Documentation	1207
4.501std::basic_string< _CharT, _Traits, _Alloc > Class Template Reference	1212
4.501.1 Detailed Description	1215
4.501.2 Constructor & Destructor Documentation	1216
4.501.3 Member Function Documentation	1220
4.501.4 Member Data Documentation	1263
4.502std::bernoulli_distribution Class Reference	1264
4.502.1 Detailed Description	1264
4.502.2 Member Typedef Documentation	1265

4.502.3 Constructor & Destructor Documentation	1265
4.502.4 Member Function Documentation	1265
4.502.5 Friends And Related Function Documentation	1266
4.503std::bernoulli_distribution::param_type Struct Reference	1267
4.503.1 Detailed Description	1267
4.504std::bidirectional_iterator_tag Struct Reference	1267
4.504.1 Detailed Description	1268
4.505std::binary_function< _Arg1, _Arg2, _Result > Struct Template Reference	1268
4.505.1 Detailed Description	1268
4.505.2 Member Typedef Documentation	1269
4.506std::binary_negate< _Predicate > Class Template Reference	1269
4.506.1 Detailed Description	1270
4.506.2 Member Typedef Documentation	1270
4.507std::binder1st< _Operation > Class Template Reference	1271
4.507.1 Detailed Description	1271
4.507.2 Member Typedef Documentation	1272
4.508std::binder2nd< _Operation > Class Template Reference	1272
4.508.1 Detailed Description	1273
4.508.2 Member Typedef Documentation	1273
4.509std::binomial_distribution< _IntType > Class Template Reference	1273
4.509.1 Detailed Description	1274
4.509.2 Member Typedef Documentation	1275
4.509.3 Member Function Documentation	1275
4.509.4 Friends And Related Function Documentation	1276
4.510std::binomial_distribution< _IntType >::param_type Struct Reference	1277
4.510.1 Detailed Description	1277
4.511std::cauchy_distribution< _RealType > Class Template Reference	1278
4.511.1 Detailed Description	1278

4.511.2 Member Typedef Documentation	1279
4.511.3 Member Function Documentation	1279
4.511.4 Friends And Related Function Documentation	1280
4.512std::cauchy_distribution< _RealType >::param_type Struct Reference	1280
4.512.1 Detailed Description	1281
4.513std::char_traits< _CharT > Struct Template Reference	1281
4.513.1 Detailed Description	1282
4.514std::char_traits< __gnu_cxx::character< _Value, _Int, _St > > Struct Template Reference	1282
4.514.1 Detailed Description	1283
4.515std::char_traits< char > Struct Template Reference	1283
4.515.1 Detailed Description	1284
4.516std::char_traits< wchar_t > Struct Template Reference	1284
4.516.1 Detailed Description	1285
4.517std::chi_squared_distribution< _RealType > Class Template Reference	1285
4.517.1 Detailed Description	1286
4.517.2 Member Typedef Documentation	1287
4.517.3 Member Function Documentation	1287
4.517.4 Friends And Related Function Documentation	1288
4.518std::chi_squared_distribution< _RealType >::param_type Struct Reference	1289
4.518.1 Detailed Description	1289
4.519std::codecvt< _InternT, _ExternT, _StateT > Class Template Reference	1290
4.519.1 Detailed Description	1291
4.519.2 Member Function Documentation	1292
4.520std::codecvt< _InternT, _ExternT, encoding_state > Class Template Reference	1295
4.520.1 Detailed Description	1296
4.520.2 Member Function Documentation	1296
4.521std::codecvt< char, char, mbstate_t > Class Template Reference	1299
4.521.1 Detailed Description	1300

4.521.2 Member Function Documentation	1301
4.522std::codecvt< wchar_t, char, mbstate_t > Class Template Reference	1304
4.522.1 Detailed Description	1305
4.522.2 Member Function Documentation	1305
4.523std::codecvt_base Class Reference	1308
4.523.1 Detailed Description	1308
4.524std::codecvt_byname< _InternT, _ExternT, _StateT > Class Template Reference	1309
4.524.1 Detailed Description	1310
4.524.2 Member Function Documentation	1311
4.525std::collate< _CharT > Class Template Reference	1313
4.525.1 Detailed Description	1315
4.525.2 Member Typedef Documentation	1315
4.525.3 Constructor & Destructor Documentation	1315
4.525.4 Member Function Documentation	1316
4.525.5 Member Data Documentation	1319
4.526std::collate_byname< _CharT > Class Template Reference	1319
4.526.1 Detailed Description	1320
4.526.2 Member Typedef Documentation	1321
4.526.3 Member Function Documentation	1321
4.526.4 Member Data Documentation	1324
4.527std::const_mem_fun1_ref_t< _Ret, _Tp, _Arg > Class Template Reference	1324
4.527.1 Detailed Description	1325
4.527.2 Member Typedef Documentation	1325
4.528std::const_mem_fun1_t< _Ret, _Tp, _Arg > Class Template Reference	1325
4.528.1 Detailed Description	1326
4.528.2 Member Typedef Documentation	1326
4.529std::const_mem_fun_ref_t< _Ret, _Tp > Class Template Reference	1327
4.529.1 Detailed Description	1327

4.529.2 Member Typedef Documentation	1327
4.530std::const_mem_fun_t< _Ret, _Tp > Class Template Reference	1328
4.530.1 Detailed Description	1328
4.530.2 Member Typedef Documentation	1329
4.531std::ctype< _CharT > Class Template Reference	1329
4.531.1 Detailed Description	1331
4.531.2 Member Function Documentation	1331
4.531.3 Member Data Documentation	1343
4.532std::ctype< char > Class Template Reference	1343
4.532.1 Detailed Description	1345
4.532.2 Member Typedef Documentation	1345
4.532.3 Constructor & Destructor Documentation	1345
4.532.4 Member Function Documentation	1346
4.532.5 Member Data Documentation	1356
4.533std::ctype< wchar_t > Class Template Reference	1357
4.533.1 Detailed Description	1359
4.533.2 Member Typedef Documentation	1359
4.533.3 Constructor & Destructor Documentation	1359
4.533.4 Member Function Documentation	1360
4.533.5 Member Data Documentation	1372
4.534std::ctype_base Struct Reference	1372
4.534.1 Detailed Description	1373
4.535std::ctype_byname< _CharT > Class Template Reference	1374
4.535.1 Detailed Description	1375
4.535.2 Member Function Documentation	1376
4.535.3 Member Data Documentation	1388
4.536std::ctype_byname< char > Class Template Reference	1389
4.536.1 Detailed Description	1391

4.536.2 Member Typedef Documentation	1391
4.536.3 Member Function Documentation	1391
4.536.4 Member Data Documentation	1401
4.537std::default_delete<_Tp> Struct Template Reference	1401
4.537.1 Detailed Description	1402
4.537.2 Constructor & Destructor Documentation	1402
4.537.3 Member Function Documentation	1402
4.538std::default_delete<_Tp[]> Struct Template Reference	1402
4.538.1 Detailed Description	1403
4.538.2 Constructor & Destructor Documentation	1403
4.538.3 Member Function Documentation	1403
4.539std::deque<_Tp, _Alloc> Class Template Reference	1404
4.539.1 Detailed Description	1407
4.539.2 Constructor & Destructor Documentation	1409
4.539.3 Member Function Documentation	1411
4.540std::discard_block_engine<_RandomNumberEngine, __p, __r> Class Template Reference	1427
4.540.1 Detailed Description	1428
4.540.2 Member Typedef Documentation	1428
4.540.3 Constructor & Destructor Documentation	1428
4.540.4 Member Function Documentation	1430
4.540.5 Friends And Related Function Documentation	1431
4.541std::discrete_distribution<_IntType> Class Template Reference	1432
4.541.1 Detailed Description	1433
4.541.2 Member Typedef Documentation	1433
4.541.3 Member Function Documentation	1434
4.541.4 Friends And Related Function Documentation	1435
4.542std::discrete_distribution<_IntType>::param_type Struct Reference	1436
4.542.1 Detailed Description	1436

4.543std::divides<_Tp> Struct Template Reference	1437
4.543.1 Detailed Description	1437
4.543.2 Member Typedef Documentation	1437
4.544std::enable_shared_from_this<_Tp> Class Template Reference	1438
4.544.1 Detailed Description	1438
4.545std::equal_to<_Tp> Struct Template Reference	1439
4.545.1 Detailed Description	1439
4.545.2 Member Typedef Documentation	1439
4.546std::exponential_distribution<_RealType> Class Template Reference	1440
4.546.1 Detailed Description	1441
4.546.2 Member Typedef Documentation	1441
4.546.3 Constructor & Destructor Documentation	1441
4.546.4 Member Function Documentation	1442
4.546.5 Friends And Related Function Documentation	1443
4.547std::exponential_distribution<_RealType>::param_type Struct Reference	1443
4.547.1 Detailed Description	1444
4.548std::extreme_value_distribution<_RealType> Class Template Reference	1444
4.548.1 Detailed Description	1445
4.548.2 Member Typedef Documentation	1445
4.548.3 Member Function Documentation	1445
4.548.4 Friends And Related Function Documentation	1447
4.549std::extreme_value_distribution<_RealType>::param_type Struct Reference	1447
4.549.1 Detailed Description	1447
4.550std::fisher_f_distribution<_RealType> Class Template Reference	1448
4.550.1 Detailed Description	1449
4.550.2 Member Typedef Documentation	1449
4.550.3 Member Function Documentation	1449
4.550.4 Friends And Related Function Documentation	1450

4.551	std::fisher_f_distribution<_RealType>::param_type Struct Reference	1451
4.551.1	Detailed Description	1451
4.552	std::forward_iterator_tag Struct Reference	1452
4.552.1	Detailed Description	1452
4.553	std::forward_list<_Tp, _Alloc> Class Template Reference	1453
4.553.1	Detailed Description	1455
4.553.2	Constructor & Destructor Documentation	1456
4.553.3	Member Function Documentation	1459
4.554	std::fpos<_StateT> Class Template Reference	1471
4.554.1	Detailed Description	1472
4.554.2	Constructor & Destructor Documentation	1473
4.554.3	Member Function Documentation	1473
4.555	std::front_insert_iterator<_Container> Class Template Reference	1474
4.555.1	Detailed Description	1475
4.555.2	Member Typedef Documentation	1475
4.555.3	Constructor & Destructor Documentation	1476
4.555.4	Member Function Documentation	1476
4.556	std::gamma_distribution<_RealType> Class Template Reference	1477
4.556.1	Detailed Description	1478
4.556.2	Member Typedef Documentation	1478
4.556.3	Constructor & Destructor Documentation	1478
4.556.4	Member Function Documentation	1479
4.556.5	Friends And Related Function Documentation	1480
4.557	std::gamma_distribution<_RealType>::param_type Struct Reference	1481
4.557.1	Detailed Description	1481
4.558	std::geometric_distribution<_IntType> Class Template Reference	1482
4.558.1	Detailed Description	1482
4.558.2	Member Typedef Documentation	1483

4.558.3 Member Function Documentation	1483
4.558.4 Friends And Related Function Documentation	1484
4.559std::geometric_distribution< _IntType >::param_type Struct Reference	1484
4.559.1 Detailed Description	1485
4.560std::greater< _Tp > Struct Template Reference	1485
4.560.1 Detailed Description	1485
4.560.2 Member Typedef Documentation	1486
4.561std::greater_equal< _Tp > Struct Template Reference	1486
4.561.1 Detailed Description	1487
4.561.2 Member Typedef Documentation	1487
4.562std::gslice Class Reference	1487
4.562.1 Detailed Description	1488
4.563std::gslice_array< _Tp > Class Template Reference	1488
4.563.1 Detailed Description	1489
4.564std::hash< _Tp > Struct Template Reference	1490
4.564.1 Detailed Description	1490
4.565std::hash< __gnu_cxx::__u16vstring > Struct Template Reference	1490
4.565.1 Detailed Description	1490
4.566std::hash< __gnu_cxx::__u32vstring > Struct Template Reference	1490
4.566.1 Detailed Description	1491
4.567std::hash< __gnu_cxx::__vstring > Struct Template Reference	1491
4.567.1 Detailed Description	1491
4.568std::hash< __gnu_cxx::__wvstring > Struct Template Reference	1492
4.568.1 Detailed Description	1492
4.569std::hash< __gnu_cxx::throw_value_limit > Struct Template Reference	1492
4.569.1 Detailed Description	1493
4.569.2 Member Typedef Documentation	1493
4.570std::hash< __gnu_cxx::throw_value_random > Struct Template Reference	1494

4.570.1 Detailed Description	1494
4.570.2 Member Typedef Documentation	1494
4.571std::hash< __shared_ptr< _Tp, _Lp > > Struct Template Reference	1495
4.571.1 Detailed Description	1495
4.572std::hash< _Tp * > Struct Template Reference	1495
4.572.1 Detailed Description	1496
4.573std::hash< bool > Struct Template Reference	1496
4.573.1 Detailed Description	1496
4.574std::hash< char > Struct Template Reference	1497
4.574.1 Detailed Description	1497
4.575std::hash< char16_t > Struct Template Reference	1497
4.575.1 Detailed Description	1498
4.576std::hash< char32_t > Struct Template Reference	1498
4.576.1 Detailed Description	1498
4.577std::hash< double > Struct Template Reference	1498
4.577.1 Detailed Description	1499
4.578std::hash< float > Struct Template Reference	1499
4.578.1 Detailed Description	1499
4.579std::hash< int > Struct Template Reference	1500
4.579.1 Detailed Description	1500
4.580std::hash< long > Struct Template Reference	1500
4.580.1 Detailed Description	1501
4.581std::hash< long double > Struct Template Reference	1501
4.581.1 Detailed Description	1501
4.582std::hash< long long > Struct Template Reference	1501
4.582.1 Detailed Description	1502
4.583std::hash< shared_ptr< _Tp > > Struct Template Reference	1502
4.583.1 Detailed Description	1502

4.584std::hash< short > Struct Template Reference	1503
4.584.1 Detailed Description	1503
4.585std::hash< signed char > Struct Template Reference	1503
4.585.1 Detailed Description	1504
4.586std::hash< string > Struct Template Reference	1504
4.586.1 Detailed Description	1504
4.587std::hash< u16string > Struct Template Reference	1504
4.587.1 Detailed Description	1505
4.588std::hash< u32string > Struct Template Reference	1505
4.588.1 Detailed Description	1505
4.589std::hash< unique_ptr< _Tp, _Dp > > Struct Template Reference	1506
4.589.1 Detailed Description	1506
4.590std::hash< unsigned char > Struct Template Reference	1506
4.590.1 Detailed Description	1507
4.591std::hash< unsigned int > Struct Template Reference	1507
4.591.1 Detailed Description	1507
4.592std::hash< unsigned long > Struct Template Reference	1507
4.592.1 Detailed Description	1508
4.593std::hash< unsigned long long > Struct Template Reference	1508
4.593.1 Detailed Description	1508
4.594std::hash< unsigned short > Struct Template Reference	1509
4.594.1 Detailed Description	1509
4.595std::hash< wchar_t > Struct Template Reference	1509
4.595.1 Detailed Description	1510
4.596std::hash< wstring > Struct Template Reference	1510
4.596.1 Detailed Description	1510
4.597std::hash<::vector< bool, _Alloc > > Struct Template Reference	1510
4.597.1 Detailed Description	1511

4.598	std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType > Class Template Reference	1511
4.598.1	Detailed Description	1512
4.598.2	Member Typedef Documentation	1512
4.598.3	Constructor & Destructor Documentation	1512
4.598.4	Member Function Documentation	1514
4.598.5	Friends And Related Function Documentation	1515
4.599	std::indirect_array<_Tp > Class Template Reference	1516
4.599.1	Detailed Description	1517
4.600	std::input_iterator_tag Struct Reference	1518
4.600.1	Detailed Description	1518
4.601	std::insert_iterator<_Container > Class Template Reference	1519
4.601.1	Detailed Description	1520
4.601.2	Member Typedef Documentation	1520
4.601.3	Constructor & Destructor Documentation	1521
4.601.4	Member Function Documentation	1521
4.602	std::ios_base Class Reference	1522
4.602.1	Detailed Description	1525
4.602.2	Member Typedef Documentation	1525
4.602.3	Member Enumeration Documentation	1527
4.602.4	Constructor & Destructor Documentation	1527
4.602.5	Member Function Documentation	1527
4.602.6	Member Data Documentation	1533
4.603	std::ios_base::failure Class Reference	1538
4.603.1	Detailed Description	1538
4.604	std::istream_iterator<_Tp, _CharT, _Traits, _Dist > Class Template Reference	1538
4.604.1	Detailed Description	1539
4.604.2	Member Typedef Documentation	1539
4.604.3	Constructor & Destructor Documentation	1540

4.605std::istreambuf_iterator< _CharT, _Traits > Class Template Reference	1541
4.605.1 Detailed Description	1542
4.605.2 Member Typedef Documentation	1542
4.605.3 Constructor & Destructor Documentation	1543
4.605.4 Member Function Documentation	1544
4.606std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference > Struct Template Reference	1545
4.606.1 Detailed Description	1545
4.606.2 Member Typedef Documentation	1545
4.607std::iterator_traits< _Tp * > Struct Template Reference	1546
4.607.1 Detailed Description	1546
4.608std::iterator_traits< const _Tp * > Struct Template Reference	1546
4.608.1 Detailed Description	1547
4.609std::less< _Tp > Struct Template Reference	1547
4.609.1 Detailed Description	1547
4.609.2 Member Typedef Documentation	1548
4.610std::less_equal< _Tp > Struct Template Reference	1548
4.610.1 Detailed Description	1549
4.610.2 Member Typedef Documentation	1549
4.611std::linear_congruential_engine< _UIntType, __a, __c, __m > Class Template Reference	1549
4.611.1 Detailed Description	1550
4.611.2 Member Typedef Documentation	1551
4.611.3 Constructor & Destructor Documentation	1551
4.611.4 Member Function Documentation	1551
4.611.5 Friends And Related Function Documentation	1553
4.611.6 Member Data Documentation	1554
4.612std::list< _Tp, _Alloc > Class Template Reference	1554
4.612.1 Detailed Description	1557
4.612.2 Constructor & Destructor Documentation	1558

4.612.3 Member Function Documentation	1560
4.613std::locale Class Reference	1575
4.613.1 Detailed Description	1576
4.613.2 Member Typedef Documentation	1576
4.613.3 Constructor & Destructor Documentation	1577
4.613.4 Member Function Documentation	1578
4.613.5 Member Data Documentation	1581
4.614std::locale::facet Class Reference	1584
4.614.1 Detailed Description	1585
4.614.2 Constructor & Destructor Documentation	1585
4.615std::locale::id Class Reference	1586
4.615.1 Detailed Description	1586
4.615.2 Constructor & Destructor Documentation	1586
4.616std::logical_and<_Tp> Struct Template Reference	1587
4.616.1 Detailed Description	1587
4.616.2 Member Typedef Documentation	1587
4.617std::logical_not<_Tp> Struct Template Reference	1588
4.617.1 Detailed Description	1589
4.617.2 Member Typedef Documentation	1589
4.618std::logical_or<_Tp> Struct Template Reference	1589
4.618.1 Detailed Description	1590
4.618.2 Member Typedef Documentation	1590
4.619std::lognormal_distribution<_RealType> Class Template Reference	1591
4.619.1 Detailed Description	1592
4.619.2 Member Typedef Documentation	1592
4.619.3 Member Function Documentation	1592
4.619.4 Friends And Related Function Documentation	1593
4.620std::lognormal_distribution<_RealType>::param_type Struct Reference	1594

4.620.1 Detailed Description	1594
4.621std::map<_Key, _Tp, _Compare, _Alloc > Class Template Reference	1595
4.621.1 Detailed Description	1597
4.621.2 Constructor & Destructor Documentation	1597
4.621.3 Member Function Documentation	1600
4.622std::mask_array<_Tp > Class Template Reference	1615
4.622.1 Detailed Description	1617
4.623std::match_results<_Bi_iter, _Alloc > Class Template Reference	1617
4.623.1 Detailed Description	1621
4.623.2 Constructor & Destructor Documentation	1622
4.623.3 Member Function Documentation	1622
4.624std::mem_fun1_ref_t<_Ret, _Tp, _Arg > Class Template Reference	1629
4.624.1 Detailed Description	1629
4.624.2 Member Typedef Documentation	1629
4.625std::mem_fun1_t<_Ret, _Tp, _Arg > Class Template Reference	1630
4.625.1 Detailed Description	1631
4.625.2 Member Typedef Documentation	1631
4.626std::mem_fun_ref_t<_Ret, _Tp > Class Template Reference	1631
4.626.1 Detailed Description	1632
4.626.2 Member Typedef Documentation	1632
4.627std::mem_fun_t<_Ret, _Tp > Class Template Reference	1633
4.627.1 Detailed Description	1633
4.627.2 Member Typedef Documentation	1633
4.628std::mersenne_twister_engine<_UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > Class Template Reference	1634
4.628.1 Detailed Description	1635
4.628.2 Member Typedef Documentation	1636
4.628.3 Constructor & Destructor Documentation	1636

4.628.4 Member Function Documentation	1636
4.628.5 Friends And Related Function Documentation	1637
4.629std::messages< _CharT > Class Template Reference	1639
4.629.1 Detailed Description	1640
4.629.2 Member Typedef Documentation	1641
4.629.3 Constructor & Destructor Documentation	1641
4.629.4 Member Function Documentation	1642
4.629.5 Member Data Documentation	1642
4.630std::messages_base Struct Reference	1642
4.630.1 Detailed Description	1643
4.631std::messages_byname< _CharT > Class Template Reference	1643
4.631.1 Detailed Description	1644
4.631.2 Member Function Documentation	1645
4.631.3 Member Data Documentation	1645
4.632std::minus< _Tp > Struct Template Reference	1645
4.632.1 Detailed Description	1646
4.632.2 Member Typedef Documentation	1646
4.633std::modulus< _Tp > Struct Template Reference	1647
4.633.1 Detailed Description	1647
4.633.2 Member Typedef Documentation	1647
4.634std::money_base Class Reference	1648
4.634.1 Detailed Description	1649
4.635std::money_get< _CharT, _InIter > Class Template Reference	1649
4.635.1 Detailed Description	1650
4.635.2 Member Typedef Documentation	1651
4.635.3 Constructor & Destructor Documentation	1651
4.635.4 Member Function Documentation	1652
4.635.5 Member Data Documentation	1653

4.636std::money_put<_CharT, _Outlter > Class Template Reference	1654
4.636.1 Detailed Description	1655
4.636.2 Member Typedef Documentation	1655
4.636.3 Constructor & Destructor Documentation	1655
4.636.4 Member Function Documentation	1656
4.636.5 Member Data Documentation	1658
4.637std::moneypunct<_CharT, _Intl > Class Template Reference	1659
4.637.1 Detailed Description	1660
4.637.2 Member Typedef Documentation	1661
4.637.3 Constructor & Destructor Documentation	1661
4.637.4 Member Function Documentation	1662
4.637.5 Member Data Documentation	1669
4.638std::moneypunct_byname<_CharT, _Intl > Class Template Reference	1669
4.638.1 Detailed Description	1671
4.638.2 Member Function Documentation	1671
4.638.3 Member Data Documentation	1678
4.639std::move_iterator<_Iterator > Class Template Reference	1678
4.639.1 Detailed Description	1679
4.640std::multimap<_Key, _Tp, _Compare, _Alloc > Class Template Reference	1679
4.640.1 Detailed Description	1681
4.640.2 Constructor & Destructor Documentation	1682
4.640.3 Member Function Documentation	1685
4.641std::multiplies<_Tp > Struct Template Reference	1697
4.641.1 Detailed Description	1698
4.641.2 Member Typedef Documentation	1698
4.642std::multiset<_Key, _Compare, _Alloc > Class Template Reference	1699
4.642.1 Detailed Description	1700
4.642.2 Constructor & Destructor Documentation	1701

4.642.3 Member Function Documentation	1704
4.643std::negate<_Tp> Struct Template Reference	1716
4.643.1 Detailed Description	1716
4.643.2 Member Typedef Documentation	1716
4.644std::negative_binomial_distribution<_IntType> Class Template Reference	1717
4.644.1 Detailed Description	1718
4.644.2 Member Typedef Documentation	1718
4.644.3 Member Function Documentation	1718
4.644.4 Friends And Related Function Documentation	1719
4.645std::negative_binomial_distribution<_IntType>::param_type Struct Reference	1720
4.645.1 Detailed Description	1721
4.646std::nested_exception Class Reference	1721
4.646.1 Detailed Description	1721
4.647std::normal_distribution<_RealType> Class Template Reference	1722
4.647.1 Detailed Description	1723
4.647.2 Member Typedef Documentation	1723
4.647.3 Constructor & Destructor Documentation	1723
4.647.4 Member Function Documentation	1723
4.647.5 Friends And Related Function Documentation	1725
4.648std::normal_distribution<_RealType>::param_type Struct Reference	1726
4.648.1 Detailed Description	1727
4.649std::not_equal_to<_Tp> Struct Template Reference	1727
4.649.1 Detailed Description	1728
4.649.2 Member Typedef Documentation	1728
4.650std::num_get<_CharT, _InIter> Class Template Reference	1728
4.650.1 Detailed Description	1730
4.650.2 Member Typedef Documentation	1730
4.650.3 Constructor & Destructor Documentation	1731

4.650.4 Member Function Documentation	1731
4.650.5 Member Data Documentation	1744
4.651std::num_put< _CharT, _OutIter > Class Template Reference	1744
4.651.1 Detailed Description	1746
4.651.2 Member Typedef Documentation	1746
4.651.3 Constructor & Destructor Documentation	1746
4.651.4 Member Function Documentation	1747
4.651.5 Member Data Documentation	1756
4.652std::numprint< _CharT > Class Template Reference	1757
4.652.1 Detailed Description	1758
4.652.2 Member Typedef Documentation	1759
4.652.3 Constructor & Destructor Documentation	1759
4.652.4 Member Function Documentation	1760
4.652.5 Member Data Documentation	1763
4.653std::numprint_byname< _CharT > Class Template Reference	1764
4.653.1 Detailed Description	1765
4.653.2 Member Function Documentation	1765
4.653.3 Member Data Documentation	1769
4.654std::ostream_iterator< _Tp, _CharT, _Traits > Class Template Reference	1769
4.654.1 Detailed Description	1770
4.654.2 Member Typedef Documentation	1770
4.654.3 Constructor & Destructor Documentation	1771
4.654.4 Member Function Documentation	1772
4.655std::ostreambuf_iterator< _CharT, _Traits > Class Template Reference	1772
4.655.1 Detailed Description	1773
4.655.2 Member Typedef Documentation	1773
4.655.3 Constructor & Destructor Documentation	1775
4.655.4 Member Function Documentation	1775

4.656std::output_iterator_tag Struct Reference	1776
4.656.1 Detailed Description	1776
4.657std::owner_less< _Tp > Struct Template Reference	1776
4.657.1 Detailed Description	1776
4.658std::owner_less< shared_ptr< _Tp > > Struct Template Reference	1776
4.658.1 Detailed Description	1777
4.658.2 Member Typedef Documentation	1777
4.659std::owner_less< weak_ptr< _Tp > > Struct Template Reference	1778
4.659.1 Detailed Description	1778
4.659.2 Member Typedef Documentation	1778
4.660std::pair< _T1, _T2 > Struct Template Reference	1779
4.660.1 Detailed Description	1779
4.660.2 Member Typedef Documentation	1780
4.660.3 Constructor & Destructor Documentation	1780
4.660.4 Member Data Documentation	1781
4.661std::piecewise_constant_distribution< _RealType > Class Template Reference	1781
4.661.1 Detailed Description	1782
4.661.2 Member Typedef Documentation	1783
4.661.3 Member Function Documentation	1783
4.661.4 Friends And Related Function Documentation	1784
4.662std::piecewise_constant_distribution< _RealType >::param_type Struct Reference	1785
4.662.1 Detailed Description	1785
4.663std::piecewise_construct_t Struct Reference	1786
4.663.1 Detailed Description	1786
4.664std::piecewise_linear_distribution< _RealType > Class Template Reference	1786
4.664.1 Detailed Description	1787
4.664.2 Member Typedef Documentation	1787
4.664.3 Member Function Documentation	1787

4.664.4 Friends And Related Function Documentation	1789
4.665std::piecewise_linear_distribution< _RealType >::param_type Struct Reference	1790
4.665.1 Detailed Description	1790
4.666std::plus< _Tp > Struct Template Reference	1791
4.666.1 Detailed Description	1791
4.666.2 Member Typedef Documentation	1791
4.667std::pointer_to_binary_function< _Arg1, _Arg2, _Result > Class Template Reference	1792
4.667.1 Detailed Description	1793
4.667.2 Member Typedef Documentation	1793
4.668std::pointer_to_unary_function< _Arg, _Result > Class Template Reference	1794
4.668.1 Detailed Description	1794
4.668.2 Member Typedef Documentation	1795
4.669std::pointer_traits< _Ptr > Struct Template Reference	1795
4.669.1 Detailed Description	1795
4.669.2 Member Typedef Documentation	1796
4.670std::pointer_traits< _Tp * > Struct Template Reference	1796
4.670.1 Detailed Description	1796
4.670.2 Member Typedef Documentation	1797
4.670.3 Member Function Documentation	1797
4.671std::poisson_distribution< _IntType > Class Template Reference	1798
4.671.1 Detailed Description	1799
4.671.2 Member Typedef Documentation	1799
4.671.3 Member Function Documentation	1799
4.671.4 Friends And Related Function Documentation	1800
4.672std::poisson_distribution< _IntType >::param_type Struct Reference	1801
4.672.1 Detailed Description	1801
4.673std::priority_queue< _Tp, _Sequence, _Compare > Class Template Reference	1802
4.673.1 Detailed Description	1802

4.673.2 Constructor & Destructor Documentation	1803
4.673.3 Member Function Documentation	1804
4.674std::queue< _Tp, _Sequence > Class Template Reference	1805
4.674.1 Detailed Description	1806
4.674.2 Constructor & Destructor Documentation	1806
4.674.3 Member Function Documentation	1807
4.674.4 Member Data Documentation	1808
4.675std::random_access_iterator_tag Struct Reference	1809
4.675.1 Detailed Description	1809
4.676std::random_device Class Reference	1809
4.676.1 Detailed Description	1810
4.676.2 Member Typedef Documentation	1810
4.677std::raw_storage_iterator< _OutputIterator, _Tp > Class Template Reference	1810
4.677.1 Detailed Description	1811
4.677.2 Member Typedef Documentation	1811
4.678std::regex_error Class Reference	1812
4.678.1 Detailed Description	1812
4.678.2 Constructor & Destructor Documentation	1812
4.678.3 Member Function Documentation	1813
4.679std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits > Class Template Reference	1813
4.679.1 Detailed Description	1814
4.679.2 Constructor & Destructor Documentation	1814
4.679.3 Member Function Documentation	1815
4.680std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits > Class Template Reference	1816
4.680.1 Detailed Description	1817
4.680.2 Constructor & Destructor Documentation	1817
4.680.3 Member Function Documentation	1819
4.681std::regex_traits< _Ch_type > Struct Template Reference	1820

4.681.1 Detailed Description	1821
4.681.2 Constructor & Destructor Documentation	1821
4.681.3 Member Function Documentation	1821
4.682std::reverse_iterator< _Iterator > Class Template Reference	1826
4.682.1 Detailed Description	1827
4.682.2 Member Typedef Documentation	1828
4.682.3 Constructor & Destructor Documentation	1828
4.682.4 Member Function Documentation	1829
4.683std::seed_seq Class Reference	1832
4.683.1 Detailed Description	1832
4.683.2 Member Typedef Documentation	1832
4.683.3 Constructor & Destructor Documentation	1832
4.684std::set< _Key, _Compare, _Alloc > Class Template Reference	1833
4.684.1 Detailed Description	1834
4.684.2 Member Typedef Documentation	1835
4.684.3 Constructor & Destructor Documentation	1837
4.684.4 Member Function Documentation	1840
4.685std::shared_ptr< _Tp > Class Template Reference	1852
4.685.1 Detailed Description	1853
4.685.2 Constructor & Destructor Documentation	1854
4.685.3 Friends And Related Function Documentation	1859
4.686std::shuffle_order_engine< _RandomNumberEngine, __k > Class Template Reference	1860
4.686.1 Detailed Description	1861
4.686.2 Member Typedef Documentation	1861
4.686.3 Constructor & Destructor Documentation	1861
4.686.4 Member Function Documentation	1863
4.686.5 Friends And Related Function Documentation	1864
4.687std::slice Class Reference	1865

4.687.1 Detailed Description	1865
4.688std::slice_array<_Tp> Class Template Reference	1866
4.688.1 Detailed Description	1867
4.689std::stack<_Tp, _Sequence> Class Template Reference	1867
4.689.1 Detailed Description	1868
4.689.2 Constructor & Destructor Documentation	1868
4.689.3 Member Function Documentation	1869
4.690std::student_t_distribution<_RealType> Class Template Reference	1870
4.690.1 Detailed Description	1871
4.690.2 Member Typedef Documentation	1871
4.690.3 Member Function Documentation	1871
4.690.4 Friends And Related Function Documentation	1872
4.691std::student_t_distribution<_RealType>::param_type Struct Reference	1873
4.691.1 Detailed Description	1874
4.692std::sub_match<_Bilter> Class Template Reference	1874
4.692.1 Detailed Description	1875
4.692.2 Member Typedef Documentation	1875
4.692.3 Member Function Documentation	1875
4.692.4 Member Data Documentation	1877
4.693std::time_base Class Reference	1878
4.693.1 Detailed Description	1878
4.694std::time_get<_CharT, _InIter> Class Template Reference	1879
4.694.1 Detailed Description	1880
4.694.2 Member Typedef Documentation	1881
4.694.3 Constructor & Destructor Documentation	1881
4.694.4 Member Function Documentation	1881
4.694.5 Member Data Documentation	1887
4.695std::time_get_byname<_CharT, _InIter> Class Template Reference	1888

4.695.1 Detailed Description	1889
4.695.2 Member Function Documentation	1890
4.695.3 Member Data Documentation	1895
4.696std::time_put< _CharT, _OutIter > Class Template Reference	1896
4.696.1 Detailed Description	1897
4.696.2 Member Typedef Documentation	1897
4.696.3 Constructor & Destructor Documentation	1897
4.696.4 Member Function Documentation	1898
4.696.5 Member Data Documentation	1900
4.697std::time_put_byname< _CharT, _OutIter > Class Template Reference	1901
4.697.1 Detailed Description	1902
4.697.2 Member Function Documentation	1902
4.697.3 Member Data Documentation	1904
4.698std::unary_function< _Arg, _Result > Struct Template Reference	1904
4.698.1 Detailed Description	1904
4.698.2 Member Typedef Documentation	1905
4.699std::unary_negate< _Predicate > Class Template Reference	1905
4.699.1 Detailed Description	1906
4.699.2 Member Typedef Documentation	1906
4.700std::uniform_int_distribution< _IntType > Class Template Reference	1906
4.700.1 Detailed Description	1907
4.700.2 Member Typedef Documentation	1907
4.700.3 Constructor & Destructor Documentation	1908
4.700.4 Member Function Documentation	1908
4.700.5 Friends And Related Function Documentation	1909
4.701std::uniform_int_distribution< _IntType >::param_type Struct Reference	1909
4.701.1 Detailed Description	1910
4.702std::uniform_real_distribution< _RealType > Class Template Reference	1910

4.702.1 Detailed Description	1911
4.702.2 Member Typedef Documentation	1911
4.702.3 Constructor & Destructor Documentation	1911
4.702.4 Member Function Documentation	1911
4.702.5 Friends And Related Function Documentation	1913
4.703std::uniform_real_distribution< _RealType >::param_type Struct Reference	1913
4.703.1 Detailed Description	1913
4.704std::unique_ptr< _Tp, _Dp > Class Template Reference	1914
4.704.1 Detailed Description	1914
4.704.2 Constructor & Destructor Documentation	1915
4.704.3 Member Function Documentation	1917
4.705std::unique_ptr< _Tp[], _Dp > Class Template Reference	1919
4.705.1 Detailed Description	1920
4.705.2 Constructor & Destructor Documentation	1920
4.705.3 Member Function Documentation	1922
4.706std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > Class Template Reference	1924
4.706.1 Detailed Description	1926
4.706.2 Member Typedef Documentation	1927
4.706.3 Constructor & Destructor Documentation	1929
4.706.4 Member Function Documentation	1931
4.707std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > Class Template Reference	1948
4.707.1 Detailed Description	1950
4.707.2 Member Typedef Documentation	1950
4.707.3 Constructor & Destructor Documentation	1953
4.707.4 Member Function Documentation	1955
4.708std::unordered_multiset< _Value, _Hash, _Pred, _Alloc > Class Template Reference	1970
4.708.1 Detailed Description	1972
4.708.2 Member Typedef Documentation	1973

4.708.3 Constructor & Destructor Documentation	1975
4.708.4 Member Function Documentation	1977
4.709std::unordered_set< _Value, _Hash, _Pred, _Alloc > Class Template Reference	1993
4.709.1 Detailed Description	1995
4.709.2 Member Typedef Documentation	1996
4.709.3 Constructor & Destructor Documentation	1998
4.709.4 Member Function Documentation	2000
4.710std::uses_allocator< _Tp, _Alloc > Struct Template Reference	2015
4.710.1 Detailed Description	2015
4.711std::vector< _Tp, _Alloc > Class Template Reference	2015
4.711.1 Detailed Description	2018
4.711.2 Constructor & Destructor Documentation	2018
4.711.3 Member Function Documentation	2021
4.712std::vector< bool, _Alloc > Class Template Reference	2035
4.712.1 Detailed Description	2038
4.713std::weak_ptr< _Tp > Class Template Reference	2038
4.713.1 Detailed Description	2039
4.714std::weibull_distribution< _RealType > Class Template Reference	2039
4.714.1 Detailed Description	2040
4.714.2 Member Typedef Documentation	2040
4.714.3 Member Function Documentation	2041
4.714.4 Friends And Related Function Documentation	2042
4.715std::weibull_distribution< _RealType >::param_type Struct Reference	2042
4.715.1 Detailed Description	2043

5	File Documentation	2043
5.1	algo.h File Reference	2043
5.1.1	Detailed Description	2053
5.2	algbase.h File Reference	2053
5.2.1	Detailed Description	2054
5.3	algorithmfwd.h File Reference	2054
5.3.1	Detailed Description	2059
5.4	algorithmfwd.h File Reference	2059
5.4.1	Detailed Description	2068
5.5	aligned_buffer.h File Reference	2068
5.5.1	Detailed Description	2068
5.6	alloc_traits.h File Reference	2068
5.6.1	Detailed Description	2069
5.7	alloc_traits.h File Reference	2069
5.7.1	Detailed Description	2069
5.8	allocator.h File Reference	2070
5.8.1	Detailed Description	2070
5.9	array_allocator.h File Reference	2070
5.9.1	Detailed Description	2071
5.10	assoc_container.hpp File Reference	2071
5.10.1	Detailed Description	2071
5.11	atomic_base.h File Reference	2071
5.11.1	Detailed Description	2073
5.12	atomic_lockfree_defines.h File Reference	2073
5.12.1	Detailed Description	2073
5.13	atomic_word.h File Reference	2074
5.13.1	Detailed Description	2074
5.14	atomicity.h File Reference	2074

5.14.1 Detailed Description	2074
5.15 auto_ptr.h File Reference	2074
5.15.1 Detailed Description	2075
5.16 backward_warning.h File Reference	2075
5.16.1 Detailed Description	2075
5.17 balanced_quicksort.h File Reference	2075
5.17.1 Detailed Description	2076
5.18 base.h File Reference	2076
5.18.1 Detailed Description	2076
5.19 base.h File Reference	2076
5.19.1 Detailed Description	2077
5.20 basic_file.h File Reference	2077
5.20.1 Detailed Description	2077
5.21 basic_ios.h File Reference	2077
5.21.1 Detailed Description	2078
5.22 basic_iterator.h File Reference	2078
5.22.1 Detailed Description	2078
5.23 basic_string.h File Reference	2078
5.23.1 Detailed Description	2080
5.24 bin_search_tree_.hpp File Reference	2081
5.24.1 Detailed Description	2081
5.25 binary_heap_.hpp File Reference	2081
5.25.1 Detailed Description	2081
5.26 binders.h File Reference	2082
5.26.1 Detailed Description	2082
5.27 binomial_heap_.hpp File Reference	2082
5.27.1 Detailed Description	2082
5.28 binomial_heap_base_.hpp File Reference	2083

5.28.1 Detailed Description	2083
5.29 bitmap_allocator.h File Reference	2083
5.29.1 Detailed Description	2084
5.29.2 Macro Definition Documentation	2084
5.30 boost_concept_check.h File Reference	2084
5.30.1 Detailed Description	2085
5.31 branch_policy.hpp File Reference	2085
5.31.1 Detailed Description	2085
5.32 c++0x_warning.h File Reference	2085
5.32.1 Detailed Description	2085
5.33 c++14_warning.h File Reference	2086
5.33.1 Detailed Description	2086
5.34 c++allocator.h File Reference	2086
5.34.1 Detailed Description	2086
5.35 c++config.h File Reference	2086
5.35.1 Detailed Description	2091
5.36 c++io.h File Reference	2091
5.36.1 Detailed Description	2091
5.37 c++locale.h File Reference	2091
5.37.1 Detailed Description	2092
5.38 c++locale_internal.h File Reference	2092
5.38.1 Detailed Description	2092
5.39 cast.h File Reference	2092
5.39.1 Detailed Description	2093
5.40 cc_hash_max_collision_check_resize_trigger_imp.hpp File Reference	2093
5.40.1 Detailed Description	2093
5.41 cc_ht_map_.hpp File Reference	2093
5.41.1 Detailed Description	2093

5.42	char_traits.h File Reference	2093
5.42.1	Detailed Description	2094
5.43	checkers.h File Reference	2094
5.43.1	Detailed Description	2094
5.44	cmp_fn_imps.hpp File Reference	2094
5.44.1	Detailed Description	2094
5.45	codecvt.h File Reference	2094
5.45.1	Detailed Description	2095
5.46	codecvt_specializations.h File Reference	2095
5.46.1	Detailed Description	2095
5.47	compatibility.h File Reference	2095
5.47.1	Detailed Description	2095
5.48	compatibility.h File Reference	2095
5.48.1	Detailed Description	2096
5.49	compiletime_settings.h File Reference	2096
5.49.1	Detailed Description	2096
5.49.2	Macro Definition Documentation	2096
5.50	complex.h File Reference	2098
5.50.1	Detailed Description	2098
5.51	concept_check.h File Reference	2098
5.51.1	Detailed Description	2098
5.52	concurrency.h File Reference	2098
5.52.1	Detailed Description	2099
5.53	cond_dealtor.hpp File Reference	2099
5.53.1	Detailed Description	2099
5.54	cond_key_dtor_entry_dealtor.hpp File Reference	2099
5.54.1	Detailed Description	2099
5.55	const_iterator.hpp File Reference	2099

5.55.1 Detailed Description	2100
5.56 const_iterator.hpp File Reference	2100
5.56.1 Detailed Description	2100
5.57 const_iterator.hpp File Reference	2100
5.57.1 Detailed Description	2100
5.58 constructor_destructor_fn_imps.hpp File Reference	2101
5.58.1 Detailed Description	2101
5.59 constructor_destructor_fn_imps.hpp File Reference	2101
5.59.1 Detailed Description	2101
5.60 constructor_destructor_fn_imps.hpp File Reference	2101
5.61 constructor_destructor_no_store_hash_fn_imps.hpp File Reference	2101
5.61.1 Detailed Description	2101
5.62 constructor_destructor_no_store_hash_fn_imps.hpp File Reference	2101
5.62.1 Detailed Description	2101
5.63 constructor_destructor_store_hash_fn_imps.hpp File Reference	2101
5.63.1 Detailed Description	2101
5.64 constructor_destructor_store_hash_fn_imps.hpp File Reference	2101
5.64.1 Detailed Description	2101
5.65 constructors_destructor_fn_imps.hpp File Reference	2102
5.65.1 Detailed Description	2102
5.66 constructors_destructor_fn_imps.hpp File Reference	2102
5.66.1 Detailed Description	2102
5.67 constructors_destructor_fn_imps.hpp File Reference	2102
5.67.1 Detailed Description	2102
5.68 constructors_destructor_fn_imps.hpp File Reference	2102
5.68.1 Detailed Description	2102
5.69 constructors_destructor_fn_imps.hpp File Reference	2102
5.69.1 Detailed Description	2102

5.70	constructors_destructor_fn_impls.hpp File Reference	2102
5.70.1	Detailed Description	2102
5.71	constructors_destructor_fn_impls.hpp File Reference	2103
5.71.1	Detailed Description	2103
5.72	constructors_destructor_fn_impls.hpp File Reference	2103
5.72.1	Detailed Description	2103
5.73	constructors_destructor_fn_impls.hpp File Reference	2103
5.73.1	Detailed Description	2103
5.74	constructors_destructor_fn_impls.hpp File Reference	2103
5.74.1	Detailed Description	2103
5.75	constructors_destructor_fn_impls.hpp File Reference	2103
5.75.1	Detailed Description	2103
5.76	constructors_destructor_fn_impls.hpp File Reference	2103
5.76.1	Detailed Description	2103
5.77	container_base_dispatch.hpp File Reference	2104
5.77.1	Detailed Description	2104
5.78	cpp_type_traits.h File Reference	2104
5.78.1	Detailed Description	2105
5.79	cpu_defines.h File Reference	2105
5.79.1	Detailed Description	2105
5.80	ctype_base.h File Reference	2105
5.80.1	Detailed Description	2105
5.81	ctype_inline.h File Reference	2105
5.81.1	Detailed Description	2105
5.82	cxxabi.h File Reference	2106
5.82.1	Detailed Description	2107
5.82.2	Function Documentation	2107
5.83	cxxabi_forced.h File Reference	2108

5.83.1 Detailed Description	2108
5.84 cxxabi_tweaks.h File Reference	2108
5.84.1 Detailed Description	2109
5.85 debug.h File Reference	2109
5.85.1 Detailed Description	2109
5.86 debug_allocator.h File Reference	2109
5.86.1 Detailed Description	2110
5.87 debug_fn_imps.hpp File Reference	2110
5.87.1 Detailed Description	2110
5.88 debug_fn_imps.hpp File Reference	2110
5.88.1 Detailed Description	2110
5.89 debug_fn_imps.hpp File Reference	2110
5.89.1 Detailed Description	2110
5.90 debug_fn_imps.hpp File Reference	2110
5.90.1 Detailed Description	2110
5.91 debug_fn_imps.hpp File Reference	2111
5.91.1 Detailed Description	2111
5.92 debug_fn_imps.hpp File Reference	2111
5.92.1 Detailed Description	2111
5.93 debug_fn_imps.hpp File Reference	2111
5.93.1 Detailed Description	2111
5.94 debug_fn_imps.hpp File Reference	2111
5.94.1 Detailed Description	2111
5.95 debug_fn_imps.hpp File Reference	2111
5.95.1 Detailed Description	2111
5.96 debug_fn_imps.hpp File Reference	2111
5.96.1 Detailed Description	2111
5.97 debug_fn_imps.hpp File Reference	2112

5.97.1 Detailed Description	2112
5.98 debug_fn_imps.hpp File Reference	2112
5.98.1 Detailed Description	2112
5.99 debug_fn_imps.hpp File Reference	2112
5.99.1 Detailed Description	2112
5.100debug_fn_imps.hpp File Reference	2112
5.100.1 Detailed Description	2112
5.101debug_fn_imps.hpp File Reference	2112
5.101.1 Detailed Description	2112
5.102debug_map_base.hpp File Reference	2112
5.102.1 Detailed Description	2112
5.103debug_no_store_hash_fn_imps.hpp File Reference	2113
5.103.1 Detailed Description	2113
5.104debug_no_store_hash_fn_imps.hpp File Reference	2113
5.104.1 Detailed Description	2113
5.105debug_store_hash_fn_imps.hpp File Reference	2113
5.105.1 Detailed Description	2113
5.106debug_store_hash_fn_imps.hpp File Reference	2113
5.106.1 Detailed Description	2113
5.107direct_mask_range_hashing_imp.hpp File Reference	2113
5.107.1 Detailed Description	2113
5.108direct_mod_range_hashing_imp.hpp File Reference	2113
5.108.1 Detailed Description	2113
5.109enable_special_members.h File Reference	2114
5.109.1 Detailed Description	2114
5.110enc_filebuf.h File Reference	2114
5.110.1 Detailed Description	2114
5.111entry_cmp.hpp File Reference	2114

5.111.1 Detailed Description	2115
5.112entry_list_fn_imps.hpp File Reference	2115
5.112.1 Detailed Description	2115
5.113entry_metadata_base.hpp File Reference	2115
5.113.1 Detailed Description	2115
5.114entry_pred.hpp File Reference	2115
5.114.1 Detailed Description	2115
5.115eq_by_less.hpp File Reference	2116
5.115.1 Detailed Description	2116
5.116equally_split.h File Reference	2116
5.116.1 Detailed Description	2116
5.117erase_fn_imps.hpp File Reference	2116
5.117.1 Detailed Description	2116
5.118erase_fn_imps.hpp File Reference	2117
5.118.1 Detailed Description	2117
5.119erase_fn_imps.hpp File Reference	2117
5.119.1 Detailed Description	2117
5.120erase_fn_imps.hpp File Reference	2117
5.120.1 Detailed Description	2117
5.121erase_fn_imps.hpp File Reference	2117
5.121.1 Detailed Description	2117
5.122erase_fn_imps.hpp File Reference	2117
5.122.1 Detailed Description	2117
5.123erase_fn_imps.hpp File Reference	2117
5.123.1 Detailed Description	2117
5.124erase_fn_imps.hpp File Reference	2118
5.124.1 Detailed Description	2118
5.125erase_fn_imps.hpp File Reference	2118

5.125.1 Detailed Description	2118
5.126erase_fn_imps.hpp File Reference	2118
5.126.1 Detailed Description	2118
5.127erase_fn_imps.hpp File Reference	2118
5.127.1 Detailed Description	2118
5.128erase_fn_imps.hpp File Reference	2118
5.128.1 Detailed Description	2118
5.129erase_fn_imps.hpp File Reference	2118
5.129.1 Detailed Description	2118
5.130erase_fn_imps.hpp File Reference	2119
5.130.1 Detailed Description	2119
5.131erase_no_store_hash_fn_imps.hpp File Reference	2119
5.131.1 Detailed Description	2119
5.132erase_no_store_hash_fn_imps.hpp File Reference	2119
5.132.1 Detailed Description	2119
5.133erase_store_hash_fn_imps.hpp File Reference	2119
5.133.1 Detailed Description	2119
5.134erase_store_hash_fn_imps.hpp File Reference	2119
5.134.1 Detailed Description	2119
5.135error_constants.h File Reference	2119
5.135.1 Detailed Description	2120
5.136exception.hpp File Reference	2120
5.136.1 Detailed Description	2121
5.137exception_defines.h File Reference	2121
5.137.1 Detailed Description	2121
5.138exception_ptr.h File Reference	2121
5.138.1 Detailed Description	2121
5.139extc++.h File Reference	2122

5.139.1 Detailed Description	2122
5.140extptr_allocator.h File Reference	2122
5.140.1 Detailed Description	2122
5.141features.h File Reference	2122
5.141.1 Detailed Description	2123
5.141.2 Macro Definition Documentation	2123
5.142fenv.h File Reference	2125
5.142.1 Detailed Description	2125
5.143find.h File Reference	2125
5.143.1 Detailed Description	2125
5.144find_fn_imps.hpp File Reference	2125
5.144.1 Detailed Description	2125
5.145find_fn_imps.hpp File Reference	2125
5.145.1 Detailed Description	2125
5.146find_fn_imps.hpp File Reference	2126
5.146.1 Detailed Description	2126
5.147find_fn_imps.hpp File Reference	2126
5.147.1 Detailed Description	2126
5.148find_fn_imps.hpp File Reference	2126
5.148.1 Detailed Description	2126
5.149find_fn_imps.hpp File Reference	2126
5.149.1 Detailed Description	2126
5.150find_fn_imps.hpp File Reference	2126
5.150.1 Detailed Description	2126
5.151find_fn_imps.hpp File Reference	2126
5.151.1 Detailed Description	2126
5.152find_fn_imps.hpp File Reference	2127
5.152.1 Detailed Description	2127

5.153find_fn_imps.hpp File Reference	2127
5.153.1 Detailed Description	2127
5.154find_fn_imps.hpp File Reference	2127
5.154.1 Detailed Description	2127
5.155find_no_store_hash_fn_imps.hpp File Reference	2127
5.155.1 Detailed Description	2127
5.156find_selectors.h File Reference	2127
5.156.1 Detailed Description	2127
5.157find_store_hash_fn_imps.hpp File Reference	2128
5.157.1 Detailed Description	2128
5.158find_store_hash_fn_imps.hpp File Reference	2128
5.158.1 Detailed Description	2128
5.159for_each.h File Reference	2128
5.159.1 Detailed Description	2128
5.160for_each_selectors.h File Reference	2129
5.160.1 Detailed Description	2129
5.161formatter.h File Reference	2129
5.161.1 Detailed Description	2130
5.162forward_list.h File Reference	2130
5.162.1 Detailed Description	2131
5.163functexcept.h File Reference	2131
5.163.1 Detailed Description	2132
5.164functional_hash.h File Reference	2132
5.164.1 Detailed Description	2133
5.165functions.h File Reference	2133
5.165.1 Detailed Description	2135
5.166gp_ht_map.hpp File Reference	2135
5.166.1 Detailed Description	2136

5.167gslice.h File Reference	2136
5.167.1 Detailed Description	2136
5.168gslice_array.h File Reference	2136
5.168.1 Detailed Description	2137
5.169hash_bytes.h File Reference	2137
5.169.1 Detailed Description	2137
5.170hash_eq_fn.hpp File Reference	2137
5.170.1 Detailed Description	2138
5.171hash_exponential_size_policy_imp.hpp File Reference	2138
5.171.1 Detailed Description	2138
5.172hash_fun.h File Reference	2138
5.172.1 Detailed Description	2138
5.173hash_load_check_resize_trigger_imp.hpp File Reference	2138
5.173.1 Detailed Description	2138
5.174hash_load_check_resize_trigger_size_base.hpp File Reference	2138
5.174.1 Detailed Description	2139
5.175hash_policy.hpp File Reference	2139
5.175.1 Detailed Description	2140
5.176hash_prime_size_policy_imp.hpp File Reference	2140
5.176.1 Detailed Description	2140
5.177hash_standard_resize_policy_imp.hpp File Reference	2140
5.177.1 Detailed Description	2140
5.178hashtable.h File Reference	2140
5.178.1 Detailed Description	2141
5.179hashtable.h File Reference	2141
5.179.1 Detailed Description	2141
5.180hashtable_policy.h File Reference	2142
5.180.1 Detailed Description	2144

5.181indirect_array.h File Reference	2144
5.181.1 Detailed Description	2144
5.182info_fn_imps.hpp File Reference	2144
5.182.1 Detailed Description	2144
5.183info_fn_imps.hpp File Reference	2144
5.183.1 Detailed Description	2144
5.184info_fn_imps.hpp File Reference	2144
5.184.1 Detailed Description	2144
5.185info_fn_imps.hpp File Reference	2145
5.185.1 Detailed Description	2145
5.186info_fn_imps.hpp File Reference	2145
5.186.1 Detailed Description	2145
5.187info_fn_imps.hpp File Reference	2145
5.187.1 Detailed Description	2145
5.188info_fn_imps.hpp File Reference	2145
5.188.1 Detailed Description	2145
5.189info_fn_imps.hpp File Reference	2145
5.189.1 Detailed Description	2145
5.190info_fn_imps.hpp File Reference	2145
5.190.1 Detailed Description	2145
5.191info_fn_imps.hpp File Reference	2146
5.191.1 Detailed Description	2146
5.192insert_fn_imps.hpp File Reference	2146
5.192.1 Detailed Description	2146
5.193insert_fn_imps.hpp File Reference	2146
5.193.1 Detailed Description	2146
5.194insert_fn_imps.hpp File Reference	2146
5.194.1 Detailed Description	2146

5.195insert_fn_imps.hpp File Reference	2146
5.195.1 Detailed Description	2146
5.196insert_fn_imps.hpp File Reference	2146
5.196.1 Detailed Description	2146
5.197insert_fn_imps.hpp File Reference	2147
5.197.1 Detailed Description	2147
5.198insert_fn_imps.hpp File Reference	2147
5.198.1 Detailed Description	2147
5.199insert_fn_imps.hpp File Reference	2147
5.199.1 Detailed Description	2147
5.200insert_fn_imps.hpp File Reference	2147
5.200.1 Detailed Description	2147
5.201insert_fn_imps.hpp File Reference	2147
5.201.1 Detailed Description	2147
5.202insert_fn_imps.hpp File Reference	2147
5.202.1 Detailed Description	2147
5.203insert_fn_imps.hpp File Reference	2148
5.203.1 Detailed Description	2148
5.204insert_fn_imps.hpp File Reference	2148
5.204.1 Detailed Description	2148
5.205insert_join_fn_imps.hpp File Reference	2148
5.205.1 Detailed Description	2148
5.206insert_no_store_hash_fn_imps.hpp File Reference	2148
5.206.1 Detailed Description	2148
5.207insert_no_store_hash_fn_imps.hpp File Reference	2148
5.207.1 Detailed Description	2148
5.208insert_store_hash_fn_imps.hpp File Reference	2148
5.208.1 Detailed Description	2148

5.209	insert_store_hash_fn_imps.hpp File Reference	2149
5.209.1	Detailed Description	2149
5.210	ios_base.h File Reference	2149
5.210.1	Detailed Description	2150
5.211	iterator.h File Reference	2150
5.211.1	Detailed Description	2151
5.212	iterator.hpp File Reference	2151
5.212.1	Detailed Description	2151
5.213	iterator_fn_imps.hpp File Reference	2151
5.213.1	Detailed Description	2151
5.214	iterator_tracker.h File Reference	2151
5.214.1	Detailed Description	2152
5.215	iterators_fn_imps.hpp File Reference	2152
5.215.1	Detailed Description	2152
5.216	iterators_fn_imps.hpp File Reference	2152
5.216.1	Detailed Description	2152
5.217	iterators_fn_imps.hpp File Reference	2153
5.217.1	Detailed Description	2153
5.218	iterators_fn_imps.hpp File Reference	2153
5.218.1	Detailed Description	2153
5.219	iterators_fn_imps.hpp File Reference	2153
5.219.1	Detailed Description	2153
5.220	iterators_fn_imps.hpp File Reference	2153
5.220.1	Detailed Description	2153
5.221	iterators_fn_imps.hpp File Reference	2153
5.221.1	Detailed Description	2153
5.222	left_child_next_sibling_heap.hpp File Reference	2153
5.222.1	Detailed Description	2154

5.223	linear_probe_fn_imp.hpp File Reference	2154
5.223.1	Detailed Description	2154
5.224	list_partition.h File Reference	2154
5.224.1	Detailed Description	2154
5.225	list_update_policy.hpp File Reference	2154
5.225.1	Detailed Description	2155
5.226	locale_classes.h File Reference	2155
5.226.1	Detailed Description	2155
5.227	locale_facets.h File Reference	2155
5.227.1	Detailed Description	2157
5.228	locale_facets_nonio.h File Reference	2157
5.228.1	Detailed Description	2157
5.229	localefwd.h File Reference	2157
5.229.1	Detailed Description	2159
5.230	losertree.h File Reference	2159
5.230.1	Detailed Description	2159
5.231	lu_counter_metadata.hpp File Reference	2159
5.231.1	Detailed Description	2160
5.232	lu_map_.hpp File Reference	2160
5.232.1	Detailed Description	2160
5.233	macros.h File Reference	2160
5.233.1	Detailed Description	2161
5.233.2	Macro Definition Documentation	2161
5.234	malloc_allocator.h File Reference	2163
5.234.1	Detailed Description	2164
5.235	map.h File Reference	2164
5.235.1	Detailed Description	2164
5.236	map.h File Reference	2164

5.236.1 Detailed Description	2165
5.237mask_array.h File Reference	2165
5.237.1 Detailed Description	2166
5.238mask_based_range_hashing.hpp File Reference	2166
5.238.1 Detailed Description	2166
5.239memoryfwd.h File Reference	2166
5.239.1 Detailed Description	2166
5.240merge.h File Reference	2166
5.240.1 Detailed Description	2167
5.241messages_members.h File Reference	2167
5.241.1 Detailed Description	2167
5.242mod_based_range_hashing.hpp File Reference	2167
5.242.1 Detailed Description	2168
5.243move.h File Reference	2168
5.243.1 Detailed Description	2168
5.244mt_allocator.h File Reference	2169
5.244.1 Detailed Description	2169
5.245multimap.h File Reference	2169
5.245.1 Detailed Description	2170
5.246multimap.h File Reference	2170
5.246.1 Detailed Description	2171
5.247multiseg_selection.h File Reference	2171
5.247.1 Detailed Description	2172
5.248multiset.h File Reference	2172
5.248.1 Detailed Description	2173
5.249multiset.h File Reference	2173
5.249.1 Detailed Description	2173
5.250multiway_merge.h File Reference	2174

5.250.1 Detailed Description	2176
5.250.2 Macro Definition Documentation	2177
5.251multiway_mergesort.h File Reference	2177
5.251.1 Detailed Description	2177
5.252nested_exception.h File Reference	2177
5.252.1 Detailed Description	2178
5.253new_allocator.h File Reference	2178
5.253.1 Detailed Description	2178
5.254node.hpp File Reference	2179
5.254.1 Detailed Description	2179
5.255node.hpp File Reference	2179
5.255.1 Detailed Description	2179
5.256node.hpp File Reference	2179
5.256.1 Detailed Description	2179
5.257node_iterators.hpp File Reference	2180
5.257.1 Detailed Description	2180
5.258node_iterators.hpp File Reference	2180
5.258.1 Detailed Description	2180
5.259node_metadata_selector.hpp File Reference	2181
5.259.1 Detailed Description	2181
5.260node_metadata_selector.hpp File Reference	2181
5.260.1 Detailed Description	2181
5.261null_node_metadata.hpp File Reference	2181
5.261.1 Detailed Description	2182
5.262numeric_traits.h File Reference	2182
5.262.1 Detailed Description	2182
5.263numeric_fwd.h File Reference	2182
5.263.1 Detailed Description	2184

5.264omp_loop.h File Reference	2184
5.264.1 Detailed Description	2184
5.265omp_loop_static.h File Reference	2185
5.265.1 Detailed Description	2185
5.266opt_random.h File Reference	2185
5.266.1 Detailed Description	2185
5.267order_statistics_imp.hpp File Reference	2185
5.267.1 Detailed Description	2185
5.268order_statistics_imp.hpp File Reference	2185
5.268.1 Detailed Description	2185
5.269os_defines.h File Reference	2186
5.269.1 Detailed Description	2186
5.270ostream_insert.h File Reference	2186
5.270.1 Detailed Description	2186
5.271ov_tree_map_.hpp File Reference	2186
5.271.1 Detailed Description	2187
5.272pairing_heap_.hpp File Reference	2187
5.272.1 Detailed Description	2187
5.273par_loop.h File Reference	2187
5.273.1 Detailed Description	2188
5.274parallel.h File Reference	2188
5.274.1 Detailed Description	2188
5.275parse_numbers.h File Reference	2188
5.275.1 Detailed Description	2188
5.276partial_sum.h File Reference	2188
5.276.1 Detailed Description	2188
5.277partition.h File Reference	2189
5.277.1 Detailed Description	2189

5.277.2 Macro Definition Documentation	2189
5.278pat_trie_.hpp File Reference	2189
5.278.1 Detailed Description	2190
5.279pat_trie_base.hpp File Reference	2190
5.279.1 Detailed Description	2191
5.280pod_char_traits.h File Reference	2191
5.280.1 Detailed Description	2191
5.281point_const_iterator.hpp File Reference	2191
5.281.1 Detailed Description	2191
5.282point_const_iterator.hpp File Reference	2192
5.282.1 Detailed Description	2192
5.283point_const_iterator.hpp File Reference	2192
5.283.1 Detailed Description	2192
5.284point_iterator.hpp File Reference	2192
5.284.1 Detailed Description	2192
5.285point_iterators.hpp File Reference	2193
5.285.1 Detailed Description	2193
5.286pointer.h File Reference	2193
5.286.1 Detailed Description	2195
5.287policy_access_fn_imps.hpp File Reference	2195
5.287.1 Detailed Description	2195
5.288policy_access_fn_imps.hpp File Reference	2195
5.288.1 Detailed Description	2195
5.289policy_access_fn_imps.hpp File Reference	2195
5.289.1 Detailed Description	2195
5.290policy_access_fn_imps.hpp File Reference	2196
5.290.1 Detailed Description	2196
5.291policy_access_fn_imps.hpp File Reference	2196

5.291.1 Detailed Description	2196
5.292policy_access_fn_imps.hpp File Reference	2196
5.292.1 Detailed Description	2196
5.293policy_access_fn_imps.hpp File Reference	2196
5.293.1 Detailed Description	2196
5.294pool_allocator.h File Reference	2196
5.294.1 Detailed Description	2197
5.295postypes.h File Reference	2197
5.295.1 Detailed Description	2197
5.296predefined_ops.h File Reference	2197
5.296.1 Detailed Description	2198
5.297prefix_search_node_update_imp.hpp File Reference	2198
5.297.1 Detailed Description	2198
5.298priority_queue.hpp File Reference	2199
5.298.1 Detailed Description	2199
5.299priority_queue_base_dispatch.hpp File Reference	2199
5.299.1 Detailed Description	2199
5.300probe_fn_base.hpp File Reference	2199
5.300.1 Detailed Description	2200
5.301profiler.h File Reference	2200
5.301.1 Detailed Description	2202
5.302profiler_algos.h File Reference	2202
5.302.1 Detailed Description	2203
5.303profiler_container_size.h File Reference	2203
5.303.1 Detailed Description	2203
5.304profiler_hash_func.h File Reference	2203
5.304.1 Detailed Description	2204
5.305profiler_hashtable_size.h File Reference	2204

5.305.1 Detailed Description	2204
5.306profiler_list_to_slist.h File Reference	2204
5.306.1 Detailed Description	2205
5.307profiler_list_to_vector.h File Reference	2205
5.307.1 Detailed Description	2205
5.308profiler_map_to_unordered_map.h File Reference	2205
5.308.1 Detailed Description	2206
5.309profiler_node.h File Reference	2206
5.309.1 Detailed Description	2207
5.310profiler_state.h File Reference	2207
5.310.1 Detailed Description	2207
5.311profiler_trace.h File Reference	2207
5.311.1 Detailed Description	2209
5.312profiler_vector_size.h File Reference	2209
5.312.1 Detailed Description	2210
5.313profiler_vector_to_list.h File Reference	2210
5.313.1 Detailed Description	2210
5.314ptr_traits.h File Reference	2210
5.314.1 Detailed Description	2211
5.315quadratic_probe_fn_imp.hpp File Reference	2211
5.315.1 Detailed Description	2211
5.316queue.h File Reference	2211
5.316.1 Detailed Description	2211
5.316.2 Macro Definition Documentation	2211
5.317quicksort.h File Reference	2212
5.317.1 Detailed Description	2212
5.318r_erase_fnimps.hpp File Reference	2212
5.318.1 Detailed Description	2212

5.319r_erase_fn_imps.hpp File Reference	2212
5.319.1 Detailed Description	2212
5.320random.h File Reference	2213
5.320.1 Detailed Description	2217
5.321random_number.h File Reference	2217
5.321.1 Detailed Description	2217
5.322random_shuffle.h File Reference	2217
5.322.1 Detailed Description	2218
5.323range_access.h File Reference	2218
5.323.1 Detailed Description	2218
5.324ranged_hash_fn.hpp File Reference	2219
5.324.1 Detailed Description	2219
5.325ranged_probe_fn.hpp File Reference	2219
5.325.1 Detailed Description	2220
5.326rb_tree_.hpp File Reference	2220
5.326.1 Detailed Description	2220
5.327rc.hpp File Reference	2220
5.327.1 Detailed Description	2221
5.328rc_binomial_heap_.hpp File Reference	2221
5.328.1 Detailed Description	2221
5.329rc_string_base.h File Reference	2221
5.329.1 Detailed Description	2221
5.330regex.h File Reference	2222
5.330.1 Detailed Description	2227
5.331regex_automaton.h File Reference	2227
5.331.1 Detailed Description	2227
5.332regex_compiler.h File Reference	2227
5.332.1 Detailed Description	2228

5.333regex_constants.h File Reference	2228
5.333.1 Detailed Description	2229
5.334regex_error.h File Reference	2229
5.334.1 Detailed Description	2230
5.335regex_executor.h File Reference	2230
5.335.1 Detailed Description	2230
5.336regex_scanner.h File Reference	2231
5.336.1 Detailed Description	2231
5.337resize_fn_imps.hpp File Reference	2231
5.337.1 Detailed Description	2231
5.338resize_fn_imps.hpp File Reference	2231
5.338.1 Detailed Description	2231
5.339resize_no_store_hash_fn_imps.hpp File Reference	2231
5.339.1 Detailed Description	2231
5.340resize_no_store_hash_fn_imps.hpp File Reference	2231
5.340.1 Detailed Description	2231
5.341resize_policy.hpp File Reference	2232
5.341.1 Detailed Description	2232
5.342resize_store_hash_fn_imps.hpp File Reference	2232
5.342.1 Detailed Description	2232
5.343resize_store_hash_fn_imps.hpp File Reference	2232
5.343.1 Detailed Description	2232
5.344ropeimpl.h File Reference	2232
5.344.1 Detailed Description	2233
5.345rotate_fn_imps.hpp File Reference	2233
5.345.1 Detailed Description	2233
5.346rotate_fn_imps.hpp File Reference	2233
5.346.1 Detailed Description	2233

5.347safe_base.h File Reference	2233
5.347.1 Detailed Description	2233
5.348safe_iterator.h File Reference	2233
5.348.1 Detailed Description	2235
5.349safe_local_iterator.h File Reference	2235
5.349.1 Detailed Description	2236
5.350safe_sequence.h File Reference	2236
5.350.1 Detailed Description	2236
5.351safe_unordered_base.h File Reference	2236
5.351.1 Detailed Description	2236
5.352safe_unordered_container.h File Reference	2236
5.352.1 Detailed Description	2237
5.353sample_probe_fn.hpp File Reference	2237
5.353.1 Detailed Description	2237
5.354sample_range_hashing.hpp File Reference	2237
5.354.1 Detailed Description	2237
5.355sample_ranged_hash_fn.hpp File Reference	2237
5.355.1 Detailed Description	2238
5.356sample_ranged_probe_fn.hpp File Reference	2238
5.356.1 Detailed Description	2238
5.357sample_resize_policy.hpp File Reference	2238
5.357.1 Detailed Description	2238
5.358sample_resize_trigger.hpp File Reference	2238
5.358.1 Detailed Description	2239
5.359sample_size_policy.hpp File Reference	2239
5.359.1 Detailed Description	2239
5.360sample_tree_node_update.hpp File Reference	2239
5.360.1 Detailed Description	2239

5.361sample_trie_access_traits.hpp File Reference	2239
5.361.1 Detailed Description	2240
5.362sample_trie_node_update.hpp File Reference	2240
5.362.1 Detailed Description	2240
5.363sample_update_policy.hpp File Reference	2240
5.363.1 Detailed Description	2240
5.364search.h File Reference	2240
5.364.1 Detailed Description	2241
5.365set.h File Reference	2241
5.365.1 Detailed Description	2242
5.366set.h File Reference	2242
5.366.1 Detailed Description	2242
5.367set_operations.h File Reference	2242
5.367.1 Detailed Description	2243
5.368settings.h File Reference	2243
5.368.1 Detailed Description	2243
5.368.2 parallelization_decision	2244
5.368.3 Macro Definition Documentation	2244
5.369shared_ptr.h File Reference	2245
5.369.1 Detailed Description	2246
5.370shared_ptr_base.h File Reference	2246
5.370.1 Detailed Description	2248
5.371size_fn_imps.hpp File Reference	2248
5.371.1 Detailed Description	2248
5.372slice_array.h File Reference	2248
5.372.1 Detailed Description	2249
5.373sort.h File Reference	2249
5.373.1 Detailed Description	2249

5.374splay_fn_imps.hpp File Reference	2249
5.374.1 Detailed Description	2249
5.375splay_tree_.hpp File Reference	2250
5.375.1 Detailed Description	2250
5.376split_fn_imps.hpp File Reference	2250
5.376.1 Detailed Description	2250
5.377split_join_fn_imps.hpp File Reference	2250
5.377.1 Detailed Description	2250
5.378split_join_fn_imps.hpp File Reference	2250
5.378.1 Detailed Description	2250
5.379split_join_fn_imps.hpp File Reference	2251
5.379.1 Detailed Description	2251
5.380split_join_fn_imps.hpp File Reference	2251
5.380.1 Detailed Description	2251
5.381split_join_fn_imps.hpp File Reference	2251
5.381.1 Detailed Description	2251
5.382split_join_fn_imps.hpp File Reference	2251
5.382.1 Detailed Description	2251
5.383split_join_fn_imps.hpp File Reference	2251
5.383.1 Detailed Description	2251
5.384split_join_fn_imps.hpp File Reference	2251
5.384.1 Detailed Description	2251
5.385split_join_fn_imps.hpp File Reference	2252
5.385.1 Detailed Description	2252
5.386sso_string_base.h File Reference	2252
5.386.1 Detailed Description	2252
5.387standard_policies.hpp File Reference	2252
5.387.1 Detailed Description	2253

5.387.2 Enumeration Type Documentation	2253
5.388stdc++.h File Reference	2253
5.388.1 Detailed Description	2253
5.389stdio_filebuf.h File Reference	2253
5.389.1 Detailed Description	2253
5.390stdio_sync_filebuf.h File Reference	2253
5.390.1 Detailed Description	2254
5.391stdtr1c++.h File Reference	2254
5.391.1 Detailed Description	2254
5.392stl_algo.h File Reference	2254
5.392.1 Detailed Description	2263
5.393stl_algobase.h File Reference	2263
5.393.1 Detailed Description	2266
5.394stl_bvector.h File Reference	2266
5.394.1 Detailed Description	2266
5.395stl_construct.h File Reference	2266
5.395.1 Detailed Description	2267
5.396stl_deque.h File Reference	2267
5.396.1 Detailed Description	2269
5.396.2 Macro Definition Documentation	2269
5.397stl_function.h File Reference	2270
5.397.1 Detailed Description	2271
5.398stl_heap.h File Reference	2271
5.398.1 Detailed Description	2272
5.399stl_iterator.h File Reference	2272
5.399.1 Detailed Description	2275
5.400stl_iterator_base_funcs.h File Reference	2275
5.400.1 Detailed Description	2276

5.401stl_iterator_base_types.h File Reference	2276
5.401.1 Detailed Description	2277
5.402stl_list.h File Reference	2277
5.402.1 Detailed Description	2278
5.403stl_map.h File Reference	2278
5.403.1 Detailed Description	2278
5.404stl_multimap.h File Reference	2278
5.404.1 Detailed Description	2279
5.405stl_multiset.h File Reference	2279
5.405.1 Detailed Description	2280
5.406stl_numeric.h File Reference	2280
5.406.1 Detailed Description	2281
5.407stl_pair.h File Reference	2281
5.407.1 Detailed Description	2282
5.408stl_queue.h File Reference	2282
5.408.1 Detailed Description	2282
5.409stl_raw_storage_iter.h File Reference	2283
5.409.1 Detailed Description	2283
5.410stl_relops.h File Reference	2283
5.410.1 Detailed Description	2283
5.411stl_set.h File Reference	2284
5.411.1 Detailed Description	2284
5.412stl_stack.h File Reference	2284
5.412.1 Detailed Description	2285
5.413stl_tempbuf.h File Reference	2285
5.413.1 Detailed Description	2285
5.414stl_tree.h File Reference	2286
5.414.1 Detailed Description	2287

5.415	stl_uninitialized.h File Reference	2287
5.415.1	Detailed Description	2288
5.416	stl_vector.h File Reference	2288
5.416.1	Detailed Description	2289
5.417	stream_iterator.h File Reference	2289
5.417.1	Detailed Description	2289
5.418	streambuf_iterator.h File Reference	2290
5.418.1	Detailed Description	2290
5.419	string_conversions.h File Reference	2290
5.419.1	Detailed Description	2291
5.420	stringfwd.h File Reference	2291
5.420.1	Detailed Description	2291
5.421	synth_access_traits.hpp File Reference	2291
5.421.1	Detailed Description	2292
5.422	tag_and_trait.hpp File Reference	2292
5.422.1	Detailed Description	2293
5.423	tags.h File Reference	2293
5.423.1	Detailed Description	2294
5.424	tgmath.h File Reference	2294
5.424.1	Detailed Description	2294
5.425	thin_heap_.hpp File Reference	2294
5.425.1	Detailed Description	2295
5.426	throw_allocator.h File Reference	2295
5.426.1	Detailed Description	2296
5.427	time_members.h File Reference	2296
5.427.1	Detailed Description	2296
5.428	trace_fn_imps.hpp File Reference	2297
5.428.1	Detailed Description	2297

5.429trace_fn_imps.hpp File Reference	2297
5.429.1 Detailed Description	2297
5.430trace_fn_imps.hpp File Reference	2297
5.430.1 Detailed Description	2297
5.431trace_fn_imps.hpp File Reference	2297
5.431.1 Detailed Description	2297
5.432trace_fn_imps.hpp File Reference	2297
5.432.1 Detailed Description	2297
5.433trace_fn_imps.hpp File Reference	2297
5.433.1 Detailed Description	2297
5.434trace_fn_imps.hpp File Reference	2298
5.434.1 Detailed Description	2298
5.435trace_fn_imps.hpp File Reference	2298
5.435.1 Detailed Description	2298
5.436traits.hpp File Reference	2298
5.436.1 Detailed Description	2298
5.437traits.hpp File Reference	2298
5.437.1 Detailed Description	2299
5.438traits.hpp File Reference	2299
5.438.1 Detailed Description	2299
5.439traits.hpp File Reference	2299
5.439.1 Detailed Description	2299
5.440traits.hpp File Reference	2299
5.440.1 Detailed Description	2300
5.441traits.hpp File Reference	2300
5.441.1 Detailed Description	2300
5.442tree_policy.hpp File Reference	2300
5.442.1 Detailed Description	2300

5.443tree_trace_base.hpp File Reference	2301
5.443.1 Detailed Description	2301
5.444trie_policy.hpp File Reference	2301
5.444.1 Detailed Description	2301
5.445trie_policy_base.hpp File Reference	2301
5.445.1 Detailed Description	2302
5.446trie_string_access_traits_imp.hpp File Reference	2302
5.446.1 Detailed Description	2302
5.447type_traits.h File Reference	2302
5.447.1 Detailed Description	2302
5.448type_utils.hpp File Reference	2302
5.448.1 Detailed Description	2303
5.449typelist.h File Reference	2303
5.449.1 Detailed Description	2304
5.450types.h File Reference	2304
5.450.1 Detailed Description	2305
5.451types_traits.hpp File Reference	2305
5.451.1 Detailed Description	2305
5.452unique_copy.h File Reference	2305
5.452.1 Detailed Description	2306
5.453unique_ptr.h File Reference	2306
5.453.1 Detailed Description	2307
5.454unordered_base.h File Reference	2307
5.454.1 Detailed Description	2307
5.455unordered_map.h File Reference	2308
5.455.1 Detailed Description	2309
5.456unordered_set.h File Reference	2309
5.456.1 Detailed Description	2310

5.457update_fn_imps.hpp File Reference	2310
5.457.1 Detailed Description	2310
5.458valarray_after.h File Reference	2310
5.458.1 Detailed Description	2320
5.459valarray_array.h File Reference	2320
5.459.1 Detailed Description	2327
5.460valarray_before.h File Reference	2328
5.460.1 Detailed Description	2328
5.461vstring.h File Reference	2328
5.461.1 Detailed Description	2330
5.462vstring_fwd.h File Reference	2330
5.462.1 Detailed Description	2331
5.463vstring_util.h File Reference	2331
5.463.1 Detailed Description	2331
5.464workstealing.h File Reference	2331
5.464.1 Detailed Description	2332
Index	2333

1 Todo List

Member [__gnu_debug::_Safe_iterator<_Iterator,_Sequence>::operator->\(\)](#) const noexcept

Make this correct w.r.t. iterators that return proxies

Member [__gnu_debug::_Safe_local_iterator<_Iterator,_Sequence>::operator->\(\)](#) const

Make this correct w.r.t. iterators that return proxies

Class [std::basic_string<_CharT,_Traits,_Alloc>](#)

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↔_style.html

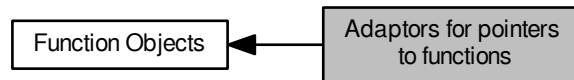
Member [std::regex_traits<_Ch_type>::transform_primary](#) (_Fwd_iter __first, _Fwd_iter __last) const

Implement this function correctly.

2 Module Documentation

2.1 Adaptors for pointers to functions

Collaboration diagram for Adaptors for pointers to functions:



Classes

- class `std::pointer_to_binary_function< _Arg1, _Arg2, _Result >`
- class `std::pointer_to_unary_function< _Arg, _Result >`

Functions

- `template<typename _Arg, typename _Result >`
`pointer_to_unary_function< _Arg, _Result > std::ptr_fun (_Result(*)(_Arg))`
- `template<typename _Arg1, typename _Arg2, typename _Result >`
`pointer_to_binary_function< _Arg1, _Arg2, _Result > std::ptr_fun (_Result(*)(_Arg1, _Arg2))`

2.1.1 Detailed Description

The advantage of function objects over pointers to functions is that the objects in the standard library declare nested typedefs describing their argument and result types with uniform names (e.g., `result_type` from the base classes `unary_function` and `binary_function`). Sometimes those typedefs are required, not just optional.

Adaptors are provided to turn pointers to unary (single-argument) and binary (double-argument) functions into function objects. The long-winded functor `pointer_to_unary_function` is constructed with a function pointer `f`, and its `operator()` called with argument `x` returns `f(x)`. The functor `pointer_to_binary_function` does the same thing, but with a double-argument `f` and `operator()`.

The function `ptr_fun` takes a pointer-to-function `f` and constructs an instance of the appropriate functor.

2.1.2 Function Documentation

2.1.2.1 `template<typename _Arg, typename _Result > pointer_to_unary_function< _Arg, _Result > std::ptr_fun (_Result(*)(_Arg) __x) [inline]`

One of the [adaptors for function pointers](#).

Definition at line 795 of file `stl_function.h`.

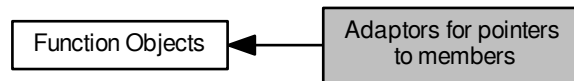
```
2.1.2.2  template<typename _Arg1, typename _Arg2, typename _Result > pointer_to_binary_function<_Arg1, _Arg2, _Result>  
        std::ptr_fun ( _Result(*)( _Arg1, _Arg2) __x )  [inline]
```

One of the [adaptors for function pointers](#).

Definition at line 821 of file `stl_function.h`.

2.2 Adaptors for pointers to members

Collaboration diagram for Adaptors for pointers to members:



Classes

- class `std::const_mem_fun1_ref_t<_Ret, _Tp, _Arg>`
- class `std::const_mem_fun1_t<_Ret, _Tp, _Arg>`
- class `std::const_mem_fun_ref_t<_Ret, _Tp>`
- class `std::const_mem_fun_t<_Ret, _Tp>`
- class `std::mem_fun1_ref_t<_Ret, _Tp, _Arg>`
- class `std::mem_fun1_t<_Ret, _Tp, _Arg>`
- class `std::mem_fun_ref_t<_Ret, _Tp>`
- class `std::mem_fun_t<_Ret, _Tp>`

Functions

- `template<typename _Ret, typename _Tp>`
`mem_fun_t<_Ret, _Tp> std::mem_fun (_Ret(_Tp::*__f)())`
- `template<typename _Ret, typename _Tp, typename _Arg>`
`mem_fun1_t<_Ret, _Tp, _Arg> std::mem_fun (_Ret(_Tp::*__f)(_Arg))`
- `template<typename _Ret, typename _Tp>`
`mem_fun_ref_t<_Ret, _Tp> std::mem_fun_ref (_Ret(_Tp::*__f)())`
- `template<typename _Ret, typename _Tp, typename _Arg>`
`mem_fun1_ref_t<_Ret, _Tp, _Arg> std::mem_fun_ref (_Ret(_Tp::*__f)(_Arg))`

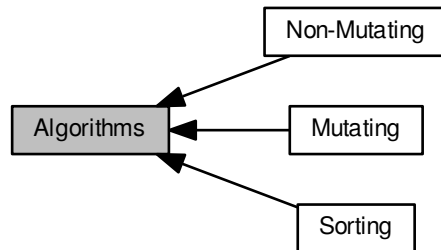
2.2.1 Detailed Description

There are a total of $8 = 2^3$ function objects in this family. (1) Member functions taking no arguments vs member functions taking one argument. (2) Call through pointer vs call through reference. (3) Const vs non-const member function.

All of this complexity is in the function objects themselves. You can ignore it by using the helper function `mem_fun` and `mem_fun_ref`, which create whichever type of adaptor is appropriate.

2.3 Algorithms

Collaboration diagram for Algorithms:



Modules

- [Mutating](#)
- [Non-Mutating](#)
- [Sorting](#)

2.3.1 Detailed Description

Components for performing algorithmic operations. Includes non-modifying sequence, modifying (mutating) sequence, sorting, searching, merge, partition, heap, set, minima, maxima, and permutation operations.

2.4 Allocators

Classes

- struct `__gnu_cxx::__alloc_traits<_Alloc>`
- class `__gnu_cxx::__mt_alloc<_Tp, _Poolp>`
- class `__gnu_cxx::__pool_alloc<_Tp>`
- class `__gnu_cxx::__ExtPtr_allocator<_Tp>`
- class `__gnu_cxx::array_allocator<_Tp, _Array>`
- class `__gnu_cxx::bitmap_allocator<_Tp>`
- class `__gnu_cxx::debug_allocator<_Alloc>`
- class `__gnu_cxx::malloc_allocator<_Tp>`
- class `__gnu_cxx::new_allocator<_Tp>`
- class `__gnu_cxx::throw_allocator_base<_Tp, _Cond>`
- class `std::allocator<_Tp>`
- class `std::allocator<void>`
- struct `std::allocator_traits<_Alloc>`
- struct `std::uses_allocator<_Tp, _Alloc>`

Typedefs

- template<typename `_Tp`>
using `std::__allocator_base` = `__gnu_cxx::new_allocator<_Tp>`

Functions

- template<typename `_T1`, typename `_T2`>
bool **`std::operator!=`** (const allocator< `_T1` > &, const allocator< `_T2` > &)
- template<typename `_Tp`>
bool **`std::operator!=`** (const allocator< `_Tp` > &, const allocator< `_Tp` > &)
- template<typename `_T1`, typename `_T2`>
bool **`std::operator==`** (const allocator< `_T1` > &, const allocator< `_T2` > &)
- template<typename `_Tp`>
bool **`std::operator==`** (const allocator< `_Tp` > &, const allocator< `_Tp` > &)

2.4.1 Detailed Description

Classes encapsulating memory operations.

2.4.2 Typedef Documentation

2.4.2.1 `template<typename _Tp> using std::__allocator_base = typedef __gnu_cxx::new_allocator<_Tp>`

An alias to the base class for `std::allocator`.

Used to set the `std::allocator` base class to `__gnu_cxx::new_allocator`.

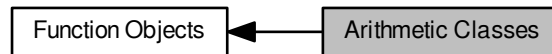
Template Parameters

<code>_Tp</code>	Type of allocated object.
------------------	---------------------------

Definition at line 48 of file c++allocator.h.

2.5 Arithmetic Classes

Collaboration diagram for Arithmetic Classes:



Classes

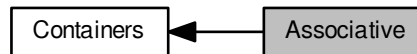
- struct [std::divides<_Tp>](#)
- struct [std::minus<_Tp>](#)
- struct [std::modulus<_Tp>](#)
- struct [std::multiplies<_Tp>](#)
- struct [std::negate<_Tp>](#)
- struct [std::plus<_Tp>](#)

2.5.1 Detailed Description

Because basic math often needs to be done during an algorithm, the library provides functors for those operations. See the documentation for [the base classes](#) for examples of their use.

2.6 Associative

Collaboration diagram for Associative:



Classes

- class `std::map< _Key, _Tp, _Compare, _Alloc >`
- class `std::multimap< _Key, _Tp, _Compare, _Alloc >`
- class `std::multiset< _Key, _Compare, _Alloc >`
- class `std::set< _Key, _Compare, _Alloc >`

2.6.1 Detailed Description

Associative containers allow fast retrieval of data based on keys.

Each container type is parameterized on a `Key` type, and an ordering relation used to sort the elements of the container.

All associative containers must meet certain requirements, summarized in [tables](#).

2.7 Atomics

Classes

- struct [std::__atomic_base<_ITp>](#)
- struct [std::__atomic_base<_PTp*>](#)
- struct [std::__atomic_flag_base](#)
- struct [std::atomic_flag](#)

Macros

- [#define ATOMIC_BOOL_LOCK_FREE](#)
- [#define ATOMIC_CHAR16_T_LOCK_FREE](#)
- [#define ATOMIC_CHAR32_T_LOCK_FREE](#)
- [#define ATOMIC_CHAR_LOCK_FREE](#)
- [#define ATOMIC_FLAG_INIT](#)
- [#define ATOMIC_INT_LOCK_FREE](#)
- [#define ATOMIC_LLONG_LOCK_FREE](#)
- [#define ATOMIC_LONG_LOCK_FREE](#)
- [#define ATOMIC_POINTER_LOCK_FREE](#)
- [#define ATOMIC_SHORT_LOCK_FREE](#)
- [#define ATOMIC_VAR_INIT\(_VI\)](#)
- [#define ATOMIC_WCHAR_T_LOCK_FREE](#)

Typedefs

- typedef unsigned char [std::__atomic_flag_data_type](#)
- typedef [__atomic_base<char>](#) [std::atomic_char](#)
- typedef [__atomic_base<char16_t>](#) [std::atomic_char16_t](#)
- typedef [__atomic_base<char32_t>](#) [std::atomic_char32_t](#)
- typedef [__atomic_base<int>](#) [std::atomic_int](#)
- typedef [__atomic_base<int_fast16_t>](#) [std::atomic_int_fast16_t](#)
- typedef [__atomic_base<int_fast32_t>](#) [std::atomic_int_fast32_t](#)
- typedef [__atomic_base<int_fast64_t>](#) [std::atomic_int_fast64_t](#)
- typedef [__atomic_base<int_fast8_t>](#) [std::atomic_int_fast8_t](#)
- typedef [__atomic_base<int_least16_t>](#) [std::atomic_int_least16_t](#)
- typedef [__atomic_base<int_least32_t>](#) [std::atomic_int_least32_t](#)
- typedef [__atomic_base<int_least64_t>](#) [std::atomic_int_least64_t](#)
- typedef [__atomic_base<int_least8_t>](#) [std::atomic_int_least8_t](#)
- typedef [__atomic_base<intmax_t>](#) [std::atomic_intmax_t](#)
- typedef [__atomic_base<intptr_t>](#) [std::atomic_intptr_t](#)
- typedef [__atomic_base<long long>](#) [std::atomic_llong](#)
- typedef [__atomic_base<long>](#) [std::atomic_long](#)
- typedef [__atomic_base<ptrdiff_t>](#) [std::atomic_ptrdiff_t](#)
- typedef [__atomic_base<signed char>](#) [std::atomic_schar](#)
- typedef [__atomic_base<short>](#) [std::atomic_short](#)
- typedef [__atomic_base<size_t>](#) [std::atomic_size_t](#)
- typedef [__atomic_base<unsigned char>](#) [std::atomic_uchar](#)
- typedef [__atomic_base<unsigned int>](#) [std::atomic_uint](#)

- typedef __atomic_base< uint_fast16_t > std::atomic_uint_fast16_t
- typedef __atomic_base< uint_fast32_t > std::atomic_uint_fast32_t
- typedef __atomic_base< uint_fast64_t > std::atomic_uint_fast64_t
- typedef __atomic_base< uint_fast8_t > std::atomic_uint_fast8_t
- typedef __atomic_base< uint_least16_t > std::atomic_uint_least16_t
- typedef __atomic_base< uint_least32_t > std::atomic_uint_least32_t
- typedef __atomic_base< uint_least64_t > std::atomic_uint_least64_t
- typedef __atomic_base< uint_least8_t > std::atomic_uint_least8_t
- typedef __atomic_base< uintmax_t > std::atomic_uintmax_t
- typedef __atomic_base< uintptr_t > std::atomic_uintptr_t
- typedef __atomic_base< unsigned long long > std::atomic_ullong
- typedef __atomic_base< unsigned long > std::atomic_ulong
- typedef __atomic_base< unsigned short > std::atomic_ushort
- typedef __atomic_base< wchar_t > std::atomic_wchar_t
- typedef enum std::memory_order std::memory_order

Enumerations

- enum __memory_order_modifier { __memory_order_mask, __memory_order_modifier_mask, __memory_order_hle_acquire, __memory_order_hle_release }
- enum std::memory_order { memory_order_relaxed, memory_order_consume, memory_order_acquire, memory_order_release, memory_order_acq_rel, memory_order_seq_cst }

Functions

- std::__attribute__((always_inline)) void atomic_thread_fence(memory_order __m) noexcept
- constexpr memory_order std::__cmpexch_failure_order (memory_order __m) noexcept
- constexpr memory_order std::__cmpexch_failure_order2 (memory_order __m) noexcept
- template<typename _Tp >
_Tp std::kill_dependency (_Tp __y) noexcept
- constexpr memory_order std::operator& (memory_order __m, __memory_order_modifier __mod)
- constexpr memory_order std::operator| (memory_order __m, __memory_order_modifier __mod)

2.7.1 Detailed Description

Components for performing atomic operations.

2.7.2 Macro Definition Documentation

2.7.2.1 #define ATOMIC_BOOL_LOCK_FREE

Lock-free property.

0 indicates that the types are never lock-free. 1 indicates that the types are sometimes lock-free. 2 indicates that the types are always lock-free.

Definition at line 49 of file atomic_lockfree_defines.h.

2.7.3 Typedef Documentation

2.7.3.1 typedef __atomic_base<char> std::atomic_char

atomic_char

Definition at line 121 of file atomic_base.h.

2.7.3.2 typedef __atomic_base<char16_t> std::atomic_char16_t

atomic_char16_t

Definition at line 160 of file atomic_base.h.

2.7.3.3 typedef __atomic_base< char32_t > std::atomic_char32_t

atomic_char32_t

Definition at line 163 of file atomic_base.h.

2.7.3.4 typedef __atomic_base<int> std::atomic_int

atomic_int

Definition at line 139 of file atomic_base.h.

2.7.3.5 typedef __atomic_base<int_fast16_t> std::atomic_int_fast16_t

atomic_int_fast16_t

Definition at line 201 of file atomic_base.h.

2.7.3.6 typedef __atomic_base<int_fast32_t> std::atomic_int_fast32_t

atomic_int_fast32_t

Definition at line 207 of file atomic_base.h.

2.7.3.7 typedef __atomic_base<int_fast64_t> std::atomic_int_fast64_t

atomic_int_fast64_t

Definition at line 213 of file atomic_base.h.

2.7.3.8 typedef __atomic_base<int_fast8_t> std::atomic_int_fast8_t

atomic_int_fast8_t

Definition at line 195 of file atomic_base.h.

2.7.3.9 `typedef __atomic_base<int_least16_t> std::atomic_int_least16_t`

`atomic_int_least16_t`

Definition at line 176 of file `atomic_base.h`.

2.7.3.10 `typedef __atomic_base<int_least32_t> std::atomic_int_least32_t`

`atomic_int_least32_t`

Definition at line 182 of file `atomic_base.h`.

2.7.3.11 `typedef __atomic_base<int_least64_t> std::atomic_int_least64_t`

`atomic_int_least64_t`

Definition at line 188 of file `atomic_base.h`.

2.7.3.12 `typedef __atomic_base<int_least8_t> std::atomic_int_least8_t`

`atomic_int_least8_t`

Definition at line 170 of file `atomic_base.h`.

2.7.3.13 `typedef __atomic_base<intmax_t> std::atomic_intmax_t`

`atomic_intmax_t`

Definition at line 229 of file `atomic_base.h`.

2.7.3.14 `typedef __atomic_base<intptr_t> std::atomic_intptr_t`

`atomic_intptr_t`

Definition at line 220 of file `atomic_base.h`.

2.7.3.15 `typedef __atomic_base<long long> std::atomic_llong`

`atomic_llong`

Definition at line 151 of file `atomic_base.h`.

2.7.3.16 `typedef __atomic_base<long> std::atomic_long`

`atomic_long`

Definition at line 145 of file `atomic_base.h`.

2.7.3.17 typedef __atomic_base<ptrdiff_t> std::atomic_ptrdiff_t

atomic_ptrdiff_t

Definition at line 235 of file atomic_base.h.

2.7.3.18 typedef __atomic_base<signed char> std::atomic_schar

atomic_schar

Definition at line 127 of file atomic_base.h.

2.7.3.19 typedef __atomic_base<short> std::atomic_short

atomic_short

Definition at line 133 of file atomic_base.h.

2.7.3.20 typedef __atomic_base<size_t> std::atomic_size_t

atomic_size_t

Definition at line 226 of file atomic_base.h.

2.7.3.21 typedef __atomic_base<unsigned char> std::atomic_uchar

atomic_uchar

Definition at line 130 of file atomic_base.h.

2.7.3.22 typedef __atomic_base<unsigned int> std::atomic_uint

atomic_uint

Definition at line 142 of file atomic_base.h.

2.7.3.23 typedef __atomic_base<uint_fast16_t> std::atomic_uint_fast16_t

atomic_uint_fast16_t

Definition at line 204 of file atomic_base.h.

2.7.3.24 typedef __atomic_base<uint_fast32_t> std::atomic_uint_fast32_t

atomic_uint_fast32_t

Definition at line 210 of file atomic_base.h.

2.7.3.25 `typedef __atomic_base<uint_fast64_t> std::atomic_uint_fast64_t`

`atomic_uint_fast64_t`

Definition at line 216 of file `atomic_base.h`.

2.7.3.26 `typedef __atomic_base<uint_fast8_t> std::atomic_uint_fast8_t`

`atomic_uint_fast8_t`

Definition at line 198 of file `atomic_base.h`.

2.7.3.27 `typedef __atomic_base<uint_least16_t> std::atomic_uint_least16_t`

`atomic_uint_least16_t`

Definition at line 179 of file `atomic_base.h`.

2.7.3.28 `typedef __atomic_base<uint_least32_t> std::atomic_uint_least32_t`

`atomic_uint_least32_t`

Definition at line 185 of file `atomic_base.h`.

2.7.3.29 `typedef __atomic_base<uint_least64_t> std::atomic_uint_least64_t`

`atomic_uint_least64_t`

Definition at line 191 of file `atomic_base.h`.

2.7.3.30 `typedef __atomic_base<uint_least8_t> std::atomic_uint_least8_t`

`atomic_uint_least8_t`

Definition at line 173 of file `atomic_base.h`.

2.7.3.31 `typedef __atomic_base<uintmax_t> std::atomic_uintmax_t`

`atomic_uintmax_t`

Definition at line 232 of file `atomic_base.h`.

2.7.3.32 `typedef __atomic_base<uintptr_t> std::atomic_uintptr_t`

`atomic_uintptr_t`

Definition at line 223 of file `atomic_base.h`.

2.7.3.33 typedef __atomic_base<unsigned long long> std::atomic_ullong

atomic_ullong

Definition at line 154 of file atomic_base.h.

2.7.3.34 typedef __atomic_base<unsigned long> std::atomic_ulong

atomic_ulong

Definition at line 148 of file atomic_base.h.

2.7.3.35 typedef __atomic_base<unsigned short> std::atomic_ushort

atomic_ushort

Definition at line 136 of file atomic_base.h.

2.7.3.36 typedef __atomic_base<wchar_t> std::atomic_wchar_t

atomic_wchar_t

Definition at line 157 of file atomic_base.h.

2.7.3.37 typedef enum std::memory_order std::memory_order

Enumeration for memory_order.

2.7.4 Enumeration Type Documentation**2.7.4.1 enum std::memory_order**

Enumeration for memory_order.

Definition at line 56 of file atomic_base.h.

2.7.5 Function Documentation**2.7.5.1 template<typename _Tp> _Tp std::kill_dependency (_Tp__y) [inline],[noexcept]**

kill_dependency

Definition at line 112 of file atomic_base.h.

2.8 Base and Implementation Classes

Collaboration diagram for Base and Implementation Classes:



Classes

- struct std::__detail::__Default_ranged_hash
- struct std::__detail::__Equal_helper< _Key, _Value, _ExtractKey, _Equal, _HashCodeType, __cache_hash_code >
- struct std::__detail::__Equal_helper< _Key, _Value, _ExtractKey, _Equal, _HashCodeType, false >
- struct std::__detail::__Equal_helper< _Key, _Value, _ExtractKey, _Equal, _HashCodeType, true >
- struct std::__detail::__Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, _Unique_keys >
- struct std::__detail::__Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false >
- struct std::__detail::__Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true >
- struct std::__detail::__Equality_base
- struct std::__detail::__Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache_hash_code >
- struct std::__detail::__Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Default_ranged_hash, false >
- struct std::__detail::__Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Default_ranged_hash, true >
- struct std::__detail::__Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, false >
- struct std::__detail::__Hash_node< _Value, _Cache_hash_code >
- struct std::__detail::__Hash_node< _Value, false >
- struct std::__detail::__Hash_node< _Value, true >
- struct std::__detail::__Hash_node_base
- struct std::__detail::__Hash_node_value_base< _Value >
- struct std::__detail::__Hashtable_alloc< _NodeAlloc >
- struct std::__detail::__Hashtable_base< _Key, _Value, _ExtractKey, _Equal, _H1, _H2, _Hash, _Traits >
- struct std::__detail::__Hashtable_ebo_helper< _Nm, _Tp, __use_ebo >
- struct std::__detail::__Hashtable_ebo_helper< _Nm, _Tp, false >
- struct std::__detail::__Hashtable_ebo_helper< _Nm, _Tp, true >
- struct std::__detail::__Hashtable_traits< _Cache_hash_code, _Constant_iterators, _Unique_keys >
- struct std::__detail::__Insert< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, __ Traits, _Constant_iterators, _Unique_keys >
- struct std::__detail::__Insert< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, __ Traits, false, _Unique_keys >
- struct std::__detail::__Insert< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, __ Traits, true, false >

- `struct std::__detail::Insert< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true, true >`
- `struct std::__detail::Insert_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >`
- `struct std::__detail::Local_const_iterator< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __constant_iterators, __cache >`
- `struct std::__detail::Local_iterator< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __constant_iterators, __cache >`
- `struct std::__detail::Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache_hash_code >`
- `struct std::__detail::Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, true >`
- `struct std::__detail::Map_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, _Unique_keys >`
- `struct std::__detail::Map_base< _Key, _Pair, _Alloc, _Select1st, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false >`
- `struct std::__detail::Map_base< _Key, _Pair, _Alloc, _Select1st, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true >`
- `struct std::__detail::Mod_range_hashing`
- `struct std::__detail::Node_const_iterator< _Value, __constant_iterators, __cache >`
- `struct std::__detail::Node_iterator< _Value, __constant_iterators, __cache >`
- `struct std::__detail::Node_iterator_base< _Value, _Cache_hash_code >`
- `struct std::__detail::Prime_rehash_policy`
- `struct std::__detail::Rehash_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >`
- `struct std::__detail::Rehash_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _Prime_rehash_policy, _Traits >`
- `class std::__detail::Hashtable< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >`

Typedefs

- `template<typename _Key, typename _Value, typename _ExtractKey, typename _H1, typename _H2, typename _Hash >
using std::__detail::hash_code_for_local_iter = _Hash_code_storage< _Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, false >>`

Functions

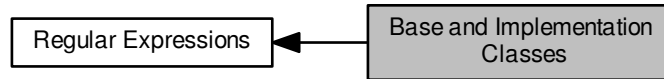
- `template<class _Iterator >
std::iterator_traits< _Iterator >::difference_type std::__detail::__distance_fw (_Iterator __first, _Iterator __last, std::input_iterator_tag)`
- `template<class _Iterator >
std::iterator_traits< _Iterator >::difference_type std::__detail::__distance_fw (_Iterator __first, _Iterator __last, std::forward_iterator_tag)`
- `template<class _Iterator >
std::iterator_traits< _Iterator >::difference_type std::__detail::__distance_fw (_Iterator __first, _Iterator __last)`
- `__bucket_type * std::__detail::Hashtable_alloc< _NodeAlloc >::M_allocate_buckets (std::size_t __n)`
- `template<typename... _Args>
__node_type * std::__detail::Hashtable_alloc< _NodeAlloc >::M_allocate_node (_Args &&... __args)`
- `void std::__detail::Hashtable_alloc< _NodeAlloc >::M_deallocate_buckets (__bucket_type *, std::size_t __n)`
- `void std::__detail::Hashtable_alloc< _NodeAlloc >::M_deallocate_node (__node_type * __n)`
- `void std::__detail::Hashtable_alloc< _NodeAlloc >::M_deallocate_nodes (__node_type * __n)`

- `template<typename _InputIterator, typename _NodeGetter >`
`void std::__detail::Insert_base<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _Rehash, _Policy, _Traits >::M_insert_range` (`_InputIterator __first, _InputIterator __last, const _NodeGetter &`)
- `template<typename _Uiterator >`
`static bool std::__detail::Equality_base::S_is_permutation` (`_Uiterator, _Uiterator, _Uiterator`)
- `template<typename _Value, bool _Cache_hash_code>`
`bool std::__detail::operator!=` (`const _Node_iterator_base< _Value, _Cache_hash_code > &__x, const _Node_iterator_base< _Value, _Cache_hash_code > &__y`) `noexcept`
- `template<typename _Key, typename _Value, typename _ExtractKey, typename _H1, typename _H2, typename _Hash, bool __cache>`
`bool std::__detail::operator!=` (`const _Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache > &__x, const _Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache > &__y`)
- `template<typename _Value, bool _Cache_hash_code>`
`bool std::__detail::operator==` (`const _Node_iterator_base< _Value, _Cache_hash_code > &__x, const _Node_iterator_base< _Value, _Cache_hash_code > &__y`) `noexcept`
- `template<typename _Key, typename _Value, typename _ExtractKey, typename _H1, typename _H2, typename _Hash, bool __cache>`
`bool std::__detail::operator==` (`const _Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache > &__x, const _Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache > &__y`)

2.8.1 Detailed Description

2.9 Base and Implementation Classes

Collaboration diagram for Base and Implementation Classes:



Classes

- struct `std::__detail::BracketMatcher<_TraitsT, __icase, __collate >`
- class `std::__detail::Compiler<_TraitsT >`
- class `std::__detail::Executor<_Bilter, _Alloc, _TraitsT, __dfs_mode >`
- class `std::__detail::Scanner<_CharT >`
- class `std::__detail::StateSeq<_TraitsT >`

Typedefs

- template<typename _CharT >
using `std::__detail::Matcher` = `std::function< bool(_CharT)>`
- typedef long `std::__detail::StateldT`

Enumerations

- enum `std::__detail::Opcode` : int {
`_S_opcode_unknown`, `_S_opcode_alternative`, `_S_opcode_backref`, `_S_opcode_line_begin_assertion`,
`_S_opcode_line_end_assertion`, `_S_opcode_word_boundary`, `_S_opcode_subexpr_lookahead`, `_S_opcode_subexpr_begin`,
`_S_opcode_subexpr_end`, `_S_opcode_dummy`, `_S_opcode_match`, `_S_opcode_accept` }

Functions

- template<typename _TraitsT >
`std::shared_ptr<_NFA<_TraitsT > >` `std::__detail::compile_nfa` (const typename _TraitsT::char_type * __first, const typename _TraitsT::char_type * __last, const _TraitsT & __traits, `regex_constants::syntax_option_type` __flags)

Variables

- static const `_StateldT` `std::__detail::_S_invalid_state_id`

2.9.1 Detailed Description

2.9.2 Enumeration Type Documentation

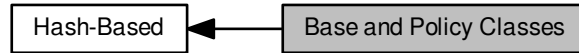
2.9.2.1 enum std::__detail::_Opcode : int

Operation codes that define the type of transitions within the base NFA that represents the regular expression.

Definition at line 51 of file regex_automaton.h.

2.10 Base and Policy Classes

Collaboration diagram for Base and Policy Classes:



Classes

- [class `__gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >`](#)
- [class `__gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >`](#)

2.10.1 Detailed Description

2.11 Base and Policy Classes

Collaboration diagram for Base and Policy Classes:



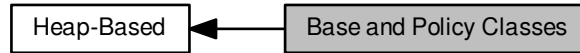
Classes

- [class `__gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >`](#)
- [class `__gnu_pbds::detail::pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >`](#)
- [class `__gnu_pbds::detail::rb_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >`](#)
- [class `__gnu_pbds::detail::splay_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >`](#)

2.11.1 Detailed Description

2.12 Base and Policy Classes

Collaboration diagram for Base and Policy Classes:



Classes

- class `__gnu_pbds::detail::binary_heap< Value_Type, Cmp_Fn, _Alloc >`
- class `__gnu_pbds::detail::binomial_heap< Value_Type, Cmp_Fn, _Alloc >`
- class `__gnu_pbds::detail::pairing_heap< Value_Type, Cmp_Fn, _Alloc >`
- class `__gnu_pbds::detail::rc_binomial_heap< Value_Type, Cmp_Fn, _Alloc >`
- class `__gnu_pbds::detail::thin_heap< Value_Type, Cmp_Fn, _Alloc >`

2.12.1 Detailed Description

2.13 Bernoulli Distributions

Collaboration diagram for Bernoulli Distributions:



Classes

- class `std::bernoulli_distribution`
- class `std::binomial_distribution< _IntType >`
- class `std::geometric_distribution< _IntType >`
- class `std::negative_binomial_distribution< _IntType >`

Functions

- `bool std::operator!= (const std::bernoulli_distribution &__d1, const std::bernoulli_distribution &__d2)`
- `template<typename _IntType >
bool std::operator!= (const std::binomial_distribution< _IntType > &__d1, const std::binomial_distribution< _IntType > &__d2)`
- `template<typename _IntType >
bool std::operator!= (const std::geometric_distribution< _IntType > &__d1, const std::geometric_distribution< _IntType > &__d2)`
- `template<typename _IntType >
bool std::operator!= (const std::negative_binomial_distribution< _IntType > &__d1, const std::negative_binomial_distribution< _IntType > &__d2)`
- `template<typename _CharT, typename _Traits >
std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::bernoulli_distribution &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >
std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::geometric_distribution< _IntType > &__x)`
- `template<typename _CharT, typename _Traits >
std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::bernoulli_distribution &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >
std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::geometric_distribution< _IntType > &__x)`

2.13.1 Detailed Description

2.13.2 Function Documentation

2.13.2.1 `bool std::operator!=(const std::bernoulli_distribution & __d1, const std::bernoulli_distribution & __d2)`
`[inline]`

Return true if two Bernoulli distributions have different parameters.

Definition at line 3731 of file random.h.

2.13.2.2 `template<typename _IntType> bool std::operator!=(const std::binomial_distribution<_IntType> & __d1, const`
`std::binomial_distribution<_IntType> & __d2) [inline]`

Return true if two binomial distributions are different.

Definition at line 3997 of file random.h.

2.13.2.3 `template<typename _IntType> bool std::operator!=(const std::geometric_distribution<_IntType> & __d1, const`
`std::geometric_distribution<_IntType> & __d2) [inline]`

Return true if two geometric distributions have different parameters.

Definition at line 4166 of file random.h.

References `std::operator>>()`.

2.13.2.4 `template<typename _IntType> bool std::operator!=(const std::negative_binomial_distribution<_IntType> &`
`__d1, const std::negative_binomial_distribution<_IntType> & __d2) [inline]`

Return true if two negative binomial distributions are different.

Definition at line 4411 of file random.h.

2.13.2.5 `template<typename _CharT, typename _Traits> std::basic_ostream<_CharT, _Traits> & std::operator<< (`
`std::basic_ostream<_CharT, _Traits> & __os, const std::bernoulli_distribution & __x)`

Inserts a `bernoulli_distribution` random number distribution `__x` into the output stream `__os`.

Parameters

<code>__os</code>	An output stream.
<code>__x</code>	A <code>bernoulli_distribution</code> random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

2.13.2.6 `template<typename _IntType, typename _CharT, typename _Traits> std::basic_ostream<_CharT, _Traits>& std::operator<< (std::basic_ostream<_CharT, _Traits> & __os, const std::geometric_distribution<_IntType> & __x)`

Inserts a `geometric_distribution` random number distribution `__x` into the output stream `__os`.

Parameters

<code>__os</code>	An output stream.
<code>__x</code>	A <code>geometric_distribution</code> random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

2.13.2.7 `template<typename _CharT, typename _Traits> std::basic_istream<_CharT, _Traits>& std::operator>> (std::basic_istream<_CharT, _Traits> & __is, std::bernoulli_distribution & __x)`

Extracts a `bernoulli_distribution` random number distribution `__x` from the input stream `__is`.

Parameters

<code>__is</code>	An input stream.
<code>__x</code>	A <code>bernoulli_distribution</code> random number generator engine.

Returns

The input stream with `__x` extracted or in an error state.

Definition at line 3761 of file `random.h`.

References `std::bernoulli_distribution::param()`.

2.13.2.8 `template<typename _IntType, typename _CharT, typename _Traits> std::basic_istream<_CharT, _Traits>& std::operator>> (std::basic_istream<_CharT, _Traits> & __is, std::geometric_distribution<_IntType> & __x)`

Extracts a `geometric_distribution` random number distribution `__x` from the input stream `__is`.

Parameters

<code>__is</code>	An input stream.
<code>__x</code>	A <code>geometric_distribution</code> random number generator engine.

Returns

The input stream with ___x extracted or in an error state.

2.14 Binary Search

Collaboration diagram for Binary Search:



Functions

- `template<typename _ForwardIterator, typename _Tp >`
`bool std::binary_search (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`
`bool std::binary_search (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Tp >`
`pair< _ForwardIterator, _ForwardIterator > std::equal_range (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`
`pair< _ForwardIterator, _ForwardIterator > std::equal_range (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Tp >`
`_ForwardIterator std::lower_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`
`_ForwardIterator std::lower_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Tp >`
`_ForwardIterator std::upper_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`
`_ForwardIterator std::upper_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)`

2.14.1 Detailed Description

These algorithms are variations of a classic binary search, and all assume that the sequence being searched is already sorted.

The number of comparisons will be logarithmic (and as few as possible). The number of steps through the sequence will be logarithmic for random-access iterators (e.g., pointers), and linear otherwise.

The LWG has passed Defect Report 270, which notes: *The proposed resolution reinterprets binary search. Instead of thinking about searching for a value in a sorted range, we view that as an important special case of a more general algorithm: searching for the partition point in a partitioned range. We also add a guarantee that the old wording did not: we ensure that the upper bound is no earlier than the lower bound, that the pair returned by equal_range is a valid range, and that the first part of that pair is the lower bound.*

The actual effect of the first sentence is that a comparison functor passed by the user doesn't necessarily need to induce a strict weak ordering relation. Rather, it partitions the range.

2.14.2 Function Documentation

2.14.2.1 `template<typename _ForwardIterator, typename _Tp> bool std::binary_search (_ForwardIterator __first, _ForwardIterator __last, const _Tp & __val)`

Determines whether an element exists in a range.

Parameters

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.
<code>__val</code>	The search term.

Returns

True if `__val` (or its equivalent) is in `[__first, __last]`.

Note that this does not actually return an iterator to `__val`. For that, use `std::find` or a container's specialized find member functions.

Definition at line 2258 of file `stl_algo.h`.

2.14.2.2 `template<typename _ForwardIterator, typename _Tp, typename _Compare> bool std::binary_search (_ForwardIterator __first, _ForwardIterator __last, const _Tp & __val, _Compare __comp)`

Determines whether an element exists in a range.

Parameters

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.
<code>__val</code>	The search term.
<code>__comp</code>	A functor to use for comparisons.

Returns

True if `__val` (or its equivalent) is in `[__first, __last]`.

Note that this does not actually return an iterator to `__val`. For that, use `std::find` or a container's specialized find member functions.

The comparison function should have the same effects on ordering as the function used for the initial sort.

Definition at line 2293 of file `stl_algo.h`.

2.14.2.3 `template<typename _ForwardIterator, typename _Tp> pair<_ForwardIterator, _ForwardIterator> std::equal_range (_ForwardIterator __first, _ForwardIterator __last, const _Tp & __val) [inline]`

Finds the largest subrange in which `__val` could be inserted at any place in it without changing the ordering.

Parameters

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.
<code>__val</code>	The search term.

Returns

An pair of iterators defining the subrange.

This is equivalent to

```
std::make_pair(lower_bound(__first, __last, __val),
               upper_bound(__first, __last, __val))
```

but does not actually call those functions.

Definition at line 2185 of file `stl_algo.h`.

```
2.14.2.4 template<typename _ForwardIterator, typename _Tp, typename _Compare > pair<_ForwardIterator, _ForwardIterator>
std::equal_range ( _ForwardIterator __first, _ForwardIterator __last, const _Tp & __val, _Compare __comp )
[inline]
```

Finds the largest subrange in which `__val` could be inserted at any place in it without changing the ordering.

Parameters

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.
<code>__val</code>	The search term.
<code>__comp</code>	A functor to use for comparisons.

Returns

An pair of iterators defining the subrange.

This is equivalent to

```
std::make_pair(lower_bound(__first, __last, __val, __comp),
               upper_bound(__first, __last, __val, __comp))
```

but does not actually call those functions.

Definition at line 2222 of file `stl_algo.h`.

```
2.14.2.5 template<typename _ForwardIterator, typename _Tp > _ForwardIterator std::lower_bound ( _ForwardIterator __first,
_FForwardIterator __last, const _Tp & __val ) [inline]
```

Finds the first position in which `val` could be inserted without changing the ordering.

Parameters

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.
<code>__val</code>	The search term.

Returns

An iterator pointing to the first element *not less than* `val`, or `end()` if every element is less than `val`.

Definition at line 996 of file `stl_algobase.h`.

2.14.2.6 `template<typename _ForwardIterator, typename _Tp, typename _Compare > _ForwardIterator std::lower_bound (`
`_ForwardIterator __first, _ForwardIterator __last, const _Tp & __val, _Compare __comp) [inline]`

Finds the first position in which `__val` could be inserted without changing the ordering.

Parameters

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.
<code>__val</code>	The search term.
<code>__comp</code>	A functor to use for comparisons.

Returns

An iterator pointing to the first element *not less than* `__val`, or `end()` if every element is less than `__val`.

The comparison function should have the same effects on ordering as the function used for the initial sort.

Definition at line 2022 of file `stl_algo.h`.

References `std::advance()`, and `std::distance()`.

2.14.2.7 `template<typename _ForwardIterator, typename _Tp > _ForwardIterator std::upper_bound (`
`_ForwardIterator __first, _ForwardIterator __last, const _Tp & __val) [inline]`

Finds the last position in which `__val` could be inserted without changing the ordering.

Parameters

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.
<code>__val</code>	The search term.

Returns

An iterator pointing to the first element greater than `__val`, or `end()` if no elements are greater than `__val`.

Definition at line 2079 of file `stl_algo.h`.

2.14.2.8 `template<typename _ForwardIterator, typename _Tp, typename _Compare> _ForwardIterator std::upper_bound (`
`_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp) [inline]`

Finds the last position in which `__val` could be inserted without changing the ordering.

Parameters

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.
<code>__val</code>	The search term.
<code>__comp</code>	A functor to use for comparisons.

Returns

An iterator pointing to the first element greater than `__val`, or `end()` if no elements are greater than `__val`.

The comparison function should have the same effects on ordering as the function used for the initial sort.

Definition at line 2111 of file `stl_algo.h`.

References `std::advance()`, and `std::distance()`.

2.15 Binder Classes

Collaboration diagram for Binder Classes:



Classes

- class `std::binder1st<_Operation>`
- class `std::binder2nd<_Operation>`

Functions

- template<typename `_Operation` , typename `_Tp` >
`binder1st<_Operation>` `std::bind1st` (const `_Operation` &`__fn`, const `_Tp` &`__x`)
- template<typename `_Operation` , typename `_Tp` >
`binder2nd<_Operation>` `std::bind2nd` (const `_Operation` &`__fn`, const `_Tp` &`__x`)

2.15.1 Detailed Description

Binders turn functions/functors with two arguments into functors with a single argument, storing an argument to be applied later. For example, a variable `B` of type `binder1st` is constructed from a functor `f` and an argument `x`. Later, `B`'s `operator()` is called with a single argument `y`. The return value is the value of `f(x, y)`. `B` can be *called* with various arguments (`y1`, `y2`, ...) and will in turn call `f(x, y1)`, `f(x, y2)`, ...

The function `bind1st` is provided to save some typing. It takes the function and an argument as parameters, and returns an instance of `binder1st`.

The type `binder2nd` and its creator function `bind2nd` do the same thing, but the stored argument is passed as the second parameter instead of the first, e.g., `bind2nd(std::minus<float>(), 1.3)` will create a functor whose `operator()` accepts a floating-point number, subtracts 1.3 from it, and returns the result. (If `bind1st` had been used, the functor would perform `1.3 - x` instead.

Creator-wrapper functions like `bind1st` are intended to be used in calling algorithms. Their return values will be temporary objects. (The goal is to not require you to type names like `std::binder1st<std::plus<int>>` for declaring a variable to hold the return value from `bind1st(std::plus<int>(), 5)`).

These become more useful when combined with the composition functions.

These functions are deprecated in C++11 and can be replaced by `std::bind` (or `std::tr1::bind`) which is more powerful and flexible, supporting functions with any number of arguments. Uses of `bind1st` can be replaced by `std::bind(f, x, std::placeholders::_1)` and `bind2nd` by `std::bind(f, std::placeholders::_1, x)`.

2.15.2 Function Documentation

2.15.2.1 `template<typename _Operation, typename _Tp> binder1st<_Operation> std::bind1st (const _Operation & __fn, const _Tp & __x) [inline]`

One of the [binder functors](#).

Definition at line 131 of file binders.h.

2.15.2.2 `template<typename _Operation, typename _Tp> binder2nd<_Operation> std::bind2nd (const _Operation & __fn, const _Tp & __x) [inline]`

One of the [binder functors](#).

Definition at line 166 of file binders.h.

2.16 Boolean Operations Classes

Collaboration diagram for Boolean Operations Classes:



Classes

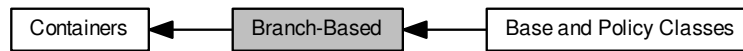
- struct `std::logical_and<_Tp>`
- struct `std::logical_not<_Tp>`
- struct `std::logical_or<_Tp>`

2.16.1 Detailed Description

Here are wrapper functors for Boolean operations: `&&`, `||`, and `!`.

2.17 Branch-Based

Collaboration diagram for Branch-Based:



Modules

- [Base and Policy Classes](#)

Classes

- `class __gnu_pbds::basic_branch< Key, Mapped, Tag, Node_Update, Policy_Tl, _Alloc >`
- `class __gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc >`
- `class __gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc >`

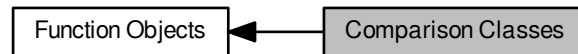
Macros

- `#define PB_DS_BRANCH_BASE`
- `#define PB_DS_TREE_BASE`
- `#define PB_DS_TREE_NODE_AND_IT_TRAITS`
- `#define PB_DS_TRIE_BASE`
- `#define PB_DS_TRIE_NODE_AND_IT_TRAITS`

2.17.1 Detailed Description

2.18 Comparison Classes

Collaboration diagram for Comparison Classes:



Classes

- struct `std::equal_to<_Tp>`
- struct `std::greater<_Tp>`
- struct `std::greater_equal<_Tp>`
- struct `std::less<_Tp>`
- struct `std::less_equal<_Tp>`
- struct `std::not_equal_to<_Tp>`

2.18.1 Detailed Description

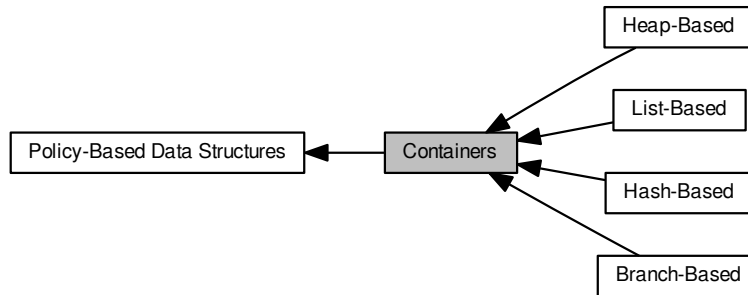
The library provides six wrapper functors for all the basic comparisons in C++, like `<`.

2.19 Concurrency

Components for concurrent operations, including threads, mutexes, and condition variables.

2.20 Containers

Collaboration diagram for Containers:



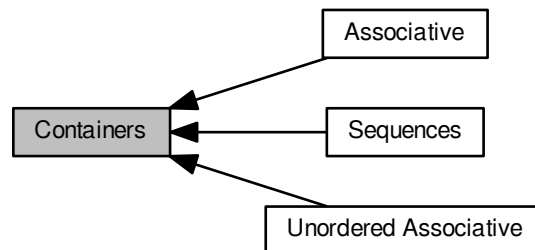
Modules

- [Branch-Based](#)
- [Hash-Based](#)
- [Heap-Based](#)
- [List-Based](#)

2.20.1 Detailed Description

2.21 Containers

Collaboration diagram for Containers:



Modules

- [Associative](#)
- [Sequences](#)
- [Unordered Associative](#)

2.21.1 Detailed Description

Containers are collections of objects.

A container may hold any type which meets certain requirements, but the type of contained object is chosen at compile time, and all objects in a given container must be of the same type. (Polymorphism is possible by declaring a container of pointers to a base class and then populating it with pointers to instances of derived classes. Variant value types such as the `any` class from `Boost` can also be used.

All contained types must be `Assignable` and `CopyConstructible`. Specific containers may place additional requirements on the types of their contained objects.

Containers manage memory allocation and deallocation themselves when storing your objects. The objects are destroyed when the container is itself destroyed. Note that if you are storing pointers in a container, `delete` is *not* automatically called on the pointers before destroying them.

All containers must meet certain requirements, summarized in `tables`.

The standard containers are further refined into [Sequences](#) and [Associative Containers](#). [Unordered Associative Containers](#).

2.22 Data Structure Type

Collaboration diagram for Data Structure Type:



Classes

- struct [__gnu_pbds::associative_tag](#)
- struct [__gnu_pbds::basic_branch_tag](#)
- struct [__gnu_pbds::basic_hash_tag](#)
- struct [__gnu_pbds::binary_heap_tag](#)
- struct [__gnu_pbds::binomial_heap_tag](#)
- struct [__gnu_pbds::cc_hash_tag](#)
- struct [__gnu_pbds::container_tag](#)
- struct [__gnu_pbds::gp_hash_tag](#)
- struct [__gnu_pbds::list_update_tag](#)
- struct [__gnu_pbds::ov_tree_tag](#)
- struct [__gnu_pbds::pairing_heap_tag](#)
- struct [__gnu_pbds::pat_trie_tag](#)
- struct [__gnu_pbds::priority_queue_tag](#)
- struct [__gnu_pbds::rb_tree_tag](#)
- struct [__gnu_pbds::rc_binomial_heap_tag](#)
- struct [__gnu_pbds::sequence_tag](#)
- struct [__gnu_pbds::splay_tree_tag](#)
- struct [__gnu_pbds::string_tag](#)
- struct [__gnu_pbds::thin_heap_tag](#)
- struct [__gnu_pbds::tree_tag](#)
- struct [__gnu_pbds::trie_tag](#)

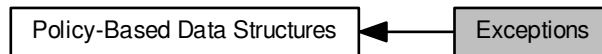
2.22.1 Detailed Description

2.23 Diagnostics

Components for error handling, reporting, and diagnostic operations.

2.24 Exceptions

Collaboration diagram for Exceptions:



Classes

- struct [__gnu_pbds::container_error](#)
- struct [__gnu_pbds::insert_error](#)
- struct [__gnu_pbds::join_error](#)
- struct [__gnu_pbds::resize_error](#)

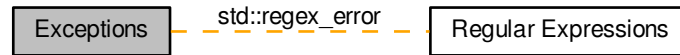
Functions

- void [__gnu_pbds::__throw_container_error\(\)](#)
- void [__gnu_pbds::__throw_insert_error\(\)](#)
- void [__gnu_pbds::__throw_join_error\(\)](#)
- void [__gnu_pbds::__throw_resize_error\(\)](#)

2.24.1 Detailed Description

2.25 Exceptions

Collaboration diagram for Exceptions:



Classes

- class [__cxxabiv1::__forced_unwind](#)
- struct [__gnu_cxx::forced_error](#)
- class [__gnu_cxx::recursive_init_error](#)
- class [std::__exception_ptr::exception_ptr](#)
- class [std::bad_weak_ptr](#)
- class [std::ios_base::failure](#)
- class [std::nested_exception](#)
- class [std::regex_error](#)

Functions

- `template<typename _Ex >`
`const nested_exception * std::__get_nested_exception (const _Ex &__ex)`
- `template<typename _Ex >`
`void std::__throw_with_nested (_Ex &&, const nested_exception *==0) __attribute__\(\(__noreturn__\)\)`
- `template<typename _Ex >`
`void std::__throw_with_nested (_Ex &&,...) __attribute__\(\(__noreturn__\)\)`
- `template<typename _Ex >`
`exception_ptr std::copy_exception (_Ex __ex) noexcept 1`
- `exception_ptr std::current_exception () noexcept`
- `template<typename _Ex >`
`exception_ptr std::make_exception_ptr (_Ex __ex) noexcept`
- `void std::rethrow_exception (exception_ptr) __attribute__\(\(__noreturn__\)\)`
- `template<typename _Ex >`
`void std::rethrow_if_nested (const _Ex &__ex)`
- `void std::rethrow_if_nested (const nested_exception &__ex)`
- `template<typename _Ex >`
`void std::throw_with_nested (_Ex __ex)`

2.25.1 Detailed Description

2.25.2 Function Documentation

2.25.2.1 `template<typename _Ex > exception_ptr std::copy_exception (_Ex __ex) [noexcept]`

Obtain an `exception_ptr` pointing to a copy of the supplied object. This function is deprecated, use `std::make_exception_ptr` instead.

Definition at line 193 of file `exception_ptr.h`.

Referenced by `std::make_exception_ptr()`.

2.25.2.2 `exception_ptr std::current_exception () [noexcept]`

Obtain an `exception_ptr` to the currently handled exception. If there is none, or the currently handled exception is foreign, return the null value.

Referenced by `std::make_exception_ptr()`.

2.25.2.3 `template<typename _Ex > exception_ptr std::make_exception_ptr (_Ex __ex) [noexcept]`

Obtain an `exception_ptr` pointing to a copy of the supplied object.

Definition at line 169 of file `exception_ptr.h`.

References `std::copy_exception()`, and `std::current_exception()`.

2.25.2.4 `void std::rethrow_exception (exception_ptr)`

Throw the object pointed to by the `exception_ptr`.

2.25.2.5 `template<typename _Ex > void std::rethrow_if_nested (const _Ex & __ex) [inline]`

If `__ex` is derived from `nested_exception`, `__ex.rethrow_nested()`.

Definition at line 146 of file `nested_exception.h`.

2.25.2.6 `void std::rethrow_if_nested (const nested_exception & __ex) [inline]`

Overload, See N2619.

Definition at line 154 of file `nested_exception.h`.

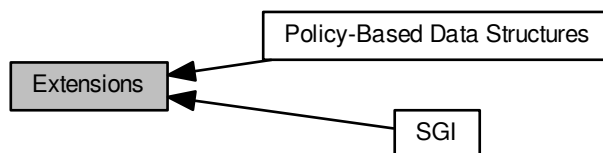
2.25.2.7 `template<typename _Ex > void std::throw_with_nested (_Ex __ex) [inline]`

If `__ex` is derived from `nested_exception`, `__ex`. Else, an implementation-defined object derived from both.

Definition at line 136 of file `nested_exception.h`.

2.26 Extensions

Collaboration diagram for Extensions:



Modules

- [Policy-Based Data Structures](#)
- [SGI](#)

Classes

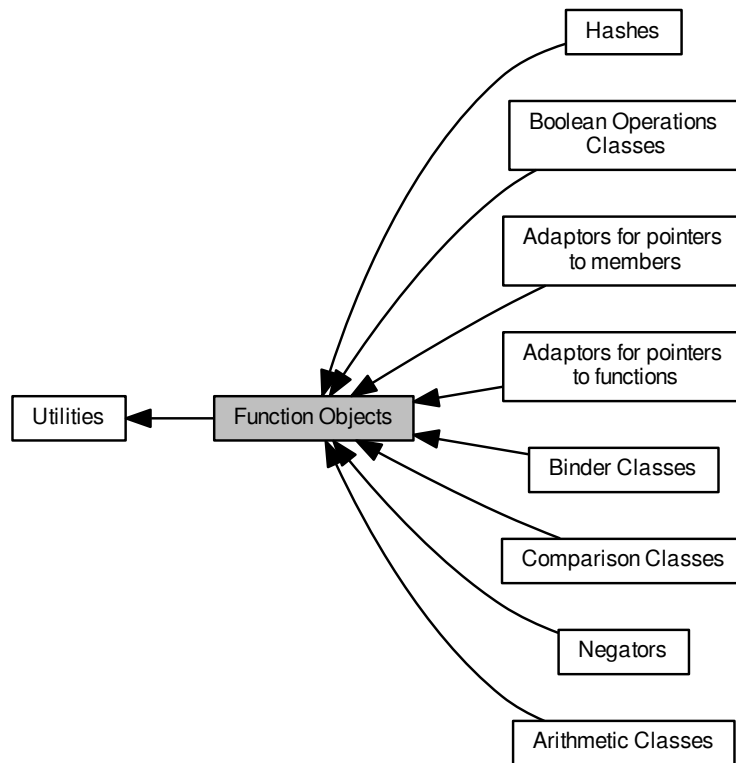
- [class `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>`](#)

2.26.1 Detailed Description

Components generally useful that are not part of any standard.

2.27 Function Objects

Collaboration diagram for Function Objects:



Modules

- [Adaptors for pointers to functions](#)
- [Adaptors for pointers to members](#)
- [Arithmetic Classes](#)
- [Binder Classes](#)
- [Boolean Operations Classes](#)
- [Comparison Classes](#)
- [Hashes](#)
- [Negators](#)

Classes

- `struct std::binary_function< _Arg1, _Arg2, _Result >`
- `struct std::unary_function< _Arg, _Result >`

2.27.1 Detailed Description

Function objects, or *functors*, are objects with an `operator()` defined and accessible. They can be passed as arguments to algorithm templates and used in place of a function pointer. Not only is the resulting expressiveness of the library increased, but the generated code can be more efficient than what you might write by hand. When we refer to *functors*, then, generally we include function pointers in the description as well.

Often, functors are only created as temporaries passed to algorithm calls, rather than being created as named variables.

Two examples taken from the standard itself follow. To perform a by-element addition of two vectors `a` and `b` containing `double`, and put the result in `a`, use

```
transform (a.begin(), a.end(), b.begin(), a.begin(), plus<double>());
```

To negate every element in `a`, use

```
transform(a.begin(), a.end(), a.begin(), negate<double>());
```

The addition and negation functions will be inlined directly.

The standard functors are derived from structs named `unary_function` and `binary_function`. These two classes contain nothing but typedefs, to aid in generic (template) programming. If you write your own functors, you might consider doing the same.

2.28 Hash-Based

Collaboration diagram for Hash-Based:



Modules

- [Base and Policy Classes](#)

Classes

- `class __gnu_pbds::basic_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Resize_Policy, Store_Hash, Tag, Policy_Ti, _Alloc >`
- `class __gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc >`
- `class __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >`

Macros

- `#define PB_DS_CC_HASH_BASE`
- `#define PB_DS_GP_HASH_BASE`
- `#define PB_DS_HASH_BASE`

2.28.1 Detailed Description

2.29 Hashes

Collaboration diagram for Hashes:



Classes

- struct `std::hash< _Tp >`
- struct `std::hash< _Tp * >`
- struct `std::hash< bool >`
- struct `std::hash< char >`
- struct `std::hash< char16_t >`
- struct `std::hash< char32_t >`
- struct `std::hash< double >`
- struct `std::hash< float >`
- struct `std::hash< int >`
- struct `std::hash< long >`
- struct `std::hash< long double >`
- struct `std::hash< long long >`
- struct `std::hash< short >`
- struct `std::hash< signed char >`
- struct `std::hash< unsigned char >`
- struct `std::hash< unsigned int >`
- struct `std::hash< unsigned long >`
- struct `std::hash< unsigned long long >`
- struct `std::hash< unsigned short >`
- struct `std::hash< wchar_t >`

Macros

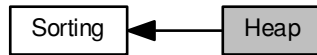
- `#define _Cxx_hashtable_define_trivial_hash(_Tp)`

2.29.1 Detailed Description

Hashing functors taking a variable type and returning a `std::size_t`.

2.30 Heap

Collaboration diagram for Heap:



Functions

- `template<typename _RandomAccessIterator >`
`bool std::is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator , typename _Compare >`
`bool std::is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`_RandomAccessIterator std::is_heap_until (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator , typename _Compare >`
`_RandomAccessIterator std::is_heap_until (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`void std::make_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator , typename _Compare >`
`void std::make_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`void std::pop_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator , typename _Compare >`
`void std::pop_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`void std::push_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator , typename _Compare >`
`void std::push_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`void std::sort_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator , typename _Compare >`
`void std::sort_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`

2.30.1 Detailed Description

2.30.2 Function Documentation

2.30.2.1 `template<typename _RandomAccessIterator > bool std::is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last) [inline]`

Determines whether a range is a heap.

Parameters

<code>__first</code>	Start of range.
<code>__last</code>	End of range.

Returns

True if range is a heap, false otherwise.

Definition at line 511 of file `stl_heap.h`.

References `std::is_heap_until()`.

```
2.30.2.2 template<typename _RandomAccessIterator, typename _Compare> bool std::is_heap ( _RandomAccessIterator __first,  
_RandomAccessIterator __last, _Compare __comp ) [inline]
```

Determines whether a range is a heap using comparison functor.

Parameters

<code>__first</code>	Start of range.
<code>__last</code>	End of range.
<code>__comp</code>	Comparison functor to use.

Returns

True if range is a heap, false otherwise.

Definition at line 524 of file `stl_heap.h`.

References `std::is_heap_until()`.

```
2.30.2.3 template<typename _RandomAccessIterator> _RandomAccessIterator std::is_heap_until ( _RandomAccessIterator  
__first, _RandomAccessIterator __last ) [inline]
```

Search the end of a heap.

Parameters

<code>__first</code>	Start of range.
<code>__last</code>	End of range.

Returns

An iterator pointing to the first element not in the heap.

This operation returns the last iterator *i* in [*__first*, *__last*) for which the range [*__first*, *i*) is a heap.

Definition at line 462 of file `stl_heap.h`.

References `std::distance()`.

2.30.2.4 `template<typename _RandomAccessIterator, typename _Compare > _RandomAccessIterator std::is_heap_until (`
`_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp) [inline]`

Search the end of a heap using comparison functor.

Parameters

<i>__first</i>	Start of range.
<i>__last</i>	End of range.
<i>__comp</i>	Comparison functor to use.

Returns

An iterator pointing to the first element not in the heap.

This operation returns the last iterator *i* in [*__first*, *__last*) for which the range [*__first*, *i*) is a heap. Comparisons are made using *__comp*.

Definition at line 489 of file `stl_heap.h`.

References `std::distance()`.

Referenced by `std::is_heap()`.

2.30.2.5 `template<typename _RandomAccessIterator > void std::make_heap (_RandomAccessIterator __first,`
`_RandomAccessIterator __last) [inline]`

Construct a heap over a range.

Parameters

<i>__first</i>	Start of heap.
<i>__last</i>	End of heap.

This operation makes the elements in [*__first*, *__last*) into a heap.

Definition at line 351 of file `stl_heap.h`.

2.30.2.6 `template<typename _RandomAccessIterator, typename _Compare > void std::make_heap (_RandomAccessIterator`
`__first, _RandomAccessIterator __last, _Compare __comp) [inline]`

Construct a heap over a range using comparison functor.

Parameters

<code>__first</code>	Start of heap.
<code>__last</code>	End of heap.
<code>__comp</code>	Comparison functor to use.

This operation makes the elements in `[__first,__last)` into a heap. Comparisons are made using `__comp`.

Definition at line 376 of file `stl_heap.h`.

Referenced by `std::priority_queue<_Tp, _Sequence, _Compare >::priority_queue()`.

2.30.2.7 `template<typename _RandomAccessIterator > void std::pop_heap (_RandomAccessIterator __first, _RandomAccessIterator __last) [inline]`

Pop an element off a heap.

Parameters

<code>__first</code>	Start of heap.
<code>__last</code>	End of heap.

Precondition

`[__first, __last)` is a valid, non-empty range.

This operation pops the top of the heap. The elements `__first` and `__last-1` are swapped and `[__first,__last-1)` is made into a heap.

Definition at line 263 of file `stl_heap.h`.

2.30.2.8 `template<typename _RandomAccessIterator, typename _Compare > void std::pop_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp) [inline]`

Pop an element off a heap using comparison functor.

Parameters

<code>__first</code>	Start of heap.
<code>__last</code>	End of heap.
<code>__comp</code>	Comparison functor to use.

This operation pops the top of the heap. The elements `__first` and `__last-1` are swapped and `[__first,__last-1)` is made into a heap. Comparisons are made using `comp`.

Definition at line 297 of file `stl_heap.h`.

Referenced by `std::priority_queue<_Tp, _Sequence, _Compare >::pop()`.

2.30.2.9 `template<typename _RandomAccessIterator > void std::push_heap (_RandomAccessIterator __first,
_RandomAccessIterator __last) [inline]`

Push an element onto a heap.

Parameters

<code>__first</code>	Start of heap.
<code>__last</code>	End of heap + element.

This operation pushes the element at `last-1` onto the valid heap over the range `[__first,__last-1)`. After completion, `[__first,__last)` is a valid heap.

Definition at line 150 of file `stl_heap.h`.

2.30.2.10 `template<typename _RandomAccessIterator , typename _Compare > void std::push_heap (_RandomAccessIterator
__first, _RandomAccessIterator __last, _Compare __comp) [inline]`

Push an element onto a heap using comparison functor.

Parameters

<code>__first</code>	Start of heap.
<code>__last</code>	End of heap + element.
<code>__comp</code>	Comparison functor.

This operation pushes the element at `__last-1` onto the valid heap over the range `[__first,__last-1)`. After completion, `[__first,__last)` is a valid heap. Compare operations are performed using `comp`.

Definition at line 184 of file `stl_heap.h`.

Referenced by `std::priority_queue<_Tp, _Sequence, _Compare >::push()`.

2.30.2.11 `template<typename _RandomAccessIterator > void std::sort_heap (_RandomAccessIterator __first,
_RandomAccessIterator __last) [inline]`

Sort a heap.

Parameters

<code>__first</code>	Start of heap.
<code>__last</code>	End of heap.

This operation sorts the valid heap in the range `[__first,__last)`.

Definition at line 410 of file `stl_heap.h`.

2.30.2.12 `template<typename _RandomAccessIterator, typename _Compare> void std::sort_heap (_RandomAccessIterator
__first, _RandomAccessIterator __last, _Compare __comp) [inline]`

Sort a heap using comparison functor.

Parameters

<code>__first</code>	Start of heap.
<code>__last</code>	End of heap.
<code>__comp</code>	Comparison functor to use.

This operation sorts the valid heap in the range [`__first`,`__last`). Comparisons are made using `__comp`.

Definition at line 436 of file `stl_heap.h`.

2.31 Heap-Based

Collaboration diagram for Heap-Based:



Modules

- [Base and Policy Classes](#)

Classes

- class [__gnu_pbds::priority_queue<_Tv, Cmp_Fn, Tag, _Alloc>](#)

Typedefs

- typedef `_Alloc` [__gnu_pbds::priority_queue<_Tv, Cmp_Fn, Tag, _Alloc>::allocator_type](#)
- typedef `Cmp_Fn` [__gnu_pbds::priority_queue<_Tv, Cmp_Fn, Tag, _Alloc>::cmp_fn](#)
- typedef `base_type::const_iterator` [__gnu_pbds::priority_queue<_Tv, Cmp_Fn, Tag, _Alloc>::const_iterator](#)
- typedef `__rebind_va::const_pointer` [__gnu_pbds::priority_queue<_Tv, Cmp_Fn, Tag, _Alloc>::const_pointer](#)
- typedef `__rebind_va::const_reference` [__gnu_pbds::priority_queue<_Tv, Cmp_Fn, Tag, _Alloc>::const_reference](#)
- typedef `Tag` [__gnu_pbds::priority_queue<_Tv, Cmp_Fn, Tag, _Alloc>::container_category](#)
- typedef `allocator_type::difference_type` [__gnu_pbds::priority_queue<_Tv, Cmp_Fn, Tag, _Alloc>::difference_type](#)
- typedef `base_type::iterator` [__gnu_pbds::priority_queue<_Tv, Cmp_Fn, Tag, _Alloc>::iterator](#)
- typedef `base_type::point_const_iterator` [__gnu_pbds::priority_queue<_Tv, Cmp_Fn, Tag, _Alloc>::point_const_iterator](#)
- typedef `base_type::point_iterator` [__gnu_pbds::priority_queue<_Tv, Cmp_Fn, Tag, _Alloc>::point_iterator](#)
- typedef `__rebind_va::pointer` [__gnu_pbds::priority_queue<_Tv, Cmp_Fn, Tag, _Alloc>::pointer](#)
- typedef `__rebind_va::reference` [__gnu_pbds::priority_queue<_Tv, Cmp_Fn, Tag, _Alloc>::reference](#)
- typedef `allocator_type::size_type` [__gnu_pbds::priority_queue<_Tv, Cmp_Fn, Tag, _Alloc>::size_type](#)
- typedef `_Tv` [__gnu_pbds::priority_queue<_Tv, Cmp_Fn, Tag, _Alloc>::value_type](#)

Functions

- [__gnu_pbds::priority_queue<_Tv, Cmp_Fn, Tag, _Alloc >::priority_queue](#) (const cmp_fn &r_cmp_fn)
- [template<typename It > __gnu_pbds::priority_queue<_Tv, Cmp_Fn, Tag, _Alloc >::priority_queue](#) (It first_it, It last_it)
- [template<typename It > __gnu_pbds::priority_queue<_Tv, Cmp_Fn, Tag, _Alloc >::priority_queue](#) (It first_it, It last_it, const cmp_fn &r_cmp_fn)
- [__gnu_pbds::priority_queue<_Tv, Cmp_Fn, Tag, _Alloc >::priority_queue](#) (const priority_queue &other)
- [priority_queue & __gnu_pbds::priority_queue<_Tv, Cmp_Fn, Tag, _Alloc >::operator=](#) (const priority_queue &other)
- [void __gnu_pbds::priority_queue<_Tv, Cmp_Fn, Tag, _Alloc >::swap](#) (priority_queue &other)

2.31.1 Detailed Description

2.31.2 Function Documentation

2.31.2.1 `template<typename _Tv, typename Cmp_Fn = std::less<_Tv>, typename Tag = pairing_heap_tag, typename _Alloc = std::allocator<char>> __gnu_pbds::priority_queue<_Tv, Cmp_Fn, Tag, _Alloc >::priority_queue (const cmp_fn & r_cmp_fn) [inline]`

Constructor taking some policy objects. r_cmp_fn will be copied by the Cmp_Fn object of the container object.

Definition at line 116 of file priority_queue.hpp.

2.31.2.2 `template<typename _Tv, typename Cmp_Fn = std::less<_Tv>, typename Tag = pairing_heap_tag, typename _Alloc = std::allocator<char>> template<typename It > __gnu_pbds::priority_queue<_Tv, Cmp_Fn, Tag, _Alloc >::priority_queue (It first_it, It last_it) [inline]`

Constructor taking __iterators to a range of value_types. The value_types between first_it and last_it will be inserted into the container object.

Definition at line 122 of file priority_queue.hpp.

2.31.2.3 `template<typename _Tv, typename Cmp_Fn = std::less<_Tv>, typename Tag = pairing_heap_tag, typename _Alloc = std::allocator<char>> template<typename It > __gnu_pbds::priority_queue<_Tv, Cmp_Fn, Tag, _Alloc >::priority_queue (It first_it, It last_it, const cmp_fn & r_cmp_fn) [inline]`

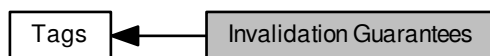
Constructor taking __iterators to a range of value_types and some policy objects The value_types between first_it and last_it will be inserted into the container object. r_cmp_fn will be copied by the cmp_fn object of the container object.

Definition at line 130 of file priority_queue.hpp.

References `std::swap()`.

2.32 Invalidation Guarantees

Collaboration diagram for Invalidation Guarantees:



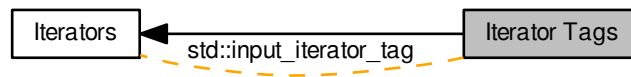
Classes

- struct [__gnu_pbds::basic_invalidation_guarantee](#)
- struct [__gnu_pbds::point_invalidation_guarantee](#)
- struct [__gnu_pbds::range_invalidation_guarantee](#)

2.32.1 Detailed Description

2.33 Iterator Tags

Collaboration diagram for Iterator Tags:



Classes

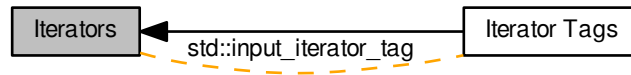
- struct [std::bidirectional_iterator_tag](#)
- struct [std::forward_iterator_tag](#)
- struct [std::input_iterator_tag](#)
- struct [std::output_iterator_tag](#)
- struct [std::random_access_iterator_tag](#)

2.33.1 Detailed Description

These are empty types, used to distinguish different iterators. The distinction is not made by what they contain, but simply by what they are. Different underlying algorithms can then be used based on the different operations supported by different iterator types.

2.34 Iterators

Collaboration diagram for Iterators:



Modules

- [Iterator Tags](#)

Classes

- class [std::__has_iterator_category_helper< _Tp >](#)
- class [std::back_insert_iterator< _Container >](#)
- class [std::front_insert_iterator< _Container >](#)
- struct [std::input_iterator_tag](#)
- class [std::insert_iterator< _Container >](#)
- class [std::istream_iterator< _Tp, _CharT, _Traits, _Dist >](#)
- class [std::istreambuf_iterator< _CharT, _Traits >](#)
- struct [std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >](#)
- struct [std::iterator_traits< _Tp * >](#)
- struct [std::iterator_traits< const _Tp * >](#)
- class [std::move_iterator< _Iterator >](#)
- class [std::ostream_iterator< _Tp, _CharT, _Traits >](#)
- class [std::ostreambuf_iterator< _CharT, _Traits >](#)
- class [std::reverse_iterator< _Iterator >](#)

Functions

- template<bool _IsMove, typename _CharT >
[__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT > >::__type std::__copy_move_a2](#) ([_CharT *](#)__first, [_CharT *](#)__last, [ostreambuf_iterator< _CharT >](#) __result)
- template<bool _IsMove, typename _CharT >
[__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT > >::__type std::__copy_move_a2](#) ([const _CharT *](#)__first, [const _CharT *](#)__last, [ostreambuf_iterator< _CharT >](#) __result)
- template<bool _IsMove, typename _CharT >
[__gnu_cxx::__enable_if< __is_char< _CharT >::__value, _CharT * >::__type std::__copy_move_a2](#) ([istreambuf_iterator< _CharT >](#) __first, [istreambuf_iterator< _CharT >](#) __last, [_CharT *](#)__result)
- template<typename _Iter >
[iterator_traits< _Iter >::iterator_category std::__iterator_category](#) ([const _Iter &](#))

- `template<typename _Iterator, typename _ReturnType = typename conditional<__move_if_noexcept_cond <typename iterator_traits<_Iterator>::value_type>::value, _Iterator, move_iterator<_Iterator>>::type>
_ReturnType std::make_move_if_noexcept_iterator (_Iterator __i)`
- `template<typename _Container >
back_insert_iterator< _Container > std::back_inserter (_Container &__x)`
- `template<typename _CharT >
__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT > >::__type std::copy
(istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, ostreambuf_iterator< _CharT >
__result)`
- `template<typename _CharT >
__gnu_cxx::__enable_if< __is_char< _CharT >::__value, istreambuf_iterator< _CharT > >::__type std::find
(istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, const _CharT &__val)`
- `template<typename _Container >
front_insert_iterator< _Container > std::front_inserter (_Container &__x)`
- `template<typename _Container, typename _Iterator >
insert_iterator< _Container > std::inserter (_Container &__x, _Iterator __i)`
- `template<typename _Iterator >
move_iterator< _Iterator > std::make_move_iterator (_Iterator __i)`
- `template<class _Tp, class _CharT, class _Traits, class _Dist >
bool std::operator!= (const istream_iterator< _Tp, _CharT, _Traits, _Dist > &__x, const istream_iterator< _Tp,
_CharT, _Traits, _Dist > &__y)`
- `template<typename _CharT, typename _Traits >
bool std::operator!= (const istreambuf_iterator< _CharT, _Traits > &__a, const istreambuf_iterator< _CharT,
_Traits > &__b)`
- `template<typename _Iterator >
bool std::operator!= (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR >
bool std::operator!= (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR >
bool std::operator!= (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`
- `template<typename _Iterator >
bool std::operator!= (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)`
- `template<typename _Iterator >
reverse_iterator< _Iterator > std::operator+ (typename reverse_iterator< _Iterator >::difference_type __n,
const reverse_iterator< _Iterator > &__x)`
- `template<typename _Iterator >
move_iterator< _Iterator > std::operator+ (typename move_iterator< _Iterator >::difference_type __n, const
move_iterator< _Iterator > &__x)`
- `template<typename _Iterator >
reverse_iterator< _Iterator >::difference_type std::operator- (const reverse_iterator< _Iterator > &__x, const
reverse_iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR >
auto std::operator- (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y) ->
decltype(__y.base()-__x.base())`
- `template<typename _IteratorL, typename _IteratorR >
auto std::operator- (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y) ->
decltype(__x.base()-__y.base())`
- `template<typename _Iterator >
auto std::operator- (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y) ->
decltype(__x.base()-__y.base())`
- `template<typename _Iterator >
bool std::operator< (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`

- `template<typename _IteratorL, typename _IteratorR >`
`bool std::operator< (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool std::operator< (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`
- `template<typename _Iterator >`
`bool std::operator< (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)`
- `template<typename _Iterator >`
`bool std::operator<= (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool std::operator<= (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool std::operator<= (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`
- `template<typename _Iterator >`
`bool std::operator<= (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)`
- `template<typename _Tp, typename _CharT, typename _Traits, typename _Dist >`
`bool std::operator== (const istream_iterator< _Tp, _CharT, _Traits, _Dist > &__x, const istream_iterator< _Tp, _CharT, _Traits, _Dist > &__y)`
- `template<typename _CharT, typename _Traits >`
`bool std::operator== (const istreambuf_iterator< _CharT, _Traits > &__a, const istreambuf_iterator< _CharT, _Traits > &__b)`
- `template<typename _Iterator >`
`bool std::operator== (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool std::operator== (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool std::operator== (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`
- `template<typename _Iterator >`
`bool std::operator== (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)`
- `template<typename _Iterator >`
`bool std::operator> (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool std::operator> (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool std::operator> (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`
- `template<typename _Iterator >`
`bool std::operator> (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)`
- `template<typename _Iterator >`
`bool std::operator>= (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool std::operator>= (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool std::operator>= (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`
- `template<typename _Iterator >`
`bool std::operator>= (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)`

2.34.1 Detailed Description

Abstractions for uniform iterating through various underlying types.

2.34.2 Function Documentation

2.34.2.1 `template<typename _Iter > iterator_traits<_Iter>::iterator_category std::__iterator_category (const _Iter &)`
`[inline]`

This function is not a part of the C++ standard but is syntactic sugar for internal library use only.

Definition at line 201 of file `stl_iterator_base_types.h`.

Referenced by `__gnu_debug::__check_string()`, `std::__find_if()`, `std::__find_if_not()`, `std::__search_n_aux()`, `__gnu_debug::__valid_range_aux()`, `std::advance()`, `std::copy_n()`, `std::distance()`, `std::find_end()`, `std::includes()`, `std::next_permutation()`, `std::partition()`, `std::reverse()`, `std::rotate()`, `std::uninitialized_copy_n()`, and `std::unique_copy()`.

2.34.2.2 `template<typename _Container > back_insert_iterator<_Container> std::back_inserter (_Container & __x)`
`[inline]`

Parameters

<code>__x</code>	A container of arbitrary type.
------------------	--------------------------------

Returns

An instance of `back_insert_iterator` working on `__x`.

This wrapper function helps in creating `back_insert_iterator` instances. Typing the name of the iterator requires knowing the precise full type of the container, which can be tedious and impedes generic programming. Using this function lets you take advantage of automatic template parameter deduction, making the compiler match the correct types for you.

Definition at line 480 of file `stl_iterator.h`.

Referenced by `std::match_results<_Bi_iter >::format()`, and `std::regex_replace()`.

2.34.2.3 `template<typename _Container > front_insert_iterator<_Container> std::front_inserter (_Container & __x)`
`[inline]`

Parameters

<code>__x</code>	A container of arbitrary type.
------------------	--------------------------------

Returns

An instance of `front_insert_iterator` working on `x`.

This wrapper function helps in creating `front_insert_iterator` instances. Typing the name of the iterator requires knowing the precise full type of the container, which can be tedious and impedes generic programming. Using this function lets you take advantage of automatic template parameter deduction, making the compiler match the correct types for you.

Definition at line 570 of file `stl_iterator.h`.

2.34.2.4 `template<typename _Container, typename _Iterator> insert_iterator<_Container> std::inserter (_Container & __x, _Iterator __i) [inline]`

Parameters

<code>__x</code>	A container of arbitrary type.
------------------	--------------------------------

Returns

An instance of `insert_iterator` working on `__x`.

This wrapper function helps in creating `insert_iterator` instances. Typing the name of the iterator requires knowing the precise full type of the container, which can be tedious and impedes generic programming. Using this function lets you take advantage of automatic template parameter deduction, making the compiler match the correct types for you.

Definition at line 684 of file `stl_iterator.h`.

References `std::reverse_iterator<_Iterator>::base()`, `std::move()`, `std::reverse_iterator<_Iterator>::operator*()`, `std::reverse_iterator<_Iterator>::operator+()`, `std::reverse_iterator<_Iterator>::operator++()`, `std::reverse_iterator<_Iterator>::operator+=()`, `std::reverse_iterator<_Iterator>::operator-()`, `std::reverse_iterator<_Iterator>::operator--()`, `std::reverse_iterator<_Iterator>::operator-=()`, `std::reverse_iterator<_Iterator>::operator->()`, `std::operator>()`, `std::operator>=()`, and `std::reverse_iterator<_Iterator>::operator[]()`.

2.34.2.5 `template<class _Tp, class _CharT, class _Traits, class _Dist> bool std::operator!=(const istream_iterator<_Tp, _CharT, _Traits, _Dist> & __x, const istream_iterator<_Tp, _CharT, _Traits, _Dist> & __y) [inline]`

Return false if `x` and `y` are both end or not end, or `x` and `y` are the same.

Definition at line 137 of file `stream_iterator.h`.

2.34.2.6 `template<typename _Tp, typename _CharT, typename _Traits, typename _Dist> bool std::operator==(const istream_iterator<_Tp, _CharT, _Traits, _Dist> & __x, const istream_iterator<_Tp, _CharT, _Traits, _Dist> & __y) [inline]`

Return true if `x` and `y` are both end or not end, or `x` and `y` are the same.

Definition at line 130 of file `stream_iterator.h`.

2.34.2.7 `template<typename _Iterator> bool std::operator==(const reverse_iterator<_Iterator> & __x, const reverse_iterator<_Iterator> & __y) [inline]`

Parameters

<code>__x</code>	A <code>reverse_iterator</code> .
<code>__y</code>	A <code>reverse_iterator</code> .

Returns

A simple bool.

Reverse iterators forward many operations to their underlying `base()` iterators. Others are implemented in terms of one another.

Definition at line 292 of file `stl_iterator.h`.

References `std::reverse_iterator<_Iterator>::base()`, `std::reverse_iterator<_Iterator>::operator+()`, `std::reverse_iterator<_Iterator>::operator-()`, `std::operator>()`, and `std::operator>=()`.

2.35 List-Based

Collaboration diagram for List-Based:



Classes

- class `__gnu_pbds::list_update< Key, Mapped, Eq_Fn, Update_Policy, _Alloc >`

Macros

- `#define PB_DS_LU_BASE`

2.35.1 Detailed Description

2.36 Locales

Classes

- class `std::codecvt<_InternT, _ExternT, _StateT >`
- class `std::ctype<_CharT >`
- class `std::ctype< char >`
- class `std::ctype< wchar_t >`
- class `std::locale`
- class `std::locale::facet`
- class `std::locale::id`
- class `std::messages<_CharT >`
- struct `std::messages_base`
- class `std::money_base`
- class `std::money_get<_CharT, _InIter >`
- class `std::money_put<_CharT, _OutIter >`
- class `std::moneypunct<_CharT, _Intl >`
- class `std::num_get<_CharT, _InIter >`
- class `std::num_put<_CharT, _OutIter >`
- class `std::numpunct<_CharT >`
- class `std::time_base`
- class `std::time_get<_CharT, _InIter >`
- class `std::time_put<_CharT, _OutIter >`

2.36.1 Detailed Description

Classes and functions for internationalization and localization.

2.37 Mutating

Collaboration diagram for Mutating:



Functions

- `template<typename _II, typename _OI >`
`_OI std::copy (_II __first, _II __last, _OI __result)`
- `template<typename _BI1, typename _BI2 >`
`_BI2 std::copy_backward (_BI1 __first, _BI1 __last, _BI2 __result)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate >`
`_OutputIterator std::copy_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred)`
- `template<typename _InputIterator, typename _Size, typename _OutputIterator >`
`_OutputIterator std::copy_n (_InputIterator __first, _Size __n, _OutputIterator __result)`
- `template<typename _ForwardIterator, typename _Tp >`
`void std::fill (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)`
- `template<typename _OI, typename _Size, typename _Tp >`
`_OI std::fill_n (_OI __first, _Size __n, const _Tp &__value)`
- `template<typename _ForwardIterator, typename _Generator >`
`void std::generate (_ForwardIterator __first, _ForwardIterator __last, _Generator __gen)`
- `template<typename _OutputIterator, typename _Size, typename _Generator >`
`_OutputIterator std::generate_n (_OutputIterator __first, _Size __n, _Generator __gen)`
- `template<typename _InputIterator, typename _Predicate >`
`bool std::is_partitioned (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`
`void std::iter_swap (_ForwardIterator1 __a, _ForwardIterator2 __b)`
- `template<typename _II, typename _OI >`
`_OI std::move (_II __first, _II __last, _OI __result)`
- `template<typename _BI1, typename _BI2 >`
`_BI2 std::move_backward (_BI1 __first, _BI1 __last, _BI2 __result)`
- `template<typename _ForwardIterator, typename _Predicate >`
`_ForwardIterator std::partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _OutputIterator1, typename _OutputIterator2, typename _Predicate >`
`pair< _OutputIterator1, _OutputIterator2 > std::partition_copy (_InputIterator __first, _InputIterator __last, _OutputIterator1 __out_true, _OutputIterator2 __out_false, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Predicate >`
`_ForwardIterator std::partition_point (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _RandomAccessIterator >`
`void std::random_shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last)`

- `template<typename _RandomAccessIterator, typename _RandomNumberGenerator >`
`void std::random_shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last, _RandomNumberGenerator &&__rand)`
- `template<typename _ForwardIterator, typename _Tp >`
`_ForwardIterator std::remove (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Tp >`
`_OutputIterator std::remove_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, const _Tp &__value)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate >`
`_OutputIterator std::remove_copy_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Predicate >`
`_ForwardIterator std::remove_if (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Tp >`
`void std::replace (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__old_value, const _Tp &__new_value)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate, typename _Tp >`
`_OutputIterator std::replace_copy_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred, const _Tp &__new_value)`
- `template<typename _ForwardIterator, typename _Predicate, typename _Tp >`
`void std::replace_if (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, const _Tp &__new_value)`
- `template<typename _BidirectionalIterator >`
`void std::reverse (_BidirectionalIterator __first, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _OutputIterator >`
`_OutputIterator std::reverse_copy (_BidirectionalIterator __first, _BidirectionalIterator __last, _OutputIterator __result)`
- `template<typename _ForwardIterator >`
`void std::rotate (_ForwardIterator __first, _ForwardIterator __middle, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _OutputIterator >`
`_OutputIterator std::rotate_copy (_ForwardIterator __first, _ForwardIterator __middle, _ForwardIterator __last, _OutputIterator __result)`
- `template<typename _RandomAccessIterator, typename _UniformRandomNumberGenerator >`
`void std::shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last, _UniformRandomNumberGenerator &&__g)`
- `template<typename _ForwardIterator, typename _Predicate >`
`_ForwardIterator std::stable_partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`
`_ForwardIterator2 std::swap_ranges (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2)`
- `template<typename _InputIterator, typename _OutputIterator, typename _UnaryOperation >`
`_OutputIterator std::transform (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _UnaryOperation __unary_op)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _BinaryOperation >`
`_OutputIterator std::transform (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _OutputIterator __result, _BinaryOperation __binary_op)`
- `template<typename _ForwardIterator >`
`_ForwardIterator std::unique (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _BinaryPredicate >`
`_ForwardIterator std::unique (_ForwardIterator __first, _ForwardIterator __last, _BinaryPredicate __binary_pred)`
- `template<typename _InputIterator, typename _OutputIterator >`
`_OutputIterator std::unique_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result)`

- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryPredicate > _OutputIterator std::unique_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryPredicate __binary_pred)`

2.37.1 Detailed Description

2.37.2 Function Documentation

2.37.2.1 `template<typename _II, typename _OI > _OI std::copy (_II __first, _II __last, _OI __result) [inline]`

Copies the range [first,last) into result.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__result</code>	An output iterator.

Returns

`result + (first - last)`

This inline function will boil down to a call to `memmove` whenever possible. Failing that, if random access iterators are passed, then the loop count will be known (and therefore a candidate for compiler optimizations such as unrolling). Result may not be contained within [first,last); the `copy_backward` function should be used instead.

Note that the end of the output range is permitted to be contained within [first,last).

Definition at line 458 of file `stl_algobase.h`.

2.37.2.2 `template<typename _BI1, typename _BI2 > _BI2 std::copy_backward (_BI1 __first, _BI1 __last, _BI2 __result) [inline]`

Copies the range [first,last) into result.

Parameters

<code>__first</code>	A bidirectional iterator.
<code>__last</code>	A bidirectional iterator.
<code>__result</code>	A bidirectional iterator.

Returns

`result - (first - last)`

The function has the same effect as `copy`, but starts at the end of the range and works its way to the start, returning the start of the result. This inline function will boil down to a call to `memmove` whenever possible. Failing that, if random

access iterators are passed, then the loop count will be known (and therefore a candidate for compiler optimizations such as unrolling).

Result may not be in the range `(first,last]`. Use `copy` instead. Note that the start of the output range may overlap `[first,last)`.

Definition at line 634 of file `stl_algobase.h`.

2.37.2.3 `template<typename _Inputiterator , typename _Outputiterator , typename _Predicate > _Outputiterator std::copy_if (`
`_Inputiterator __first, _Inputiterator __last, _Outputiterator __result, _Predicate __pred)`

Copy the elements of a sequence for which a predicate is true.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__result</code>	An output iterator.
<code>__pred</code>	A predicate.

Returns

An iterator designating the end of the resulting sequence.

Copies each element in the range `[__first,__last)` for which `__pred` returns true to the range beginning at `__result`.

`copy_if()` is stable, so the relative order of elements that are copied is unchanged.

Definition at line 734 of file `stl_algo.h`.

2.37.2.4 `template<typename _Inputiterator , typename _Size , typename _Outputiterator > _Outputiterator std::copy_n (`
`_Inputiterator __first, _Size __n, _Outputiterator __result) [inline]`

Copies the range `[first,first+n)` into `[result,result+n)`.

Parameters

<code>__first</code>	An input iterator.
<code>__n</code>	The number of elements to copy.
<code>__result</code>	An output iterator.

Returns

`result+n`.

This inline function will boil down to a call to `memmove` whenever possible. Failing that, if random access iterators are passed, then the loop count will be known (and therefore a candidate for compiler optimizations such as unrolling).

Definition at line 796 of file `stl_algo.h`.

References `std::__iterator_category()`.

2.37.2.5 `template<typename _ForwardIterator, typename _Tp> void std::fill (_ForwardIterator __first, _ForwardIterator __last, const _Tp & __value) [inline]`

Fills the range `[first,last)` with copies of `value`.

Parameters

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.
<code>__value</code>	A reference-to-const of arbitrary type.

Returns

Nothing.

This function fills a range with copies of the same value. For char types filling contiguous areas of memory, this becomes an inline call to `memset` or `wmemset`.

Definition at line 736 of file `stl_algobase.h`.

2.37.2.6 `template<typename _OI, typename _Size, typename _Tp> _OI std::fill_n (_OI __first, _Size __n, const _Tp & __value) [inline]`

Fills the range `[first,first+n)` with copies of `value`.

Parameters

<code>__first</code>	An output iterator.
<code>__n</code>	The count of copies to perform.
<code>__value</code>	A reference-to-const of arbitrary type.

Returns

The iterator at `first+n`.

This function fills a range with copies of the same value. For char types filling contiguous areas of memory, this becomes an inline call to `memset` or `@ wmemset`.

`_GLIBCXX_RESOLVE_LIB_DEFECTS` DR 865. More algorithms that throw away information

Definition at line 796 of file `stl_algobase.h`.

References `std::advance()`, `std::distance()`, `std::equal()`, and `std::min()`.

Referenced by `std::uninitialized_fill()`, and `std::uninitialized_fill_n()`.

2.37.2.7 `template<typename _ForwardIterator, typename _Generator> void std::generate (_ForwardIterator __first, _ForwardIterator __last, _Generator __gen)`

Assign the result of a function object to each value in a sequence.

Parameters

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.
<code>__gen</code>	A function object taking no arguments and returning <code>std::iterator_traits<_ForwardIterator>::value_type</code>

Returns

`generate()` returns no value.

Performs the assignment `*i = __gen()` for each `i` in the range `[__first, __last)`.

Definition at line 4286 of file `stl_algo.h`.

2.37.2.8 `template<typename _OutputIterator, typename _Size, typename _Generator> _OutputIterator std::generate_n (_OutputIterator __first, _Size __n, _Generator __gen)`

Assign the result of a function object to each value in a sequence.

Parameters

<code>__first</code>	A forward iterator.
<code>__n</code>	The length of the sequence.
<code>__gen</code>	A function object taking no arguments and returning <code>std::iterator_traits<_ForwardIterator>::value_type</code>

Returns

The end of the sequence, `__first+__n`

Performs the assignment `*i = __gen()` for each `i` in the range `[__first, __first+__n)`.

_GLIBCXX_RESOLVE_LIB_DEFECTS DR 865. More algorithms that throw away information

Definition at line 4317 of file `stl_algo.h`.

2.37.2.9 `template<typename _InputIterator, typename _Predicate> bool std::is_partitioned (_InputIterator __first, _InputIterator __last, _Predicate __pred) [inline]`

Checks whether the sequence is partitioned.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__pred</code>	A predicate.

Returns

True if the range `[__first,__last)` is partitioned by `__pred`, i.e. if all elements that satisfy `__pred` appear before those that do not.

Definition at line 582 of file `stl_algo.h`.

References `std::find_if_not()`, and `std::none_of()`.

2.37.2.10 `template<typename _ForwardIterator1, typename _ForwardIterator2> void std::iter_swap (_ForwardIterator1 __a, _ForwardIterator2 __b) [inline]`

Swaps the contents of two iterators.

Parameters

<code>__↔ _a</code>	An iterator.
<code>__↔ _b</code>	Another iterator.

Returns

Nothing.

This function swaps the values pointed to by two iterators, not the iterators themselves.

Definition at line 120 of file `stl_algobase.h`.

References `std::swap()`.

Referenced by `std::__introsort_loop()`, `std::__merge_without_buffer()`, `std::__move_median_to_first()`, `std::__partition()`, `std::__reverse()`, `std::__rotate()`, `std::__unguarded_partition()`, `std::includes()`, `std::next_permutation()`, `std::random_↔shuffle()`, `std::shuffle()`, and `std::swap_ranges()`.

2.37.2.11 `template<typename _II, typename _OI> _OI std::move (_II __first, _II __last, _OI __result) [inline]`

Moves the range `[first,last)` into `result`.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__result</code>	An output iterator.

Returns

result + (first - last)

This inline function will boil down to a call to `memmove` whenever possible. Failing that, if random access iterators are passed, then the loop count will be known (and therefore a candidate for compiler optimizations such as unrolling). Result may not be contained within `[first,last)`; the `move_backward` function should be used instead.

Note that the end of the output range is permitted to be contained within `[first,last)`.

Definition at line 491 of file `stl_algobase.h`.

References `std::move()`.

```
2.37.2.12  template<typename _BI1 , typename _BI2 > _BI2 std::move_backward ( _BI1 __first, _BI1 __last, _BI2 __result )
           [inline]
```

Moves the range `[first,last)` into `result`.

Parameters

<code>__first</code>	A bidirectional iterator.
<code>__last</code>	A bidirectional iterator.
<code>__result</code>	A bidirectional iterator.

Returns

result - (first - last)

The function has the same effect as `move`, but starts at the end of the range and works its way to the start, returning the start of the result. This inline function will boil down to a call to `memmove` whenever possible. Failing that, if random access iterators are passed, then the loop count will be known (and therefore a candidate for compiler optimizations such as unrolling).

Result may not be in the range `(first,last]`. Use `move` instead. Note that the start of the output range may overlap `[first,last)`.

Definition at line 670 of file `stl_algobase.h`.

Referenced by `std::_Deque_iterator< _Tp, _Tp &, _Tp * >::_M_set_node()`.

```
2.37.2.13  template<typename _ForwardIterator , typename _Predicate > _ForwardIterator std::partition ( _ForwardIterator __first,
           _ForwardIterator __last, _Predicate __pred ) [inline]
```

Move elements for which a predicate is true to the beginning of a sequence.

Parameters

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.
<code>__pred</code>	A predicate functor.

Returns

An iterator `middle` such that `__pred(i)` is true for each iterator `i` in the range `[_first,middle)` and false for each `i` in the range `[middle,__last)`.

`__pred` must not modify its operand. `partition()` does not preserve the relative ordering of elements in each group, use `stable_partition()` if this is needed.

Definition at line 4498 of file `stl_algo.h`.

References `std::__iterator_category()`, and `std::__partition()`.

```
2.37.2.14  template<typename _InputIterator , typename _OutputIterator1 , typename _OutputIterator2 , typename _Predicate
> pair<_OutputIterator1, _OutputIterator2> std::partition_copy ( _InputIterator __first, _InputIterator __last,
    _OutputIterator1 __out_true, _OutputIterator2 __out_false, _Predicate __pred )
```

Copy the elements of a sequence to separate output sequences depending on the truth value of a predicate.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__out_true</code>	An output iterator.
<code>__out_false</code>	An output iterator.
<code>__pred</code>	A predicate.

Returns

A pair designating the ends of the resulting sequences.

Copies each element in the range `[_first,__last)` for which `__pred` returns true to the range beginning at `out_true` and each element for which `__pred` returns false to `__out_false`.

Definition at line 825 of file `stl_algo.h`.

References `std::__find_if()`.

```
2.37.2.15  template<typename _ForwardIterator , typename _Predicate > _ForwardIterator std::partition_point ( _ForwardIterator
    __first, _ForwardIterator __last, _Predicate __pred )
```

Find the partition point of a partitioned range.

Parameters

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.
<code>__pred</code>	A predicate.

Returns

An iterator `mid` such that `all_of(__first, mid, __pred)` and `none_of(mid, __last, __pred)` are both true.

Definition at line 600 of file `stl_algo.h`.

References `std::advance()`, and `std::distance()`.

```
2.37.2.16 template<typename _RandomAccessIterator > void std::random_shuffle ( _RandomAccessIterator __first,
    _RandomAccessIterator __last ) [inline]
```

Randomly shuffle the elements of a sequence.

Parameters

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.

Returns

Nothing.

Reorder the elements in the range `[__first,__last)` using a random distribution, so that every possible ordering of the sequence is equally likely.

Definition at line 4425 of file `stl_algo.h`.

References `std::iter_swap()`.

```
2.37.2.17 template<typename _RandomAccessIterator , typename _RandomNumberGenerator > void std::random_shuffle (
    _RandomAccessIterator __first, _RandomAccessIterator __last, _RandomNumberGenerator && __rand )
```

Shuffle the elements of a sequence using a random number generator.

Parameters

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.
<code>__rand</code>	The RNG functor or function.

Returns

Nothing.

Reorders the elements in the range `[__first,__last)` using `__rand` to provide a random distribution. Calling `__rand(↵N)` for a positive integer `N` should return a randomly chosen integer from the range `[0,N)`.

Definition at line 4458 of file `stl_algo.h`.

References `std::iter_swap()`.

2.37.2.18 `template<typename _ForwardIterator, typename _Tp> _ForwardIterator std::remove (_ForwardIterator __first, _ForwardIterator __last, const _Tp & __value) [inline]`

Remove elements from a sequence.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__value</code>	The value to be removed.

Returns

An iterator designating the end of the resulting sequence.

All elements equal to `__value` are removed from the range `[__first, __last)`.

`remove()` is stable, so the relative order of elements that are not removed is unchanged.

Elements between the end of the resulting sequence and `__last` are still present, but their value is unspecified.

Definition at line 893 of file `stl_algo.h`.

2.37.2.19 `template<typename _InputIterator, typename _OutputIterator, typename _Tp> _OutputIterator std::remove_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, const _Tp & __value) [inline]`

Copy a sequence, removing elements of a given value.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__result</code>	An output iterator.
<code>__value</code>	The value to be removed.

Returns

An iterator designating the end of the resulting sequence.

Copies each element in the range `[__first, __last)` not equal to `__value` to the range beginning at `__result`. `remove_copy()` is stable, so the relative order of elements that are copied is unchanged.

Definition at line 667 of file `stl_algo.h`.

2.37.2.20 `template<typename _InputIterator, typename _OutputIterator, typename _Predicate> _OutputIterator std::remove_copy_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred) [inline]`

Copy a sequence, removing elements for which a predicate is true.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__result</code>	An output iterator.
<code>__pred</code>	A predicate.

Returns

An iterator designating the end of the resulting sequence.

Copies each element in the range `[__first,__last)` for which `__pred` returns false to the range beginning at `__result`.

`remove_copy_if()` is stable, so the relative order of elements that are copied is unchanged.

Definition at line 700 of file `stl_algo.h`.

```
2.37.2.21  template<typename _ForwardIterator, typename _Predicate> _ForwardIterator std::remove_if ( _ForwardIterator __first,
    _ForwardIterator __last, _Predicate __pred )  [inline]
```

Remove elements from a sequence using a predicate.

Parameters

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.
<code>__pred</code>	A predicate.

Returns

An iterator designating the end of the resulting sequence.

All elements for which `__pred` returns true are removed from the range `[__first,__last)`.

`remove_if()` is stable, so the relative order of elements that are not removed is unchanged.

Elements between the end of the resulting sequence and `__last` are still present, but their value is unspecified.

Definition at line 926 of file `stl_algo.h`.

```
2.37.2.22  template<typename _ForwardIterator, typename _Tp> void std::replace ( _ForwardIterator __first, _ForwardIterator
    __last, const _Tp & __old_value, const _Tp & __new_value )
```

Replace each occurrence of one value in a sequence with another value.

Parameters

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.
<code>__old_value</code>	The value to be replaced.
<code>__new_value</code>	The replacement value.

Returns

`replace()` returns no value.

For each iterator `i` in the range `[__first,__last)` if `*i == __old_value` then the assignment `*i = __new_value` is performed.

Definition at line 4222 of file `stl_algo.h`.

```
2.37.2.23 template<typename _InputIterator, typename _OutputIterator, typename _Predicate, typename _Tp > _OutputIterator
std::replace_copy_if ( _InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred, const _Tp
& __new_value ) [inline]
```

Copy a sequence, replacing each value for which a predicate returns true with another value.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__result</code>	An output iterator.
<code>__pred</code>	A predicate.
<code>__new_value</code>	The replacement value.

Returns

The end of the output sequence, `__result+(__last-__first)`.

Copies each element in the range `[__first,__last)` to the range `[__result,__result+(__last-__first))` replacing elements for which `__pred` returns true with `__new_value`.

Definition at line 3170 of file `stl_algo.h`.

```
2.37.2.24 template<typename _ForwardIterator, typename _Predicate, typename _Tp > void std::replace_if ( _ForwardIterator
__first, _ForwardIterator __last, _Predicate __pred, const _Tp & __new_value )
```

Replace each value in a sequence for which a predicate returns true with another value.

Parameters

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.
<code>__pred</code>	A predicate.
<code>__new_value</code>	The replacement value.

Returns

`replace_if()` returns no value.

For each iterator `i` in the range `[__first, __last)` if `__pred(*i)` is true then the assignment `*i = __new_value` is performed.

Definition at line 4254 of file `stl_algo.h`.

```
2.37.2.25  template<typename _BidirectionalIterator > void std::reverse ( _BidirectionalIterator __first, _BidirectionalIterator __last
    ) [inline]
```

Reverse a sequence.

Parameters

<code>__first</code>	A bidirectional iterator.
<code>__last</code>	A bidirectional iterator.

Returns

`reverse()` returns no value.

Reverses the order of the elements in the range `[__first, __last)`, so that the first element becomes the last etc. For every `i` such that $0 \leq i \leq (_\text{last} - _\text{first})/2$, `reverse()` swaps `*(__first+i)` and `*(__last-(i+1))`

Definition at line 1177 of file `stl_algo.h`.

References `std::__iterator_category()`, and `std::__reverse()`.

```
2.37.2.26  template<typename _BidirectionalIterator, typename _OutputIterator > _OutputIterator std::reverse_copy (
    _BidirectionalIterator __first, _BidirectionalIterator __last, _OutputIterator __result )
```

Copy a sequence, reversing its elements.

Parameters

<code>__first</code>	A bidirectional iterator.
<code>__last</code>	A bidirectional iterator.
<code>__result</code>	An output iterator.

Returns

An iterator designating the end of the resulting sequence.

Copies the elements in the range `[__first, __last)` to the range `[__result, __result+(__last-__first))` such that the order of the elements is reversed. For every `i` such that $0 \leq i \leq (_\text{last} - _\text{first})$, `reverse_copy()` performs the assignment

$*(__result+(__last-__first)-1-i) = *(__first+i)$. The ranges $[__first, __last)$ and $[__result, __result+(__last-__first))$ must not overlap.

Definition at line 1204 of file `stl_algo.h`.

2.37.2.27 `template<typename _ForwardIterator > void std::rotate (_ForwardIterator __first, _ForwardIterator __middle, _ForwardIterator __last) [inline]`

Rotate the elements of a sequence.

Parameters

<code>__first</code>	A forward iterator.
<code>__middle</code>	A forward iterator.
<code>__last</code>	A forward iterator.

Returns

Nothing.

Rotates the elements of the range $[__first, __last)$ by $(__middle - __first)$ positions so that the element at `__middle` is moved to `__first`, the element at `__middle+1` is moved to `__first+1` and so on for each element in the range $[__first, __last)$.

This effectively swaps the ranges $[__first, __middle)$ and $[__middle, __last)$.

Performs $*(__first+(n+(__last-__middle))\%(__last-__first))=*(__first+n)$ for each n in the range $[0, __last-__first)$.

Definition at line 1410 of file `stl_algo.h`.

References `std::__iterator_category()`, and `std::__rotate()`.

Referenced by `std::__inplace_stable_partition()`, `std::__merge_without_buffer()`, `std::__rotate_adaptive()`, and `std::__↵stable_partition_adaptive()`.

2.37.2.28 `template<typename _ForwardIterator, typename _OutputIterator > _OutputIterator std::rotate_copy (_ForwardIterator __first, _ForwardIterator __middle, _ForwardIterator __last, _OutputIterator __result) [inline]`

Copy a sequence, rotating its elements.

Parameters

<code>__first</code>	A forward iterator.
<code>__middle</code>	A forward iterator.
<code>__last</code>	A forward iterator.
<code>__result</code>	An output iterator.

Returns

An iterator designating the end of the resulting sequence.

Copies the elements of the range `[__first,__last)` to the range beginning at

Returns

, rotating the copied elements by `(__middle-__first)` positions so that the element at `__middle` is moved to \leftarrow `__result`, the element at `__middle+1` is moved to `__result+1` and so on for each element in the range `[__first,__last)`.

Performs $*(__result+(n+(__last-__middle))\%(__last-__first))=*(__first+n)$ for each `n` in the range `[0,__last-__first)`.

Definition at line 1445 of file `stl_algo.h`.

2.37.2.29 `template<typename _RandomAccessIterator, typename _UniformRandomNumberGenerator > void std::shuffle (`
`_RandomAccessIterator __first, _RandomAccessIterator __last, _UniformRandomNumberGenerator && __g)`

Shuffle the elements of a sequence using a uniform random number generator.

Parameters

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.
<code>__g</code>	A <code>UniformRandomNumberGenerator</code> (26.5.1.3).

Returns

Nothing.

Reorders the elements in the range `[__first,__last)` using `__g` to provide random numbers.

Definition at line 3706 of file `stl_algo.h`.

References `std::iter_swap()`.

2.37.2.30 `template<typename _ForwardIterator, typename _Predicate > _ForwardIterator std::stable_partition (`
`_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred) [inline]`

Move elements for which a predicate is true to the beginning of a sequence, preserving relative ordering.

Parameters

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.
<code>__pred</code>	A predicate functor.

Returns

An iterator `middle` such that `__pred(i)` is true for each iterator `i` in the range `[first,middle)` and false for each `i` in the range `[middle,last)`.

Performs the same function as `partition()` with the additional guarantee that the relative ordering of elements in each group is preserved, so any two elements `x` and `y` in the range `[__first,__last)` such that `__pred(x) == __pred(y)` will have the same relative ordering after calling `stable_partition()`.

Definition at line 1652 of file `stl_algo.h`.

2.37.2.31 `template<typename _ForwardIterator1, typename _ForwardIterator2 > _ForwardIterator2 std::swap_ranges (`
`_FowardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2)`

Swap the elements of two sequences.

Parameters

<code>__first1</code>	A forward iterator.
<code>__last1</code>	A forward iterator.
<code>__first2</code>	A forward iterator.

Returns

An iterator equal to `first2+(last1-first1)`.

Swaps each element in the range `[first1,last1)` with the corresponding element in the range `[first2,(last1-first1))`. The ranges must not overlap.

Definition at line 166 of file `stl_algobase.h`.

References `std::iter_swap()`.

Referenced by `std::__rotate()`.

2.37.2.32 `template<typename _InputIterator, typename _OutputIterator, typename _UnaryOperation > _OutputIterator`
`std::transform (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _UnaryOperation __unary_op)`

Perform an operation on a sequence.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__result</code>	An output iterator.
<code>__unary_op</code>	A unary operator.

Returns

An output iterator equal to `__result+(__last-__first)`.

Applies the operator to each element in the input range and assigns the results to successive elements of the output sequence. Evaluates `*(__result+N)=unary_op(*(__first+N))` for each `N` in the range `[0,__last-__first)`.

`unary_op` must not alter its argument.

Definition at line 4153 of file `stl_algo.h`.

```
2.37.2.33 template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _BinaryOperation >
    _OutputIterator std::transform ( _InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _OutputIterator
    __result, _BinaryOperation __binary_op )
```

Perform an operation on corresponding elements of two sequences.

Parameters

<code>__first1</code>	An input iterator.
<code>__last1</code>	An input iterator.
<code>__first2</code>	An input iterator.
<code>__result</code>	An output iterator.
<code>__binary_op</code>	A binary operator.

Returns

An output iterator equal to `result+(last-first)`.

Applies the operator to the corresponding elements in the two input ranges and assigns the results to successive elements of the output sequence. Evaluates `*(__result+N)=__binary_op(*(__first1+N),*(__first2+N))` for each `N` in the range `[0,__last1-__first1)`.

`binary_op` must not alter either of its arguments.

Definition at line 4190 of file `stl_algo.h`.

```
2.37.2.34 template<typename _ForwardIterator > _ForwardIterator std::unique ( _ForwardIterator __first, _ForwardIterator __last )
    [inline]
```

Remove consecutive duplicate values from a sequence.

Parameters

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.

Returns

An iterator designating the end of the resulting sequence.

Removes all but the first element from each group of consecutive values that compare equal. `unique()` is stable, so the relative order of elements that are not removed is unchanged. Elements between the end of the resulting sequence and `__last` are still present, but their value is unspecified.

Definition at line 992 of file `stl_algo.h`.

```
2.37.2.35  template<typename _ForwardIterator, typename _BinaryPredicate> _ForwardIterator std::unique ( _ForwardIterator
    __first, _ForwardIterator __last, _BinaryPredicate __binary_pred ) [inline]
```

Remove consecutive values from a sequence using a predicate.

Parameters

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.
<code>__binary_pred</code>	A binary predicate.

Returns

An iterator designating the end of the resulting sequence.

Removes all but the first element from each group of consecutive values for which `__binary_pred` returns true. `unique()` is stable, so the relative order of elements that are not removed is unchanged. Elements between the end of the resulting sequence and `__last` are still present, but their value is unspecified.

Definition at line 1022 of file `stl_algo.h`.

```
2.37.2.36  template<typename _InputIterator, typename _OutputIterator> _OutputIterator std::unique_copy ( _InputIterator __first,
    _InputIterator __last, _OutputIterator __result ) [inline]
```

Copy a sequence, removing consecutive duplicate values.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__result</code>	An output iterator.

Returns

An iterator designating the end of the resulting sequence.

Copies each element in the range `[__first, __last)` to the range beginning at `__result`, except that only the first element is copied from groups of consecutive elements that compare equal. `unique_copy()` is stable, so the relative order of elements that are copied is unchanged.

`_GLIBCXX_RESOLVE_LIB_DEFECTS` DR 241. Does `unique_copy()` require `CopyConstructible` and `Assignable`?

`_GLIBCXX_RESOLVE_LIB_DEFECTS` DR 538. 241 again: Does `unique_copy()` require `CopyConstructible` and `Assignable`?

Definition at line 4353 of file `stl_algo.h`.

References `std::__iterator_category()`, and `std::__unique_copy()`.

```
2.37.2.37  template<typename _InputIterator, typename _OutputIterator, typename _BinaryPredicate> _OutputIterator
            std::unique_copy ( _InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryPredicate __binary_pred
            ) [inline]
```

Copy a sequence, removing consecutive values using a predicate.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__result</code>	An output iterator.
<code>__binary_pred</code>	A binary predicate.

Returns

An iterator designating the end of the resulting sequence.

Copies each element in the range `[__first,__last)` to the range beginning at `__result`, except that only the first element is copied from groups of consecutive elements for which `__binary_pred` returns true. `unique_copy()` is stable, so the relative order of elements that are copied is unchanged.

`_GLIBCXX_RESOLVE_LIB_DEFECTS` DR 241. Does `unique_copy()` require `CopyConstructible` and `Assignable`?

Definition at line 4394 of file `stl_algo.h`.

References `std::__iterator_category()`, and `std::__unique_copy()`.

2.38 Negators

Collaboration diagram for Negators:



Classes

- class `std::binary_negate<_Predicate>`
- class `std::unary_negate<_Predicate>`

Functions

- `template<typename _Predicate>`
`unary_negate<_Predicate> std::not1 (const _Predicate &__pred)`
- `template<typename _Predicate>`
`binary_negate<_Predicate> std::not2 (const _Predicate &__pred)`

2.38.1 Detailed Description

The functions `not1` and `not2` each take a predicate functor and return an instance of `unary_negate` or `binary_negate`, respectively. These classes are functors whose `operator()` performs the stored predicate function and then returns the negation of the result.

For example, given a vector of integers and a trivial predicate,

```

struct IntGreaterThanThree
: public std::unary_function<int, bool>
{
    bool operator() (int x) { return x > 3; }
};

std::find_if (v.begin(), v.end(), not1(IntGreaterThanThree()));
  
```

The call to `find_if` will locate the first index (*i*) of *v* for which `!(v[i] > 3)` is true.

The `not1/unary_negate` combination works on predicates taking a single argument. The `not2/binary_negate` combination works on predicates which take two arguments.

2.38.2 Function Documentation

2.38.2.1 `template<typename _Predicate > unary_negate<_Predicate> std::not1 (const _Predicate & __pred) [inline]`

One of the [negation functors](#).

Definition at line 722 of file `stl_function.h`.

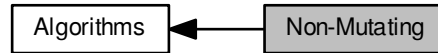
2.38.2.2 `template<typename _Predicate > binary_negate<_Predicate> std::not2 (const _Predicate & __pred) [inline]`

One of the [negation functors](#).

Definition at line 747 of file `stl_function.h`.

2.39 Non-Mutating

Collaboration diagram for Non-Mutating:



Functions

- `template<typename _ForwardIterator >`
`_ForwardIterator std::adjacent_find (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _BinaryPredicate >`
`_ForwardIterator std::adjacent_find (_ForwardIterator __first, _ForwardIterator __last, _BinaryPredicate __↵
binary_pred)`
- `template<typename _InputIterator, typename _Predicate >`
`bool std::all_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _Predicate >`
`bool std::any_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _Tp >`
`iterator_traits< _InputIterator >::difference_type std::count (_InputIterator __first, _InputIterator __last, const _Tp
&__value)`
- `template<typename _InputIterator, typename _Predicate >`
`iterator_traits< _InputIterator >::difference_type std::count_if (_InputIterator __first, _InputIterator __last, _↵
Predicate __pred)`
- `template<typename _Iter1, typename _Iter2, typename _BinaryPredicate >`
`bool std::equal (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _BinaryPredicate __binary_pred)`
- `template<typename _II1, typename _II2 >`
`bool std::equal (_II1 __first1, _II1 __last1, _II2 __first2)`
- `template<typename _InputIterator, typename _Tp >`
`_InputIterator std::find (_InputIterator __first, _InputIterator __last, const _Tp &__val)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`
`_ForwardIterator1 std::find_end (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __↵
first2, _ForwardIterator2 __last2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate >`
`_ForwardIterator1 std::find_end (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __↵
first2, _ForwardIterator2 __last2, _BinaryPredicate __comp)`
- `template<typename _InputIterator, typename _ForwardIterator >`
`_InputIterator std::find_first_of (_InputIterator __first1, _InputIterator __last1, _ForwardIterator __first2, _↵
ForwardIterator __last2)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _BinaryPredicate >`
`_InputIterator std::find_first_of (_InputIterator __first1, _InputIterator __last1, _ForwardIterator __first2, _↵
ForwardIterator __last2, _BinaryPredicate __comp)`
- `template<typename _InputIterator, typename _Predicate >`
`_InputIterator std::find_if (_InputIterator __first, _InputIterator __last, _Predicate __pred)`

- `template<typename _InputIterator, typename _Predicate >`
`_InputIterator std::find_if_not (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _Function >`
`_Function std::for_each (_InputIterator __first, _InputIterator __last, _Function __f)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`
`bool std::is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate >`
`bool std::is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _BinaryPredicate __pred)`
- `template<typename _InputIterator1, typename _InputIterator2 >`
`pair< _InputIterator1, _InputIterator2 > std::mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate >`
`pair< _InputIterator1, _InputIterator2 > std::mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _BinaryPredicate __binary_pred)`
- `template<typename _InputIterator, typename _Predicate >`
`bool std::none_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`
`_ForwardIterator1 std::search (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate >`
`_ForwardIterator1 std::search (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate __predicate)`
- `template<typename _ForwardIterator, typename _Integer, typename _Tp >`
`_ForwardIterator std::search_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp &__val)`
- `template<typename _ForwardIterator, typename _Integer, typename _Tp, typename _BinaryPredicate >`
`_ForwardIterator std::search_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred)`

2.39.1 Detailed Description

2.39.2 Function Documentation

2.39.2.1 `template<typename _ForwardIterator > _ForwardIterator std::adjacent_find (_ForwardIterator __first, _ForwardIterator __last) [inline]`

Find two adjacent values in a sequence that are equal.

Parameters

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.

Returns

The first iterator `i` such that `i` and `i+1` are both valid iterators in `[__first,__last)` and such that `*i == *(i+1)`, or `__last` if no such iterator exists.

Definition at line 3898 of file `stl_algo.h`.

2.39.2.2 `template<typename _ForwardIterator, typename _BinaryPredicate> _ForwardIterator std::adjacent_find (`
`_ForwardIterator __first, _ForwardIterator __last, _BinaryPredicate __binary_pred) [inline]`

Find two adjacent values in a sequence using a predicate.

Parameters

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.
<code>__binary_pred</code>	A binary predicate.

Returns

The first iterator `i` such that `i` and `i+1` are both valid iterators in `[__first, __last)` and such that `__binary_pred(*i, *(i+1))` is true, or `__last` if no such iterator exists.

Definition at line 3923 of file `stl_algo.h`.

2.39.2.3 `template<typename _InputIterator, typename _Predicate> bool std::all_of (_InputIterator __first, _InputIterator __last,`
`_Predicate __pred) [inline]`

Checks that a predicate is true for all the elements of a sequence.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__pred</code>	A predicate.

Returns

True if the check is true, false otherwise.

Returns true if `__pred` is true for each element in the range `[__first, __last)`, and false otherwise.

Definition at line 508 of file `stl_algo.h`.

References `std::find_if_not()`.

2.39.2.4 `template<typename _InputIterator, typename _Predicate> bool std::any_of (_InputIterator __first, _InputIterator __last,`
`_Predicate __pred) [inline]`

Checks that a predicate is false for at least an element of a sequence.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__pred</code>	A predicate.

Returns

True if the check is true, false otherwise.

Returns true if an element exists in the range `[__first,__last)` such that `__pred` is true, and false otherwise.

Definition at line 543 of file `stl_algo.h`.

References `std::none_of()`.

2.39.2.5 `template<typename _InputIterator, typename _Tp> iterator_traits<_InputIterator>::difference_type std::count (`
`_InputIterator __first, _InputIterator __last, const _Tp & __value) [inline]`

Count the number of copies of a value in a sequence.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__value</code>	The value to be counted.

Returns

The number of iterators `i` in the range `[__first,__last)` for which `*i == __value`

Definition at line 3948 of file `stl_algo.h`.

2.39.2.6 `template<typename _InputIterator, typename _Predicate> iterator_traits<_InputIterator>::difference_type std::count_if (`
`_InputIterator __first, _InputIterator __last, _Predicate __pred) [inline]`

Count the elements of a sequence for which a predicate is true.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__pred</code>	A predicate.

Returns

The number of iterators `i` in the range `[__first,__last)` for which `__pred(*i)` is true.

Definition at line 3971 of file `stl_algo.h`.

2.39.2.7 `template<typename _Iter1, typename _Iter2, typename _BinaryPredicate> bool std::equal (_Iter1 __first1, _Iter1`
`__last1, _Iter2 __first2, _BinaryPredicate __binary_pred) [inline]`

Tests a range for element-wise equality.

Parameters

<code>__first1</code>	An input iterator.
<code>__last1</code>	An input iterator.
<code>__first2</code>	An input iterator.
<code>__binary_pred</code>	A binary predicate functor .

Returns

A boolean true or false.

This compares the elements of two ranges using the `binary_pred` parameter, and returns true or false depending on whether all of the corresponding elements of the ranges are equal.

Definition at line 1085 of file `stl_algobase.h`.

References `std::distance()`, and `std::equal()`.

2.39.2.8 `template<typename _I1, typename _I2 > bool std::equal (_I1 __first1, _I1 __last1, _I2 __first2) [inline]`

Tests a range for element-wise equality.

Parameters

<code>__first1</code>	An input iterator.
<code>__last1</code>	An input iterator.
<code>__first2</code>	An input iterator.

Returns

A boolean true or false.

This compares the elements of two ranges using `==` and returns true or false depending on whether all of the corresponding elements of the ranges are equal.

Definition at line 1053 of file `stl_algobase.h`.

Referenced by `std::equal()`, `std::fill_n()`, `std::operator!=()`, and `std::operator==()`.

2.39.2.9 `template<typename _InputIterator, typename _Tp > _InputIterator std::find (_InputIterator __first, _InputIterator __last, const _Tp & __val) [inline]`

Find the first occurrence of a value in a sequence.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__val</code>	The value to find.

Returns

The first iterator `i` in the range `[__first, __last)` such that `*i == __val`, or `__last` if no such iterator exists.

Definition at line 3771 of file `stl_algo.h`.

References `std::__find_if()`.

```
2.39.2.10 template<typename _ForwardIterator1, typename _ForwardIterator2 > _ForwardIterator1 std::find_end (
    _ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2 )
    [inline]
```

Find last matching subsequence in a sequence.

Parameters

<code>__first1</code>	Start of range to search.
<code>__last1</code>	End of range to search.
<code>__first2</code>	Start of sequence to match.
<code>__last2</code>	End of sequence to match.

Returns

The last iterator `i` in the range `[__first1, __last1 - (__last2 - __first2))` such that `*(i+N) == *(__first2+N)` for each `N` in the range `[0, __last2 - __first2)`, or `__last1` if no such iterator exists.

Searches the range `[__first1, __last1)` for a sub-sequence that compares equal value-by-value with the sequence given by `[__first2, __last2)` and returns an iterator to the `__first` element of the sub-sequence, or `__last1` if the sub-sequence is not found. The sub-sequence will be the last such subsequence contained in `[__first1, __last1)`.

Because the sub-sequence must lie completely within the range `[__first1, __last1)` it must start at a position less than `__last1 - (__last2 - __first2)` where `__last2 - __first2` is the length of the sub-sequence. This means that the returned iterator `i` will be in the range `[__first1, __last1 - (__last2 - __first2))`.

Definition at line 425 of file `stl_algo.h`.

References `std::__iterator_category()`.

```
2.39.2.11 template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate > _ForwardIterator1
    std::find_end ( _ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2
        __last2, _BinaryPredicate __comp ) [inline]
```

Find last matching subsequence in a sequence using a predicate.

Parameters

<code>__first1</code>	Start of range to search.
<code>__last1</code>	End of range to search.
<code>__first2</code>	Start of sequence to match.
<code>__last2</code>	End of sequence to match.
<code>comp</code>	The predicate to use.

Returns

The last iterator `i` in the range `[__first1, __last1 - (__last2 - __first2))` such that `predicate(*(i+N), (__first2+N))` is true for each `N` in the range `[0, __last2 - __first2)`, or `__last1` if no such iterator exists.

Searches the range `[__first1, __last1)` for a sub-sequence that compares equal value-by-value with the sequence given by `[__first2, __last2)` using `comp` as a predicate and returns an iterator to the first element of the sub-sequence, or `__last1` if the sub-sequence is not found. The sub-sequence will be the last such subsequence contained in `[__first1, __last1)`.

Because the sub-sequence must lie completely within the range `[__first1, __last1)` it must start at a position less than `__last1 - (__last2 - __first2)` where `__last2 - __first2` is the length of the sub-sequence. This means that the returned iterator `i` will be in the range `[__first1, __last1 - (__last2 - __first2))`

Definition at line 474 of file `stl_algo.h`.

References `std::__iterator_category()`.

2.39.2.12 `template<typename _InputIterator, typename _ForwardIterator> _InputIterator std::find_first_of (_InputIterator __first1, _InputIterator __last1, _ForwardIterator __first2, _ForwardIterator __last2)`

Find element from a set in a sequence.

Parameters

<code>__first1</code>	Start of range to search.
<code>__last1</code>	End of range to search.
<code>__first2</code>	Start of match candidates.
<code>__last2</code>	End of match candidates.

Returns

The first iterator `i` in the range `[__first1, __last1)` such that `*i == *(i2)` such that `i2` is an iterator in `[__first2, __last2)`, or `__last1` if no such iterator exists.

Searches the range `[__first1, __last1)` for an element that is equal to some element in the range `[__first2, __last2)`. If found, returns an iterator in the range `[__first1, __last1)`, otherwise returns `__last1`.

Definition at line 3826 of file `stl_algo.h`.

2.39.2.13 `template<typename _InputIterator, typename _ForwardIterator, typename _BinaryPredicate> _InputIterator std::find_first_of (_InputIterator __first1, _InputIterator __last1, _ForwardIterator __first2, _ForwardIterator __last2, _BinaryPredicate __comp)`

Find element from a set in a sequence using a predicate.

Parameters

<code>__first1</code>	Start of range to search.
<code>__last1</code>	End of range to search.
<code>__first2</code>	Start of match candidates.
<code>__last2</code>	End of match candidates.
<code>__comp</code>	Predicate to use.

Returns

The first iterator `i` in the range `[__first1, __last1)` such that `comp(*i, *(i2))` is true and `i2` is an iterator in `[__first2, __last2)`, or `__last1` if no such iterator exists.

Searches the range `[__first1, __last1)` for an element that is equal to some element in the range `[__first2, __last2)`. If found, returns an iterator in the range `[__first1, __last1)`, otherwise returns `__last1`.

Definition at line 3867 of file `stl_algo.h`.

```
2.39.2.14  template<typename _InputIterator, typename _Predicate> _InputIterator std::find_if ( _InputIterator __first,
    _InputIterator __last, _Predicate __pred ) [inline]
```

Find the first element in a sequence for which a predicate is true.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__pred</code>	A predicate.

Returns

The first iterator `i` in the range `[__first, __last)` such that `__pred(*i)` is true, or `__last` if no such iterator exists.

Definition at line 3795 of file `stl_algo.h`.

References `std::__find_if()`.

Referenced by `std::none_of()`.

```
2.39.2.15  template<typename _InputIterator, typename _Predicate> _InputIterator std::find_if_not ( _InputIterator __first,
    _InputIterator __last, _Predicate __pred ) [inline]
```

Find the first element in a sequence for which a predicate is false.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__pred</code>	A predicate.

Returns

The first iterator `i` in the range `[__first, __last)` such that `__pred(*i)` is false, or `__last` if no such iterator exists.

Definition at line 558 of file `stl_algo.h`.

References `std::__find_if_not()`.

Referenced by `std::all_of()`, and `std::is_partitioned()`.

2.39.2.16 `template<typename _InputIterator, typename _Function> _Function std::for_each (_InputIterator __first, _InputIterator __last, _Function __f)`

Apply a function to every element of a sequence.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__f</code>	A unary function object.

Returns

`__f` (`std::move(__f)` in C++0x).

Applies the function object `__f` to each element in the range `[first,last)`. `__f` must not modify the order of the sequence. If `__f` has a return value it is ignored.

Definition at line 3750 of file `stl_algo.h`.

2.39.2.17 `template<typename _ForwardIterator1, typename _ForwardIterator2> bool std::is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2) [inline]`

Checks whether a permutation of the second sequence is equal to the first sequence.

Parameters

<code>__first1</code>	Start of first range.
<code>__last1</code>	End of first range.
<code>__first2</code>	Start of second range.

Returns

true if there exists a permutation of the elements in the range `[__first2, __first2 + (__last1 - __first1))`, beginning with `ForwardIterator2` begin, such that `equal(__first1, __last1, begin)` returns true; otherwise, returns false.

Definition at line 3526 of file `stl_algo.h`.

2.39.2.18 `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate> bool std::is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _BinaryPredicate __pred) [inline]`

Checks whether a permutation of the second sequence is equal to the first sequence.

Parameters

<code>__first1</code>	Start of first range.
<code>__last1</code>	End of first range.
<code>__first2</code>	Start of second range.
<code>__pred</code>	A binary predicate.

Returns

true if there exists a permutation of the elements in the range `[__first2, __first2 + (__last1 - __first1))`, beginning with `ForwardIterator2` begin, such that `equal(__first1, __last1, __begin, __pred)` returns true; otherwise, returns false.

Definition at line 3558 of file `stl_algo.h`.

References `std::__find_if()`, and `std::distance()`.

2.39.2.19 `template<typename _InputIterator1, typename _InputIterator2> pair<_InputIterator1, _InputIterator2> std::mismatch (`
`_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2) [inline]`

Finds the places in ranges which don't match.

Parameters

<code>__first1</code>	An input iterator.
<code>__last1</code>	An input iterator.
<code>__first2</code>	An input iterator.

Returns

A pair of iterators pointing to the first mismatch.

This compares the elements of two ranges using `==` and returns a pair of iterators. The first iterator points into the first range, the second iterator points into the second range, and the elements pointed to by the iterators are not equal.

Definition at line 1292 of file `stl_algobase.h`.

2.39.2.20 `template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate> pair<_InputIterator1,`
`_InputIterator2> std::mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2,`
`_BinaryPredicate __binary_pred) [inline]`

Finds the places in ranges which don't match.

Parameters

<code>__first1</code>	An input iterator.
<code>__last1</code>	An input iterator.
<code>__first2</code>	An input iterator.
<code>__binary_pred</code>	A binary predicate functor .

Returns

A pair of iterators pointing to the first mismatch.

This compares the elements of two ranges using the `binary_pred` parameter, and returns a pair of iterators. The first iterator points into the first range, the second iterator points into the second range, and the elements pointed to by the iterators are not equal.

Definition at line 1326 of file `stl_algobase.h`.

2.39.2.21 `template<typename _InputIterator, typename _Predicate> bool std::none_of (_InputIterator __first, _InputIterator __last, _Predicate __pred) [inline]`

Checks that a predicate is false for all the elements of a sequence.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__pred</code>	A predicate.

Returns

True if the check is true, false otherwise.

Returns true if `__pred` is false for each element in the range `[__first,__last)`, and false otherwise.

Definition at line 525 of file `stl_algo.h`.

References `std::find_if()`.

Referenced by `std::any_of()`, and `std::is_partitioned()`.

2.39.2.22 `template<typename _ForwardIterator1, typename _ForwardIterator2> _ForwardIterator1 std::search (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2) [inline]`

Search a sequence for a matching sub-sequence.

Parameters

<code>__first1</code>	A forward iterator.
<code>__last1</code>	A forward iterator.
<code>__first2</code>	A forward iterator.
<code>__last2</code>	A forward iterator.

Returns

The first iterator `i` in the range `[__first1, __last1 - (__last2 - __first2))` such that `*(i+N) == *(__first2+N)` for each `N` in the range `[0, __last2 - __first2)`, or `__last1` if no such iterator exists.

Searches the range `[__first1, __last1)` for a sub-sequence that compares equal value-by-value with the sequence given by `[__first2, __last2)` and returns an iterator to the first element of the sub-sequence, or `__last1` if the sub-sequence is not found.

Because the sub-sequence must lie completely within the range `[__first1, __last1)` it must start at a position less than `__last1 - (__last2 - __first2)` where `__last2 - __first2` is the length of the sub-sequence.

This means that the returned iterator `i` will be in the range `[__first1, __last1 - (__last2 - __first2))`

Definition at line 4011 of file `stl_algo.h`.

```
2.39.2.23  template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate > _ForwardIterator1
std::search( _ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2,
             _BinaryPredicate __predicate ) [inline]
```

Search a sequence for a matching sub-sequence using a predicate.

Parameters

<code>__first1</code>	A forward iterator.
<code>__last1</code>	A forward iterator.
<code>__first2</code>	A forward iterator.
<code>__last2</code>	A forward iterator.
<code>__predicate</code>	A binary predicate.

Returns

The first iterator `i` in the range `[__first1, __last1 - (__last2 - __first2))` such that `__predicate(*(i+N), *(__first2+N))` is true for each `N` in the range `[0, __last2 - __first2)`, or `__last1` if no such iterator exists.

Searches the range `[__first1, __last1)` for a sub-sequence that compares equal value-by-value with the sequence given by `[__first2, __last2)`, using `__predicate` to determine equality, and returns an iterator to the first element of the sub-sequence, or `__last1` if no such iterator exists.

See also

`search(_ForwardIter1, _ForwardIter1, _ForwardIter2, _ForwardIter2)`

Definition at line 4051 of file `stl_algo.h`.

```
2.39.2.24  template<typename _ForwardIterator, typename _Integer, typename _Tp > _ForwardIterator std::search_n(
             _ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp & __val ) [inline]
```

Search a sequence for a number of consecutive values.

Parameters

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.
<code>__count</code>	The number of consecutive values.
<code>__val</code>	The value to find.

Returns

The first iterator `i` in the range `[__first, __last-__count)` such that `*(i+N) == __val` for each `N` in the range `[0, __count)`, or `__last` if no such iterator exists.

Searches the range `[__first, __last)` for `count` consecutive elements equal to `__val`.

Definition at line 4085 of file `stl_algo.h`.

```
2.39.2.25  template<typename _ForwardIterator, typename _Integer, typename _Tp, typename _BinaryPredicate > _ForwardIterator
std::search_n( _ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp & __val, _BinaryPredicate
__binary_pred ) [inline]
```

Search a sequence for a number of consecutive values using a predicate.

Parameters

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.
<code>__count</code>	The number of consecutive values.
<code>__val</code>	The value to find.
<code>__binary_pred</code>	A binary predicate.

Returns

The first iterator `i` in the range `[__first, __last-__count)` such that `__binary_pred(*(i+N), __val)` is true for each `N` in the range `[0, __count)`, or `__last` if no such iterator exists.

Searches the range `[__first, __last)` for `__count` consecutive elements for which the predicate returns true.

Definition at line 4119 of file `stl_algo.h`.

2.40 Normal Distributions

Collaboration diagram for Normal Distributions:



Classes

- class `std::cauchy_distribution<_RealType>`
- class `std::chi_squared_distribution<_RealType>`
- class `std::fisher_f_distribution<_RealType>`
- class `std::gamma_distribution<_RealType>`
- class `std::lognormal_distribution<_RealType>`
- class `std::normal_distribution<_RealType>`
- class `std::student_t_distribution<_RealType>`

Functions

- template<typename _RealType>
bool `std::operator!=` (const `std::normal_distribution<_RealType>` &__d1, const `std::normal_distribution<_RealType>` &__d2)
- template<typename _RealType>
bool `std::operator!=` (const `std::lognormal_distribution<_RealType>` &__d1, const `std::lognormal_distribution<_RealType>` &__d2)
- template<typename _RealType>
bool `std::operator!=` (const `std::gamma_distribution<_RealType>` &__d1, const `std::gamma_distribution<_RealType>` &__d2)
- template<typename _RealType>
bool `std::operator!=` (const `std::chi_squared_distribution<_RealType>` &__d1, const `std::chi_squared_distribution<_RealType>` &__d2)
- template<typename _RealType>
bool `std::operator!=` (const `std::cauchy_distribution<_RealType>` &__d1, const `std::cauchy_distribution<_RealType>` &__d2)
- template<typename _RealType>
bool `std::operator!=` (const `std::fisher_f_distribution<_RealType>` &__d1, const `std::fisher_f_distribution<_RealType>` &__d2)
- template<typename _RealType>
bool `std::operator!=` (const `std::student_t_distribution<_RealType>` &__d1, const `std::student_t_distribution<_RealType>` &__d2)
- template<typename _RealType, typename _CharT, typename _Traits>
`std::basic_ostream<_CharT, _Traits>` & `std::operator<<` (`std::basic_ostream<_CharT, _Traits>` &__os, const `std::cauchy_distribution<_RealType>` &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
`std::basic_istream<_CharT, _Traits>` & `std::operator>>` (`std::basic_istream<_CharT, _Traits>` &__is, `std::cauchy_distribution<_RealType>` &__x)

2.40.1 Detailed Description

2.40.2 Function Documentation

2.40.2.1 `template<typename _RealType> bool std::operator!=(const std::normal_distribution< _RealType> & __d1, const std::normal_distribution< _RealType> & __d2) [inline]`

Return true if two normal distributions are different.

Definition at line 2285 of file random.h.

2.40.2.2 `template<typename _RealType> bool std::operator!=(const std::lognormal_distribution< _RealType> & __d1, const std::lognormal_distribution< _RealType> & __d2) [inline]`

Return true if two lognormal distributions are different.

Definition at line 2489 of file random.h.

2.40.2.3 `template<typename _RealType> bool std::operator!=(const std::gamma_distribution< _RealType> & __d1, const std::gamma_distribution< _RealType> & __d2) [inline]`

Return true if two gamma distributions are different.

Definition at line 2709 of file random.h.

2.40.2.4 `template<typename _RealType> bool std::operator!=(const std::chi_squared_distribution< _RealType> & __d1, const std::chi_squared_distribution< _RealType> & __d2) [inline]`

Return true if two Chi-squared distributions are different.

Definition at line 2919 of file random.h.

2.40.2.5 `template<typename _RealType> bool std::operator!=(const std::cauchy_distribution< _RealType> & __d1, const std::cauchy_distribution< _RealType> & __d2) [inline]`

Return true if two Cauchy distributions have different parameters.

Definition at line 3086 of file random.h.

References `std::operator>>()`.

2.40.2.6 `template<typename _RealType> bool std::operator!=(const std::fisher_f_distribution< _RealType> & __d1, const std::fisher_f_distribution< _RealType> & __d2) [inline]`

Return true if two Fisher f distributions are different.

Definition at line 3342 of file random.h.

2.40.2.7 `template<typename _RealType> bool std::operator!=(const std::student_t_distribution< _RealType> & __d1, const std::student_t_distribution< _RealType> & __d2) [inline]`

Return true if two Student t distributions are different.

Definition at line 3555 of file random.h.

2.40.2.8 `template<typename _RealType, typename _CharT, typename _Traits> std::basic_ostream< _CharT, _Traits> & std::operator<< (std::basic_ostream< _CharT, _Traits> & __os, const std::cauchy_distribution< _RealType> & __x)`

Inserts a `cauchy_distribution` random number distribution `__x` into the output stream `__os`.

Parameters

<code>__os</code>	An output stream.
<code>__x</code>	A <code>cauchy_distribution</code> random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

2.40.2.9 `template<typename _RealType , typename _CharT , typename _Traits > std::basic_istream<_CharT, _Traits>& std::operator>> (std::basic_istream<_CharT, _Traits> &__is, std::cauchy_distribution<_RealType> &__x)`

Extracts a `cauchy_distribution` random number distribution `__x` from the input stream `__is`.

Parameters

<code>__is</code>	An input stream.
<code>__x</code>	A <code>cauchy_distribution</code> random number generator engine.

Returns

The input stream with `__x` extracted or in an error state.

2.41 Numeric_arrays

Classes

- class `std::gslice`
- class `std::gslice_array< _Tp >`
- class `std::indirect_array< _Tp >`
- class `std::mask_array< _Tp >`
- class `std::slice`
- class `std::slice_array< _Tp >`

Macros

- `#define _DEFINE_VALARRAY_OPERATOR(_Op, _Name)`
- `#define _DEFINE_VALARRAY_OPERATOR(_Op, _Name)`
- `#define _DEFINE_VALARRAY_OPERATOR(_Op, _Name)`
- `#define _DEFINE_VALARRAY_OPERATOR(_Op, _Name)`

Functions

- `std::gslice::gslice ()`
- `std::gslice::gslice (size_t __o, const valarray< size_t > &__l, const valarray< size_t > &__s)`
- `std::gslice::gslice (const gslice &)`
- `std::gslice_array< _Tp >::gslice_array (const gslice_array &)`
- `std::indirect_array< _Tp >::indirect_array (const indirect_array &)`
- `std::mask_array< _Tp >::mask_array (const mask_array &)`
- `std::slice::slice ()`
- `std::slice::slice (size_t __o, size_t __d, size_t __s)`
- `std::slice_array< _Tp >::slice_array (const slice_array &)`
- `std::gslice::~~gslice ()`
- `void std::gslice_array< _Tp >::operator%= (const valarray< _Tp > &) const`
- `void std::mask_array< _Tp >::operator%= (const valarray< _Tp > &) const`
- `void std::indirect_array< _Tp >::operator%= (const valarray< _Tp > &) const`
- `template<class _Dom >`
`void std::gslice_array< _Tp >::operator%= (const _Expr< _Dom, _Tp > &) const`
- `template<class _Dom >`
`void std::mask_array< _Tp >::operator%= (const _Expr< _Dom, _Tp > &) const`
- `template<class _Dom >`
`void std::indirect_array< _Tp >::operator%= (const _Expr< _Dom, _Tp > &) const`
- `void std::slice_array< _Tp >::operator%= (const valarray< _Tp > &) const`
- `template<class _Dom >`
`void std::slice_array< _Tp >::operator%= (const _Expr< _Dom, _Tp > &) const`
- `void std::gslice_array< _Tp >::operator&= (const valarray< _Tp > &) const`
- `void std::mask_array< _Tp >::operator&= (const valarray< _Tp > &) const`
- `void std::indirect_array< _Tp >::operator&= (const valarray< _Tp > &) const`
- `template<class _Dom >`
`void std::gslice_array< _Tp >::operator&= (const _Expr< _Dom, _Tp > &) const`
- `template<class _Dom >`
`void std::mask_array< _Tp >::operator&= (const _Expr< _Dom, _Tp > &) const`

- [illegible]

- `template<class _Dom >`
`void std::slice_array<_Tp>::operator/=(const _Expr<_Dom, _Tp> &) const`
- `void std::gslice_array<_Tp>::operator<<= (const valarray<_Tp> &) const`
- `void std::mask_array<_Tp>::operator<<= (const valarray<_Tp> &) const`
- `void std::indirect_array<_Tp>::operator<<= (const valarray<_Tp> &) const`
- `template<class _Dom >`
`void std::gslice_array<_Tp>::operator<<= (const _Expr<_Dom, _Tp> &) const`
- `template<class _Dom >`
`void std::mask_array<_Tp>::operator<<= (const _Expr<_Dom, _Tp> &) const`
- `template<class _Dom >`
`void std::indirect_array<_Tp>::operator<<= (const _Expr<_Dom, _Tp> &) const`
- `void std::slice_array<_Tp>::operator<<= (const valarray<_Tp> &) const`
- `template<class _Dom >`
`void std::slice_array<_Tp>::operator<<= (const _Expr<_Dom, _Tp> &) const`
- `gslice_array & std::gslice_array<_Tp>::operator= (const gslice_array &)`
- `indirect_array & std::indirect_array<_Tp>::operator= (const indirect_array &)`
- `mask_array & std::mask_array<_Tp>::operator= (const mask_array &)`
- `void std::gslice_array<_Tp>::operator= (const valarray<_Tp> &) const`
- `void std::mask_array<_Tp>::operator= (const valarray<_Tp> &) const`
- `void std::indirect_array<_Tp>::operator= (const valarray<_Tp> &) const`
- `gslice & std::gslice::operator= (const gslice &)`
- `void std::gslice_array<_Tp>::operator= (const _Tp &) const`
- `void std::mask_array<_Tp>::operator= (const _Tp &) const`
- `void std::indirect_array<_Tp>::operator= (const _Tp &) const`
- `template<class _Dom >`
`void std::gslice_array<_Tp>::operator= (const _Expr<_Dom, _Tp> &) const`
- `template<class _Dom >`
`void std::indirect_array<_Tp>::operator= (const _Expr<_Dom, _Tp> &) const`
- `slice_array & std::slice_array<_Tp>::operator= (const slice_array &)`
- `void std::slice_array<_Tp>::operator= (const valarray<_Tp> &) const`
- `void std::slice_array<_Tp>::operator= (const _Tp &) const`
- `template<class _Dom >`
`void std::slice_array<_Tp>::operator= (const _Expr<_Dom, _Tp> &) const`
- `template<class _Ex >`
`void std::mask_array<_Tp>::operator= (const _Expr<_Ex, _Tp> &_e) const`
- `void std::gslice_array<_Tp>::operator>>= (const valarray<_Tp> &) const`
- `void std::mask_array<_Tp>::operator>>= (const valarray<_Tp> &) const`
- `void std::indirect_array<_Tp>::operator>>= (const valarray<_Tp> &) const`
- `template<class _Dom >`
`void std::gslice_array<_Tp>::operator>>= (const _Expr<_Dom, _Tp> &) const`
- `template<class _Dom >`
`void std::indirect_array<_Tp>::operator>>= (const _Expr<_Dom, _Tp> &) const`
- `template<class _Dom >`
`void std::mask_array<_Tp>::operator>>= (const _Expr<_Dom, _Tp> &) const`
- `void std::slice_array<_Tp>::operator>>= (const valarray<_Tp> &) const`
- `template<class _Dom >`
`void std::slice_array<_Tp>::operator>>= (const _Expr<_Dom, _Tp> &) const`
- `void std::gslice_array<_Tp>::operator^= (const valarray<_Tp> &) const`
- `void std::mask_array<_Tp>::operator^= (const valarray<_Tp> &) const`
- `void std::indirect_array<_Tp>::operator^= (const valarray<_Tp> &) const`
- `template<class _Dom >`
`void std::gslice_array<_Tp>::operator^= (const _Expr<_Dom, _Tp> &) const`

- `template<class _Dom >`
`void std::indirect_array<_Tp>::operator^= (const _Expr<_Dom, _Tp> &) const`
- `template<class _Dom >`
`void std::mask_array<_Tp>::operator^= (const _Expr<_Dom, _Tp> &) const`
- `void std::slice_array<_Tp>::operator^= (const valarray<_Tp> &) const`
- `template<class _Dom >`
`void std::slice_array<_Tp>::operator^= (const _Expr<_Dom, _Tp> &) const`
- `void std::gslice_array<_Tp>::operator|= (const valarray<_Tp> &) const`
- `void std::mask_array<_Tp>::operator|= (const valarray<_Tp> &) const`
- `void std::indirect_array<_Tp>::operator|= (const valarray<_Tp> &) const`
- `template<class _Dom >`
`void std::gslice_array<_Tp>::operator|= (const _Expr<_Dom, _Tp> &) const`
- `template<class _Dom >`
`void std::mask_array<_Tp>::operator|= (const _Expr<_Dom, _Tp> &) const`
- `template<class _Dom >`
`void std::indirect_array<_Tp>::operator|= (const _Expr<_Dom, _Tp> &) const`
- `void std::slice_array<_Tp>::operator|= (const valarray<_Tp> &) const`
- `template<class _Dom >`
`void std::slice_array<_Tp>::operator|= (const _Expr<_Dom, _Tp> &) const`
- `size_t std::slice::size () const`
- `valarray< size_t > std::gslice::size () const`
- `size_t std::slice::start () const`
- `size_t std::gslice::start () const`
- `size_t std::slice::stride () const`
- `valarray< size_t > std::gslice::stride () const`

2.41.1 Detailed Description

2.41.2 Function Documentation

2.41.2.1 `std::gslice::gslice ()` [inline]

Construct an empty slice.

Definition at line 149 of file `gslice.h`.

2.41.2.2 `std::gslice::gslice (size_t __o, const valarray< size_t > & __l, const valarray< size_t > & __s)` [inline]

Construct a slice.

Constructs a slice with as many dimensions as the length of the `l` and `s` arrays.

Parameters

<code>__o</code>	Offset in array of first element.
<code>__l</code>	Array of dimension lengths.
<code>__s</code>	Array of dimension strides between array elements.

Definition at line 153 of file gslice.h.

2.41.2.3 `std::gslice::gslice (const gslice & __g) [inline]`

Copy constructor.

Definition at line 158 of file gslice.h.

2.41.2.4 `template<typename _Tp> std::gslice_array<_Tp>::gslice_array (const gslice_array<_Tp> & __a) [inline]`

Copy constructor. Both slices refer to the same underlying array.

Definition at line 143 of file gslice_array.h.

2.41.2.5 `template<typename _Tp> std::indirect_array<_Tp>::indirect_array (const indirect_array<_Tp> & __a) [inline]`

Copy constructor. Both slices refer to the same underlying array.

Definition at line 143 of file indirect_array.h.

2.41.2.6 `template<typename _Tp> std::mask_array<_Tp>::mask_array (const mask_array<_Tp> & a) [inline]`

Copy constructor. Both slices refer to the same underlying array.

Definition at line 139 of file mask_array.h.

2.41.2.7 `std::slice::slice () [inline]`

Construct an empty slice.

Definition at line 90 of file slice_array.h.

2.41.2.8 `std::slice::slice (size_t __o, size_t __d, size_t __s) [inline]`

Construct a slice.

Parameters

<code>__o</code>	Offset in array of first element.
<code>__d</code>	Number of elements in slice.
<code>__s</code>	Stride between array elements.

Definition at line 94 of file slice_array.h.

2.41.2.9 `template<typename _Tp> std::slice_array<_Tp>::slice_array (const slice_array<_Tp> & a) [inline]`

Copy constructor. Both slices refer to the same underlying array.

Definition at line 207 of file slice_array.h.

2.41.2.10 `std::gslice::~~gslice () [inline]`

Destructor.

Definition at line 163 of file gslice.h.

2.41.2.11 `template<typename _Tp> void std::gslice_array<_Tp>::operator%=(const valarray<_Tp> & __v) const [inline]`

Modulo slice elements by corresponding elements of *v*.

Definition at line 202 of file gslice_array.h.

2.41.2.12 `template<typename _Tp> void std::mask_array<_Tp>::operator%=(const valarray<_Tp> & __v) const [inline]`

Modulo slice elements by corresponding elements of *v*.

Definition at line 192 of file mask_array.h.

2.41.2.13 `template<typename _Tp> void std::indirect_array<_Tp>::operator%=(const valarray<_Tp> & __v) const [inline]`

Modulo slice elements by corresponding elements of *v*.

Definition at line 196 of file indirect_array.h.

2.41.2.14 `template<typename _Tp> void std::slice_array<_Tp>::operator%=(const valarray<_Tp> & __v) const [inline]`

Modulo slice elements by corresponding elements of *v*.

Definition at line 258 of file slice_array.h.

2.41.2.15 `template<typename _Tp> void std::gslice_array<_Tp>::operator&=(const valarray<_Tp> & __v) const [inline]`

Logical and slice elements with corresponding elements of *v*.

Definition at line 206 of file gslice_array.h.

2.41.2.16 `template<typename _Tp> void std::mask_array<_Tp>::operator&= (const valarray<_Tp> &__v) const`
`[inline]`

Logical and slice elements with corresponding elements of *v*.

Definition at line 196 of file `mask_array.h`.

2.41.2.17 `template<typename _Tp> void std::indirect_array<_Tp>::operator&= (const valarray<_Tp> &__v) const`
`[inline]`

Logical and slice elements with corresponding elements of *v*.

Definition at line 200 of file `indirect_array.h`.

2.41.2.18 `template<typename _Tp> void std::slice_array<_Tp>::operator&= (const valarray<_Tp> &__v) const`
`[inline]`

Logical and slice elements with corresponding elements of *v*.

Definition at line 262 of file `slice_array.h`.

2.41.2.19 `template<typename _Tp> void std::gslice_array<_Tp>::operator*=(const valarray<_Tp> &__v) const`
`[inline]`

Multiply slice elements by corresponding elements of *v*.

Definition at line 200 of file `gslice_array.h`.

2.41.2.20 `template<typename _Tp> void std::mask_array<_Tp>::operator*=(const valarray<_Tp> &__v) const`
`[inline]`

Multiply slice elements by corresponding elements of *v*.

Definition at line 190 of file `mask_array.h`.

2.41.2.21 `template<typename _Tp> void std::indirect_array<_Tp>::operator*=(const valarray<_Tp> &__v) const`
`[inline]`

Multiply slice elements by corresponding elements of *v*.

Definition at line 194 of file `indirect_array.h`.

2.41.2.22 `template<typename _Tp> void std::slice_array<_Tp>::operator*=(const valarray<_Tp> &__v) const`
`[inline]`

Multiply slice elements by corresponding elements of *v*.

Definition at line 256 of file `slice_array.h`.

2.41.2.23 `template<typename _Tp> void std::gslice_array<_Tp>::operator+=(const valarray<_Tp> &__v) const`
`[inline]`

Add corresponding elements of *v* to slice elements.

Definition at line 203 of file `gslice_array.h`.

2.41.2.24 `template<typename _Tp> void std::mask_array<_Tp>::operator+=(const valarray<_Tp> &__v) const`
`[inline]`

Add corresponding elements of *v* to slice elements.

Definition at line 193 of file `mask_array.h`.

2.41.2.25 `template<typename _Tp> void std::indirect_array<_Tp>::operator+=(const valarray<_Tp> &__v) const`
`[inline]`

Add corresponding elements of *v* to slice elements.

Definition at line 197 of file `indirect_array.h`.

2.41.2.26 `template<typename _Tp> void std::slice_array<_Tp>::operator+=(const valarray<_Tp> &__v) const`
`[inline]`

Add corresponding elements of *v* to slice elements.

Definition at line 259 of file `slice_array.h`.

2.41.2.27 `template<typename _Tp> void std::gslice_array<_Tp>::operator-=(const valarray<_Tp> &__v) const`
`[inline]`

Subtract corresponding elements of *v* from slice elements.

Definition at line 204 of file `gslice_array.h`.

2.41.2.28 `template<typename _Tp> void std::mask_array<_Tp>::operator-=(const valarray<_Tp> &__v) const`
`[inline]`

Subtract corresponding elements of *v* from slice elements.

Definition at line 194 of file `mask_array.h`.

2.41.2.29 `template<typename _Tp> void std::indirect_array<_Tp>::operator-=(const valarray<_Tp> &__v) const`
`[inline]`

Subtract corresponding elements of *v* from slice elements.

Definition at line 198 of file `indirect_array.h`.

2.41.2.30 `template<typename _Tp> void std::slice_array<_Tp>::operator-= (const valarray<_Tp> &__v) const`
`[inline]`

Subtract corresponding elements of *v* from slice elements.

Definition at line 260 of file `slice_array.h`.

2.41.2.31 `template<typename _Tp> void std::gslice_array<_Tp>::operator/= (const valarray<_Tp> &__v) const`
`[inline]`

Divide slice elements by corresponding elements of *v*.

Definition at line 201 of file `gslice_array.h`.

2.41.2.32 `template<typename _Tp> void std::mask_array<_Tp>::operator/= (const valarray<_Tp> &__v) const`
`[inline]`

Divide slice elements by corresponding elements of *v*.

Definition at line 191 of file `mask_array.h`.

2.41.2.33 `template<typename _Tp> void std::indirect_array<_Tp>::operator/= (const valarray<_Tp> &__v) const`
`[inline]`

Divide slice elements by corresponding elements of *v*.

Definition at line 195 of file `indirect_array.h`.

2.41.2.34 `template<typename _Tp> void std::slice_array<_Tp>::operator<=<= (const valarray<_Tp> &__v) const`
`[inline]`

Divide slice elements by corresponding elements of *v*.

Definition at line 257 of file `slice_array.h`.

2.41.2.35 `template<typename _Tp> void std::gslice_array<_Tp>::operator<<= (const valarray<_Tp> &__v) const`
`[inline]`

Left shift slice elements by corresponding elements of *v*.

Definition at line 208 of file `gslice_array.h`.

2.41.2.36 `template<typename _Tp> void std::mask_array<_Tp>::operator<<= (const valarray<_Tp> &__v) const`
`[inline]`

Left shift slice elements by corresponding elements of *v*.

Definition at line 198 of file `mask_array.h`.

2.41.2.37 `template<typename _Tp> void std::indirect_array<_Tp>::operator<<= (const valarray<_Tp> & __v) const [inline]`

Left shift slice elements by corresponding elements of *v*.

Definition at line 202 of file indirect_array.h.

2.41.2.38 `template<typename _Tp> void std::slice_array<_Tp>::operator<<= (const valarray<_Tp> & __v) const [inline]`

Left shift slice elements by corresponding elements of *v*.

Definition at line 264 of file slice_array.h.

2.41.2.39 `template<typename _Tp> gslice_array<_Tp> & std::gslice_array<_Tp>::operator= (const gslice_array<_Tp> & __a) [inline]`

Assignment operator. Assigns slice elements to corresponding elements of *a*.

Definition at line 148 of file gslice_array.h.

Referenced by std::gslice_array<_Tp>::operator=().

2.41.2.40 `template<typename _Tp> indirect_array<_Tp> & std::indirect_array<_Tp>::operator= (const indirect_array<_Tp> & __a) [inline]`

Assignment operator. Assigns elements to corresponding elements of *a*.

Definition at line 154 of file indirect_array.h.

Referenced by std::indirect_array<_Tp>::operator=().

2.41.2.41 `template<typename _Tp> mask_array<_Tp> & std::mask_array<_Tp>::operator= (const mask_array<_Tp> & __a) [inline]`

Assignment operator. Assigns elements to corresponding elements of *a*.

Definition at line 149 of file mask_array.h.

Referenced by std::mask_array<_Tp>::operator=().

2.41.2.42 `template<typename _Tp> void std::gslice_array<_Tp>::operator= (const valarray<_Tp> & __v) const [inline]`

Assign slice elements to corresponding elements of *v*.

Definition at line 166 of file gslice_array.h.

References std::gslice_array<_Tp>::operator=().

2.41.2.43 `template<typename _Tp> void std::indirect_array<_Tp>::operator=(const valarray<_Tp> &__v) const`
`[inline]`

Assign slice elements to corresponding elements of *v*.

Definition at line 168 of file `indirect_array.h`.

References `std::indirect_array<_Tp>::operator=()`.

2.41.2.44 `gslice & std::gslice::operator=(const gslice &__g)` `[inline]`

Assignment operator.

Definition at line 170 of file `gslice.h`.

2.41.2.45 `template<typename _Tp> void std::gslice_array<_Tp>::operator=(const _Tp &__t) const` `[inline]`

Assign all slice elements to *t*.

Definition at line 158 of file `gslice_array.h`.

2.41.2.46 `template<typename _Tp> void std::mask_array<_Tp>::operator=(const _Tp &__t) const` `[inline]`

Assign all slice elements to *t*.

Definition at line 158 of file `mask_array.h`.

References `std::mask_array<_Tp>::operator=()`.

2.41.2.47 `template<typename _Tp> void std::indirect_array<_Tp>::operator=(const _Tp &__t) const` `[inline]`

Assign all slice elements to *t*.

Definition at line 163 of file `indirect_array.h`.

2.41.2.48 `template<typename _Tp> slice_array<_Tp> & std::slice_array<_Tp>::operator=(const slice_array<_Tp> &__a)` `[inline]`

Assignment operator. Assigns slice elements to corresponding elements of *a*.

Definition at line 215 of file `slice_array.h`.

Referenced by `std::slice_array<_Tp>::operator=()`.

2.41.2.49 `template<typename _Tp> void std::slice_array<_Tp>::operator=(const valarray<_Tp> &__v) const`
`[inline]`

Assign slice elements to corresponding elements of *v*.

Definition at line 229 of file `slice_array.h`.

References `std::slice_array<_Tp>::operator=()`.

2.41.2.50 `template<typename _Tp> void std::slice_array<_Tp>::operator=(const _Tp & __t) const [inline]`

Assign all slice elements to *t*.

Definition at line 224 of file slice_array.h.

2.41.2.51 `template<typename _Tp> void std::gslice_array<_Tp>::operator>=(const valarray<_Tp> & __v) const [inline]`

Right shift slice elements by corresponding elements of *v*.

Definition at line 209 of file gslice_array.h.

2.41.2.52 `template<typename _Tp> void std::mask_array<_Tp>::operator>=(const valarray<_Tp> & __v) const [inline]`

Right shift slice elements by corresponding elements of *v*.

Definition at line 199 of file mask_array.h.

2.41.2.53 `template<typename _Tp> void std::indirect_array<_Tp>::operator>=(const valarray<_Tp> & __v) const [inline]`

Right shift slice elements by corresponding elements of *v*.

Definition at line 203 of file indirect_array.h.

2.41.2.54 `template<typename _Tp> void std::slice_array<_Tp>::operator>=(const valarray<_Tp> & __v) const [inline]`

Right shift slice elements by corresponding elements of *v*.

Definition at line 265 of file slice_array.h.

2.41.2.55 `template<typename _Tp> void std::gslice_array<_Tp>::operator^=(const valarray<_Tp> & __v) const [inline]`

Logical xor slice elements with corresponding elements of *v*.

Definition at line 205 of file gslice_array.h.

2.41.2.56 `template<typename _Tp> void std::mask_array<_Tp>::operator^=(const valarray<_Tp> & __v) const [inline]`

Logical xor slice elements with corresponding elements of *v*.

Definition at line 195 of file mask_array.h.

2.41.2.57 `template<typename _Tp> void std::indirect_array<_Tp>::operator^= (const valarray<_Tp> & __v) const`
`[inline]`

Logical xor slice elements with corresponding elements of *v*.

Definition at line 199 of file indirect_array.h.

2.41.2.58 `template<typename _Tp> void std::slice_array<_Tp>::operator^= (const valarray<_Tp> & __v) const`
`[inline]`

Logical xor slice elements with corresponding elements of *v*.

Definition at line 261 of file slice_array.h.

2.41.2.59 `template<typename _Tp> void std::gslice_array<_Tp>::operator|= (const valarray<_Tp> & __v) const`
`[inline]`

Logical or slice elements with corresponding elements of *v*.

Definition at line 207 of file gslice_array.h.

2.41.2.60 `template<typename _Tp> void std::mask_array<_Tp>::operator|= (const valarray<_Tp> & __v) const`
`[inline]`

Logical or slice elements with corresponding elements of *v*.

Definition at line 197 of file mask_array.h.

2.41.2.61 `template<typename _Tp> void std::indirect_array<_Tp>::operator|= (const valarray<_Tp> & __v) const`
`[inline]`

Logical or slice elements with corresponding elements of *v*.

Definition at line 201 of file indirect_array.h.

2.41.2.62 `template<typename _Tp> void std::slice_array<_Tp>::operator|= (const valarray<_Tp> & __v) const`
`[inline]`

Logical or slice elements with corresponding elements of *v*.

Definition at line 263 of file slice_array.h.

2.41.2.63 `size_t std::slice::size () const` `[inline]`

Return size of slice.

Definition at line 102 of file slice_array.h.

2.41.2.64 `valarray< size_t > std::gslice::size () const` `[inline]`

Return array of sizes of slice dimensions.

Definition at line 139 of file gslice.h.

2.41.2.65 `size_t std::slice::start () const` `[inline]`

Return array offset of first slice element.

Definition at line 98 of file slice_array.h.

2.41.2.66 `size_t std::gslice::start () const` `[inline]`

Return array offset of first slice element.

Definition at line 135 of file gslice.h.

2.41.2.67 `size_t std::slice::stride () const` `[inline]`

Return array stride of slice.

Definition at line 106 of file slice_array.h.

2.41.2.68 `valarray< size_t > std::gslice::stride () const` `[inline]`

Return array of array strides for each dimension.

Definition at line 143 of file gslice.h.

2.42 Pointer_abstractions

Classes

- struct `std::default_delete<_Tp>`
- struct `std::default_delete<_Tp[]>`
- class `std::enable_shared_from_this<_Tp>`
- struct `std::hash<shared_ptr<_Tp>>`
- struct `std::hash<unique_ptr<_Tp,_Dp>>`
- struct `std::owner_less<_Tp>`
- struct `std::owner_less<shared_ptr<_Tp>>`
- struct `std::owner_less<weak_ptr<_Tp>>`
- struct `std::pointer_traits<_Ptr>`
- struct `std::pointer_traits<_Tp*>`
- class `std::shared_ptr<_Tp>`
- class `std::unique_ptr<_Tp,_Dp>`
- class `std::unique_ptr<_Tp[],_Dp>`
- class `std::weak_ptr<_Tp>`

Functions

- template<typename _Tp1, typename _Tp2>
void **std::enable_shared_from_this_helper** (const __shared_count<> &__pn, const enable_shared_from_this<_Tp1> *__pe, const _Tp2 *__px) noexcept
- template<typename _Tp, typename _Alloc, typename... _Args>
shared_ptr<_Tp> **std::allocate_shared** (const _Alloc &__a, _Args &&...__args)
- template<typename _Tp, typename _Tp1>
shared_ptr<_Tp> **std::const_pointer_cast** (const shared_ptr<_Tp1> &__r) noexcept
- template<typename _Tp, typename _Tp1>
shared_ptr<_Tp> **std::dynamic_pointer_cast** (const shared_ptr<_Tp1> &__r) noexcept
- template<typename _Del, typename _Tp, _Lock_policy _Lp>
_Del * **std::get_deleter** (const __shared_ptr<_Tp,_Lp> &__p) noexcept
- template<typename _Tp, typename... _Args>
shared_ptr<_Tp> **std::make_shared** (_Args &&...__args)
- template<typename _Tp1, typename _Tp2>
bool **std::operator!=** (const shared_ptr<_Tp1> &__a, const shared_ptr<_Tp2> &__b) noexcept
- template<typename _Tp>
bool **std::operator!=** (const shared_ptr<_Tp> &__a, nullptr_t) noexcept
- template<typename _Tp>
bool **std::operator!=** (nullptr_t, const shared_ptr<_Tp> &__a) noexcept
- template<typename _Tp, typename _Dp, typename _Up, typename _Ep>
bool **std::operator!=** (const unique_ptr<_Tp,_Dp> &__x, const unique_ptr<_Up,_Ep> &__y)
- template<typename _Tp, typename _Dp>
bool **std::operator!=** (const unique_ptr<_Tp,_Dp> &__x, nullptr_t) noexcept
- template<typename _Tp, typename _Dp>
bool **std::operator!=** (nullptr_t, const unique_ptr<_Tp,_Dp> &__x) noexcept
- template<typename _Tp1, typename _Tp2>
bool **std::operator<** (const shared_ptr<_Tp1> &__a, const shared_ptr<_Tp2> &__b) noexcept
- template<typename _Tp>
bool **std::operator<** (const shared_ptr<_Tp> &__a, nullptr_t) noexcept

- `template<typename _Tp >`
`bool std::operator< (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`
`bool std::operator< (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp >`
`bool std::operator< (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)`
- `template<typename _Tp, typename _Dp >`
`bool std::operator< (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)`
- `template<typename _Ch, typename _Tr, typename _Tp, _Lock_policy _Lp>`
`std::basic_ostream< _Ch, _Tr > & std::operator<< (std::basic_ostream< _Ch, _Tr > &__os, const __shared_ptr< _Tp, _Lp > &__p)`
- `template<typename _Tp1, typename _Tp2 >`
`bool std::operator<= (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept`
- `template<typename _Tp >`
`bool std::operator<= (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp >`
`bool std::operator<= (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`
`bool std::operator<= (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp >`
`bool std::operator<= (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)`
- `template<typename _Tp, typename _Dp >`
`bool std::operator<= (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)`
- `template<typename _Tp1, typename _Tp2 >`
`bool std::operator== (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept`
- `template<typename _Tp >`
`bool std::operator== (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp >`
`bool std::operator== (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`
`bool std::operator== (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp >`
`bool std::operator== (const unique_ptr< _Tp, _Dp > &__x, nullptr_t) noexcept`
- `template<typename _Tp, typename _Dp >`
`bool std::operator== (nullptr_t, const unique_ptr< _Tp, _Dp > &__x) noexcept`
- `template<typename _Tp1, typename _Tp2 >`
`bool std::operator> (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept`
- `template<typename _Tp >`
`bool std::operator> (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp >`
`bool std::operator> (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`
`bool std::operator> (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp >`
`bool std::operator> (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)`
- `template<typename _Tp, typename _Dp >`
`bool std::operator> (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)`
- `template<typename _Tp1, typename _Tp2 >`
`bool std::operator>= (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept`
- `template<typename _Tp >`
`bool std::operator>= (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp >`
`bool std::operator>= (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`

- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`
`bool std::operator>= (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp >`
`bool std::operator>= (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)`
- `template<typename _Tp, typename _Dp >`
`bool std::operator>= (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)`
- `template<typename _Tp, typename _Tp1 >`
`shared_ptr< _Tp > std::static_pointer_cast (const shared_ptr< _Tp1 > &__r) noexcept`
- `template<typename _Tp >`
`void std::swap (shared_ptr< _Tp > &__a, shared_ptr< _Tp > &__b) noexcept`
- `template<typename _Tp >`
`void std::swap (weak_ptr< _Tp > &__a, weak_ptr< _Tp > &__b) noexcept`
- `template<typename _Tp, typename _Dp >`
`void std::swap (unique_ptr< _Tp, _Dp > &__x, unique_ptr< _Tp, _Dp > &__y) noexcept`

2.42.1 Detailed Description

2.42.2 Function Documentation

- 2.42.2.1 `template<typename _Tp, typename _Alloc, typename... _Args> shared_ptr<_Tp> std::allocate_shared (const _Alloc & __a, _Args &&... __args) [inline]`

Create an object that is owned by a `shared_ptr`.

Parameters

<code>__a</code>	An allocator.
<code>__args</code>	Arguments for the <code>_Tp</code> object's constructor.

Returns

A `shared_ptr` that owns the newly created object.

Exceptions

<i>An</i>	exception thrown from <code>_Alloc::allocate</code> or from the constructor of <code>_Tp</code> .
-----------	---

A copy of `__a` will be used to allocate memory for the `shared_ptr` and the new object.

Definition at line 591 of file `shared_ptr.h`.

- 2.42.2.2 `template<typename _Del, typename _Tp, _Lock_policy _Lp> _Del* std::get_deleter (const __shared_ptr< _Tp, _Lp > & __p) [inline], [noexcept]`

20.7.2.2.10 `shared_ptr` `get_deleter`

Definition at line 76 of file `shared_ptr.h`.

Referenced by `std::unique_ptr< std::vector< bool > >::operator=()`, `std::unique_ptr< _Tp[], _Dp >::operator=()`, `std::unique_ptr< std::vector< bool > >::reset()`, `std::unique_ptr< _Tp[], _Dp >::reset()`, `std::unique_ptr< std::vector< bool > >::~unique_ptr()`, and `std::unique_ptr< _Tp[], _Dp >::~unique_ptr()`.

2.42.2.3 `template<typename _Tp, typename... _Args> shared_ptr<_Tp> std::make_shared (_Args &&... __args) [inline]`

Create an object that is owned by a `shared_ptr`.

Parameters

<code>__args</code>	Arguments for the <code>_Tp</code> object's constructor.
---------------------	--

Returns

A `shared_ptr` that owns the newly created object.

Exceptions

<code>std::bad_alloc</code> , or	an exception thrown from the constructor of <code>_Tp</code> .
----------------------------------	--

Definition at line 606 of file `shared_ptr.h`.

2.42.2.4 `template<typename _Ch, typename _Tr, typename _Tp, _Lock_policy _Lp> std::basic_ostream<_Ch, _Tr>& std::operator<< (std::basic_ostream<_Ch, _Tr> &__os, const __shared_ptr<_Tp, _Lp> &__p) [inline]`

20.7.2.2.11 `shared_ptr` I/O

Definition at line 66 of file `shared_ptr.h`.

2.43 Poisson Distributions

Collaboration diagram for Poisson Distributions:



Classes

- class `std::discrete_distribution< _IntType >`
- class `std::exponential_distribution< _RealType >`
- class `std::extreme_value_distribution< _RealType >`
- class `std::piecewise_constant_distribution< _RealType >`
- class `std::piecewise_linear_distribution< _RealType >`
- class `std::poisson_distribution< _IntType >`
- class `std::weibull_distribution< _RealType >`

Functions

- `template<typename _IntType >`
`bool std::operator!= (const std::poisson_distribution< _IntType > &__d1, const std::poisson_distribution< _IntType > &__d2)`
- `template<typename _RealType >`
`bool std::operator!= (const std::exponential_distribution< _RealType > &__d1, const std::exponential_distribution< _RealType > &__d2)`
- `template<typename _RealType >`
`bool std::operator!= (const std::weibull_distribution< _RealType > &__d1, const std::weibull_distribution< _RealType > &__d2)`
- `template<typename _RealType >`
`bool std::operator!= (const std::extreme_value_distribution< _RealType > &__d1, const std::extreme_value_distribution< _RealType > &__d2)`
- `template<typename _IntType >`
`bool std::operator!= (const std::discrete_distribution< _IntType > &__d1, const std::discrete_distribution< _IntType > &__d2)`
- `template<typename _RealType >`
`bool std::operator!= (const std::piecewise_constant_distribution< _RealType > &__d1, const std::piecewise_constant_distribution< _RealType > &__d2)`
- `template<typename _RealType >`
`bool std::operator!= (const std::piecewise_linear_distribution< _RealType > &__d1, const std::piecewise_linear_distribution< _RealType > &__d2)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::exponential_distribution< _RealType > &__x)`

- `template<typename _RealType, typename _CharT, typename _Traits >
std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const
std::weibull_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >
std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const
std::extreme_value_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >
std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::←
::exponential_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >
std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::←
::weibull_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >
std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::←
::extreme_value_distribution< _RealType > &__x)`

2.43.1 Detailed Description

2.43.2 Function Documentation

2.43.2.1 `template<typename _IntType> bool std::operator!=(const std::poisson_distribution< _IntType > &__d1, const
std::poisson_distribution< _IntType > &__d2) [inline]`

Return true if two Poisson distributions are different.

Definition at line 4627 of file random.h.

2.43.2.2 `template<typename _RealType> bool std::operator!=(const std::exponential_distribution< _RealType > &__d1,
const std::exponential_distribution< _RealType > &__d2) [inline]`

Return true if two exponential distributions have different parameters.

Definition at line 4805 of file random.h.

References `std::operator>>()`.

2.43.2.3 `template<typename _RealType> bool std::operator!=(const std::weibull_distribution< _RealType > &__d1, const
std::weibull_distribution< _RealType > &__d2) [inline]`

Return true if two Weibull distributions have different parameters.

Definition at line 5008 of file random.h.

References `std::operator>>()`.

2.43.2.4 `template<typename _RealType> bool std::operator!=(const std::extreme_value_distribution< _RealType > &
__d1, const std::extreme_value_distribution< _RealType > &__d2) [inline]`

Return true if two extreme value distributions have different parameters.

Definition at line 5211 of file random.h.

References `std::operator>>()`.

2.43.2.5 `template<typename _IntType> bool std::operator!= (const std::discrete_distribution< _IntType> &__d1, const std::discrete_distribution< _IntType> &__d2) [inline]`

Return true if two discrete distributions have different parameters.

Definition at line 5471 of file random.h.

2.43.2.6 `template<typename _RealType> bool std::operator!= (const std::piecewise_constant_distribution< _RealType> &__d1, const std::piecewise_constant_distribution< _RealType> &__d2) [inline]`

Return true if two piecewise constant distributions have different parameters.

Definition at line 5738 of file random.h.

2.43.2.7 `template<typename _RealType> bool std::operator!= (const std::piecewise_linear_distribution< _RealType> &__d1, const std::piecewise_linear_distribution< _RealType> &__d2) [inline]`

Return true if two piecewise linear distributions have different parameters.

Definition at line 6008 of file random.h.

2.43.2.8 `template<typename _RealType, typename _CharT, typename _Traits> std::basic_ostream< _CharT, _Traits> & std::operator<< (std::basic_ostream< _CharT, _Traits> &__os, const std::exponential_distribution< _RealType> &__x)`

Inserts a `exponential_distribution` random number distribution `__x` into the output stream `__os`.

Parameters

<code>__os</code>	An output stream.
<code>__x</code>	A <code>exponential_distribution</code> random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

2.43.2.9 `template<typename _RealType, typename _CharT, typename _Traits> std::basic_ostream< _CharT, _Traits> & std::operator<< (std::basic_ostream< _CharT, _Traits> &__os, const std::weibull_distribution< _RealType> &__x)`

Inserts a `weibull_distribution` random number distribution `__x` into the output stream `__os`.

Parameters

<code>__os</code>	An output stream.
<code>__x</code>	A <code>weibull_distribution</code> random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

```
2.43.2.10 template<typename _RealType , typename _CharT , typename _Traits > std::basic_ostream<_CharT, _Traits>&
std::operator<< ( std::basic_ostream< _CharT, _Traits > & __os, const std::extreme_value_distribution<
_RealType > & __x )
```

Inserts a `extreme_value_distribution` random number distribution `__x` into the output stream `__os`.

Parameters

<code>__os</code>	An output stream.
<code>__x</code>	A <code>extreme_value_distribution</code> random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

```
2.43.2.11 template<typename _RealType , typename _CharT , typename _Traits > std::basic_istream<_CharT, _Traits>&
std::operator>> ( std::basic_istream< _CharT, _Traits > & __is, std::exponential_distribution< _RealType > &
__x )
```

Extracts a `exponential_distribution` random number distribution `__x` from the input stream `__is`.

Parameters

<code>__is</code>	An input stream.
<code>__x</code>	A <code>exponential_distribution</code> random number generator engine.

Returns

The input stream with `__x` extracted or in an error state.

```
2.43.2.12 template<typename _RealType , typename _CharT , typename _Traits > std::basic_istream<_CharT, _Traits>&
std::operator>> ( std::basic_istream< _CharT, _Traits > & __is, std::weibull_distribution< _RealType > & __x )
```

Extracts a `weibull_distribution` random number distribution `__x` from the input stream `__is`.

Parameters

<code>__is</code>	An input stream.
<code>__x</code>	A <code>weibull_distribution</code> random number generator engine.

Returns

The input stream with ___x extracted or in an error state.

2.43.2.13 `template<typename _RealType , typename _CharT , typename _Traits > std::basic_istream<_CharT, _Traits>& std::operator>> (std::basic_istream<_CharT, _Traits> & __is, std::extreme_value_distribution<_RealType> & __x)`

Extracts a `extreme_value_distribution` random number distribution ___x from the input stream ___is.

Parameters

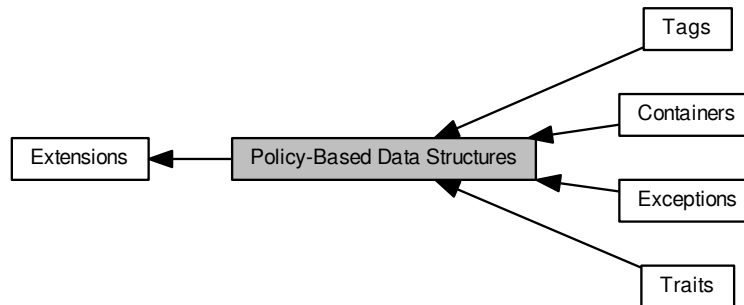
<code>__is</code>	An input stream.
<code>__x</code>	A <code>extreme_value_distribution</code> random number generator engine.

Returns

The input stream with ___x extracted or in an error state.

2.44 Policy-Based Data Structures

Collaboration diagram for Policy-Based Data Structures:



Modules

- [Containers](#)
- [Exceptions](#)
- [Tags](#)
- [Traits](#)

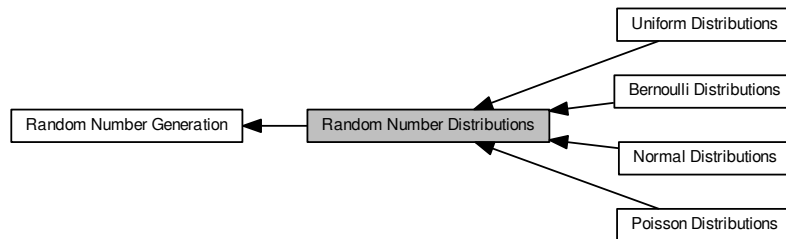
2.44.1 Detailed Description

This is a library of policy-based elementary data structures: associative containers and priority queues. It is designed for high-performance, flexibility, semantic safety, and conformance to the corresponding containers in std (except for some points where it differs by design).

For details, see: http://gcc.gnu.org/onlinedocs/libstdc++/ext/pb_ds/index.html

2.45 Random Number Distributions

Collaboration diagram for Random Number Distributions:



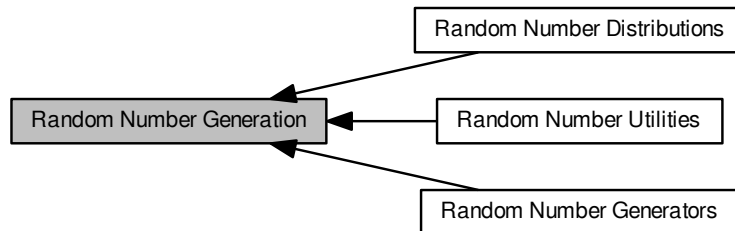
Modules

- [Bernoulli Distributions](#)
- [Normal Distributions](#)
- [Poisson Distributions](#)
- [Uniform Distributions](#)

2.45.1 Detailed Description

2.46 Random Number Generation

Collaboration diagram for Random Number Generation:



Modules

- [Random Number Distributions](#)
- [Random Number Generators](#)
- [Random Number Utilities](#)

Namespaces

- [std::__detail](#)

Functions

- `template<typename _RealType, size_t __bits, typename _UniformRandomNumberGenerator > _RealType std::generate_canonical (_UniformRandomNumberGenerator &__g)`

2.46.1 Detailed Description

A facility for generating random numbers on selected distributions.

2.46.2 Function Documentation

2.46.2.1 `template<typename _RealType, size_t __bits, typename _UniformRandomNumberGenerator > _RealType std::generate_canonical (_UniformRandomNumberGenerator &__g)`

A function template for converting the output of a (integral) uniform random number generator to a floatng point result in the range [0-1).

2.47 Random Number Generators

Collaboration diagram for Random Number Generators:



Classes

- class `std::discard_block_engine< _RandomNumberEngine, __p, __r >`
- class `std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >`
- class `std::linear_congruential_engine< _UIntType, __a, __c, __m >`
- class `std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >`
- class `std::random_device`
- class `std::shuffle_order_engine< _RandomNumberEngine, __k >`

Typedefs

- typedef `minstd_rand0` **`std::default_random_engine`**
- typedef `shuffle_order_engine< minstd_rand0, 256 >` **`std::knuth_b`**
- typedef `linear_congruential_engine< uint_fast32_t, 48271UL, 0UL, 2147483647UL >` `std::minstd_rand`
- typedef `linear_congruential_engine< uint_fast32_t, 16807UL, 0UL, 2147483647UL >` `std::minstd_rand0`
- typedef `mersenne_twister_engine< uint_fast32_t, 32, 624, 397, 31, 0x9908b0dfUL, 11, 0xffffffffUL, 7, 0x9d2c5680UL, 15, 0xefc60000UL, 18, 1812433253UL >` `std::mt19937`
- typedef `mersenne_twister_engine< uint_fast64_t, 64, 312, 156, 31, 0xb5026f5aa96619e9ULL, 29, 0x5555555555555555ULL, 17, 0x71d67ffeda60000ULL, 37, 0xfff7eee000000000ULL, 43, 6364136223846793005ULL >` `std::mt19937_64`
- typedef `discard_block_engine< ranlux24_base, 223, 23 >` **`std::ranlux24`**
- typedef `subtract_with_carry_engine< uint_fast32_t, 24, 10, 24 >` **`std::ranlux24_base`**
- typedef `discard_block_engine< ranlux48_base, 389, 11 >` **`std::ranlux48`**
- typedef `subtract_with_carry_engine< uint_fast64_t, 48, 5, 12 >` **`std::ranlux48_base`**

Functions

- template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
bool `std::operator!=` (const `std::linear_congruential_engine< _UIntType, __a, __c, __m >` &__lhs, const `std::linear_congruential_engine< _UIntType, __a, __c, __m >` &__rhs)
- template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f>
bool `std::operator!=` (const `std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >` &__lhs, const `std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >` &__rhs)

- `template<typename _UIntType, size_t __w, size_t __s, size_t __r>`
`bool std::operator!= (const std::subtract_with_carry_engine< _UIntType, __w, __s, __r > &__lhs, const std::subtract_with_carry_engine< _UIntType, __w, __s, __r > &__rhs)`
- `template<typename _RandomNumberEngine, size_t __p, size_t __r>`
`bool std::operator!= (const std::discard_block_engine< _RandomNumberEngine, __p, __r > &__lhs, const std::discard_block_engine< _RandomNumberEngine, __p, __r > &__rhs)`
- `template<typename _RandomNumberEngine, size_t __w, typename _UIntType >`
`bool std::operator!= (const std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > &__lhs, const std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > &__rhs)`
- `template<typename _RandomNumberEngine, size_t __k>`
`bool std::operator!= (const std::shuffle_order_engine< _RandomNumberEngine, __k > &__lhs, const std::shuffle_order_engine< _RandomNumberEngine, __k > &__rhs)`
- `template<typename _RandomNumberEngine, size_t __w, typename _UIntType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > &__x)`

2.47.1 Detailed Description

These classes define objects which provide random or pseudorandom numbers, either from a discrete or a continuous interval. The random number generator supplied as a part of this library are all uniform random number generators which provide a sequence of random number uniformly distributed over their range.

A number generator is a function object with an operator() that takes zero arguments and returns a number.

A compliant random number generator must satisfy the following requirements.

Table 106 Random Number Generator Requirements

To be documented.

2.47.2 Typedef Documentation

2.47.2.1 `typedef linear_congruential_engine<uint_fast32_t, 48271UL, 0UL, 2147483647UL> std::minstd_rand`

An alternative LCR (Lehmer Generator function).

Definition at line 1529 of file random.h.

2.47.2.2 `typedef linear_congruential_engine<uint_fast32_t, 16807UL, 0UL, 2147483647UL> std::minstd_rand0`

The classic Minimum Standard rand0 of Lewis, Goodman, and Miller.

Definition at line 1523 of file random.h.

2.47.2.3 `typedef mersenne_twister_engine< uint_fast32_t, 32, 624, 397, 31, 0x9908b0dfUL, 11, 0xffffffffUL, 7, 0x9d2c5680UL, 15, 0xefc60000UL, 18, 1812433253UL> std::mt19937`

The classic Mersenne Twister.

Reference: M. Matsumoto and T. Nishimura, Mersenne Twister: A 623-Dimensionally Equidistributed Uniform Pseudo-Random Number Generator, ACM Transactions on Modeling and Computer Simulation, Vol. 8, No. 1, January 1998, pp 3-30.

Definition at line 1545 of file random.h.

2.47.2.4 `typedef mersenne_twister_engine< uint_fast64_t, 64, 312, 156, 31, 0xb5026f5aa96619e9ULL, 29, 0x5555555555555555ULL, 17, 0x71d67ffeda60000ULL, 37, 0xffff7eee00000000ULL, 43, 6364136223846793005ULL> std::mt19937_64`

An alternative Mersenne Twister.

Definition at line 1557 of file random.h.

2.47.3 Function Documentation

2.47.3.1 `template<typename UIntType, UIntType __a, UIntType __c, UIntType __m> bool std::operator!=(const std::linear_congruential_engine< UIntType, __a, __c, __m > & __lhs, const std::linear_congruential_engine< UIntType, __a, __c, __m > & __rhs) [inline]`

Compares two linear congruential random number generator objects of the same type for inequality.

Parameters

<code>__lhs</code>	A linear congruential random number generator object.
<code>__rhs</code>	Another linear congruential random number generator object.

Returns

true if the infinite sequences of generated values would be different, false otherwise.

Definition at line 411 of file random.h.

2.47.3.2 `template<typename UIntType, size_t __w, size_t __n, size_t __m, size_t __r, UIntType __a, size_t __u, UIntType __d, size_t __s, UIntType __b, size_t __t, UIntType __c, size_t __l, UIntType __f> bool std::operator!=(const std::mersenne_twister_engine< UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > & __lhs, const std::mersenne_twister_engine< UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > & __rhs) [inline]`

Compares two % mersenne_twister_engine random number generator objects of the same type for inequality.

Parameters

<code>__lhs</code>	A % mersenne_twister_engine random number generator object.
<code>__rhs</code>	Another % mersenne_twister_engine random number generator object.

Returns

true if the infinite sequences of generated values would be different, false otherwise.

Definition at line 643 of file random.h.

References `std::equal()`, `std::max()`, `std::min()`, and `std::operator>>()`.

2.47.3.3 `template<typename UIntType , size_t __w, size_t __s, size_t __r> bool std::operator!= (const std::subtract_with_carry_engine< UIntType, __w, __s, __r > & __lhs, const std::subtract_with_carry_engine< UIntType, __w, __s, __r > & __rhs) [inline]`

Compares two % subtract_with_carry_engine random number generator objects of the same type for inequality.

Parameters

<code>__lhs</code>	A % subtract_with_carry_engine random number generator object.
<code>__rhs</code>	Another % subtract_with_carry_engine random number generator object.

Returns

true if the infinite sequences of generated values would be different, false otherwise.

Definition at line 842 of file random.h.

2.47.3.4 `template<typename _RandomNumberEngine , size_t __p, size_t __r> bool std::operator!= (const std::discard_block_engine< _RandomNumberEngine, __p, __r > & __lhs, const std::discard_block_engine< _RandomNumberEngine, __p, __r > & __rhs) [inline]`

Compares two discard_block_engine random number generator objects of the same type for inequality.

Parameters

<code>__lhs</code>	A discard_block_engine random number generator object.
<code>__rhs</code>	Another discard_block_engine random number generator object.

Returns

true if the infinite sequences of generated values would be different, false otherwise.

Definition at line 1064 of file random.h.

2.47.3.5 `template<typename _RandomNumberEngine , size_t __w, typename UIntType > bool std::operator!= (const std::independent_bits_engine< _RandomNumberEngine, __w, UIntType > & __lhs, const std::independent_bits_engine< _RandomNumberEngine, __w, UIntType > & __rhs) [inline]`

Compares two independent_bits_engine random number generator objects of the same type for inequality.

Parameters

<code>__lhs</code>	A independent_bits_engine random number generator object.
<code>__rhs</code>	Another independent_bits_engine random number generator object.

Returns

true if the infinite sequences of generated values would be different, false otherwise.

Definition at line 1260 of file random.h.

```
2.47.3.6 template<typename _RandomNumberEngine, size_t __k> bool std::operator!=( const std::shuffle_order_engine<
    _RandomNumberEngine, __k> & __lhs, const std::shuffle_order_engine< _RandomNumberEngine, __k> & __rhs )
    [inline]
```

Compares two shuffle_order_engine random number generator objects of the same type for inequality.

Parameters

<code>__lhs</code>	A shuffle_order_engine random number generator object.
<code>__rhs</code>	Another shuffle_order_engine random number generator object.

Returns

true if the infinite sequences of generated values would be different, false otherwise.

Definition at line 1512 of file random.h.

```
2.47.3.7 template<typename _RandomNumberEngine, size_t __w, typename _UIntType, typename _CharT, typename _Traits
    > std::basic_ostream<_CharT, _Traits>& std::operator<< ( std::basic_ostream<_CharT, _Traits> & __os, const
    std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType> & __x )
```

Inserts the current state of a independent_bits_engine random number generator engine `__x` into the output stream `__os`.

Parameters

<code>__os</code>	An output stream.
<code>__x</code>	A independent_bits_engine random number generator engine.

Returns

The output stream with the state of `__x` inserted or in an error state.

Definition at line 1279 of file random.h.

References `std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType>::base()`.

2.48 Random Number Utilities

Collaboration diagram for Random Number Utilities:



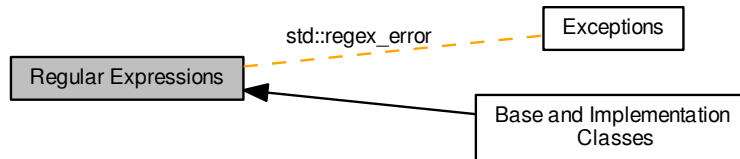
Classes

- class [std::seed_seq](#)

2.48.1 Detailed Description

2.49 Regular Expressions

Collaboration diagram for Regular Expressions:



Modules

- [Base and Implementation Classes](#)

Namespaces

- [std::regex_constants](#)

Classes

- class [std::basic_regex< _Ch_type, _Rx_traits >](#)
- class [std::match_results< _Bi_iter, _Alloc >](#)
- class [std::regex_error](#)
- class [std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >](#)
- class [std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >](#)
- struct [std::regex_traits< _Ch_type >](#)
- class [std::sub_match< _Biter >](#)

Typedefs

- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`using std::__sub_match_string = basic_string< typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc >`
- `typedef match_results< const char * > std::cmatch`
- `typedef regex_iterator< const char * > std::cregex_iterator`
- `typedef regex_token_iterator< const char * > std::cregex_token_iterator`
- `typedef sub_match< const char * > std::csub_match`
- `typedef basic_regex< char > std::regex`
- `typedef match_results< string::const_iterator > std::smatch`
- `typedef regex_iterator< string::const_iterator > std::sregex_iterator`
- `typedef regex_token_iterator< string::const_iterator > std::sregex_token_iterator`

- `typedef sub_match< string::const_iterator > std::ssub_match`
- `typedef match_results< const wchar_t * > std::wcmatch`
- `typedef regex_iterator< const wchar_t * > std::wcregex_iterator`
- `typedef regex_token_iterator< const wchar_t * > std::wcregex_token_iterator`
- `typedef sub_match< const wchar_t * > std::wcsub_match`
- `typedef basic_regex< wchar_t > std::wregex`
- `typedef match_results< wstring::const_iterator > std::wsmatch`
- `typedef regex_iterator< wstring::const_iterator > std::wsregex_iterator`
- `typedef regex_token_iterator< wstring::const_iterator > std::wsregex_token_iterator`
- `typedef sub_match< wstring::const_iterator > std::wssub_match`

Functions

- `template<typename _Bi_iter >`
`const sub_match< _Bi_iter > & std::__unmatched_sub ()`
- `template<typename _Biliter >`
`bool std::operator!= (const sub_match< _Biliter > &__lhs, const sub_match< _Biliter > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool std::operator!= (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool std::operator!= (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator!= (typename iterator_traits< _Bi_iter >::value_type const *__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator!= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const *__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator!= (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator!= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Bi_iter, class _Alloc >`
`bool std::operator!= (const match_results< _Bi_iter, _Alloc > &__m1, const match_results< _Bi_iter, _Alloc > &__m2)`
- `template<typename _Biliter >`
`bool std::operator< (const sub_match< _Biliter > &__lhs, const sub_match< _Biliter > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool std::operator< (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc >`
`bool std::operator< (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator< (typename iterator_traits< _Bi_iter >::value_type const *__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator< (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const *__rhs)`

- `template<typename _Bi_iter >`
`bool std::operator< (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator< (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Ch_type, typename _Ch_traits, typename _Bi_iter >`
`basic_ostream< _Ch_type, _Ch_traits > & std::operator<< (basic_ostream< _Ch_type, _Ch_traits > &__os, const sub_match< _Bi_iter > &__m)`
- `template<typename _Bilter >`
`bool std::operator<= (const sub_match< _Bilter > &__lhs, const sub_match< _Bilter > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool std::operator<= (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc >`
`bool std::operator<= (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator<= (typename iterator_traits< _Bi_iter >::value_type const *__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator<= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const *__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator<= (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator<= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Bilter >`
`bool std::operator== (const sub_match< _Bilter > &__lhs, const sub_match< _Bilter > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool std::operator== (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool std::operator== (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator== (typename iterator_traits< _Bi_iter >::value_type const *__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator== (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const *__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator== (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator== (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Bi_iter, typename _Alloc >`
`bool std::operator== (const match_results< _Bi_iter, _Alloc > &__m1, const match_results< _Bi_iter, _Alloc > &__m2)`

- `template<typename _Bilter >`
`bool std::operator> (const sub_match< _Bilter > &__lhs, const sub_match< _Bilter > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool std::operator> (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc >`
`bool std::operator> (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator> (typename iterator_traits< _Bi_iter >::value_type const *__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator> (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const *__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator> (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator> (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Bilter >`
`bool std::operator>= (const sub_match< _Bilter > &__lhs, const sub_match< _Bilter > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool std::operator>= (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc >`
`bool std::operator>= (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator>= (typename iterator_traits< _Bi_iter >::value_type const *__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator>= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const *__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator>= (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator>= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Ch_type, typename _Rx_traits >`
`void std::swap (basic_regex< _Ch_type, _Rx_traits > &__lhs, basic_regex< _Ch_type, _Rx_traits > &__rhs)`
- `template<typename _Bi_iter, typename _Alloc >`
`void std::swap (match_results< _Bi_iter, _Alloc > &__lhs, match_results< _Bi_iter, _Alloc > &__rhs)`

Matching, Searching, and Replacing

- `template<typename _Bi_iter, typename _Alloc, typename _Ch_type, typename _Rx_traits >`
`bool std::regex_match (_Bi_iter __s, _Bi_iter __e, match_results< _Bi_iter, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits >`
`bool std::regex_match (_Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)`

- `template<typename _Ch_type, typename _Alloc, typename _Rx_traits >`
`bool std::regex_match (const _Ch_type * __s, match_results< const _Ch_type *, _Alloc > & __m, const basic_regex< _Ch_type, _Rx_traits > & __re, regex_constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits >`
`bool std::regex_match (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > & __s, match_results< typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > & __m, const basic_regex< _Ch_type, _Rx_traits > & __re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Ch_type, class _Rx_traits >`
`bool std::regex_match (const _Ch_type * __s, const basic_regex< _Ch_type, _Rx_traits > & __re, regex_constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _Str_allocator, typename _Ch_type, typename _Rx_traits >`
`bool std::regex_match (const basic_string< _Ch_type, _Ch_traits, _Str_allocator > & __s, const basic_regex< _Ch_type, _Rx_traits > & __re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Bi_iter, typename _Alloc, typename _Ch_type, typename _Rx_traits >`
`bool std::regex_search (_Bi_iter __s, _Bi_iter __e, match_results< _Bi_iter, _Alloc > & __m, const basic_regex< _Ch_type, _Rx_traits > & __re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits >`
`bool std::regex_search (_Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits > & __re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Ch_type, class _Alloc, class _Rx_traits >`
`bool std::regex_search (const _Ch_type * __s, match_results< const _Ch_type *, _Alloc > & __m, const basic_regex< _Ch_type, _Rx_traits > & __e, regex_constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Ch_type, typename _Rx_traits >`
`bool std::regex_search (const _Ch_type * __s, const basic_regex< _Ch_type, _Rx_traits > & __e, regex_constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _String_allocator, typename _Ch_type, typename _Rx_traits >`
`bool std::regex_search (const basic_string< _Ch_type, _Ch_traits, _String_allocator > & __s, const basic_regex< _Ch_type, _Rx_traits > & __e, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits >`
`bool std::regex_search (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > & __s, match_results< typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > & __m, const basic_regex< _Ch_type, _Rx_traits > & __e, regex_constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Out_iter, typename _Bi_iter, typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa >`
`_Out_iter std::regex_replace (_Out_iter __out, _Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits > & __e, const basic_string< _Ch_type, _St, _Sa > & __fmt, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Out_iter, typename _Bi_iter, typename _Rx_traits, typename _Ch_type >`
`_Out_iter std::regex_replace (_Out_iter __out, _Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits > & __e, const _Ch_type * __fmt, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa, typename _Fst, typename _Fsa >`
`basic_string< _Ch_type, _St, _Sa > std::regex_replace (const basic_string< _Ch_type, _St, _Sa > & __s, const basic_regex< _Ch_type, _Rx_traits > & __e, const basic_string< _Ch_type, _Fst, _Fsa > & __fmt, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa >`
`basic_string< _Ch_type, _St, _Sa > std::regex_replace (const basic_string< _Ch_type, _St, _Sa > & __s, const basic_regex< _Ch_type, _Rx_traits > & __e, const _Ch_type * __fmt, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa >`
`basic_string< _Ch_type > std::regex_replace (const _Ch_type * __s, const basic_regex< _Ch_type, _Rx_traits > & __e, const basic_string< _Ch_type, _St, _Sa > & __fmt, regex_constants::match_flag_type __flags=regex_constants::match_default)`

- `template<typename _Rx_traits, typename _Ch_type >`
`basic_string< _Ch_type > std::regex_replace (const _Ch_type *__s, const basic_regex< _Ch_type, _Rx_traits`
`> &__e, const _Ch_type *__fmt, regex_constants::match_flag_type __flags=regex_constants::match_default)`

2.49.1 Detailed Description

A facility for performing regular expression pattern matching.

2.49.2 Typedef Documentation

2.49.2.1 `typedef regex_token_iterator<const char*> std::cregex_token_iterator`

Token iterator for C-style NULL-terminated strings.

Definition at line 2737 of file `regex.h`.

2.49.2.2 `typedef sub_match<const char*> std::csub_match`

Standard regex submatch over a C-style null-terminated string.

Definition at line 950 of file `regex.h`.

2.49.2.3 `typedef basic_regex<char> std::regex`

Standard regular expressions.

Definition at line 822 of file `regex.h`.

2.49.2.4 `typedef regex_token_iterator<string::const_iterator> std::sregex_token_iterator`

Token iterator for standard strings.

Definition at line 2740 of file `regex.h`.

2.49.2.5 `typedef sub_match<string::const_iterator> std::ssub_match`

Standard regex submatch over a standard string.

Definition at line 953 of file `regex.h`.

2.49.2.6 `typedef regex_token_iterator<const wchar_t*> std::wcregex_token_iterator`

Token iterator for C-style NULL-terminated wide strings.

Definition at line 2744 of file `regex.h`.

2.49.2.7 `typedef sub_match<const wchar_t*> std::wcsub_match`

Regex submatch over a C-style null-terminated wide string.

Definition at line 957 of file regex.h.

2.49.2.8 `typedef basic_regex<wchar_t> std::wregex`

Standard wide-character regular expressions.

Definition at line 826 of file regex.h.

2.49.2.9 `typedef regex_token_iterator<wstring::const_iterator> std::wsregex_token_iterator`

Token iterator for standard wide-character strings.

Definition at line 2747 of file regex.h.

2.49.2.10 `typedef sub_match<wstring::const_iterator> std::wssub_match`

Regex submatch over a standard wide string.

Definition at line 960 of file regex.h.

2.49.3 Function Documentation

2.49.3.1 `template<typename _Bilter> bool std::operator!=(const sub_match<_Bilter> & __lhs, const sub_match<_Bilter> & __rhs) [inline]`

Tests the inequivalence of two regular expression submatches.

Parameters

<code>__lhs</code>	First regular expression submatch.
<code>__rhs</code>	Second regular expression submatch.

Returns

true if `__lhs` is not equivalent to `__rhs`, false otherwise.

Definition at line 984 of file regex.h.

References `std::sub_match<_Bilter>::compare()`.

```
2.49.3.2 template<typename _Bi_iter , typename _Ch_traits , typename _Ch_alloc > bool std::operator!= ( const  
    __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > & __lhs, const sub_match< _Bi_iter > & __rhs )  
    [inline]
```

Tests the inequivalence of a string and a regular expression submatch.

Parameters

<code>__lhs</code>	A string.
<code>__rhs</code>	A regular expression submatch.

Returns

true if `__lhs` is not equivalent to `__rhs`, false otherwise.

Definition at line 1059 of file `regex.h`.

2.49.3.3 `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc> bool std::operator!=(const sub_match<_Bi_iter> & __lhs, const __sub_match_string<_Bi_iter, _Ch_traits, _Ch_alloc> & __rhs) [inline]`

Tests the inequivalence of a regular expression submatch and a string.

Parameters

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A string.

Returns

true if `__lhs` is not equivalent to `__rhs`, false otherwise.

Definition at line 1133 of file `regex.h`.

2.49.3.4 `template<typename _Bi_iter> bool std::operator!=(typename iterator_traits<_Bi_iter>::value_type const * __lhs, const sub_match<_Bi_iter> & __rhs) [inline]`

Tests the inequivalence of an iterator value and a regular expression submatch.

Parameters

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A string.

Returns

true if `__lhs` is not equivalent to `__rhs`, false otherwise.

Definition at line 1207 of file `regex.h`.

2.49.3.5 `template<typename _Bi_iter> bool std::operator!=(const sub_match<_Bi_iter> & __lhs, typename iterator_traits<_Bi_iter>::value_type const * __rhs) [inline]`

Tests the inequivalence of a regular expression submatch and a string.

Parameters

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A pointer to a string.

Returns

true if `__lhs` is not equivalent to `__rhs`, false otherwise.

Definition at line 1281 of file `regex.h`.

```
2.49.3.6 template<typename _Bi_iter > bool std::operator!=( typename iterator_traits< _Bi_iter >::value_type const & __lhs,
const sub_match< _Bi_iter > & __rhs ) [inline]
```

Tests the inequivalence of a string and a regular expression submatch.

Parameters

<code>__lhs</code>	A string.
<code>__rhs</code>	A regular expression submatch.

Returns

true if `__lhs` is not equivalent to `__rhs`, false otherwise.

Definition at line 1358 of file `regex.h`.

```
2.49.3.7 template<typename _Bi_iter > bool std::operator!=( const sub_match< _Bi_iter > & __lhs, typename iterator_traits<
_Bi_iter >::value_type const & __rhs ) [inline]
```

Tests the inequivalence of a regular expression submatch and a string.

Parameters

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A const string reference.

Returns

true if `__lhs` is not equivalent to `__rhs`, false otherwise.

Definition at line 1438 of file `regex.h`.

```
2.49.3.8 template<typename _Bi_iter, class _Alloc > bool std::operator!=( const match_results< _Bi_iter, _Alloc > & __m1,
const match_results< _Bi_iter, _Alloc > & __m2 ) [inline]
```

Compares two `match_results` for inequality.

Returns

true if the two objects do not refer to the same match, false otherwise.

Definition at line 1953 of file regex.h.

2.49.3.9 `template<typename _Bilter> bool std::operator< (const sub_match< _Bilter> &__lhs, const sub_match< _Bilter> &__rhs) [inline]`

Tests the ordering of two regular expression submatches.

Parameters

<code>__lhs</code>	First regular expression submatch.
<code>__rhs</code>	Second regular expression submatch.

Returns

true if `__lhs` precedes `__rhs`, false otherwise.

Definition at line 995 of file regex.h.

References `std::sub_match< _Bilter>::compare()`.

2.49.3.10 `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc> bool std::operator< (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc> &__lhs, const sub_match< _Bi_iter> &__rhs) [inline]`

Tests the ordering of a string and a regular expression submatch.

Parameters

<code>__lhs</code>	A string.
<code>__rhs</code>	A regular expression submatch.

Returns

true if `__lhs` precedes `__rhs`, false otherwise.

Definition at line 1071 of file regex.h.

References `std::sub_match< _Bilter>::compare()`.

2.49.3.11 `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc> bool std::operator< (const sub_match< _Bi_iter> &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc> &__rhs) [inline]`

Tests the ordering of a regular expression submatch and a string.

Parameters

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A string.

Returns

true if `__lhs` precedes `__rhs`, false otherwise.

Definition at line 1145 of file `regex.h`.

References `std::basic_string<_CharT, _Traits, _Alloc>::compare()`.

2.49.3.12 `template<typename _Bi_iter> bool std::operator< (typename iterator_traits<_Bi_iter>::value_type const * __lhs, const sub_match<_Bi_iter> & __rhs) [inline]`

Tests the ordering of a string and a regular expression submatch.

Parameters

<code>__lhs</code>	A string.
<code>__rhs</code>	A regular expression submatch.

Returns

true if `__lhs` precedes `__rhs`, false otherwise.

Definition at line 1219 of file `regex.h`.

References `std::sub_match<_Biter>::compare()`.

2.49.3.13 `template<typename _Bi_iter> bool std::operator< (const sub_match<_Bi_iter> & __lhs, typename iterator_traits<_Bi_iter>::value_type const * __rhs) [inline]`

Tests the ordering of a regular expression submatch and a string.

Parameters

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A string.

Returns

true if `__lhs` precedes `__rhs`, false otherwise.

Definition at line 1293 of file `regex.h`.

References `std::basic_string<_CharT, _Traits, _Alloc>::compare()`.

2.49.3.14 `template<typename _Bi_iter > bool std::operator< (typename iterator_traits< _Bi_iter >::value_type const & __lhs, const sub_match< _Bi_iter > & __rhs) [inline]`

Tests the ordering of a string and a regular expression submatch.

Parameters

<code>__lhs</code>	A string.
<code>__rhs</code>	A regular expression submatch.

Returns

true if `__lhs` precedes `__rhs`, false otherwise.

Definition at line 1370 of file `regex.h`.

References `std::sub_match< _Biter >::compare()`.

2.49.3.15 `template<typename _Bi_iter > bool std::operator< (const sub_match< _Bi_iter > & __lhs, typename iterator_traits< _Bi_iter >::value_type const & __rhs) [inline]`

Tests the ordering of a regular expression submatch and a string.

Parameters

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A const string reference.

Returns

true if `__lhs` precedes `__rhs`, false otherwise.

Definition at line 1450 of file `regex.h`.

References `std::sub_match< _Biter >::compare()`.

2.49.3.16 `template<typename _Ch_type , typename _Ch_traits , typename _Bi_iter > basic_ostream< _Ch_type, _Ch_traits>& std::operator<< (basic_ostream< _Ch_type, _Ch_traits > & __os, const sub_match< _Bi_iter > & __m) [inline]`

Inserts a matched string into an output stream.

Parameters

<code>__os</code>	The output stream.
<code>__m</code>	A submatch string.

Returns

the output stream with the submatch string inserted.

Definition at line 1504 of file regex.h.

References `std::sub_match<_Bilter>::str()`.

```
2.49.3.17  template<typename _Bilter > bool std::operator<= ( const sub_match<_Bilter > & __lhs, const sub_match<
    _Bilter > & __rhs )  [inline]
```

Tests the ordering of two regular expression submatches.

Parameters

<code>__lhs</code>	First regular expression submatch.
<code>__rhs</code>	Second regular expression submatch.

Returns

true if `__lhs` does not succeed `__rhs`, false otherwise.

Definition at line 1006 of file regex.h.

References `std::sub_match<_Bilter>::compare()`.

```
2.49.3.18  template<typename _Bi_iter , typename _Ch_traits , typename _Ch_alloc > bool std::operator<= ( const
    __sub_match_string<_Bi_iter, _Ch_traits, _Ch_alloc > & __lhs, const sub_match<_Bi_iter > & __rhs )
    [inline]
```

Tests the ordering of a string and a regular expression submatch.

Parameters

<code>__lhs</code>	A string.
<code>__rhs</code>	A regular expression submatch.

Returns

true if `__lhs` does not succeed `__rhs`, false otherwise.

Definition at line 1107 of file regex.h.

```
2.49.3.19  template<typename _Bi_iter , class _Ch_traits , class _Ch_alloc > bool std::operator<= ( const sub_match<_Bi_iter
    > & __lhs, const __sub_match_string<_Bi_iter, _Ch_traits, _Ch_alloc > & __rhs )  [inline]
```

Tests the ordering of a regular expression submatch and a string.

Parameters

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A string.

Returns

true if `__lhs` does not succeed `__rhs`, false otherwise.

Definition at line 1181 of file `regex.h`.

```
2.49.3.20  template<typename _Bi_iter > bool std::operator<= ( typename iterator_traits< _Bi_iter >::value_type const * __lhs,  
const sub_match< _Bi_iter > & __rhs ) [inline]
```

Tests the ordering of a string and a regular expression submatch.

Parameters

<code>__lhs</code>	A string.
<code>__rhs</code>	A regular expression submatch.

Returns

true if `__lhs` does not succeed `__rhs`, false otherwise.

Definition at line 1255 of file `regex.h`.

```
2.49.3.21  template<typename _Bi_iter > bool std::operator<= ( const sub_match< _Bi_iter > & __lhs, typename  
iterator_traits< _Bi_iter >::value_type const * __rhs ) [inline]
```

Tests the ordering of a regular expression submatch and a string.

Parameters

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A string.

Returns

true if `__lhs` does not succeed `__rhs`, false otherwise.

Definition at line 1329 of file `regex.h`.

```
2.49.3.22  template<typename _Bi_iter > bool std::operator<= ( typename iterator_traits< _Bi_iter >::value_type const & __lhs,  
const sub_match< _Bi_iter > & __rhs ) [inline]
```

Tests the ordering of a string and a regular expression submatch.

Parameters

<code>__lhs</code>	A string.
<code>__rhs</code>	A regular expression submatch.

Returns

true if `__lhs` does not succeed `__rhs`, false otherwise.

Definition at line 1409 of file `regex.h`.

2.49.3.23 `template<typename _Bi_iter > bool std::operator<= (const sub_match< _Bi_iter > & __lhs, typename iterator_traits< _Bi_iter >::value_type const & __rhs) [inline]`

Tests the ordering of a regular expression submatch and a string.

Parameters

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A const string reference.

Returns

true if `__lhs` does not succeed `__rhs`, false otherwise.

Definition at line 1489 of file `regex.h`.

2.49.3.24 `template<typename _Biter > bool std::operator== (const sub_match< _Biter > & __lhs, const sub_match< _Biter > & __rhs) [inline]`

Tests the equivalence of two regular expression submatches.

Parameters

<code>__lhs</code>	First regular expression submatch.
<code>__rhs</code>	Second regular expression submatch.

Returns

true if `__lhs` is equivalent to `__rhs`, false otherwise.

Definition at line 973 of file `regex.h`.

References `std::sub_match< _Biter >::compare()`.

2.49.3.25 `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc > bool std::operator==(const
__sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > & __lhs, const sub_match< _Bi_iter > & __rhs)
[inline]`

Tests the equivalence of a string and a regular expression submatch.

Parameters

<code>__lhs</code>	A string.
<code>__rhs</code>	A regular expression submatch.

Returns

true if `__lhs` is equivalent to `__rhs`, false otherwise.

Definition at line 1046 of file `regex.h`.

References `std::basic_string< _CharT, _Traits, _Alloc >::c_str()`, and `std::sub_match< _Bilter >::compare()`.

2.49.3.26 `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc > bool std::operator==(const sub_match<
_Bi_iter > & __lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > & __rhs) [inline]`

Tests the equivalence of a regular expression submatch and a string.

Parameters

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A string.

Returns

true if `__lhs` is equivalent to `__rhs`, false otherwise.

Definition at line 1120 of file `regex.h`.

References `std::basic_string< _CharT, _Traits, _Alloc >::c_str()`, and `std::sub_match< _Bilter >::compare()`.

2.49.3.27 `template<typename _Bi_iter > bool std::operator==(typename iterator_traits< _Bi_iter >::value_type const * __lhs,
const sub_match< _Bi_iter > & __rhs) [inline]`

Tests the equivalence of a C string and a regular expression submatch.

Parameters

<code>__lhs</code>	A C string.
<code>__rhs</code>	A regular expression submatch.

Returns

true if `__lhs` is equivalent to `__rhs`, false otherwise.

Definition at line 1194 of file `regex.h`.

References `std::sub_match<_Biter>::compare()`.

2.49.3.28 `template<typename _Bi_iter> bool std::operator==(const sub_match<_Bi_iter> & __lhs, typename iterator_traits<_Bi_iter>::value_type const * __rhs) [inline]`

Tests the equivalence of a regular expression submatch and a string.

Parameters

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A pointer to a string?

Returns

true if `__lhs` is equivalent to `__rhs`, false otherwise.

Definition at line 1268 of file `regex.h`.

References `std::sub_match<_Biter>::compare()`.

2.49.3.29 `template<typename _Bi_iter> bool std::operator==(typename iterator_traits<_Bi_iter>::value_type const & __lhs, const sub_match<_Bi_iter> & __rhs) [inline]`

Tests the equivalence of a string and a regular expression submatch.

Parameters

<code>__lhs</code>	A string.
<code>__rhs</code>	A regular expression submatch.

Returns

true if `__lhs` is equivalent to `__rhs`, false otherwise.

Definition at line 1342 of file `regex.h`.

References `std::sub_match<_Biter>::compare()`.

2.49.3.30 `template<typename _Bi_iter> bool std::operator==(const sub_match<_Bi_iter> & __lhs, typename iterator_traits<_Bi_iter>::value_type const & __rhs) [inline]`

Tests the equivalence of a regular expression submatch and a string.

Parameters

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A const string reference.

Returns

true if `__lhs` is equivalent to `__rhs`, false otherwise.

Definition at line 1422 of file `regex.h`.

References `std::sub_match<_Bilter>::compare()`.

2.49.3.31 `template<typename _Bi_iter, typename _Alloc> bool std::operator==(const match_results<_Bi_iter, _Alloc> & __m1, const match_results<_Bi_iter, _Alloc> & __m2) [inline]`

Compares two `match_results` for equality.

Returns

true if the two objects refer to the same match, false otherwise.

Definition at line 1929 of file `regex.h`.

References `std::match_results<_Bi_iter, _Alloc>::begin()`, `std::match_results<_Bi_iter, _Alloc>::empty()`, `std::match_results<_Bi_iter, _Alloc>::end()`, `std::equal()`, `std::match_results<_Bi_iter, _Alloc>::prefix()`, `std::match_results<_Bi_iter, _Alloc>::ready()`, `std::match_results<_Bi_iter, _Alloc>::size()`, and `std::match_results<_Bi_iter, _Alloc>::suffix()`.

2.49.3.32 `template<typename _Bilter> bool std::operator> (const sub_match<_Bilter> & __lhs, const sub_match<_Bilter> & __rhs) [inline]`

Tests the ordering of two regular expression submatches.

Parameters

<code>__lhs</code>	First regular expression submatch.
<code>__rhs</code>	Second regular expression submatch.

Returns

true if `__lhs` succeeds `__rhs`, false otherwise.

Definition at line 1028 of file `regex.h`.

References `std::sub_match<_Bilter>::compare()`.

2.49.3.33 `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc > bool std::operator> (const
__sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > & __lhs, const sub_match< _Bi_iter > & __rhs)
[inline]`

Tests the ordering of a string and a regular expression submatch.

Parameters

<code>__lhs</code>	A string.
<code>__rhs</code>	A regular expression submatch.

Returns

true if `__lhs` succeeds `__rhs`, false otherwise.

Definition at line 1083 of file `regex.h`.

2.49.3.34 `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc > bool std::operator> (const sub_match< _Bi_iter >
& __lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > & __rhs) [inline]`

Tests the ordering of a regular expression submatch and a string.

Parameters

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A string.

Returns

true if `__lhs` succeeds `__rhs`, false otherwise.

Definition at line 1157 of file `regex.h`.

2.49.3.35 `template<typename _Bi_iter > bool std::operator> (typename iterator_traits< _Bi_iter >::value_type const * __lhs,
const sub_match< _Bi_iter > & __rhs) [inline]`

Tests the ordering of a string and a regular expression submatch.

Parameters

<code>__lhs</code>	A string.
<code>__rhs</code>	A regular expression submatch.

Returns

true if `__lhs` succeeds `__rhs`, false otherwise.

Definition at line 1231 of file `regex.h`.

2.49.3.36 `template<typename _Bi_iter > bool std::operator> (const sub_match<_Bi_iter > & __lhs, typename iterator_traits<_Bi_iter >::value_type const * __rhs) [inline]`

Tests the ordering of a regular expression submatch and a string.

Parameters

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A string.

Returns

true if `__lhs` succeeds `__rhs`, false otherwise.

Definition at line 1305 of file `regex.h`.

2.49.3.37 `template<typename _Bi_iter > bool std::operator> (typename iterator_traits<_Bi_iter >::value_type const & __lhs, const sub_match<_Bi_iter > & __rhs) [inline]`

Tests the ordering of a string and a regular expression submatch.

Parameters

<code>__lhs</code>	A string.
<code>__rhs</code>	A regular expression submatch.

Returns

true if `__lhs` succeeds `__rhs`, false otherwise.

Definition at line 1385 of file `regex.h`.

2.49.3.38 `template<typename _Bi_iter > bool std::operator> (const sub_match<_Bi_iter > & __lhs, typename iterator_traits<_Bi_iter >::value_type const & __rhs) [inline]`

Tests the ordering of a regular expression submatch and a string.

Parameters

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A const string reference.

Returns

true if `__lhs` succeeds `__rhs`, false otherwise.

Definition at line 1465 of file `regex.h`.

```
2.49.3.39 template<typename _Bilter > bool std::operator>= ( const sub_match< _Bilter > & __lhs, const sub_match<
    _Bilter > & __rhs ) [inline]
```

Tests the ordering of two regular expression submatches.

Parameters

<code>__lhs</code>	First regular expression submatch.
<code>__rhs</code>	Second regular expression submatch.

Returns

true if `__lhs` does not precede `__rhs`, false otherwise.

Definition at line 1017 of file `regex.h`.

References `std::sub_match< _Bilter >::compare()`.

```
2.49.3.40 template<typename _Bi_iter , typename _Ch_traits , typename _Ch_alloc > bool std::operator>= ( const
    __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > & __lhs, const sub_match< _Bi_iter > & __rhs )
    [inline]
```

Tests the ordering of a string and a regular expression submatch.

Parameters

<code>__lhs</code>	A string.
<code>__rhs</code>	A regular expression submatch.

Returns

true if `__lhs` does not precede `__rhs`, false otherwise.

Definition at line 1095 of file `regex.h`.

```
2.49.3.41 template<typename _Bi_iter , class _Ch_traits , class _Ch_alloc > bool std::operator>= ( const sub_match< _Bi_iter
    > & __lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > & __rhs ) [inline]
```

Tests the ordering of a regular expression submatch and a string.

Parameters

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A string.

Returns

true if `__lhs` does not precede `__rhs`, false otherwise.

Definition at line 1169 of file `regex.h`.

```
2.49.3.42  template<typename _Bi_iter > bool std::operator>= ( typename iterator_traits< _Bi_iter >::value_type const * __lhs,
               const sub_match< _Bi_iter > & __rhs )  [inline]
```

Tests the ordering of a string and a regular expression submatch.

Parameters

<code>__lhs</code>	A string.
<code>__rhs</code>	A regular expression submatch.

Returns

true if `__lhs` does not precede `__rhs`, false otherwise.

Definition at line 1243 of file `regex.h`.

```
2.49.3.43  template<typename _Bi_iter > bool std::operator>= ( const sub_match< _Bi_iter > & __lhs, typename
               iterator_traits< _Bi_iter >::value_type const * __rhs )  [inline]
```

Tests the ordering of a regular expression submatch and a string.

Parameters

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A string.

Returns

true if `__lhs` does not precede `__rhs`, false otherwise.

Definition at line 1317 of file `regex.h`.

```
2.49.3.44  template<typename _Bi_iter > bool std::operator>= ( typename iterator_traits< _Bi_iter >::value_type const & __lhs,
               const sub_match< _Bi_iter > & __rhs )  [inline]
```

Tests the ordering of a string and a regular expression submatch.

Parameters

<code>__lhs</code>	A string.
<code>__rhs</code>	A regular expression submatch.

Returns

true if `__lhs` does not precede `__rhs`, false otherwise.

Definition at line 1397 of file `regex.h`.

```
2.49.3.45 template<typename _Bi_iter > bool std::operator>= ( const sub_match<_Bi_iter > & __lhs, typename
iterator_traits<_Bi_iter >::value_type const & __rhs ) [inline]
```

Tests the ordering of a regular expression submatch and a string.

Parameters

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A const string reference.

Returns

true if `__lhs` does not precede `__rhs`, false otherwise.

Definition at line 1477 of file `regex.h`.

```
2.49.3.46 template<typename _Bi_iter, typename _Alloc, typename _Ch_type, typename _Rx_traits > bool std::regex_match (
_Bi_iter __s, _Bi_iter __e, match_results<_Bi_iter, _Alloc > & __m, const basic_regex<_Ch_type, _Rx_traits > &
__re, regex_constants::match_flag_type __flags = regex_constants::match_default ) [inline]
```

Determines if there is a match between the regular expression `e` and all of the character sequence `[first, last)`.

Parameters

<code>__s</code>	Start of the character sequence to match.
<code>__e</code>	One-past-the-end of the character sequence to match.
<code>__m</code>	The match results.
<code>__re</code>	The regular expression.
<code>__flags</code>	Controls how the regular expression is matched.

Return values

<code>true</code>	A match exists.
<code>false</code>	Otherwise.

Exceptions

<i>an</i>	exception of type <code>regex_error</code> .
-----------	--

Definition at line 1995 of file `regex.h`.

Referenced by `std::regex_match()`.

2.49.3.47 `template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits> bool std::regex_match (_Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits> & __re, regex_constants::match_flag_type __flags = regex_constants::match_default) [inline]`

Indicates if there is a match between the regular expression `e` and all of the character sequence `[first, last)`.

Parameters

<code>__first</code>	Beginning of the character sequence to match.
<code>__last</code>	One-past-the-end of the character sequence to match.
<code>__re</code>	The regular expression.
<code>__flags</code>	Controls how the regular expression is matched.

Return values

<code>true</code>	A match exists.
<code>false</code>	Otherwise.

Exceptions

<i>an</i>	exception of type <code>regex_error</code> .
-----------	--

Definition at line 2023 of file `regex.h`.

References `std::regex_match()`.

2.49.3.48 `template<typename _Ch_type, typename _Alloc, typename _Rx_traits> bool std::regex_match (const _Ch_type * __s, match_results< const _Ch_type *, _Alloc> & __m, const basic_regex< _Ch_type, _Rx_traits> & __re, regex_constants::match_flag_type __f = regex_constants::match_default) [inline]`

Determines if there is a match between the regular expression `e` and a C-style null-terminated string.

Parameters

<code>__s</code>	The C-style null-terminated string to match.
<code>__m</code>	The match results.

Parameters

<code>_re</code>	The regular expression.
<code>_f</code>	Controls how the regular expression is matched.

Return values

<code>true</code>	A match exists.
<code>false</code>	Otherwise.

Exceptions

<code>an</code>	exception of type <code>regex_error</code> .
-----------------	--

Definition at line 2048 of file `regex.h`.

References `std::regex_match()`.

```
2.49.3.49 template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits >
bool std::regex_match ( const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > & __s, match_results< typename
basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > & __m, const basic_regex< _Ch_type,
_Rx_traits > & __re, regex_constants::match_flag_type __flags = regex_constants::match_default )
[inline]
```

Determines if there is a match between the regular expression `e` and a string.

Parameters

<code>__s</code>	The string to match.
<code>__m</code>	The match results.
<code>__re</code>	The regular expression.
<code>__flags</code>	Controls how the regular expression is matched.

Return values

<code>true</code>	A match exists.
<code>false</code>	Otherwise.

Exceptions

<code>an</code>	exception of type <code>regex_error</code> .
-----------------	--

Definition at line 2072 of file `regex.h`.

References `std::basic_string< _CharT, _Traits, _Alloc >::begin()`, `std::basic_string< _CharT, _Traits, _Alloc >::end()`, and `std::regex_match()`.

2.49.3.50 `template<typename _Ch_type, class _Rx_traits > bool std::regex_match (const _Ch_type * __s, const basic_regex< _Ch_type, _Rx_traits > & __re, regex_constants::match_flag_type __f = regex_constants::match_default) [inline]`

Indicates if there is a match between the regular expression `e` and a C-style null-terminated string.

Parameters

<code>__s</code>	The C-style null-terminated string to match.
<code>__re</code>	The regular expression.
<code>__f</code>	Controls how the regular expression is matched.

Return values

<code>true</code>	A match exists.
<code>false</code>	Otherwise.

Exceptions

<code>an</code>	exception of type <code>regex_error</code> .
-----------------	--

Definition at line 2095 of file `regex.h`.

References `std::regex_match()`.

2.49.3.51 `template<typename _Ch_traits, typename _Str_allocator, typename _Ch_type, typename _Rx_traits > bool std::regex_match (const basic_string< _Ch_type, _Ch_traits, _Str_allocator > & __s, const basic_regex< _Ch_type, _Rx_traits > & __re, regex_constants::match_flag_type __flags = regex_constants::match_default) [inline]`

Indicates if there is a match between the regular expression `e` and a string.

Parameters

<code>__s</code>	[IN] The string to match.
<code>__re</code>	[IN] The regular expression.
<code>__flags</code>	[IN] Controls how the regular expression is matched.

Return values

<code>true</code>	A match exists.
-------------------	-----------------

Return values

<i>false</i>	Otherwise.
--------------	------------

Exceptions

<i>an</i>	exception of type <code>regex_error</code> .
-----------	--

Definition at line 2117 of file `regex.h`.

References `std::basic_string<_CharT, _Traits, _Alloc>::begin()`, `std::basic_string<_CharT, _Traits, _Alloc>::end()`, and `std::regex_match()`.

2.49.3.52 `template<typename _Out_iter, typename _Bi_iter, typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa> _Out_iter std::regex_replace (_Out_iter __out, _Bi_iter __first, _Bi_iter __last, const basic_regex<_Ch_type, _Rx_traits> & __e, const basic_string<_Ch_type, _St, _Sa> & __fmt, regex_constants::match_flag_type __flags = regex_constants::match_default) [inline]`

Search for a regular expression within a range for multiple times, and replace the matched parts through filling a format string.

Parameters

<code>__out</code>	[OUT] The output iterator.
<code>__first</code>	[IN] The start of the string to search.
<code>__last</code>	[IN] One-past-the-end of the string to search.
<code>__e</code>	[IN] The regular expression to search for.
<code>__fmt</code>	[IN] The format string.
<code>__flags</code>	[IN] Search and replace policy flags.

Returns

`__out`

Exceptions

<i>an</i>	exception of type <code>regex_error</code> .
-----------	--

Definition at line 2274 of file `regex.h`.

References `std::basic_string<_CharT, _Traits, _Alloc>::c_str()`, and `std::regex_constants::match_default`.

Referenced by `std::regex_replace()`.

2.49.3.53 `template<typename _Out_iter , typename _Bi_iter , typename _Rx_traits , typename _Ch_type > _Out_iter
std::regex_replace (_Out_iter __out, _Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits > &
__e, const _Ch_type * __fmt, regex_constants::match_flag_type __flags = regex_constants::match_default
)`

Search for a regular expression within a range for multiple times, and replace the matched parts through filling a format C-string.

Parameters

<code>__out</code>	[OUT] The output iterator.
<code>__first</code>	[IN] The start of the string to search.
<code>__last</code>	[IN] One-past-the-end of the string to search.
<code>__e</code>	[IN] The regular expression to search for.
<code>__fmt</code>	[IN] The format C-string.
<code>__flags</code>	[IN] Search and replace policy flags.

Returns

`__out`

Exceptions

<code>an</code>	exception of type <code>regex_error</code> .
-----------------	--

2.49.3.54 `template<typename _Rx_traits , typename _Ch_type , typename _St , typename _Sa , typename _Fst , typename _Fsa
> basic_string<_Ch_type, _St, _Sa> std::regex_replace (const basic_string< _Ch_type, _St, _Sa > & __s,
const basic_regex< _Ch_type, _Rx_traits > & __e, const basic_string< _Ch_type, _Fst, _Fsa > & __fmt,
regex_constants::match_flag_type __flags = regex_constants::match_default) [inline]`

Search for a regular expression within a string for multiple times, and replace the matched parts through filling a format string.

Parameters

<code>__s</code>	[IN] The string to search and replace.
<code>__e</code>	[IN] The regular expression to search for.
<code>__fmt</code>	[IN] The format string.
<code>__flags</code>	[IN] Search and replace policy flags.

Returns

The string after replacing.

Exceptions

<i>an</i>	exception of type <code>regex_error</code> .
-----------	--

Definition at line 2319 of file `regex.h`.

References `std::back_inserter()`, `std::basic_string<_CharT, _Traits, _Alloc >::begin()`, `std::basic_string<_CharT, _Traits, _Alloc >::end()`, and `std::regex_replace()`.

2.49.3.55 `template<typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa > basic_string<_Ch_type, _St, _Sa> std::regex_replace (const basic_string<_Ch_type, _St, _Sa > & __s, const basic_regex<_Ch_type, _Rx_traits > & __e, const _Ch_type * __fmt, regex_constants::match_flag_type __flags = regex_constants::match_default) [inline]`

Search for a regular expression within a string for multiple times, and replace the matched parts through filling a format C-string.

Parameters

<code>__s</code>	[IN] The string to search and replace.
<code>__e</code>	[IN] The regular expression to search for.
<code>__fmt</code>	[IN] The format C-string.
<code>__flags</code>	[IN] Search and replace policy flags.

Returns

The string after replacing.

Exceptions

<i>an</i>	exception of type <code>regex_error</code> .
-----------	--

Definition at line 2345 of file `regex.h`.

References `std::back_inserter()`, `std::basic_string<_CharT, _Traits, _Alloc >::begin()`, `std::basic_string<_CharT, _Traits, _Alloc >::end()`, and `std::regex_replace()`.

2.49.3.56 `template<typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa > basic_string<_Ch_type> std::regex_replace (const _Ch_type * __s, const basic_regex<_Ch_type, _Rx_traits > & __e, const basic_string<_Ch_type, _St, _Sa > & __fmt, regex_constants::match_flag_type __flags = regex_constants::match_default) [inline]`

Search for a regular expression within a C-string for multiple times, and replace the matched parts through filling a format string.

Parameters

<code>__s</code>	[IN] The C-string to search and replace.
<code>__e</code>	[IN] The regular expression to search for.
<code>__fmt</code>	[IN] The format string.
<code>__flags</code>	[IN] Search and replace policy flags.

Returns

The string after replacing.

Exceptions

<i>an</i>	exception of type <code>regex_error</code> .
-----------	--

Definition at line 2371 of file `regex.h`.

References `std::back_inserter()`, and `std::regex_replace()`.

```
2.49.3.57 template<typename _Rx_traits, typename _Ch_type > basic_string<_Ch_type> std::regex_replace
( const _Ch_type * __s, const basic_regex<_Ch_type, _Rx_traits > & __e, const _Ch_type * __fmt,
  regex_constants::match_flag_type __flags = regex_constants::match_default ) [inline]
```

Search for a regular expression within a C-string for multiple times, and replace the matched parts through filling a format C-string.

Parameters

<code>__s</code>	[IN] The C-string to search and replace.
<code>__e</code>	[IN] The regular expression to search for.
<code>__fmt</code>	[IN] The format C-string.
<code>__flags</code>	[IN] Search and replace policy flags.

Returns

The string after replacing.

Exceptions

<i>an</i>	exception of type <code>regex_error</code> .
-----------	--

Definition at line 2397 of file `regex.h`.

References `std::back_inserter()`, and `std::regex_replace()`.

2.49.3.58 `template<typename _Bi_iter, typename _Alloc, typename _Ch_type, typename _Rx_traits> bool std::regex_search (_Bi_iter __s, _Bi_iter __e, match_results<_Bi_iter, _Alloc> & __m, const basic_regex<_Ch_type, _Rx_traits> & __re, regex_constants::match_flag_type __flags = regex_constants::match_default) [inline]`

Searches for a regular expression within a range.

Parameters

<code>__s</code>	[IN] The start of the string to search.
<code>__e</code>	[IN] One-past-the-end of the string to search.
<code>__m</code>	[OUT] The match results.
<code>__re</code>	[IN] The regular expression to search for.
<code>__flags</code>	[IN] Search policy flags.

Return values

<code>true</code>	A match was found within the string.
<code>false</code>	No match was found within the string, the content of <code>m</code> is undefined.

Exceptions

<code>an</code>	exception of type <code>regex_error</code> .
-----------------	--

Definition at line 2140 of file `regex.h`.

Referenced by `std::regex_iterator<_Bi_iter, _Ch_type, _Rx_traits>::regex_iterator()`, and `std::regex_search()`.

2.49.3.59 `template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits> bool std::regex_search (_Bi_iter __first, _Bi_iter __last, const basic_regex<_Ch_type, _Rx_traits> & __re, regex_constants::match_flag_type __flags = regex_constants::match_default) [inline]`

Searches for a regular expression within a range.

Parameters

<code>__first</code>	[IN] The start of the string to search.
<code>__last</code>	[IN] One-past-the-end of the string to search.
<code>__re</code>	[IN] The regular expression to search for.
<code>__flags</code>	[IN] Search policy flags.

Return values

<code>true</code>	A match was found within the string.
<code>false</code>	No match was found within the string.

Exceptions

<i>an</i>	exception of type <code>regex_error</code> .
-----------	--

Definition at line 2164 of file `regex.h`.

References `std::regex_search()`.

2.49.3.60 `template<typename _Ch_type, class _Alloc, class _Rx_traits> bool std::regex_search (const _Ch_type * __s, match_results< const _Ch_type *, _Alloc > & __m, const basic_regex< _Ch_type, _Rx_traits > & __e, regex_constants::match_flag_type __f=regex_constants::match_default) [inline]`

Searches for a regular expression within a C-string.

Parameters

<code>__s</code>	[IN] A C-string to search for the regex.
<code>__m</code>	[OUT] The set of regex matches.
<code>__e</code>	[IN] The regex to search for in <code>s</code> .
<code>__f</code>	[IN] The search flags.

Return values

<code>true</code>	A match was found within the string.
<code>false</code>	No match was found within the string, the content of <code>m</code> is undefined.

Exceptions

<i>an</i>	exception of type <code>regex_error</code> .
-----------	--

Definition at line 2187 of file `regex.h`.

References `std::regex_search()`.

2.49.3.61 `template<typename _Ch_type, typename _Rx_traits> bool std::regex_search (const _Ch_type * __s, const basic_regex< _Ch_type, _Rx_traits > & __e, regex_constants::match_flag_type __f=regex_constants::match_default) [inline]`

Searches for a regular expression within a C-string.

Parameters

<code>_↔ _s</code>	[IN] The C-string to search.
<code>_↔ _e</code>	[IN] The regular expression to search for.
<code>_↔ _f</code>	[IN] Search policy flags.

Return values

<code>true</code>	A match was found within the string.
<code>false</code>	No match was found within the string.

Exceptions

<code>an</code>	exception of type <code>regex_error</code> .
-----------------	--

Definition at line 2206 of file `regex.h`.

References `std::regex_search()`.

```
2.49.3.62 template<typename _Ch_traits, typename _String_allocator, typename _Ch_type, typename _Rx_traits > bool
std::regex_search ( const basic_string< _Ch_type, _Ch_traits, _String_allocator > & __s, const basic_regex<
_Ch_type, _Rx_traits > & __e, regex_constants::match_flag_type __flags = regex_constants::match_default
) [inline]
```

Searches for a regular expression within a string.

Parameters

<code>__s</code>	[IN] The string to search.
<code>__e</code>	[IN] The regular expression to search for.
<code>__flags</code>	[IN] Search policy flags.

Return values

<code>true</code>	A match was found within the string.
<code>false</code>	No match was found within the string.

Exceptions

<code>an</code>	exception of type <code>regex_error</code> .
-----------------	--

Definition at line 2225 of file `regex.h`.

References `std::regex_search()`.

```
2.49.3.63 template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits >
bool std::regex_search ( const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > & __s, match_results<
typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > & __m, const basic_regex<
_Ch_type, _Rx_traits > & __e, regex_constants::match_flag_type __f=regex_constants::match_default )
[inline]
```

Searches for a regular expression within a string.

Parameters

<code>__s</code>	[IN] A C++ string to search for the regex.
<code>__m</code>	[OUT] The set of regex matches.
<code>__e</code>	[IN] The regex to search for in <code>s</code> .
<code>__f</code>	[IN] The search flags.

Return values

<code>true</code>	A match was found within the string.
<code>false</code>	No match was found within the string, the content of <code>m</code> is undefined.

Exceptions

<code>an</code>	exception of type <code>regex_error</code> .
-----------------	--

Definition at line 2248 of file `regex.h`.

References `std::basic_string< _CharT, _Traits, _Alloc >::begin()`, `std::basic_string< _CharT, _Traits, _Alloc >::end()`, and `std::regex_search()`.

```
2.49.3.64 template<typename _Ch_type, typename _Rx_traits > void std::swap ( basic_regex< _Ch_type, _Rx_traits > & __lhs,
basic_regex< _Ch_type, _Rx_traits > & __rhs ) [inline]
```

Swaps the contents of two regular expression objects.

Parameters

<code>__lhs</code>	First regular expression.
<code>__rhs</code>	Second regular expression.

Definition at line 838 of file `regex.h`.

References `std::basic_regex< _Ch_type, _Rx_traits >::swap()`.

2.49.3.65 `template<typename _Bi_iter, typename _Alloc> void std::swap (match_results< _Bi_iter, _Alloc> & __lhs,
match_results< _Bi_iter, _Alloc> & __rhs) [inline]`

Swaps two match results.

Parameters

<code>__lhs</code>	A match result.
<code>__rhs</code>	A match result.

The contents of the two `match_results` objects are swapped.

Definition at line 1967 of file `regex.h`.

References `std::match_results< _Bi_iter, _Alloc>::swap()`.

Referenced by `std::match_results< _Bi_iter>::swap()`.

2.50 SGI

Collaboration diagram for SGI:



Because libstdc++ based its implementation of the STL subsections of the library on the SGI 3.3 implementation, we inherited their extensions as well.

They are additionally documented in the [online documentation](#), a copy of which is also shipped with the library source code (in `.../docs/html/documentation.html`). You can also read the documentation [on SGI's site](#), which is still running even though the code is not maintained.

NB that the following notes are pulled from various comments all over the place, so they may seem stilted.

2.51 Sequences

Collaboration diagram for Sequences:



Classes

- class `std::basic_string<_CharT, _Traits, _Alloc>`
- class `std::deque<_Tp, _Alloc>`
- class `std::forward_list<_Tp, _Alloc>`
- class `std::list<_Tp, _Alloc>`
- class `std::priority_queue<_Tp, _Sequence, _Compare>`
- class `std::queue<_Tp, _Sequence>`
- class `std::stack<_Tp, _Sequence>`
- class `std::vector<_Tp, _Alloc>`
- class `std::vector<bool, _Alloc>`

2.51.1 Detailed Description

Sequences arrange a collection of objects into a strictly linear order.

The differences between sequences are usually due to one or both of the following:

- memory management
- algorithmic complexity

As an example of the first case, `vector` is required to use a contiguous memory layout, while other sequences such as `deque` are not.

The prime reason for choosing one sequence over another should be based on the second category of differences, algorithmic complexity. For example, if you need to perform many inserts and removals from the middle of a sequence, `list` would be ideal. But if you need to perform constant-time access to random elements of the sequence, then `list` should not be used.

All sequences must meet certain requirements, summarized in [tables](#).

2.52 Set Operation

Collaboration diagram for Set Operation:



Functions

- `template<typename _InputIterator1, typename _InputIterator2 >`
`bool std::includes (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Compare >`
`bool std::includes (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`
`_OutputIterator std::set_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`
`_OutputIterator std::set_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`
`_OutputIterator std::set_intersection (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`
`_OutputIterator std::set_intersection (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`
`_OutputIterator std::set_symmetric_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`
`_OutputIterator std::set_symmetric_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`
`_OutputIterator std::set_union (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`
`_OutputIterator std::set_union (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`

2.52.1 Detailed Description

These algorithms are common set operations performed on sequences that are already sorted. The number of comparisons will be linear.

2.52.2 Function Documentation

2.52.2.1 `template<typename _InputIterator1, typename _InputIterator2 > bool std::includes (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2) [inline]`

Determines whether all elements of a sequence exists in a range.

Parameters

<code>__first1</code>	Start of search range.
<code>__last1</code>	End of search range.
<code>__first2</code>	Start of sequence
<code>__last2</code>	End of sequence.

Returns

True if each element in `[__first2,__last2)` is contained in order within `[__first1,__last1)`. False otherwise.

This operation expects both `[__first1,__last1)` and `[__first2,__last2)` to be sorted. Searches for the presence of each element in `[__first2,__last2)` within `[__first1,__last1)`. The iterators over each range only move forward, so this is a linear algorithm. If an element in `[__first2,__last2)` is not found before the search iterator reaches `__last2`, false is returned.

Definition at line 2833 of file `stl_algo.h`.

2.52.2.2 `template<typename _InputIterator1, typename _InputIterator2, typename _Compare > bool std::includes (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _Compare __comp) [inline]`

Determines whether all elements of a sequence exists in a range using comparison.

Parameters

<code>__first1</code>	Start of search range.
<code>__last1</code>	End of search range.
<code>__first2</code>	Start of sequence
<code>__last2</code>	End of sequence.
<code>__comp</code>	Comparison function to use.

Returns

True if each element in `[__first2,__last2)` is contained in order within `[__first1,__last1)` according to `comp`. False otherwise.

This operation expects both `[__first1,__last1)` and `[__first2,__last2)` to be sorted. Searches for the presence of each element in `[__first2,__last2)` within `[__first1,__last1)`, using `comp` to decide. The iterators over each range only move forward, so this is a linear algorithm. If an element in `[__first2,__last2)` is not found before the search iterator reaches `__last2`, false is returned.

Definition at line 2876 of file `stl_algo.h`.

References `std::__iterator_category()`, `std::__reverse()`, and `std::iter_swap()`.

```
2.52.2.3 template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator > _OutputIterator
std::set_difference ( _InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2,
    _OutputIterator __result ) [inline]
```

Return the difference of two sorted ranges.

Parameters

<code>__first1</code>	Start of first range.
<code>__last1</code>	End of first range.
<code>__first2</code>	Start of second range.
<code>__last2</code>	End of second range.

Returns

End of the output range.

This operation iterates over both ranges, copying elements present in the first range but not the second in order to the output range. Iterators increment for each range. When the current element of the first range is less than the second, that element is copied and the iterator advances. If the current element of the second range is less, the iterator advances, but no element is copied. If an element is contained in both ranges, no elements are copied and both ranges advance. The output range may not overlap either input range.

Definition at line 5208 of file `stl_algo.h`.

```
2.52.2.4 template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare
    > _OutputIterator std::set_difference ( _InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2,
    _InputIterator2 __last2, _OutputIterator __result, _Compare __comp ) [inline]
```

Return the difference of two sorted ranges using comparison functor.

Parameters

<code>__first1</code>	Start of first range.
<code>__last1</code>	End of first range.
<code>__first2</code>	Start of second range.
<code>__last2</code>	End of second range.
<code>__comp</code>	The comparison functor.

Returns

End of the output range.

This operation iterates over both ranges, copying elements present in the first range but not the second in order to the output range. Iterators increment for each range. When the current element of the first range is less than the second according to `__comp`, that element is copied and the iterator advances. If the current element of the second range is less, no element is copied and the iterator advances. If an element is contained in both ranges according to `__comp`, no elements are copied and both ranges advance. The output range may not overlap either input range.

Definition at line 5256 of file `stl_algo.h`.

2.52.2.5 `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator > _OutputIterator
std::set_intersection (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2,
_OutputIterator __result) [inline]`

Return the intersection of two sorted ranges.

Parameters

<code>__first1</code>	Start of first range.
<code>__last1</code>	End of first range.
<code>__first2</code>	Start of second range.
<code>__last2</code>	End of second range.

Returns

End of the output range.

This operation iterates over both ranges, copying elements present in both ranges in order to the output range. Iterators increment for each range. When the current element of one range is less than the other, that iterator advances. If an element is contained in both ranges, the element from the first range is copied and both ranges advance. The output range may not overlap either input range.

Definition at line 5092 of file `stl_algo.h`.

2.52.2.6 `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare
> _OutputIterator std::set_intersection (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2,
_InputIterator2 __last2, _OutputIterator __result, _Compare __comp) [inline]`

Return the intersection of two sorted ranges using comparison functor.

Parameters

<code>__first1</code>	Start of first range.
<code>__last1</code>	End of first range.
<code>__first2</code>	Start of second range.
<code>__last2</code>	End of second range.
<code>__comp</code>	The comparison functor.

Returns

End of the output range.

This operation iterates over both ranges, copying elements present in both ranges in order to the output range. Iterators increment for each range. When the current element of one range is less than the other according to `__comp`, that iterator advances. If an element is contained in both ranges according to `__comp`, the element from the first range is copied and both ranges advance. The output range may not overlap either input range.

Definition at line 5138 of file `stl_algo.h`.

```
2.52.2.7 template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator > _OutputIterator
std::set_symmetric_difference ( _InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2
__last2, _OutputIterator __result ) [inline]
```

Return the symmetric difference of two sorted ranges.

Parameters

<code>__first1</code>	Start of first range.
<code>__last1</code>	End of first range.
<code>__first2</code>	Start of second range.
<code>__last2</code>	End of second range.

Returns

End of the output range.

This operation iterates over both ranges, copying elements present in one range but not the other in order to the output range. Iterators increment for each range. When the current element of one range is less than the other, that element is copied and the iterator advances. If an element is contained in both ranges, no elements are copied and both ranges advance. The output range may not overlap either input range.

Definition at line 5332 of file `stl_algo.h`.

```
2.52.2.8 template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >
_OutputIterator std::set_symmetric_difference ( _InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2,
_InputIterator2 __last2, _OutputIterator __result, _Compare __comp ) [inline]
```

Return the symmetric difference of two sorted ranges using comparison functor.

Parameters

<code>__first1</code>	Start of first range.
<code>__last1</code>	End of first range.
<code>__first2</code>	Start of second range.
<code>__last2</code>	End of second range.
<code>__comp</code>	The comparison functor.

Returns

End of the output range.

This operation iterates over both ranges, copying elements present in one range but not the other in order to the output range. Iterators increment for each range. When the current element of one range is less than the other according to `comp`, that element is copied and the iterator advances. If an element is contained in both ranges according to `__comp`, no elements are copied and both ranges advance. The output range may not overlap either input range.

Definition at line 5380 of file `stl_algo.h`.

```
2.52.2.9 template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator > _OutputIterator std::set_union
( _InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result
) [inline]
```

Return the union of two sorted ranges.

Parameters

<code>__first1</code>	Start of first range.
<code>__last1</code>	End of first range.
<code>__first2</code>	Start of second range.
<code>__last2</code>	End of second range.

Returns

End of the output range.

This operation iterates over both ranges, copying elements present in each range in order to the output range. Iterators increment for each range. When the current element of one range is less than the other, that element is copied and the iterator advanced. If an element is contained in both ranges, the element from the first range is copied and both ranges advance. The output range may not overlap either input range.

Definition at line 4977 of file `stl_algo.h`.

```
2.52.2.10 template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >
_OutputIterator std::set_union ( _InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2
__last2, _OutputIterator __result, _Compare __comp ) [inline]
```

Return the union of two sorted ranges using a comparison functor.

Parameters

<code>__first1</code>	Start of first range.
<code>__last1</code>	End of first range.
<code>__first2</code>	Start of second range.
<code>__last2</code>	End of second range.
<code>__comp</code>	The comparison functor.

Returns

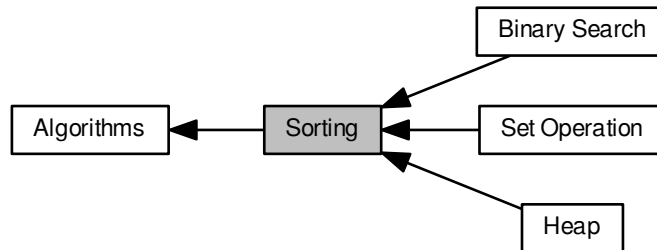
End of the output range.

This operation iterates over both ranges, copying elements present in each range in order to the output range. Iterators increment for each range. When the current element of one range is less than the other according to `__comp`, that element is copied and the iterator advanced. If an equivalent element according to `__comp` is contained in both ranges, the element from the first range is copied and both ranges advance. The output range may not overlap either input range.

Definition at line 5024 of file `stl_algo.h`.

2.53 Sorting

Collaboration diagram for Sorting:



Modules

- [Binary Search](#)
- [Heap](#)
- [Set Operation](#)

Functions

- `template<typename _BidirectionalIterator >`
`void std::inplace_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _Compare >`
`void std::inplace_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator >`
`bool std::is_sorted (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >`
`bool std::is_sorted (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator >`
`_ForwardIterator std::is_sorted_until (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >`
`_ForwardIterator std::is_sorted_until (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _II1, typename _II2 >`
`bool std::lexicographical_compare (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2)`
- `template<typename _II1, typename _II2, typename _Compare >`
`bool std::lexicographical_compare (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2, _Compare __comp)`
- `template<typename _Tp >`
`const _Tp & std::max (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp, typename _Compare >`
`const _Tp & std::max (const _Tp &__a, const _Tp &__b, _Compare __comp)`

- `template<typename _ForwardIterator >`
`_ForwardIterator std::max_element (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >`
`_ForwardIterator std::max_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`
`_OutputIterator std::merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`
`_OutputIterator std::merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _Tp >`
`const _Tp & std::min (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp, typename _Compare >`
`const _Tp & std::min (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _ForwardIterator >`
`_ForwardIterator std::min_element (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >`
`_ForwardIterator std::min_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _Tp >`
`pair< const _Tp &, const _Tp & > std::minmax (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp, typename _Compare >`
`pair< const _Tp &, const _Tp & > std::minmax (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _ForwardIterator >`
`pair< _ForwardIterator, _ForwardIterator > std::minmax_element (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >`
`pair< _ForwardIterator, _ForwardIterator > std::minmax_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _BidirectionalIterator >`
`bool std::next_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _Compare >`
`bool std::next_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`void std::nth_element (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void std::nth_element (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`void std::partial_sort (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void std::partial_sort (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _InputIterator, typename _RandomAccessIterator >`
`_RandomAccessIterator std::partial_sort_copy (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __result_first, _RandomAccessIterator __result_last)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _Compare >`
`_RandomAccessIterator std::partial_sort_copy (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __result_first, _RandomAccessIterator __result_last, _Compare __comp)`
- `template<typename _BidirectionalIterator >`
`bool std::prev_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last)`

- `template<typename _BidirectionalIterator, typename _Compare >`
`bool std::prev_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`void std::sort (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void std::sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`void std::stable_sort (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void std::stable_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`

2.53.1 Detailed Description

2.53.2 Function Documentation

2.53.2.1 `template<typename _BidirectionalIterator > void std::inplace_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last) [inline]`

Merges two sorted ranges in place.

Parameters

<code>__first</code>	An iterator.
<code>__middle</code>	Another iterator.
<code>__last</code>	Another iterator.

Returns

Nothing.

Merges two sorted and consecutive ranges, [`__first`,`__middle`) and [`__middle`,`__last`), and puts the result in [`__first`,`__last`). The output will be sorted. The sort is *stable*, that is, for equivalent elements in the two ranges, elements from the first range will always come before elements from the second.

If enough additional memory is available, this takes (`__last`-`__first`)-1 comparisons. Otherwise an NlogN algorithm is used, where N is distance(`__first`,`__last`).

Definition at line 2586 of file `stl_algo.h`.

2.53.2.2 `template<typename _BidirectionalIterator, typename _Compare > void std::inplace_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Compare __comp) [inline]`

Merges two sorted ranges in place.

Parameters

<code>__first</code>	An iterator.
<code>__middle</code>	Another iterator.
<code>__last</code>	Another iterator.
<code>__comp</code>	A functor to use for comparisons.

Returns

Nothing.

Merges two sorted and consecutive ranges, [`__first`,`__middle`) and [`middle`,`last`), and puts the result in [`__first`,`last`). The output will be sorted. The sort is *stable*, that is, for equivalent elements in the two ranges, elements from the first range will always come before elements from the second.

If enough additional memory is available, this takes $(\text{__last} - \text{__first}) - 1$ comparisons. Otherwise an $N \log N$ algorithm is used, where N is $\text{distance}(\text{__first}, \text{__last})$.

The comparison function should have the same effects on ordering as the function used for the initial sort.

Definition at line 2626 of file `stl_algo.h`.

```
2.53.2.3  template<typename _ForwardIterator > bool std::is_sorted ( _ForwardIterator __first, _ForwardIterator __last )
           [inline]
```

Determines whether the elements of a sequence are sorted.

Parameters

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.

Returns

True if the elements are sorted, false otherwise.

Definition at line 3208 of file `stl_algo.h`.

References `std::is_sorted_until()`.

```
2.53.2.4  template<typename _ForwardIterator , typename _Compare > bool std::is_sorted ( _ForwardIterator __first,
           _ForwardIterator __last, _Compare __comp ) [inline]
```

Determines whether the elements of a sequence are sorted according to a comparison functor.

Parameters

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.
<code>__comp</code>	A comparison functor.

Returns

True if the elements are sorted, false otherwise.

Definition at line 3222 of file `stl_algo.h`.

References `std::is_sorted_until()`.

2.53.2.5 `template<typename _ForwardIterator> _ForwardIterator std::is_sorted_until (_ForwardIterator __first, _ForwardIterator __last) [inline]`

Determines the end of a sorted sequence.

Parameters

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.

Returns

An iterator pointing to the last iterator `i` in `[__first, __last)` for which the range `[__first, i)` is sorted.

Definition at line 3251 of file `stl_algo.h`.

2.53.2.6 `template<typename _ForwardIterator, typename _Compare> _ForwardIterator std::is_sorted_until (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp) [inline]`

Determines the end of a sorted sequence using comparison functor.

Parameters

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.
<code>__comp</code>	A comparison functor.

Returns

An iterator pointing to the last iterator `i` in `[__first, __last)` for which the range `[__first, i)` is sorted.

Definition at line 3274 of file `stl_algo.h`.

Referenced by `std::is_sorted()`.

2.53.2.7 `template<typename _I1, typename _I2> bool std::lexicographical_compare (_I1 __first1, _I1 __last1, _I2 __first2, _I2 __last2) [inline]`

Performs **dictionary** comparison on ranges.

Parameters

<code>__first1</code>	An input iterator.
<code>__last1</code>	An input iterator.
<code>__first2</code>	An input iterator.
<code>__last2</code>	An input iterator.

Returns

A boolean true or false.

Returns true if the sequence of elements defined by the range `[first1,last1)` is lexicographically less than the sequence of elements defined by the range `[first2,last2)`. Returns false otherwise. (Quoted from [25.3.8]/1.) If the iterators are all character pointers, then this is an inline call to `memcmp`.

Definition at line 1213 of file `stl_algobase.h`.

```
2.53.2.8  template<typename _I1, typename _I2, typename _Compare > bool std::lexicographical_compare ( _I1 __first1, _I1
    __last1, _I2 __first2, _I2 __last2, _Compare __comp ) [inline]
```

Performs **dictionary** comparison on ranges.

Parameters

<code>__first1</code>	An input iterator.
<code>__last1</code>	An input iterator.
<code>__first2</code>	An input iterator.
<code>__last2</code>	An input iterator.
<code>__comp</code>	A comparison functor .

Returns

A boolean true or false.

The same as the four-parameter `lexicographical_compare`, but uses the `comp` parameter instead of `<`.

Definition at line 1249 of file `stl_algobase.h`.

Referenced by `std::operator<()`.

```
2.53.2.9  template<typename _Tp > const _Tp & std::max ( const _Tp & __a, const _Tp & __b ) [inline]
```

This does what you think it does.

Parameters

<code>__a</code>	A thing of arbitrary type.
<code>__b</code>	Another thing of arbitrary type.

Returns

The greater of the parameters.

This is the simple classic generic implementation. It will work on temporary expressions, since they are only evaluated once, unlike a preprocessor macro.

Definition at line 217 of file `stl_algobase.h`.

Referenced by `__gnu_parallel::parallel_nth_element()`, `__gnu_parallel::parallel_random_shuffle_drs()`, `std::vector< sub_match< _Bi_iter >, allocator< sub_match< _Bi_iter > > >::M_allocate_and_copy()`, `std::Deque< _base< _Tp, _Alloc >::M_initialize_map()`, `std::discard_block_engine< _RandomNumberEngine, __p, __r >::max()`, `std::shuffle_order_engine< _RandomNumberEngine, __k >::max()`, `std::normal_distribution< result_type >::max()`, `std::lognormal_distribution< _RealType >::max()`, `std::gamma_distribution< result_type >::max()`, `std::chi_squared_distribution< _RealType >::max()`, `std::cauchy_distribution< _RealType >::max()`, `std::fisher_f_distribution< _RealType >::max()`, `std::student_t_distribution< _RealType >::max()`, `std::bernoulli_distribution::max()`, `std::geometric_distribution< _IntType >::max()`, `std::negative_binomial_distribution< _IntType >::max()`, `std::poisson_distribution< _IntType >::max()`, `std::exponential_distribution< _RealType >::max()`, `std::weibull_distribution< _RealType >::max()`, `std::extreme_value_distribution< _RealType >::max()`, `std::minmax_element()`, `__gnu_parallel::multiseq_partition()`, `__gnu_parallel::multiseq_selection()`, and `std::operator!=()`.

2.53.2.10 `template<typename _Tp, typename _Compare> const _Tp & std::max (const _Tp & __a, const _Tp & __b, _Compare __comp) [inline]`

This does what you think it does.

Parameters

<code>__a</code>	A thing of arbitrary type.
<code>__b</code>	Another thing of arbitrary type.
<code>__comp</code>	A comparison functor .

Returns

The greater of the parameters.

This will work on temporary expressions, since they are only evaluated once, unlike a preprocessor macro.

Definition at line 261 of file `stl_algobase.h`.

References `std::move()`.

2.53.2.11 `template<typename _ForwardIterator> _ForwardIterator std::max_element (_ForwardIterator __first, _ForwardIterator __last) [inline]`

Return the maximum element in a range.

Parameters

<code>__first</code>	Start of range.
<code>__last</code>	End of range.

Returns

Iterator referencing the first instance of the largest value.

Definition at line 5488 of file `stl_algo.h`.

```
2.53.2.12 template<typename _ForwardIterator, typename _Compare> _ForwardIterator std::max_element ( _ForwardIterator
    __first, _ForwardIterator __last, _Compare __comp ) [inline]
```

Return the maximum element in a range using comparison functor.

Parameters

<code>__first</code>	Start of range.
<code>__last</code>	End of range.
<code>__comp</code>	Comparison functor.

Returns

Iterator referencing the first instance of the largest value according to `__comp`.

Definition at line 5511 of file `stl_algo.h`.

Referenced by `std::minmax_element()`.

```
2.53.2.13 template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator> _OutputIterator std::merge (
    _InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result )
    [inline]
```

Merges two sorted ranges.

Parameters

<code>__first1</code>	An iterator.
<code>__first2</code>	Another iterator.
<code>__last1</code>	Another iterator.
<code>__last2</code>	Another iterator.
<code>__result</code>	An iterator pointing to the end of the merged range.

Returns

An iterator pointing to the first element *not less than* *val*.

Merges the ranges `[__first1, __last1)` and `[__first2, __last2)` into the sorted range `[__result, __result + (__last1 - __first1) + (__last2 - __first2))`. Both input ranges must be sorted, and the output range must not overlap with either of the input ranges. The sort is *stable*, that is, for equivalent elements in the two ranges, elements from the first range will always come before elements from the second.

Definition at line 4767 of file `stl_algo.h`.

2.53.2.14 `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >
_OutputIterator std::merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2
__last2, _OutputIterator __result, _Compare __comp) [inline]`

Merges two sorted ranges.

Parameters

<code>__first1</code>	An iterator.
<code>__first2</code>	Another iterator.
<code>__last1</code>	Another iterator.
<code>__last2</code>	Another iterator.
<code>__result</code>	An iterator pointing to the end of the merged range.
<code>__comp</code>	A functor to use for comparisons.

Returns

An iterator pointing to the first element "not less than" *val*.

Merges the ranges [`__first1`,`__last1`) and [`__first2`,`__last2`) into the sorted range [`__result`, `__result` + (`__last1`-`__first1`) + (`__last2`-`__first2`)). Both input ranges must be sorted, and the output range must not overlap with either of the input ranges. The sort is *stable*, that is, for equivalent elements in the two ranges, elements from the first range will always come before elements from the second.

The comparison function should have the same effects on ordering as the function used for the initial sort.

Definition at line 4815 of file `stl_algo.h`.

References `std::__inplace_stable_sort()`.

2.53.2.15 `template<typename _Tp > const _Tp & std::min (const _Tp & __a, const _Tp & __b) [inline]`

This does what you think it does.

Parameters

<code>__a</code>	A thing of arbitrary type.
<code>__b</code>	Another thing of arbitrary type.

Returns

The lesser of the parameters.

This is the simple classic generic implementation. It will work on temporary expressions, since they are only evaluated once, unlike a preprocessor macro.

Definition at line 194 of file `stl_algobase.h`.

Referenced by `std::__move_merge()`, `__gnu_parallel::__parallel_random_shuffle_drs()`, `__gnu_parallel::__parallel__sort_qs_divide()`, `__gnu_profile::__report()`, `__gnu_parallel::__search_template()`, `__gnu_parallel::__sequential__random_shuffle()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::compare()`, `std::basic_string<_Ch_type >::compare()`, `std::codecvt<_InternT, _ExternT, encoding_state >::do_out()`, `std::fill_n()`, `std::discard__block_engine<_RandomNumberEngine, __p, __r >::min()`, `std::shuffle_order_engine<_RandomNumberEngine, __k >::min()`, `std::bernoulli_distribution::min()`, `std::minmax_element()`, `__gnu_parallel::multiseq_partition()`, `__gnu__parallel::multiseq_selection()`, and `std::operator!=()`.

2.53.2.16 `template<typename _Tp, typename _Compare > const _Tp & std::min (const _Tp & __a, const _Tp & __b, _Compare __comp) [inline]`

This does what you think it does.

Parameters

<code>__a</code>	A thing of arbitrary type.
<code>__b</code>	Another thing of arbitrary type.
<code>__comp</code>	A comparison functor .

Returns

The lesser of the parameters.

This will work on temporary expressions, since they are only evaluated once, unlike a preprocessor macro.

Definition at line 240 of file `stl_algobase.h`.

2.53.2.17 `template<typename _ForwardIterator > _ForwardIterator std::min_element (_ForwardIterator __first, _ForwardIterator __last) [inline]`

Return the minimum element in a range.

Parameters

<code>__first</code>	Start of range.
<code>__last</code>	End of range.

Returns

Iterator referencing the first instance of the smallest value.

Definition at line 5429 of file `stl_algo.h`.

2.53.2.18 `template<typename _ForwardIterator, typename _Compare > _ForwardIterator std::min_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp) [inline]`

Return the minimum element in a range using comparison functor.

Parameters

<code>__first</code>	Start of range.
<code>__last</code>	End of range.
<code>__comp</code>	Comparison functor.

Returns

Iterator referencing the first instance of the smallest value according to `__comp`.

Definition at line 5452 of file `stl_algo.h`.

Referenced by `std::minmax_element()`.

2.53.2.19 `template<typename _Tp > pair< const _Tp &, const _Tp & > std::minmax (const _Tp & __a, const _Tp & __b)`
`[inline]`

Determines min and max at once as an ordered pair.

Parameters

<code>__↵ __a</code>	A thing of arbitrary type.
<code>__↵ __b</code>	Another thing of arbitrary type.

Returns

A pair(`__b`, `__a`) if `__b` is smaller than `__a`, pair(`__a`, `__b`) otherwise.

Definition at line 3298 of file `stl_algo.h`.

Referenced by `std::minmax_element()`.

2.53.2.20 `template<typename _Tp, typename _Compare > pair< const _Tp &, const _Tp & > std::minmax (const _Tp & __a, const _Tp & __b, _Compare __comp)` `[inline]`

Determines min and max at once as an ordered pair.

Parameters

<code>__a</code>	A thing of arbitrary type.
<code>__b</code>	Another thing of arbitrary type.
<code>__comp</code>	A comparison functor .

Returns

A pair(`__b`, `__a`) if `__b` is smaller than `__a`, pair(`__a`, `__b`) otherwise.

Definition at line 3318 of file `stl_algo.h`.

References `std::make_pair()`.

2.53.2.21 `template<typename _ForwardIterator > pair<_ForwardIterator, _ForwardIterator> std::minmax_element (`
`_FForwardIterator __first, _ForwardIterator __last) [inline]`

Return a pair of iterators pointing to the minimum and maximum elements in a range.

Parameters

<code>__first</code>	Start of range.
<code>__last</code>	End of range.

Returns

make_pair(`m`, `M`), where `m` is the first iterator `i` in [`__first`, `__last`) such that no other element in the range is smaller, and where `M` is the last iterator `i` in [`__first`, `__last`) such that no other element in the range is larger.

Definition at line 3396 of file `stl_algo.h`.

2.53.2.22 `template<typename _ForwardIterator, typename _Compare > pair<_ForwardIterator, _ForwardIterator>`
`std::minmax_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp) [inline]`

Return a pair of iterators pointing to the minimum and maximum elements in a range.

Parameters

<code>__first</code>	Start of range.
<code>__last</code>	End of range.
<code>__comp</code>	Comparison functor.

Returns

make_pair(`m`, `M`), where `m` is the first iterator `i` in [`__first`, `__last`) such that no other element in the range is smaller, and where `M` is the last iterator `i` in [`__first`, `__last`) such that no other element in the range is larger.

Definition at line 3422 of file `stl_algo.h`.

References `std::__find_if()`, `std::advance()`, `std::distance()`, `std::pair<_T1, _T2>::first`, `std::make_pair()`, `std::max()`, `std::max_element()`, `std::min()`, `std::min_element()`, `std::minmax()`, `std::minmax_element()`, and `std::pair<_T1, _T2>::second`.

Referenced by `std::minmax_element()`.

2.53.2.23 `template<typename _BidirectionalIterator > bool std::next_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last) [inline]`

Permute range into the next *dictionary* ordering.

Parameters

<code>__first</code>	Start of range.
<code>__last</code>	End of range.

Returns

False if wrapped to first permutation, true otherwise.

Treats all permutations of the range as a set of *dictionary* sorted sequences. Permutes the current sequence into the next one of this set. Returns true if there are more sequences to generate. If the sequence is the largest of the set, the smallest is generated and false returned.

Definition at line 2957 of file `stl_algo.h`.

2.53.2.24 `template<typename _BidirectionalIterator , typename _Compare > bool std::next_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __comp) [inline]`

Permute range into the next *dictionary* ordering using comparison functor.

Parameters

<code>__first</code>	Start of range.
<code>__last</code>	End of range.
<code>__comp</code>	A comparison functor.

Returns

False if wrapped to first permutation, true otherwise.

Treats all permutations of the range [`__first`,`__last`) as a set of *dictionary* sorted sequences ordered by `__comp`. Permutes the current sequence into the next one of this set. Returns true if there are more sequences to generate. If the sequence is the largest of the set, the smallest is generated and false returned.

Definition at line 2988 of file `stl_algo.h`.

References `std::__iterator_category()`, `std::__reverse()`, and `std::iter_swap()`.

2.53.2.25 `template<typename _RandomAccessIterator > void std::nth_element (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __last) [inline]`

Sort a sequence just enough to find a particular position.

Parameters

<code>__first</code>	An iterator.
<code>__nth</code>	Another iterator.
<code>__last</code>	Another iterator.

Returns

Nothing.

Rearranges the elements in the range `[__first,__last)` so that `*__nth` is the same element that would have been in that position had the whole sequence been sorted. The elements either side of `*__nth` are not completely sorted, but for any iterator *i* in the range `[__first,__nth)` and any iterator *j* in the range `[__nth,__last)` it holds that `*j < *i` is false.

Definition at line 4603 of file `stl_algo.h`.

References `std::__lg()`.

```
2.53.2.26  template<typename _RandomAccessIterator, typename _Compare> void std::nth_element ( _RandomAccessIterator
    __first, _RandomAccessIterator __nth, _RandomAccessIterator __last, _Compare __comp ) [inline]
```

Sort a sequence just enough to find a particular position using a predicate for comparison.

Parameters

<code>__first</code>	An iterator.
<code>__nth</code>	Another iterator.
<code>__last</code>	Another iterator.
<code>__comp</code>	A comparison functor.

Returns

Nothing.

Rearranges the elements in the range `[__first,__last)` so that `*__nth` is the same element that would have been in that position had the whole sequence been sorted. The elements either side of `*__nth` are not completely sorted, but for any iterator *i* in the range `[__first,__nth)` and any iterator *j* in the range `[__nth,__last)` it holds that `__comp(*j,*i)` is false.

Definition at line 4641 of file `stl_algo.h`.

References `std::__lg()`.

```
2.53.2.27  template<typename _RandomAccessIterator> void std::partial_sort ( _RandomAccessIterator __first,
    _RandomAccessIterator __middle, _RandomAccessIterator __last ) [inline]
```

Sort the smallest elements of a sequence.

Parameters

<code>__first</code>	An iterator.
<code>__middle</code>	Another iterator.
<code>__last</code>	Another iterator.

Returns

Nothing.

Sorts the smallest (`__middle-__first`) elements in the range `[first,last)` and moves them to the range `[__first,__middle)`. The order of the remaining elements in the range `[__middle,__last)` is undefined. After the sort if *i* and *j* are iterators in the range `[__first,__middle)` such that *i* precedes *j* and *k* is an iterator in the range `[__middle,__last)` then `*j < *i` and `*k < *i` are both false.

Definition at line 4531 of file `stl_algo.h`.

```
2.53.2.28  template<typename _RandomAccessIterator, typename _Compare> void std::partial_sort ( _RandomAccessIterator
    __first, _RandomAccessIterator __middle, _RandomAccessIterator __last, _Compare __comp ) [inline]
```

Sort the smallest elements of a sequence using a predicate for comparison.

Parameters

<code>__first</code>	An iterator.
<code>__middle</code>	Another iterator.
<code>__last</code>	Another iterator.
<code>__comp</code>	A comparison functor.

Returns

Nothing.

Sorts the smallest (`__middle-__first`) elements in the range `[__first,__last)` and moves them to the range `[__first,__middle)`. The order of the remaining elements in the range `[__middle,__last)` is undefined. After the sort if *i* and *j* are iterators in the range `[__first,__middle)` such that *i* precedes *j* and *k* is an iterator in the range `[__middle,__last)` then `*__comp(j,*i)` and `__comp(*k,*i)` are both false.

Definition at line 4568 of file `stl_algo.h`.

```
2.53.2.29  template<typename _InputIterator, typename _RandomAccessIterator> _RandomAccessIterator std::partial_sort_copy (
    _InputIterator __first, _InputIterator __last, _RandomAccessIterator __result_first, _RandomAccessIterator __result_last )
    [inline]
```

Copy the smallest elements of a sequence.

Parameters

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.
<code>__result_first</code>	A random-access iterator.
<code>__result_last</code>	Another random-access iterator.

Returns

An iterator indicating the end of the resulting sequence.

Copies and sorts the smallest N values from the range $[_\text{first}, _\text{last})$ to the range beginning at `__result_first`, where the number of elements to be copied, N , is the smaller of $(_\text{last} - _\text{first})$ and $(_\text{result_last} - _\text{result_first})$. After the sort if i and j are iterators in the range $[_\text{result_first}, _\text{result_first} + N)$ such that i precedes j then $*j < *i$ is false. The value returned is `__result_first + N`.

Definition at line 1738 of file `stl_algo.h`.

```
2.53.2.30  template<typename _InputIterator, typename _RandomAccessIterator, typename _Compare> _RandomAccessIterator
std::partial_sort_copy ( _InputIterator __first, _InputIterator __last, _RandomAccessIterator __result_first,
    _RandomAccessIterator __result_last, _Compare __comp ) [inline]
```

Copy the smallest elements of a sequence using a predicate for comparison.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	Another input iterator.
<code>__result_first</code>	A random-access iterator.
<code>__result_last</code>	Another random-access iterator.
<code>__comp</code>	A comparison functor.

Returns

An iterator indicating the end of the resulting sequence.

Copies and sorts the smallest N values from the range $[_\text{first}, _\text{last})$ to the range beginning at `result_first`, where the number of elements to be copied, N , is the smaller of $(_\text{last} - _\text{first})$ and $(_\text{result_last} - _\text{result_first})$. After the sort if i and j are iterators in the range $[_\text{result_first}, _\text{result_first} + N)$ such that i precedes j then `__comp(*j, *i)` is false. The value returned is `__result_first + N`.

Definition at line 1787 of file `stl_algo.h`.

```
2.53.2.31  template<typename _BidirectionalIterator> bool std::prev_permutation ( _BidirectionalIterator __first,
    _BidirectionalIterator __last ) [inline]
```

Permute range into the previous *dictionary* ordering.

Parameters

<code>__first</code>	Start of range.
<code>__last</code>	End of range.

Returns

False if wrapped to last permutation, true otherwise.

Treats all permutations of the range as a set of *dictionary* sorted sequences. Permutes the current sequence into the previous one of this set. Returns true if there are more sequences to generate. If the sequence is the smallest of the set, the largest is generated and false returned.

Definition at line 3055 of file `stl_algo.h`.

```
2.53.2.32  template<typename _BidirectionalIterator, typename _Compare> bool std::prev_permutation ( _BidirectionalIterator
    __first, _BidirectionalIterator __last, _Compare __comp ) [inline]
```

Permute range into the previous *dictionary* ordering using comparison functor.

Parameters

<code>__first</code>	Start of range.
<code>__last</code>	End of range.
<code>__comp</code>	A comparison functor.

Returns

False if wrapped to last permutation, true otherwise.

Treats all permutations of the range [`__first`,`__last`) as a set of *dictionary* sorted sequences ordered by `__comp`. Permutes the current sequence into the previous one of this set. Returns true if there are more sequences to generate. If the sequence is the smallest of the set, the largest is generated and false returned.

Definition at line 3086 of file `stl_algo.h`.

```
2.53.2.33  template<typename _RandomAccessIterator> void std::sort ( _RandomAccessIterator __first, _RandomAccessIterator
    __last ) [inline]
```

Sort the elements of a sequence.

Parameters

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.

Returns

Nothing.

Sorts the elements in the range $[_first, _last)$ in ascending order, such that for each iterator i in the range $[_first, _last-1)$, $*(i+1) < *i$ is false.

The relative ordering of equivalent elements is not preserved, use `stable_sort()` if this is needed.

Definition at line 4677 of file `stl_algo.h`.

```
2.53.2.34 template<typename _RandomAccessIterator, typename _Compare> void std::sort ( _RandomAccessIterator __first,
    _RandomAccessIterator __last, _Compare __comp ) [inline]
```

Sort the elements of a sequence using a predicate for comparison.

Parameters

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.
<code>__comp</code>	A comparison functor.

Returns

Nothing.

Sorts the elements in the range $[_first, _last)$ in ascending order, such that `__comp(*(i+1),*i)` is false for every iterator i in the range $[_first, _last-1)$.

The relative ordering of equivalent elements is not preserved, use `stable_sort()` if this is needed.

Definition at line 4706 of file `stl_algo.h`.

```
2.53.2.35 template<typename _RandomAccessIterator> void std::stable_sort ( _RandomAccessIterator __first,
    _RandomAccessIterator __last ) [inline]
```

Sort the elements of a sequence, preserving the relative order of equivalent elements.

Parameters

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.

Returns

Nothing.

Sorts the elements in the range $[_first, _last)$ in ascending order, such that for each iterator i in the range $[_first, _last-1)$, $*(i+1) < *i$ is false.

The relative ordering of equivalent elements is preserved, so any two elements x and y in the range $[_first, _last)$ such that $x < y$ is false and $y < x$ is false will have the same relative ordering after calling `stable_sort()`.

Definition at line 4876 of file `stl_algo.h`.

2.53.2.36 `template<typename _RandomAccessIterator, typename _Compare> void std::stable_sort (_RandomAccessIterator
__first, _RandomAccessIterator __last, _Compare __comp) [inline]`

Sort the elements of a sequence using a predicate for comparison, preserving the relative order of equivalent elements.

Parameters

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.
<code>__comp</code>	A comparison functor.

Returns

Nothing.

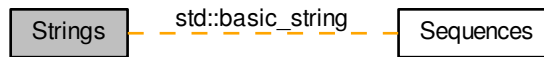
Sorts the elements in the range $[_first, _last)$ in ascending order, such that for each iterator i in the range $[_first, _last-1)$, `__comp(*(i+1),*i)` is false.

The relative ordering of equivalent elements is preserved, so any two elements x and y in the range $[_first, _last)$ such that `__comp(x,y)` is false and `__comp(y,x)` is false will have the same relative ordering after calling `stable_sort()`.

Definition at line 4909 of file `stl_algo.h`.

2.54 Strings

Collaboration diagram for Strings:



Classes

- class `std::basic_string< _CharT, _Traits, _Alloc >`
- struct `std::char_traits< _CharT >`

Typedefs

- typedef `basic_string< char >` `std::string`
- typedef `basic_string< char16_t >` `std::u16string`
- typedef `basic_string< char32_t >` `std::u32string`
- typedef `basic_string< wchar_t >` `std::wstring`

2.54.1 Detailed Description

2.54.2 Typedef Documentation

2.54.2.1 typedef `basic_string<char>` `std::string`

A string of `char`.

Definition at line 62 of file `stringfwd.h`.

2.54.2.2 typedef `basic_string<char16_t>` `std::u16string`

A string of `char16_t`.

Definition at line 78 of file `stringfwd.h`.

2.54.2.3 typedef `basic_string<char32_t>` `std::u32string`

A string of `char32_t`.

Definition at line 81 of file `stringfwd.h`.

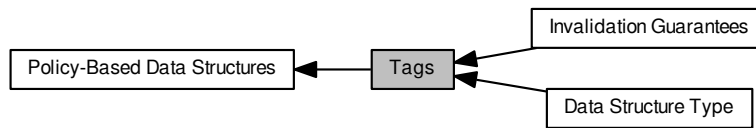
2.54.2.4 typedef `basic_string<wchar_t>` `std::wstring`

A string of `wchar_t`.

Definition at line 68 of file `stringfwd.h`.

2.55 Tags

Collaboration diagram for Tags:



Modules

- [Data Structure Type](#)
- [Invalidation Guarantees](#)

Classes

- [struct `__gnu_pbds::trivial_iterator_tag`](#)

Typedefs

- [typedef void `__gnu_pbds::trivial_iterator_difference_type`](#)

2.55.1 Detailed Description

2.55.2 Typedef Documentation

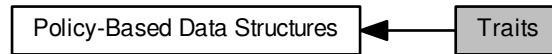
2.55.2.1 typedef void `__gnu_pbds::trivial_iterator_difference_type`

Prohibit moving trivial iterators.

Definition at line 79 of file `tag_and_trait.hpp`.

2.56 Traits

Collaboration diagram for Traits:



Classes

- struct `__gnu_pbds::container_traits< Cntnr >`
- struct `__gnu_pbds::container_traits_base< _Tag >`
- struct `__gnu_pbds::container_traits_base< binary_heap_tag >`
- struct `__gnu_pbds::container_traits_base< binomial_heap_tag >`
- struct `__gnu_pbds::container_traits_base< cc_hash_tag >`
- struct `__gnu_pbds::container_traits_base< gp_hash_tag >`
- struct `__gnu_pbds::container_traits_base< list_update_tag >`
- struct `__gnu_pbds::container_traits_base< ov_tree_tag >`
- struct `__gnu_pbds::container_traits_base< pairing_heap_tag >`
- struct `__gnu_pbds::container_traits_base< pat_trie_tag >`
- struct `__gnu_pbds::container_traits_base< rb_tree_tag >`
- struct `__gnu_pbds::container_traits_base< rc_binomial_heap_tag >`
- struct `__gnu_pbds::container_traits_base< splay_tree_tag >`
- struct `__gnu_pbds::container_traits_base< thin_heap_tag >`
- struct `__gnu_pbds::detail::bin_search_tree_traits< Key, Mapped, Cmp_Fn, Node_Update, Node, _Alloc >`
- struct `__gnu_pbds::detail::bin_search_tree_traits< Key, null_type, Cmp_Fn, Node_Update, Node, _Alloc >`
- struct `__gnu_pbds::detail::no_throw_copies< Key, Mapped >`
- struct `__gnu_pbds::detail::no_throw_copies< Key, null_type >`
- struct `__gnu_pbds::detail::stored_data< _Tv, _Th >`
- struct `__gnu_pbds::detail::stored_data< _Tv, null_type >`
- struct `__gnu_pbds::detail::stored_hash< _Th >`
- struct `__gnu_pbds::detail::stored_value< _Tv >`
- struct `__gnu_pbds::detail::tree_metadata_helper< Node_Update, _BTp >`
- struct `__gnu_pbds::detail::tree_metadata_helper< Node_Update, false >`
- struct `__gnu_pbds::detail::tree_metadata_helper< Node_Update, true >`
- struct `__gnu_pbds::detail::tree_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc >`
- struct `__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >`
- struct `__gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >`
- struct `__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >`
- struct `__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >`
- struct `__gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >`
- struct `__gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >`
- struct `__gnu_pbds::detail::trie_metadata_helper< Node_Update, _BTp >`
- struct `__gnu_pbds::detail::trie_metadata_helper< Node_Update, false >`

- struct `__gnu_pbds::detail::trie_metadata_helper< Node_Update, true >`
- struct `__gnu_pbds::detail::trie_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc >`
- struct `__gnu_pbds::detail::trie_traits< Key, Mapped, _ATraits, Node_Update, pat_trie_tag, _Alloc >`
- struct `__gnu_pbds::detail::trie_traits< Key, null_type, _ATraits, Node_Update, pat_trie_tag, _Alloc >`
- struct `__gnu_pbds::detail::type_base< Key, Mapped, _Alloc, Store_Hash >`
- struct `__gnu_pbds::detail::type_base< Key, Mapped, _Alloc, false >`
- struct `__gnu_pbds::detail::type_base< Key, Mapped, _Alloc, true >`
- struct `__gnu_pbds::detail::type_base< Key, null_type, _Alloc, false >`
- struct `__gnu_pbds::detail::type_base< Key, null_type, _Alloc, true >`
- struct `__gnu_pbds::detail::type_dispatch< Key, Mapped, _Alloc, Store_Hash >`
- struct `__gnu_pbds::detail::types_traits< Key, Mapped, _Alloc, Store_Hash >`
- struct `__gnu_pbds::null_node_update< _Tp1, _Tp2, _Tp3, _Tp4 >`
- struct `__gnu_pbds::null_type`

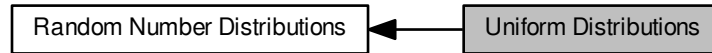
Variables

- static null_type `__gnu_pbds::detail::type_base< Key, null_type, _Alloc, false >::s_null_type`
- static null_type `__gnu_pbds::detail::type_base< Key, null_type, _Alloc, true >::s_null_type`

2.56.1 Detailed Description

2.57 Uniform Distributions

Collaboration diagram for Uniform Distributions:



Classes

- class `std::uniform_int_distribution<_IntType>`
- class `std::uniform_real_distribution<_RealType>`

Functions

- `template<typename _IntType>`
`bool std::operator!= (const std::uniform_int_distribution<_IntType> &__d1, const std::uniform_int_distribution<_IntType> &__d2)`
- `template<typename _IntType>`
`bool std::operator!= (const std::uniform_real_distribution<_IntType> &__d1, const std::uniform_real_distribution<_IntType> &__d2)`
- `template<typename _IntType, typename _CharT, typename _Traits>`
`std::basic_ostream<_CharT, _Traits> & std::operator<< (std::basic_ostream<_CharT, _Traits> &, const std::uniform_int_distribution<_IntType> &)`
- `template<typename _RealType, typename _CharT, typename _Traits>`
`std::basic_ostream<_CharT, _Traits> & std::operator<< (std::basic_ostream<_CharT, _Traits> &, const std::uniform_real_distribution<_RealType> &)`
- `template<typename _IntType, typename _CharT, typename _Traits>`
`std::basic_istream<_CharT, _Traits> & std::operator>> (std::basic_istream<_CharT, _Traits> &, std::uniform_int_distribution<_IntType> &)`
- `template<typename _RealType, typename _CharT, typename _Traits>`
`std::basic_istream<_CharT, _Traits> & std::operator>> (std::basic_istream<_CharT, _Traits> &, std::uniform_real_distribution<_RealType> &)`

2.57.1 Detailed Description

2.57.2 Function Documentation

2.57.2.1 `template<typename _IntType> bool std::operator!= (const std::uniform_int_distribution<_IntType> & __d1, const std::uniform_int_distribution<_IntType> & __d2) [inline]`

Return true if two uniform integer distributions have different parameters.

Definition at line 1827 of file random.h.

References `std::operator>>()`.

2.57.2.2 `template<typename _IntType> bool std::operator!= (const std::uniform_real_distribution<_IntType> & __d1, const std::uniform_real_distribution<_IntType> & __d2) [inline]`

Return true if two uniform real distributions have different parameters.

Definition at line 2036 of file random.h.

References `std::operator>>()`.

2.57.2.3 `template<typename _IntType, typename _CharT, typename _Traits> std::basic_ostream<_CharT, _Traits>& std::operator<< (std::basic_ostream<_CharT, _Traits> &, const std::uniform_int_distribution<_IntType> &)`

Inserts a `uniform_int_distribution` random number distribution `__x` into the output stream `os`.

Parameters

<code>__os</code>	An output stream.
<code>__x</code>	A <code>uniform_int_distribution</code> random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

2.57.2.4 `template<typename _RealType, typename _CharT, typename _Traits> std::basic_ostream<_CharT, _Traits>& std::operator<< (std::basic_ostream<_CharT, _Traits> &, const std::uniform_real_distribution<_RealType> &)`

Inserts a `uniform_real_distribution` random number distribution `__x` into the output stream `__os`.

Parameters

<code>__os</code>	An output stream.
<code>__x</code>	A <code>uniform_real_distribution</code> random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

2.57.2.5 `template<typename _IntType, typename _CharT, typename _Traits> std::basic_istream<_CharT, _Traits>& std::operator>> (std::basic_istream<_CharT, _Traits> &, std::uniform_int_distribution<_IntType> &)`

Extracts a `uniform_int_distribution` random number distribution `__x` from the input stream `__is`.

Parameters

<code>__is</code>	An input stream.
<code>__x</code>	A <code>uniform_int_distribution</code> random number generator engine.

Returns

The input stream with ___x extracted or in an error state.

2.57.2.6 `template<typename _RealType, typename _CharT, typename _Traits> std::basic_istream<_CharT, _Traits>& std::operator>>(std::basic_istream<_CharT, _Traits> &, std::uniform_real_distribution<_RealType> &)`

Extracts a `uniform_real_distribution` random number distribution ___x from the input stream ___is.

Parameters

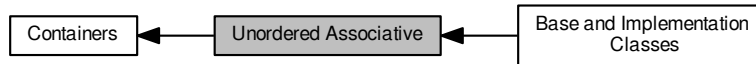
<code>__is</code>	An input stream.
<code>__x</code>	A <code>uniform_real_distribution</code> random number generator engine.

Returns

The input stream with ___x extracted or in an error state.

2.58 Unordered Associative

Collaboration diagram for Unordered Associative:



Modules

- [Base and Implementation Classes](#)

Classes

- class `std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>`
- class `std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>`
- class `std::unordered_multiset<_Value, _Hash, _Pred, _Alloc>`
- class `std::unordered_set<_Value, _Hash, _Pred, _Alloc>`

2.58.1 Detailed Description

Unordered associative containers allow fast retrieval of data based on keys.

Each container type is parameterized on a `Key` type, a `Hash` type providing a hashing functor, and an ordering relation used to sort the elements of the container.

All unordered associative containers must meet certain requirements, summarized in [tables](#).

2.59 Utilities

Collaboration diagram for Utilities:



Modules

- [Function Objects](#)

Classes

- struct [std::pair<_T1, _T2>](#)
- struct [std::piecewise_construct_t](#)

Functions

- `template<typename _Tp>`
`_Tp * std::__addressof (_Tp &__r) noexcept`
- `template<typename _Tp>`
`_Tp * std::addressof (_Tp &__r) noexcept`
- `template<typename _Tp>`
`constexpr _Tp && std::forward (typename std::remove_reference<_Tp>::type &__t) noexcept`
- `template<typename _Tp>`
`constexpr _Tp && std::forward (typename std::remove_reference<_Tp>::type &&__t) noexcept`
- `template<class _T1, class _T2>`
`constexpr pair< typename __decay_and_strip<_T1>::__type, typename __decay_and_strip<_T2>::__type > std::make_pair (_T1 &&__x, _T2 &&__y)`
- `template<typename _Tp>`
`constexpr std::remove_reference<_Tp>::type && std::move (_Tp &&__t) noexcept`
- `template<typename _Tp>`
`constexpr conditional< __move_if_noexcept_cond<_Tp>::value, const _Tp &, _Tp && >::type std::move_if_noexcept (_Tp &__x) noexcept`
- `template<class _T1, class _T2>`
`constexpr bool std::operator!= (const pair<_T1, _T2> &__x, const pair<_T1, _T2> &__y)`
- `template<class _T1, class _T2>`
`constexpr bool std::operator< (const pair<_T1, _T2> &__x, const pair<_T1, _T2> &__y)`
- `template<class _T1, class _T2>`
`constexpr bool std::operator<= (const pair<_T1, _T2> &__x, const pair<_T1, _T2> &__y)`
- `template<class _T1, class _T2>`
`constexpr bool std::operator== (const pair<_T1, _T2> &__x, const pair<_T1, _T2> &__y)`

- `template<class _T1, class _T2 >`
`constexpr bool std::operator> (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<class _T1, class _T2 >`
`constexpr bool std::operator>= (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<class _T1, class _T2 >`
`void std::swap (pair< _T1, _T2 > &__x, pair< _T1, _T2 > &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename _Tp >`
`void std::swap (_Tp &__a, _Tp &__b) noexcept(__and_< is_nothrow_move_constructible< _Tp >, is_nothrow_↵
_↵_move_assignable< _Tp >>::value)`
- `template<typename _Tp, size_t _Nm>`
`void std::swap (_Tp(&__a)[_Nm], _Tp(&__b)[_Nm]) noexcept(noexcept(swap(*__a,*__b)))`

Variables

- `constexpr piecewise_construct_t std::piecewise_construct`

2.59.1 Detailed Description

2.59.2 Function Documentation

2.59.2.1 `template<typename _Tp > _Tp* std::__addressof (_Tp &__r) [inline], [noexcept]`

Same as C++11 `std::addressof`.

Definition at line 47 of file `move.h`.

Referenced by `__gnu_debug::__valid_range()`, `std::_Destroy()`, `std::vector< sub_match< _Bi_iter >, allocator< sub_↵
_match< _Bi_iter > > >::__M_allocate_and_copy()`, `__gnu_cxx::bitmap_allocator< _Tp >::__M_deallocate_single_↵
_object()`, `std::addressof()`, `std::_Temporary_buffer< _ForwardIterator, _Tp >::end()`, `__gnu_debug::_Safe_local_↵
iterator< _Iterator, _Sequence >::operator->()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator->()`,
`std::list< __inp, __rebind_inp >::reverse()`, `std::uninitialized_copy()`, `std::uninitialized_fill()`, and `std::uninitialized_fill_n()`.

2.59.2.2 `template<typename _Tp > _Tp* std::addressof (_Tp &__r) [inline], [noexcept]`

Returns the actual address of the object or function referenced by `r`, even in the presence of an overloaded operator`&`.

Parameters

↵	Reference to an object or function.
↵	
↵	
↵	
<i>r</i>	

Returns

The actual address.

Definition at line 135 of file move.h.

References `std::__addressof()`.

Referenced by `__gnu_cxx::operator==(, and std::pointer_traits< _Tp * >::pointer_to().`

2.59.2.3 `template<typename _Tp > constexpr _Tp&& std::forward (typename std::remove_reference< _Tp >::type & __t)`
`[noexcept]`

Forward an lvalue.

Returns

The parameter cast to the specified type.

This function is used to implement "perfect forwarding".

Definition at line 76 of file move.h.

2.59.2.4 `template<typename _Tp > constexpr _Tp&& std::forward (typename std::remove_reference< _Tp >::type && __t)`
`[noexcept]`

Forward an rvalue.

Returns

The parameter cast to the specified type.

This function is used to implement "perfect forwarding".

Definition at line 87 of file move.h.

2.59.2.5 `template<class _T1 , class _T2 > constexpr pair<typename __decay_and_strip<_T1>::__type, typename`
`__decay_and_strip<_T2>::__type> std::make_pair (_T1 && __x, _T2 && __y)`

A convenience wrapper for creating a pair from two objects.

Parameters

<code>__x</code>	The first object.
<code>__y</code>	The second object.

Returns

A newly-constructed `pair<>` object of the appropriate type.

The standard requires that the objects be passed by reference-to-const, but LWG issue #181 says they should be passed by const value. We follow the LWG by default.

Definition at line 276 of file `stl_pair.h`.

Referenced by `__gnu_parallel::__find_template()`, `__gnu_debug::__get_distance()`, `__gnu_parallel::__parallel_merge`, `__advance()`, `__gnu_parallel::__parallel_sort_qsb()`, `__gnu_parallel::__qsb_local_sort_with_helping()`, `__gnu_cxx::free_list::M_insert()`, `__gnu_parallel::__find_if_selector::M_sequential_algorithm()`, `__gnu_parallel::__adjacent`, `__find_selector::M_sequential_algorithm()`, `__gnu_parallel::__find_first_of_selector<_FIterator>::M_sequential`, `__algorithm()`, `__gnu_pbds::detail::pat_trie_base::Node_iter<Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc>::get_child()`, `std::minmax()`, `std::minmax_element()`, `__gnu_parallel::multiseq_partition()`, `__gnu_parallel::multiseq`, `__selection()`, `__gnu_pbds::detail::pat_trie_map<Key, Mapped, Node_And_It_Traits, _Alloc>::node_end()`, `__gnu`, `__parallel::parallel_multiway_merge()`, `__gnu_parallel::parallel_sort_mwms_pu()`, and `__gnu_pbds::detail::pat_trie`, `base::_Node_citer<Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc>::valid_prefix()`.

2.59.2.6 `template<typename _Tp> constexpr std::remove_reference<_Tp>::type&& std::move (_Tp && __t) [noexcept]`

Convert a value to an rvalue.

Parameters

<code>__t</code>	A thing of arbitrary type.
<code>__t</code>	
<code>__t</code>	
<code>__t</code>	
<code>t</code>	

Returns

The parameter cast to an rvalue-reference to allow moving it.

Definition at line 101 of file `move.h`.

Referenced by `std::vector<sub_match<_Bi_iter>, allocator<sub_match<_Bi_iter>>>::M_allocate_and`, `copy()`, `std::Deque_iterator<_Tp, _Tp &, _Tp * >::M_set_node()`, `std::basic_regex<_Ch_type, _Rx_traits>`, `__assign()`, `std::auto_ptr<_Tp>::auto_ptr()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::compare()`, `std::unordered_set<_Value, _Hash, _Pred, _Alloc>::insert()`, `std::multiset<_Key, _Compare, _Alloc>::insert()`, `std`, `__set<_Key, _Compare, _Alloc>::insert()`, `std::unordered_multiset<_Value, _Hash, _Pred, _Alloc>::insert()`, `std`, `__vector<sub_match<_Bi_iter>, allocator<sub_match<_Bi_iter>>>::insert()`, `std::list<__inp, __rebind_inp`, `>::insert()`, `std::deque<_StateSeqT>::insert()`, `std::forward_list<_Tp, _Alloc>::insert_after()`, `std::inserter()`, `std`, `__max()`, `std::forward_list<_Tp, _Alloc>::merge()`, `std::move()`, `std::move_if_noexcept()`, `std::operator+()`, `std::back`, `__insert_iterator<_Container>::operator=()`, `std::vector<sub_match<_Bi_iter>, allocator<sub_match<_Bi`, `iter>>::operator=()`, `std::front_insert_iterator<_Container>::operator=()`, `std::forward_list<_Tp, _Alloc>`, `__operator=()`, `std::basic_regex<_Ch_type, _Rx_traits>::operator=()`, `std::insert_iterator<_Container>::operator=()`, `std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::operator[]()`, `std::priority_queue<_Tp, _Sequence,` , `__Compare>::priority_queue()`, `std::stack<_StateSeqT>::push()`, `std::queue<_Tp, _Sequence>::push()`, `std`, `__priority_queue<_Tp, _Sequence, _Compare>::push()`, `std::vector<sub_match<_Bi_iter>, allocator<sub`, `match<_Bi_iter>>::push_back()`, `std::list<__inp, __rebind_inp>::push_back()`, `std::deque<_StateSeqT>`, `__push_back()`, `std::forward_list<_Tp, _Alloc>::push_front()`, `std::list<__inp, __rebind_inp>::push_front()`, `std`, `__deque<_StateSeqT>::push_front()`, `std::queue<_Tp, _Sequence>::queue()`, `std::forward_list<_Tp, _Alloc>`, `__reverse()`, `std::allocator_traits<allocator<_Tp>>::select_on_container_copy_construction()`, `std::shared_ptr<__`

`_detail::_NFA<_Rx_traits>::shared_ptr()`, `std::list<__inp, __rebind_inp>::splice()`, `std::forward_list<_Tp, __Alloc>::splice_after()`, `std::stack<_StateSeqT>::stack()`, `std::basic_regex<_Ch_type, _Rx_traits>::swap()`, `std::__unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::unordered_map()`, `std::unordered_multimap<_Key, _Tp, __Hash, _Pred, _Alloc>::unordered_multimap()`, `std::unordered_multiset<_Value, _Hash, _Pred, _Alloc>::unordered_multiset()`, and `std::unordered_set<_Value, _Hash, _Pred, _Alloc>::unordered_set()`.

2.59.2.7 `template<typename _Tp> constexpr conditional<__move_if_noexcept_cond<_Tp>::value, const _Tp&, _Tp&&>::type
std::move_if_noexcept(_Tp &__x) [noexcept]`

Conditionally convert a value to an rvalue.

Parameters

<code>__x</code>	A thing of arbitrary type.
------------------	----------------------------

Returns

The parameter, possibly cast to an rvalue-reference.

Same as `std::move` unless the type's move constructor could throw and the type is copyable, in which case an lvalue-reference is returned instead.

Definition at line 121 of file `move.h`.

References `std::move()`.

2.59.2.8 `template<class _T1, class _T2> constexpr bool std::operator!=(const pair<_T1, _T2> &__x, const pair<_T1, _T2> &__y) [inline]`

Uses `operator==` to find the result.

Definition at line 227 of file `stl_pair.h`.

2.59.2.9 `template<class _T1, class _T2> constexpr bool std::operator< (const pair<_T1, _T2> &__x, const pair<_T1, _T2> &__y) [inline]`

<http://gcc.gnu.org/onlinedocs/libstdc++/manual/utilities.html>

Definition at line 220 of file `stl_pair.h`.

References `std::pair<_T1, _T2>::first`.

2.59.2.10 `template<class _T1, class _T2> constexpr bool std::operator<= (const pair<_T1, _T2> &__x, const pair<_T1, _T2> &__y) [inline]`

Uses `operator<` to find the result.

Definition at line 239 of file `stl_pair.h`.

2.59.2.11 `template<class _T1, class _T2> constexpr bool std::operator==(const pair<_T1, _T2> &__x, const pair<_T1, _T2> &__y) [inline]`

Two pairs of the same type are equal iff their members are equal.

Definition at line 214 of file `stl_pair.h`.

References `std::pair<_T1, _T2>::first`, and `std::pair<_T1, _T2>::second`.

2.59.2.12 `template<class _T1, class _T2> constexpr bool std::operator>(const pair<_T1, _T2> &__x, const pair<_T1, _T2> &__y) [inline]`

Uses `operator<` to find the result.

Definition at line 233 of file `stl_pair.h`.

2.59.2.13 `template<class _T1, class _T2> constexpr bool std::operator>=(const pair<_T1, _T2> &__x, const pair<_T1, _T2> &__y) [inline]`

Uses `operator<` to find the result.

Definition at line 245 of file `stl_pair.h`.

2.59.2.14 `template<class _T1, class _T2> void std::swap(pair<_T1, _T2> &__x, pair<_T1, _T2> &__y) [inline], [noexcept]`

See `std::pair::swap()`.

Definition at line 254 of file `stl_pair.h`.

2.59.2.15 `template<typename _Tp> void std::swap(_Tp &__a, _Tp &__b) const [inline], [noexcept]`

Swaps two values.

Parameters

<code>__↔ _a</code>	A thing of arbitrary type.
<code>__↔ _b</code>	Another thing of arbitrary type.

Returns

Nothing.

Definition at line 166 of file `move.h`.

Referenced by `__gnu_parallel::LoserTree< __stable, _Tp, _Compare >::__delete_min_insert()`, `__gnu_parallel::LoserTree< false, _Tp, _Compare >::__delete_min_insert()`, `std::__rotate()`, `__gnu_debug::Safe_iterator< __iterator, _Sequence >::Safe_iterator()`, `__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc >::cc_hash_table()`, `std::dynamic_pointer_cast()`, `__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >::gp_hash_table()`, `std::regex_traits< _Ch_type >::imbue()`, `std::iter_swap()`, `__gnu_pbds::list_update< Key, Mapped, Eq_Fn, Update_Policy, _Alloc >::list_update()`, `__gnu_pbds::detail::pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >::node_end()`, `std::list< __inp, __rebind_inp >::operator=()`, `std::deque< _StateSeqT >::operator=()`, `std::pair< __Bilter, _Bilter >::pair()`, `std::stack< _StateSeqT >::pop()`, `std::queue< _Tp, _Sequence >::pop()`, `std::priority_queue< _Tp, _Sequence, _Compare >::pop()`, `__gnu_pbds::priority_queue< _Tv, Cmp_Fn, Tag, _Alloc >::priority_queue()`, `std::unique_ptr< std::vector< bool > >::reset()`, `std::unique_ptr< _Tp[], _Dp >::reset()`, `std::forward_list< _Tp, __Alloc >::reverse()`, `std::allocator_traits< allocator< _Tp > >::select_on_container_copy_construction()`, `std::shared_ptr< __detail::NFA< _Rx_traits > >::shared_ptr()`, `std::unique_ptr< std::vector< bool > >::swap()`, `std::swap()`, `std::unique_ptr< _Tp[], _Dp >::swap()`, `std::basic_regex< _Ch_type, _Rx_traits >::swap()`, `std::forward_list< _Tp, __Alloc >::swap()`, `std::deque< _StateSeqT >::swap()`, `std::match_results< _Bi_iter >::swap()`, `__gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc >::tree()`, and `__gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc >::trie()`.

2.59.2.16 `template<typename _Tp, size_t _Nm> void std::swap (_Tp(&) __a[_Nm], _Tp(&) __b[_Nm]) [inline], [noexcept]`

Swap the contents of two arrays.

Definition at line 185 of file `move.h`.

References `std::swap()`.

2.59.3 Variable Documentation

2.59.3.1 `constexpr piecewise_construct_t std::piecewise_construct`

`piecewise_construct`

Definition at line 79 of file `stl_pair.h`.

Referenced by `std::map< _Key, _Tp, _Compare, _Alloc >::operator[]()`.

3 Namespace Documentation

3.1 `__gnu_cxx` Namespace Reference

Namespaces

- [__detail](#)
- [typelist](#)

Classes

- struct [__alloc_traits](#)
- struct [__common_pool_policy](#)
- class [__mt_alloc](#)
- class [__mt_alloc_base](#)
- struct [__per_type_pool_policy](#)
- class [__pool](#)
- class [__pool< false >](#)
- class [__pool< true >](#)
- class [__pool_alloc](#)
- class [__pool_alloc_base](#)
- struct [__pool_base](#)
- class [__rc_string_base](#)
- class [__scoped_lock](#)
- class [__versa_string](#)
- struct [_Caster](#)
- struct [_Char_types](#)
- class [_ExtPtr_allocator](#)
- struct [_Invalid_type](#)
- class [_Pointer_adapter](#)
- class [_Relative_pointer_impl](#)
- class [_Relative_pointer_impl< const _Tp >](#)
- class [_Std_pointer_impl](#)
- struct [_Unqualified_type](#)
- struct [annotate_base](#)
- class [array_allocator](#)
- class [array_allocator_base](#)
- class [bitmap_allocator](#)
- struct [char_traits](#)
- struct [character](#)
- struct [condition_base](#)
- class [debug_allocator](#)
- class [enc_filebuf](#)
- struct [encoding_char_traits](#)
- class [encoding_state](#)
- struct [forced_error](#)
- class [free_list](#)
- struct [limit_condition](#)
- class [malloc_allocator](#)

- class [new_allocator](#)
- struct [random_condition](#)
- class [recursive_init_error](#)
- class [stdio_filebuf](#)
- class [stdio_sync_filebuf](#)
- class [throw_allocator_base](#)
- struct [throw_allocator_limit](#)
- struct [throw_allocator_random](#)
- struct [throw_value_base](#)
- struct [throw_value_limit](#)
- struct [throw_value_random](#)

Typedefs

- typedef void(* [__destroy_handler](#)) (void *)
- typedef [__versa_string](#)< char, [std::char_traits](#)< char >, [std::allocator](#)< char >, [__rc_string_base](#) > [__rc_string](#)
- typedef [__vstring](#) [__sso_string](#)
- typedef [__versa_string](#)< char16_t, [std::char_traits](#)< char16_t >, [std::allocator](#)< char16_t >, [__rc_string_base](#) > [__u16rc_string](#)
- typedef [__u16vstring](#) [__u16sso_string](#)
- typedef [__versa_string](#)< char16_t > [__u16vstring](#)
- typedef [__versa_string](#)< char32_t, [std::char_traits](#)< char32_t >, [std::allocator](#)< char32_t >, [__rc_string_base](#) > [__u32rc_string](#)
- typedef [__u32vstring](#) [__u32sso_string](#)
- typedef [__versa_string](#)< char32_t > [__u32vstring](#)
- typedef [__versa_string](#)< char > [__vstring](#)
- typedef [__versa_string](#)< wchar_t, [std::char_traits](#)< wchar_t >, [std::allocator](#)< wchar_t >, [__rc_string_base](#) > [__wrc_string](#)
- typedef [__wvstring](#) [__wsso_string](#)
- typedef [__versa_string](#)< wchar_t > [__wvstring](#)

Enumerations

- enum { [_S_num_primes](#) }
- enum [_Lock_policy](#) { [_S_single](#), [_S_mutex](#), [_S_atomic](#) }

Functions

- static void [__atomic_add_single](#) (_Atomic_word * __mem, int __val)
- else [__atomic_add_single](#) (__mem, __val)
- [_Atomic_word](#) [__attribute__](#) ((__unused__)) [__exchange_and_add](#)(volatile [_Atomic_word](#) *
- template<class [_Tp](#) >
void [__aux_require_boolean_expr](#) (const [_Tp](#) & __t)
- template<typename [_ToType](#) , typename [_FromType](#) >
[_ToType](#) [__const_pointer_cast](#) (const [_FromType](#) & __arg)
- template<typename [_ToType](#) , typename [_FromType](#) >
[_ToType](#) [__const_pointer_cast](#) ([_FromType](#) * __arg)
- template<typename [_ToType](#) , typename [_FromType](#) >
[_ToType](#) [__dynamic_pointer_cast](#) (const [_FromType](#) & __arg)

- `template<typename _ToType, typename _FromType >`
`_ToType __dynamic_pointer_cast (_FromType *__arg)`
- `void __error_type_must_be_a_signed_integer_type ()`
- `void __error_type_must_be_an_integer_type ()`
- `void __error_type_must_be_an_unsigned_integer_type ()`
- `static _Atomic_word __exchange_and_add_single (_Atomic_word *__mem, int __val)`
- `else return __exchange_and_add_single (__mem, __val)`
- `template<class _Concept >`
`void __function_requires ()`
- `template<typename _Type >`
`bool __is_null_pointer (_Type *__ptr)`
- `template<typename _Type >`
`bool __is_null_pointer (_Type)`
- `bool __is_null_pointer (std::nullptr_t)`
- `template<typename _ToType, typename _FromType >`
`_ToType __reinterpret_pointer_cast (const _FromType &__arg)`
- `template<typename _ToType, typename _FromType >`
`_ToType __reinterpret_pointer_cast (_FromType *__arg)`
- `template<typename _ToType, typename _FromType >`
`_ToType __static_pointer_cast (const _FromType &__arg)`
- `template<typename _ToType, typename _FromType >`
`_ToType __static_pointer_cast (_FromType *__arg)`
- `size_t __stl_hash_string (const char *__s)`
- `unsigned long __stl_next_prime (unsigned long __n)`
- `template<typename _TRet, typename _Ret = _TRet, typename _CharT, typename... _Base>`
`_Ret __stoa (_TRet>(*__convf)(const _CharT *, _CharT **, _Base...), const char *__name, const _CharT *__str,`
`std::size_t *__idx, _Base... __base)`
- `void __throw_concurrency_lock_error ()`
- `void __throw_concurrency_unlock_error ()`
- `void __throw_forced_error ()`
- `template<typename _String, typename _CharT = typename _String::value_type>`
`_String __to_xstring (int(*__convf)(_CharT *, std::size_t, const _CharT *, __builtin_va_list), std::size_t __n,`
`const _CharT *__fmt,...)`
- `size_t __Bit_scan_forward (size_t __num)`
- `template<class _CharT, class _Traits >`
`void __Rope_fill (basic_ostream< _CharT, _Traits > &__o, size_t __n)`
- `template<class _CharT >`
`bool __Rope_is_simple (_CharT *)`
- `bool __Rope_is_simple (char *)`
- `bool __Rope_is_simple (wchar_t *)`
- `template<class _Rope_iterator >`
`void __Rope_rotate (_Rope_iterator __first, _Rope_iterator __middle, _Rope_iterator __last)`
- `template<typename _Tp >`
`bool operator!= (const new_allocator< _Tp > &, const new_allocator< _Tp > &)`
- `template<typename _Tp >`
`bool operator!= (const malloc_allocator< _Tp > &, const malloc_allocator< _Tp > &)`
- `template<typename _Tp, typename _Array >`
`bool operator!= (const array_allocator< _Tp, _Array > &, const array_allocator< _Tp, _Array > &)`
- `template<typename _Alloc >`
`bool operator!= (const debug_allocator< _Alloc > &__lhs, const debug_allocator< _Alloc > &__rhs)`
- `template<typename _Tp >`
`bool operator!= (const __pool_alloc< _Tp > &, const __pool_alloc< _Tp > &)`

- `template<typename _Tp1, typename _Tp2 >`
`bool operator!= (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator!= (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator!= (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp >`
`bool operator!= (const _Pointer_adapter< _Tp > &__lhs, int __rhs)`
- `template<typename _Tp >`
`bool operator!= (int __lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<typename _Tp >`
`bool operator!= (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<class _Val, class _Key, class _HF, class _Ex, class _Eq, class _All >`
`bool operator!= (const hashtable< _Val, _Key, _HF, _Ex, _Eq, _All > &__ht1, const hashtable< _Val, _Key, _HF, _Ex, _Eq, _All > &__ht2)`
- `template<typename _Tp, typename _Poolp >`
`bool operator!= (const __mt_alloc< _Tp, _Poolp > &, const __mt_alloc< _Tp, _Poolp > &)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >`
`bool operator!= (const __normal_iterator< _IteratorL, _Container > &__lhs, const __normal_iterator< _IteratorR, _Container > &__rhs) noexcept`
- `template<typename _Iterator, typename _Container >`
`bool operator!= (const __normal_iterator< _Iterator, _Container > &__lhs, const __normal_iterator< _Iterator, _Container > &__rhs) noexcept`
- `template<typename _Tp, typename _Cond >`
`bool operator!= (const throw_allocator_base< _Tp, _Cond > &, const throw_allocator_base< _Tp, _Cond > &)`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator!= (const bitmap_allocator< _Tp1 > &, const bitmap_allocator< _Tp2 > &) throw ()`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool operator!= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool operator!= (const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool operator!= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *__rhs)`
- `template<typename _Cond >`
`throw_value_base< _Cond > operator* (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)`
- `template<typename _Cond >`
`throw_value_base< _Cond > operator+ (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)`
- `template<typename _Iterator, typename _Container >`
`__normal_iterator< _Iterator, _Container > operator+ (typename __normal_iterator< _Iterator, _Container >::difference_type __n, const __normal_iterator< _Iterator, _Container > &__i) noexcept`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`__versa_string< _CharT, _Traits, _Alloc, _Base > operator+ (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`__versa_string< _CharT, _Traits, _Alloc, _Base > operator+ (const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`__versa_string< _CharT, _Traits, _Alloc, _Base > operator+ (_CharT __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`

- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base > operator+ (const __versa_string< _CharT, _Traits, _Alloc,
_Base > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base > operator+ (const __versa_string< _CharT, _Traits, _Alloc,
_Base > &__lhs, _CharT __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base > operator+ (__versa_string< _CharT, _Traits, _Alloc, _Base >
&&__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base > operator+ (const __versa_string< _CharT, _Traits, _Alloc,
_Base > &__lhs, __versa_string< _CharT, _Traits, _Alloc, _Base > &&__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base > operator+ (__versa_string< _CharT, _Traits, _Alloc, _Base >
&&__lhs, __versa_string< _CharT, _Traits, _Alloc, _Base > &&__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base > operator+ (const _CharT *__lhs, __versa_string< _CharT,
_Traits, _Alloc, _Base > &&__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base > operator+ (_CharT __lhs, __versa_string< _CharT, _Traits,
_Alloc, _Base > &&__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base > operator+ (__versa_string< _CharT, _Traits, _Alloc, _Base >
&&__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base > operator+ (__versa_string< _CharT, _Traits, _Alloc, _Base >
&&__lhs, _CharT __rhs)`
- `template<typename _Cond >
throw_value_base< _Cond > operator- (const throw_value_base< _Cond > &__a, const throw_value_base<
_Cond > &__b)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >
auto operator- (const __normal_iterator< _IteratorL, _Container > &__lhs, const __normal_iterator< _IteratorR,
_Container > &__rhs) noexcept-> decltype(__lhs.base()-__rhs.base())`
- `template<typename _Iterator, typename _Container >
__normal_iterator< _Iterator, _Container >::difference_type operator- (const __normal_iterator< _Iterator, _↵
_Container > &__lhs, const __normal_iterator< _Iterator, _Container > &__rhs) noexcept`
- `template<typename _Value, typename _Int, typename _St >
bool operator< (const character< _Value, _Int, _St > &lhs, const character< _Value, _Int, _St > &rhs)`
- `template<typename _Tp1, typename _Tp2 >
bool operator< (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >
bool operator< (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >
bool operator< (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)`
- `template<typename _Cond >
bool operator< (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >
bool operator< (const __normal_iterator< _IteratorL, _Container > &__lhs, const __normal_iterator< _↵
_IteratorR, _Container > &__rhs) noexcept`
- `template<typename _Iterator, typename _Container >
bool operator< (const __normal_iterator< _Iterator, _Container > &__lhs, const __normal_iterator< _Iterator,
_Container > &__rhs) noexcept`

- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool operator< (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT,`
`_Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool operator< (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT * __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool operator< (const _CharT * __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `std::ostream & operator<< (std::ostream &os, const annotate_base &__b)`
- `template<typename _CharT, typename _Traits, typename _StoreT >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`
`_Pointer_adapter< _StoreT > &__p)`
- `template<class _CharT, class _Traits, class _Alloc >`
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__o, const rope< _↵`
`_CharT, _Alloc > &__r)`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator<= (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator<= (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator<= (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp >`
`bool operator<= (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >`
`bool operator<= (const __normal_iterator< _IteratorL, _Container > &__lhs, const __normal_iterator< _↵`
`_IteratorR, _Container > &__rhs) noexcept`
- `template<typename _Iterator, typename _Container >`
`bool operator<= (const __normal_iterator< _Iterator, _Container > &__lhs, const __normal_iterator< _Iterator,`
`_Container > &__rhs) noexcept`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool operator<= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _↵`
`_CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool operator<= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT * __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool operator<= (const _CharT * __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _Value, typename _Int, typename _St >`
`bool operator== (const character< _Value, _Int, _St > &lhs, const character< _Value, _Int, _St > &rhs)`
- `template<typename _Tp >`
`bool operator== (const new_allocator< _Tp > &, const new_allocator< _Tp > &)`
- `template<typename _Tp >`
`bool operator== (const malloc_allocator< _Tp > &, const malloc_allocator< _Tp > &)`
- `template<typename _Tp, typename _Array >`
`bool operator== (const array_allocator< _Tp, _Array > &, const array_allocator< _Tp, _Array > &)`
- `template<typename _Tp >`
`bool operator== (const __pool_alloc< _Tp > &, const __pool_alloc< _Tp > &)`
- `template<class _Val, class _Key, class _HF, class _Ex, class _Eq, class _All >`
`bool operator== (const hashtable< _Val, _Key, _HF, _Ex, _Eq, _All > &__ht1, const hashtable< _Val, _Key,`
`_HF, _Ex, _Eq, _All > &__ht2)`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator== (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator== (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)`

- `template<typename _Tp1, typename _Tp2 >`
`bool operator== (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp >`
`bool operator== (const _Pointer_adapter< _Tp > &__lhs, int __rhs)`
- `template<typename _Tp >`
`bool operator== (int __lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<typename _Tp >`
`bool operator== (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<typename _Cond >`
`bool operator== (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)`
- `template<typename _Tp, typename _Poolp >`
`bool operator== (const __mt_alloc< _Tp, _Poolp > &, const __mt_alloc< _Tp, _Poolp > &)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >`
`bool operator== (const __normal_iterator< _IteratorL, _Container > &__lhs, const __normal_iterator< _↵IteratorR, _Container > &__rhs) noexcept`
- `template<typename _Iterator, typename _Container >`
`bool operator== (const __normal_iterator< _Iterator, _Container > &__lhs, const __normal_iterator< _Iterator, _Container > &__rhs) noexcept`
- `template<typename _Tp, typename _Cond >`
`bool operator== (const throw_allocator_base< _Tp, _Cond > &, const throw_allocator_base< _Tp, _Cond > &)`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator== (const bitmap_allocator< _Tp1 > &, const bitmap_allocator< _Tp2 > &) throw ()`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool operator== (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, template< typename, typename, typename > class _Base>`
`__enable_if< std::is_char< _CharT >::value, bool >::type operator== (const __versa_string< _CharT, std::char_traits< _CharT >, std::allocator< _CharT >, _Base > &__lhs, const __versa_string< _CharT, std::↵char_traits< _CharT >, std::allocator< _CharT >, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool operator== (const _CharT * __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool operator== (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT * __rhs)`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator> (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator> (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator> (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp >`
`bool operator> (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >`
`bool operator> (const __normal_iterator< _IteratorL, _Container > &__lhs, const __normal_iterator< _↵IteratorR, _Container > &__rhs) noexcept`
- `template<typename _Iterator, typename _Container >`
`bool operator> (const __normal_iterator< _Iterator, _Container > &__lhs, const __normal_iterator< _Iterator, _Container > &__rhs) noexcept`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool operator> (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool operator> (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT * __rhs)`

- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool operator> (const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator>= (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator>= (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator>= (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp >`
`bool operator>= (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >`
`bool operator>= (const __normal_iterator< _IteratorL, _Container > &__lhs, const __normal_iterator< _↵
_IteratorR, _Container > &__rhs) noexcept`
- `template<typename _Iterator, typename _Container >`
`bool operator>= (const __normal_iterator< _Iterator, _Container > &__lhs, const __normal_iterator< _Iterator,
_Container > &__rhs) noexcept`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool operator>= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _↵
_CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool operator>= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool operator>= (const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `void rotate (_Rope_iterator< char, __STL_DEFAULT_ALLOCATOR(char)> __first, _Rope_iterator< char,
__STL_DEFAULT_ALLOCATOR(char)> __middle, _Rope_iterator< char, __STL_DEFAULT_ALLOCAT↵
OR(char)> __last)`
- `template<typename _Tp >`
`void swap (_ExtPtr_allocator< _Tp > &__larg, _ExtPtr_allocator< _Tp > &__rarg)`
- `template<typename _Cond >`
`void swap (throw_value_base< _Cond > &__a, throw_value_base< _Cond > &__b)`
- `template<class _Val, class _Key, class _HF, class _Extract, class _EqKey, class _All >`
`void swap (hashtable< _Val, _Key, _HF, _Extract, _EqKey, _All > &__ht1, hashtable< _Val, _Key, _HF, _Extract,
_EqKey, _All > &__ht2)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`void swap (__versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, __versa_string< _CharT, _Traits, _Alloc,
_Base > &__rhs)`
- `_Atomic_word int throw ()`

Variables

- `static const _Lock_policy __default_lock_policy`
- `static _Atomic_word int __val`

3.1.1 Detailed Description

GNU extensions for public use.

3.1.2 Function Documentation

3.1.2.1 `template<typename _ToType , typename _FromType > _ToType __gnu_cxx::__static_pointer_cast (const _FromType & __arg) [inline]`

Casting operations for cases where `_FromType` is not a standard pointer. `_ToType` can be a standard or non-standard pointer. Given that `_FromType` is not a pointer, it must have a `get()` method that returns the standard pointer equivalent of the address it points to, and must have an `element_type` typedef which names the type it points to.

Definition at line 68 of file `cast.h`.

3.1.2.2 `template<typename _ToType , typename _FromType > _ToType __gnu_cxx::__static_pointer_cast (_FromType * __arg) [inline]`

Casting operations for cases where `_FromType` is a standard pointer. `_ToType` can be a standard or non-standard pointer.

Definition at line 96 of file `cast.h`.

3.1.2.3 `size_t __gnu_cxx::__Bit_scan_forward (size_t __num) [inline]`

Generic Version of the `bsf` instruction.

Definition at line 513 of file `bitmap_allocator.h`.

Referenced by `__gnu_cxx::bitmap_allocator< _Tp >::M_allocate_single_object()`.

3.1.2.4 `template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base> bool __gnu_cxx::operator!=(const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs) [inline]`

Test difference of two strings.

Parameters

<code>__lhs</code>	First string.
<code>__rhs</code>	Second string.

Returns

True if `__lhs.compare(__rhs) != 0`. False otherwise.

Definition at line 2388 of file `vstring.h`.

3.1.2.5 `template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base> bool __gnu_cxx::operator!=(const _CharT * __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs) [inline]`

Test difference of C string and string.

Parameters

<code>__lhs</code>	C string.
<code>__rhs</code>	String.

Returns

True if `__rhs.compare(__lhs) != 0`. False otherwise.

Definition at line 2401 of file `vstring.h`.

```
3.1.2.6 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
_Base> bool __gnu_cxx::operator!=( const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs, const _CharT *
__rhs ) [inline]
```

Test difference of string and C string.

Parameters

<code>__lhs</code>	String.
<code>__rhs</code>	C string.

Returns

True if `__lhs.compare(__rhs) != 0`. False otherwise.

Definition at line 2414 of file `vstring.h`.

```
3.1.2.7 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
_Base> __versa_string< _CharT, _Traits, _Alloc, _Base> __gnu_cxx::operator+ ( const __versa_string< _CharT,
_Traits, _Alloc, _Base > & __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs )
```

Concatenate two strings.

Parameters

<code>__lhs</code>	First string.
<code>__rhs</code>	Last string.

Returns

New string with value of `__lhs` followed by `__rhs`.

3.1.2.8 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string<_CharT, _Traits, _Alloc, _Base> __gnu_cxx::operator+ (const _CharT * __lhs, const __versa_string<_CharT, _Traits, _Alloc, _Base > & __rhs)`

Concatenate C string and string.

Parameters

<code>__lhs</code>	First string.
<code>__rhs</code>	Last string.

Returns

New string with value of `__lhs` followed by `__rhs`.

3.1.2.9 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string<_CharT, _Traits, _Alloc, _Base> __gnu_cxx::operator+ (_CharT __lhs, const __versa_string<_CharT, _Traits, _Alloc, _Base > & __rhs)`

Concatenate character and string.

Parameters

<code>__lhs</code>	First string.
<code>__rhs</code>	Last string.

Returns

New string with `__lhs` followed by `__rhs`.

3.1.2.10 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string<_CharT, _Traits, _Alloc, _Base> __gnu_cxx::operator+ (const __versa_string<_CharT, _Traits, _Alloc, _Base > & __lhs, const _CharT * __rhs)`

Concatenate string and C string.

Parameters

<code>__lhs</code>	First string.
<code>__rhs</code>	Last string.

Returns

New string with `__lhs` followed by `__rhs`.

3.1.2.11 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string<_CharT, _Traits, _Alloc, _Base> __gnu_cxx::operator+ (const __versa_string<_CharT, _Traits, _Alloc, _Base> & __lhs, _CharT __rhs)`

Concatenate string and character.

Parameters

<code>__lhs</code>	First string.
<code>__rhs</code>	Last string.

Returns

New string with `__lhs` followed by `__rhs`.

3.1.2.12 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool __gnu_cxx::operator< (const __versa_string<_CharT, _Traits, _Alloc, _Base> & __lhs, const __versa_string<_CharT, _Traits, _Alloc, _Base> & __rhs) [inline]`

Test if string precedes string.

Parameters

<code>__lhs</code>	First string.
<code>__rhs</code>	Second string.

Returns

True if `__lhs` precedes `__rhs`. False otherwise.

Definition at line 2428 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::compare()`.

3.1.2.13 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool __gnu_cxx::operator< (const __versa_string<_CharT, _Traits, _Alloc, _Base> & __lhs, const _CharT * __rhs) [inline]`

Test if string precedes C string.

Parameters

<code>__lhs</code>	String.
<code>__rhs</code>	C string.

Returns

True if `__lhs` precedes `__rhs`. False otherwise.

Definition at line 2441 of file `vstring.h`.

```
3.1.2.14 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
_Base> bool __gnu_cxx::operator< ( const _CharT* __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &
__rhs ) [inline]
```

Test if C string precedes string.

Parameters

<code>__lhs</code>	C string.
<code>__rhs</code>	String.

Returns

True if `__lhs` precedes `__rhs`. False otherwise.

Definition at line 2454 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare()`.

```
3.1.2.15 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
_Base> bool __gnu_cxx::operator<= ( const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs, const
__versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs ) [inline]
```

Test if string doesn't follow string.

Parameters

<code>__lhs</code>	First string.
<code>__rhs</code>	Second string.

Returns

True if `__lhs` doesn't follow `__rhs`. False otherwise.

Definition at line 2508 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare()`.

```
3.1.2.16 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
_Base> bool __gnu_cxx::operator<= ( const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs, const _CharT
* __rhs ) [inline]
```

Test if string doesn't follow C string.

Parameters

<code>__lhs</code>	String.
<code>__rhs</code>	C string.

Returns

True if `__lhs` doesn't follow `__rhs`. False otherwise.

Definition at line 2521 of file `vstring.h`.

```
3.1.2.17 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
_Base> bool __gnu_cxx::operator<= ( const _CharT * __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base >
& __rhs ) [inline]
```

Test if C string doesn't follow string.

Parameters

<code>__lhs</code>	C string.
<code>__rhs</code>	String.

Returns

True if `__lhs` doesn't follow `__rhs`. False otherwise.

Definition at line 2534 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare()`.

```
3.1.2.18 template<typename _Tp > bool __gnu_cxx::operator==( const _Pointer_adapter< _Tp > & __lhs, const
_PPointer_adapter< _Tp > & __rhs ) [inline]
```

Comparison operators for `_Pointer_adapter` defer to the base class' comparison operators, when possible.

Definition at line 529 of file `pointer.h`.

References `std::addressof()`.

```
3.1.2.19 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
_Base> bool __gnu_cxx::operator==( const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs, const
__versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs ) [inline]
```

Test equivalence of two strings.

Parameters

<code>__lhs</code>	First string.
<code>__rhs</code>	Second string.

Returns

True if `__lhs.compare(__rhs) == 0`. False otherwise.

Definition at line 2337 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::compare()`.

```
3.1.2.20  template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
           _Base> bool __gnu_cxx::operator==( const _CharT * __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &
           __rhs ) [inline]
```

Test equivalence of C string and string.

Parameters

<code>__lhs</code>	C string.
<code>__rhs</code>	String.

Returns

True if `__rhs.compare(__lhs) == 0`. False otherwise.

Definition at line 2361 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::compare()`.

```
3.1.2.21  template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
           _Base> bool __gnu_cxx::operator==( const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs, const _CharT *
           __rhs ) [inline]
```

Test equivalence of string and C string.

Parameters

<code>__lhs</code>	String.
<code>__rhs</code>	C string.

Returns

True if `__lhs.compare(__rhs) == 0`. False otherwise.

Definition at line 2374 of file vstring.h.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::compare()`.

3.1.2.22 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool __gnu_cxx::operator> (const __versa_string<_CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string<_CharT, _Traits, _Alloc, _Base > &__rhs) [inline]`

Test if string follows string.

Parameters

<code>__lhs</code>	First string.
<code>__rhs</code>	Second string.

Returns

True if `__lhs` follows `__rhs`. False otherwise.

Definition at line 2468 of file vstring.h.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::compare()`.

3.1.2.23 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool __gnu_cxx::operator> (const __versa_string<_CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT * __rhs) [inline]`

Test if string follows C string.

Parameters

<code>__lhs</code>	String.
<code>__rhs</code>	C string.

Returns

True if `__lhs` follows `__rhs`. False otherwise.

Definition at line 2481 of file vstring.h.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::compare()`.

3.1.2.24 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool __gnu_cxx::operator> (const _CharT * __lhs, const __versa_string<_CharT, _Traits, _Alloc, _Base > & __rhs) [inline]`

Test if C string follows string.

Parameters

<code>__lhs</code>	C string.
<code>__rhs</code>	String.

Returns

True if `__lhs` follows `__rhs`. False otherwise.

Definition at line 2494 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::compare()`.

```
3.1.2.25  template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
           _Base> bool __gnu_cxx::operator>= ( const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs, const
           __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs ) [inline]
```

Test if string doesn't precede string.

Parameters

<code>__lhs</code>	First string.
<code>__rhs</code>	Second string.

Returns

True if `__lhs` doesn't precede `__rhs`. False otherwise.

Definition at line 2548 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::compare()`.

```
3.1.2.26  template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
           _Base> bool __gnu_cxx::operator>= ( const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs, const _CharT
           * __rhs ) [inline]
```

Test if string doesn't precede C string.

Parameters

<code>__lhs</code>	String.
<code>__rhs</code>	C string.

Returns

True if `__lhs` doesn't precede `__rhs`. False otherwise.

Definition at line 2561 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::compare()`.

```
3.1.2.27 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
_Base> bool __gnu_cxx::operator>= ( const _CharT * __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base >
& __rhs ) [inline]
```

Test if C string doesn't precede string.

Parameters

<code>__lhs</code>	C string.
<code>__rhs</code>	String.

Returns

True if `__lhs` doesn't precede `__rhs`. False otherwise.

Definition at line 2574 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::compare()`.

```
3.1.2.28 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
_Base> void __gnu_cxx::swap ( __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs, __versa_string<
_CharT, _Traits, _Alloc, _Base > & __rhs ) [inline]
```

Swap contents of two strings.

Parameters

<code>__lhs</code>	First string.
<code>__rhs</code>	Second string.

Exchanges the contents of `__lhs` and `__rhs` in constant time.

Definition at line 2588 of file `vstring.h`.

References `std::operator>>()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::swap()`.

3.2 `__gnu_cxx::__detail` Namespace Reference

Classes

- class [__mini_vector](#)
- class [_Bitmap_counter](#)
- class [_Ffit_finder](#)

Enumerations

- enum { **bits_per_byte**, **bits_per_block** }

Functions

- void **__bit_allocate** (size_t * __pbitmap, size_t __pos) throw ()
- void **__bit_free** (size_t * __pbitmap, size_t __pos) throw ()
- template<typename _ForwardIterator, typename _Tp, typename _Compare >
_ForwardIterator **__lower_bound** (_ForwardIterator __first, _ForwardIterator __last, const _Tp & __val, _Compare __comp)
- template<typename _AddrPair >
size_t **__num_bitmaps** (_AddrPair __ap)
- template<typename _AddrPair >
size_t **__num_blocks** (_AddrPair __ap)

3.2.1 Detailed Description

Implementation details not part of the namespace `__gnu_cxx` interface.

3.2.2 Function Documentation

3.2.2.1 void __gnu_cxx::__detail::__bit_allocate (size_t * __pbitmap, size_t __pos) throw () `[inline]`

Mark a memory address as allocated by re-setting the corresponding bit in the bit-map.

Definition at line 488 of file `bitmap_allocator.h`.

Referenced by `__gnu_cxx::bitmap_allocator<_Tp>::_M_allocate_single_object()`.

3.2.2.2 void __gnu_cxx::__detail::__bit_free (size_t * __pbitmap, size_t __pos) throw () `[inline]`

Mark a memory address as free by setting the corresponding bit in the bit-map.

Definition at line 499 of file `bitmap_allocator.h`.

Referenced by `__gnu_cxx::bitmap_allocator<_Tp>::_M_deallocate_single_object()`.

3.2.2.3 template<typename _AddrPair > size_t __gnu_cxx::__detail::__num_bitmaps (_AddrPair __ap) `[inline]`

The number of Bit-maps pointed to by the address pair passed to the function.

Definition at line 276 of file `bitmap_allocator.h`.

References `__num_blocks()`.

Referenced by `__gnu_cxx::bitmap_allocator<_Tp>::_M_allocate_single_object()`, `__gnu_cxx::bitmap_allocator<_Tp>::_M_deallocate_single_object()`, and `__gnu_cxx::free_list::_M_insert()`.

3.2.2.4 `template<typename _AddrPair> size_t __gnu_cxx::__detail::__num_blocks (_AddrPair __ap) [inline]`

The number of Blocks pointed to by the address pair passed to the function.

Definition at line 268 of file `bitmap_allocator.h`.

Referenced by `__num_bitmaps()`.

3.3 `__gnu_cxx::typelist` Namespace Reference

Functions

- `template<typename Fn , typename Typelist>`
`void apply (Fn &, Typelist)`
- `template<typename Gn , typename Typelist>`
`void apply_generator (Gn &, Typelist)`
- `template<typename Gn , typename TypelistT , typename TypelistV>`
`void apply_generator (Gn &, TypelistT, TypelistV)`
- `template<typename Fn , typename Typelist>`
`void apply_generator (Fn &fn, Typelist)`
- `template<typename Fn , typename TypelistT , typename TypelistV>`
`void apply_generator (Fn &fn, TypelistT, TypelistV)`

3.3.1 Detailed Description

GNU typelist extensions for public compile-time use.

3.3.2 Function Documentation

3.3.2.1 `template<typename Gn , typename Typelist> void __gnu_cxx::typelist::apply_generator (Gn & , Typelist)`

Apply all typelist types to generator functor.

3.4 `__gnu_debug` Namespace Reference

Classes

- class `_After_nth_from`
- struct `_BeforeBeginHelper`
- class `_Equal_to`
- class `_Not_equal_to`
- class `_Safe_iterator`
- class `_Safe_iterator_base`
- class `_Safe_local_iterator`
- class `_Safe_local_iterator_base`
- class `_Safe_sequence`
- class `_Safe_sequence_base`
- class `_Safe_unordered_container`
- class `_Safe_unordered_container_base`

Enumerations

- enum `_Debug_msg_id` {
`__msg_valid_range`, `__msg_insert_singular`, `__msg_insert_different`, `__msg_erase_bad`,
`__msg_erase_different`, `__msg_subscript_oob`, `__msg_empty`, `__msg_unpartitioned`,
`__msg_unpartitioned_pred`, `__msg_unsorted`, `__msg_unsorted_pred`, `__msg_not_heap`,
`__msg_not_heap_pred`, `__msg_bad_bitset_write`, `__msg_bad_bitset_read`, `__msg_bad_bitset_flip`,
`__msg_self_splice`, `__msg_splice_alloc`, `__msg_splice_bad`, `__msg_splice_other`,
`__msg_splice_overlap`, `__msg_init_singular`, `__msg_init_copy_singular`, `__msg_init_const_singular`,
`__msg_copy_singular`, `__msg_bad_deref`, `__msg_bad_inc`, `__msg_bad_dec`,
`__msg_iter_subscript_oob`, `__msg_advance_oob`, `__msg_retreat_oob`, `__msg_iter_compare_bad`,
`__msg_compare_different`, `__msg_iter_order_bad`, `__msg_order_different`, `__msg_distance_bad`,
`__msg_distance_different`, `__msg_deref_istream`, `__msg_inc_istream`, `__msg_output_ostream`,
`__msg_deref_istreambuf`, `__msg_inc_istreambuf`, `__msg_insert_after_end`, `__msg_erase_after_bad`,
`__msg_valid_range2`, `__msg_local_iter_compare_bad`, `__msg_non_empty_range`, `__msg_self_move`↵
`assign`,
`__msg_bucket_index_oob`, `__msg_valid_load_factor`, `__msg_equal_allocs`, `__msg_insert_range_from`↵
`_self` }
- enum `_Distance_precision` { `__dp_equality`, `__dp_sign`, `__dp_exact` }

Functions

- template<typename `_Iterator` >
`_Siter_base`< `_Iterator` >::iterator_type `__base` (`_Iterator` __it)
- template<typename `_Iterator` >
bool `__check_dereferenceable` (const `_Iterator` &)
- template<typename `_Tp` >
bool `__check_dereferenceable` (const `_Tp` * __ptr)
- template<typename `_Iterator` , typename `_Sequence` >
bool `__check_dereferenceable` (const `_Safe_iterator`< `_Iterator`, `_Sequence` > & __x)
- template<typename `_Iterator` , typename `_Sequence` >
bool `__check_dereferenceable` (const `_Safe_local_iterator`< `_Iterator`, `_Sequence` > & __x)
- template<typename `_ForwardIterator` , typename `_Tp` >
bool `__check_partitioned_lower` (`_ForwardIterator` __first, `_ForwardIterator` __last, const `_Tp` & __value)
- template<typename `_ForwardIterator` , typename `_Tp` , typename `_Pred` >
bool `__check_partitioned_lower` (`_ForwardIterator` __first, `_ForwardIterator` __last, const `_Tp` & __value, `_Pred` __pred)
- template<typename `_ForwardIterator` , typename `_Tp` >
bool `__check_partitioned_upper` (`_ForwardIterator` __first, `_ForwardIterator` __last, const `_Tp` & __value)
- template<typename `_ForwardIterator` , typename `_Tp` , typename `_Pred` >
bool `__check_partitioned_upper` (`_ForwardIterator` __first, `_ForwardIterator` __last, const `_Tp` & __value, `_Pred` __pred)
- template<typename `_Iterator` >
bool `__check_singular` (const `_Iterator` &)
- template<typename `_Tp` >
bool `__check_singular` (const `_Tp` * __ptr)
- bool `__check_singular_aux` (const void *)
- bool `__check_singular_aux` (const `_Safe_iterator_base` * __x)
- template<typename `_InputIterator` >
bool `__check_sorted` (const `_InputIterator` & __first, const `_InputIterator` & __last)
- template<typename `_InputIterator` , typename `_Predicate` >
bool `__check_sorted` (const `_InputIterator` & __first, const `_InputIterator` & __last, `_Predicate` __pred)

- `template<typename _InputIterator >`
`bool __check_sorted_aux (const _InputIterator &, const _InputIterator &, std::input_iterator_tag)`
- `template<typename _ForwardIterator >`
`bool __check_sorted_aux (_ForwardIterator __first, _ForwardIterator __last, std::forward_iterator_tag)`
- `template<typename _InputIterator, typename _Predicate >`
`bool __check_sorted_aux (const _InputIterator &, const _InputIterator &, _Predicate, std::input_iterator_tag)`
- `template<typename _ForwardIterator, typename _Predicate >`
`bool __check_sorted_aux (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, std::forward_iterator_tag)`
- `template<typename _InputIterator1, typename _InputIterator2 >`
`bool __check_sorted_set (const _InputIterator1 & __first, const _InputIterator1 & __last, const _InputIterator2 &)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Predicate >`
`bool __check_sorted_set (const _InputIterator1 & __first, const _InputIterator1 & __last, const _InputIterator2 &, _Predicate __pred)`
- `template<typename _InputIterator >`
`bool __check_sorted_set_aux (const _InputIterator & __first, const _InputIterator & __last, std::true_type)`
- `template<typename _InputIterator >`
`bool __check_sorted_set_aux (const _InputIterator &, const _InputIterator &, std::false_type)`
- `template<typename _InputIterator, typename _Predicate >`
`bool __check_sorted_set_aux (const _InputIterator & __first, const _InputIterator & __last, _Predicate __pred, std::true_type)`
- `template<typename _InputIterator, typename _Predicate >`
`bool __check_sorted_set_aux (const _InputIterator &, const _InputIterator &, _Predicate, std::false_type)`
- `template<typename _CharT, typename _Integer >`
`const _CharT * __check_string (const _CharT * __s, const _Integer & __n __attribute__((__unused__)))`
- `template<typename _CharT >`
`const _CharT * __check_string (const _CharT * __s)`
- `template<typename _InputIterator >`
`_InputIterator __check_valid_range (const _InputIterator & __first, const _InputIterator & __last __attribute__((__unused__)))`
- `template<typename _Iterator, typename _Sequence, typename _InputIterator >`
`bool __foreign_iterator (const _Safe_iterator< _Iterator, _Sequence > & __it, _InputIterator __other, _InputIterator __other_end)`
- `template<typename _Iterator, typename _Sequence, typename _Integral >`
`bool __foreign_iterator_aux (const _Safe_iterator< _Iterator, _Sequence > &, _Integral, _Integral, std::true_type)`
- `template<typename _Iterator, typename _Sequence, typename _InputIterator >`
`bool __foreign_iterator_aux (const _Safe_iterator< _Iterator, _Sequence > & __it, _InputIterator __other, _InputIterator __other_end, std::false_type)`
- `template<typename _Iterator, typename _Sequence, typename _OtherIterator >`
`bool __foreign_iterator_aux2 (const _Safe_iterator< _Iterator, _Sequence > & __it, const _Safe_iterator< _OtherIterator, _Sequence > & __other, const _Safe_iterator< _OtherIterator, _Sequence > &)`
- `template<typename _Iterator, typename _Sequence, typename _OtherIterator, typename _OtherSequence >`
`bool __foreign_iterator_aux2 (const _Safe_iterator< _Iterator, _Sequence > & __it, const _Safe_iterator< _OtherIterator, _OtherSequence > &, const _Safe_iterator< _OtherIterator, _OtherSequence > &)`
- `template<typename _Iterator, typename _Sequence, typename _InputIterator >`
`bool __foreign_iterator_aux2 (const _Safe_iterator< _Iterator, _Sequence > & __it, const _InputIterator & __other, const _InputIterator & __other_end)`
- `template<typename _Iterator, typename _Sequence, typename _InputIterator >`
`bool __foreign_iterator_aux3 (const _Safe_iterator< _Iterator, _Sequence > & __it, const _InputIterator & __other, const _InputIterator & __other_end, std::true_type)`

- `template<typename _Iterator, typename _Sequence, typename _InputIterator >`
`bool __foreign_iterator_aux3 (const __Safe_iterator< _Iterator, _Sequence > &, const _InputIterator &, const`
`_InputIterator &, std::__false_type)`
- `template<typename _Iterator, typename _Sequence >`
`bool __foreign_iterator_aux4 (const __Safe_iterator< _Iterator, _Sequence > &__it, const typename _↵`
`Sequence::value_type *__other)`
- `template<typename _Iterator, typename _Sequence >`
`bool __foreign_iterator_aux4 (const __Safe_iterator< _Iterator, _Sequence > &,...)`
- `template<typename _Iterator >`
`std::pair< typename std::iterator_traits< _Iterator >::difference_type, _Distance_precision > __get_distance`
`(const _Iterator &__lhs, const _Iterator &__rhs, std::random_access_iterator_tag)`
- `template<typename _Iterator >`
`std::pair< typename std::iterator_traits< _Iterator >::difference_type, _Distance_precision > __get_distance`
`(const _Iterator &__lhs, const _Iterator &__rhs, std::forward_iterator_tag)`
- `template<typename _Iterator >`
`std::pair< typename std::iterator_traits< _Iterator >::difference_type, _Distance_precision > __get_distance`
`(const _Iterator &__lhs, const _Iterator &__rhs)`
- `template<typename _InputIterator >`
`bool __valid_range (const _InputIterator &__first, const _InputIterator &__last)`
- `template<typename _Iterator, typename _Sequence >`
`bool __valid_range (const __Safe_iterator< _Iterator, _Sequence > &__first, const __Safe_iterator< _Iterator, _↵`
`Sequence > &__last)`
- `template<typename _Iterator, typename _Sequence >`
`bool __valid_range (const __Safe_local_iterator< _Iterator, _Sequence > &__first, const __Safe_local_iterator<`
`_Iterator, _Sequence > &__last)`
- `template<typename _Integral >`
`bool __valid_range_aux (const _Integral &, const _Integral &, std::__true_type)`
- `template<typename _InputIterator >`
`bool __valid_range_aux (const _InputIterator &__first, const _InputIterator &__last, std::__false_type)`
- `template<typename _RandomAccessIterator >`
`bool __valid_range_aux2 (const _RandomAccessIterator &__first, const _RandomAccessIterator &__last, std:↵`
`::random_access_iterator_tag)`
- `template<typename _InputIterator >`
`bool __valid_range_aux2 (const _InputIterator &, const _InputIterator &, std::input_iterator_tag)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`
`bool operator!= (const __Safe_local_iterator< _IteratorL, _Sequence > &__lhs, const __Safe_local_iterator< _↵`
`IteratorR, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`
`bool operator!= (const __Safe_local_iterator< _Iterator, _Sequence > &__lhs, const __Safe_local_iterator< _↵`
`Iterator, _Sequence > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`
`bool operator!= (const __Safe_iterator< _IteratorL, _Sequence > &__lhs, const __Safe_iterator< _IteratorR, _↵`
`Sequence > &__rhs) noexcept`
- `template<typename _Iterator, typename _Sequence >`
`bool operator!= (const __Safe_iterator< _Iterator, _Sequence > &__lhs, const __Safe_iterator< _Iterator, _↵`
`Sequence > &__rhs) noexcept`
- `template<typename _Iterator, typename _Sequence >`
`__Safe_iterator< _Iterator, _Sequence > operator+ (typename __Safe_iterator< _Iterator, _Sequence >↵`
`::difference_type __n, const __Safe_iterator< _Iterator, _Sequence > &__i) noexcept`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`
`__Safe_iterator< _IteratorL, _Sequence >::difference_type operator- (const __Safe_iterator< _IteratorL, _↵`
`Sequence > &__lhs, const __Safe_iterator< _IteratorR, _Sequence > &__rhs) noexcept`

- `template<typename _Iterator, typename _Sequence >`
`_Safe_iterator< _Iterator, _Sequence >::difference_type operator- (const _Safe_iterator< _Iterator, _Sequence`
`> &__lhs, const _Safe_iterator< _Iterator, _Sequence > &__rhs) noexcept`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`
`bool operator< (const _Safe_iterator< _IteratorL, _Sequence > &__lhs, const _Safe_iterator< _IteratorR, _`
`Sequence > &__rhs) noexcept`
- `template<typename _Iterator, typename _Sequence >`
`bool operator< (const _Safe_iterator< _Iterator, _Sequence > &__lhs, const _Safe_iterator< _Iterator, _`
`Sequence > &__rhs) noexcept`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`
`bool operator<= (const _Safe_iterator< _IteratorL, _Sequence > &__lhs, const _Safe_iterator< _IteratorR, _`
`Sequence > &__rhs) noexcept`
- `template<typename _Iterator, typename _Sequence >`
`bool operator<= (const _Safe_iterator< _Iterator, _Sequence > &__lhs, const _Safe_iterator< _Iterator, _`
`Sequence > &__rhs) noexcept`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`
`bool operator== (const _Safe_local_iterator< _IteratorL, _Sequence > &__lhs, const _Safe_local_iterator< _`
`IteratorR, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`
`bool operator== (const _Safe_local_iterator< _Iterator, _Sequence > &__lhs, const _Safe_local_iterator< _`
`Iterator, _Sequence > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`
`bool operator== (const _Safe_iterator< _IteratorL, _Sequence > &__lhs, const _Safe_iterator< _IteratorR, _`
`Sequence > &__rhs) noexcept`
- `template<typename _Iterator, typename _Sequence >`
`bool operator== (const _Safe_iterator< _Iterator, _Sequence > &__lhs, const _Safe_iterator< _Iterator, _`
`Sequence > &__rhs) noexcept`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`
`bool operator> (const _Safe_iterator< _IteratorL, _Sequence > &__lhs, const _Safe_iterator< _IteratorR, _`
`Sequence > &__rhs) noexcept`
- `template<typename _Iterator, typename _Sequence >`
`bool operator> (const _Safe_iterator< _Iterator, _Sequence > &__lhs, const _Safe_iterator< _Iterator, _`
`Sequence > &__rhs) noexcept`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`
`bool operator>= (const _Safe_iterator< _IteratorL, _Sequence > &__lhs, const _Safe_iterator< _IteratorR, _`
`Sequence > &__rhs) noexcept`
- `template<typename _Iterator, typename _Sequence >`
`bool operator>= (const _Safe_iterator< _Iterator, _Sequence > &__lhs, const _Safe_iterator< _Iterator, _`
`Sequence > &__rhs) noexcept`

3.4.1 Detailed Description

GNU debug classes for public use.

3.4.2 Enumeration Type Documentation

3.4.2.1 enum __gnu_debug::Distance_precision

The precision to which we can calculate the distance between two iterators.

Definition at line 68 of file `safe_iterator.h`.

3.4.3 Function Documentation

3.4.3.1 `template<typename _Iterator > _Siter_base<_Iterator>::iterator_type __gnu_debug::__base (_Iterator __it)`
`[inline]`

Helper function to extract base iterator of random access safe iterator in order to reduce performance impact of debug mode. Limited to random access iterator because it is the only category for which it is possible to check for correct iterators order in the `__valid_range` function thanks to the `<` operator.

Definition at line 558 of file `functions.h`.

Referenced by `__gnu_parallel::__for_each_template_random_access_workstealing()`, `__gnu_debug::_Safe_iterator<_Iterator, _Sequence >::__M_before_dereferenceable()`, `std::boolalpha()`, `std::dec()`, `std::fixed()`, `std::getline()`, `std::hex()`, `std::internal()`, `std::left()`, `std::noboolalpha()`, `std::noshowbase()`, `std::noshowpoint()`, `std::noshowpos()`, `std::noskipws()`, `std::nounitbuf()`, `std::nouppercase()`, `std::oct()`, `std::locale::operator!=()`, `std::right()`, `std::scientific()`, `std::showbase()`, `std::showpoint()`, `std::showpos()`, `std::skipws()`, `std::unitbuf()`, and `std::uppercase()`.

3.4.3.2 `template<typename _Iterator > bool __gnu_debug::__check_dereferenceable (const _Iterator &)` `[inline]`

Assume that some arbitrary iterator is dereferenceable, because we can't prove that it isn't.

Definition at line 79 of file `functions.h`.

3.4.3.3 `template<typename _Tp > bool __gnu_debug::__check_dereferenceable (const _Tp * __ptr)` `[inline]`

Non-NULL pointers are dereferenceable.

Definition at line 85 of file `functions.h`.

3.4.3.4 `template<typename _Iterator, typename _Sequence > bool __gnu_debug::__check_dereferenceable (const _Safe_iterator<_Iterator, _Sequence > & __x)` `[inline]`

Safe iterators know if they are dereferenceable.

Definition at line 91 of file `functions.h`.

References `__gnu_debug::_Safe_iterator<_Iterator, _Sequence >::__M_dereferenceable()`.

3.4.3.5 `template<typename _Iterator, typename _Sequence > bool __gnu_debug::__check_dereferenceable (const _Safe_local_iterator<_Iterator, _Sequence > & __x)` `[inline]`

Safe local iterators know if they are dereferenceable.

Definition at line 97 of file `functions.h`.

3.4.3.6 `template<typename _Tp > bool __gnu_debug::__check_singular (const _Tp * __ptr)` `[inline]`

Non-NULL pointers are nonsingular.

Definition at line 72 of file `functions.h`.

3.4.3.7 `bool __gnu_debug::__check_singular_aux (const _Safe_iterator_base * __x) [inline]`

Iterators that derive from `_Safe_iterator_base` can be determined singular or non-singular.

Definition at line 62 of file `safe_iterator.h`.

References `__gnu_debug::_Safe_iterator_base::M_singular()`.

3.4.3.8 `template<typename _CharT, typename _Integer> const _CharT* __gnu_debug::__check_string (const _CharT * __s, const _Integer & __n __attribute__((unused))) [inline]`

Checks that `__s` is non-NULL or `__n == 0`, and then returns `__s`.

Definition at line 300 of file `functions.h`.

3.4.3.9 `template<typename _CharT> const _CharT* __gnu_debug::__check_string (const _CharT * __s) [inline]`

Checks that `__s` is non-NULL and then returns `__s`.

Definition at line 312 of file `functions.h`.

References `std::__iterator_category()`.

3.4.3.10 `template<typename _Iterator, typename _Sequence, typename _OtherIterator> bool __gnu_debug::__foreign_iterator_↵
aux2 (const _Safe_iterator< _Iterator, _Sequence> & __it, const _Safe_iterator< _OtherIterator, _Sequence> &
__other, const _Safe_iterator< _OtherIterator, _Sequence> &) [inline]`

Handle debug iterators from the same type of container.

Definition at line 234 of file `functions.h`.

Referenced by `__foreign_iterator_aux2()`.

3.4.3.11 `template<typename _Iterator, typename _Sequence, typename _OtherIterator, typename _OtherSequence> bool
__gnu_debug::__foreign_iterator_aux2 (const _Safe_iterator< _Iterator, _Sequence> & __it, const _Safe_iterator<
_OtherIterator, _OtherSequence> &, const _Safe_iterator< _OtherIterator, _OtherSequence> &) [inline]`

Handle debug iterators from different types of container.

Definition at line 243 of file `functions.h`.

References `__foreign_iterator_aux2()`.

3.4.3.12 `template<typename _Iterator> std::pair<typename std::iterator_traits<_Iterator>::difference_type,
_Distance_precision> __gnu_debug::__get_distance (const _Iterator & __lhs, const _Iterator & __rhs,
std::random_access_iterator_tag) [inline]`

Determine the distance between two iterators with some known precision.

Definition at line 81 of file `safe_iterator.h`.

References `std::make_pair()`.

```
3.4.3.13 template<typename _InputIterator > bool __gnu_debug::__valid_range ( const _InputIterator & __first, const _InputIterator
& __last ) [inline]
```

Don't know what these iterators are, or if they are even iterators (we may get an integral type for `_InputIterator`), so see if they are integral and pass them on to the next phase otherwise.

Definition at line 147 of file `functions.h`.

References `__valid_range_aux()`.

```
3.4.3.14 template<typename _Iterator , typename _Sequence > bool __gnu_debug::__valid_range ( const _Safe_iterator<
_Iterator, _Sequence > & __first, const _Safe_iterator< _Iterator, _Sequence > & __last ) [inline]
```

Safe iterators know how to check if they form a valid range.

Definition at line 156 of file `functions.h`.

```
3.4.3.15 template<typename _Iterator , typename _Sequence > bool __gnu_debug::__valid_range ( const
_Safe_local_iterator< _Iterator, _Sequence > & __first, const _Safe_local_iterator< _Iterator, _Sequence > &
__last ) [inline]
```

Safe local iterators know how to check if they form a valid range.

Definition at line 163 of file `functions.h`.

References `std::__addressof()`.

```
3.4.3.16 template<typename _Integral > bool __gnu_debug::__valid_range_aux ( const _Integral & , const _Integral & ,
std::__true_type ) [inline]
```

We say that integral types for a valid range, and defer to other routines to realize what to do with integral types instead of iterators.

Definition at line 127 of file `functions.h`.

Referenced by `__valid_range()`.

```
3.4.3.17 template<typename _InputIterator > bool __gnu_debug::__valid_range_aux ( const _InputIterator & __first, const
_InputIterator & __last, std::__false_type ) [inline]
```

We have iterators, so figure out what kind of iterators that are to see if we can check the range ahead of time.

Definition at line 135 of file `functions.h`.

References `std::__iterator_category()`, and `__valid_range_aux2()`.

```
3.4.3.18 template<typename _RandomAccessIterator > bool __gnu_debug::__valid_range_aux2 ( const _RandomAccessIterator &
__first, const _RandomAccessIterator & __last, std::random_access_iterator_tag ) [inline]
```

If the distance between two random access iterators is nonnegative, assume the range is valid.

Definition at line 106 of file `functions.h`.

Referenced by `__valid_range_aux()`.

```
3.4.3.19 template<typename _InputIterator > bool __gnu_debug::__valid_range_aux2( const _InputIterator &, const _InputIterator
&, std::input_iterator_tag ) [inline]
```

Can't test for a valid range with input iterators, because iteration may be destructive. So we just assume that the range is valid.

Definition at line 117 of file functions.h.

3.5 `__gnu_internal` Namespace Reference

3.5.1 Detailed Description

GNU implementation details, not for public use or export. Used only when anonymous namespaces cannot be substituted.

3.6 `__gnu_parallel` Namespace Reference

Classes

- struct [__accumulate_binop_reduct](#)
- struct [__accumulate_selector](#)
- struct [__adjacent_difference_selector](#)
- struct [__adjacent_find_selector](#)
- class [__binder1st](#)
- class [__binder2nd](#)
- struct [__count_if_selector](#)
- struct [__count_selector](#)
- struct [__fill_selector](#)
- struct [__find_first_of_selector](#)
- struct [__find_if_selector](#)
- struct [__for_each_selector](#)
- struct [__generate_selector](#)
- struct [__generic_find_selector](#)
- struct [__generic_for_each_selector](#)
- struct [__identity_selector](#)
- struct [__inner_product_selector](#)
- struct [__max_element_reduct](#)
- struct [__min_element_reduct](#)
- struct [__mismatch_selector](#)
- struct [__multiway_merge_3_variant_sentinel_switch](#)
- struct [__multiway_merge_3_variant_sentinel_switch< true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >](#)
- struct [__multiway_merge_4_variant_sentinel_switch](#)
- struct [__multiway_merge_4_variant_sentinel_switch< true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >](#)
- struct [__multiway_merge_k_variant_sentinel_switch](#)
- struct [__multiway_merge_k_variant_sentinel_switch< false, __stable, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >](#)

- struct [__replace_if_selector](#)
- struct [__replace_selector](#)
- struct [__transform1_selector](#)
- struct [__transform2_selector](#)
- class [__unary_negate](#)
- struct [_DRandomShufflingGlobalData](#)
- struct [_DRSSorterPU](#)
- struct [_DummyReduct](#)
- class [_EqualFromLess](#)
- struct [_EqualTo](#)
- class [_GuardedIterator](#)
- class [_IteratorPair](#)
- class [_IteratorTriple](#)
- struct [_Job](#)
- struct [_Less](#)
- class [_Lexicographic](#)
- class [_LexicographicReverse](#)
- class [_LoserTree](#)
- class [_LoserTree< false, _Tp, _Compare >](#)
- class [_LoserTreeBase](#)
- class [_LoserTreePointer](#)
- class [_LoserTreePointer< false, _Tp, _Compare >](#)
- class [_LoserTreePointerBase](#)
- class [_LoserTreePointerUnguarded](#)
- class [_LoserTreePointerUnguarded< false, _Tp, _Compare >](#)
- class [_LoserTreePointerUnguardedBase](#)
- struct [_LoserTreeTraits](#)
- class [_LoserTreeUnguarded](#)
- class [_LoserTreeUnguarded< false, _Tp, _Compare >](#)
- class [_LoserTreeUnguardedBase](#)
- struct [_Multiplies](#)
- struct [_Nothing](#)
- struct [_Piece](#)
- struct [_Plus](#)
- struct [_PMWMSSortingData](#)
- class [_PseudoSequence](#)
- class [_PseudoSequenceIterator](#)
- struct [_QSBThreadLocal](#)
- class [_RandomNumber](#)
- class [_RestrictedBoundedConcurrentQueue](#)
- struct [_SamplingSorter](#)
- struct [_SamplingSorter< false, _RAIter, _StrictWeakOrdering >](#)
- struct [_Settings](#)
- struct [_SplitConsistently](#)
- struct [_SplitConsistently< false, _RAIter, _Compare, _SortingPlacesIterator >](#)
- struct [_SplitConsistently< true, _RAIter, _Compare, _SortingPlacesIterator >](#)
- struct [balanced_quicksort_tag](#)
- struct [balanced_tag](#)
- struct [constant_size_blocks_tag](#)
- struct [default_parallel_tag](#)
- struct [equal_split_tag](#)

- struct [exact_tag](#)
- struct [find_tag](#)
- struct [growing_blocks_tag](#)
- struct [multiway_mergesort_exact_tag](#)
- struct [multiway_mergesort_sampling_tag](#)
- struct [multiway_mergesort_tag](#)
- struct [omp_loop_static_tag](#)
- struct [omp_loop_tag](#)
- struct [parallel_tag](#)
- struct [quicksort_tag](#)
- struct [sampling_tag](#)
- struct [sequential_tag](#)
- struct [unbalanced_tag](#)

Typedefs

- typedef unsigned short [_BinIndex](#)
- typedef int64_t [_CASable](#)
- typedef uint64_t [_SequenceIndex](#)
- typedef uint16_t [_ThreadIndex](#)

Enumerations

- enum [_AlgorithmStrategy](#) { **heuristic**, **force_sequential**, **force_parallel** }
- enum [_FindAlgorithm](#) { **GROWING_BLOCKS**, **CONSTANT_SIZE_BLOCKS**, **EQUAL_SPLIT** }
- enum [_MultiwayMergeAlgorithm](#) { **LOSER_TREE** }
- enum [_Parallelism](#) { [sequential](#), [parallel_unbalanced](#), [parallel_balanced](#), [parallel_omp_loop](#), [parallel_omp_loop_static](#), [parallel_taskqueue](#) }
- enum [_PartialSumAlgorithm](#) { **RECURSIVE**, **LINEAR** }
- enum [_SortAlgorithm](#) { **MWMS**, **QS**, **QS_BALANCED** }
- enum [_SplittingAlgorithm](#) { **SAMPLING**, **EXACT** }

Functions

- template<typename _Tp >
 _Tp **__add_omp** (volatile _Tp *__ptr, _Tp __addend)
- template<typename _RAIter, typename _DifferenceTp >
 void **__calc_borders** (_RAIter __elements, _DifferenceTp __length, _DifferenceTp *__off)
- template<typename _Tp >
 bool **__cas_omp** (volatile _Tp *__ptr, _Tp __comparand, _Tp __replacement)
- template<typename _Tp >
 bool **__compare_and_swap** (volatile _Tp *__ptr, _Tp __comparand, _Tp __replacement)
- template<typename _Iter, typename _OutputIterator >
 _OutputIterator **__copy_tail** (std::pair<_Iter, _Iter> __b, std::pair<_Iter, _Iter> __e, _OutputIterator __r)
- void **__decode2** (_CASable __x, int &__a, int &__b)
- template<typename _RAIter, typename _DifferenceTp >
 void **__determine_samples** (_PMWMSortingData<_RAIter> *__sd, _DifferenceTp __num_samples)
- [_CASable](#) **__encode2** (int __a, int __b)

- `template<typename _DifferenceType, typename _OutputIterator >`
`_OutputIterator __equally_split (_DifferenceType __n, _ThreadIndex __num_threads, _OutputIterator __s)`
- `template<typename _DifferenceType >`
`_DifferenceType __equally_split_point (_DifferenceType __n, _ThreadIndex __num_threads, _ThreadIndex __thread_no)`
- `template<typename _Tp >`
`_Tp __fetch_and_add (volatile _Tp * __ptr, _Tp __addend)`
- `template<typename _RAlter1, typename _RAlter2, typename _Pred, typename _Selector >`
`std::pair< _RAlter1, _RAlter2 > __find_template (_RAlter1 __begin1, _RAlter1 __end1, _RAlter2 __begin2, _Pred __pred, _Selector __selector)`
- `template<typename _RAlter1, typename _RAlter2, typename _Pred, typename _Selector >`
`std::pair< _RAlter1, _RAlter2 > __find_template (_RAlter1 __begin1, _RAlter1 __end1, _RAlter2 __begin2, _Pred __pred, _Selector __selector, equal_split_tag)`
- `template<typename _RAlter1, typename _RAlter2, typename _Pred, typename _Selector >`
`std::pair< _RAlter1, _RAlter2 > __find_template (_RAlter1 __begin1, _RAlter1 __end1, _RAlter2 __begin2, _Pred __pred, _Selector __selector, growing_blocks_tag)`
- `template<typename _RAlter1, typename _RAlter2, typename _Pred, typename _Selector >`
`std::pair< _RAlter1, _RAlter2 > __find_template (_RAlter1 __begin1, _RAlter1 __end1, _RAlter2 __begin2, _Pred __pred, _Selector __selector, constant_size_blocks_tag)`
- `template<typename _Iter, typename _UserOp, typename _Functionality, typename _Red, typename _Result >`
`_UserOp __for_each_template_random_access (_Iter __begin, _Iter __end, _UserOp __user_op, _Functionality & __functionality, _Red __reduction, _Result __reduction_start, _Result & __output, typename std::iterator_traits< _Iter >::difference_type __bound, _Parallelism __parallelism_tag)`
- `template<typename _RAlter, typename _Op, typename _Fu, typename _Red, typename _Result >`
`_Op __for_each_template_random_access_ed (_RAlter __begin, _RAlter __end, _Op __o, _Fu & __f, _Red __r, _Result __base, _Result & __output, typename std::iterator_traits< _RAlter >::difference_type __bound)`
- `template<typename _RAlter, typename _Op, typename _Fu, typename _Red, typename _Result >`
`_Op __for_each_template_random_access_omp_loop (_RAlter __begin, _RAlter __end, _Op __o, _Fu & __f, _Red __r, _Result __base, _Result & __output, typename std::iterator_traits< _RAlter >::difference_type __bound)`
- `template<typename _RAlter, typename _Op, typename _Fu, typename _Red, typename _Result >`
`_Op __for_each_template_random_access_omp_loop_static (_RAlter __begin, _RAlter __end, _Op __o, _Fu & __f, _Red __r, _Result __base, _Result & __output, typename std::iterator_traits< _RAlter >::difference_type __bound)`
- `template<typename _RAlter, typename _Op, typename _Fu, typename _Red, typename _Result >`
`_Op __for_each_template_random_access_workstealing (_RAlter __begin, _RAlter __end, _Op __op, _Fu & __f, _Red __r, _Result __base, _Result & __output, typename std::iterator_traits< _RAlter >::difference_type __bound)`
- `_ThreadIndex __get_max_threads ()`
- `bool __is_parallel (const _Parallelism __p)`
- `template<typename _Iter, typename _Compare >`
`bool __is_sorted (_Iter __begin, _Iter __end, _Compare __comp)`
- `template<typename _RAlter, typename _Compare >`
`_RAlter __median_of_three_iterators (_RAlter __a, _RAlter __b, _RAlter __c, _Compare __comp)`
- `template<typename _RAlter1, typename _RAlter2, typename _OutputIterator, typename _DifferenceTp, typename _Compare >`
`_OutputIterator __merge_advance (_RAlter1 & __begin1, _RAlter1 __end1, _RAlter2 & __begin2, _RAlter2 __end2, _OutputIterator __target, _DifferenceTp __max_length, _Compare __comp)`
- `template<typename _RAlter1, typename _RAlter2, typename _OutputIterator, typename _DifferenceTp, typename _Compare >`
`_OutputIterator __merge_advance_movc (_RAlter1 & __begin1, _RAlter1 __end1, _RAlter2 & __begin2, _RAlter2 __end2, _OutputIterator __target, _DifferenceTp __max_length, _Compare __comp)`
- `template<typename _RAlter1, typename _RAlter2, typename _OutputIterator, typename _DifferenceTp, typename _Compare >`
`_OutputIterator __merge_advance_usual (_RAlter1 & __begin1, _RAlter1 __end1, _RAlter2 & __begin2, _RAlter2 __end2, _OutputIterator __target, _DifferenceTp __max_length, _Compare __comp)`

- `template<typename _RAIter1, typename _RAIter2, typename _RAIter3, typename _Compare >`
`_RAIter3 __parallel_merge_advance (_RAIter1 &__begin1, _RAIter1 __end1, _RAIter2 &__begin2, _RAIter2 &__end2, _RAIter3 __target, typename std::iterator_traits<_RAIter1 >::difference_type __max_length, _Compare __comp)`
- `template<typename _RAIter1, typename _RAIter3, typename _Compare >`
`_RAIter3 __parallel_merge_advance (_RAIter1 &__begin1, _RAIter1 __end1, _RAIter1 &__begin2, _RAIter1 &__end2, _RAIter3 __target, typename std::iterator_traits<_RAIter1 >::difference_type __max_length, _Compare __comp)`
- `template<typename _RAIter, typename _Compare >`
`void __parallel_nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter, typename _Compare >`
`void __parallel_partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end, _Compare __comp)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`
`_OutputIterator __parallel_partial_sum (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`
`_OutputIterator __parallel_partial_sum_basecase (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, typename std::iterator_traits<_Iter >::value_type __value)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`
`_OutputIterator __parallel_partial_sum_linear (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, typename std::iterator_traits<_Iter >::difference_type __n)`
- `template<typename _RAIter, typename _Predicate >`
`std::iterator_traits<_RAIter >::difference_type __parallel_partition (_RAIter __begin, _RAIter __end, _Predicate __pred, _ThreadIndex __num_threads)`
- `template<typename _RAIter, typename _RandomNumberGenerator >`
`void __parallel_random_shuffle (_RAIter __begin, _RAIter __end, _RandomNumberGenerator __rng=RandomNumber())`
- `template<typename _RAIter, typename _RandomNumberGenerator >`
`void __parallel_random_shuffle_drs (_RAIter __begin, _RAIter __end, typename std::iterator_traits<_RAIter >::difference_type __n, _ThreadIndex __num_threads, _RandomNumberGenerator &__rng)`
- `template<typename _RAIter, typename _RandomNumberGenerator >`
`void __parallel_random_shuffle_drs_pu (_DRSSorterPU<_RAIter, _RandomNumberGenerator > *__pus)`
- `template<typename _Iter, typename _OutputIterator, typename _Compare >`
`_OutputIterator __parallel_set_difference (_Iter __begin1, _Iter __end1, _Iter __begin2, _Iter __end2, _OutputIterator __result, _Compare __comp)`
- `template<typename _Iter, typename _OutputIterator, typename _Compare >`
`_OutputIterator __parallel_set_intersection (_Iter __begin1, _Iter __end1, _Iter __begin2, _Iter __end2, _OutputIterator __result, _Compare __comp)`
- `template<typename _Iter, typename _OutputIterator, typename _Operation >`
`_OutputIterator __parallel_set_operation (_Iter __begin1, _Iter __end1, _Iter __begin2, _Iter __end2, _OutputIterator __result, _Operation __op)`
- `template<typename _Iter, typename _OutputIterator, typename _Compare >`
`_OutputIterator __parallel_set_symmetric_difference (_Iter __begin1, _Iter __end1, _Iter __begin2, _Iter __end2, _OutputIterator __result, _Compare __comp)`
- `template<typename _Iter, typename _OutputIterator, typename _Compare >`
`_OutputIterator __parallel_set_union (_Iter __begin1, _Iter __end1, _Iter __begin2, _Iter __end2, _OutputIterator __result, _Compare __comp)`
- `template<bool __stable, typename _RAIter, typename _Compare, typename _Parallelism >`
`void __parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, _Parallelism __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare >`
`void __parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, multiway_mergesort_tag __parallelism)`

- `template<bool __stable, typename _RAIter, typename _Compare >`
`void __parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, multiway_mergesort_exact_tag __↵`
`parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare >`
`void __parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, multiway_mergesort_sampling_tag`
`__parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare >`
`void __parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, quicksort_tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare >`
`void __parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, balanced_quicksort_tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare >`
`void __parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, default_parallel_tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare >`
`void __parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, parallel_tag __parallelism)`
- `template<typename _RAIter, typename _Compare >`
`void __parallel_sort_qs (_RAIter __begin, _RAIter __end, _Compare __comp, ThreadIndex __num_threads)`
- `template<typename _RAIter, typename _Compare >`
`void __parallel_sort_qs_conquer (_RAIter __begin, _RAIter __end, _Compare __comp, ThreadIndex __num↵`
`__threads)`
- `template<typename _RAIter, typename _Compare >`
`std::iterator_traits< _RAIter >::difference_type __parallel_sort_qs_divide (_RAIter __begin, _RAIter __end, ↵`
`_Compare __comp, typename std::iterator_traits< _RAIter >::difference_type __pivot_rank, typename std::↵`
`::iterator_traits< _RAIter >::difference_type __num_samples, ThreadIndex __num_threads)`
- `template<typename _RAIter, typename _Compare >`
`void __parallel_sort_qsb (_RAIter __begin, _RAIter __end, _Compare __comp, ThreadIndex __num_threads)`
- `template<typename _Iter, class _OutputIterator, class _BinaryPredicate >`
`_OutputIterator __parallel_unique_copy (_Iter __first, _Iter __last, _OutputIterator __result, _BinaryPredicate`
`__binary_pred)`
- `template<typename _Iter, class _OutputIterator >`
`_OutputIterator __parallel_unique_copy (_Iter __first, _Iter __last, _OutputIterator __result)`
- `template<typename _RAIter, typename _Compare >`
`void __qsb_conquer (_QSBThreadLocal< _RAIter > *__tls, _RAIter __begin, _RAIter __end, _Compare __↵`
`comp, ThreadIndex __iam, ThreadIndex __num_threads, bool __parent_wait)`
- `template<typename _RAIter, typename _Compare >`
`std::iterator_traits< _RAIter >::difference_type __qsb_divide (_RAIter __begin, _RAIter __end, _Compare __↵`
`comp, ThreadIndex __num_threads)`
- `template<typename _RAIter, typename _Compare >`
`void __qsb_local_sort_with_helping (_QSBThreadLocal< _RAIter > *__tls, _Compare &__comp, Thread↵`
`Index __iam, bool __wait)`
- `template<typename _RandomNumberGenerator >`
`int __random_number_pow2 (int __logp, _RandomNumberGenerator &__rng)`
- `template<typename _Size >`
`_Size __rd_log2 (_Size __n)`
- `template<typename _Tp >`
`_Tp __round_up_to_pow2 (_Tp __x)`
- `template<typename _RAIter1, typename _RAIter2, typename _Pred >`
`__RAIter1 __search_template (__RAIter1 __begin1, __RAIter1 __end1, __RAIter2 __begin2, __RAIter2 __end2,`
`__Pred __pred)`
- `template<bool __stable, bool __sentinels, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare`
`>`
`_RAIter3 __sequential_multiway_merge (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3`
`__target, const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator >::value_type↵`
`::first_type >::value_type &__sentinel, _DifferenceTp __length, _Compare __comp)`

- `template<typename _RAIter, typename _RandomNumberGenerator >`
`void __sequential_random_shuffle (_RAIter __begin, _RAIter __end, _RandomNumberGenerator &__rng)`
- `template<typename _Iter >`
`void __shrink (std::vector< _Iter > &__os_starts, size_t &__count_to_two, size_t &__range_length)`
- `template<typename _Iter >`
`void __shrink_and_double (std::vector< _Iter > &__os_starts, size_t &__count_to_two, size_t &__range_length, const bool __make_twice)`
- `void __yield ()`
- `template<typename _Iter, typename _FunctorType >`
`size_t list_partition (const _Iter __begin, const _Iter __end, _Iter * __starts, size_t * __lengths, const int __num_parts, _FunctorType &__f, int __oversampling=0)`
- `template<typename _Tp >`
`const _Tp &max (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp >`
`const _Tp &min (const _Tp &__a, const _Tp &__b)`
- `template<typename _RanSeqs, typename _RankType, typename _RankIterator, typename _Compare >`
`void multiseq_partition (_RanSeqs __begin_seqs, _RanSeqs __end_seqs, _RankType __rank, _RankIterator __begin_offsets, _Compare __comp=std::less< typename std::iterator_traits< typename std::iterator_traits< _RanSeqs >::value_type::first_type >::value_type >())`
- `template<typename _Tp, typename _RanSeqs, typename _RankType, typename _Compare >`
`_Tp multiseq_selection (_RanSeqs __begin_seqs, _RanSeqs __end_seqs, _RankType __rank, _RankType __offset, _Compare __comp=std::less< _Tp >())`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`
`_RAIterOut multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::sequential_tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`
`_RAIterOut multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::exact_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`
`_RAIterOut multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::sampling_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`
`_RAIterOut multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, parallel_tag __tag=parallel_tag(0))`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`
`_RAIterOut multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, default_parallel_tag __tag)`
- `template<template< typename RAI, typename C > class iterator, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare >`
`_RAIter3 multiway_merge_3_variant (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp)`
- `template<template< typename RAI, typename C > class iterator, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare >`
`_RAIter3 multiway_merge_4_variant (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp)`
- `template<bool __stable, typename _RAIterIterator, typename _Compare, typename _DifferenceType >`
`void multiway_merge_exact_splitting (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _DifferenceType __length, _DifferenceType __total_length, _Compare __comp, std::vector< std::pair< _DifferenceType, _DifferenceType > > * __pieces)`
- `template<typename _LT, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare >`
`_RAIter3 multiway_merge_loser_tree (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp)`

- `template<typename UnguardedLoserTree, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare >`
`_RAIter3 multiway_merge_loser_tree_sentinel (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator >::value_type::first_type >::value_type & __sentinel, _DifferenceTp __length, _Compare __comp)`
- `template<typename _LT, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare >`
`_RAIter3 multiway_merge_loser_tree_unguarded (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator >::value_type::first_type >::value_type & __sentinel, _DifferenceTp __length, _Compare __comp)`
- `template<bool __stable, typename _RAIterIterator, typename _Compare, typename _DifferenceType >`
`void multiway_merge_sampling_splitting (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _DifferenceType __length, _DifferenceType __total_length, _Compare __comp, std::vector< std::pair< _DifferenceType, _DifferenceType > > * __pieces)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`
`_RAIterOut multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, gnu_parallel::sequential_tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`
`_RAIterOut multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, gnu_parallel::exact_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`
`_RAIterOut multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, sampling_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`
`_RAIterOut multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, parallel_tag __tag=parallel_tag(0))`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`
`_RAIterOut multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, default_parallel_tag __tag)`
- `template<bool __stable, bool __sentinels, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Splitter, typename _Compare >`
`_RAIter3 parallel_multiway_merge (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _Splitter __splitter, _DifferenceTp __length, _Compare __comp, ThreadIndex __num_threads)`
- `template<bool __stable, bool __exact, typename _RAIter, typename _Compare >`
`void parallel_sort_mwms (_RAIter __begin, _RAIter __end, _Compare __comp, ThreadIndex __num_threads)`
- `template<bool __stable, bool __exact, typename _RAIter, typename _Compare >`
`void parallel_sort_mwms_pu (_PMWMSortingData< _RAIter > * __sd, _Compare & __comp)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`
`_RAIterOut stable_multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, gnu_parallel::sequential_tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`
`_RAIterOut stable_multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, gnu_parallel::exact_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`
`_RAIterOut stable_multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, sampling_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`
`_RAIterOut stable_multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, parallel_tag __tag=parallel_tag(0))`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`
`_RAIterOut stable_multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, default_parallel_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`
`_RAIterOut stable_multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, gnu_parallel::sequential_tag)`

- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`
`_RAIterOut stable_multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __↵`
`seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::exact_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`
`_RAIterOut stable_multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __↵`
`seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, sampling_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`
`_RAIterOut stable_multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __↵`
`seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, parallel_tag __tag=parallel_tag(0))`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`
`_RAIterOut stable_multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __↵`
`seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, default_parallel_tag __tag)`

Variables

- static const int [_CASable_bits](#)
- static const [_CASable](#) [_CASable_mask](#)

3.6.1 Detailed Description

GNU parallel code for public use.

3.6.2 Typedef Documentation

3.6.2.1 `typedef unsigned short __gnu_parallel::_BinIndex`

Type to hold the index of a bin.

Since many variables of this type are allocated, it should be chosen as small as possible.

Definition at line 47 of file `random_shuffle.h`.

3.6.2.2 `typedef int64_t __gnu_parallel::_CASable`

Longest compare-and-swappable integer type on this platform.

Definition at line 127 of file `types.h`.

3.6.2.3 `typedef uint64_t __gnu_parallel::_SequenceIndex`

Unsigned integer to index `__elements`. The total number of elements for each algorithm must fit into this type.

Definition at line 117 of file `types.h`.

3.6.2.4 `typedef uint16_t __gnu_parallel::_ThreadIndex`

Unsigned integer to index a thread number. The maximum thread number (for each processor) must fit into this type.

Definition at line 123 of file types.h.

3.6.3 Enumeration Type Documentation

3.6.3.1 `enum __gnu_parallel::_AlgorithmStrategy`

Strategies for run-time algorithm selection:

Definition at line 67 of file types.h.

3.6.3.2 `enum __gnu_parallel::_FindAlgorithm`

Find algorithms:

Definition at line 106 of file types.h.

3.6.3.3 `enum __gnu_parallel::_MultiwayMergeAlgorithm`

Merging algorithms:

Definition at line 85 of file types.h.

3.6.3.4 `enum __gnu_parallel::_Parallelism`

Run-time equivalents for the compile-time tags.

Enumerator

sequential Not parallel.

parallel_unbalanced Parallel unbalanced (equal-sized chunks).

parallel_balanced Parallel balanced (work-stealing).

parallel_omp_loop Parallel with OpenMP dynamic load-balancing.

parallel_omp_loop_static Parallel with OpenMP static load-balancing.

parallel_taskqueue Parallel with OpenMP taskqueue construct.

Definition at line 44 of file types.h.

3.6.3.5 `enum __gnu_parallel::_PartialSumAlgorithm`

Partial sum algorithms: recursive, linear.

Definition at line 91 of file types.h.

3.6.3.6 `enum __gnu_parallel::_SortAlgorithm`

Sorting algorithms:

Definition at line 76 of file `types.h`.

3.6.3.7 `enum __gnu_parallel::_SplittingAlgorithm`

Sorting/merging algorithms: sampling, `__exact`.

Definition at line 98 of file `types.h`.

3.6.4 Function Documentation

3.6.4.1 `template<typename _RAIter, typename _DifferenceTp> void __gnu_parallel::_calc_borders (_RAIter __elements, _DifferenceTp __length, _DifferenceTp * __off)`

Precalculate `__advances` for Knuth-Morris-Pratt algorithm.

Parameters

<code>__elements</code>	Begin iterator of sequence to search for.
<code>__length</code>	Length of sequence to search for.
<code>__off</code>	Returned <code>__offsets</code> .

Definition at line 51 of file `search.h`.

Referenced by `__search_template()`.

3.6.4.2 `template<typename _Tp> bool __gnu_parallel::_compare_and_swap (volatile _Tp * __ptr, _Tp __comparand, _Tp __replacement) [inline]`

Compare-and-swap.

Compare `*__ptr` and `__comparand`. If equal, let `*__ptr=__replacement` and return true, return false otherwise.

Parameters

<code>__ptr</code>	Pointer to signed integer.
<code>__comparand</code>	Compare value.
<code>__replacement</code>	Replacement value.

Definition at line 108 of file `parallel/compatibility.h`.

Referenced by `__parallel_partition()`, `__gnu_parallel::_RestrictedBoundedConcurrentQueue< pair< _RAIter, _RAIter`

> >::pop_back(), and __gnu_parallel::RestrictedBoundedConcurrentQueue< pair< _RAIter, _RAIter > >::pop_↵front().

3.6.4.3 void __gnu_parallel::__decode2 (_CASable __x, int &__a, int &__b) [inline]

Decode two integers from one gnu_parallel::__CASable.

Parameters

↵ __x	__gnu_parallel::__CASable to decode integers from.
↵ __a	First integer, to be decoded from the most-significant _CASable_bits/2 bits of __x.
↵ __b	Second integer, to be encoded in the least-significant _CASable_bits/2 bits of __x.

See also

__encode2

Definition at line 133 of file parallel/base.h.

References _CASable_bits, and _CASable_mask.

Referenced by __gnu_parallel::RestrictedBoundedConcurrentQueue< pair< _RAIter, _RAIter > >::pop_back(), ↵__gnu_parallel::RestrictedBoundedConcurrentQueue< pair< _RAIter, _RAIter > >::pop_front(), and ↵__gnu_parallel::RestrictedBoundedConcurrentQueue< pair< _RAIter, _RAIter > >::push_front().

3.6.4.4 template<typename _RAIter, typename _DifferenceTp> void __gnu_parallel::__determine_samples (_PMWMSSortingData< _RAIter > * __sd, _DifferenceTp __num_samples)

Select _M_samples from a sequence.

Parameters

__sd	Pointer to algorithm data. _Result will be placed in __sd->_M_samples.
__num_samples	Number of _M_samples to select.

Definition at line 97 of file multiway_mergesort.h.

References __equally_split(), __gnu_parallel::PMWMSSortingData< _RAIter >::_M_samples, ↵__gnu_parallel::P↵MWMSSortingData< _RAIter >::_M_source, and __gnu_parallel::PMWMSSortingData< _RAIter >::_M_starts.

3.6.4.5 _CASable __gnu_parallel::__encode2 (int __a, int __b) [inline]

Encode two integers into one gnu_parallel::__CASable.

Parameters

<code>__a</code>	First integer, to be encoded in the most-significant <code>_CASable_bits/2</code> bits.
<code>__b</code>	Second integer, to be encoded in the least-significant <code>_CASable_bits/2</code> bits.

Returns

value encoding `__a` and `__b`.

See also

`__decode2`

Definition at line 119 of file `parallel/base.h`.

References `_CASable_bits`.

Referenced by `__gnu_parallel::RestrictedBoundedConcurrentQueue< pair< _RAIter, _RAIter > >::RestrictedBoundedConcurrentQueue()`, `__gnu_parallel::RestrictedBoundedConcurrentQueue< pair< _RAIter, _RAIter > >::pop_back()`, `__gnu_parallel::RestrictedBoundedConcurrentQueue< pair< _RAIter, _RAIter > >::pop_front()`, and `__gnu_parallel::RestrictedBoundedConcurrentQueue< pair< _RAIter, _RAIter > >::push_front()`.

3.6.4.6 `template<typename _DifferenceType, typename _OutputIterator> _OutputIterator __gnu_parallel::__equally_split (_DifferenceType __n, _ThreadIndex __num_threads, _OutputIterator __s)`

function to split a sequence into parts of almost equal size.

The resulting sequence `__s` of length `__num_threads+1` contains the splitting positions when splitting the range `[0, __n)` into parts of almost equal size (plus minus 1). The first entry is 0, the last one `n`. There may result empty parts.

Parameters

<code>__n</code>	Number of elements
<code>__num_threads</code>	Number of parts
<code>__s</code>	Splitters

Returns

End of `__splitter` sequence, i.e. `__s+__num_threads+1`

Definition at line 48 of file `equally_split.h`.

Referenced by `__determine_samples()`, `__find_template()`, `__parallel_partial_sum_linear()`, `__parallel_unique_copy()`, `__search_template()`, and `multiway_merge_exact_splitting()`.

3.6.4.7 `template<typename _DifferenceType > _DifferenceType __gnu_parallel::__equally_split_point (_DifferenceType __n, _ThreadIndex __num_threads, _ThreadIndex __thread_no)`

function to split a sequence into parts of almost equal size.

Returns the position of the splitting point between thread number `__thread_no` (included) and thread number `__thread_no+1` (excluded).

Parameters

<code>__n</code>	Number of elements
<code>__num_threads</code>	Number of parts
<code>__thread_no</code>	Number of threads

Returns

splitting point

Definition at line 75 of file `equally_split.h`.

Referenced by `__for_each_template_random_access_ed()`.

3.6.4.8 `template<typename _Tp > _Tp __gnu_parallel::__fetch_and_add (volatile _Tp * __ptr, _Tp __addend) [inline]`

Add a value to a variable, atomically.

Parameters

<code>__ptr</code>	Pointer to a signed integer.
<code>__addend</code>	Value to add.

Definition at line 74 of file `parallel/compatibility.h`.

Referenced by `__parallel_partition()`, and `__gnu_parallel::__RestrictedBoundedConcurrentQueue< pair< _RAIter, _RAIter > >::push_front()`.

3.6.4.9 `template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector > std::pair< _RAIter1, _RAIter2> __gnu_parallel::__find_template (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred, _Selector __selector) [inline]`

Parallel `std::find`, switch for different algorithms.

Parameters

<code>__begin1</code>	Begin iterator of first sequence.
<code>__end1</code>	End iterator of first sequence.
<code>__begin2</code>	Begin iterator of second sequence. Must have same length as first sequence.
<code>__pred</code>	Find predicate.
<code>__selector</code>	<code>_Functionality</code> (e. g. <code>std::find_if()</code> , <code>std::equal()</code> ,...)

Returns

Place of finding in both sequences.

Definition at line 60 of file find.h.

References __gnu_parallel::_Settings::get(), and std::make_pair().

```
3.6.4.10 template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector > std::pair<_RAIter1,
    _RAIter2> __gnu_parallel::_find_template ( _RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred,
    _Selector __selector, equal_split_tag )
```

Parallel std::find, equal splitting variant.

Parameters

<code>__begin1</code>	Begin iterator of first sequence.
<code>__end1</code>	End iterator of first sequence.
<code>__begin2</code>	Begin iterator of second sequence. Second __sequence must have same length as first sequence.
<code>__pred</code>	Find predicate.
<code>__selector</code>	_Functionality (e. g. std::find_if(), std::equal(),...)

Returns

Place of finding in both sequences.

Definition at line 97 of file find.h.

References __equally_split(), and _GLIBCXX_CALL.

```
3.6.4.11 template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector > std::pair<_RAIter1,
    _RAIter2> __gnu_parallel::_find_template ( _RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred,
    _Selector __selector, growing_blocks_tag )
```

Parallel std::find, growing block size variant.

Parameters

<code>__begin1</code>	Begin iterator of first sequence.
<code>__end1</code>	End iterator of first sequence.
<code>__begin2</code>	Begin iterator of second sequence. Second __sequence must have same length as first sequence.
<code>__pred</code>	Find predicate.
<code>__selector</code>	_Functionality (e. g. std::find_if(), std::equal(),...)

Returns

Place of finding in both sequences.

See also

`__gnu_parallel::_Settings::find_sequential_search_size`
`__gnu_parallel::_Settings::find_scale_factor`

There are two main differences between the growing blocks and the constant-size blocks variants. 1. For GB, the block size grows; for CSB, the block size is fixed. 2. For GB, the blocks are allocated dynamically; for CSB, the blocks are allocated in a predetermined manner, namely spacial round-robin.

Definition at line 185 of file find.h.

References `_GLIBCXX_CALL`, `__gnu_parallel::_Settings::find_scale_factor`, `__gnu_parallel::_Settings::find_sequential_search_size`, and `__gnu_parallel::_Settings::get()`.

3.6.4.12 `template<typename _RAIter1 , typename _RAIter2 , typename _Pred , typename _Selector > std::pair<_RAIter1, _RAIter2> __gnu_parallel:: find_template (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred, _Selector __selector, constant_size_blocks_tag)`

Parallel `std::find`, constant block size variant.

Parameters

<code>__begin1</code>	Begin iterator of first sequence.
<code>__end1</code>	End iterator of first sequence.
<code>__begin2</code>	Begin iterator of second sequence. Second <code>__sequence</code> must have same length as first sequence.
<code>__pred</code>	Find predicate.
<code>__selector</code>	<code>_Functionality</code> (e. g. <code>std::find_if()</code> , <code>std::equal()</code> ,...)

Returns

Place of finding in both sequences.

See also

`__gnu_parallel::_Settings::find_sequential_search_size`
`__gnu_parallel::_Settings::find_block_size` There are two main differences between the growing blocks and the constant-size blocks variants. 1. For GB, the block size grows; for CSB, the block size is fixed. 2. For GB, the blocks are allocated dynamically; for CSB, the blocks are allocated in a predetermined manner, namely spacial round-robin.

Definition at line 315 of file find.h.

References `_GLIBCXX_CALL`, `__gnu_parallel::_Settings::find_initial_block_size`, `__gnu_parallel::_Settings::find_sequential_search_size`, and `__gnu_parallel::_Settings::get()`.

3.6.4.13 `template<typename _Iter , typename _UserOp , typename _Functionality , typename _Red , typename _Result > _UserOp __gnu_parallel::_for_each_template_random_access (_Iter __begin, _Iter __end, _UserOp __user_op, _Functionality & __functionality, _Red __reduction, _Result __reduction_start, _Result & __output, typename std::iterator_traits<_Iter>::difference_type __bound, _Parallelism __parallelism_tag)`

Chose the desired algorithm by evaluating `__parallelism_tag`.

Parameters

<code>__begin</code>	Begin iterator of input sequence.
<code>__end</code>	End iterator of input sequence.
<code>__user_op</code>	A user-specified functor (comparator, predicate, associative operator,...)
<code>__functionality</code>	functor to <i>process</i> an element with <code>__user_op</code> (depends on desired functionality, e. g. accumulate, <code>for_each</code> ,...)
<code>__reduction</code>	Reduction functor.
<code>__reduction_start</code>	Initial value for reduction.
<code>__output</code>	Output iterator.
<code>__bound</code>	Maximum number of elements processed.
<code>__parallelism_tag</code>	Parallelization method

Definition at line 61 of file `for_each.h`.

References `__for_each_template_random_access_ed()`, `__for_each_template_random_access_omp_loop()`, `__↵`
`for_each_template_random_access_workstealing()`, `parallel_omp_loop`, `parallel_omp_loop_static`, and `parallel_↵`
unbalanced.

3.6.4.14 `template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result> _Op`
`__gnu_parallel::__for_each_template_random_access_ed(_RAIter __begin, _RAIter __end, _Op __o, _Fu & __f, _Red __r,`
`_Result __base, _Result & __output, typename std::iterator_traits< _RAIter >::difference_type __bound)`

Embarrassingly parallel algorithm for random access iterators, using hand-crafted parallelization by equal splitting the work.

Parameters

<code>__begin</code>	Begin iterator of element sequence.
<code>__end</code>	End iterator of element sequence.
<code>__o</code>	User-supplied functor (comparator, predicate, adding functor, ...)
<code>__f</code>	Functor to "process" an element with <code>__op</code> (depends on desired functionality, e. g. for <code>std::for_each()</code> , ...).
<code>__r</code>	Functor to "add" a single <code>__result</code> to the already processed elements (depends on functionality).
<code>__base</code>	Base value for reduction.
<code>__output</code>	Pointer to position where final result is written to
<code>__bound</code>	Maximum number of elements processed (e. g. for <code>std::count_n()</code>).

Returns

User-supplied functor (that may contain a part of the result).

Definition at line 67 of file `par_loop.h`.

References `__equally_split_point()`.

Referenced by `__for_each_template_random_access()`.

3.6.4.15 `template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result> _Op
__gnu_parallel::__for_each_template_random_access_omp_loop(_RAIter __begin, _RAIter __end, _Op __o, _Fu & __f,
_Red __r, _Result __base, _Result & __output, typename std::iterator_traits<_RAIter>::difference_type __bound)`

Embarrassingly parallel algorithm for random access iterators, using an OpenMP for loop.

Parameters

<code>__begin</code>	Begin iterator of element sequence.
<code>__end</code>	End iterator of element sequence.
<code>__o</code>	User-supplied functor (comparator, predicate, adding functor, etc.).
<code>__f</code>	Functor to <i>process</i> an element with <code>__op</code> (depends on desired functionality, e. g. for <code>std::for_each()</code> , ...).
<code>__r</code>	Functor to <i>add</i> a single <code>__result</code> to the already processed elements (depends on functionality).
<code>__base</code>	Base value for reduction.
<code>__output</code>	Pointer to position where final result is written to
<code>__bound</code>	Maximum number of elements processed (e. g. for <code>std::count_n()</code>).

Returns

User-supplied functor (that may contain a part of the result).

Definition at line 67 of file `omp_loop.h`.

Referenced by `__for_each_template_random_access()`.

3.6.4.16 `template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result> _Op
__gnu_parallel::__for_each_template_random_access_omp_loop_static(_RAIter __begin, _RAIter __end, _Op __o, _Fu &
__f, _Red __r, _Result __base, _Result & __output, typename std::iterator_traits<_RAIter>::difference_type __bound)`

Embarrassingly parallel algorithm for random access iterators, using an OpenMP for loop with static scheduling.

Parameters

<code>__begin</code>	Begin iterator of element sequence.
<code>__end</code>	End iterator of element sequence.
<code>__o</code>	User-supplied functor (comparator, predicate, adding functor, ...).
<code>__f</code>	Functor to <i>process</i> an element with <code>__op</code> (depends on desired functionality, e. g. for <code>std::for_each()</code> , ...).
<code>__r</code>	Functor to <i>add</i> a single <code>__result</code> to the already processed <code>__elements</code> (depends on functionality).
<code>__base</code>	Base value for reduction.
<code>__output</code>	Pointer to position where final result is written to
<code>__bound</code>	Maximum number of elements processed (e. g. for <code>std::count_n()</code>).

Returns

User-supplied functor (that may contain a part of the result).

Definition at line 66 of file `omp_loop_static.h`.

3.6.4.17 `template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result> _Op
__gnu_parallel::__for_each_template_random_access_workstealing (_RAIter __begin, _RAIter __end, _Op __op, _Fu &
__f, _Red __r, _Result __base, _Result & __output, typename std::iterator_traits< _RAIter >::difference_type __bound)`

Work stealing algorithm for random access iterators.

Uses O(1) additional memory. Synchronization at job lists is done with atomic operations.

Parameters

<code>__begin</code>	Begin iterator of element sequence.
<code>__end</code>	End iterator of element sequence.
<code>__op</code>	User-supplied functor (comparator, predicate, adding functor, ...).
<code>__f</code>	Functor to <i>process</i> an element with <code>__op</code> (depends on desired functionality, e. g. for <code>std::for_each()</code> , ...).
<code>__r</code>	Functor to <i>add</i> a single <code>__result</code> to the already processed elements (depends on functionality).
<code>__base</code>	Base value for reduction.
<code>__output</code>	Pointer to position where final result is written to
<code>__bound</code>	Maximum number of elements processed (e. g. for <code>std::count_n()</code>).

Returns

User-supplied functor (that may contain a part of the result).

Definition at line 99 of file `workstealing.h`.

References `__gnu_debug::__base()`, `__yield()`, `_GLIBCXX_CALL`, `__gnu_parallel::__Job< _DifferenceTp >::__M_first`, `__gnu_parallel::__Job< _DifferenceTp >::__M_last`, `__gnu_parallel::__Job< _DifferenceTp >::__M_load`, `__gnu_parallel::__Settings::cache_line_size`, `__gnu_parallel::__Settings::get()`, and `min()`.

Referenced by `__for_each_template_random_access()`.

3.6.4.18 `template<typename _lIter, typename _Compare> bool __gnu_parallel::__is_sorted (_lIter __begin, _lIter __end,
_Compare __comp)`

Check whether `[__begin, __end)` is sorted according to `__comp`.

Parameters

<code>__begin</code>	Begin iterator of sequence.
<code>__end</code>	End iterator of sequence.
<code>__comp</code>	Comparator.

Returns

`true` if sorted, `false` otherwise.

Definition at line 51 of file `checkers.h`.

Referenced by `__sequential_multiway_merge()`, `multiway_merge_loser_tree_sentinel()`, and `parallel_multiway_merge()`.

3.6.4.19 `template<typename _RAIter , typename _Compare > _RAIter __gnu_parallel::__median_of_three_iterators (_RAIter __a, _RAIter __b, _RAIter __c, _Compare __comp)`

Compute the median of three referenced elements, according to `__comp`.

Parameters

<code>__a</code>	First iterator.
<code>__b</code>	Second iterator.
<code>__c</code>	Third iterator.
<code>__comp</code>	Comparator.

Definition at line 398 of file `parallel/base.h`.

Referenced by `__qsb_divide()`.

3.6.4.20 `template<typename _RAIter1 , typename _RAIter2 , typename _OutputIterator , typename _DifferenceTp , typename _Compare > _OutputIterator __gnu_parallel::__merge_advance (_RAIter1 & __begin1, _RAIter1 __end1, _RAIter2 & __begin2, _RAIter2 __end2, _OutputIterator __target, _DifferenceTp __max_length, _Compare __comp) [inline]`

Merge routine being able to merge only the `__max_length` smallest elements.

The `__begin` iterators are advanced accordingly, they might not reach `__end`, in contrast to the usual variant. Static switch on whether to use the conditional-move variant.

Parameters

<code>__begin1</code>	Begin iterator of first sequence.
<code>__end1</code>	End iterator of first sequence.
<code>__begin2</code>	Begin iterator of second sequence.
<code>__end2</code>	End iterator of second sequence.
<code>__target</code>	Target begin iterator.
<code>__max_length</code>	Maximum number of elements to merge.
<code>__comp</code>	Comparator.

Returns

Output end iterator.

Definition at line 171 of file `merge.h`.

References `__merge_advance_movc()`, and `_GLIBCXX_CALL`.

Referenced by `__parallel_merge_advance()`, and `__sequential_multiway_merge()`.

3.6.4.21 `template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename _DifferenceTp, typename _Compare> _OutputIterator __gnu_parallel::__merge_advance_movc (_RAIter1 & __begin1, _RAIter1 __end1, _RAIter2 & __begin2, _RAIter2 __end2, _OutputIterator __target, _DifferenceTp __max_length, _Compare __comp)`

Merge routine being able to merge only the `__max_length` smallest elements.

The `__begin` iterators are advanced accordingly, they might not reach `__end`, in contrast to the usual variant. Specially designed code should allow the compiler to generate conditional moves instead of branches.

Parameters

<code>__begin1</code>	Begin iterator of first sequence.
<code>__end1</code>	End iterator of first sequence.
<code>__begin2</code>	Begin iterator of second sequence.
<code>__end2</code>	End iterator of second sequence.
<code>__target</code>	Target begin iterator.
<code>__max_length</code>	Maximum number of elements to merge.
<code>__comp</code>	Comparator.

Returns

Output end iterator.

Definition at line 105 of file `merge.h`.

Referenced by `__merge_advance()`.

3.6.4.22 `template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename _DifferenceTp, typename _Compare> _OutputIterator __gnu_parallel::__merge_advance_usual (_RAIter1 & __begin1, _RAIter1 __end1, _RAIter2 & __begin2, _RAIter2 __end2, _OutputIterator __target, _DifferenceTp __max_length, _Compare __comp)`

Merge routine being able to merge only the `__max_length` smallest elements.

The `__begin` iterators are advanced accordingly, they might not reach `__end`, in contrast to the usual variant.

Parameters

<code>__begin1</code>	Begin iterator of first sequence.
<code>__end1</code>	End iterator of first sequence.
<code>__begin2</code>	Begin iterator of second sequence.
<code>__end2</code>	End iterator of second sequence.
<code>__target</code>	Target begin iterator.
<code>__max_length</code>	Maximum number of elements to merge.
<code>__comp</code>	Comparator.

Returns

Output end iterator.

Definition at line 57 of file merge.h.

```
3.6.4.23 template<typename _RAIter1, typename _RAIter2, typename _RAIter3, typename _Compare > _RAIter3
    __gnu_parallel::__parallel_merge_advance ( _RAIter1 & __begin1, _RAIter1 __end1, _RAIter2 & __begin2, _RAIter2
    __end2, _RAIter3 __target, typename std::iterator_traits< _RAIter1 >::difference_type __max_length, _Compare __comp
    ) [inline]
```

Merge routine fallback to sequential in case the iterators of the two input sequences are of different type.

Parameters

<code>__begin1</code>	Begin iterator of first sequence.
<code>__end1</code>	End iterator of first sequence.
<code>__begin2</code>	Begin iterator of second sequence.
<code>__end2</code>	End iterator of second sequence.
<code>__target</code>	Target begin iterator.
<code>__max_length</code>	Maximum number of elements to merge.
<code>__comp</code>	Comparator.

Returns

Output end iterator.

Definition at line 195 of file merge.h.

References `__merge_advance()`.

```
3.6.4.24 template<typename _RAIter1, typename _RAIter3, typename _Compare > _RAIter3 __gnu_parallel::__parallel_merge_↵
advance ( _RAIter1 & __begin1, _RAIter1 __end1, _RAIter1 & __begin2, _RAIter1 __end2, _RAIter3 __target, typename
std::iterator_traits< _RAIter1 >::difference_type __max_length, _Compare __comp ) [inline]
```

Parallel merge routine being able to merge only the `__max_length` smallest elements.

The `__begin` iterators are advanced accordingly, they might not reach `__end`, in contrast to the usual variant. The functionality is projected onto `parallel_multiway_merge`.

Parameters

<code>__begin1</code>	Begin iterator of first sequence.
<code>__end1</code>	End iterator of first sequence.
<code>__begin2</code>	Begin iterator of second sequence.
<code>__end2</code>	End iterator of second sequence.
<code>__target</code>	Target begin iterator.
<code>__max_length</code>	Maximum number of elements to merge.
<code>__comp</code>	Comparator.

Returns

Output end iterator.

Definition at line 223 of file `merge.h`.

References `std::make_pair()`, `multiway_merge_exact_splitting()`, and `parallel_multiway_merge()`.

3.6.4.25 `template<typename _RAIter, typename _Compare> void __gnu_parallel::__parallel_nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end, _Compare __comp)`

Parallel implementation of `std::nth_element()`.

Parameters

<code>__begin</code>	Begin iterator of input sequence.
<code>__nth</code>	Iterator of element that must be in position afterwards.
<code>__end</code>	End iterator of input sequence.
<code>__comp</code>	Comparator.

Definition at line 332 of file `partition.h`.

References `__parallel_partition()`, `_GLIBCXX_CALL`, `__gnu_parallel::_Settings::get()`, `std::max()`, `__gnu_parallel::_Settings::nth_element_minimal_n`, and `__gnu_parallel::_Settings::partition_minimal_n`.

Referenced by `__parallel_partial_sort()`.

3.6.4.26 `template<typename _RAIter, typename _Compare> void __gnu_parallel::__parallel_partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end, _Compare __comp)`

Parallel implementation of `std::partial_sort()`.

Parameters

<code>__begin</code>	Begin iterator of input sequence.
<code>__middle</code>	Sort until this position.
<code>__end</code>	End iterator of input sequence.
<code>__comp</code>	Comparator.

Definition at line 422 of file `partition.h`.

References `__parallel_nth_element()`.

3.6.4.27 `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation> _OutputIterator __gnu_parallel::__parallel_partial_sum (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op)`

Parallel partial sum front-__end.

Parameters

<code>__begin</code>	Begin iterator of input sequence.
<code>__end</code>	End iterator of input sequence.
<code>__result</code>	Begin iterator of output sequence.
<code>__bin_op</code>	Associative binary function.

Returns

End iterator of output sequence.

Definition at line 205 of file `partial_sum.h`.

References `__parallel_partial_sum_linear()`, `_GLIBCXX_CALL`, and `__gnu_parallel::_Settings::get()`.

```
3.6.4.28 template<typename _Iter , typename _OutputIterator , typename _BinaryOperation > _OutputIterator
    __gnu_parallel::_parallel_partial_sum_basecase ( _Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation
    __bin_op, typename std::iterator_traits< _Iter >::value_type __value )
```

Base case prefix sum routine.

Parameters

<code>__begin</code>	Begin iterator of input sequence.
<code>__end</code>	End iterator of input sequence.
<code>__result</code>	Begin iterator of output sequence.
<code>__bin_op</code>	Associative binary function.
<code>__value</code>	Start value. Must be passed since the neutral element is unknown in general.

Returns

End iterator of output sequence.

Definition at line 58 of file `partial_sum.h`.

Referenced by `__parallel_partial_sum_linear()`.

```
3.6.4.29 template<typename _Iter , typename _OutputIterator , typename _BinaryOperation > _OutputIterator
    __gnu_parallel::_parallel_partial_sum_linear ( _Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation
    __bin_op, typename std::iterator_traits< _Iter >::difference_type __n )
```

Parallel partial sum implementation, two-phase approach, no recursion.

Parameters

<code>__begin</code>	Begin iterator of input sequence.
<code>__end</code>	End iterator of input sequence.
<code>__result</code>	Begin iterator of output sequence.
<code>__bin_op</code>	Associative binary function.
<code>__n</code>	Length of sequence.

Returns

End iterator of output sequence.

Definition at line 89 of file `partial_sum.h`.

References `__equally_split()`, `__parallel_partial_sum_basecase()`, `std::accumulate()`, `__gnu_parallel::_Settings::get()`, and `__gnu_parallel::_Settings::partial_sum_dilation`.

Referenced by `__parallel_partial_sum()`.

3.6.4.30 `template<typename _RAIter, typename _Predicate> std::iterator_traits<_RAIter>::difference_type
__gnu_parallel::_parallel_partition (_RAIter __begin, _RAIter __end, _Predicate __pred, _ThreadIndex __num_threads
)`

Parallel implementation of `std::partition`.

Parameters

<code>__begin</code>	Begin iterator of input sequence to split.
<code>__end</code>	End iterator of input sequence to split.
<code>__pred</code>	Partition predicate, possibly including some kind of pivot.
<code>__num_threads</code>	Maximum number of threads to use for this task.

Returns

Number of elements not fulfilling the predicate.

Definition at line 56 of file `partition.h`.

References `__compare_and_swap()`, `__fetch_and_add()`, `_GLIBCXX_CALL`, `_GLIBCXX_VOLATILE`, `__gnu_parallel::_Settings::get()`, `__gnu_parallel::_Settings::partition_chunk_share`, and `__gnu_parallel::_Settings::partition_chunk_size`.

Referenced by `__parallel_nth_element()`, `__parallel_sort_qs_divide()`, and `__qsb_divide()`.

3.6.4.31 `template<typename _RAIter, typename _RandomNumberGenerator> void __gnu_parallel::_parallel_random_shuffle (
_RAIter __begin, _RAIter __end, _RandomNumberGenerator __rng = _RandomNumber()) [inline]`

Parallel random public call.

Parameters

<code>__begin</code>	Begin iterator of sequence.
<code>__end</code>	End iterator of sequence.
<code>__rng</code>	Random number generator to use.

Definition at line 522 of file `random_shuffle.h`.

References `__parallel_random_shuffle_drs()`.

```
3.6.4.32 template<typename _RAIter, typename _RandomNumberGenerator > void __gnu_parallel::__parallel_random_shuffle_drs
( _RAIter __begin, _RAIter __end, typename std::iterator_traits< _RAIter >::difference_type __n, _ThreadIndex
  __num_threads, _RandomNumberGenerator & __rng )
```

Main parallel random shuffle step.

Parameters

<code>__begin</code>	Begin iterator of sequence.
<code>__end</code>	End iterator of sequence.
<code>__n</code>	Length of sequence.
<code>__num_threads</code>	Number of threads to use.
<code>__rng</code>	Random number generator to use.

Definition at line 265 of file `random_shuffle.h`.

References `__gnu_parallel::DRSSorterPU< _RAIter, _RandomNumberGenerator >::__bins_end`, `__parallel_random_shuffle_drs_pu()`, `__rd_log2()`, `__round_up_to_pow2()`, `__sequential_random_shuffle()`, `_GLIBCXX_CALL`, `__gnu_parallel::DRandomShufflingGlobalData< _RAIter >::M_bin_proc`, `__gnu_parallel::DRSSorterPU< _RAIter, _RandomNumberGenerator >::M_bins_begin`, `__gnu_parallel::DRandomShufflingGlobalData< _RAIter >::M_dist`, `__gnu_parallel::DRandomShufflingGlobalData< _RAIter >::M_num_bins`, `__gnu_parallel::DRandomShufflingGlobalData< _RAIter >::M_num_bits`, `__gnu_parallel::DRSSorterPU< _RAIter, _RandomNumberGenerator >::M_num_threads`, `__gnu_parallel::DRSSorterPU< _RAIter, _RandomNumberGenerator >::M_sd`, `__gnu_parallel::DRSSorterPU< _RAIter, _RandomNumberGenerator >::M_seed`, `__gnu_parallel::DRandomShufflingGlobalData< _RAIter >::M_starts`, `__gnu_parallel::DRandomShufflingGlobalData< _RAIter >::M_temporaries`, `__gnu_parallel::Settings::get()`, `__gnu_parallel::Settings::L2_cache_size`, `std::max()`, `std::min()`, and `__gnu_parallel::Settings::TLB_size`.

Referenced by `__parallel_random_shuffle()`.

```
3.6.4.33 template<typename _RAIter, typename _RandomNumberGenerator > void __gnu_parallel::__parallel_random_shuffle_drs_pu (
  _DRSSorterPU< _RAIter, _RandomNumberGenerator > * __pus
)
```

Random shuffle code executed by each thread.

Parameters

<code>__pus</code>	Array of thread-local data records.
--------------------	-------------------------------------

Definition at line 122 of file `random_shuffle.h`.

References `__random_number_pow2()`, `__sequential_random_shuffle()`, `__gnu_parallel::DRandomShufflingGlobalData< _RAIter >::M_dist`, `__gnu_parallel::DRandomShufflingGlobalData< _RAIter >::M_num_bins`, `__gnu_parallel::DRandomShufflingGlobalData< _RAIter >::M_num_bits`, `__gnu_parallel::DRSSorterPU< _RAIter, _RandomNumberGenerator >::M_num_threads`, `__gnu_parallel::DRSSorterPU< _RAIter, _RandomNumberGenerator >::M_sd`, `__gnu_parallel::DRSSorterPU< _RAIter, _RandomNumberGenerator >::M_seed`, `__gnu_parallel::DRandomShufflingGlobalData< _RAIter >::M_starts`, and `std::partial_sum()`.

Referenced by `__parallel_random_shuffle_drs()`.

3.6.4.34 `template<bool __stable, typename _RAIter, typename _Compare> void __gnu_parallel::__parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, multiway_mergesort_tag __parallelism) [inline]`

Choose multiway mergesort, splitting variant at run-time, for parallel sorting.

Parameters

<code>__begin</code>	Begin iterator of input sequence.
<code>__end</code>	End iterator of input sequence.
<code>__comp</code>	Comparator.

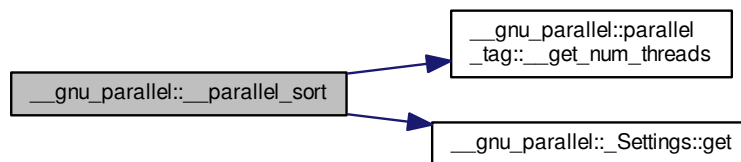
Template Parameters

<code>__stable</code>	Sort stable.
-----------------------	--------------

Definition at line 75 of file `sort.h`.

References `__gnu_parallel::parallel_tag::__get_num_threads()`, `_GLIBCXX_CALL`, and `__gnu_parallel::_Settings::__get()`.

Here is the call graph for this function:



3.6.4.35 `template<bool __stable, typename _RAIter, typename _Compare> void __gnu_parallel::__parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, multiway_mergesort_exact_tag __parallelism) [inline]`

Choose multiway mergesort with exact splitting, for parallel sorting.

Parameters

<code>__begin</code>	Begin iterator of input sequence.
<code>__end</code>	End iterator of input sequence.
<code>__comp</code>	Comparator.

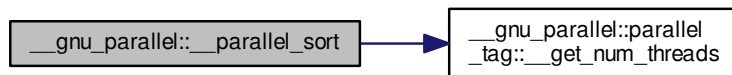
Template Parameters

<code>__stable</code>	Sort stable.
-----------------------	--------------

Definition at line 99 of file sort.h.

References `__gnu_parallel::parallel_tag::__get_num_threads()`, and `_GLIBCXX_CALL`.

Here is the call graph for this function:



3.6.4.36 `template<bool __stable, typename _RAIter, typename _Compare> void __gnu_parallel::__parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, multiway_mergesort_sampling_tag __parallelism) [inline]`

Choose multiway mergesort with splitting by sampling, for parallel sorting.

Parameters

<code>__begin</code>	Begin iterator of input sequence.
<code>__end</code>	End iterator of input sequence.
<code>__comp</code>	Comparator.

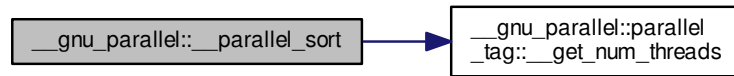
Template Parameters

<code>__stable</code>	Sort stable.
-----------------------	--------------

Definition at line 120 of file sort.h.

References `__gnu_parallel::parallel_tag::__get_num_threads()`, and `_GLIBCXX_CALL`.

Here is the call graph for this function:



3.6.4.37 `template<bool __stable, typename _RAIter, typename _Compare> void __gnu_parallel::__parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, quicksort_tag __parallelism) [inline]`

Choose quicksort for parallel sorting.

Parameters

<code>__begin</code>	Begin iterator of input sequence.
<code>__end</code>	End iterator of input sequence.
<code>__comp</code>	Comparator.

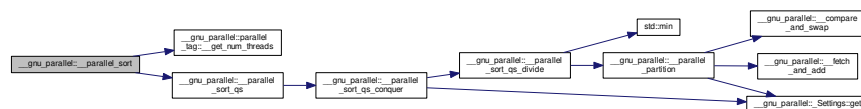
Template Parameters

<code>__stable</code>	Sort stable.
-----------------------	--------------

Definition at line 140 of file sort.h.

References `__gnu_parallel::parallel_tag::__get_num_threads()`, `__parallel_sort_qs()`, and `_GLIBCXX_CALL`.

Here is the call graph for this function:



3.6.4.38 `template<bool __stable, typename _RAIter, typename _Compare> void __gnu_parallel::__parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, balanced_quicksort_tag __parallelism) [inline]`

Choose balanced quicksort for parallel sorting.

Parameters

<code>__begin</code>	Begin iterator of input sequence.
<code>__end</code>	End iterator of input sequence.
<code>__comp</code>	Comparator.

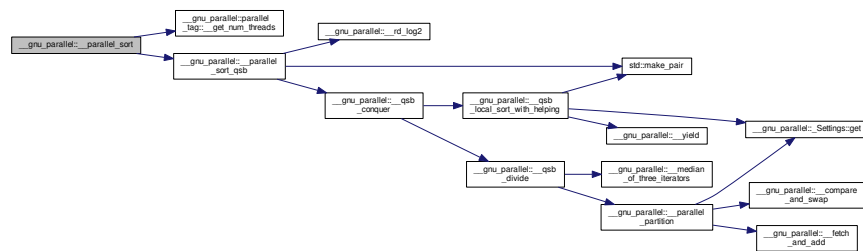
Template Parameters

<code>__stable</code>	Sort stable.
-----------------------	--------------

Definition at line 161 of file sort.h.

References `__gnu_parallel::parallel_tag::__get_num_threads()`, `__parallel_sort_qsb()`, and `_GLIBCXX_CALL`.

Here is the call graph for this function:



3.6.4.39 `template<bool __stable, typename _RAIter, typename _Compare> void __gnu_parallel::__parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, default_parallel_tag __parallelism) [inline]`

Choose multiway mergesort with exact splitting, for parallel sorting.

Parameters

<code>__begin</code>	Begin iterator of input sequence.
<code>__end</code>	End iterator of input sequence.
<code>__comp</code>	Comparator.

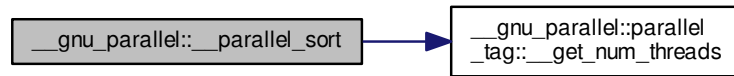
Template Parameters

<code>__stable</code>	Sort stable.
-----------------------	--------------

Definition at line 183 of file sort.h.

References `__gnu_parallel::parallel_tag::__get_num_threads()`, and `_GLIBCXX_CALL`.

Here is the call graph for this function:



3.6.4.40 `template<bool __stable, typename _RAIter, typename _Compare> void __gnu_parallel::__parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, parallel_tag __parallelism) [inline]`

Choose a parallel sorting algorithm.

Parameters

<code>__begin</code>	Begin iterator of input sequence.
<code>__end</code>	End iterator of input sequence.
<code>__comp</code>	Comparator.

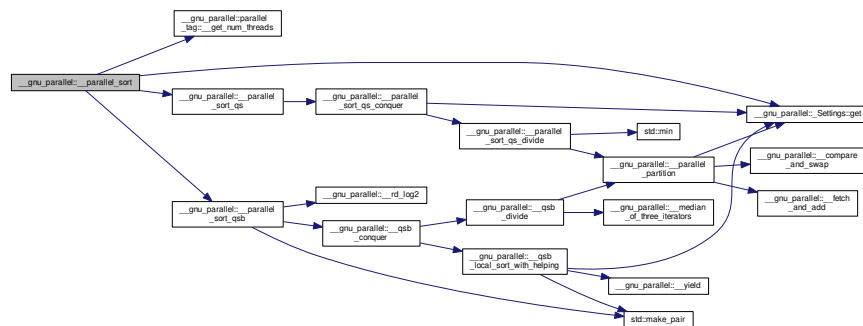
Template Parameters

<code>__stable</code>	Sort stable.
-----------------------	--------------

Definition at line 203 of file `sort.h`.

References `__gnu_parallel::parallel_tag::__get_num_threads()`, `__parallel_sort_qs()`, `__parallel_sort_qsb()`, `_GLIBCXX_CALL`, and `__gnu_parallel::_Settings::get()`.

Here is the call graph for this function:



3.6.4.41 `template<typename _RAIter, typename _Compare> void __gnu_parallel::__parallel_sort_qs (_RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __num_threads)`

Unbalanced quicksort main call.

Parameters

<code>__begin</code>	Begin iterator of input sequence.
<code>__end</code>	End iterator input sequence, ignored.
<code>__comp</code>	Comparator.
<code>__num_threads</code>	Number of threads that are allowed to work on this part.

Definition at line 154 of file quicksort.h.

References `__parallel_sort_qs_conquer()`, and `_GLIBCXX_CALL`.

Referenced by `__parallel_sort()`.

3.6.4.42 `template<typename _RAIter, typename _Compare> void __gnu_parallel::__parallel_sort_qs_conquer (_RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __num_threads)`

Unbalanced quicksort conquer step.

Parameters

<code>__begin</code>	Begin iterator of subsequence.
<code>__end</code>	End iterator of subsequence.
<code>__comp</code>	Comparator.
<code>__num_threads</code>	Number of threads that are allowed to work on this part.

Definition at line 101 of file quicksort.h.

References `__parallel_sort_qs_divide()`, and `__gnu_parallel::_Settings::get()`.

Referenced by `__parallel_sort_qs()`.

3.6.4.43 `template<typename _RAIter, typename _Compare> std::iterator_traits<_RAIter>::difference_type __gnu_parallel::__parallel_sort_qs_divide (_RAIter __begin, _RAIter __end, _Compare __comp, typename std::iterator_traits<_RAIter>::difference_type __pivot_rank, typename std::iterator_traits<_RAIter>::difference_type __num_samples, _ThreadIndex __num_threads)`

Unbalanced quicksort divide step.

Parameters

<code>__begin</code>	Begin iterator of subsequence.
<code>__end</code>	End iterator of subsequence.
<code>__comp</code>	Comparator.
<code>__pivot_rank</code>	Desired rank of the pivot.
<code>__num_samples</code>	Choose pivot from that many samples.
<code>__num_threads</code>	Number of threads that are allowed to work on this part.

Definition at line 51 of file quicksort.h.

References `__parallel_partition()`, and `std::min()`.

Referenced by `__parallel_sort_qs_conquer()`.

3.6.4.44 `template<typename _RAIter, typename _Compare> void __gnu_parallel::__parallel_sort_qsb (_RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __num_threads)`

Top-level quicksort routine.

Parameters

<code>__begin</code>	Begin iterator of sequence.
<code>__end</code>	End iterator of sequence.
<code>__comp</code>	Comparator.
<code>__num_threads</code>	Number of threads that are allowed to work on this part.

Definition at line 430 of file `balanced_quicksort.h`.

References `__qsb_conquer()`, `__rd_log2()`, `_GLIBCXX_CALL`, `__gnu_parallel::__QSBThreadLocal<_RAIter>::__M_←elements_leftover`, `__gnu_parallel::__QSBThreadLocal<_RAIter>::__M_leftover_parts`, and `std::make_pair()`.

Referenced by `__parallel_sort()`.

3.6.4.45 `template<typename _Iter, class _OutputIterator, class _BinaryPredicate> _OutputIterator __gnu_parallel::__parallel_unique_copy (_Iter __first, _Iter __last, _OutputIterator __result, _BinaryPredicate __binary_pred)`

Parallel `std::unique_copy()`, w/ `__o` explicit equality predicate.

Parameters

<code>__first</code>	Begin iterator of input sequence.
<code>__last</code>	End iterator of input sequence.
<code>__result</code>	Begin iterator of result <code>__sequence</code> .
<code>__binary_pred</code>	Equality predicate.

Returns

End iterator of result `__sequence`.

Definition at line 50 of file `unique_copy.h`.

References `__equally_split()`, and `_GLIBCXX_CALL`.

Referenced by `__parallel_unique_copy()`.

3.6.4.46 `template<typename _Iter, class _OutputIterator > _OutputIterator __gnu_parallel::__parallel_unique_copy (_Iter __first, _Iter __last, _OutputIterator __result) [inline]`

Parallel `std::unique_copy()`, without explicit equality predicate.

Parameters

<code>__first</code>	Begin iterator of input sequence.
<code>__last</code>	End iterator of input sequence.
<code>__result</code>	Begin iterator of result sequence.

Returns

End iterator of result sequence.

Definition at line 186 of file `unique_copy.h`.

References `__parallel_unique_copy()`.

3.6.4.47 `template<typename _RAIter, typename _Compare > void __gnu_parallel::__qsb_conquer (_QSBThreadLocal< _RAIter > ** __tls, _RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __iam, _ThreadIndex __num_threads, bool __parent_wait)`

Quicksort conquer step.

Parameters

<code>__tls</code>	Array of thread-local storages.
<code>__begin</code>	Begin iterator of subsequence.
<code>__end</code>	End iterator of subsequence.
<code>__comp</code>	Comparator.
<code>__iam</code>	Number of the thread processing this function.
<code>__num_threads</code>	Number of threads that are allowed to work on this part.

Definition at line 171 of file `balanced_quicksort.h`.

References `__qsb_divide()`, `__qsb_local_sort_with_helping()`, `__gnu_parallel::__QSBThreadLocal< _RAIter >::__M_↵ elements_leftover`, `__gnu_parallel::__QSBThreadLocal< _RAIter >::__M_initial`, `std::pair< _T1, _T2 >::first`, and `std::↵ ::pair< _T1, _T2 >::second`.

Referenced by `__parallel_sort_qsb()`.

3.6.4.48 `template<typename _RAIter, typename _Compare > std::iterator_traits< _RAIter >::difference_type __gnu_parallel::__qsb_divide (_RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __num_threads)`

Balanced quicksort divide step.

Parameters

<code>__begin</code>	Begin iterator of subsequence.
<code>__end</code>	End iterator of subsequence.
<code>__comp</code>	Comparator.
<code>__num_threads</code>	Number of threads that are allowed to work on this part.

Precondition

`(__end-__begin)>=1`

Definition at line 100 of file `balanced_quicksort.h`.

References `__median_of_three_iterators()`, and `__parallel_partition()`.

Referenced by `__qsb_conquer()`.

3.6.4.49 `template<typename _RAIter, typename _Compare> void __gnu_parallel::__qsb_local_sort_with_helping (`
`__QSBThreadLocal<_RAIter> ** __tls, _Compare & __comp, _ThreadIndex __iam, bool __wait)`

Quicksort step doing load-balanced local sort.

Parameters

<code>__tls</code>	Array of thread-local storages.
<code>__comp</code>	Comparator.
<code>__iam</code>	Number of the thread processing this function.

Definition at line 247 of file `balanced_quicksort.h`.

References `__yield()`, `_GLIBCXX_ASSERTIONS`, `__gnu_parallel::__QSBThreadLocal<_RAIter>::_M_elements_↵`
`leftover`, `__gnu_parallel::__QSBThreadLocal<_RAIter>::_M_initial`, `__gnu_parallel::__QSBThreadLocal<_RAIter>::_↵`
`_M_leftover_parts`, `__gnu_parallel::__QSBThreadLocal<_RAIter>::_M_num_threads`, `__gnu_parallel::_Settings::get()`,
`std::make_pair()`, and `__gnu_parallel::_Settings::sort_qsb_base_case_maximal_n`.

Referenced by `__qsb_conquer()`.

3.6.4.50 `template<typename _RandomNumberGenerator> int __gnu_parallel::__random_number_pow2 (int __logp,`
`_RandomNumberGenerator & __rng) [inline]`

Generate a random number in $[0, 2^{\text{__logp}})$.

Parameters

<code>__logp</code>	Logarithm (basis 2) of the upper range __bound.
<code>__rng</code>	Random number generator to use.

Definition at line 115 of file random_shuffle.h.

Referenced by `__parallel_random_shuffle_drs_pu()`, and `__sequential_random_shuffle()`.

3.6.4.51 `template<typename _Size > _Size __gnu_parallel::__rd_log2 (_Size __n) [inline]`

Calculates the rounded-down logarithm of `__n` for base 2.

Parameters

<code>__n</code>	Argument.
------------------	-----------

Returns

Returns 0 for any argument < 1 .

Definition at line 102 of file parallel/base.h.

Referenced by `__parallel_random_shuffle_drs()`, `__parallel_sort_qsb()`, `__round_up_to_pow2()`, `__sequential_random_shuffle()`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_LoserTreeBase()`, `multiseq_partition()`, and `multiseq_selection()`.

3.6.4.52 `template<typename _Tp > _Tp __gnu_parallel::__round_up_to_pow2 (_Tp __x)`

Round up to the next greater power of 2.

Parameters

<code>__x</code>	Integer to round up
------------------	---------------------

Definition at line 248 of file random_shuffle.h.

References `__rd_log2()`.

Referenced by `__parallel_random_shuffle_drs()`, `__sequential_random_shuffle()`, and `multiseq_selection()`.

3.6.4.53 `template<typename _RAIter1, typename _RAIter2, typename _Pred > _RAIter1 __gnu_parallel::__search_template (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Pred __pred)`

Parallel `std::search`.

Parameters

<code>__begin1</code>	Begin iterator of first sequence.
<code>__end1</code>	End iterator of first sequence.
<code>__begin2</code>	Begin iterator of second sequence.
<code>__end2</code>	End iterator of second sequence.
<code>__pred</code>	Find predicate.

Returns

Place of finding in first sequences.

Definition at line 81 of file search.h.

References `__calc_borders()`, `__equally_split()`, `_GLIBCXX_CALL`, and `std::min()`.

```
3.6.4.54 template<bool __stable, bool __sentinels, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp,
typename _Compare > _RAIter3 __gnu_parallel::__sequential_multiway_merge ( _RAIterIterator __seqs_begin,
_RAlterIterator __seqs_end, _RAIter3 __target, const typename std::iterator_traits< typename std::iterator_traits<
_RAlterIterator >::value_type::first_type >::value_type & __sentinel, _DifferenceTp __length, _Compare __comp )
```

Sequential multi-way merging switch.

The `_GLIBCXX_PARALLEL_DECISION` is based on the branching factor and runtime settings.

Parameters

<code>__seqs_begin</code>	Begin iterator of iterator pair input sequence.
<code>__seqs_end</code>	End iterator of iterator pair input sequence.
<code>__target</code>	Begin iterator of output sequence.
<code>__comp</code>	Comparator.
<code>__length</code>	Maximum length to merge, possibly larger than the number of elements available.
<code>__sentinel</code>	The sequences have <code>__a</code> <code>__sentinel</code> element.

Returns

End iterator of output sequence.

Definition at line 920 of file multiway_merge.h.

References `__is_sorted()`, `__merge_advance()`, `_GLIBCXX_CALL`, and `_GLIBCXX_PARALLEL_LENGTH`.

Referenced by `multiway_merge()`, and `multiway_merge_sentinels()`.

```
3.6.4.55 template<typename _RAIter, typename _RandomNumberGenerator > void __gnu_parallel::__sequential_random_shuffle (
_RAlter __begin, _RAIter __end, _RandomNumberGenerator & __rng )
```

Sequential cache-efficient random shuffle.

Parameters

<code>__begin</code>	Begin iterator of sequence.
<code>__end</code>	End iterator of sequence.
<code>__rng</code>	Random number generator to use.

Definition at line 410 of file random_shuffle.h.

References `__random_number_pow2()`, `__rd_log2()`, `__round_up_to_pow2()`, `__gnu_parallel::Settings::get()`, `__gnu_parallel::Settings::L2_cache_size`, `std::min()`, `std::partial_sum()`, and `__gnu_parallel::Settings::TLB_size`.

Referenced by `__parallel_random_shuffle_drs()`, and `__parallel_random_shuffle_drs_pu()`.

3.6.4.56 `template<typename _Iter > void __gnu_parallel::__shrink (std::vector< _Iter > & __os_starts, size_t & __count_to_two, size_t & __range_length)`

Combines two ranges into one and thus halves the number of ranges.

Parameters

<code>__os_starts</code>	Start positions worked on (oversampled).
<code>__count_to_two</code>	Counts up to 2.
<code>__range_length</code>	Current length of a chunk.

Definition at line 70 of file `list_partition.h`.

References `std::vector< _Tp, _Alloc >::size()`.

Referenced by `__shrink_and_double()`.

3.6.4.57 `template<typename _Iter > void __gnu_parallel::__shrink_and_double (std::vector< _Iter > & __os_starts, size_t & __count_to_two, size_t & __range_length, const bool __make_twice)`

Shrinks and doubles the ranges.

Parameters

<code>__os_starts</code>	Start positions worked on (oversampled).
<code>__count_to_two</code>	Counts up to 2.
<code>__range_length</code>	Current length of a chunk.
<code>__make_twice</code>	Whether the <code>__os_starts</code> is allowed to be grown or not

Definition at line 50 of file `list_partition.h`.

References `__shrink()`, `std::vector< _Tp, _Alloc >::resize()`, and `std::vector< _Tp, _Alloc >::size()`.

Referenced by `list_partition()`.

3.6.4.58 `void __gnu_parallel::__yield () [inline]`

Yield control to another thread, without waiting for the end of the time slice.

Definition at line 121 of file `parallel/compatibility.h`.

Referenced by `__for_each_template_random_access_workstealing()`, and `__qsb_local_sort_with_helping()`.

3.6.4.59 `template<typename _Iter, typename _FunctorType> size_t __gnu_parallel::list_partition (const _Iter __begin, const _Iter __end, _Iter * __starts, size_t * __lengths, const int __num_parts, _FunctorType & __f, int __oversampling = 0)`

Splits a sequence given by input iterators into parts of almost equal size.

The function needs only one pass over the sequence.

Parameters

<code>__begin</code>	Begin iterator of input sequence.
<code>__end</code>	End iterator of input sequence.
<code>__starts</code>	Start iterators for the resulting parts, dimension <code>__num_parts+1</code> . For convenience, <code>__starts [__num_parts]</code> contains the end iterator of the sequence.
<code>__lengths</code>	Length of the resulting parts.
<code>__num_parts</code>	Number of parts to split the sequence into.
<code>__f</code>	Functor to be applied to each element by traversing <code>__it</code>
<code>__oversampling</code>	Oversampling factor. If 0, then the partitions will differ in at most $\{ \{ _end \} - \{ _begin \} \}$ elements. Otherwise, the ratio between the longest and the shortest part is bounded by $1 / (\{ _oversampling \} \{ num \})$

Returns

Length of the whole sequence.

Definition at line 101 of file `list_partition.h`.

References `__shrink_and_double()`, and `std::vector<_Tp, _Alloc>::size()`.

3.6.4.60 `template<typename _Tp> const _Tp& __gnu_parallel::max (const _Tp & __a, const _Tp & __b) [inline]`

Equivalent to `std::max`.

Definition at line 150 of file `parallel/base.h`.

3.6.4.61 `template<typename _Tp> const _Tp& __gnu_parallel::min (const _Tp & __a, const _Tp & __b) [inline]`

Equivalent to `std::min`.

Definition at line 144 of file `parallel/base.h`.

Referenced by `__for_each_template_random_access_workstealing()`.

3.6.4.62 `template<typename _RanSeqs, typename _RankType, typename _RankIterator, typename _Compare> void __gnu_parallel::multiseq_partition (_RanSeqs __begin_seqs, _RanSeqs __end_seqs, _RankType __rank, _RankIterator __begin_offsets, _Compare __comp = std::less< typename std::iterator_traits<typename std::iterator<_traits<_RanSeqs>::value_type::first_type>::value_type>()>())`

Splits several sorted sequences at a certain global `__rank`, resulting in a splitting point for each sequence. The sequences are passed via a sequence of random-access iterator pairs, none of the sequences may be empty. If there are several equal elements across the split, the ones on the `__left` side will be chosen from sequences with smaller number.

Parameters

<code>__begin_seqs</code>	Begin of the sequence of iterator pairs.
<code>__end_seqs</code>	End of the sequence of iterator pairs.
<code>__rank</code>	The global rank to partition at.
<code>__begin_offsets</code>	A random-access <code>__sequence</code> <code>__begin</code> where the <code>__result</code> will be stored in. Each element of the sequence is an iterator that points to the first element on the greater part of the respective <code>__sequence</code> .
<code>__comp</code>	The ordering functor, defaults to <code>std::less<_Tp></code> .

Definition at line 122 of file `multiseq_selection.h`.

References `__rd_log2()`, `_GLIBCXX_CALL`, `std::vector<_Tp, _Alloc>::begin()`, `std::distance()`, `std::priority_queue<_Tp, _Sequence, _Compare>::empty()`, `std::vector<_Tp, _Alloc>::end()`, `std::make_pair()`, `std::max()`, `std::min()`, `std::priority_queue<_Tp, _Sequence, _Compare>::pop()`, `std::priority_queue<_Tp, _Sequence, _Compare>::push()`, `std::vector<_Tp, _Alloc>::push_back()`, and `std::priority_queue<_Tp, _Sequence, _Compare>::top()`.

Referenced by `multiway_merge_exact_splitting()`.

```
3.6.4.63 template<typename _Tp, typename _RanSeqs, typename _RankType, typename _Compare> _Tp
    __gnu_parallel::multiseq_selection ( _RanSeqs __begin_seqs, _RanSeqs __end_seqs, _RankType __rank, _RankType &
    __offset, _Compare __comp = std::less<_Tp>() )
```

Selects the element at a certain global `__rank` from several sorted sequences.

The sequences are passed via a sequence of random-access iterator pairs, none of the sequences may be empty.

Parameters

<code>__begin_seqs</code>	Begin of the sequence of iterator pairs.
<code>__end_seqs</code>	End of the sequence of iterator pairs.
<code>__rank</code>	The global rank to partition at.
<code>__offset</code>	The rank of the selected element in the global subsequence of elements equal to the selected element. If the selected element is unique, this number is 0.
<code>__comp</code>	The ordering functor, defaults to <code>std::less</code> .

Definition at line 388 of file `multiseq_selection.h`.

References `__rd_log2()`, `__round_up_to_pow2()`, `_GLIBCXX_CALL`, `std::vector<_Tp, _Alloc>::begin()`, `std::distance()`, `std::priority_queue<_Tp, _Sequence, _Compare>::empty()`, `std::vector<_Tp, _Alloc>::end()`, `std::make_pair()`, `std::max()`, `std::min()`, `std::priority_queue<_Tp, _Sequence, _Compare>::pop()`, `std::priority_queue<_Tp, _Sequence, _Compare>::push()`, `std::vector<_Tp, _Alloc>::push_back()`, and `std::priority_queue<_Tp, _Sequence, _Compare>::top()`.

```
3.6.4.64 template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
    _RAIterOut __gnu_parallel::multiway_merge ( _RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end,
    _RAIterOut __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::sequential_tag )
```

Multiway Merge Frontend.

Merge the sequences specified by `seqs_begin` and `__seqs_end` into `__target`. `__seqs_begin` and `__seqs_end` must point to a sequence of pairs. These pairs must contain an iterator to the beginning of a sequence in their first entry and an iterator the `_M_end` of the same sequence in their second entry.

Ties are broken arbitrarily. See `stable_multiway_merge` for a variant that breaks ties by sequence number but is slower.

The first entries of the pairs (i.e. the begin iterators) will be moved forward.

The output sequence has to provide enough space for all elements that are written to it.

This function will merge the input sequences:

- not stable
- parallel, depending on the input size and Settings
- using sampling for splitting
- not using sentinels

Example:

```
int sequences[10][10];
for (int __i = 0; __i < 10; ++__i)
    for (int __j = 0; __j < 10; ++__j)
        sequences[__i][__j] = __j;

int __out[33];
std::vector<std::pair<int*> > seqs;
for (int __i = 0; __i < 10; ++__i)
    { seqs.push(std::make_pair<int*>(sequences[__i],
                                     sequences[__i] + 10)) }

multiway_merge(seqs.begin(), seqs.end(), __target, std::less<int>(), 33);
```

See also

`stable_multiway_merge`

Precondition

All input sequences must be sorted.

Target must provide enough space to merge out length elements or the number of elements in all sequences, whichever is smaller.

Postcondition

`[__target, return __value)` contains merged `__elements` from the input sequences.
`return __value - __target = min(__length, number of elements in all sequences).`

Template Parameters

<code>_RAIterPairIterator</code>	iterator over sequence of pairs of iterators
<code>_RAIterOut</code>	iterator over target sequence
<code>_DifferenceTp</code>	difference type for the sequence
<code>_Compare</code>	strict weak ordering type to compare elements in sequences

Parameters

<code>__seqs_begin</code>	__begin of sequence __sequence
<code>__seqs_end</code>	__M_end of sequence __sequence
<code>__target</code>	target sequence to merge to.
<code>__comp</code>	strict weak ordering to use for element comparison.
<code>__length</code>	Maximum length to merge, possibly larger than the number of elements available.

Returns

`_M_end` iterator of output sequence

Definition at line 1418 of file `multiway_merge.h`.

References `__gnu_parallel::parallel_tag::get_num_threads()`, `__sequential_multiway_merge()`, `_GLIBCXX_CALL`, `↔` `_GLIBCXX_PARALLEL_CONDITION`, `__gnu_parallel::Settings::get()`, `multiway_merge_exact_splitting()`, `multiway_↔` `merge_sampling_splitting()`, and `parallel_multiway_merge()`.

3.6.4.65 `template<template< typename RAI, typename C > class iterator, typename _RAIterIterator , typename _RAIter3 , typename _DifferenceTp , typename _Compare > _RAIter3 __gnu_parallel::multiway_merge_3_variant (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp)`

Highly efficient 3-way merging procedure.

Merging is done with the algorithm implementation described by Peter Sanders. Basically, the idea is to minimize the number of necessary comparison after merging an element. The implementation trick that makes this fast is that the order of the sequences is stored in the instruction pointer (translated into labels in C++).

This works well for merging up to 4 sequences.

Note that making the merging stable does *not* come at a performance hit.

Whether the merging is done guarded or unguarded is selected by the used iterator class.

Parameters

<code>__seqs_begin</code>	Begin iterator of iterator pair input sequence.
<code>__seqs_end</code>	End iterator of iterator pair input sequence.
<code>__target</code>	Begin iterator of output sequence.
<code>__comp</code>	Comparator.
<code>__length</code>	Maximum length to merge, less equal than the total number of elements available.

Returns

End iterator of output sequence.

Definition at line 241 of file multiway_merge.h.

References `_GLIBCXX_CALL`.

3.6.4.66 `template<template< typename RAI, typename C > class iterator, typename _RAIterliterator , typename _RAIter3 , typename _DifferenceTp , typename _Compare > _RAIter3 __gnu_parallel::multiway_merge_4_variant (_RAIterliterator __seqs_begin, _RAIterliterator __seqs_end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp)`

Highly efficient 4-way merging procedure.

Merging is done with the algorithm implementation described by Peter Sanders. Basically, the idea is to minimize the number of necessary comparison after merging an element. The implementation trick that makes this fast is that the order of the sequences is stored in the instruction pointer (translated into goto labels in C++).

This works well for merging up to 4 sequences.

Note that making the merging stable does *not* come at a performance hit.

Whether the merging is done guarded or unguarded is selected by the used iterator class.

Parameters

<code>__seqs_begin</code>	Begin iterator of iterator pair input sequence.
<code>__seqs_end</code>	End iterator of iterator pair input sequence.
<code>__target</code>	Begin iterator of output sequence.
<code>__comp</code>	Comparator.
<code>__length</code>	Maximum length to merge, less equal than the total number of elements available.

Returns

End iterator of output sequence.

Definition at line 360 of file multiway_merge.h.

References `_GLIBCXX_CALL`.

3.6.4.67 `template<bool __stable, typename _RAIterliterator , typename _Compare , typename _DifferenceType > void __gnu_parallel::multiway_merge_exact_splitting (_RAIterliterator __seqs_begin, _RAIterliterator __seqs_end, _DifferenceType __length, _DifferenceType __total_length, _Compare __comp, std::vector< std::pair< _DifferenceType, _DifferenceType > > * __pieces)`

Exact splitting for parallel multiway-merge routine.

None of the passed sequences may be empty.

Definition at line 1120 of file multiway_merge.h.

References `__equally_split()`, `_GLIBCXX_PARALLEL_LENGTH`, `std::vector< _Tp, _Alloc >::begin()`, `std::vector< _Tp, _Alloc >::end()`, `multiseq_partition()`, and `std::vector< _Tp, _Alloc >::resize()`.

Referenced by `__parallel_merge_advance()`, `multiway_merge()`, and `multiway_merge_sentinels()`.

3.6.4.68 `template<typename _LT, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare > _RAIter3 __gnu_parallel::multiway_merge_loser_tree (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp)`

Multi-way merging procedure for a high branching factor, guarded case.

This merging variant uses a `LoserTree` class as selected by `_LT`.

Stability is selected through the used `LoserTree` class `_LT`.

At least one non-empty sequence is required.

Parameters

<code>__seqs_begin</code>	Begin iterator of iterator pair input sequence.
<code>__seqs_end</code>	End iterator of iterator pair input sequence.
<code>__target</code>	Begin iterator of output sequence.
<code>__comp</code>	Comparator.
<code>__length</code>	Maximum length to merge, less equal than the total number of elements available.

Returns

End iterator of output sequence.

Definition at line 491 of file `multiway_merge.h`.

References `_GLIBCXX_CALL`, and `_GLIBCXX_PARALLEL_LENGTH`.

3.6.4.69 `template<typename UnguardedLoserTree, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare > _RAIter3 __gnu_parallel::multiway_merge_loser_tree_sentinel (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator >::value_type::first_type >::value_type & __sentinel, _DifferenceTp __length, _Compare __comp)`

Multi-way merging procedure for a high branching factor, requiring sentinels to exist.

Template Parameters

<code>UnguardedLoserTree</code>	<code>_LoserTree</code> variant to use for the unguarded merging.
---------------------------------	---

Parameters

<code>__seqs_begin</code>	Begin iterator of iterator pair input sequence.
<code>__seqs_end</code>	End iterator of iterator pair input sequence.
<code>__target</code>	Begin iterator of output sequence.
<code>__comp</code>	Comparator.
<code>__length</code>	Maximum length to merge, less equal than the total number of elements available.

Returns

End iterator of output sequence.

Definition at line 662 of file `multiway_merge.h`.

References `__is_sorted()`, and `_GLIBCXX_CALL`.

```
3.6.4.70 template<typename _LT, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare
> _RAIter3 __gnu_parallel::multiway_merge_loser_tree_unguarded ( _RAIterIterator __seqs_begin, _RAIterIterator
__seqs_end, _RAIter3 __target, const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator
>::value_type::first_type >::value_type & __sentinel, _DifferenceTp __length, _Compare __comp )
```

Multi-way merging procedure for a high branching factor, unguarded case.

Merging is done using the `LoserTree` class `_LT`.

Stability is selected by the used `LoserTrees`.

Precondition

No input will run out of elements during the merge.

Parameters

<code>__seqs_begin</code>	Begin iterator of iterator pair input sequence.
<code>__seqs_end</code>	End iterator of iterator pair input sequence.
<code>__target</code>	Begin iterator of output sequence.
<code>__comp</code>	Comparator.
<code>__length</code>	Maximum length to merge, less equal than the total number of elements available.

Returns

End iterator of output sequence.

Definition at line 574 of file `multiway_merge.h`.

References `_GLIBCXX_CALL`.

```
3.6.4.71 template<bool __stable, typename _RAIterIterator, typename _Compare, typename _DifferenceType > void
__gnu_parallel::multiway_merge_sampling_splitting ( _RAIterIterator __seqs_begin, _RAIterIterator __seqs_end,
_DifferenceType __length, _DifferenceType __total_length, _Compare __comp, std::vector< std::pair<
_DifferenceType, _DifferenceType > > * __pieces )
```

Sampling based splitting for parallel multiway-merge routine.

Definition at line 1035 of file `multiway_merge.h`.

References `_GLIBCXX_PARALLEL_LENGTH`, `__gnu_parallel::_Settings::get()`, and `__gnu_parallel::_Settings::merge_oversampling`.

Referenced by `multiway_merge()`, and `multiway_merge_sentinels()`.

3.6.4.72 `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >
 _RAIterOut __gnu_parallel::multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end,
 _RAIterOut __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::sequential_tag)`

Multiway Merge Frontend.

Merge the sequences specified by `seqs_begin` and `__seqs_end` into `__target`. `__seqs_begin` and `__seqs_end` must point to a sequence of pairs. These pairs must contain an iterator to the beginning of a sequence in their first entry and an iterator the `_M_end` of the same sequence in their second entry.

Ties are broken arbitrarily. See `stable_multiway_merge` for a variant that breaks ties by sequence number but is slower.

The first entries of the pairs (i.e. the begin iterators) will be moved forward accordingly.

The output sequence has to provide enough space for all elements that are written to it.

This function will merge the input sequences:

- not stable
- parallel, depending on the input size and Settings
- using sampling for splitting
- using sentinels

You have to take care that the element the `_M_end` iterator points to is readable and contains a value that is greater than any other non-sentinel value in all sequences.

Example:

```
int sequences[10][11];
for (int __i = 0; __i < 10; ++__i)
    for (int __j = 0; __j < 11; ++__j)
        sequences[__i][__j] = __j; // __last one is sentinel!

int __out[33];
std::vector<std::pair<int*> > seqs;
for (int __i = 0; __i < 10; ++__i)
    { seqs.push(std::make_pair<int*>(sequences[__i],
                                     sequences[__i] + 10)) }

multiway_merge(seqs.begin(), seqs.end(), __target, std::less<int>(), 33);
```

Precondition

All input sequences must be sorted.

Target must provide enough space to merge out length elements or the number of elements in all sequences, whichever is smaller.

For each `__i`, `__seqs_begin[__i].second` must be the end marker of the sequence, but also reference the one more `__sentinel` element.

Postcondition

`[__target, return __value)` contains merged `__elements` from the input sequences.

`return __value - __target = min(__length, number of elements in all sequences).`

See also

`stable_multiway_merge_sentinels`

Template Parameters

<code>_RAIterPairIterator</code>	iterator over sequence of pairs of iterators
<code>_RAIterOut</code>	iterator over target sequence
<code>_DifferenceTp</code>	difference type for the sequence
<code>_Compare</code>	strict weak ordering type to compare elements in sequences

Parameters

<code>__seqs_begin</code>	begin of sequence <code>__sequence</code>
<code>__seqs_end</code>	<code>_M_end</code> of sequence <code>__sequence</code>
<code>__target</code>	target sequence to merge to.
<code>__comp</code>	strict weak ordering to use for element comparison.
<code>__length</code>	Maximum length to merge, possibly larger than the number of elements available.

Returns

`_M_end` iterator of output sequence

Definition at line 1782 of file `multiway_merge.h`.

References `__gnu_parallel::parallel_tag::__get_num_threads()`, `__sequential_multiway_merge()`, `_GLIBCXX_CALL`, `__GLIBCXX_PARALLEL_CONDITION`, `__gnu_parallel::Settings::get()`, `multiway_merge_exact_splitting()`, `multiway_merge_sampling_splitting()`, and `parallel_multiway_merge()`.

```
3.6.4.73 template<bool __stable, bool __sentinels, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp,
typename _Splitter, typename _Compare> _RAIter3 __gnu_parallel::parallel_multiway_merge ( _RAIterIterator
__seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _Splitter __splitter, _DifferenceTp __length, _Compare
__comp, _ThreadIndex __num_threads )
```

Parallel multi-way merge routine.

The `_GLIBCXX_PARALLEL_DECISION` is based on the branching factor and runtime settings.

Must not be called if the number of sequences is 1.

Template Parameters

<code>_Splitter</code>	functor to split input (either <code>__exact</code> or sampling based)
<code>__stable</code>	Stable merging incurs a performance penalty.
<code>__sentinel</code>	Ignored.

Parameters

<code>__seqs_begin</code>	Begin iterator of iterator pair input sequence.
<code>__seqs_end</code>	End iterator of iterator pair input sequence.

Parameters

<code>__target</code>	Begin iterator of output sequence.
<code>__comp</code>	Comparator.
<code>__length</code>	Maximum length to merge, possibly larger than the number of elements available.

Returns

End iterator of output sequence.

Definition at line 1225 of file `multiway_merge.h`.

References `__is_sorted()`, `_GLIBCXX_CALL`, `_GLIBCXX_PARALLEL_LENGTH`, `__gnu_parallel::_Settings::get()`, `std::make_pair()`, and `__gnu_parallel::_Settings::merge_oversampling`.

Referenced by `__parallel_merge_advance()`, `multiway_merge()`, and `multiway_merge_sentinels()`.

3.6.4.74 `template<bool __stable, bool __exact, typename _RAIter, typename _Compare > void __gnu_parallel::parallel_sort_mwms (_RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __num_threads)`

PMWMS main call.

Parameters

<code>__begin</code>	Begin iterator of sequence.
<code>__end</code>	End iterator of sequence.
<code>__comp</code>	Comparator.
<code>__num_threads</code>	Number of threads to use.

Definition at line 395 of file `multiway_mergesort.h`.

References `_GLIBCXX_CALL`, `__gnu_parallel::PMWMSortingData<_RAIter>::M_num_threads`, `__gnu_parallel::PMWMSortingData<_RAIter>::M_offsets`, `__gnu_parallel::PMWMSortingData<_RAIter>::M_pieces`, `__gnu_parallel::PMWMSortingData<_RAIter>::M_samples`, `__gnu_parallel::PMWMSortingData<_RAIter>::M_source`, `__gnu_parallel::PMWMSortingData<_RAIter>::M_starts`, `__gnu_parallel::PMWMSortingData<_RAIter>::M_temporary`, `__gnu_parallel::_Settings::get()`, and `__gnu_parallel::_Settings::sort_mwms_oversampling`.

3.6.4.75 `template<bool __stable, bool __exact, typename _RAIter, typename _Compare > void __gnu_parallel::parallel_sort_mwms_pu (_PMWMSortingData<_RAIter> * __sd, _Compare & __comp)`

PMWMS code executed by each thread.

Parameters

<code>__sd</code>	Pointer to algorithm data.
<code>__comp</code>	Comparator.

Definition at line 308 of file multiway_mergesort.h.

References `__gnu_parallel::PMWMSSortingData<_RAIter>::M_num_threads`, `__gnu_parallel::PMWMSSortingData<_RAIter>::M_pieces`, `__gnu_parallel::PMWMSSortingData<_RAIter>::M_source`, `__gnu_parallel::PMWMSSortingData<_RAIter>::M_starts`, `__gnu_parallel::PMWMSSortingData<_RAIter>::M_temporary`, `__gnu_parallel::Settings::get()`, `std::make_pair()`, `__gnu_parallel::Settings::sort_mwms_oversampling`, and `std::uninitialized_copy()`.

3.6.5 Variable Documentation

3.6.5.1 `const int __gnu_parallel::_CASable_bits` [static]

Number of bits of `_CASable`.

Definition at line 130 of file types.h.

Referenced by `__decode2()`, and `__encode2()`.

3.6.5.2 `const _CASable __gnu_parallel::_CASable_mask` [static]

`_CASable` with the right half of bits set to 1.

Definition at line 133 of file types.h.

Referenced by `__decode2()`.

3.7 __gnu_pbds Namespace Reference

Classes

- struct [associative_tag](#)
- class [basic_branch](#)
- struct [basic_branch_tag](#)
- class [basic_hash_table](#)
- struct [basic_hash_tag](#)
- struct [basic_invalidation_guarantee](#)
- struct [binary_heap_tag](#)
- struct [binomial_heap_tag](#)
- class [cc_hash_max_collision_check_resize_trigger](#)
- class [cc_hash_table](#)
- struct [cc_hash_tag](#)
- struct [container_error](#)
- struct [container_tag](#)
- struct [container_traits](#)
- struct [container_traits_base](#)
- struct [container_traits_base<binary_heap_tag>](#)
- struct [container_traits_base<binomial_heap_tag>](#)
- struct [container_traits_base<cc_hash_tag>](#)

- struct [container_traits_base< gp_hash_tag >](#)
- struct [container_traits_base< list_update_tag >](#)
- struct [container_traits_base< ov_tree_tag >](#)
- struct [container_traits_base< pairing_heap_tag >](#)
- struct [container_traits_base< pat_trie_tag >](#)
- struct [container_traits_base< rb_tree_tag >](#)
- struct [container_traits_base< rc_binomial_heap_tag >](#)
- struct [container_traits_base< splay_tree_tag >](#)
- struct [container_traits_base< thin_heap_tag >](#)
- class [direct_mask_range_hashing](#)
- class [direct_mod_range_hashing](#)
- class [gp_hash_table](#)
- struct [gp_hash_tag](#)
- class [hash_exponential_size_policy](#)
- class [hash_load_check_resize_trigger](#)
- class [hash_prime_size_policy](#)
- class [hash_standard_resize_policy](#)
- struct [insert_error](#)
- struct [join_error](#)
- class [linear_probe_fn](#)
- class [list_update](#)
- struct [list_update_tag](#)
- class [lu_counter_policy](#)
- class [lu_move_to_front_policy](#)
- struct [null_node_update](#)
- struct [null_type](#)
- struct [ov_tree_tag](#)
- struct [pairing_heap_tag](#)
- struct [pat_trie_tag](#)
- struct [point_invalidation_guarantee](#)
- class [priority_queue](#)
- struct [priority_queue_tag](#)
- class [quadratic_probe_fn](#)
- struct [range_invalidation_guarantee](#)
- struct [rb_tree_tag](#)
- struct [rc_binomial_heap_tag](#)
- struct [resize_error](#)
- class [sample_probe_fn](#)
- class [sample_range_hashing](#)
- class [sample_ranged_hash_fn](#)
- class [sample_ranged_probe_fn](#)
- class [sample_resize_policy](#)
- class [sample_resize_trigger](#)
- class [sample_size_policy](#)
- class [sample_tree_node_update](#)
- struct [sample_trie_access_traits](#)
- class [sample_trie_node_update](#)
- struct [sample_update_policy](#)
- struct [sequence_tag](#)
- struct [splay_tree_tag](#)
- struct [string_tag](#)

- struct [thin_heap_tag](#)
- class [tree](#)
- class [tree_order_statistics_node_update](#)
- struct [tree_tag](#)
- class [trie](#)
- class [trie_order_statistics_node_update](#)
- class [trie_prefix_search_node_update](#)
- struct [trie_string_access_traits](#)
- struct [trie_tag](#)
- struct [trivial_iterator_tag](#)

Typedefs

- typedef void [trivial_iterator_difference_type](#)

Functions

- void [__throw_container_error](#) ()
- void [__throw_insert_error](#) ()
- void [__throw_join_error](#) ()
- void [__throw_resize_error](#) ()

3.7.1 Detailed Description

GNU extensions for policy-based data structures for public use.

3.8 `__gnu_profile` Namespace Reference

Classes

- class [__container_size_info](#)
- class [__container_size_stack_info](#)
- class [__hashfunc_info](#)
- class [__hashfunc_stack_info](#)
- class [__list2vector_info](#)
- class [__map2umap_info](#)
- class [__map2umap_stack_info](#)
- class [__object_info_base](#)
- struct [__reentrance_guard](#)
- class [__stack_hash](#)
- class [__stack_info_base](#)
- class [__trace_base](#)
- class [__trace_container_size](#)
- class [__trace_hash_func](#)
- class [__trace_hashtable_size](#)
- class [__trace_map2umap](#)
- class [__trace_vector_size](#)
- class [__trace_vector_to_list](#)
- class [__vector2list_info](#)
- class [__vector2list_stack_info](#)
- struct [__warning_data](#)

Typedefs

- typedef std::vector< __cost_factor * > **__cost_factor_vector**
- typedef std::unordered_map< [std::string](#), [std::string](#) > **__env_t**
- typedef void * **__instruction_address_t**
- typedef const void * **__object_t**
- typedef std::vector< __instruction_address_t > **__stack_npt**
- typedef __stack_npt * **__stack_t**
- typedef std::vector< [__warning_data](#) > **__warning_vector_t**

Enumerations

- enum **__state_type** { **__ON**, **__OFF**, **__INVALID** }

Functions

- std::size_t **__env_to_size_t** (const char * __env_var, std::size_t __default_value)
- template<typename _InputIterator, typename _Function >
_Function **__for_each** (_InputIterator __first, _InputIterator __last, _Function __f)
- __stack_t **__get_stack** ()
- template<typename _Container >
void **__insert_top_n** (_Container & __output, const typename _Container::value_type & __value, typename _Container::size_type __n)
- bool **__is_invalid** ()
- bool **__is_off** ()
- bool **__is_on** ()
- int **__log2** (std::size_t __size)
- int **__log_magnitude** (float __f)
- float **__map_erase_cost** (std::size_t __size)
- float **__map_find_cost** (std::size_t __size)
- float **__map_insert_cost** (std::size_t __size)
- std::size_t **__max_mem** ()
- FILE * **__open_output_file** (const char * __extension)
- bool [__profcxx_init](#) ()
- void **__profcxx_init_unconditional** ()
- void **__read_cost_factors** ()
- template<typename _ForwardIterator, typename _Tp >
_ForwardIterator **__remove** (_ForwardIterator __first, _ForwardIterator __last, const _Tp & __value)
- void [__report](#) (void)
- void **__set_cost_factors** ()
- void **__set_max_mem** ()
- void **__set_max_stack_trace_depth** ()
- void **__set_max_warn_count** ()
- void **__set_trace_path** ()
- std::size_t **__size** (__stack_t __stack)
- std::size_t **__stack_max_depth** ()
- template<typename _Container >
void **__top_n** (const _Container & __input, _Container & __output, typename _Container::size_type __n)
- void **__trace_hash_func_construct** (const void *)

- void `__trace_hash_func_destruct` (const void *, std::size_t, std::size_t, std::size_t)
- void `__trace_hash_func_init` ()
- void `__trace_hash_func_report` (FILE * __f, __warning_vector_t & __warnings)
- void `__trace_hashtable_size_construct` (const void *, std::size_t)
- void `__trace_hashtable_size_destruct` (const void *, std::size_t, std::size_t)
- void `__trace_hashtable_size_init` ()
- void `__trace_hashtable_size_report` (FILE * __f, __warning_vector_t & __warnings)
- void `__trace_hashtable_size_resize` (const void *, std::size_t, std::size_t)
- void `__trace_list_to_set_construct` (const void *)
- void `__trace_list_to_set_destruct` (const void *)
- void `__trace_list_to_set_find` (const void *, std::size_t)
- void `__trace_list_to_set_insert` (const void *, std::size_t, std::size_t)
- void `__trace_list_to_set_invalid_operator` (const void *)
- void `__trace_list_to_set_iterate` (const void *, std::size_t)
- void `__trace_list_to_slist_construct` (const void *)
- void `__trace_list_to_slist_destruct` (const void *)
- void `__trace_list_to_slist_init` ()
- void `__trace_list_to_slist_operation` (const void *)
- void `__trace_list_to_slist_report` (FILE * __f, __warning_vector_t & __warnings)
- void `__trace_list_to_slist_rewind` (const void *)
- void `__trace_list_to_vector_construct` (const void *)
- void `__trace_list_to_vector_destruct` (const void *)
- void `__trace_list_to_vector_init` ()
- void `__trace_list_to_vector_insert` (const void *, std::size_t, std::size_t)
- void `__trace_list_to_vector_invalid_operator` (const void *)
- void `__trace_list_to_vector_iterate` (const void *, std::size_t)
- void `__trace_list_to_vector_report` (FILE * __f, __warning_vector_t & __warnings)
- void `__trace_list_to_vector_resize` (const void *, std::size_t, std::size_t)
- void `__trace_map_to_unordered_map_construct` (const void *)
- void `__trace_map_to_unordered_map_destruct` (const void *)
- void `__trace_map_to_unordered_map_erase` (const void *, std::size_t, std::size_t)
- void `__trace_map_to_unordered_map_find` (const void *, std::size_t)
- void `__trace_map_to_unordered_map_init` ()
- void `__trace_map_to_unordered_map_insert` (const void *, std::size_t, std::size_t)
- void `__trace_map_to_unordered_map_invalidate` (const void *)
- void `__trace_map_to_unordered_map_iterate` (const void *, std::size_t)
- void `__trace_map_to_unordered_map_report` (FILE * __f, __warning_vector_t & __warnings)
- void `__trace_vector_size_construct` (const void *, std::size_t)
- void `__trace_vector_size_destruct` (const void *, std::size_t, std::size_t)
- void `__trace_vector_size_init` ()
- void `__trace_vector_size_report` (FILE *, __warning_vector_t &)
- void `__trace_vector_size_resize` (const void *, std::size_t, std::size_t)
- void `__trace_vector_to_list_construct` (const void *)
- void `__trace_vector_to_list_destruct` (const void *)
- void `__trace_vector_to_list_find` (const void *, std::size_t)
- void `__trace_vector_to_list_init` ()
- void `__trace_vector_to_list_insert` (const void *, std::size_t, std::size_t)
- void `__trace_vector_to_list_invalid_operator` (const void *)
- void `__trace_vector_to_list_iterate` (const void *, std::size_t)
- void `__trace_vector_to_list_report` (FILE *, __warning_vector_t &)
- void `__trace_vector_to_list_resize` (const void *, std::size_t, std::size_t)

- `bool __turn (__state_type __s)`
- `bool __turn_off ()`
- `bool __turn_on ()`
- `void __write (FILE * __f, __stack_t __stack)`
- `void __write_cost_factors ()`
- `_GLIBCXX_PROFILE_DEFINE_DATA (__state_type, __state, __INVALID)`
- `_GLIBCXX_PROFILE_DEFINE_DATA (__trace_hash_func *, __S_hash_func, 0)`
- `_GLIBCXX_PROFILE_DEFINE_DATA (__trace_hashtable_size *, __S_hashtable_size, 0)`
- `_GLIBCXX_PROFILE_DEFINE_DATA (__trace_map2umap *, __S_map2umap, 0)`
- `_GLIBCXX_PROFILE_DEFINE_DATA (__trace_vector_size *, __S_vector_size, 0)`
- `_GLIBCXX_PROFILE_DEFINE_DATA (__trace_vector_to_list *, __S_vector_to_list, 0)`
- `_GLIBCXX_PROFILE_DEFINE_DATA (__trace_list_to_slist *, __S_list_to_slist, 0)`
- `_GLIBCXX_PROFILE_DEFINE_DATA (__trace_list_to_vector *, __S_list_to_vector, 0)`
- `_GLIBCXX_PROFILE_DEFINE_DATA (__cost_factor, __vector_shift_cost_factor, {"__vector_shift_cost_factor", 1.0})`
- `_GLIBCXX_PROFILE_DEFINE_DATA (__cost_factor, __vector_iterate_cost_factor, {"__vector_iterate_cost_factor", 1.0})`
- `_GLIBCXX_PROFILE_DEFINE_DATA (__cost_factor, __vector_resize_cost_factor, {"__vector_resize_cost_factor", 1.0})`
- `_GLIBCXX_PROFILE_DEFINE_DATA (__cost_factor, __list_shift_cost_factor, {"__list_shift_cost_factor", 0.0})`
- `_GLIBCXX_PROFILE_DEFINE_DATA (__cost_factor, __list_iterate_cost_factor, {"__list_iterate_cost_factor", 10.0})`
- `_GLIBCXX_PROFILE_DEFINE_DATA (__cost_factor, __list_resize_cost_factor, {"__list_resize_cost_factor", 0.0})`
- `_GLIBCXX_PROFILE_DEFINE_DATA (__cost_factor, __map_insert_cost_factor, {"__map_insert_cost_factor", 1.5})`
- `_GLIBCXX_PROFILE_DEFINE_DATA (__cost_factor, __map_erase_cost_factor, {"__map_erase_cost_factor", 1.5})`
- `_GLIBCXX_PROFILE_DEFINE_DATA (__cost_factor, __map_find_cost_factor, {"__map_find_cost_factor", 1})`
- `_GLIBCXX_PROFILE_DEFINE_DATA (__cost_factor, __map_iterate_cost_factor, {"__map_iterate_cost_factor", 2.3})`
- `_GLIBCXX_PROFILE_DEFINE_DATA (__cost_factor, __umap_insert_cost_factor, {"__umap_insert_cost_factor", 12.0})`
- `_GLIBCXX_PROFILE_DEFINE_DATA (__cost_factor, __umap_erase_cost_factor, {"__umap_erase_cost_factor", 12.0})`
- `_GLIBCXX_PROFILE_DEFINE_DATA (__cost_factor, __umap_find_cost_factor, {"__umap_find_cost_factor", 10.0})`
- `_GLIBCXX_PROFILE_DEFINE_DATA (__cost_factor, __umap_iterate_cost_factor, {"__umap_iterate_cost_factor", 1.7})`
- `_GLIBCXX_PROFILE_DEFINE_DATA (__cost_factor_vector *, __cost_factors, 0)`
- `_GLIBCXX_PROFILE_DEFINE_DATA (const char *, __S_trace_file_name, _GLIBCXX_PROFILE_TRACE_PATH_ROOT)`
- `_GLIBCXX_PROFILE_DEFINE_DATA (std::size_t, __S_max_warn_count, _GLIBCXX_PROFILE_MAX_WARN_COUNT)`
- `_GLIBCXX_PROFILE_DEFINE_DATA (std::size_t, __S_max_stack_depth, _GLIBCXX_PROFILE_MAX_STACK_DEPTH)`
- `_GLIBCXX_PROFILE_DEFINE_DATA (std::size_t, __S_max_mem, _GLIBCXX_PROFILE_MEM_PER_DIAGNOSTIC)`
- `_GLIBCXX_PROFILE_DEFINE_UNINIT_DATA (__env_t, __env)`
- `_GLIBCXX_PROFILE_DEFINE_UNINIT_DATA (__gnu_cxx::__mutex, __global_lock)`

3.8.1 Detailed Description

GNU profile code for public use.

3.8.2 Typedef Documentation

3.8.2.1 `typedef std:: ::unordered_map<std::string, std::string> __gnu_profile::__env_t`

Internal environment. Values can be set one of two ways: 1. In config file "var = value". The default config file path is libstdc++-profile.conf. 2. By setting process environment variables. For instance, in a Bash shell you can set the unit cost of iterating through a map like this: `export __map_iterate_cost_factor=5.0`. If a value is set both in the input file and through an environment variable, the environment value takes precedence.

Definition at line 65 of file `profiler_trace.h`.

3.8.3 Function Documentation

3.8.3.1 `bool __gnu_profile::__profcxx_init() [inline]`

This function must be called by each instrumentation point.

The common path is inlined fully.

Definition at line 649 of file `profiler_trace.h`.

3.8.3.2 `void __gnu_profile::__report(void) [inline]`

Final report method, registered with **atexit**.

This can also be called directly by user code, including signal handlers. It is protected against deadlocks by the reentrance guard in `profiler.h`. However, when called from a signal handler that triggers while within `__gnu_profile` (under the guarded zone), no output will be produced.

Definition at line 440 of file `profiler_trace.h`.

References `std::begin()`, `std::basic_string<_CharT, _Traits, _Alloc>::begin()`, `std::basic_string<_CharT, _Traits, _Alloc>::c_str()`, `std::end()`, `std::basic_string<_CharT, _Traits, _Alloc>::end()`, `std::basic_string<_CharT, _Traits, _Alloc>::erase()`, `std::basic_string<_CharT, _Traits, _Alloc>::find()`, `std::basic_string<_CharT, _Traits, _Alloc>::find_first_of()`, `std::basic_string<_CharT, _Traits, _Alloc>::find_first_not_of()`, `std::getline()`, `std::basic_string<_CharT, _Traits, _Alloc>::length()`, `std::min()`, and `std::basic_string<_CharT, _Traits, _Alloc>::substr()`.

3.8.3.3 `__gnu_profile::__GLIBCXX_PROFILE_DEFINE_UNINIT_DATA(__gnu_cxx::__mutex, __global_lock)`

Master lock.

3.9 `__gnu_sequential` Namespace Reference

3.9.1 Detailed Description

GNU sequential classes for public use.

3.10 `abi` Namespace Reference

3.10.1 Detailed Description

The cross-vendor C++ Application Binary Interface. A namespace alias to `__cxxabiv1`, but user programs should use the alias `'abi'`.

A brief overview of an ABI is given in the libstdc++ FAQ, question 5.8 (you may have a copy of the FAQ locally, or you can view the online version at http://gcc.gnu.org/onlinedocs/libstdc++/faq.html#5_8).

GCC subscribes to a cross-vendor ABI for C++, sometimes called the IA64 ABI because it happens to be the native ABI for that platform. It is summarized at <http://www.codesourcery.com/cxx-abi/> along with the current specification.

For users of GCC greater than or equal to 3.x, entry points are available in `<cxxabi.h>`, which notes, *'It is not normally necessary for user programs to include this header, or use the entry points directly. However, this header is available should that be needed.'*

3.11 `std` Namespace Reference

Namespaces

- [__debug](#)
- [__detail](#)
- [__parallel](#)
- [__profile](#)
- [regex_constants](#)
- [rel_ops](#)
- [tr1](#)
- [tr2](#)

Classes

- struct [__atomic_base](#)
- struct [__atomic_base< _TPp * >](#)
- struct [__atomic_flag_base](#)
- class [__codecvt_abstract_base](#)
- class [__ctype_abstract_base](#)
- class [__has_iterator_category_helper](#)
- class [_Deque_base](#)
- struct [_Deque_iterator](#)
- struct [_Enable_copy_move](#)
- struct [_Enable_default_constructor](#)
- struct [_Enable_destructor](#)
- struct [_Enable_special_members](#)
- struct [_Fwd_list_base](#)
- struct [_Fwd_list_const_iterator](#)
- struct [_Fwd_list_iterator](#)
- struct [_Fwd_list_node](#)
- struct [_Fwd_list_node_base](#)
- class [_Hashtable](#)
- class [_List_base](#)
- struct [_List_const_iterator](#)
- struct [_List_iterator](#)
- struct [_List_node](#)
- struct [_Sp_ebo_helper< _Nm, _Tp, false >](#)
- struct [_Sp_ebo_helper< _Nm, _Tp, true >](#)
- class [_Temporary_buffer](#)
- struct [_Vector_base](#)
- class [allocator](#)
- class [allocator< void >](#)
- struct [allocator_arg_t](#)
- struct [allocator_traits](#)
- struct [allocator_traits< allocator< _Tp > >](#)
- struct [atomic_flag](#)
- class [auto_ptr](#)
- struct [auto_ptr_ref](#)
- class [back_insert_iterator](#)
- class [bad_weak_ptr](#)
- class [basic_ios](#)
- class [basic_regex](#)
- class [basic_string](#)
- class [bernoulli_distribution](#)
- struct [bidirectional_iterator_tag](#)
- struct [binary_function](#)
- class [binary_negate](#)
- class [binder1st](#)
- class [binder2nd](#)
- class [binomial_distribution](#)
- class [cauchy_distribution](#)
- struct [char_traits](#)
- struct [char_traits< __gnu_cxx::character< _Value, _Int, _St > >](#)

- struct [char_traits< char >](#)
- struct [char_traits< wchar_t >](#)
- class [chi_squared_distribution](#)
- class [codecvt](#)
- class [codecvt< _InternT, _ExternT, encoding_state >](#)
- class [codecvt< char, char, mbstate_t >](#)
- class [codecvt< wchar_t, char, mbstate_t >](#)
- class [codecvt_base](#)
- class [codecvt_byname](#)
- class [collate](#)
- class [collate_byname](#)
- class [const_mem_fun1_ref_t](#)
- class [const_mem_fun1_t](#)
- class [const_mem_fun_ref_t](#)
- class [const_mem_fun_t](#)
- class [ctype](#)
- class [ctype< char >](#)
- class [ctype< wchar_t >](#)
- struct [ctype_base](#)
- class [ctype_byname](#)
- class [ctype_byname< char >](#)
- struct [default_delete](#)
- struct [default_delete< _Tp\[\]>](#)
- class [deque](#)
- class [discard_block_engine](#)
- class [discrete_distribution](#)
- struct [divides](#)
- class [enable_shared_from_this](#)
- struct [equal_to](#)
- class [exponential_distribution](#)
- class [extreme_value_distribution](#)
- class [fisher_f_distribution](#)
- struct [forward_iterator_tag](#)
- class [forward_list](#)
- class [fpos](#)
- class [front_insert_iterator](#)
- class [gamma_distribution](#)
- class [geometric_distribution](#)
- struct [greater](#)
- struct [greater_equal](#)
- class [gslice](#)
- class [gslice_array](#)
- struct [hash](#)
- struct [hash< __gnu_cxx::__u16vstring >](#)
- struct [hash< __gnu_cxx::__u32vstring >](#)
- struct [hash< __gnu_cxx::__vstring >](#)
- struct [hash< __gnu_cxx::__wvstring >](#)
- struct [hash< __gnu_cxx::throw_value_limit >](#)
- struct [hash< __gnu_cxx::throw_value_random >](#)
- struct [hash< __shared_ptr< _Tp, _Lp > >](#)
- struct [hash< _Tp * >](#)

- struct `hash< bool >`
- struct `hash< char >`
- struct `hash< char16_t >`
- struct `hash< char32_t >`
- struct `hash< double >`
- struct `hash< float >`
- struct `hash< int >`
- struct `hash< long >`
- struct `hash< long double >`
- struct `hash< long long >`
- struct `hash< shared_ptr< _Tp > >`
- struct `hash< short >`
- struct `hash< signed char >`
- struct `hash< string >`
- struct `hash< u16string >`
- struct `hash< u32string >`
- struct `hash< unique_ptr< _Tp, _Dp > >`
- struct `hash< unsigned char >`
- struct `hash< unsigned int >`
- struct `hash< unsigned long >`
- struct `hash< unsigned long long >`
- struct `hash< unsigned short >`
- struct `hash< wchar_t >`
- struct `hash< wstring >`
- struct `hash<::vector< bool, _Alloc > >`
- class `independent_bits_engine`
- class `indirect_array`
- struct `input_iterator_tag`
- class `insert_iterator`
- class `ios_base`
- class `istream_iterator`
- class `istreambuf_iterator`
- struct `iterator`
- struct `iterator_traits< _Tp * >`
- struct `iterator_traits< const _Tp * >`
- struct `less`
- struct `less_equal`
- class `linear_congruential_engine`
- class `list`
- class `locale`
- struct `logical_and`
- struct `logical_not`
- struct `logical_or`
- class `lognormal_distribution`
- class `map`
- class `mask_array`
- class `match_results`
- class `mem_fun1_ref_t`
- class `mem_fun1_t`
- class `mem_fun_ref_t`
- class `mem_fun_t`

- class [mersenne_twister_engine](#)
- class [messages](#)
- struct [messages_base](#)
- class [messages_byname](#)
- struct [minus](#)
- struct [modulus](#)
- class [money_base](#)
- class [money_get](#)
- class [money_put](#)
- class [moneypunct](#)
- class [moneypunct_byname](#)
- class [move_iterator](#)
- class [multimap](#)
- struct [multiplies](#)
- class [multiset](#)
- struct [negate](#)
- class [negative_binomial_distribution](#)
- class [nested_exception](#)
- class [normal_distribution](#)
- struct [not_equal_to](#)
- class [num_get](#)
- class [num_put](#)
- class [numpunct](#)
- class [numpunct_byname](#)
- class [ostream_iterator](#)
- class [ostreambuf_iterator](#)
- struct [output_iterator_tag](#)
- struct [owner_less](#)
- struct [owner_less< shared_ptr< _Tp > >](#)
- struct [owner_less< weak_ptr< _Tp > >](#)
- struct [pair](#)
- class [piecewise_constant_distribution](#)
- struct [piecewise_construct_t](#)
- class [piecewise_linear_distribution](#)
- struct [plus](#)
- class [pointer_to_binary_function](#)
- class [pointer_to_unary_function](#)
- struct [pointer_traits](#)
- struct [pointer_traits< _Tp * >](#)
- class [poisson_distribution](#)
- class [priority_queue](#)
- class [queue](#)
- struct [random_access_iterator_tag](#)
- class [random_device](#)
- class [raw_storage_iterator](#)
- class [regex_error](#)
- class [regex_iterator](#)
- class [regex_token_iterator](#)
- struct [regex_traits](#)
- class [reverse_iterator](#)
- class [seed_seq](#)

- class [set](#)
- class [shared_ptr](#)
- class [shuffle_order_engine](#)
- class [slice](#)
- class [slice_array](#)
- class [stack](#)
- class [student_t_distribution](#)
- class [sub_match](#)
- class [time_base](#)
- class [time_get](#)
- class [time_get_byname](#)
- class [time_put](#)
- class [time_put_byname](#)
- struct [unary_function](#)
- class [unary_negate](#)
- class [uniform_int_distribution](#)
- class [uniform_real_distribution](#)
- class [unique_ptr](#)
- class [unique_ptr< _Tp\[\], _Dp >](#)
- class [unordered_map](#)
- class [unordered_multimap](#)
- class [unordered_multiset](#)
- class [unordered_set](#)
- struct [uses_allocator](#)
- class [vector](#)
- class [vector< bool, _Alloc >](#)
- class [weak_ptr](#)
- class [weibull_distribution](#)

Typedefs

- `template<typename _Tp >`
`using __allocator_base = __gnu_cxx::new_allocator< _Tp >`
- `typedef unsigned char __atomic_flag_data_type`
- `typedef FILE __c_file`
- `typedef __locale_t __c_locale`
- `typedef __pthread_mutex_t __c_lock`
- `template<typename _Tp, typename _Hash >`
`using __cache_default = __not< __and< __is_fast_hash< _Hash >, __detail::__is_noexcept_hash< _Tp, _Hash >>>`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`using __sub_match_string = basic_string< typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc >`
- `template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = std::equal_to<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >, typename _Tr = __umap_traits<__cache_default<_Key, _Hash>::value>>`
`using __umap_hashtable = _Hashtable< _Key, std::pair< const _Key, _Tp >, _Alloc, __detail::__Select1st, _Pred, _Hash, __detail::__Mod_range_hashing, __detail::__Default_ranged_hash, __detail::__Prime_rehash_policy, _Tr >`
- `template<bool _Cache>`
`using __umap_traits = __detail::__Hashtable_traits< _Cache, false, true >`

- `template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = std::equal_to<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >, typename _Tr = __ummap_traits<__cache_default<_Key, _Hash>::value>>`
`using __ummap_hashtable = _Hashtable<_Key, std::pair<const _Key, _Tp>, _Alloc, __detail::_Select1st, _Pred, _Hash, __detail::_Mod_range_hashing, __detail::_Default_ranged_hash, __detail::_Prime_rehash_policy, _Tr>`
- `template<bool _Cache>`
`using __ummap_traits = __detail::_Hashtable_traits<_Cache, false, false>`
- `template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = std::equal_to<_Value>, typename _Alloc = std::allocator<_Value>, typename _Tr = __umset_traits<__cache_default<_Value, _Hash>::value>>`
`using __umset_hashtable = _Hashtable<_Value, _Value, _Alloc, __detail::_Identity, _Pred, _Hash, __detail::_Mod_range_hashing, __detail::_Default_ranged_hash, __detail::_Prime_rehash_policy, _Tr>`
- `template<bool _Cache>`
`using __umset_traits = __detail::_Hashtable_traits<_Cache, true, false>`
- `template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = std::equal_to<_Value>, typename _Alloc = std::allocator<_Value>, typename _Tr = __uset_traits<__cache_default<_Value, _Hash>::value>>`
`using __uset_hashtable = _Hashtable<_Value, _Value, _Alloc, __detail::_Identity, _Pred, _Hash, __detail::_Mod_range_hashing, __detail::_Default_ranged_hash, __detail::_Prime_rehash_policy, _Tr>`
- `template<bool _Cache>`
`using __uset_traits = __detail::_Hashtable_traits<_Cache, true, true>`
- `typedef unsigned long _Bit_type`
- `template<typename _InIter>`
`using _RequireInputIter = typename enable_if<is_convertible<typename iterator_traits<_InIter>::iterator, __category, input_iterator_tag>::value>::type`
- `typedef __atomic_base<char> atomic_char`
- `typedef __atomic_base<char16_t> atomic_char16_t`
- `typedef __atomic_base<char32_t> atomic_char32_t`
- `typedef __atomic_base<int> atomic_int`
- `typedef __atomic_base<int_fast16_t> atomic_int_fast16_t`
- `typedef __atomic_base<int_fast32_t> atomic_int_fast32_t`
- `typedef __atomic_base<int_fast64_t> atomic_int_fast64_t`
- `typedef __atomic_base<int_fast8_t> atomic_int_fast8_t`
- `typedef __atomic_base<int_least16_t> atomic_int_least16_t`
- `typedef __atomic_base<int_least32_t> atomic_int_least32_t`
- `typedef __atomic_base<int_least64_t> atomic_int_least64_t`
- `typedef __atomic_base<int_least8_t> atomic_int_least8_t`
- `typedef __atomic_base<intmax_t> atomic_intmax_t`
- `typedef __atomic_base<intptr_t> atomic_intptr_t`
- `typedef __atomic_base<long long> atomic_llong`
- `typedef __atomic_base<long> atomic_long`
- `typedef __atomic_base<ptrdiff_t> atomic_ptrdiff_t`
- `typedef __atomic_base<signed char> atomic_schar`
- `typedef __atomic_base<short> atomic_short`
- `typedef __atomic_base<size_t> atomic_size_t`
- `typedef __atomic_base<unsigned char> atomic_uchar`
- `typedef __atomic_base<unsigned int> atomic_uint`
- `typedef __atomic_base<uint_fast16_t> atomic_uint_fast16_t`
- `typedef __atomic_base<uint_fast32_t> atomic_uint_fast32_t`
- `typedef __atomic_base<uint_fast64_t> atomic_uint_fast64_t`
- `typedef __atomic_base<uint_fast8_t> atomic_uint_fast8_t`
- `typedef __atomic_base<uint_least16_t> atomic_uint_least16_t`
- `typedef __atomic_base<uint_least32_t> atomic_uint_least32_t`
- `typedef __atomic_base<uint_least64_t> atomic_uint_least64_t`

- typedef [__atomic_base](#)< uint_least8_t > [atomic_uint_least8_t](#)
- typedef [__atomic_base](#)< uintmax_t > [atomic_uintmax_t](#)
- typedef [__atomic_base](#)< uintptr_t > [atomic_uintptr_t](#)
- typedef [__atomic_base](#)< unsigned long long > [atomic_ullong](#)
- typedef [__atomic_base](#)< unsigned long > [atomic_ulong](#)
- typedef [__atomic_base](#)< unsigned short > [atomic_ushort](#)
- typedef [__atomic_base](#)< wchar_t > [atomic_wchar_t](#)
- typedef [match_results](#)< const char * > [cmatch](#)
- typedef [regex_iterator](#)< const char * > [cregex_iterator](#)
- typedef [regex_token_iterator](#)< const char * > [cregex_token_iterator](#)
- typedef [sub_match](#)< const char * > [csub_match](#)
- typedef [minstd_rand0](#) [default_random_engine](#)
- typedef [shuffle_order_engine](#)< [minstd_rand0](#), 256 > [knuth_b](#)
- typedef enum [std::memory_order](#) [memory_order](#)
- typedef [linear_congruential_engine](#)< uint_fast32_t, 48271UL, 0UL, 2147483647UL > [minstd_rand](#)
- typedef [linear_congruential_engine](#)< uint_fast32_t, 16807UL, 0UL, 2147483647UL > [minstd_rand0](#)
- typedef [mersenne_twister_engine](#)< uint_fast32_t, 32, 624, 397, 31, 0x9908b0dfUL, 11, 0xffffffffUL, 7, 0x9d2c5680UL, 15, 0xefc60000UL, 18, 1812433253UL > [mt19937](#)
- typedef [mersenne_twister_engine](#)< uint_fast64_t, 64, 312, 156, 31, 0xb5026f5aa96619e9ULL, 29, 0x5555555555555555ULL, 17, 0x71d67ffeda60000ULL, 37, 0xfff7eee000000000ULL, 43, 6364136223846793005ULL > [mt19937_64](#)
- typedef __PTRDIFF_TYPE__ [ptrdiff_t](#)
- typedef [discard_block_engine](#)< [ranlux24_base](#), 223, 23 > [ranlux24](#)
- typedef [subtract_with_carry_engine](#)< uint_fast32_t, 24, 10, 24 > [ranlux24_base](#)
- typedef [discard_block_engine](#)< [ranlux48_base](#), 389, 11 > [ranlux48](#)
- typedef [subtract_with_carry_engine](#)< uint_fast64_t, 48, 5, 12 > [ranlux48_base](#)
- typedef [basic_regex](#)< char > [regex](#)
- typedef __SIZE_TYPE__ [size_t](#)
- typedef [match_results](#)< string::const_iterator > [smatch](#)
- typedef [regex_iterator](#)< string::const_iterator > [sregex_iterator](#)
- typedef [regex_token_iterator](#)< string::const_iterator > [sregex_token_iterator](#)
- typedef [sub_match](#)< string::const_iterator > [ssub_match](#)
- typedef long long [streamoff](#)
- typedef [fpos](#)< mbstate_t > [streampos](#)
- typedef [ptrdiff_t](#) [streamsize](#)
- typedef [basic_string](#)< char > [string](#)
- typedef [fpos](#)< mbstate_t > [u16streampos](#)
- typedef [basic_string](#)< char16_t > [u16string](#)
- typedef [fpos](#)< mbstate_t > [u32streampos](#)
- typedef [basic_string](#)< char32_t > [u32string](#)
- typedef [match_results](#)< const wchar_t * > [wcmatch](#)
- typedef [regex_iterator](#)< const wchar_t * > [wcregex_iterator](#)
- typedef [regex_token_iterator](#)< const wchar_t * > [wcregex_token_iterator](#)
- typedef [sub_match](#)< const wchar_t * > [wcsub_match](#)
- typedef [basic_regex](#)< wchar_t > [wregex](#)
- typedef [match_results](#)< wstring::const_iterator > [wsmatch](#)
- typedef [regex_iterator](#)< wstring::const_iterator > [wsregex_iterator](#)
- typedef [regex_token_iterator](#)< wstring::const_iterator > [wsregex_token_iterator](#)
- typedef [sub_match](#)< wstring::const_iterator > [wssub_match](#)
- typedef [fpos](#)< mbstate_t > [wstreampos](#)
- typedef [basic_string](#)< wchar_t > [wstring](#)

Enumerations

- enum { **_S_threshold** }
- enum { **_S_chunk_size** }
- enum { **_S_word_bit** }
- enum **__memory_order_modifier** { **__memory_order_mask**, **__memory_order_modifier_mask**, **__memory_order_hle_acquire**, **__memory_order_hle_release** }
- enum **_ios_Fmtflags** { **_S_boolalpha**, **_S_dec**, **_S_fixed**, **_S_hex**, **_S_internal**, **_S_left**, **_S_oct**, **_S_right**, **_S_scientific**, **_S_showbase**, **_S_showpoint**, **_S_showpos**, **_S_skipws**, **_S_unitbuf**, **_S_uppercase**, **_S_adjustfield**, **_S_basefield**, **_S_floatfield**, **_S_ios_fmtflags_end** }
- enum **_ios_iostate** { **_S_goodbit**, **_S_badbit**, **_S_eofbit**, **_S_failbit**, **_S_ios_iostate_end** }
- enum **_ios_Openmode** { **_S_app**, **_S_ate**, **_S_bin**, **_S_in**, **_S_out**, **_S_trunc**, **_S_ios_openmode_end** }
- enum **_ios_Seekdir** { **_S_beg**, **_S_cur**, **_S_end**, **_S_ios_seekdir_end** }
- enum **_Rb_tree_color** { **_S_red**, **_S_black** }
- enum **errc** { **address_family_not_supported**, **address_in_use**, **address_not_available**, **already_connected**, **argument_list_too_long**, **argument_out_of_domain**, **bad_address**, **bad_file_descriptor**, **broken_pipe**, **connection_aborted**, **connection_already_in_progress**, **connection_refused**, **connection_reset**, **cross_device_link**, **destination_address_required**, **device_or_resource_busy**, **directory_not_empty**, **executable_format_error**, **file_exists**, **file_too_large**, **filename_too_long**, **function_not_supported**, **host_unreachable**, **illegal_byte_sequence**, **inappropriate_io_control_operation**, **interrupted**, **invalid_argument**, **invalid_seek**, **io_error**, **is_a_directory**, **message_size**, **network_down**, **network_reset**, **network_unreachable**, **no_buffer_space**, **no_child_process**, **no_lock_available**, **no_message**, **no_protocol_option**, **no_space_on_device**, **no_such_device_or_address**, **no_such_device**, **no_such_file_or_directory**, **no_such_process**, **not_a_directory**, **not_a_socket**, **not_connected**, **not_enough_memory**, **operation_in_progress**, **operation_not_permitted**, **operation_not_supported**, **operation_would_block**, **permission_denied**, **protocol_not_supported**, **read_only_file_system**, **resource_deadlock_would_occur**, **resource_unavailable_try_again**, **result_out_of_range**, **timed_out**, **too_many_files_open_in_system**, **too_many_files_open**, **too_many_links**, **too_many_symbolic_link_levels**, **wrong_protocol_type** }
- enum **memory_order** { **memory_order_relaxed**, **memory_order_consume**, **memory_order_acquire**, **memory_order_release**, **memory_order_acq_rel**, **memory_order_seq_cst** }

Functions

- template<typename **_CharT** >
_CharT * __add_grouping (**_CharT * __s**, **_CharT __sep**, const char * **__gbeg**, size_t **__gsize**, const **_CharT * __first**, const **_CharT * __last**)
- template<typename **_Tp** >
_Tp * __addressof (**_Tp &__r**) noexcept
- template<typename **_ForwardIterator**, typename **_BinaryPredicate** >
_ForwardIterator __adjacent_find (**_ForwardIterator __first**, **_ForwardIterator __last**, **_BinaryPredicate __binary_pred**)

- `template<typename _RandomAccessIterator, typename _Distance, typename _Tp, typename _Compare >`
`void __adjust_heap (_RandomAccessIterator __first, _Distance __holeIndex, _Distance __len, _Tp __value, ↵`
`_Compare __comp)`
- `template<typename _InputIterator, typename _Distance >`
`void __advance (_InputIterator &__i, _Distance __n, input_iterator_tag)`
- `template<typename _BidirectionalIterator, typename _Distance >`
`void __advance (_BidirectionalIterator &__i, _Distance __n, bidirectional_iterator_tag)`
- `template<typename _RandomAccessIterator, typename _Distance >`
`void __advance (_RandomAccessIterator &__i, _Distance __n, random_access_iterator_tag)`
- `template<typename _Alloc >`
`void __alloc_on_copy (_Alloc &__one, const _Alloc &__two)`
- `template<typename _Alloc >`
`_Alloc __alloc_on_copy (const _Alloc &__a)`
- `template<typename _Alloc >`
`void __alloc_on_move (_Alloc &__one, _Alloc &__two)`
- `template<typename _Alloc >`
`void __alloc_on_swap (_Alloc &__one, _Alloc &__two)`
- `template<typename _Tp, _Lock_policy _Lp, typename _Alloc, typename... _Args>`
`__shared_ptr< _Tp, _Lp > __allocate_shared (const _Alloc &__a, _Args &&... __args)`
- `__attribute__((always_inline)) void atomic_thread_fence(memory_order __m) noexcept`
- `template<typename _Facet >`
`const _Facet & __check_facet (const _Facet *__f)`
- `template<typename _RandomAccessIterator, typename _Distance, typename _Compare >`
`void __chunk_insertion_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Distance __↵`
`chunk_size, _Compare __comp)`
- `constexpr memory_order __cmpexch_failure_order (memory_order __m) noexcept`
- `constexpr memory_order __cmpexch_failure_order2 (memory_order __m) noexcept`
- `int __convert_from_v (const __c_locale &__cloc __attribute__((__unused__)), char *__out, const int __size ↵`
`__attribute__((__unused__)), const char *__fmt,...)`
- `template<typename _Tp >`
`void __convert_to_v (const char *, _Tp &, ios_base::iostate &, const __c_locale &) throw ()`
- `template<>`
`void __convert_to_v (const char *, float &, ios_base::iostate &, const __c_locale &) throw ()`
- `template<>`
`void __convert_to_v (const char *, double &, ios_base::iostate &, const __c_locale &) throw ()`
- `template<>`
`void __convert_to_v (const char *, long double &, ios_base::iostate &, const __c_locale &) throw ()`
- `template<bool _IsMove, typename _II, typename _OI >`
`_OI __copy_move_a (_II __first, _II __last, _OI __result)`
- `template<bool _IsMove, typename _CharT >`
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT > >::__type __copy↵`
`__move_a2 (_CharT * __first, _CharT * __last, ostreambuf_iterator< _CharT > __result)`
- `template<bool _IsMove, typename _CharT >`
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT > >::__type __copy↵`
`__move_a2 (const _CharT * __first, const _CharT * __last, ostreambuf_iterator< _CharT > __result)`
- `template<bool _IsMove, typename _CharT >`
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, _CharT * >::__type __copy_move_a2 (istreambuf↵`
`__iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, _CharT * __result)`
- `template<bool _IsMove, typename _CharT >`
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT, char_traits< _CharT ↵`
`> >::__type __copy_move_a2 (_CharT *, _CharT *, ostreambuf_iterator< _CharT, char_traits< _CharT ↵`
`>)`

- `template<bool _IsMove, typename _CharT >`
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT, char_traits< _CharT > >::__type __copy_move_a2 (const _CharT *, const _CharT *, ostreambuf_iterator< _CharT, char_traits< _CharT > >)`
- `template<bool _IsMove, typename _CharT >`
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, _CharT * >::__type __copy_move_a2 (istreambuf_iterator< _CharT, char_traits< _CharT > >, istreambuf_iterator< _CharT, char_traits< _CharT > >, _CharT *)`
- `template<bool _IsMove, typename _II, typename _OI >`
`_OI __copy_move_a2 (_II __first, _II __last, _OI __result)`
- `template<bool _IsMove, typename _BI1, typename _BI2 >`
`_BI2 __copy_move_backward_a (_BI1 __first, _BI1 __last, _BI2 __result)`
- `template<bool _IsMove, typename _BI1, typename _BI2 >`
`_BI2 __copy_move_backward_a2 (_BI1 __first, _BI1 __last, _BI2 __result)`
- `template<typename _InputIterator, typename _Size, typename _OutputIterator >`
`_OutputIterator __copy_n (_InputIterator __first, _Size __n, _OutputIterator __result, input_iterator_tag)`
- `template<typename _RandomAccessIterator, typename _Size, typename _OutputIterator >`
`_OutputIterator __copy_n (_RandomAccessIterator __first, _Size __n, _OutputIterator __result, random_access_iterator_tag)`
- `template<typename _InputIterator, typename _Predicate >`
`iterator_traits< _InputIterator >::difference_type __count_if (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `size_t __deque_buf_size (size_t __size)`
- `template<typename _InputIterator >`
`iterator_traits< _InputIterator >::difference_type __distance (_InputIterator __first, _InputIterator __last, input_iterator_tag)`
- `template<typename _RandomAccessIterator >`
`iterator_traits< _RandomAccessIterator >::difference_type __distance (_RandomAccessIterator __first, _RandomAccessIterator __last, random_access_iterator_tag)`
- `template<typename _Alloc >`
`void __do_alloc_on_copy (_Alloc &__one, const _Alloc &__two, true_type)`
- `template<typename _Alloc >`
`void __do_alloc_on_copy (_Alloc &, const _Alloc &, false_type)`
- `template<typename _Alloc >`
`void __do_alloc_on_move (_Alloc &__one, _Alloc &__two, true_type)`
- `template<typename _Alloc >`
`void __do_alloc_on_move (_Alloc &, _Alloc &, false_type)`
- `template<typename _Alloc >`
`void __do_alloc_on_swap (_Alloc &__one, _Alloc &__two, true_type)`
- `template<typename _Alloc >`
`void __do_alloc_on_swap (_Alloc &, _Alloc &, false_type)`
- `template<typename _Tp1, typename _Tp2 >`
`void __enable_shared_from_this_helper (const __shared_count<> &__pn, const enable_shared_from_this< _Tp1 > *__pe, const _Tp2 *__px) noexcept`
- `template<_Lock_policy _Lp, typename _Tp1, typename _Tp2 >`
`void __enable_shared_from_this_helper (const __shared_count< _Lp > &, const __enable_shared_from_this< _Tp1, _Lp > *__pe, const _Tp2 *__px) noexcept`
- `template<_Lock_policy _Lp>`
`void __enable_shared_from_this_helper (const __shared_count< _Lp > &,...) noexcept`
- `template<_Lock_policy _Lp1, typename _Tp1, typename _Tp2 >`
`void __enable_shared_from_this_helper (const __shared_count< _Lp1 > &__pn, const __enable_shared_from_this< _Tp1, _Lp1 > *__pe, const _Tp2 *__px) noexcept`

- `template<typename _I1, typename _I2 >`
`bool __equal_aux (_I1 __first1, _I1 __last1, _I2 __first2)`
- `template<typename _ForwardIterator, typename _Tp, typename _CompareItTp, typename _CompareTpIt >`
`pair< _ForwardIterator, _ForwardIterator > __equal_range (_ForwardIterator __first, _ForwardIterator __last, const _Tp & __val, _CompareItTp __comp_it_val, _CompareTpIt __comp_val_it)`
- `template<typename _ForwardIterator, typename _Tp >`
`__gnu_cxx::__enable_if<! __is_scalar< _Tp >::value, void >::type __fill_a (_ForwardIterator __first, ↵ ForwardIterator __last, const _Tp & __value)`
- `template<typename _ForwardIterator, typename _Tp >`
`__gnu_cxx::__enable_if< __is_scalar< _Tp >::value, void >::type __fill_a (_ForwardIterator __first, ↵ ForwardIterator __last, const _Tp & __value)`
- `template<typename _Tp >`
`__gnu_cxx::__enable_if< __is_byte< _Tp >::value, void >::type __fill_a (_Tp * __first, _Tp * __last, const _Tp & __c)`
- `void __fill_bvector (_Bit_iterator __first, _Bit_iterator __last, bool __x)`
- `template<typename _OutputIterator, typename _Size, typename _Tp >`
`__gnu_cxx::__enable_if<! __is_scalar< _Tp >::value, _OutputIterator >::type __fill_n_a (_OutputIterator __first, _Size __n, const _Tp & __value)`
- `template<typename _OutputIterator, typename _Size, typename _Tp >`
`__gnu_cxx::__enable_if< __is_scalar< _Tp >::value, _OutputIterator >::type __fill_n_a (_OutputIterator __first, _Size __n, const _Tp & __value)`
- `template<typename _Size, typename _Tp >`
`__gnu_cxx::__enable_if< __is_byte< _Tp >::value, _Tp * >::type __fill_n_a (_Tp * __first, _Size __n, const _Tp & __c)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void __final_insertion_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate >`
`_ForwardIterator1 __find_end (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, forward_iterator_tag, forward_iterator_tag, _BinaryPredicate __comp)`
- `template<typename _BidirectionalIterator1, typename _BidirectionalIterator2, typename _BinaryPredicate >`
`_BidirectionalIterator1 __find_end (_BidirectionalIterator1 __first1, _BidirectionalIterator1 __last1, ↵ BidirectionalIterator2 __first2, BidirectionalIterator2 __last2, bidirectional_iterator_tag, bidirectional_iterator_tag, _BinaryPredicate __comp)`
- `template<typename _InputIterator, typename _Predicate >`
`_InputIterator __find_if (_InputIterator __first, _InputIterator __last, _Predicate __pred, input_iterator_tag)`
- `template<typename _RandomAccessIterator, typename _Predicate >`
`_RandomAccessIterator __find_if (_RandomAccessIterator __first, _RandomAccessIterator __last, _Predicate ↵ __pred, random_access_iterator_tag)`
- `template<typename _Iterator, typename _Predicate >`
`_Iterator __find_if (_Iterator __first, _Iterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _Predicate >`
`_InputIterator __find_if_not (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _Predicate, typename _Distance >`
`_InputIterator __find_if_not_n (_InputIterator __first, _Distance & __len, _Predicate __pred)`
- `template<typename _EuclideanRingElement >`
`_EuclideanRingElement __gcd (_EuclideanRingElement __m, _EuclideanRingElement __n)`
- `template<typename _Ex >`
`const nested_exception * __get_nested_exception (const _Ex & __ex)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void __heap_select (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last, _Compare __comp)`

- `template<typename _Tp >`
`size_t __iconv_adapter (size_t(*__func)(iconv_t, _Tp, size_t *, char **, size_t *), iconv_t __cd, char **__inbuf,`
`size_t *__inbytes, char **__outbuf, size_t *__outbytes)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Compare >`
`bool __includes (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2,`
`_Compare __comp)`
- `template<typename _BidirectionalIterator, typename _Compare >`
`void __inplace_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last,`
`_Compare __comp)`
- `template<typename _ForwardIterator, typename _Predicate, typename _Distance >`
`_ForwardIterator __inplace_stable_partition (_ForwardIterator __first, _Predicate __pred, _Distance __len)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void __inplace_stable_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void __insertion_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Size, typename _Compare >`
`void __introsselect (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __↵`
`__last, _Size __depth_limit, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Size, typename _Compare >`
`void __introsort_loop (_RandomAccessIterator __first, _RandomAccessIterator __last, _Size __depth_limit, __↵`
`_Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Distance >`
`bool __is_heap (_RandomAccessIterator __first, _Distance __n)`
- `template<typename _RandomAccessIterator, typename _Compare, typename _Distance >`
`bool __is_heap (_RandomAccessIterator __first, _Compare __comp, _Distance __n)`
- `template<typename _RandomAccessIterator >`
`bool __is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`bool __is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Distance, typename _Compare >`
`_Distance __is_heap_until (_RandomAccessIterator __first, _Distance __n, _Compare __comp)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate >`
`bool __is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, __↵`
`_BinaryPredicate __pred)`
- `template<typename _ForwardIterator, typename _Compare >`
`_ForwardIterator __is_sorted_until (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _Iter >`
`iterator_traits< _Iter >::iterator_category __iterator_category (const _Iter &)`
- `template<typename _II1, typename _II2 >`
`bool __lexicographical_compare_aux (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2)`
- `template<typename _II1, typename _II2, typename _Compare >`
`bool __lexicographical_compare_impl (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2, _Compare __↵`
`comp)`
- `constexpr int __lg (int __n)`
- `constexpr unsigned __lg (unsigned __n)`
- `constexpr long __lg (long __n)`
- `constexpr unsigned long __lg (unsigned long __n)`
- `constexpr long long __lg (long long __n)`
- `constexpr unsigned long long __lg (unsigned long long __n)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`
`_ForwardIterator __lower_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare`
`__comp)`

- `template<typename _RandomAccessIterator, typename _Compare >`
`void __make_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _Iterator, typename _ReturnType = typename conditional<__move_if_noexcept_cond <typename iterator_traits<_Iterator>::value_type>::value, _Iterator, move_iterator<_Iterator>>::type>`
`_ReturnType __make_move_if_noexcept_iterator (_Iterator __i)`
- `template<typename _Tp, _Lock_policy _Lp, typename... _Args>`
`__shared_ptr<_Tp, _Lp> __make_shared (_Args &&...__args)`
- `template<typename _ForwardIterator, typename _Compare >`
`_ForwardIterator __max_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`
`_OutputIterator __merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _BidirectionalIterator, typename _Distance, typename _Pointer, typename _Compare >`
`void __merge_adaptive (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Distance __len1, _Distance __len2, _Pointer __buffer, _Distance __buffer_size, _Compare __comp)`
- `template<typename _RandomAccessIterator1, typename _RandomAccessIterator2, typename _Distance, typename _Compare >`
`void __merge_sort_loop (_RandomAccessIterator1 __first, _RandomAccessIterator1 __last, _RandomAccessIterator2 __result, _Distance __step_size, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Pointer, typename _Compare >`
`void __merge_sort_with_buffer (_RandomAccessIterator __first, _RandomAccessIterator __last, _Pointer __buffer, _Compare __comp)`
- `template<typename _BidirectionalIterator, typename _Distance, typename _Compare >`
`void __merge_without_buffer (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Distance __len1, _Distance __len2, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Compare >`
`_ForwardIterator __min_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Compare >`
`pair<_ForwardIterator, _ForwardIterator> __minmax_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate >`
`pair<_InputIterator1, _InputIterator2> __mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _BinaryPredicate __binary_pred)`
- `template<typename _Iterator >`
`_Miter_base<_Iterator>::iterator_type __miter_base (_Iterator __it)`
- `template<typename _Iterator, typename _Compare >`
`void __move_median_to_first (_Iterator __result, _Iterator __a, _Iterator __b, _Iterator __c, _Compare __comp)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Compare >`
`_OutputIterator __move_merge (_InputIterator __first1, _InputIterator __last1, _InputIterator __first2, _InputIterator __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`
`void __move_merge_adaptive (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _BidirectionalIterator1, typename _BidirectionalIterator2, typename _BidirectionalIterator3, typename _Compare >`
`void __move_merge_adaptive_backward (_BidirectionalIterator1 __first1, _BidirectionalIterator1 __last1, _BidirectionalIterator2 __first2, _BidirectionalIterator2 __last2, _BidirectionalIterator3 __result, _Compare __comp)`
- `template<typename _BidirectionalIterator, typename _Compare >`
`bool __next_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _Iterator >`
`_Niter_base<_Iterator>::iterator_type __niter_base (_Iterator __it)`
- `template<typename _CharT, typename _Traits >`
`void __ostream_fill (basic_ostream<_CharT, _Traits> &__out, streamsize __n)`

- `template<typename _CharT, typename _Traits >`
`basic_ostream< _CharT, _Traits > & __ostream_insert (basic_ostream< _CharT, _Traits > &__out, const ↵`
`CharT *__s, streamsize __n)`
- `template<typename _CharT, typename _Traits >`
`void __ostream_write (basic_ostream< _CharT, _Traits > &__out, const _CharT *__s, streamsize __n)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void __partial_sort (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator ↵`
`__last, _Compare __comp)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _Compare >`
`_RandomAccessIterator __partial_sort_copy (_InputIterator __first, _InputIterator __last, _RandomAccess↵`
`Iterator __result_first, _RandomAccessIterator __result_last, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Predicate >`
`_ForwardIterator __partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, forward_↵`
`iterator_tag)`
- `template<typename _BidirectionalIterator, typename _Predicate >`
`_BidirectionalIterator __partition (_BidirectionalIterator __first, _BidirectionalIterator __last, _Predicate __pred,`
`bidirectional_iterator_tag)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void __pop_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _RandomAccessIterator ↵`
`__result, _Compare __comp)`
- `template<typename _BidirectionalIterator, typename _Compare >`
`bool __prev_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Distance, typename _Tp, typename _Compare >`
`void __push_heap (_RandomAccessIterator __first, _Distance __holeIndex, _Distance __topIndex, _Tp __value,`
`_Compare __comp)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate >`
`_OutputIterator __remove_copy_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, ↵`
`Predicate __pred)`
- `template<typename _ForwardIterator, typename _Predicate >`
`_ForwardIterator __remove_if (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate, typename _Tp >`
`_OutputIterator __replace_copy_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, ↵`
`Predicate __pred, const _Tp &__new_value)`
- `template<typename _BidirectionalIterator >`
`void __reverse (_BidirectionalIterator __first, _BidirectionalIterator __last, bidirectional_iterator_tag)`
- `template<typename _RandomAccessIterator >`
`void __reverse (_RandomAccessIterator __first, _RandomAccessIterator __last, random_access_iterator_tag)`
- `template<typename _ForwardIterator >`
`void __rotate (_ForwardIterator __first, _ForwardIterator __middle, _ForwardIterator __last, forward_iterator_tag)`
- `template<typename _BidirectionalIterator >`
`void __rotate (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last,`
`bidirectional_iterator_tag)`
- `template<typename _RandomAccessIterator >`
`void __rotate (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator ↵`
`__last, random_access_iterator_tag)`
- `template<typename _BidirectionalIterator1, typename _BidirectionalIterator2, typename _Distance >`
`_BidirectionalIterator1 __rotate_adaptive (_BidirectionalIterator1 __first, _BidirectionalIterator1 __middle, ↵`
`_BidirectionalIterator1 __last, _Distance __len1, _Distance __len2, _BidirectionalIterator2 __buffer, _Distance ↵`
`__buffer_size)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate >`
`_ForwardIterator1 __search (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2,`
`_ForwardIterator2 __last2, _BinaryPredicate __predicate)`

- `template<typename _ForwardIterator, typename _Integer, typename _UnaryPredicate >`
`_ForwardIterator search_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, _UnaryPredicate __unary_pred)`
- `template<typename _ForwardIterator, typename _Integer, typename _UnaryPredicate >`
`_ForwardIterator search_n_aux (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, _UnaryPredicate __unary_pred, std::forward_iterator_tag)`
- `template<typename _RandomAccessIter, typename _Integer, typename _UnaryPredicate >`
`_RandomAccessIter search_n_aux (_RandomAccessIter __first, _RandomAccessIter __last, _Integer __count, _UnaryPredicate __unary_pred, std::random_access_iterator_tag)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`
`_OutputIterator set_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`
`_OutputIterator set_intersection (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`
`_OutputIterator set_symmetric_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`
`_OutputIterator set_union (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void sort_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Predicate >`
`_ForwardIterator stable_partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Pointer, typename _Predicate, typename _Distance >`
`_ForwardIterator stable_partition_adaptive (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, _Distance __len, _Pointer __buffer, _Distance __buffer_size)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void stable_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Pointer, typename _Distance, typename _Compare >`
`void stable_sort_adaptive (_RandomAccessIterator __first, _RandomAccessIterator __last, _Pointer __buffer, _Distance __buffer_size, _Compare __comp)`
- `void throw_bad_alloc (void) __attribute__((__noreturn__))`
- `void throw_bad_cast (void) __attribute__((__noreturn__))`
- `void throw_bad_exception (void) __attribute__((__noreturn__))`
- `void throw_bad_function_call () __attribute__((__noreturn__))`
- `void throw_bad_typeid (void) __attribute__((__noreturn__))`
- `void throw_bad_weak_ptr ()`
- `void throw_domain_error (const char *) __attribute__((__noreturn__))`
- `void throw_future_error (int) __attribute__((__noreturn__))`
- `void throw_invalid_argument (const char *) __attribute__((__noreturn__))`
- `void throw_ios_failure (const char *) __attribute__((__noreturn__))`
- `void throw_length_error (const char *) __attribute__((__noreturn__))`
- `void throw_logic_error (const char *) __attribute__((__noreturn__))`
- `void throw_out_of_range (const char *) __attribute__((__noreturn__))`
- `void throw_out_of_range_fmt (const char *,...) __attribute__((__noreturn__)) __attribute__((__format__(_Printf, _Printf)))`
- `void throw_overflow_error (const char *) __attribute__((__noreturn__))`
- `void throw_range_error (const char *) __attribute__((__noreturn__))`

- void **__throw_regex_error** (regex_constants::error_type __ecode)
- void void **__throw_runtime_error** (const char *) __attribute__((__noreturn__))
- void **__throw_system_error** (int) __attribute__((__noreturn__))
- void **__throw_underflow_error** (const char *) __attribute__((__noreturn__))
- template<typename _Ex >
void **__throw_with_nested** (_Ex &&, const nested_exception *=0) __attribute__((__noreturn__))
- template<typename _Ex >
void **__throw_with_nested** (_Ex &&,...) __attribute__((__noreturn__))
- template<typename _RandomAccessIterator, typename _Compare >
void **__unguarded_insertion_sort** (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare <= __comp)
- template<typename _RandomAccessIterator, typename _Compare >
void **__unguarded_linear_insert** (_RandomAccessIterator __last, _Compare __comp)
- template<typename _RandomAccessIterator, typename _Compare >
_RandomAccessIterator **__unguarded_partition** (_RandomAccessIterator __first, _RandomAccessIterator __last, _RandomAccessIterator __pivot, _Compare __comp)
- template<typename _RandomAccessIterator, typename _Compare >
_RandomAccessIterator **__unguarded_partition_pivot** (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)
- template<typename _Pointer, typename _ForwardIterator >
void **__uninitialized_construct_buf** (_Pointer __first, _Pointer __last, _ForwardIterator __seed)
- template<typename _InputIterator, typename _ForwardIterator, typename _Allocator >
_ForwardIterator **__uninitialized_copy_a** (_InputIterator __first, _InputIterator __last, _ForwardIterator __result, _Allocator & __alloc)
- template<typename _InputIterator, typename _ForwardIterator, typename _Tp >
_ForwardIterator **__uninitialized_copy_a** (_InputIterator __first, _InputIterator __last, _ForwardIterator __result, allocator< _Tp > &)
- template<typename _InputIterator1, typename _InputIterator2, typename _ForwardIterator, typename _Allocator >
_ForwardIterator **__uninitialized_copy_move** (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _ForwardIterator __result, _Allocator & __alloc)
- template<typename _InputIterator, typename _Size, typename _ForwardIterator >
_ForwardIterator **__uninitialized_copy_n** (_InputIterator __first, _Size __n, _ForwardIterator __result, input_iterator_tag)
- template<typename _RandomAccessIterator, typename _Size, typename _ForwardIterator >
_ForwardIterator **__uninitialized_copy_n** (_RandomAccessIterator __first, _Size __n, _ForwardIterator __result, random_access_iterator_tag)
- template<typename _ForwardIterator >
void **__uninitialized_default** (_ForwardIterator __first, _ForwardIterator __last)
- template<typename _ForwardIterator, typename _Allocator >
void **__uninitialized_default_a** (_ForwardIterator __first, _ForwardIterator __last, _Allocator & __alloc)
- template<typename _ForwardIterator, typename _Tp >
void **__uninitialized_default_a** (_ForwardIterator __first, _ForwardIterator __last, allocator< _Tp > &)
- template<typename _ForwardIterator, typename _Size >
void **__uninitialized_default_n** (_ForwardIterator __first, _Size __n)
- template<typename _ForwardIterator, typename _Size, typename _Allocator >
void **__uninitialized_default_n_a** (_ForwardIterator __first, _Size __n, _Allocator & __alloc)
- template<typename _ForwardIterator, typename _Size, typename _Tp >
void **__uninitialized_default_n_a** (_ForwardIterator __first, _Size __n, allocator< _Tp > &)
- template<typename _ForwardIterator, typename _Tp, typename _Allocator >
void **__uninitialized_fill_a** (_ForwardIterator __first, _ForwardIterator __last, const _Tp & __x, _Allocator & __alloc)

- `template<typename _ForwardIterator, typename _Tp, typename _Tp2 >`
`void __uninitialized_fill_a (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__x, allocator< _Tp2 > &)`
- `template<typename _ForwardIterator, typename _Tp, typename _InputIterator, typename _Allocator >`
`_ForwardIterator __uninitialized_fill_move (_ForwardIterator __result, _ForwardIterator __mid, const _Tp &__x, _InputIterator __first, _InputIterator __last, _Allocator &__alloc)`
- `template<typename _ForwardIterator, typename _Size, typename _Tp, typename _Allocator >`
`void __uninitialized_fill_n_a (_ForwardIterator __first, _Size __n, const _Tp &__x, _Allocator &__alloc)`
- `template<typename _ForwardIterator, typename _Size, typename _Tp, typename _Tp2 >`
`void __uninitialized_fill_n_a (_ForwardIterator __first, _Size __n, const _Tp &__x, allocator< _Tp2 > &)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _Allocator >`
`_ForwardIterator __uninitialized_move_a (_InputIterator __first, _InputIterator __last, _ForwardIterator __result, _Allocator &__alloc)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _ForwardIterator, typename _Allocator >`
`_ForwardIterator __uninitialized_move_copy (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _ForwardIterator __result, _Allocator &__alloc)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _Tp, typename _Allocator >`
`void __uninitialized_move_fill (_InputIterator __first1, _InputIterator __last1, _ForwardIterator __first2, _ForwardIterator __last2, const _Tp &__x, _Allocator &__alloc)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _Allocator >`
`_ForwardIterator __uninitialized_move_if_noexcept_a (_InputIterator __first, _InputIterator __last, _ForwardIterator __result, _Allocator &__alloc)`
- `template<typename _ForwardIterator, typename _BinaryPredicate >`
`_ForwardIterator __unique (_ForwardIterator __first, _ForwardIterator __last, _BinaryPredicate __binary_pred)`
- `template<typename _ForwardIterator, typename _OutputIterator, typename _BinaryPredicate >`
`_OutputIterator __unique_copy (_ForwardIterator __first, _ForwardIterator __last, _OutputIterator __result, _BinaryPredicate __binary_pred, forward_iterator_tag, output_iterator_tag)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryPredicate >`
`_OutputIterator __unique_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryPredicate __binary_pred, input_iterator_tag, output_iterator_tag)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _BinaryPredicate >`
`_ForwardIterator __unique_copy (_InputIterator __first, _InputIterator __last, _ForwardIterator __result, _BinaryPredicate __binary_pred, input_iterator_tag, forward_iterator_tag)`
- `template<typename _Bi_iter >`
`const sub_match< _Bi_iter > & __unmatched_sub ()`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`
`_ForwardIterator __upper_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)`
- `template<typename _Tp, typename _Alloc, typename... _Args>`
`__uses_alloc_impl< _Tp, _Alloc, _Args... > __use_alloc (const _Alloc &__a)`
- `template<typename _Tp >`
`void __valarray_copy (const _Tp *__restrict __a, size_t __n, _Tp *__restrict __b)`
- `template<typename _Tp >`
`void __valarray_copy (const _Tp *__restrict __a, size_t __n, size_t __s, _Tp *__restrict __b)`
- `template<typename _Tp >`
`void __valarray_copy (const _Tp *__restrict __a, _Tp *__restrict __b, size_t __n, size_t __s)`
- `template<typename _Tp >`
`void __valarray_copy (const _Tp *__restrict __src, size_t __n, size_t __s1, _Tp *__restrict __dst, size_t __s2)`
- `template<typename _Tp >`
`void __valarray_copy (const _Tp *__restrict __a, const size_t *__restrict __i, _Tp *__restrict __b, size_t __n)`

- `template<typename _Tp >`
`void __valarray_copy (const _Tp *__restrict __a, size_t __n, _Tp *__restrict __b, const size_t *__restrict __i)`
- `template<typename _Tp >`
`void __valarray_copy (const _Tp *__restrict __src, size_t __n, const size_t *__restrict __i, _Tp *__restrict __dst, const size_t *__restrict __j)`
- `template<typename _Tp >`
`void __valarray_copy (_Array<_Tp> __a, size_t __n, _Array<_Tp> __b)`
- `template<typename _Tp >`
`void __valarray_copy (_Array<_Tp> __a, size_t __n, size_t __s, _Array<_Tp> __b)`
- `template<typename _Tp >`
`void __valarray_copy (_Array<_Tp> __a, _Array<_Tp> __b, size_t __n, size_t __s)`
- `template<typename _Tp >`
`void __valarray_copy (_Array<_Tp> __a, size_t __n, size_t __s1, _Array<_Tp> __b, size_t __s2)`
- `template<typename _Tp >`
`void __valarray_copy (_Array<_Tp> __a, _Array<size_t> __i, _Array<_Tp> __b, size_t __n)`
- `template<typename _Tp >`
`void __valarray_copy (_Array<_Tp> __a, size_t __n, _Array<_Tp> __b, _Array<size_t> __i)`
- `template<typename _Tp >`
`void __valarray_copy (_Array<_Tp> __src, size_t __n, _Array<size_t> __i, _Array<_Tp> __dst, _Array<size_t> __j)`
- `template<typename _Tp >`
`void __valarray_copy_construct (const _Tp *__b, const _Tp *__e, _Tp *__restrict __o)`
- `template<typename _Tp >`
`void __valarray_copy_construct (const _Tp *__restrict __a, size_t __n, size_t __s, _Tp *__restrict __o)`
- `template<typename _Tp >`
`void __valarray_copy_construct (const _Tp *__restrict __a, const size_t *__restrict __i, _Tp *__restrict __o, size_t __n)`
- `template<typename _Tp >`
`void __valarray_copy_construct (_Array<_Tp> __a, _Array<size_t> __i, _Array<_Tp> __b, size_t __n)`
- `template<typename _Tp >`
`void __valarray_copy_construct (_Array<_Tp> __a, size_t __n, size_t __s, _Array<_Tp> __b)`
- `template<typename _Tp >`
`void __valarray_default_construct (_Tp *__b, _Tp *__e)`
- `template<typename _Tp >`
`void __valarray_destroy_elements (_Tp *__b, _Tp *__e)`
- `template<typename _Tp >`
`void __valarray_fill (_Tp *__restrict __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`
`void __valarray_fill (_Tp *__restrict __a, size_t __n, size_t __s, const _Tp &__t)`
- `template<typename _Tp >`
`void __valarray_fill (_Tp *__restrict __a, const size_t *__restrict __i, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`
`void __valarray_fill (_Array<_Tp> __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`
`void __valarray_fill (_Array<_Tp> __a, size_t __n, size_t __s, const _Tp &__t)`
- `template<typename _Tp >`
`void __valarray_fill (_Array<_Tp> __a, _Array<size_t> __i, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`
`void __valarray_fill_construct (_Tp *__b, _Tp *__e, const _Tp __t)`
- `void *__valarray_get_memory (size_t __n)`
- `template<typename _Tp >`
`_Tp *__restrict __valarray_get_storage (size_t __n)`

- `template<typename _Ta >`
`_Ta::value_type __valarray_max (const _Ta &__a)`
- `template<typename _Ta >`
`_Ta::value_type __valarray_min (const _Ta &__a)`
- `template<typename _Tp >`
`_Tp __valarray_product (const _Tp *__f, const _Tp *__l)`
- `void __valarray_release_memory (void *__p)`
- `template<typename _Tp >`
`_Tp __valarray_sum (const _Tp *__f, const _Tp *__l)`
- `template<typename _CharT >`
`ostreambuf_iterator< _CharT > __write (ostreambuf_iterator< _CharT > __s, const _CharT *__ws, int __len)`
- `template<typename _CharT, typename _Outlter >`
`_Outlter __write (_Outlter __s, const _CharT *__ws, int __len)`
- `template<typename _Tp >`
`void __Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`
`void __Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`
`void __Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`
`void __Array_augmented__bitwise_and (_Array< _Tp > __a, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void __Array_augmented__bitwise_and (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`
`void __Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __s, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void __Array_augmented__bitwise_and (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`
`void __Array_augmented__bitwise_and (_Array< _Tp > __a, _Array< size_t > __i, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void __Array_augmented__bitwise_and (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`
`void __Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp, class _Dom >`
`void __Array_augmented__bitwise_and (_Array< _Tp > __a, _Array< bool > __m, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void __Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp >`
`void __Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`
`void __Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`
`void __Array_augmented__bitwise_or (_Array< _Tp > __a, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void __Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`

- `template<typename _Tp >`
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`

- `template<typename _Tp >`
`void _Array_augmented__divides (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`
`void _Array_augmented__divides (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__divides (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__divides (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`
`void _Array_augmented__divides (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__divides (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__divides (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__divides (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__divides (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__divides (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__divides (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__divides (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp >`
`void _Array_augmented__minus (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`
`void _Array_augmented__minus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__minus (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__minus (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`
`void _Array_augmented__minus (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__minus (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__minus (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__minus (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__minus (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__minus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`

- `template<typename _Tp, class _Dom >`
`void _Array_augmented__minus (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp >`
`&__e, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__minus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__modulus (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp >`
`&__e, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__modulus (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`
`void _Array_augmented__modulus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__modulus (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__modulus (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e,`
`size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__modulus (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t`
`__n)`
- `template<typename _Tp >`
`void _Array_augmented__modulus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t >`
`__i)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__modulus (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp >`
`&__e, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__modulus (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t`
`__n)`
- `template<typename _Tp >`
`void _Array_augmented__modulus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool >`
`__m)`
- `template<typename _Tp >`
`void _Array_augmented__modulus (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`
`void _Array_augmented__modulus (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp >`
`void _Array_augmented__multiplies (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t >`
`__i)`
- `template<typename _Tp >`
`void _Array_augmented__multiplies (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`
`void _Array_augmented__multiplies (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__multiplies (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__multiplies (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__multiplies (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e,`
`size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__multiplies (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t`
`__n)`

- `template<typename _Tp, class _Dom >`
`void _Array_augmented__multiplies (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp >`
`&__e, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__multiplies (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t`
`__n)`
- `template<typename _Tp >`
`void _Array_augmented__multiplies (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool >`
`__m)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__multiplies (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp >`
`&__e, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__multiplies (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`
`void _Array_augmented__plus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__plus (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e,`
`size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__plus (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`
`void _Array_augmented__plus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__plus (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__plus (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__plus (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t`
`__n)`
- `template<typename _Tp >`
`void _Array_augmented__plus (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__plus (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e,`
`size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__plus (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__plus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`
`void _Array_augmented__plus (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`
`void _Array_augmented__shift_left (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`
`void _Array_augmented__shift_left (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__shift_left (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__shift_left (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__shift_left (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e,`
`size_t __n)`

- `template<typename _Tp >`
`void _Array_augmented__shift_left (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__shift_left (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__shift_left (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__shift_left (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__shift_left (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`
`void _Array_augmented__shift_left (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__shift_left (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp >`
`void _Array_augmented__shift_right (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp >`
`void _Array_augmented__shift_right (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__shift_right (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__shift_right (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__shift_right (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__shift_right (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__shift_right (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__shift_right (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`
`void _Array_augmented__shift_right (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`
`void _Array_augmented__shift_right (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp >`
`void _Array_augmented__shift_right (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`
`void _Array_augmented__shift_right (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _T1, typename... _Args>`
`void _Construct (_T1 *__p, _Args &&...__args)`

- `template<typename _Tp >`
`void _Destroy (_Tp *__pointer)`
- `template<typename _ForwardIterator >`
`void _Destroy (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Allocator >`
`void _Destroy (_ForwardIterator __first, _ForwardIterator __last, _Allocator &__alloc)`
- `template<typename _ForwardIterator, typename _Tp >`
`void _Destroy (_ForwardIterator __first, _ForwardIterator __last, allocator<_Tp> &)`
- `size_t _Fnv_hash_bytes (const void *__ptr, size_t __len, size_t __seed)`
- `size_t _Hash_bytes (const void *__ptr, size_t __len, size_t __seed)`
- `unsigned int _Rb_tree_black_count (const _Rb_tree_node_base *__node, const _Rb_tree_node_base *__root) throw ()`
- `_Rb_tree_node_base * _Rb_tree_decrement (_Rb_tree_node_base *__x) throw ()`
- `const _Rb_tree_node_base * _Rb_tree_decrement (const _Rb_tree_node_base *__x) throw ()`
- `_Rb_tree_node_base * _Rb_tree_increment (_Rb_tree_node_base *__x) throw ()`
- `const _Rb_tree_node_base * _Rb_tree_increment (const _Rb_tree_node_base *__x) throw ()`
- `void _Rb_tree_insert_and_rebalance (const bool __insert_left, _Rb_tree_node_base *__x, _Rb_tree_node_base *__p, _Rb_tree_node_base &__header) throw ()`
- `_Rb_tree_node_base * _Rb_tree_rebalance_for_erase (_Rb_tree_node_base *const __z, _Rb_tree_node_base &__header) throw ()`
- `template<class _Dom >`
`_Expr< _UnClos< _Abs, _Expr, _Dom >, typename _Dom::value_type > abs (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`
`_Expr< _UnClos< _Abs, _ValArray, _Tp >, _Tp > abs (const valarray<_Tp> &__v)`
- `template<typename _InputIterator, typename _Tp >`
`_Tp accumulate (_InputIterator __first, _InputIterator __last, _Tp __init)`
- `template<typename _InputIterator, typename _Tp, typename _BinaryOperation >`
`_Tp accumulate (_InputIterator __first, _InputIterator __last, _Tp __init, _BinaryOperation __binary_op)`
- `template<class _Dom >`
`_Expr< _UnClos< _Acos, _Expr, _Dom >, typename _Dom::value_type > acos (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`
`_Expr< _UnClos< _Acos, _ValArray, _Tp >, _Tp > acos (const valarray<_Tp> &__v)`
- `template<typename _Tp >`
`_Tp * addressof (_Tp &__r) noexcept`
- `template<typename _InputIterator, typename _OutputIterator >`
`_OutputIterator adjacent_difference (_InputIterator __first, _InputIterator __last, _OutputIterator __result)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation >`
`_OutputIterator adjacent_difference (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryOperation __binary_op)`
- `template<typename _Filter >`
`_Filter adjacent_find (_Filter, _Filter)`
- `template<typename _Filter, typename _BinaryPredicate >`
`_Filter adjacent_find (_Filter, _Filter, _BinaryPredicate)`
- `template<typename _ForwardIterator >`
`_ForwardIterator adjacent_find (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _BinaryPredicate >`
`_ForwardIterator adjacent_find (_ForwardIterator __first, _ForwardIterator __last, _BinaryPredicate __binary_pred)`
- `template<typename _InputIterator, typename _Distance >`
`void advance (_InputIterator &__i, _Distance __n)`

- `template<typename _Iter, typename _Predicate >`
`bool all_of (_Iter, _Iter, _Predicate)`
- `template<typename _InputIterator, typename _Predicate >`
`bool all_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _Tp, typename _Alloc, typename... _Args>`
`shared_ptr<_Tp > allocate_shared (const _Alloc &__a, _Args &&... __args)`
- `template<typename _Iter, typename _Predicate >`
`bool any_of (_Iter, _Iter, _Predicate)`
- `template<typename _InputIterator, typename _Predicate >`
`bool any_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<class _Dom >`
`_Expr< _UnClos< _Asin, _Expr, _Dom >, typename _Dom::value_type > asin (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`
`_Expr< _UnClos< _Asin, _ValArray, _Tp >, _Tp > asin (const valarray< _Tp > &__v)`
- `template<class _Dom >`
`_Expr< _UnClos< _Atan, _Expr, _Dom >, typename _Dom::value_type > atan (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`
`_Expr< _UnClos< _Atan, _ValArray, _Tp >, _Tp > atan (const valarray< _Tp > &__v)`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< _Atan2, _Expr, _Expr, _Dom1, _Dom2 >, typename _Dom1::value_type > atan2 (const _Expr< _Dom1, typename _Dom1::value_type > &__e1, const _Expr< _Dom2, typename _Dom2::value_type > &__e2)`
- `template<class _Dom >`
`_Expr< _BinClos< _Atan2, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename _Dom::value_type > atan2 (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< _Atan2, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename _Dom::value_type > atan2 (const valarray< typename _Dom::valarray > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`
`_Expr< _BinClos< _Atan2, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename _Dom::value_type > atan2 (const _Expr< _Dom, typename _Dom::value_type > &__e, const typename _Dom::value_type &__t)`
- `template<class _Dom >`
`_Expr< _BinClos< _Atan2, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename _Dom::value_type > atan2 (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`
`_Expr< _BinClos< _Atan2, _ValArray, _ValArray, _Tp, _Tp >, _Tp > atan2 (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< _Atan2, _Constant, _ValArray, _Tp, _Tp >, _Tp > atan2 (const _Tp &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`
`_Expr< _BinClos< _Atan2, _ValArray, _Constant, _Tp, _Tp >, _Tp > atan2 (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Container >`
`back_insert_iterator<_Container > back_inserter (_Container &__x)`
- `template<class _Container >`
`auto begin (_Container &__cont) -> decltype(__cont.begin())`

- `template<class _Container >`
`auto begin (const _Container &__cont) -> decltype(__cont.begin())`
- `template<class _Tp, size_t _Nm>`
`_Tp * begin (_Tp(&__arr)[_Nm])`
- `template<typename _Filter, typename _Tp >`
`bool binary_search (_Filter, _Filter, const _Tp &)`
- `template<typename _Filter, typename _Tp, typename _Compare >`
`bool binary_search (_Filter, _Filter, const _Tp &, _Compare)`
- `template<typename _ForwardIterator, typename _Tp >`
`bool binary_search (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`
`bool binary_search (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)`
- `template<typename _Operation, typename _Tp >`
`binder1st< _Operation > bind1st (const _Operation &__fn, const _Tp &__x)`
- `template<typename _Operation, typename _Tp >`
`binder2nd< _Operation > bind2nd (const _Operation &__fn, const _Tp &__x)`
- `ios_base & boolalpha (ios_base &__base)`
- `template<typename _Tp, typename _Tp1 >`
`shared_ptr< _Tp > const_pointer_cast (const shared_ptr< _Tp1 > &__r) noexcept`
- `template<typename _Tp, typename _Tp1, _Lock_policy _Lp>`
`__shared_ptr< _Tp, _Lp > const_pointer_cast (const __shared_ptr< _Tp1, _Lp > &__r) noexcept`
- `template<typename _Iter, typename _OIter >`
`_OIter copy (_Iter, _Iter, _OIter)`
- `template<typename _CharT >`
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT > >::__type copy (istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, ostreambuf_iterator< _CharT > __result)`
- `template<typename _Tp >`
`_Deque_iterator< _Tp, _Tp &, _Tp * > copy (_Deque_iterator< _Tp, const _Tp &, const _Tp * >, _Deque_iterator< _Tp, const _Tp &, const _Tp * >, _Deque_iterator< _Tp, _Tp &, _Tp * >)`
- `template<typename _Tp >`
`_Deque_iterator< _Tp, _Tp &, _Tp * > copy (_Deque_iterator< _Tp, _Tp &, _Tp * > __first, _Deque_iterator< _Tp, _Tp &, _Tp * > __last, _Deque_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _II, typename _OI >`
`_OI copy (_II __first, _II __last, _OI __result)`
- `template<typename _BIter1, typename _BIter2 >`
`_BIter2 copy_backward (_BIter1, _BIter1, _BIter2)`
- `template<typename _Tp >`
`_Deque_iterator< _Tp, _Tp &, _Tp * > copy_backward (_Deque_iterator< _Tp, const _Tp &, const _Tp * >, _Deque_iterator< _Tp, const _Tp &, const _Tp * >, _Deque_iterator< _Tp, _Tp &, _Tp * >)`
- `template<typename _Tp >`
`_Deque_iterator< _Tp, _Tp &, _Tp * > copy_backward (_Deque_iterator< _Tp, _Tp &, _Tp * > __first, _Deque_iterator< _Tp, _Tp &, _Tp * > __last, _Deque_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _BI1, typename _BI2 >`
`_BI2 copy_backward (_BI1 __first, _BI1 __last, _BI2 __result)`
- `template<typename _Ex >`
`exception_ptr copy_exception (_Ex __ex) noexcept 1`
- `template<typename _Iter, typename _OIter, typename _Predicate >`
`_OIter copy_if (_Iter, _Iter, _OIter, _Predicate)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate >`
`_OutputIterator copy_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred)`
- `template<typename _Iter, typename _Size, typename _OIter >`
`_OIter copy_n (_Iter, _Size, _OIter)`

- `template<typename _InputIterator, typename _Size, typename _OutputIterator >`
`_OutputIterator copy_n (_InputIterator __first, _Size __n, _OutputIterator __result)`
- `template<class _Dom >`
`_Expr< _UnClos< _Cos, _Expr, _Dom >, typename _Dom::value_type > cos (const _Expr< _Dom, typename`
`_Dom::value_type > &__e)`
- `template<typename _Tp >`
`_Expr< _UnClos< _Cos, _ValArray, _Tp >, _Tp > cos (const valarray< _Tp > &__v)`
- `template<class _Dom >`
`_Expr< _UnClos< _Cosh, _Expr, _Dom >, typename _Dom::value_type > cosh (const _Expr< _Dom, type-`
`name _Dom::value_type > &__e)`
- `template<typename _Tp >`
`_Expr< _UnClos< _Cosh, _ValArray, _Tp >, _Tp > cosh (const valarray< _Tp > &__v)`
- `template<typename _Iter, typename _Tp >`
`iterator_traits< _Iter >::difference_type count (_Iter, _Iter, const _Tp &)`
- `template<typename _InputIterator, typename _Tp >`
`iterator_traits< _InputIterator >::difference_type count (_InputIterator __first, _InputIterator __last, const _Tp &←`
`__value)`
- `template<typename _Iter, typename _Predicate >`
`iterator_traits< _Iter >::difference_type count_if (_Iter, _Iter, _Predicate)`
- `template<typename _InputIterator, typename _Predicate >`
`iterator_traits< _InputIterator >::difference_type count_if (_InputIterator __first, _InputIterator __last, _Predicate`
`__pred)`
- `exception_ptr current_exception () noexcept`
- `ios_base & dec (ios_base & __base)`
- `template<typename _InputIterator >`
`iterator_traits< _InputIterator >::difference_type distance (_InputIterator __first, _InputIterator __last)`
- `template<typename _Tp, typename _Tp1 >`
`shared_ptr< _Tp > dynamic_pointer_cast (const shared_ptr< _Tp1 > &__r) noexcept`
- `template<typename _Tp, typename _Tp1, _Lock_policy _Lp>`
`__shared_ptr< _Tp, _Lp > dynamic_pointer_cast (const __shared_ptr< _Tp1, _Lp > &__r) noexcept`
- `template<class _Container >`
`auto end (_Container &__cont) -> decltype(__cont.end())`
- `template<class _Container >`
`auto end (const _Container &__cont) -> decltype(__cont.end())`
- `template<class _Tp, size_t _Nm>`
`_Tp * end (_Tp(&__arr)[_Nm])`
- `template<typename _Iter1, typename _Iter2 >`
`bool equal (_Iter1, _Iter1, _Iter2)`
- `template<typename _Iter1, typename _Iter2, typename _BinaryPredicate >`
`bool equal (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _BinaryPredicate __binary_pred)`
- `template<typename _I1, typename _I2 >`
`bool equal (_I1 __first1, _I1 __last1, _I2 __first2)`
- `template<typename _Filter, typename _Tp >`
`pair< _Filter, _Filter > equal_range (_Filter, _Filter, const _Tp &)`
- `template<typename _Filter, typename _Tp, typename _Compare >`
`pair< _Filter, _Filter > equal_range (_Filter, _Filter, const _Tp &, _Compare)`
- `template<typename _ForwardIterator, typename _Tp >`
`pair< _ForwardIterator, _ForwardIterator > equal_range (_ForwardIterator __first, _ForwardIterator __last, const`
`_Tp &__val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`
`pair< _ForwardIterator, _ForwardIterator > equal_range (_ForwardIterator __first, _ForwardIterator __last, const`
`_Tp &__val, _Compare __comp)`

- `template<class _Dom >`
`_Expr< _UnClos< _Exp, _Expr, _Dom >, typename _Dom::value_type > exp (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`
`_Expr< _UnClos< _Exp, _ValArray, _Tp >, _Tp > exp (const valarray< _Tp > &__v)`
- `template<typename _Filter, typename _Tp >`
`void fill (_Filter, _Filter, const _Tp &)`
- `template<typename _Tp >`
`void fill (const _Deque_iterator< _Tp, _Tp &, _Tp * > &, const _Deque_iterator< _Tp, _Tp &, _Tp * > &, const _Tp &)`
- `void fill (_Bit_iterator __first, _Bit_iterator __last, const bool &__x)`
- `template<typename _ForwardIterator, typename _Tp >`
`void fill (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)`
- `template<typename _OIter, typename _Size, typename _Tp >`
`_OIter fill_n (_OIter, _Size, const _Tp &)`
- `template<typename _OI, typename _Size, typename _Tp >`
`_OI fill_n (_OI __first, _Size __n, const _Tp &__value)`
- `template<typename _CharT >`
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, istreambuf_iterator< _CharT >::__type find (istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, const _CharT &__val)`
- `template<typename _Iter, typename _Tp >`
`_Iter find (_Iter, _Iter, const _Tp &)`
- `template<typename _InputIterator, typename _Tp >`
`_InputIterator find (_InputIterator __first, _InputIterator __last, const _Tp &__val)`
- `template<typename _Filter1, typename _Filter2 >`
`_Filter1 find_end (_Filter1, _Filter1, _Filter2, _Filter2)`
- `template<typename _Filter1, typename _Filter2, typename _BinaryPredicate >`
`_Filter1 find_end (_Filter1, _Filter1, _Filter2, _Filter2, _BinaryPredicate)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`
`_ForwardIterator1 find_end (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, ↵
_FowardIterator2 __last2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate >`
`_ForwardIterator1 find_end (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, ↵
_FowardIterator2 __last2, _BinaryPredicate __comp)`
- `template<typename _Filter1, typename _Filter2 >`
`_Filter1 find_first_of (_Filter1, _Filter1, _Filter2, _Filter2)`
- `template<typename _Filter1, typename _Filter2, typename _BinaryPredicate >`
`_Filter1 find_first_of (_Filter1, _Filter1, _Filter2, _Filter2, _BinaryPredicate)`
- `template<typename _InputIterator, typename _ForwardIterator >`
`_InputIterator find_first_of (_InputIterator __first1, _InputIterator __last1, _ForwardIterator __first2, _Forward↵
Iterator __last2)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _BinaryPredicate >`
`_InputIterator find_first_of (_InputIterator __first1, _InputIterator __last1, _ForwardIterator __first2, _Forward↵
Iterator __last2, _BinaryPredicate __comp)`
- `template<typename _Iter, typename _Predicate >`
`_Iter find_if (_Iter, _Iter, _Predicate)`
- `template<typename _InputIterator, typename _Predicate >`
`_InputIterator find_if (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _Iter, typename _Predicate >`
`_Iter find_if_not (_Iter, _Iter, _Predicate)`
- `template<typename _InputIterator, typename _Predicate >`
`_InputIterator find_if_not (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `ios_base & fixed (ios_base &__base)`

- `template<typename _Iter, typename _Funct >`
`_Funct for_each (_Iter, _Iter, _Funct)`
- `template<typename _InputIterator, typename _Function >`
`_Function for_each (_InputIterator __first, _InputIterator __last, _Function __f)`
- `template<typename _Tp >`
`constexpr _Tp && forward (typename std::remove_reference< _Tp >::type &__t) noexcept`
- `template<typename _Tp >`
`constexpr _Tp && forward (typename std::remove_reference< _Tp >::type &&__t) noexcept`
- `template<typename _Container >`
`front_insert_iterator< _Container > front_inserter (_Container &__x)`
- `template<typename _Filter, typename _Generator >`
`void generate (_Filter, _Filter, _Generator)`
- `template<typename _ForwardIterator, typename _Generator >`
`void generate (_ForwardIterator __first, _ForwardIterator __last, _Generator __gen)`
- `template<typename _RealType, size_t __bits, typename _UniformRandomNumberGenerator >`
`_RealType generate_canonical (_UniformRandomNumberGenerator &__g)`
- `template<typename _OIter, typename _Size, typename _Generator >`
`_OIter generate_n (_OIter, _Size, _Generator)`
- `template<typename _OutputIterator, typename _Size, typename _Generator >`
`_OutputIterator generate_n (_OutputIterator __first, _Size __n, _Generator __gen)`
- `template<typename _Del, typename _Tp, _Lock_policy _Lp>`
`_Del * get_deleter (const __shared_ptr< _Tp, _Lp > &__p) noexcept`
- `template<typename _Tp >`
`pair< _Tp *, ptrdiff_t > get_temporary_buffer (ptrdiff_t __len) noexcept`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`basic_istream< _CharT, _Traits > & getline (basic_istream< _CharT, _Traits > &__is, __gnu_cxx::__versa_↵
string< _CharT, _Traits, _Alloc, _Base > &__str, _CharT __delim)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`basic_istream< _CharT, _Traits > & getline (basic_istream< _CharT, _Traits > &__is, __gnu_cxx::__versa_↵
string< _CharT, _Traits, _Alloc, _Base > &__str)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`basic_istream< _CharT, _Traits > & getline (basic_istream< _CharT, _Traits > &__is, basic_string< _CharT,
_Traits, _Alloc > &__str, _CharT __delim)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`basic_istream< _CharT, _Traits > & getline (basic_istream< _CharT, _Traits > &__is, basic_string< _CharT,
_Traits, _Alloc > &__str)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`basic_istream< _CharT, _Traits > & getline (basic_istream< _CharT, _Traits > &&__is, basic_string< _CharT,
_Traits, _Alloc > &__str, _CharT __delim)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`basic_istream< _CharT, _Traits > & getline (basic_istream< _CharT, _Traits > &&__is, basic_string< _CharT,
_Traits, _Alloc > &__str)`
- `template<>`
`basic_istream< char > & getline (basic_istream< char > &__in, basic_string< char > &__str, char __delim)`
- `template<>`
`basic_istream< wchar_t > & getline (basic_istream< wchar_t > &__in, basic_string< wchar_t > &__str,
wchar_t __delim)`
- `template<typename _Facet >`
`bool has_facet (const locale &) throw ()`
- `ios_base & hex (ios_base &__base)`
- `template<typename _Iter1, typename _Iter2 >`
`bool includes (_Iter1, _Iter1, _Iter2, _Iter2)`

- `template<typename _Iter1, typename _Iter2, typename _Compare >`
`bool includes (_Iter1, _Iter1, _Iter2, _Iter2, _Compare)`
- `template<typename _InputIterator1, typename _InputIterator2 >`
`bool includes (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Compare >`
`bool includes (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Tp >`
`_Tp inner_product (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Tp __init)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Tp, typename _BinaryOperation1, typename _BinaryOperation2 >`
`_Tp inner_product (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Tp __init, _BinaryOperation1 __binary_op1, _BinaryOperation2 __binary_op2)`
- `template<typename _BIter >`
`void inplace_merge (_BIter, _BIter, _BIter)`
- `template<typename _BIter, typename _Compare >`
`void inplace_merge (_BIter, _BIter, _BIter, _Compare)`
- `template<typename _BidirectionalIterator >`
`void inplace_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _Compare >`
`void inplace_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _Container, typename _Iterator >`
`insert_iterator< _Container > inserter (_Container &__x, _Iterator __i)`
- `ios_base & internal (ios_base &__base)`
- `template<typename _ForwardIterator, typename _Tp >`
`void iota (_ForwardIterator __first, _ForwardIterator __last, _Tp __value)`
- `template<typename _RAIter >`
`bool is_heap (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`
`bool is_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _RandomAccessIterator >`
`bool is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`bool is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RAIter >`
`_RAIter is_heap_until (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`
`_RAIter is_heap_until (_RAIter, _RAIter, _Compare)`
- `template<typename _RandomAccessIterator >`
`_RandomAccessIterator is_heap_until (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`_RandomAccessIterator is_heap_until (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _Iter, typename _Predicate >`
`bool is_partitioned (_Iter, _Iter, _Predicate)`
- `template<typename _InputIterator, typename _Predicate >`
`bool is_partitioned (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _Filter1, typename _Filter2 >`
`bool is_permutation (_Filter1, _Filter1, _Filter2)`
- `template<typename _Filter1, typename _Filter2, typename _BinaryPredicate >`
`bool is_permutation (_Filter1, _Filter1, _Filter2, _BinaryPredicate)`

- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`
`bool is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate >`
`bool is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _BinaryPredicate __pred)`
- `template<typename _Filter >`
`bool is_sorted (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare >`
`bool is_sorted (_Filter, _Filter, _Compare)`
- `template<typename _ForwardIterator >`
`bool is_sorted (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >`
`bool is_sorted (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _Filter >`
`_Filter is_sorted_until (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare >`
`_Filter is_sorted_until (_Filter, _Filter, _Compare)`
- `template<typename _ForwardIterator >`
`_ForwardIterator is_sorted_until (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >`
`_ForwardIterator is_sorted_until (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _CharT >`
`bool isalnum (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`
`bool isalpha (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`
`bool iscntrl (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`
`bool isdigit (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`
`bool isgraph (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`
`bool islower (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`
`bool isprint (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`
`bool ispunct (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`
`bool isspace (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`
`bool isupper (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`
`bool isxdigit (_CharT __c, const locale &__loc)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`
`void iter_swap (_ForwardIterator1 __a, _ForwardIterator2 __b)`
- `template<typename _Filter1, typename _Filter2 >`
`void iter_swap (_Filter1, _Filter2)`
- `template<typename _Tp >`
`_Tp kill_dependency (_Tp __y) noexcept`
- `ios_base & left (ios_base &__base)`
- `template<typename _Iter1, typename _Iter2 >`
`bool lexicographical_compare (_Iter1, _Iter1, _Iter2, _Iter2)`

- `template<typename _Iter1, typename _Iter2, typename _Compare >`
`bool lexicographical_compare (_Iter1, _Iter1, _Iter2, _Iter2, _Compare)`
- `template<typename _I1, typename _I2 >`
`bool lexicographical_compare (_I1 __first1, _I1 __last1, _I2 __first2, _I2 __last2)`
- `template<typename _I1, typename _I2, typename _Compare >`
`bool lexicographical_compare (_I1 __first1, _I1 __last1, _I2 __first2, _I2 __last2, _Compare __comp)`
- `template<class _Dom >`
`_Expr< _UnClos< _Log, _Expr, _Dom >, typename _Dom::value_type > log (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`
`_Expr< _UnClos< _Log, _ValArray, _Tp >, _Tp > log (const valarray< _Tp > &__v)`
- `template<typename _Tp >`
`_Expr< _UnClos< _Log10, _ValArray, _Tp >, _Tp > log10 (const valarray< _Tp > &__v)`
- `template<class _Dom >`
`_Expr< _UnClos< _Log10, _Expr, _Dom >, typename _Dom::value_type > log10 (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Filter, typename _Tp >`
`_Filter lower_bound (_Filter, _Filter, const _Tp &)`
- `template<typename _Filter, typename _Tp, typename _Compare >`
`_Filter lower_bound (_Filter, _Filter, const _Tp &, _Compare)`
- `template<typename _ForwardIterator, typename _Tp >`
`_ForwardIterator lower_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`
`_ForwardIterator lower_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)`
- `template<typename _Ex >`
`exception_ptr make_exception_ptr (_Ex __ex) noexcept`
- `template<typename _RAIter >`
`void make_heap (_RAIter, _RAIter)`
- `template<typename _RandomAccessIterator >`
`void make_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RAIter, typename _Compare >`
`void make_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void make_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _Iterator >`
`move_iterator< _Iterator > make_move_iterator (_Iterator __i)`
- `template<class _T1, class _T2 >`
`constexpr pair< typename __decay_and_strip< _T1 >::__type, typename __decay_and_strip< _T2 >::__type > make_pair (_T1 &&__x, _T2 &&__y)`
- `template<typename _Tp, typename... _Args>`
`shared_ptr< _Tp > make_shared (_Args &&...__args)`
- `template<typename _Tp >`
`const _Tp & max (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp, typename _Compare >`
`const _Tp & max (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _Tp >`
`_Tp max (initializer_list< _Tp >)`
- `template<typename _Tp, typename _Compare >`
`_Tp max (initializer_list< _Tp >, _Compare)`
- `template<typename _Filter >`
`_Filter max_element (_Filter, _Filter)`

- `template<typename _Filter, typename _Compare >`
`_Filter max_element (_Filter, _Filter, _Compare)`
- `template<typename _ForwardIterator >`
`_ForwardIterator max_element (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >`
`_ForwardIterator max_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _Ret, typename _Tp >`
`mem_fun_t< _Ret, _Tp > mem_fun (_Ret(_Tp::*__f)())`
- `template<typename _Ret, typename _Tp >`
`const_mem_fun_t< _Ret, _Tp > mem_fun (_Ret(_Tp::*__f)() const)`
- `template<typename _Ret, typename _Tp, typename _Arg >`
`mem_fun1_t< _Ret, _Tp, _Arg > mem_fun (_Ret(_Tp::*__f)(_Arg))`
- `template<typename _Ret, typename _Tp, typename _Arg >`
`const_mem_fun1_t< _Ret, _Tp, _Arg > mem_fun (_Ret(_Tp::*__f)(_Arg) const)`
- `template<typename _Ret, typename _Tp >`
`mem_fun_ref_t< _Ret, _Tp > mem_fun_ref (_Ret(_Tp::*__f)())`
- `template<typename _Ret, typename _Tp >`
`const_mem_fun_ref_t< _Ret, _Tp > mem_fun_ref (_Ret(_Tp::*__f)() const)`
- `template<typename _Ret, typename _Tp, typename _Arg >`
`mem_fun1_ref_t< _Ret, _Tp, _Arg > mem_fun_ref (_Ret(_Tp::*__f)(_Arg))`
- `template<typename _Ret, typename _Tp, typename _Arg >`
`const_mem_fun1_ref_t< _Ret, _Tp, _Arg > mem_fun_ref (_Ret(_Tp::*__f)(_Arg) const)`
- `template<typename _Ilter1, typename _Ilter2, typename _Olter >`
`_Olter merge (_Ilter1, _Ilter1, _Ilter2, _Ilter2, _Olter)`
- `template<typename _Ilter1, typename _Ilter2, typename _Olter, typename _Compare >`
`_Olter merge (_Ilter1, _Ilter1, _Ilter2, _Ilter2, _Olter, _Compare)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`
`_OutputIterator merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`
`_OutputIterator merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _Tp >`
`const _Tp & min (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp, typename _Compare >`
`const _Tp & min (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _Tp >`
`_Tp min (initializer_list< _Tp >)`
- `template<typename _Tp, typename _Compare >`
`_Tp min (initializer_list< _Tp >, _Compare)`
- `template<typename _Filter >`
`_Filter min_element (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare >`
`_Filter min_element (_Filter, _Filter, _Compare)`
- `template<typename _ForwardIterator >`
`_ForwardIterator min_element (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >`
`_ForwardIterator min_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _Tp >`
`pair< const _Tp &, const _Tp & > minmax (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp, typename _Compare >`
`pair< const _Tp &, const _Tp & > minmax (const _Tp &__a, const _Tp &__b, _Compare __comp)`

- `template<typename _Tp >`
`pair< _Tp, _Tp > minmax (initializer_list< _Tp >)`
- `template<typename _Tp, typename _Compare >`
`pair< _Tp, _Tp > minmax (initializer_list< _Tp >, _Compare)`
- `template<typename _Filter >`
`pair< _Filter, _Filter > minmax_element (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare >`
`pair< _Filter, _Filter > minmax_element (_Filter, _Filter, _Compare)`
- `template<typename _ForwardIterator >`
`pair< _ForwardIterator, _ForwardIterator > minmax_element (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >`
`pair< _ForwardIterator, _ForwardIterator > minmax_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _Iter1, typename _Iter2 >`
`pair< _Iter1, _Iter2 > mismatch (_Iter1, _Iter1, _Iter2)`
- `template<typename _Iter1, typename _Iter2, typename _BinaryPredicate >`
`pair< _Iter1, _Iter2 > mismatch (_Iter1, _Iter1, _Iter2, _BinaryPredicate)`
- `template<typename _InputIterator1, typename _InputIterator2 >`
`pair< _InputIterator1, _InputIterator2 > mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate >`
`pair< _InputIterator1, _InputIterator2 > mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _BinaryPredicate __binary_pred)`
- `template<typename _Tp >`
`constexpr std::remove_reference< _Tp >::type && move (_Tp && __t) noexcept`
- `template<typename _Tp >`
`_Deque_iterator< _Tp, _Tp &, _Tp * > move (_Deque_iterator< _Tp, const _Tp &, const _Tp * >, _Deque_iterator< _Tp, const _Tp &, const _Tp * >, _Deque_iterator< _Tp, _Tp &, _Tp * >)`
- `template<typename _Tp >`
`_Deque_iterator< _Tp, _Tp &, _Tp * > move (_Deque_iterator< _Tp, _Tp &, _Tp * > __first, _Deque_iterator< _Tp, _Tp &, _Tp * > __last, _Deque_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _II, typename _OI >`
`_OI move (_II __first, _II __last, _OI __result)`
- `template<typename _Tp >`
`_Deque_iterator< _Tp, _Tp &, _Tp * > move_backward (_Deque_iterator< _Tp, const _Tp &, const _Tp * >, _Deque_iterator< _Tp, const _Tp &, const _Tp * >, _Deque_iterator< _Tp, _Tp &, _Tp * >)`
- `template<typename _Tp >`
`_Deque_iterator< _Tp, _Tp &, _Tp * > move_backward (_Deque_iterator< _Tp, _Tp &, _Tp * > __first, _Deque_iterator< _Tp, _Tp &, _Tp * > __last, _Deque_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _BI1, typename _BI2 >`
`_BI2 move_backward (_BI1 __first, _BI1 __last, _BI2 __result)`
- `template<typename _Tp >`
`constexpr conditional< __move_if_noexcept_cond< _Tp >::value, const _Tp &, _Tp && >::type move_if_noexcept (_Tp & __x) noexcept`
- `template<typename _ForwardIterator >`
`_ForwardIterator next (_ForwardIterator __x, typename iterator_traits< _ForwardIterator >::difference_type __n=1)`
- `template<typename _BIter >`
`bool next_permutation (_BIter, _BIter)`
- `template<typename _BIter, typename _Compare >`
`bool next_permutation (_BIter, _BIter, _Compare)`
- `template<typename _BidirectionalIterator >`
`bool next_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last)`

- `template<typename _BidirectionalIterator, typename _Compare >`
`bool next_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __comp)`
- `ios_base & noboolalpha (ios_base & __base)`
- `template<typename _Iter, typename _Predicate >`
`bool none_of (_Iter, _Iter, _Predicate)`
- `template<typename _InputIterator, typename _Predicate >`
`bool none_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `ios_base & noshowbase (ios_base & __base)`
- `ios_base & noshowpoint (ios_base & __base)`
- `ios_base & noshowpos (ios_base & __base)`
- `ios_base & noskipws (ios_base & __base)`
- `template<typename _Predicate >`
`unary_negate< _Predicate > not1 (const _Predicate & __pred)`
- `template<typename _Predicate >`
`binary_negate< _Predicate > not2 (const _Predicate & __pred)`
- `ios_base & nounitbuf (ios_base & __base)`
- `ios_base & nouppercase (ios_base & __base)`
- `template<typename _RAIter >`
`void nth_element (_RAIter, _RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`
`void nth_element (_RAIter, _RAIter, _RAIter, _Compare)`
- `template<typename _RandomAccessIterator >`
`void nth_element (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __↵
last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void nth_element (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __↵
last, _Compare __comp)`
- `ios_base & oct (ios_base & __base)`
- `template<class _Tp, class _CharT, class _Traits, class _Dist >`
`bool operator!= (const istream_iterator< _Tp, _CharT, _Traits, _Dist > & __x, const istream_iterator< _Tp, _↵
CharT, _Traits, _Dist > & __y)`
- `template<typename _T1, typename _T2 >`
`bool operator!= (const allocator< _T1 > &, const allocator< _T2 > &)`
- `template<typename _Tp >`
`bool operator!= (const allocator< _Tp > &, const allocator< _Tp > &)`
- `template<typename _CharT, typename _Traits >`
`bool operator!= (const istreambuf_iterator< _CharT, _Traits > & __a, const istreambuf_iterator< _CharT, _Traits
> & __b)`
- `template<typename _StateT >`
`bool operator!= (const fpos< _StateT > & __lhs, const fpos< _StateT > & __rhs)`
- `template<class _T1, class _T2 >`
`constexpr bool operator!= (const pair< _T1, _T2 > & __x, const pair< _T1, _T2 > & __y)`
- `template<typename _Tp, typename _Ref, typename _Ptr >`
`bool operator!= (const _Deque_iterator< _Tp, _Ref, _Ptr > & __x, const _Deque_iterator< _Tp, _Ref, _Ptr >
& __y) noexcept`
- `template<typename _Tp >`
`bool operator!= (const _Fwd_list_iterator< _Tp > & __x, const _Fwd_list_const_iterator< _Tp > & __y) noexcept`
- `template<typename _Tp, typename _Seq >`
`bool operator!= (const stack< _Tp, _Seq > & __x, const stack< _Tp, _Seq > & __y)`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR >`
`bool operator!= (const _Deque_iterator< _Tp, _RefL, _PtrL > & __x, const _Deque_iterator< _Tp, _RefR, _PtrR
> & __y) noexcept`

- `template<typename _Val >`
`bool operator!= (const _List_iterator< _Val > &__x, const _List_const_iterator< _Val > &__y) noexcept`
- `template<typename _Tp, typename _Seq >`
`bool operator!= (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)`
- `template<typename _Iterator >`
`bool operator!= (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`
- `template<typename _Val >`
`bool operator!= (const _Rb_tree_iterator< _Val > &__x, const _Rb_tree_const_iterator< _Val > &__y) noexcept`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator!= (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept`
- `template<typename _IteratorL, typename _IteratorR >`
`bool operator!= (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`
- `template<typename _Tp >`
`bool operator!= (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp >`
`bool operator!= (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>`
`bool operator!= (const std::linear_congruential_engine< _UIntType, __a, __c, __m > &__lhs, const std::linear_congruential_engine< _UIntType, __a, __c, __m > &__rhs)`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __not_equal_to, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __not_equal_to, typename _Dom1::value_type >::result_type > operator!= (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`
`_Expr< _BinClos< __not_equal_to, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __not_equal_to, typename _Dom::value_type >::result_type > operator!= (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __not_equal_to, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __not_equal_to, typename _Dom::value_type >::result_type > operator!= (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`
`_Expr< _BinClos< __not_equal_to, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< __not_equal_to, typename _Dom::value_type >::result_type > operator!= (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __not_equal_to, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< __not_equal_to, typename _Dom::value_type >::result_type > operator!= (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`
`bool operator!= (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f>`
`bool operator!= (const std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > &__lhs, const std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > &__rhs)`
- `template<typename _Tp, typename _Dp >`
`bool operator!= (const unique_ptr< _Tp, _Dp > &__x, nullptr_t) noexcept`
- `template<typename _Tp, typename _Dp >`
`bool operator!= (nullptr_t, const unique_ptr< _Tp, _Dp > &__x) noexcept`
- `template<typename _Key, typename _Compare, typename _Alloc >`
`bool operator!= (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)`

- `template<typename _Key, typename _Compare, typename _Alloc >`
`bool operator!= (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)`
- `template<typename _UIntType, size_t __w, size_t __s, size_t __r>`
`bool operator!= (const std::subtract_with_carry_engine< _UIntType, __w, __s, __r > &__lhs, const std::subtract_with_carry_engine< _UIntType, __w, __s, __r > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`
`bool operator!= (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Bilter >`
`bool operator!= (const sub_match< _Bilter > &__lhs, const sub_match< _Bilter > &__rhs)`
- `template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, typename _Alloc >`
`bool operator!= (const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__x, const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`
`bool operator!= (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool operator!= (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool operator!= (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`
- `template<typename _RandomNumberEngine, size_t __p, size_t __r>`
`bool operator!= (const std::discard_block_engine< _RandomNumberEngine, __p, __r > &__lhs, const std::discard_block_engine< _RandomNumberEngine, __p, __r > &__rhs)`
- `template<typename _Iterator >`
`bool operator!= (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool operator!= (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__rhs)`
- `template<typename _Tp1, typename _Tp2, _Lock_policy _Lp>`
`bool operator!= (const __shared_ptr< _Tp1, _Lp > &__a, const __shared_ptr< _Tp2, _Lp > &__b) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>`
`bool operator!= (const __shared_ptr< _Tp, _Lp > &__a, nullptr_t) noexcept`
- `template<typename _Bi_iter >`
`bool operator!= (typename iterator_traits< _Bi_iter >::value_type const *__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Tp, _Lock_policy _Lp>`
`bool operator!= (nullptr_t, const __shared_ptr< _Tp, _Lp > &__a) noexcept`
- `template<typename _RandomNumberEngine, size_t __w, typename _UIntType >`
`bool operator!= (const std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > &__lhs, const std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > &__rhs)`
- `template<typename _Bi_iter >`
`bool operator!= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const *__rhs)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc >`
`bool operator!= (const unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc >`
`bool operator!= (const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Bi_iter >`
`bool operator!= (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter > &__rhs)`

- `template<typename _Tp, typename _Alloc >`
`bool operator!= (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Bi_iter >`
`bool operator!= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc >`
`bool operator!= (const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc >`
`bool operator!= (const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _RandomNumberEngine, size_t __k>`
`bool operator!= (const std::shuffle_order_engine< _RandomNumberEngine, __k > &__lhs, const std::shuffle_order_engine< _RandomNumberEngine, __k > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator!= (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool operator!= (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`
- `template<typename _IntType >`
`bool operator!= (const std::uniform_int_distribution< _IntType > &__d1, const std::uniform_int_distribution< _IntType > &__d2)`
- `template<typename _Bi_iter, class _Alloc >`
`bool operator!= (const match_results< _Bi_iter, _Alloc > &__m1, const match_results< _Bi_iter, _Alloc > &__m2)`
- `template<typename _IntType >`
`bool operator!= (const std::uniform_real_distribution< _IntType > &__d1, const std::uniform_real_distribution< _IntType > &__d2)`
- `template<typename _Tp, typename _Alloc >`
`bool operator!= (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc > &__y)`
- `template<typename _RealType >`
`bool operator!= (const std::normal_distribution< _RealType > &__d1, const std::normal_distribution< _RealType > &__d2)`
- `template<typename _RealType >`
`bool operator!= (const std::lognormal_distribution< _RealType > &__d1, const std::lognormal_distribution< _RealType > &__d2)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool operator!= (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool operator!= (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool operator!= (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT *__rhs)`
- `template<typename _RealType >`
`bool operator!= (const std::gamma_distribution< _RealType > &__d1, const std::gamma_distribution< _RealType > &__d2)`
- `template<typename _RealType >`
`bool operator!= (const std::chi_squared_distribution< _RealType > &__d1, const std::chi_squared_distribution< _RealType > &__d2)`
- `template<typename _RealType >`
`bool operator!= (const std::cauchy_distribution< _RealType > &__d1, const std::cauchy_distribution< _RealType > &__d2)`
- `template<typename _RealType >`
`bool operator!= (const std::fisher_f_distribution< _RealType > &__d1, const std::fisher_f_distribution< _RealType > &__d2)`

- `template<typename _RealType >`
`bool operator!= (const std::student_t_distribution< _RealType > &__d1, const std::student_t_distribution< _RealType > &__d2)`
- `bool operator!= (const std::bernoulli_distribution &__d1, const std::bernoulli_distribution &__d2)`
- `template<typename _IntType >`
`bool operator!= (const std::binomial_distribution< _IntType > &__d1, const std::binomial_distribution< _IntType > &__d2)`
- `template<typename _IntType >`
`bool operator!= (const std::geometric_distribution< _IntType > &__d1, const std::geometric_distribution< _IntType > &__d2)`
- `template<typename _IntType >`
`bool operator!= (const std::negative_binomial_distribution< _IntType > &__d1, const std::negative_binomial_distribution< _IntType > &__d2)`
- `template<typename _IntType >`
`bool operator!= (const std::poisson_distribution< _IntType > &__d1, const std::poisson_distribution< _IntType > &__d2)`
- `template<typename _RealType >`
`bool operator!= (const std::exponential_distribution< _RealType > &__d1, const std::exponential_distribution< _RealType > &__d2)`
- `template<typename _RealType >`
`bool operator!= (const std::weibull_distribution< _RealType > &__d1, const std::weibull_distribution< _RealType > &__d2)`
- `template<typename _RealType >`
`bool operator!= (const std::extreme_value_distribution< _RealType > &__d1, const std::extreme_value_distribution< _RealType > &__d2)`
- `template<typename _IntType >`
`bool operator!= (const std::discrete_distribution< _IntType > &__d1, const std::discrete_distribution< _IntType > &__d2)`
- `template<typename _RealType >`
`bool operator!= (const std::piecewise_constant_distribution< _RealType > &__d1, const std::piecewise_constant_distribution< _RealType > &__d2)`
- `template<typename _RealType >`
`bool operator!= (const std::piecewise_linear_distribution< _RealType > &__d1, const std::piecewise_linear_distribution< _RealType > &__d2)`
- `template<class _Dom >`
`_Expr< _BinClos< __modulus, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __modulus, typename _Dom::value_type >::result_type > operator% (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __modulus, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __modulus, typename _Dom1::value_type >::result_type > operator% (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`
`_Expr< _BinClos< __modulus, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< __modulus, typename _Dom::value_type >::result_type > operator% (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`
`_Expr< _BinClos< __modulus, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __modulus, typename _Dom::value_type >::result_type > operator% (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __modulus, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< __modulus, typename _Dom::value_type >::result_type > operator% (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`

- constexpr `_los_Fmtflags` **operator&** (`_los_Fmtflags __a, _los_Fmtflags __b`)
- constexpr `memory_order` **operator&** (`memory_order __m, __memory_order_modifier __mod`)
- constexpr `_los_Openmode` **operator&** (`_los_Openmode __a, _los_Openmode __b`)
- constexpr `_los_losestate` **operator&** (`_los_losestate __a, _los_losestate __b`)
- template<class `_Dom1`, class `_Dom2` >
`_Expr< _BinClos< __bitwise_and, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __bitwise_and, type-
name _Dom1::value_type >::result_type > operator& (const _Expr< _Dom1, typename _Dom1::value_type >
&__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- template<class `_Dom` >
`_Expr< _BinClos< __bitwise_and, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun<
__bitwise_and, typename _Dom::value_type >::result_type > operator& (const _Expr< _Dom, typename
_Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- template<class `_Dom` >
`_Expr< _BinClos< __bitwise_and, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun<
__bitwise_and, typename _Dom::value_type >::result_type > operator& (const typename _Dom::value_type
&__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- template<class `_Dom` >
`_Expr< _BinClos< __bitwise_and, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun<
__bitwise_and, typename _Dom::value_type >::result_type > operator& (const _Expr< _Dom, typename
_Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- template<class `_Dom` >
`_Expr< _BinClos< __bitwise_and, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun<
__bitwise_and, typename _Dom::value_type >::result_type > operator& (const valarray< typename _Dom::
value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- template<class `_Dom` >
`_Expr< _BinClos< __logical_and, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun<
__logical_and, typename _Dom::value_type >::result_type > operator&& (const typename _Dom::value_type
&__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- template<class `_Dom1`, class `_Dom2` >
`_Expr< _BinClos< __logical_and, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __logical_and, typename
_Dom1::value_type >::result_type > operator&& (const _Expr< _Dom1, typename _Dom1::value_type > &__v,
const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- template<class `_Dom` >
`_Expr< _BinClos< __logical_and, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun<
__logical_and, typename _Dom::value_type >::result_type > operator&& (const _Expr< _Dom, typename
_Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- template<class `_Dom` >
`_Expr< _BinClos< __logical_and, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun<
__logical_and, typename _Dom::value_type >::result_type > operator&& (const _Expr< _Dom, typename
_Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- template<class `_Dom` >
`_Expr< _BinClos< __logical_and, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun<
__logical_and, typename _Dom::value_type >::result_type > operator&& (const valarray< typename _Dom::
value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- const `_los_Fmtflags` & **operator&=** (`_los_Fmtflags &__a, _los_Fmtflags __b`)
- const `_los_Openmode` & **operator&=** (`_los_Openmode &__a, _los_Openmode __b`)
- const `_los_losestate` & **operator&=** (`_los_losestate &__a, _los_losestate __b`)
- template<class `_Dom` >
`_Expr< _BinClos< __multiplies, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun<
__multiplies, typename _Dom::value_type >::result_type > operator* (const _Expr< _Dom, typename _Dom::
value_type > &__v, const typename _Dom::value_type &__t)`

- `template<class _Dom >`
`_Expr< _BinClos< __multiplies, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __multiplies, typename _Dom::value_type >::result_type > operator* (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __multiplies, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __multiplies, typename _Dom1::value_type >::result_type > operator* (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`
`_Expr< _BinClos< __multiplies, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< __multiplies, typename _Dom::value_type >::result_type > operator* (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __multiplies, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __multiplies, typename _Dom::value_type >::result_type > operator* (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `_Bit_iterator operator+ (ptrdiff_t __n, const _Bit_iterator &__x)`
- `template<typename _Iterator >`
`reverse_iterator< _Iterator > operator+ (typename reverse_iterator< _Iterator >::difference_type __n, const reverse_iterator< _Iterator > &__x)`
- `template<typename _Tp, typename _Ref, typename _Ptr >`
`_Deque_iterator< _Tp, _Ref, _Ptr > operator+ (ptrdiff_t __n, const _Deque_iterator< _Tp, _Ref, _Ptr > &__x) noexcept`
- `_Bit_const_iterator operator+ (ptrdiff_t __n, const _Bit_const_iterator &__x)`
- `template<class _Dom >`
`_Expr< _BinClos< __plus, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< __plus, typename _Dom::value_type >::result_type > operator+ (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __plus, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __plus, typename _Dom::value_type >::result_type > operator+ (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __plus, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __plus, typename _Dom::value_type >::result_type > operator+ (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __plus, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __plus, typename _Dom1::value_type >::result_type > operator+ (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`
`_Expr< _BinClos< __plus, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< __plus, typename _Dom::value_type >::result_type > operator+ (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<typename _Iterator >`
`move_iterator< _Iterator > operator+ (typename move_iterator< _Iterator >::difference_type __n, const move_iterator< _Iterator > &__x)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`basic_string< _CharT, _Traits, _Alloc > operator+ (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`basic_string< _CharT, _Traits, _Alloc > operator+ (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`

- `template<typename _CharT, typename _Traits, typename _Alloc >`
`basic_string< _CharT, _Traits, _Alloc > operator+ (_CharT __lhs, const basic_string< _CharT, _Traits, _Alloc >`
`&__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`basic_string< _CharT, _Traits, _Alloc > operator+ (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const`
`_CharT * __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`basic_string< _CharT, _Traits, _Alloc > operator+ (const basic_string< _CharT, _Traits, _Alloc > &__lhs, _CharT`
`__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`basic_string< _CharT, _Traits, _Alloc > operator+ (basic_string< _CharT, _Traits, _Alloc > &&__lhs, const`
`basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`basic_string< _CharT, _Traits, _Alloc > operator+ (const basic_string< _CharT, _Traits, _Alloc > &__lhs,`
`basic_string< _CharT, _Traits, _Alloc > &&__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`basic_string< _CharT, _Traits, _Alloc > operator+ (basic_string< _CharT, _Traits, _Alloc > &&__lhs, basic_↵`
`string< _CharT, _Traits, _Alloc > &&__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`basic_string< _CharT, _Traits, _Alloc > operator+ (const _CharT * __lhs, basic_string< _CharT, _Traits, _Alloc`
`> &&__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`basic_string< _CharT, _Traits, _Alloc > operator+ (_CharT __lhs, basic_string< _CharT, _Traits, _Alloc > &&↵`
`__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`basic_string< _CharT, _Traits, _Alloc > operator+ (basic_string< _CharT, _Traits, _Alloc > &&__lhs, const ↵`
`_CharT * __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`basic_string< _CharT, _Traits, _Alloc > operator+ (basic_string< _CharT, _Traits, _Alloc > &&__lhs, _CharT`
`__rhs)`
- `ptrdiff_t operator- (const _Bit_iterator_base &__x, const _Bit_iterator_base &__y)`
- `template<typename _Iterator >`
`reverse_iterator< _Iterator >::difference_type operator- (const reverse_iterator< _Iterator > &__x, const`
`reverse_iterator< _Iterator > &__y)`
- `template<typename _Tp, typename _Ref, typename _Ptr >`
`_Deque_iterator< _Tp, _Ref, _Ptr >::difference_type operator- (const _Deque_iterator< _Tp, _Ref, _Ptr > &↵`
`__x, const _Deque_iterator< _Tp, _Ref, _Ptr > &__y) noexcept`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR >`
`_Deque_iterator< _Tp, _RefL, _PtrL >::difference_type operator- (const _Deque_iterator< _Tp, _RefL, _PtrL >`
`&__x, const _Deque_iterator< _Tp, _RefR, _PtrR > &__y) noexcept`
- `template<typename _IteratorL, typename _IteratorR >`
`auto operator- (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y) ->`
`decltype(__y.base()-__x.base())`
- `template<class _Dom >`
`_Expr< _BinClos< __minus, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __↵`
`minus, typename _Dom::value_type >::result_type > operator- (const valarray< typename _Dom::value_type`
`> &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`
`_Expr< _BinClos< __minus, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< __↵`
`minus, typename _Dom::value_type >::result_type > operator- (const _Expr< _Dom, typename _Dom::value↵`
`_type > &__e, const valarray< typename _Dom::value_type > &__v)`

- `template<class _Dom >`
`_Expr< _BinClos< __minus, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< __←`
`minus, typename _Dom::value_type >::result_type > operator- (const _Expr< _Dom, typename _Dom::value←`
`_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1 , class _Dom2 >`
`_Expr< _BinClos< __minus, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __minus, typename _Dom1←`
`::value_type >::result_type > operator- (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const`
`_Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`
`_Expr< _BinClos< __minus, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __←`
`minus, typename _Dom::value_type >::result_type > operator- (const typename _Dom::value_type &__t, const`
`_Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<typename _IteratorL , typename _IteratorR >`
`auto operator- (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y) ->`
`decltype(__x.base()-__y.base())`
- `template<typename _Iterator >`
`auto operator- (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y) -> decltype(←`
`__x.base()-__y.base())`
- `template<class _Dom >`
`_Expr< _BinClos< __divides, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< ←`
`__divides, typename _Dom::value_type >::result_type > operator/ (const _Expr< _Dom, typename _Dom←`
`::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`
`_Expr< _BinClos< __divides, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __←`
`divides, typename _Dom::value_type >::result_type > operator/ (const valarray< typename _Dom::value_type`
`> &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom1 , class _Dom2 >`
`_Expr< _BinClos< __divides, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __divides, typename ←`
`_Dom1::value_type >::result_type > operator/ (const _Expr< _Dom1, typename _Dom1::value_type > &__v,`
`const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`
`_Expr< _BinClos< __divides, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< ←`
`__divides, typename _Dom::value_type >::result_type > operator/ (const _Expr< _Dom, typename _Dom←`
`::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __divides, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __←`
`divides, typename _Dom::value_type >::result_type > operator/ (const typename _Dom::value_type &__t, const`
`_Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _T1 , class _T2 >`
`constexpr bool operator< (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<typename _Tp , typename _Seq >`
`bool operator< (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<typename _Tp , typename _Ref , typename _Ptr >`
`bool operator< (const Deque_iterator< _Tp, _Ref, _Ptr > &__x, const Deque_iterator< _Tp, _Ref, _Ptr >`
`&__y) noexcept`
- `template<typename _Tp , typename _RefL , typename _PtrL , typename _RefR , typename _PtrR >`
`bool operator< (const Deque_iterator< _Tp, _RefL, _PtrL > &__x, const Deque_iterator< _Tp, _RefR, _PtrR`
`> &__y) noexcept`
- `template<typename _Tp , typename _Seq >`
`bool operator< (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)`
- `template<typename _Iterator >`
`bool operator< (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`

- `template<typename _IteratorL, typename _IteratorR >`
`bool operator< (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator< (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept`
- `template<typename _Tp >`
`bool operator< (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp >`
`bool operator< (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<class _Dom >`
`_Expr< _BinClos< __less, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< __less,`
`typename _Dom::value_type >::result_type > operator< (const _Expr< _Dom, typename _Dom::value_type >`
`&__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`
`_Expr< _BinClos< __less, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< __less,`
`typename _Dom::value_type >::result_type > operator< (const _Expr< _Dom, typename _Dom::value_type >`
`&__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __less, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __less,`
`typename _Dom::value_type >::result_type > operator< (const valarray< typename _Dom::value_type >`
`&__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __less, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __less, typename _Dom1↵`
`::value_type >::result_type > operator< (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const`
`_Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`
`_Expr< _BinClos< __less, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __less,`
`typename _Dom::value_type >::result_type > operator< (const typename _Dom::value_type &__t, const`
`_Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`
`bool operator< (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp >`
`bool operator< (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)`
- `template<typename _Tp, typename _Dp >`
`bool operator< (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)`
- `template<typename _Key, typename _Compare, typename _Alloc >`
`bool operator< (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc >`
`&__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`
`bool operator< (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`
`bool operator< (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare,`
`_Alloc > &__y)`
- `template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, typename _Alloc >`
`bool operator< (const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__x, const _Rb_tree< _Key,`
`_Val, _KeyOfValue, _Compare, _Alloc > &__y)`
- `template<typename _Bilter >`
`bool operator< (const sub_match< _Bilter > &__lhs, const sub_match< _Bilter > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`
`bool operator< (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc >`
`&__y)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool operator< (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi↵`
`_iter > &__rhs)`

- `template<typename _IteratorL, typename _IteratorR >`
`bool operator< (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`
- `template<typename _Iterator >`
`bool operator< (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)`
- `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc >`
`bool operator< (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__rhs)`
- `template<typename _Tp1, typename _Tp2, _Lock_policy _Lp>`
`bool operator< (const __shared_ptr< _Tp1, _Lp > &__a, const __shared_ptr< _Tp2, _Lp > &__b) noexcept`
- `template<typename _Bi_iter >`
`bool operator< (typename iterator_traits< _Bi_iter >::value_type const * __lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Tp, _Lock_policy _Lp>`
`bool operator< (const __shared_ptr< _Tp, _Lp > &__a, nullptr_t) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>`
`bool operator< (nullptr_t, const __shared_ptr< _Tp, _Lp > &__a) noexcept`
- `template<typename _Bi_iter >`
`bool operator< (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const * __rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator< (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Bi_iter >`
`bool operator< (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`
`bool operator< (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator< (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool operator< (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool operator< (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc > &__y)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool operator< (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool operator< (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT * __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool operator< (const _CharT * __lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _Ch, typename _Tr, typename _Tp, _Lock_policy _Lp>`
`std::basic_ostream< _Ch, _Tr > & operator<< (std::basic_ostream< _Ch, _Tr > &__os, const __shared_ptr< _Tp, _Lp > &__p)`
- `template<class _Dom >`
`__Expr< __BinClos< __shift_left, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __shift_left, typename _Dom::value_type >::result_type > operator<< (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom1, class _Dom2 >`
`__Expr< __BinClos< __shift_left, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __shift_left, typename _Dom1::value_type >::result_type > operator<< (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`

- `template<class _Dom >`
`_Expr< _BinClos< __shift_left, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun<`
`__shift_left, typename _Dom::value_type >::result_type > operator<< (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`
`_Expr< _BinClos< __shift_left, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun<`
`__shift_left, typename _Dom::value_type >::result_type > operator<< (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __shift_left, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< __`
`__shift_left, typename _Dom::value_type >::result_type > operator<< (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<typename _RandomNumberEngine, size_t __w, typename _UIntType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`
`std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > &__x)`
- `template<typename _Ch_type, typename _Ch_traits, typename _Bi_iter >`
`basic_ostream< _Ch_type, _Ch_traits > & operator<< (basic_ostream< _Ch_type, _Ch_traits > &__os, const`
`sub_match< _Bi_iter > &__m)`
- `template<typename _IntType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &, const std::`
`::uniform_int_distribution< _IntType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &, const std::`
`::uniform_real_distribution< _RealType > &)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, const __gnu_cxx::`
`versa_string< _CharT, _Traits, _Alloc, _Base > &__str)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, const basic_`
`string< _CharT, _Traits, _Alloc > &__str)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`
`std::cauchy_distribution< _RealType > &__x)`
- `template<typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`
`std::bernoulli_distribution &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`
`std::geometric_distribution< _IntType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`
`std::exponential_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`
`std::weibull_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`
`std::extreme_value_distribution< _RealType > &__x)`
- `template<class _T1, class _T2 >`
`constexpr bool operator<= (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<typename _Tp, typename _Seq >`
`bool operator<= (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`

- `template<typename _Tp, typename _Ref, typename _Ptr >`
`bool operator<= (const _Deque_iterator< _Tp, _Ref, _Ptr > &__x, const _Deque_iterator< _Tp, _Ref, _Ptr >`
`&__y) noexcept`
- `template<typename _Tp, typename _Seq >`
`bool operator<= (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR >`
`bool operator<= (const _Deque_iterator< _Tp, _RefL, _PtrL > &__x, const _Deque_iterator< _Tp, _RefR, _PtrR >`
`&__y) noexcept`
- `template<typename _Iterator >`
`bool operator<= (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool operator<= (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator<= (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept`
- `template<typename _Tp >`
`bool operator<= (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp >`
`bool operator<= (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __less_equal, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __less_equal, typename`
`_Dom1::value_type >::result_type > operator<= (const _Expr< _Dom1, typename _Dom1::value_type > &__v,`
`const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`
`_Expr< _BinClos< __less_equal, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun<`
`__less_equal, typename _Dom::value_type >::result_type > operator<= (const _Expr< _Dom, typename _Dom::value_type >`
`&__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`
`_Expr< _BinClos< __less_equal, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun<`
`__less_equal, typename _Dom::value_type >::result_type > operator<= (const typename _Dom::value_type`
`&__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __less_equal, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun<`
`__less_equal, typename _Dom::value_type >::result_type > operator<= (const valarray< typename _Dom::value_type >`
`&__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`
`_Expr< _BinClos< __less_equal, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun<`
`__less_equal, typename _Dom::value_type >::result_type > operator<= (const _Expr< _Dom, typename _Dom::value_type >`
`&__e, const valarray< typename _Dom::value_type > &__v)`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`
`bool operator<= (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp >`
`bool operator<= (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)`
- `template<typename _Tp, typename _Dp >`
`bool operator<= (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)`
- `template<typename _Key, typename _Compare, typename _Alloc >`
`bool operator<= (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc >`
`&__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`
`bool operator<= (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`
`bool operator<= (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare,`
`_Alloc > &__y)`

- `template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, typename _Alloc >`
`bool operator<= (const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__x, const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__y)`
- `template<typename _Biliter >`
`bool operator<= (const sub_match< _Biliter > &__lhs, const sub_match< _Biliter > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`
`bool operator<= (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool operator<= (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`
- `template<typename _Iterator >`
`bool operator<= (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool operator<= (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc >`
`bool operator<= (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__rhs)`
- `template<typename _Tp1, typename _Tp2, _Lock_policy _Lp>`
`bool operator<= (const __shared_ptr< _Tp1, _Lp > &__a, const __shared_ptr< _Tp2, _Lp > &__b) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>`
`bool operator<= (const __shared_ptr< _Tp, _Lp > &__a, nullptr_t) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>`
`bool operator<= (nullptr_t, const __shared_ptr< _Tp, _Lp > &__a) noexcept`
- `template<typename _Bi_iter >`
`bool operator<= (typename iterator_traits< _Bi_iter >::value_type const * __lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`
`bool operator<= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const * __rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator<= (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Bi_iter >`
`bool operator<= (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`
`bool operator<= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator<= (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool operator<= (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool operator<= (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc > &__y)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool operator<= (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool operator<= (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT * __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool operator<= (const _CharT * __lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _T1, typename _T2 >`
`bool operator== (const allocator< _T1 > &, const allocator< _T2 > &)`

- `template<typename _Tp, typename _CharT, typename _Traits, typename _Dist >`
`bool operator== (const istream_iterator< _Tp, _CharT, _Traits, _Dist > &__x, const istream_iterator< _Tp, _CharT, _Traits, _Dist > &__y)`
- `template<typename _Tp >`
`bool operator== (const allocator< _Tp > &, const allocator< _Tp > &)`
- `template<typename _CharT, typename _Traits >`
`bool operator== (const istreambuf_iterator< _CharT, _Traits > &__a, const istreambuf_iterator< _CharT, _Traits > &__b)`
- `template<class _T1, class _T2 >`
`constexpr bool operator== (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<typename _StateT >`
`bool operator== (const fpos< _StateT > &__lhs, const fpos< _StateT > &__rhs)`
- `template<typename _Tp, typename _Seq >`
`bool operator== (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Ref, typename _Ptr >`
`bool operator== (const _Deque_iterator< _Tp, _Ref, _Ptr > &__x, const _Deque_iterator< _Tp, _Ref, _Ptr > &__y) noexcept`
- `template<typename _Tp >`
`bool operator== (const _Fwd_list_iterator< _Tp > &__x, const _Fwd_list_const_iterator< _Tp > &__y) noexcept`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR >`
`bool operator== (const _Deque_iterator< _Tp, _RefL, _PtrL > &__x, const _Deque_iterator< _Tp, _RefR, _PtrR > &__y) noexcept`
- `template<typename _Tp, typename _Seq >`
`bool operator== (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)`
- `template<typename _Val >`
`bool operator== (const _List_iterator< _Val > &__x, const _List_const_iterator< _Val > &__y) noexcept`
- `template<typename _Iterator >`
`bool operator== (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`
- `template<typename _Val >`
`bool operator== (const _Rb_tree_iterator< _Val > &__x, const _Rb_tree_const_iterator< _Val > &__y) noexcept`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator== (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept`
- `template<typename _Tp >`
`bool operator== (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _IteratorL, typename _IteratorR >`
`bool operator== (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`
- `template<typename _Tp >`
`bool operator== (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<class _Dom >`
`_Expr< _BinClos< __equal_to, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< __equal_to, typename _Dom::value_type >::result_type > operator== (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`
`_Expr< _BinClos< __equal_to, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< __equal_to, typename _Dom::value_type >::result_type > operator== (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __equal_to, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __equal_to, typename _Dom::value_type >::result_type > operator== (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`

- `template<class _Dom >`
`_Expr< _BinClos< __equal_to, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __equal_to, typename _Dom::value_type >::result_type > operator== (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __equal_to, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __equal_to, typename _Dom1::value_type >::result_type > operator== (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`
`bool operator== (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp >`
`bool operator== (const unique_ptr< _Tp, _Dp > &__x, nullptr_t) noexcept`
- `template<typename _Tp, typename _Dp >`
`bool operator== (nullptr_t, const unique_ptr< _Tp, _Dp > &__x) noexcept`
- `template<typename _Key, typename _Compare, typename _Alloc >`
`bool operator== (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`
`bool operator== (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`
`bool operator== (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, typename _Alloc >`
`bool operator== (const Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__x, const Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__y)`
- `template<typename _Bilter >`
`bool operator== (const sub_match< _Bilter > &__lhs, const sub_match< _Bilter > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`
`bool operator== (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool operator== (const sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< __Bi_iter > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool operator== (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`
- `template<typename _Iterator >`
`bool operator== (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool operator== (const sub_match< _Bi_iter > &__lhs, const sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__rhs)`
- `template<typename _Tp1, typename _Tp2, _Lock_policy _Lp>`
`bool operator== (const shared_ptr< _Tp1, _Lp > &__a, const shared_ptr< _Tp2, _Lp > &__b) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>`
`bool operator== (const shared_ptr< _Tp, _Lp > &__a, nullptr_t) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>`
`bool operator== (nullptr_t, const shared_ptr< _Tp, _Lp > &__a) noexcept`
- `template<typename _Bi_iter >`
`bool operator== (typename iterator_traits< _Bi_iter >::value_type const *__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`
`bool operator== (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const *__rhs)`

- `template<class _Value, class _Hash, class _Pred, class _Alloc >`
`bool operator== (const unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Bi_iter >`
`bool operator== (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc >`
`bool operator== (const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool operator== (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Bi_iter >`
`bool operator== (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc >`
`bool operator== (const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc >`
`bool operator== (const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool operator== (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool operator== (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`
- `template<typename _Bi_iter, typename _Alloc >`
`bool operator== (const match_results< _Bi_iter, _Alloc > &__m1, const match_results< _Bi_iter, _Alloc > &__m2)`
- `template<typename _Tp, typename _Alloc >`
`bool operator== (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc > &__y)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool operator== (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT >`
`__gnu_cxx::__enable_if< __is_char< _CharT >::value, bool >::__type operator== (const basic_string< _CharT > &__lhs, const basic_string< _CharT > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool operator== (const _CharT * __lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool operator== (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT * __rhs)`
- `template<class _T1, class _T2 >`
`constexpr bool operator> (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<typename _Tp, typename _Seq >`
`bool operator> (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Ref, typename _Ptr >`
`bool operator> (const _Deque_iterator< _Tp, _Ref, _Ptr > &__x, const _Deque_iterator< _Tp, _Ref, _Ptr > &__y) noexcept`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR >`
`bool operator> (const _Deque_iterator< _Tp, _RefL, _PtrL > &__x, const _Deque_iterator< _Tp, _RefR, _PtrR > &__y) noexcept`
- `template<typename _Tp, typename _Seq >`
`bool operator> (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)`
- `template<typename _Iterator >`
`bool operator> (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`

- `template<typename _IteratorL, typename _IteratorR >`
`bool operator> (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator> (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept`
- `template<typename _Tp >`
`bool operator> (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp >`
`bool operator> (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<class _Dom >`
`_Expr< _BinClos< __greater, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __greater, typename _Dom::value_type >::result_type > operator> (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __greater, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __greater, typename _Dom::value_type >::result_type > operator> (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __greater, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __greater, typename _Dom1::value_type >::result_type > operator> (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`
`_Expr< _BinClos< __greater, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< __greater, typename _Dom::value_type >::result_type > operator> (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __greater, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< __greater, typename _Dom::value_type >::result_type > operator> (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`
`bool operator> (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp >`
`bool operator> (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)`
- `template<typename _Tp, typename _Dp >`
`bool operator> (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)`
- `template<typename _Key, typename _Compare, typename _Alloc >`
`bool operator> (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`
`bool operator> (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`
`bool operator> (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, typename _Alloc >`
`bool operator> (const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__x, const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__y)`
- `template<typename _Bilter >`
`bool operator> (const sub_match< _Bilter > &__lhs, const sub_match< _Bilter > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`
`bool operator> (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool operator> (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter > &__rhs)`

- `template<typename _IteratorL, typename _IteratorR >`
`bool operator> (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`
- `template<typename _Iterator >`
`bool operator> (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)`
- `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc >`
`bool operator> (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__rhs)`
- `template<typename _Bi_iter >`
`bool operator> (typename iterator_traits< _Bi_iter >::value_type const *__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Tp1, typename _Tp2, _Lock_policy _Lp>`
`bool operator> (const __shared_ptr< _Tp1, _Lp > &__a, const __shared_ptr< _Tp2, _Lp > &__b) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>`
`bool operator> (const __shared_ptr< _Tp, _Lp > &__a, nullptr_t) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>`
`bool operator> (nullptr_t, const __shared_ptr< _Tp, _Lp > &__a) noexcept`
- `template<typename _Bi_iter >`
`bool operator> (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const *__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator> (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Bi_iter >`
`bool operator> (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`
`bool operator> (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator> (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool operator> (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool operator> (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc > &__y)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool operator> (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool operator> (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool operator> (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<class _T1, class _T2 >`
`constexpr bool operator>= (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<typename _Tp, typename _Seq >`
`bool operator>= (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq >`
`bool operator>= (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Ref, typename _Ptr >`
`bool operator>= (const _Deque_iterator< _Tp, _Ref, _Ptr > &__x, const _Deque_iterator< _Tp, _Ref, _Ptr > &__y) noexcept`
- `template<typename _Iterator >`
`bool operator>= (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`

- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR >`
`bool operator>= (const Deque_iterator< _Tp, _RefL, _PtrL > &__x, const Deque_iterator< _Tp, _RefR, _PtrR > &__y) noexcept`
- `template<typename _IteratorL, typename _IteratorR >`
`bool operator>= (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator>= (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __greater_equal, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __greater_equal, typename _Dom1::value_type >::result_type > operator>= (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`
`_Expr< _BinClos< __greater_equal, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __greater_equal, typename _Dom::value_type >::result_type > operator>= (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __greater_equal, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __greater_equal, typename _Dom::value_type >::result_type > operator>= (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`
`_Expr< _BinClos< __greater_equal, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< __greater_equal, typename _Dom::value_type >::result_type > operator>= (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __greater_equal, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< __greater_equal, typename _Dom::value_type >::result_type > operator>= (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<typename _Tp >`
`bool operator>= (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp >`
`bool operator>= (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`
`bool operator>= (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp >`
`bool operator>= (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)`
- `template<typename _Tp, typename _Dp >`
`bool operator>= (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)`
- `template<typename _Key, typename _Compare, typename _Alloc >`
`bool operator>= (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`
`bool operator>= (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`
`bool operator>= (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, typename _Alloc >`
`bool operator>= (const Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__x, const Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__y)`
- `template<typename _Bilter >`
`bool operator>= (const sub_match< _Bilter > &__lhs, const sub_match< _Bilter > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`
`bool operator>= (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)`

- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool operator>= (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool operator>= (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`
- `template<typename _Iterator >`
`bool operator>= (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)`
- `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc >`
`bool operator>= (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__rhs)`
- `template<typename _Bi_iter >`
`bool operator>= (typename iterator_traits< _Bi_iter >::value_type const *__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Tp1, typename _Tp2, _Lock_policy _Lp>`
`bool operator>= (const __shared_ptr< _Tp1, _Lp > &__a, const __shared_ptr< _Tp2, _Lp > &__b) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>`
`bool operator>= (const __shared_ptr< _Tp, _Lp > &__a, nullptr_t) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>`
`bool operator>= (nullptr_t, const __shared_ptr< _Tp, _Lp > &__a) noexcept`
- `template<typename _Bi_iter >`
`bool operator>= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const *__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator>= (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Bi_iter >`
`bool operator>= (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`
`bool operator>= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`bool operator>= (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool operator>= (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool operator>= (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc > &__y)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool operator>= (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool operator>= (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool operator>= (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<class _Dom >`
`_Expr< _BinClos< __shift_right, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< __shift_right, typename _Dom::value_type >::result_type > operator>> (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`
`_Expr< _BinClos< __shift_right, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __shift_right, typename _Dom::value_type >::result_type > operator>> (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`

- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __shift_right, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __shift_right, typename`
`_Dom1::value_type >::result_type > operator>> (const _Expr< _Dom1, typename _Dom1::value_type > &__v,`
`const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`
`_Expr< _BinClos< __shift_right, ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun<`
`__shift_right, typename _Dom::value_type >::result_type > operator>> (const valarray< typename _Dom<`
`::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`
`_Expr< _BinClos< __shift_right, _Expr, ValArray, _Dom, typename _Dom::value_type >, typename __fun<`
`__shift_right, typename _Dom::value_type >::result_type > operator>> (const _Expr< _Dom, typename <`
`_Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<typename _IntType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &, std::uniform<`
`int_distribution< _IntType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &, std::uniform<`
`real_distribution< _RealType > &)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _Traits > &__is, __gnu_cxx::__`
`versa_string< _CharT, _Traits, _Alloc, _Base > &__str)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _Traits > &__is, basic_string< <`
`_CharT, _Traits, _Alloc > &__str)`
- `template<>`
`basic_istream< char > & operator>> (basic_istream< char > &__is, basic_string< char > &__str)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::`
`cauchy_distribution< _RealType > &__x)`
- `template<typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::`
`bernoulli_distribution &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::`
`geometric_distribution< _IntType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::`
`exponential_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::`
`weibull_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::`
`extreme_value_distribution< _RealType > &__x)`
- `constexpr _ios_Fmtflags operator^ (_ios_Fmtflags __a, _ios_Fmtflags __b)`
- `constexpr _ios_Openmode operator^ (_ios_Openmode __a, _ios_Openmode __b)`
- `constexpr _ios_ostate operator^ (_ios_ostate __a, _ios_ostate __b)`
- `template<class _Dom >`
`_Expr< _BinClos< __bitwise_xor, _Expr, ValArray, _Dom, typename _Dom::value_type >, typename __fun<`
`__bitwise_xor, typename _Dom::value_type >::result_type > operator^ (const _Expr< _Dom, typename <`
`_Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`

- `template<class _Dom >`
`_Expr< _BinClos< __bitwise_xor, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun<`
`__bitwise_xor, typename _Dom::value_type >::result_type > operator^ (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`
`_Expr< _BinClos< __bitwise_xor, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun<`
`__bitwise_xor, typename _Dom::value_type >::result_type > operator^ (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __bitwise_xor, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __bitwise_xor, typename`
`_Dom1::value_type >::result_type > operator^ (const _Expr< _Dom1, typename _Dom1::value_type > &__v,`
`const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`
`_Expr< _BinClos< __bitwise_xor, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun<`
`__bitwise_xor, typename _Dom::value_type >::result_type > operator^ (const _Expr< _Dom, typename`
`_Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `const _los_Fmtflags & operator^= (_los_Fmtflags &__a, _los_Fmtflags __b)`
- `const _los_Openmode & operator^= (_los_Openmode &__a, _los_Openmode __b)`
- `const _los_losestate & operator^= (_los_losestate &__a, _los_losestate __b)`
- `constexpr memory_order operator| (memory_order __m, __memory_order_modifier __mod)`
- `constexpr _los_Fmtflags operator| (_los_Fmtflags __a, _los_Fmtflags __b)`
- `constexpr _los_Openmode operator| (_los_Openmode __a, _los_Openmode __b)`
- `constexpr _los_losestate operator| (_los_losestate __a, _los_losestate __b)`
- `template<class _Dom >`
`_Expr< _BinClos< __bitwise_or, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun<`
`__bitwise_or, typename _Dom::value_type >::result_type > operator| (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __bitwise_or, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun<`
`__bitwise_or, typename _Dom::value_type >::result_type > operator| (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`
`_Expr< _BinClos< __bitwise_or, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun<`
`__bitwise_or, typename _Dom::value_type >::result_type > operator| (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __bitwise_or, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __bitwise_or, typename`
`_Dom1::value_type >::result_type > operator| (const _Expr< _Dom1, typename _Dom1::value_type > &__v,`
`const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`
`_Expr< _BinClos< __bitwise_or, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun<`
`__bitwise_or, typename _Dom::value_type >::result_type > operator| (const typename _Dom::value_type &__t,`
`const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `const _los_Fmtflags & operator|= (_los_Fmtflags &__a, _los_Fmtflags __b)`
- `const _los_Openmode & operator|= (_los_Openmode &__a, _los_Openmode __b)`
- `const _los_losestate & operator|= (_los_losestate &__a, _los_losestate __b)`
- `template<class _Dom >`
`_Expr< _BinClos< __logical_or, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun<`
`__logical_or, typename _Dom::value_type >::result_type > operator|| (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`
`_Expr< _BinClos< __logical_or, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun<`
`__logical_or, typename _Dom::value_type >::result_type > operator|| (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`

- `template<class _Dom >`
`_Expr< _BinClos< __logical_or, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __logical_or, typename _Dom::value_type >::result_type > operator|| (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`
`_Expr< _BinClos< __logical_or, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< __logical_or, typename _Dom::value_type >::result_type > operator|| (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __logical_or, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __logical_or, typename _Dom1::value_type >::result_type > operator|| (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `constexpr _ios_Fmtflags operator~ (_ios_Fmtflags __a)`
- `constexpr _ios_Openmode operator~ (_ios_Openmode __a)`
- `constexpr _ios_ostate operator~ (_ios_ostate __a)`
- `template<typename _RAIter >`
`void partial_sort (_RAIter, _RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`
`void partial_sort (_RAIter, _RAIter, _RAIter, _Compare)`
- `template<typename _RandomAccessIterator >`
`void partial_sort (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void partial_sort (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _Iter, typename _RAIter >`
`_RAIter partial_sort_copy (_Iter, _Iter, _RAIter, _RAIter)`
- `template<typename _Iter, typename _RAIter, typename _Compare >`
`_RAIter partial_sort_copy (_Iter, _Iter, _RAIter, _RAIter, _Compare)`
- `template<typename _InputIterator, typename _RandomAccessIterator >`
`_RandomAccessIterator partial_sort_copy (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __result_first, _RandomAccessIterator __result_last)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _Compare >`
`_RandomAccessIterator partial_sort_copy (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __result_first, _RandomAccessIterator __result_last, _Compare __comp)`
- `template<typename _InputIterator, typename _OutputIterator >`
`_OutputIterator partial_sum (_InputIterator __first, _InputIterator __last, _OutputIterator __result)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation >`
`_OutputIterator partial_sum (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryOperation __binary_op)`
- `template<typename _BIter, typename _Predicate >`
`_BIter partition (_BIter, _BIter, _Predicate)`
- `template<typename _ForwardIterator, typename _Predicate >`
`_ForwardIterator partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _Iter, typename _OIter1, typename _OIter2, typename _Predicate >`
`pair< _OIter1, _OIter2 > partition_copy (_Iter, _Iter, _OIter1, _OIter2, _Predicate)`
- `template<typename _InputIterator, typename _OutputIterator1, typename _OutputIterator2, typename _Predicate >`
`pair< _OutputIterator1, _OutputIterator2 > partition_copy (_InputIterator __first, _InputIterator __last, _OutputIterator1 __out_true, _OutputIterator2 __out_false, _Predicate __pred)`
- `template<typename _Filter, typename _Predicate >`
`_Filter partition_point (_Filter, _Filter, _Predicate)`
- `template<typename _ForwardIterator, typename _Predicate >`
`_ForwardIterator partition_point (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`

- `template<typename _RandomAccessIterator >`
`void pop_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void pop_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RAlter >`
`void pop_heap (_RAlter, _RAlter)`
- `template<typename _RAlter, typename _Compare >`
`void pop_heap (_RAlter, _RAlter, _Compare)`
- `template<class _Dom >`
`_Expr< _BinClos< _Pow, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename _Dom::value_type > pow (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`
`_Expr< _BinClos< _Pow, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename _Dom::value_type > pow (const _Expr< _Dom, typename _Dom::value_type > &__e, const typename _Dom::value_type &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< _Pow, _ValArray, _Constant, _Tp, _Tp >, _Tp > pow (const valarray< _Tp > &__v, const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< _Pow, _Constant, _ValArray, _Tp, _Tp >, _Tp > pow (const _Tp &__t, const valarray< _Tp > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< _Pow, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename _Dom::value_type > pow (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< _Pow, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename _Dom::value_type > pow (const valarray< typename _Dom::valarray > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< _Pow, _Expr, _Expr, _Dom1, _Dom2 >, typename _Dom1::value_type > pow (const _Expr< _Dom1, typename _Dom1::value_type > &__e1, const _Expr< _Dom2, typename _Dom2::value_type > &__e2)`
- `template<typename _Tp >`
`_Expr< _BinClos< _Pow, _ValArray, _ValArray, _Tp, _Tp >, _Tp > pow (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _BidirectionalIterator >`
`_BidirectionalIterator prev (_BidirectionalIterator __x, typename iterator_traits< _BidirectionalIterator >::difference_type __n=1)`
- `template<typename _Blter >`
`bool prev_permutation (_Blter, _Blter)`
- `template<typename _Blter, typename _Compare >`
`bool prev_permutation (_Blter, _Blter, _Compare)`
- `template<typename _BidirectionalIterator >`
`bool prev_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _Compare >`
`bool prev_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _Arg, typename _Result >`
`pointer_to_unary_function< _Arg, _Result > ptr_fun (_Result(*__x)(_Arg))`
- `template<typename _Arg1, typename _Arg2, typename _Result >`
`pointer_to_binary_function< _Arg1, _Arg2, _Result > ptr_fun (_Result(*__x)(_Arg1, _Arg2))`
- `template<typename _RandomAccessIterator >`
`void push_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`

- `template<typename _RandomAccessIterator, typename _Compare >`
`void push_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RAIter >`
`void push_heap (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`
`void push_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _RAIter >`
`void random_shuffle (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Generator >`
`void random_shuffle (_RAIter, _RAIter, _Generator &&)`
- `template<typename _RandomAccessIterator >`
`void random_shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _RandomNumberGenerator >`
`void random_shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last, _RandomNumberGenerator && __rand)`
- `template<typename _Filter, typename _Tp >`
`_Filter remove (_Filter, _Filter, const _Tp &)`
- `template<typename _ForwardIterator, typename _Tp >`
`_ForwardIterator remove (_ForwardIterator __first, _ForwardIterator __last, const _Tp & __value)`
- `template<typename _Iter, typename _OIter, typename _Tp >`
`_OIter remove_copy (_Iter, _Iter, _OIter, const _Tp &)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Tp >`
`_OutputIterator remove_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, const _Tp & __value)`
- `template<typename _Iter, typename _OIter, typename _Predicate >`
`_OIter remove_copy_if (_Iter, _Iter, _OIter, _Predicate)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate >`
`_OutputIterator remove_copy_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred)`
- `template<typename _Filter, typename _Predicate >`
`_Filter remove_if (_Filter, _Filter, _Predicate)`
- `template<typename _ForwardIterator, typename _Predicate >`
`_ForwardIterator remove_if (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _Filter, typename _Tp >`
`void replace (_Filter, _Filter, const _Tp &, const _Tp &)`
- `template<typename _ForwardIterator, typename _Tp >`
`void replace (_ForwardIterator __first, _ForwardIterator __last, const _Tp & __old_value, const _Tp & __new_value)`
- `template<typename _Iter, typename _OIter, typename _Tp >`
`_OIter replace_copy (_Iter, _Iter, _OIter, const _Tp &, const _Tp &)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Tp >`
`_OutputIterator replace_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, const _Tp & __old_value, const _Tp & __new_value)`
- `template<typename _Iter, typename _OIter, typename _Predicate, typename _Tp >`
`_OIter replace_copy_if (_Iter, _Iter, _OIter, _Predicate, const _Tp &)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate, typename _Tp >`
`_OutputIterator replace_copy_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred, const _Tp & __new_value)`
- `template<typename _Filter, typename _Predicate, typename _Tp >`
`void replace_if (_Filter, _Filter, _Predicate, const _Tp &)`
- `template<typename _ForwardIterator, typename _Predicate, typename _Tp >`
`void replace_if (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, const _Tp & __new_value)`
- `void rethrow_exception (exception_ptr) __attribute__((__noreturn__))`

- `template<typename _Ex >`
`void rethrow_if_nested (const _Ex &__ex)`
- `void rethrow_if_nested (const nested_exception &__ex)`
- `template<typename _Tp >`
`void return_temporary_buffer (_Tp *__p)`
- `template<typename _Blter >`
`void reverse (_Blter, _Blter)`
- `template<typename _BidirectionalIterator >`
`void reverse (_BidirectionalIterator __first, _BidirectionalIterator __last)`
- `template<typename _Blter, typename _Olter >`
`_Olter reverse_copy (_Blter, _Blter, _Olter)`
- `template<typename _BidirectionalIterator, typename _OutputIterator >`
`_OutputIterator reverse_copy (_BidirectionalIterator __first, _BidirectionalIterator __last, _OutputIterator __result)`
- `ios_base & right (ios_base &__base)`
- `template<typename _Filter >`
`void rotate (_Filter, _Filter, _Filter)`
- `template<typename _ForwardIterator >`
`void rotate (_ForwardIterator __first, _ForwardIterator __middle, _ForwardIterator __last)`
- `template<typename _Filter, typename _Olter >`
`_Olter rotate_copy (_Filter, _Filter, _Filter, _Olter)`
- `template<typename _ForwardIterator, typename _OutputIterator >`
`_OutputIterator rotate_copy (_ForwardIterator __first, _ForwardIterator __middle, _ForwardIterator __last, _↵
_OutputIterator __result)`
- `ios_base & scientific (ios_base &__base)`
- `template<typename _Filter1, typename _Filter2 >`
`_Filter1 search (_Filter1, _Filter1, _Filter2, _Filter2)`
- `template<typename _Filter1, typename _Filter2, typename _BinaryPredicate >`
`_Filter1 search (_Filter1, _Filter1, _Filter2, _Filter2, _BinaryPredicate)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`
`_ForwardIterator1 search (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _↵
_ForwardIterator2 __last2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate >`
`_ForwardIterator1 search (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _↵
_ForwardIterator2 __last2, _BinaryPredicate __predicate)`
- `template<typename _Filter, typename _Size, typename _Tp >`
`_Filter search_n (_Filter, _Filter, _Size, const _Tp &)`
- `template<typename _Filter, typename _Size, typename _Tp, typename _BinaryPredicate >`
`_Filter search_n (_Filter, _Filter, _Size, const _Tp &, _BinaryPredicate)`
- `template<typename _ForwardIterator, typename _Integer, typename _Tp >`
`_ForwardIterator search_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp &__val)`
- `template<typename _ForwardIterator, typename _Integer, typename _Tp, typename _BinaryPredicate >`
`_ForwardIterator search_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp &__val,
_BinaryPredicate __binary_pred)`
- `template<typename _Iter1, typename _Iter2, typename _Olter >`
`_Olter set_difference (_Iter1, _Iter1, _Iter2, _Iter2, _Olter)`
- `template<typename _Iter1, typename _Iter2, typename _Olter, typename _Compare >`
`_Olter set_difference (_Iter1, _Iter1, _Iter2, _Iter2, _Olter, _Compare)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`
`_OutputIterator set_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Input↵
_Iterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`
`_OutputIterator set_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Input↵
_Iterator2 __last2, _OutputIterator __result, _Compare __comp)`

- `template<typename _Iter1, typename _Iter2, typename _OIter >`
`_OIter set_intersection (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`
`_OIter set_intersection (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`
`_OutputIterator set_intersection (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`
`_OutputIterator set_intersection (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`
`_OIter set_symmetric_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`
`_OIter set_symmetric_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`
`_OutputIterator set_symmetric_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`
`_OutputIterator set_symmetric_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`
`_OIter set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`
`_OIter set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`
`_OutputIterator set_union (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`
`_OutputIterator set_union (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `ios_base & showbase (ios_base & __base)`
- `ios_base & showpoint (ios_base & __base)`
- `ios_base & showpos (ios_base & __base)`
- `template<typename _RAIter, typename _UGenerator >`
`void shuffle (_RAIter, _RAIter, _UGenerator &&)`
- `template<typename _RandomAccessIterator, typename _UniformRandomNumberGenerator >`
`void shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last, _UniformRandomNumberGenerator && __g)`
- `template<class _Dom >`
`_Expr< _UnClos< _Sin, _Expr, _Dom >, typename _Dom::value_type > sin (const _Expr< _Dom, typename _Dom::value_type > & __e)`
- `template<typename _Tp >`
`_Expr< _UnClos< _Sin, _ValArray, _Tp >, _Tp > sin (const valarray< _Tp > & __v)`
- `template<class _Dom >`
`_Expr< _UnClos< _Sinh, _Expr, _Dom >, typename _Dom::value_type > sinh (const _Expr< _Dom, typename _Dom::value_type > & __e)`
- `template<typename _Tp >`
`_Expr< _UnClos< _Sinh, _ValArray, _Tp >, _Tp > sinh (const valarray< _Tp > & __v)`
- `ios_base & skipws (ios_base & __base)`
- `template<typename _RAIter >`
`void sort (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`
`void sort (_RAIter, _RAIter, _Compare)`

- `template<typename _RandomAccessIterator >`
`void sort (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`void sort_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void sort_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RAIter >`
`void sort_heap (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`
`void sort_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _Tp >`
`_Expr< _UnClos< _Sqrt, _ValArray, _Tp >, _Tp > sqrt (const valarray< _Tp > &__v)`
- `template<class _Dom >`
`_Expr< _UnClos< _Sqrt, _Expr, _Dom >, typename _Dom::value_type > sqrt (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Blter, typename _Predicate >`
`_Blter stable_partition (_Blter, _Blter, _Predicate)`
- `template<typename _ForwardIterator, typename _Predicate >`
`_ForwardIterator stable_partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _RAIter >`
`void stable_sort (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`
`void stable_sort (_RAIter, _RAIter, _Compare)`
- `template<typename _RandomAccessIterator >`
`void stable_sort (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void stable_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _Tp, typename _Tp1 >`
`shared_ptr< _Tp > static_pointer_cast (const shared_ptr< _Tp1 > &__r) noexcept`
- `template<typename _Tp, typename _Tp1, _Lock_policy _Lp>`
`__shared_ptr< _Tp, _Lp > static_pointer_cast (const __shared_ptr< _Tp1, _Lp > &__r) noexcept`
- `void swap (_Bit_reference __x, _Bit_reference __y) noexcept`
- `void swap (_Bit_reference __x, bool &__y) noexcept`
- `void swap (bool &__x, _Bit_reference __y) noexcept`
- `template<class _T1, class _T2 >`
`void swap (pair< _T1, _T2 > &__x, pair< _T1, _T2 > &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename _Tp, typename _Seq >`
`void swap (stack< _Tp, _Seq > &__x, stack< _Tp, _Seq > &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename _Tp, typename _Seq >`
`void swap (queue< _Tp, _Seq > &__x, queue< _Tp, _Seq > &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename _Tp >`
`void swap (shared_ptr< _Tp > &__a, shared_ptr< _Tp > &__b) noexcept`
- `template<typename _Tp >`
`void swap (weak_ptr< _Tp > &__a, weak_ptr< _Tp > &__b) noexcept`
- `template<typename _Tp >`
`void swap (_Tp &__a, _Tp &__b) noexcept(__and< is_nothrow_move_constructible< _Tp >, is_nothrow_move_assignable< _Tp >::value)`
- `template<typename _Tp, typename _Sequence, typename _Compare >`
`void swap (priority_queue< _Tp, _Sequence, _Compare > &__x, priority_queue< _Tp, _Sequence, _Compare > &__y) noexcept(noexcept(__x.swap(__y)))`

- `template<typename _Tp, size_t _Nm>`
`void swap (_Tp(&__a)[_Nm], _Tp(&__b)[_Nm]) noexcept(noexcept(swap(*__a,*__b)))`
- `template<typename _Tp, typename _Dp >`
`void swap (unique_ptr< _Tp, _Dp > &__x, unique_ptr< _Tp, _Dp > &__y) noexcept`
- `template<typename _Key, typename _Compare, typename _Alloc >`
`void swap (multiset< _Key, _Compare, _Alloc > &__x, multiset< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Ch_type, typename _Rx_traits >`
`void swap (basic_regex< _Ch_type, _Rx_traits > &__lhs, basic_regex< _Ch_type, _Rx_traits > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Alloc >`
`void swap (set< _Key, _Compare, _Alloc > &__x, set< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`
`void swap (multimap< _Key, _Tp, _Compare, _Alloc > &__x, multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, typename _Alloc >`
`void swap (_Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__x, _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`
`void swap (map< _Key, _Tp, _Compare, _Alloc > &__x, map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Tp, _Lock_policy _Lp>`
`void swap (__shared_ptr< _Tp, _Lp > &__a, __shared_ptr< _Tp, _Lp > &__b) noexcept`
- `template<class _Value, class _Hash, class _Pred, class _Alloc >`
`void swap (unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc >`
`void swap (unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`
`void swap (forward_list< _Tp, _Alloc > &__lx, forward_list< _Tp, _Alloc > &__ly)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc >`
`void swap (unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc >`
`void swap (unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Tp, _Lock_policy _Lp>`
`void swap (__weak_ptr< _Tp, _Lp > &__a, __weak_ptr< _Tp, _Lp > &__b) noexcept`
- `template<typename _Tp, typename _Alloc >`
`void swap (vector< _Tp, _Alloc > &__x, vector< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`
`void swap (list< _Tp, _Alloc > &__x, list< _Tp, _Alloc > &__y)`
- `template<typename _Bi_iter, typename _Alloc >`
`void swap (match_results< _Bi_iter, _Alloc > &__lhs, match_results< _Bi_iter, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`
`void swap (deque< _Tp, _Alloc > &__x, deque< _Tp, _Alloc > &__y)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`void swap (basic_string< _CharT, _Traits, _Alloc > &__lhs, basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`
`_ForwardIterator2 swap_ranges (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2)`
- `template<typename _Filter1, typename _Filter2 >`
`_Filter2 swap_ranges (_Filter1, _Filter1, _Filter2)`
- `template<typename _Tp >`
`_Expr< _UnClos< _Tan, _ValArray, _Tp >, _Tp > tan (const valarray< _Tp > &__v)`

- `template<class _Dom >`
`_Expr< _UnClos< _Tan, _Expr, _Dom >, typename _Dom::value_type > tan (const _Expr< _Dom, typename`
`_Dom::value_type > &__e)`
- `template<class _Dom >`
`_Expr< _UnClos< _Tanh, _Expr, _Dom >, typename _Dom::value_type > tanh (const _Expr< _Dom, typename`
`_Dom::value_type > &__e)`
- `template<typename _Tp >`
`_Expr< _UnClos< _Tanh, _ValArray, _Tp >, _Tp > tanh (const valarray< _Tp > &__v)`
- `template<typename _Ex >`
`void throw_with_nested (_Ex __ex)`
- `template<typename _CharT >`
`_CharT tolower (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`
`_CharT toupper (_CharT __c, const locale &__loc)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation >`
`_OIter transform (_Iter, _Iter, _OIter, _UnaryOperation)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BinaryOperation >`
`_OIter transform (_Iter1, _Iter1, _Iter2, _OIter, _BinaryOperation)`
- `template<typename _InputIterator, typename _OutputIterator, typename _UnaryOperation >`
`_OutputIterator transform (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _UnaryOperation`
`__unary_op)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _BinaryOperation >`
`_OutputIterator transform (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _OutputIterator`
`__result, _BinaryOperation __binary_op)`
- `template<typename _InputIterator, typename _ForwardIterator >`
`_ForwardIterator uninitialized_copy (_InputIterator __first, _InputIterator __last, _ForwardIterator __result)`
- `template<typename _InputIterator, typename _Size, typename _ForwardIterator >`
`_ForwardIterator uninitialized_copy_n (_InputIterator __first, _Size __n, _ForwardIterator __result)`
- `template<typename _ForwardIterator, typename _Tp >`
`void uninitialized_fill (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__x)`
- `template<typename _ForwardIterator, typename _Size, typename _Tp >`
`void uninitialized_fill_n (_ForwardIterator __first, _Size __n, const _Tp &__x)`
- `template<typename _Filter >`
`_Filter unique (_Filter, _Filter)`
- `template<typename _Filter, typename _BinaryPredicate >`
`_Filter unique (_Filter, _Filter, _BinaryPredicate)`
- `template<typename _ForwardIterator >`
`_ForwardIterator unique (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _BinaryPredicate >`
`_ForwardIterator unique (_ForwardIterator __first, _ForwardIterator __last, _BinaryPredicate __binary_pred)`
- `template<typename _Iter, typename _OIter >`
`_OIter unique_copy (_Iter, _Iter, _OIter)`
- `template<typename _Iter, typename _OIter, typename _BinaryPredicate >`
`_OIter unique_copy (_Iter, _Iter, _OIter, _BinaryPredicate)`
- `template<typename _InputIterator, typename _OutputIterator >`
`_OutputIterator unique_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryPredicate >`
`_OutputIterator unique_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryPredicate`
`__binary_pred)`
- `ios_base & unitbuf (ios_base &__base)`
- `template<typename _Filter, typename _Tp >`
`_Filter upper_bound (_Filter, _Filter, const _Tp &)`

- `template<typename _Filter, typename _Tp, typename _Compare >`
`_Filter upper_bound (_Filter, _Filter, const _Tp &, _Compare)`
- `template<typename _ForwardIterator, typename _Tp >`
`_ForwardIterator upper_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp & __val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`
`_ForwardIterator upper_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp & __val, _Compare __comp)`
- `ios_base & uppercase (ios_base & __base)`
- `template<typename _Facet >`
`const _Facet & use_facet (const locale &)`

Matching, Searching, and Replacing

- `template<typename _Bi_iter, typename _Alloc, typename _Ch_type, typename _Rx_traits >`
`bool regex_match (_Bi_iter __s, _Bi_iter __e, match_results< _Bi_iter, _Alloc > & __m, const basic_regex< _Ch_type, _Rx_traits > & __re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits >`
`bool regex_match (_Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits > & __re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Ch_type, typename _Alloc, typename _Rx_traits >`
`bool regex_match (const _Ch_type * __s, match_results< const _Ch_type *, _Alloc > & __m, const basic_regex< _Ch_type, _Rx_traits > & __re, regex_constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits >`
`bool regex_match (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > & __s, match_results< typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > & __m, const basic_regex< _Ch_type, _Rx_traits > & __re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Ch_type, class _Rx_traits >`
`bool regex_match (const _Ch_type * __s, const basic_regex< _Ch_type, _Rx_traits > & __re, regex_constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _Str_allocator, typename _Ch_type, typename _Rx_traits >`
`bool regex_match (const basic_string< _Ch_type, _Ch_traits, _Str_allocator > & __s, const basic_regex< _Ch_type, _Rx_traits > & __re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Bi_iter, typename _Alloc, typename _Ch_type, typename _Rx_traits >`
`bool regex_search (_Bi_iter __s, _Bi_iter __e, match_results< _Bi_iter, _Alloc > & __m, const basic_regex< _Ch_type, _Rx_traits > & __re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits >`
`bool regex_search (_Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits > & __re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Ch_type, class _Alloc, class _Rx_traits >`
`bool regex_search (const _Ch_type * __s, match_results< const _Ch_type *, _Alloc > & __m, const basic_regex< _Ch_type, _Rx_traits > & __e, regex_constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Ch_type, typename _Rx_traits >`
`bool regex_search (const _Ch_type * __s, const basic_regex< _Ch_type, _Rx_traits > & __e, regex_constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _String_allocator, typename _Ch_type, typename _Rx_traits >`
`bool regex_search (const basic_string< _Ch_type, _Ch_traits, _String_allocator > & __s, const basic_regex< _Ch_type, _Rx_traits > & __e, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits >`
`bool regex_search (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > & __s, match_results< typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > & __m, const basic_regex< _Ch_type, _Rx_traits > & __e, regex_constants::match_flag_type __f=regex_constants::match_default)`

- `template<typename _Out_iter, typename _Bi_iter, typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa >`
`_Out_iter regex_replace (_Out_iter __out, _Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, ↵`
`_Rx_traits > &__e, const basic_string< _Ch_type, _St, _Sa > &__fmt, regex_constants::match_flag_type`
`__flags=regex_constants::match_default)`
- `template<typename _Out_iter, typename _Bi_iter, typename _Rx_traits, typename _Ch_type >`
`_Out_iter regex_replace (_Out_iter __out, _Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, ↵`
`_Rx_traits > &__e, const _Ch_type * __fmt, regex_constants::match_flag_type __flags=regex_constants::↵`
`::match_default)`
- `template<typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa, typename _Fst, typename _Fsa >`
`basic_string< _Ch_type, _St, _Sa > regex_replace (const basic_string< _Ch_type, _St, _Sa > &__s, const`
`basic_regex< _Ch_type, _Rx_traits > &__e, const basic_string< _Ch_type, _Fst, _Fsa > &__fmt, regex_↵`
`constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa >`
`basic_string< _Ch_type, _St, _Sa > regex_replace (const basic_string< _Ch_type, _St, _Sa > &__s, const`
`basic_regex< _Ch_type, _Rx_traits > &__e, const _Ch_type * __fmt, regex_constants::match_flag_type __↵`
`flags=regex_constants::match_default)`
- `template<typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa >`
`basic_string< _Ch_type > regex_replace (const _Ch_type * __s, const basic_regex< _Ch_type, _Rx_traits >`
`&__e, const basic_string< _Ch_type, _St, _Sa > &__fmt, regex_constants::match_flag_type __flags=regex_↵`
`_constants::match_default)`
- `template<typename _Rx_traits, typename _Ch_type >`
`basic_string< _Ch_type > regex_replace (const _Ch_type * __s, const basic_regex< _Ch_type, _Rx_traits >`
`&__e, const _Ch_type * __fmt, regex_constants::match_flag_type __flags=regex_constants::match_default)`

Variables

- constexpr `allocator_arg_t allocator_arg`
- decltype(nullptr) typedef `nullptr_t`
- constexpr `piecewise_construct_t piecewise_construct`

3.11.1 Detailed Description

ISO C++ entities toplevel namespace is std.

3.11.2 Typedef Documentation

3.11.2.1 `template<bool _Cache> using std::__umap_traits = typedef __detail::__Hashtable_traits<_Cache, false, true>`

Base types for unordered_map.

Definition at line 39 of file unordered_map.h.

3.11.2.2 `template<bool _Cache> using std::__ummap_traits = typedef __detail::__Hashtable_traits<_Cache, false, false>`

Base types for unordered_multimap.

Definition at line 56 of file unordered_map.h.

3.11.2.3 `template<bool _Cache> using std::__umset_traits = typedef __detail::_Hashtable_traits<_Cache, true, false>`

Base types for `unordered_multiset`.

Definition at line 54 of file `unordered_set.h`.

3.11.2.4 `template<bool _Cache> using std::__uset_traits = typedef __detail::_Hashtable_traits<_Cache, true, true>`

Base types for `unordered_set`.

Definition at line 39 of file `unordered_set.h`.

3.11.2.5 `typedef long long std::streamoff`

Type used by `fpos`, `char_traits<char>`, and `char_traits<wchar_t>`.

In clauses 21.1.3.1 and 27.4.1 `streamoff` is described as an implementation defined type. Note: In versions of GCC up to and including GCC 3.3, `streamoff` was `typedef long`.

Definition at line 94 of file `postypes.h`.

3.11.2.6 `typedef fpos<mbstate_t> std::streampos`

File position for `char` streams.

Definition at line 228 of file `postypes.h`.

3.11.2.7 `typedef ptrdiff_t std::streamsize`

Integral type for I/O operation counts and buffer sizes.

Definition at line 98 of file `postypes.h`.

3.11.2.8 `typedef fpos<mbstate_t> std::u16streampos`

File position for `char16_t` streams.

Definition at line 234 of file `postypes.h`.

3.11.2.9 `typedef fpos<mbstate_t> std::u32streampos`

File position for `char32_t` streams.

Definition at line 236 of file `postypes.h`.

3.11.2.10 `typedef fpos<mbstate_t> std::wstreampos`

File position for `wchar_t` streams.

Definition at line 230 of file `postypes.h`.

3.11.3 Enumeration Type Documentation

3.11.3.1 anonymous enum

Todo Needs documentation! See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation+style.html> This controls some aspect of the sort routines.

Definition at line 1874 of file stl_algo.h.

3.11.4 Function Documentation

3.11.4.1 `template<typename _RandomAccessIterator, typename _Compare> void std::__final_insertion_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`

This is a helper function for the sort routine.

Definition at line 1879 of file stl_algo.h.

References `__insertion_sort()`, and `__unguarded_insertion_sort()`.

Referenced by `__introsort_loop()`.

3.11.4.2 `template<typename _InputIterator, typename _Predicate> _InputIterator std::__find_if (_InputIterator __first, _InputIterator __last, _Predicate __pred, input_iterator_tag) [inline]`

This is an overload used by find algos for the Input Iterator case.

Definition at line 101 of file stl_algo.h.

Referenced by `__find_if()`, `__find_if_not()`, `__find_if_not_n()`, `__search_n_aux()`, `find()`, `find_if()`, `is_permutation()`, `minmax_element()`, and `partition_copy()`.

3.11.4.3 `template<typename _RandomAccessIterator, typename _Predicate> _RandomAccessIterator std::__find_if (_RandomAccessIterator __first, _RandomAccessIterator __last, _Predicate __pred, random_access_iterator_tag)`

This is an overload used by find algos for the RAI case.

Definition at line 112 of file stl_algo.h.

References `__find_if()`, and `__iterator_category()`.

3.11.4.4 `template<typename _InputIterator, typename _Predicate> _InputIterator std::__find_if_not (_InputIterator __first, _InputIterator __last, _Predicate __pred) [inline]`

Provided for `stable_partition` to use.

Definition at line 168 of file stl_algo.h.

References `__find_if()`, and `__iterator_category()`.

Referenced by `__stable_partition_adaptive()`, and `find_if_not()`.

3.11.4.5 `template<typename _InputIterator, typename _Predicate, typename _Distance > _InputIterator std::__find_if_not_n (`
`_InputIterator __first, _Distance & __len, _Predicate __pred)`

Like `find_if_not()`, but uses and updates a count of the remaining range length instead of comparing against an end iterator.

Definition at line 181 of file `stl_algo.h`.

References `__find_if()`.

Referenced by `__inplace_stable_partition()`, and `__stable_partition_adaptive()`.

3.11.4.6 `template<typename _EuclideanRingElement > _EuclideanRingElement std::__gcd (_EuclideanRingElement __m,`
`_EuclideanRingElement __n)`

This is a helper function for the rotate algorithm specialized on RAIs. It returns the greatest common divisor of two integer values.

Definition at line 1229 of file `stl_algo.h`.

3.11.4.7 `template<typename _RandomAccessIterator, typename _Compare > void std::__heap_select (_RandomAccessIterator`
`__first, _RandomAccessIterator __middle, _RandomAccessIterator __last, _Compare __comp)`

This is a helper function for the sort routines.

Definition at line 1669 of file `stl_algo.h`.

Referenced by `__introsort_loop()`, and `__unguarded_partition_pivot()`.

3.11.4.8 `template<typename _ForwardIterator, typename _Predicate, typename _Distance > _ForwardIterator`
`std::__inplace_stable_partition (_ForwardIterator __first, _Predicate __pred, _Distance __len)`

This is a helper function... Requires `__len != 0` and `!__pred(*__first)`, same as `__stable_partition_adaptive`.

Definition at line 1519 of file `stl_algo.h`.

References `__find_if_not_n()`, `advance()`, `distance()`, and `rotate()`.

Referenced by `__stable_partition_adaptive()`.

3.11.4.9 `template<typename _RandomAccessIterator, typename _Compare > void std::__inplace_stable_sort (`
`_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`

This is a helper function for the stable sorting routines.

Definition at line 2771 of file `stl_algo.h`.

References `__insertion_sort()`, and `__merge_without_buffer()`.

Referenced by `merge()`.

3.11.4.10 `template<typename _RandomAccessIterator, typename _Compare> void std::__insertion_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`

This is a helper function for the sort routine.

Definition at line 1839 of file `stl_algo.h`.

References `__unguarded_linear_insert()`.

Referenced by `__final_insertion_sort()`, `__inplace_stable_sort()`, `__introsort_loop()`, and `__move_merge()`.

3.11.4.11 `template<typename _RandomAccessIterator, typename _Size, typename _Compare> void std::__introsort_loop (_RandomAccessIterator __first, _RandomAccessIterator __last, _Size __depth_limit, _Compare __comp)`

This is a helper function for the sort routine.

Definition at line 1939 of file `stl_algo.h`.

References `__final_insertion_sort()`, `__heap_select()`, `__insertion_sort()`, `__lg()`, `__unguarded_partition_pivot()`, and `iter_swap()`.

3.11.4.12 `constexpr int std::__lg (int __n) [inline]`

This is a helper function for the sort routines and for `random.tcc`.

Definition at line 1012 of file `stl_algobase.h`.

Referenced by `__introsort_loop()`, and `nth_element()`.

3.11.4.13 `template<typename _BidirectionalIterator, typename _Distance, typename _Pointer, typename _Compare> void std::__merge_adaptive (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Distance __len1, _Distance __len2, _Pointer __buffer, _Distance __buffer_size, _Compare __comp)`

This is a helper function for the merge routines.

Definition at line 2431 of file `stl_algo.h`.

References `__move_merge_adaptive()`, `__move_merge_adaptive_backward()`, `__rotate_adaptive()`, `advance()`, and `distance()`.

Referenced by `__merge_without_buffer()`, and `__move_merge()`.

3.11.4.14 `template<typename _BidirectionalIterator, typename _Distance, typename _Compare> void std::__merge_without_buffer (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Distance __len1, _Distance __len2, _Compare __comp)`

This is a helper function for the merge routines.

Definition at line 2491 of file `stl_algo.h`.

References `__merge_adaptive()`, `advance()`, `distance()`, `iter_swap()`, and `rotate()`.

Referenced by `__inplace_stable_sort()`.

3.11.4.15 `template<typename _Iterator, typename _Compare> void std::__move_median_to_first (_Iterator __result, _Iterator __a, _Iterator __b, _Iterator __c, _Compare __comp)`

Swaps the median value of *__a, *__b and *__c under __comp to *__result.

Definition at line 78 of file `stl_algo.h`.

References `iter_swap()`.

Referenced by `__unguarded_partition_pivot()`.

3.11.4.16 `template<typename _InputIterator, typename _OutputIterator, typename _Compare> _OutputIterator std::__move_merge (_InputIterator __first1, _InputIterator __last1, _InputIterator __first2, _InputIterator __last2, _OutputIterator __result, _Compare __comp)`

This is a helper function for the `__merge_sort_loop` routines.

Definition at line 2649 of file `stl_algo.h`.

References `__insertion_sort()`, `__merge_adaptive()`, and `min()`.

3.11.4.17 `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare> void std::__move_merge_adaptive (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`

This is a helper function for the `__merge_adaptive` routines.

Definition at line 2320 of file `stl_algo.h`.

Referenced by `__merge_adaptive()`.

3.11.4.18 `template<typename _BidirectionalIterator1, typename _BidirectionalIterator2, typename _BidirectionalIterator3, typename _Compare> void std::__move_merge_adaptive_backward (_BidirectionalIterator1 __first1, _BidirectionalIterator1 __last1, _BidirectionalIterator2 __first2, _BidirectionalIterator2 __last2, _BidirectionalIterator3 __result, _Compare __comp)`

This is a helper function for the `__merge_adaptive` routines.

Definition at line 2346 of file `stl_algo.h`.

Referenced by `__merge_adaptive()`.

3.11.4.19 `template<typename _ForwardIterator, typename _Predicate> _ForwardIterator std::__partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, forward_iterator_tag)`

This is a helper function...

Definition at line 1462 of file `stl_algo.h`.

References `iter_swap()`.

Referenced by `partition()`.

3.11.4.20 `template<typename _BidirectionalIterator, typename _Predicate> _BidirectionalIterator std::__partition (`
`_BidirectionalIterator __first, _BidirectionalIterator __last, _Predicate __pred, bidirectional_iterator_tag)`

This is a helper function...

Definition at line 1487 of file `stl_algo.h`.

References `iter_swap()`.

3.11.4.21 `template<typename _BidirectionalIterator> void std::__reverse (_BidirectionalIterator __first, _BidirectionalIterator`
`__last, bidirectional_iterator_tag)`

This is an uglified `reverse(_BidirectionalIterator, _BidirectionalIterator)` overloaded for bidirectional iterators.

Definition at line 1129 of file `stl_algo.h`.

References `iter_swap()`.

Referenced by `__rotate()`, `includes()`, `next_permutation()`, and `reverse()`.

3.11.4.22 `template<typename _RandomAccessIterator> void std::__reverse (_RandomAccessIterator __first,`
`_RandomAccessIterator __last, random_access_iterator_tag)`

This is an uglified `reverse(_BidirectionalIterator, _BidirectionalIterator)` overloaded for random access iterators.

Definition at line 1149 of file `stl_algo.h`.

References `iter_swap()`.

3.11.4.23 `template<typename _ForwardIterator> void std::__rotate (_ForwardIterator __first, _ForwardIterator __middle,`
`_ForwardIterator __last, forward_iterator_tag)`

This is a helper function for the rotate algorithm.

Definition at line 1243 of file `stl_algo.h`.

References `iter_swap()`.

Referenced by `__gnu_cxx::bitmap_allocator<_Tp>::_M_deallocate_single_object()`, and `rotate()`.

3.11.4.24 `template<typename _BidirectionalIterator> void std::__rotate (_BidirectionalIterator __first, _BidirectionalIterator`
`__middle, _BidirectionalIterator __last, bidirectional_iterator_tag)`

This is a helper function for the rotate algorithm.

Definition at line 1279 of file `stl_algo.h`.

References `__reverse()`, and `iter_swap()`.

```
3.11.4.25  template<typename _RandomAccessIterator > void std::__rotate ( _RandomAccessIterator __first,
    _RandomAccessIterator __middle, _RandomAccessIterator __last, random_access_iterator_tag )
```

This is a helper function for the rotate algorithm.

Definition at line 1309 of file stl_algo.h.

References iter_swap(), swap(), and swap_ranges().

```
3.11.4.26  template<typename _BidirectionalIterator1 , typename _BidirectionalIterator2 , typename _Distance >
    _BidirectionalIterator1 std::__rotate_adaptive ( _BidirectionalIterator1 __first, _BidirectionalIterator1 __middle,
    _BidirectionalIterator1 __last, _Distance __len1, _Distance __len2, _BidirectionalIterator2 __buffer, _Distance
    __buffer_size )
```

This is a helper function for the merge routines.

Definition at line 2389 of file stl_algo.h.

References advance(), distance(), and rotate().

Referenced by __merge_adaptive().

```
3.11.4.27  template<typename _ForwardIterator , typename _Integer , typename _UnaryPredicate > _ForwardIterator
    std::__search_n_aux ( _ForwardIterator __first, _ForwardIterator __last, _Integer __count, _UnaryPredicate
    __unary_pred, std::forward_iterator_tag )
```

This is an helper function for search_n overloaded for forward iterators.

Definition at line 257 of file stl_algo.h.

References __find_if().

Referenced by __search_n_aux().

```
3.11.4.28  template<typename _RandomAccessIter , typename _Integer , typename _UnaryPredicate > _RandomAccessIter
    std::__search_n_aux ( _RandomAccessIter __first, _RandomAccessIter __last, _Integer __count, _UnaryPredicate
    __unary_pred, std::random_access_iterator_tag )
```

This is an helper function for search_n overloaded for random access iterators.

Definition at line 289 of file stl_algo.h.

References __find_if(), __iterator_category(), __search_n_aux(), advance(), and distance().

```
3.11.4.29  template<typename _ForwardIterator , typename _Pointer , typename _Predicate , typename _Distance > _ForwardIterator
    std::__stable_partition_adaptive ( _ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, _Distance __len,
    _Pointer __buffer, _Distance __buffer_size )
```

This is a helper function... Requires __first != __last and !__pred(__first) and __len == distance(__first, __last).

!__pred(__first) allows us to guarantee that we don't move-assign an element onto itself.

Definition at line 1551 of file stl_algo.h.

References __find_if_not(), __find_if_not_n(), __inplace_stable_partition(), advance(), std::_Temporary_buffer<_↵ ForwardIterator, _Tp >::begin(), distance(), std::_Temporary_buffer<_ForwardIterator, _Tp >::requested_size(), rotate(), and std::_Temporary_buffer<_ForwardIterator, _Tp >::size().

3.11.4.30 `template<typename _RandomAccessIterator, typename _Compare> void std::__unguarded_insertion_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp) [inline]`

This is a helper function for the sort routine.

Definition at line 1862 of file `stl_algo.h`.

References `__unguarded_linear_insert()`.

Referenced by `__final_insertion_sort()`.

3.11.4.31 `template<typename _RandomAccessIterator, typename _Compare> void std::__unguarded_linear_insert (_RandomAccessIterator __last, _Compare __comp)`

This is a helper function for the sort routine.

Definition at line 1820 of file `stl_algo.h`.

Referenced by `__insertion_sort()`, and `__unguarded_insertion_sort()`.

3.11.4.32 `template<typename _RandomAccessIterator, typename _Compare> _RandomAccessIterator std::__unguarded_partition (_RandomAccessIterator __first, _RandomAccessIterator __last, _RandomAccessIterator __pivot, _Compare __comp)`

This is a helper function...

Definition at line 1895 of file `stl_algo.h`.

References `iter_swap()`.

Referenced by `__unguarded_partition_pivot()`.

3.11.4.33 `template<typename _RandomAccessIterator, typename _Compare> _RandomAccessIterator std::__unguarded_partition_pivot (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp) [inline]`

This is a helper function...

Definition at line 1916 of file `stl_algo.h`.

References `__heap_select()`, `__move_median_to_first()`, and `__unguarded_partition()`.

Referenced by `__introsort_loop()`.

3.11.4.34 `template<typename _ForwardIterator, typename _OutputIterator, typename _BinaryPredicate> _OutputIterator std::__unique_copy (_ForwardIterator __first, _ForwardIterator __last, _OutputIterator __result, _BinaryPredicate __binary_pred, forward_iterator_tag, output_iterator_tag)`

This is an uglified `unique_copy(_InputIterator, _InputIterator, _OutputIterator, _BinaryPredicate)` overloaded for forward iterators and output iterator as result.

Definition at line 1046 of file `stl_algo.h`.

Referenced by `unique_copy()`.


```
3.11.4.35 template<typename _InputIterator , typename _OutputIterator , typename _BinaryPredicate > _OutputIterator
std::__unique_copy ( _InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryPredicate
__binary_pred, input_iterator_tag , output_iterator_tag )
```

This is an uglified `unique_copy(_InputIterator, _InputIterator, _OutputIterator, _BinaryPredicate)` overloaded for input iterators and output iterator as result.

Definition at line 1075 of file `stl_algo.h`.

```
3.11.4.36 template<typename _InputIterator , typename _ForwardIterator , typename _BinaryPredicate > _ForwardIterator
std::__unique_copy ( _InputIterator __first, _InputIterator __last, _ForwardIterator __result, _BinaryPredicate
__binary_pred, input_iterator_tag , forward_iterator_tag )
```

This is an uglified `unique_copy(_InputIterator, _InputIterator, _OutputIterator, _BinaryPredicate)` overloaded for input iterators and forward iterator as result.

Definition at line 1107 of file `stl_algo.h`.

```
3.11.4.37 template<typename _T1 , typename... _Args> void std::_Construct ( _T1 * __p, _Args &&... __args ) [inline]
```

Constructs an object in existing memory by invoking an allocated object's constructor with an initializer.

Definition at line 74 of file `stl_construct.h`.

Referenced by `std::_Temporary_buffer< _ForwardIterator, _Tp >::end()`, `uninitialized_copy()`, `uninitialized_fill()`, and `uninitialized_fill_n()`.

```
3.11.4.38 template<typename _Tp > void std::_Destroy ( _Tp * __pointer ) [inline]
```

Destroy the object pointed to by a pointer type.

Definition at line 92 of file `stl_construct.h`.

References `__addressof()`.

Referenced by `_Destroy()`, `std::vector< sub_match< _Bi_iter >, allocator< sub_match< _Bi_iter > > >::M_← allocate_and_copy()`, `std::_Temporary_buffer< _ForwardIterator, _Tp >::end()`, `uninitialized_copy()`, `uninitialized_fill()`, `uninitialized_fill_n()`, and `std::vector< sub_match< _Bi_iter >, allocator< sub_match< _Bi_iter > > >::~~vector()`.

```
3.11.4.39 template<typename _ForwardIterator > void std::_Destroy ( _ForwardIterator __first, _ForwardIterator __last )
[inline]
```

Destroy a range of objects. If the value_type of the object has a trivial destructor, the compiler should optimize all of this away, otherwise the objects' destructors must be invoked.

Definition at line 122 of file `stl_construct.h`.

```
3.11.4.40 template<typename _ForwardIterator , typename _Allocator > void std::_Destroy ( _ForwardIterator __first,
_FForwardIterator __last, _Allocator & __alloc )
```

Destroy a range of objects using the supplied allocator. For nondefault allocators we do not optimize away invocation of `destroy()` even if `_Tp` has a trivial destructor.

Definition at line 138 of file `stl_construct.h`.

References `__addressof()`, and `_Destroy()`.

```
3.11.4.41 template<typename _InputIterator , typename _Tp > _Tp std::accumulate ( _InputIterator __first, _InputIterator __last,
_Tp __init ) [inline]
```

Accumulate values in a range.

Accumulates the values in the range `[first,last)` using `operator+()`. The initial value is `init`. The values are processed in order.

Parameters

<code>__first</code>	Start of range.
<code>__last</code>	End of range.
<code>__init</code>	Starting value to add other values to.

Returns

The final sum.

Definition at line 120 of file `stl_numeric.h`.

Referenced by `__gnu_parallel::__parallel_partial_sum_linear()`.

```
3.11.4.42  template<typename _InputIterator, typename _Tp, typename _BinaryOperation> _Tp std::accumulate ( _InputIterator
            __first, _InputIterator __last, _Tp __init, _BinaryOperation __binary_op ) [inline]
```

Accumulate values in a range with operation.

Accumulates the values in the range `[first,last)` using the function object `__binary_op`. The initial value is `__init`. The values are processed in order.

Parameters

<code>__first</code>	Start of range.
<code>__last</code>	End of range.
<code>__init</code>	Starting value to add other values to.
<code>__binary_op</code>	Function object to accumulate with.

Returns

The final sum.

Definition at line 146 of file `stl_numeric.h`.

```
3.11.4.43  template<typename _InputIterator, typename _OutputIterator> _OutputIterator std::adjacent_difference ( _InputIterator
            __first, _InputIterator __last, _OutputIterator __result )
```

Return differences between adjacent values.

Computes the difference between adjacent values in the range `[first,last)` using operator-() and writes the result to `__result`.

Parameters

<code>__first</code>	Start of input range.
<code>__last</code>	End of input range.
<code>__result</code>	Output sums.

Returns

Iterator pointing just beyond the values written to result.

_GLIBCXX_RESOLVE_LIB_DEFECTS DR 539. partial_sum and adjacent_difference should mention requirements

Definition at line 317 of file stl_numeric.h.

```
3.11.4.44 template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation> _OutputIterator
std::adjacent_difference ( _InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryOperation
__binary_op )
```

Return differences between adjacent values.

Computes the difference between adjacent values in the range [__first,__last) using the function object __binary_op and writes the result to __result.

Parameters

__first	Start of input range.
__last	End of input range.
__result	Output sum.
__binary_op	Function object.

Returns

Iterator pointing just beyond the values written to result.

_GLIBCXX_RESOLVE_LIB_DEFECTS DR 539. partial_sum and adjacent_difference should mention requirements

Definition at line 360 of file stl_numeric.h.

```
3.11.4.45 template<typename _InputIterator, typename _Distance> void std::advance ( _InputIterator & __i, _Distance __n )
[inline]
```

A generalization of pointer arithmetic.

Parameters

$_i$	An input iterator.
$_n$	The <i>delta</i> by which to change $_i$.

Returns

Nothing.

This increments `i` by `n`. For bidirectional and random access iterators, `__n` may be negative, in which case `__i` is decremented.

For random access iterators, this uses their `+` and `-` operations and are constant time. For other iterator classes they are linear time.

Definition at line 173 of file `stl_iterator_base_funcs.h`.

References `__iterator_category()`.

Referenced by `__inplace_stable_partition()`, `__merge_adaptive()`, `__merge_without_buffer()`, `__rotate_adaptive()`, `__search_n_aux()`, `__stable_partition_adaptive()`, `std::deque<_StateSeqT>::clear()`, `fill_n()`, `__gnu_pbds::detail::pat_<_trie_base::_Node_citer<Node, Leaf, Head, Inode, _Cliterator, Iterator, _Alloc>::get_child()`, `__gnu_pbds::detail::pat_<_trie_base::_Node_iter<Node, Leaf, Head, Inode, _Cliterator, Iterator, _Alloc>::get_child()`, `lower_bound()`, `minmax_<_element()`, `partition_point()`, and `upper_bound()`.

3.11.4.46 `template<class _Container> auto std::begin (_Container & __cont)-> decltype(__cont.begin())` `[inline]`

Return an iterator pointing to the first element of the container.

Parameters

<code>__cont</code>	Container.
---------------------	------------

Definition at line 48 of file `range_access.h`.

Referenced by `__gnu_profile::_report()`, `std::match_results<_Bi_iter>::begin()`, `std::match_results<_Bi_iter>::cbegin()`, `std::deque<_StateSeqT>::clear()`, `std::vector<sub_match<_Bi_iter>, allocator<sub_match<_Bi_iter>>>::crend()`, `std::list<__inp, __rebind_inp>::crend()`, `std::vector<sub_match<_Bi_iter>, allocator<sub_match<_Bi_iter>>>::empty()`, `std::vector<sub_match<_Bi_iter>, allocator<sub_match<_Bi_iter>>>::erase()`, `std::vector<sub_match<_Bi_iter>, allocator<sub_match<_Bi_iter>>>::front()`, `std::list<__inp, __rebind_inp>::front()`, `std::deque<_StateSeqT>::front()`, `__gnu_pbds::detail::pat_<_trie_base::_Node_iter<Node, Leaf, Head, Inode, _Cliterator, Iterator, _Alloc>::get_child()`, `std::vector<sub_match<_Bi_iter>, allocator<sub_match<_Bi_iter>>>::insert()`, `std::deque<_StateSeqT>::insert()`, `__gnu_pbds::detail::pat_<_trie_map<Key, Mapped, Node_And_It_Traits, _Alloc>::node_end()`, `std::list<__inp, __rebind_inp>::pop_front()`, `std::list<__inp, __rebind_inp>::push_front()`, `std::vector<sub_match<_Bi_iter>, allocator<sub_match<_Bi_iter>>>::rend()`, `std::list<__inp, __rebind_inp>::rend()`, `std::forward_list<_Tp, _Alloc>::reverse()`, `std::list<__inp, __rebind_inp>::size()`, and `std::deque<_StateSeqT>::~~deque()`.

3.11.4.47 `template<class _Container> auto std::begin (const _Container & __cont)-> decltype(__cont.begin())` `[inline]`

Return an iterator pointing to the first element of the const container.

Parameters

<code>__cont</code>	Container.
---------------------	------------

Definition at line 58 of file `range_access.h`.

3.11.4.48 `template<class _Tp, size_t _Nm> _Tp* std::begin (_Tp(& __arr[_Nm]) [inline]`

Return an iterator pointing to the first element of the array.

Parameters

<code>__arr</code>	Array.
--------------------	--------

Definition at line 87 of file `range_access.h`.

3.11.4.49 `ios_base& std::boolalpha (ios_base & __base) [inline]`

Calls `base.setf(ios_base::boolalpha)`.

Definition at line 795 of file `ios_base.h`.

References `__gnu_debug::__base()`, `std::ios_base::boolalpha`, and `std::ios_base::setf()`.

3.11.4.50 `template<typename _Tp, typename _Tp1, _Lock_policy _Lp> __shared_ptr<_Tp, _Lp> std::const_pointer_cast (const __shared_ptr<_Tp1, _Lp> & __r) [inline], [noexcept]`

`const_pointer_cast`

Definition at line 1320 of file `shared_ptr_base.h`.

3.11.4.51 `ios_base& std::dec (ios_base & __base) [inline]`

Calls `base.setf(ios_base::dec, ios_base::basefield)`.

Definition at line 933 of file `ios_base.h`.

References `__gnu_debug::__base()`, `std::ios_base::basefield`, `std::ios_base::dec`, and `std::ios_base::setf()`.

3.11.4.52 `template<typename _InputIterator > iterator_traits<_InputIterator>::difference_type std::distance (_InputIterator __first, _InputIterator __last) [inline]`

A generalization of pointer arithmetic.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.

Returns

The distance between them.

Returns `n` such that `__first + n == __last`. This requires that `__last` must be reachable from `__first`. Note that `n` may be negative.

For random access iterators, this uses their `+` and `-` operations and are constant time. For other iterator classes they are linear time.

Definition at line 114 of file `stl_iterator_base_funcs.h`.

References `__iterator_category()`.

Referenced by `__inplace_stable_partition()`, `__merge_adaptive()`, `__merge_without_buffer()`, `__rotate_adaptive()`, `__search_n_aux()`, `__stable_partition_adaptive()`, `std::vector< sub_match< _Bi_iter >, allocator< sub_match< _Bi_iter > > >::__M_allocate_and_copy()`, `std::deque< _StateSeqT >::clear()`, `equal()`, `fill_n()`, `__gnu_pbds::detail::pat_< _trie_base::__Node_iter< Node, Leaf, Head, Inode, _Clterator, Iterator, _Alloc >::get_child()`, `is_heap_until()`, `is_permutation()`, `std::sub_match< _Bi_iter >::length()`, `lower_bound()`, `minmax_element()`, `__gnu_parallel::multiseq_partition()`, `__gnu_parallel::multiseq_selection()`, `__gnu_pbds::detail::pat_< _trie_base::__Node_citer< Node, Leaf, Head, Inode, _Clterator, Iterator, _Alloc >::num_children()`, `partition_point()`, `std::match_results< _Bi_iter >::position()`, `std::list< __inp, __rebind_inp >::size()`, and `upper_bound()`.

3.11.4.53 `template<typename _Tp, typename _Tp1, _Lock_policy _Lp> __shared_ptr<_Tp, _Lp> std::dynamic_pointer_cast (const __shared_ptr<_Tp1, _Lp> & _r) [inline], [noexcept]`

`dynamic_pointer_cast`

Definition at line 1330 of file `shared_ptr_base.h`.

References `swap()`.

3.11.4.54 `template<class _Container > auto std::end (_Container & __cont)-> decltype(__cont.end()) [inline]`

Return an iterator pointing to one past the last element of the container.

Parameters

<code>__cont</code>	Container.
---------------------	------------

Definition at line 68 of file `range_access.h`.

Referenced by `__gnu_profile::__report()`, `std::vector< sub_match< _Bi_iter >, allocator< sub_match< _Bi_iter > > >::back()`, `std::list< __inp, __rebind_inp >::back()`, `std::deque< _StateSeqT >::back()`, `std::match_results< _Bi_iter >::cend()`, `std::deque< _StateSeqT >::clear()`, `std::vector< sub_match< _Bi_iter >, allocator< sub_match< _Bi_iter > > >::crbegin()`, `std::list< __inp, __rebind_inp >::crbegin()`, `std::vector< sub_match< _Bi_iter >, allocator< sub_match< _Bi_iter > > >::empty()`, `std::match_results< _Bi_iter >::end()`, `__gnu_pbds::detail::pat_< _trie_base::__Node_iter< Node, Leaf, Head, Inode, _Clterator, Iterator, _Alloc >::get_child()`, `__gnu_pbds::detail::pat_< _trie_base::__Node_map< Key, Mapped, Node_And_It_Traits, _Alloc >::node_end()`, `std::vector< sub_match< _Bi_iter >, allocator< sub_match< _Bi_iter > > >::push_back()`, `std::list< __inp, __rebind_inp >::push_back()`, `std::vector< sub_match< _Bi_iter >, allocator< sub_match< _Bi_iter > > >::rbegin()`, `std::list< __inp, __rebind_inp >::rbegin()`, `std::vector< sub_match< _Bi_iter >, allocator< sub_match< _Bi_iter > > >::resize()`, `std::forward_list< _Tp, _Alloc >::reverse()`, `std::list< __inp, __rebind_inp >::size()`, and `std::deque< _StateSeqT >::~~deque()`.

3.11.4.55 `template<class _Container > auto std::end (const _Container & __cont)-> decltype(__cont.end())` `[inline]`

Return an iterator pointing to one past the last element of the const container.

Parameters

<code>__cont</code>	Container.
---------------------	------------

Definition at line 78 of file `range_access.h`.

3.11.4.56 `template<class _Tp, size_t _Nm> _Tp* std::end (_Tp(& __arr[_Nm]) [inline]`

Return an iterator pointing to one past the last element of the array.

Parameters

<code>__arr</code>	Array.
--------------------	--------

Definition at line 97 of file `range_access.h`.

3.11.4.57 `ios_base& std::fixed (ios_base & __base) [inline]`

Calls `base.setf(ios_base::fixed, ios_base::floatfield)`.

Definition at line 958 of file `ios_base.h`.

References `__gnu_debug::__base()`, `std::ios_base::fixed`, `std::ios_base::floatfield`, and `std::ios_base::setf()`.

3.11.4.58 `template<typename _Tp> pair<_Tp*, ptrdiff_t> std::get_temporary_buffer (ptrdiff_t __len) [noexcept]`

Allocates a temporary buffer.

Parameters

<code>__len</code>	The number of objects of type <code>Tp</code> .
--------------------	---

Returns

See full description.

Reinventing the wheel, but this time with prettier spokes!

This function tries to obtain storage for `__len` adjacent `Tp` objects. The objects themselves are not constructed, of course. A `pair<>` is returned containing *the buffer's address and capacity (in the units of `sizeof(_Tp)`)*, or a pair of 0 values if no storage can be obtained. Note that the capacity obtained may be less than that requested if the memory is unavailable; you should compare `len` with the `.second` return value.

Provides the nothrow exception guarantee.

Definition at line 85 of file `stl_tempbuf.h`.

Referenced by `std::_Temporary_buffer<_ForwardIterator, _Tp>::_Temporary_buffer()`.

3.11.4.59 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base> basic_istream<_CharT, _Traits>& std::getline (basic_istream<_CharT, _Traits> & __is,
__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base> & __str, _CharT __delim)`

Read a line from stream into a string.

Parameters

<code>__is</code>	Input stream.
<code>__str</code>	Buffer to store into.
<code>__delim</code>	Character marking end of line.

Returns

Reference to the input stream.

Stores characters from `__is` into `__str` until `__delim` is found, the end of the stream is encountered, or `str.max_size()` is reached. If `is.width()` is non-zero, that is the limit on the number of characters stored into `__str`. Any previous contents of `__str` are erased. If `delim` was encountered, it is extracted but not stored into `__str`.

3.11.4.60 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base> basic_istream<_CharT, _Traits>& std::getline (basic_istream<_CharT, _Traits> & __is,
__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base> & __str) [inline]`

Read a line from stream into a string.

Parameters

<code>__is</code>	Input stream.
<code>__str</code>	Buffer to store into.

Returns

Reference to the input stream.

Stores characters from `is` into `__str` until '`'` is found, the end of the stream is encountered, or `str.max_size()` is reached. If `is.width()` is non-zero, that is the limit on the number of characters stored into `__str`. Any previous contents of `__str` are erased. If end of line was encountered, it is extracted but not stored into `__str`.

Definition at line 2676 of file `vstring.h`.

References `__gnu_debug::__base()`.

Referenced by operator<<().

3.11.4.61 `template<typename _CharT, typename _Traits, typename _Alloc> basic_istream<_CharT, _Traits>& std::getline (basic_istream<_CharT, _Traits> & __is, basic_string<_CharT, _Traits, _Alloc> & __str, _CharT __delim)`

Read a line from stream into a string.

Parameters

<code>__is</code>	Input stream.
<code>__str</code>	Buffer to store into.
<code>__delim</code>	Character marking end of line.

Returns

Reference to the input stream.

Stores characters from `__is` into `__str` until `__delim` is found, the end of the stream is encountered, or `str.max_size()` is reached. Any previous contents of `__str` are erased. If `__delim` is encountered, it is extracted but not stored into `__str`.

Referenced by `__gnu_profile::__report()`, `getline()`, and `operator<<()`.

```
3.11.4.62  template<typename _CharT, typename _Traits, typename _Alloc> basic_istream<_CharT, _Traits>& std::getline (
            basic_istream<_CharT, _Traits> & __is, basic_string<_CharT, _Traits, _Alloc> & __str ) [inline]
```

Read a line from stream into a string.

Parameters

<code>__is</code>	Input stream.
<code>__str</code>	Buffer to store into.

Returns

Reference to the input stream.

Stores characters from `is` into `__str` until '`'` is found, the end of the stream is encountered, or `str.max_size()` is reached. Any previous contents of `__str` are erased. If end of line is encountered, it is extracted but not stored into `__str`.

Definition at line 2811 of file `basic_string.h`.

References `getline()`.

```
3.11.4.63  template<typename _CharT, typename _Traits, typename _Alloc> basic_istream<_CharT, _Traits>& std::getline
            ( basic_istream<_CharT, _Traits> && __is, basic_string<_CharT, _Traits, _Alloc> & __str, _CharT __delim )
            [inline]
```

Read a line from an rvalue stream into a string.

Definition at line 2819 of file `basic_string.h`.

References `getline()`.

3.11.4.64 `template<typename _CharT, typename _Traits, typename _Alloc> basic_istream<_CharT, _Traits>& std::getline (basic_istream<_CharT, _Traits> && __is, basic_string<_CharT, _Traits, _Alloc> & __str) [inline]`

Read a line from an rvalue stream into a string.

Definition at line 2826 of file `basic_string.h`.

References `__gnu_debug::__base()`, `std::basic_string<_CharT, _Traits, _Alloc>::c_str()`, and `getline()`.

3.11.4.65 `ios_base& std::hex (ios_base & __base) [inline]`

Calls `base.setf(ios_base::hex, ios_base::basefield)`.

Definition at line 941 of file `ios_base.h`.

References `__gnu_debug::__base()`, `std::ios_base::basefield`, `std::ios_base::hex`, and `std::ios_base::setf()`.

3.11.4.66 `template<typename _InputIterator1, typename _InputIterator2, typename _Tp> _Tp std::inner_product (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Tp __init) [inline]`

Compute inner product of two ranges.

Starting with an initial value of `__init`, multiplies successive elements from the two ranges and adds each product into the accumulated value using operator`+`(). The values in the ranges are processed in order.

Parameters

<code>__first1</code>	Start of range 1.
<code>__last1</code>	End of range 1.
<code>__first2</code>	Start of range 2.
<code>__init</code>	Starting value to add other values to.

Returns

The final inner product.

Definition at line 174 of file `stl_numeric.h`.

3.11.4.67 `template<typename _InputIterator1, typename _InputIterator2, typename _Tp, typename _BinaryOperation1, typename _BinaryOperation2> _Tp std::inner_product (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Tp __init, _BinaryOperation1 __binary_op1, _BinaryOperation2 __binary_op2) [inline]`

Compute inner product of two ranges.

Starting with an initial value of `__init`, applies `__binary_op2` to successive elements from the two ranges and accumulates each result into the accumulated value using `__binary_op1`. The values in the ranges are processed in order.

Parameters

<code>__first1</code>	Start of range 1.
<code>__last1</code>	End of range 1.
<code>__first2</code>	Start of range 2.
<code>__init</code>	Starting value to add other values to.
<code>__binary_op1</code>	Function object to accumulate with.
<code>__binary_op2</code>	Function object to apply to pairs of input values.

Returns

The final inner product.

Definition at line 206 of file `stl_numeric.h`.

3.11.4.68 `ios_base& std::internal (ios_base & __base) [inline]`

Calls `base.setf(ios_base::internal, ios_base::adjustfield)`.

Definition at line 908 of file `ios_base.h`.

References `__gnu_debug::__base()`, `std::ios_base::adjustfield`, and `std::ios_base::internal`.

3.11.4.69 `template<typename _ForwardIterator, typename _Tp> void std::iota (_ForwardIterator __first, _ForwardIterator __last, _Tp __value)`

Create a range of sequentially increasing values.

For each element in the range `[first,last)` assigns `value` and increments `value` as if by `++value`.

Parameters

<code>__first</code>	Start of range.
<code>__last</code>	End of range.
<code>__value</code>	Starting value.

Returns

Nothing.

Definition at line 82 of file `stl_numeric.h`.

3.11.4.70 `template<typename _CharT> bool std::isalnum (_CharT __c, const locale & __loc) [inline]`

Convenience interface to `ctype.is(ctype_base::alnum, __c)`.

Definition at line 2584 of file `locale_facets.h`.

References `std::__ctype_abstract_base< _CharT >::is()`.

3.11.4.71 `template<typename _CharT> bool std::isalpha (_CharT __c, const locale & __loc) [inline]`

Convenience interface to `ctype.is(ctype_base::alpha, __c)`.

Definition at line 2560 of file `locale_facets.h`.

References `std::__ctype_abstract_base< _CharT >::is()`.

3.11.4.72 `template<typename _CharT> bool std::isctrl (_CharT __c, const locale & __loc) [inline]`

Convenience interface to `ctype.is(ctype_base::cntrl, __c)`.

Definition at line 2542 of file `locale_facets.h`.

References `std::__ctype_abstract_base< _CharT >::is()`.

3.11.4.73 `template<typename _CharT> bool std::isdigit (_CharT __c, const locale & __loc) [inline]`

Convenience interface to `ctype.is(ctype_base::digit, __c)`.

Definition at line 2566 of file `locale_facets.h`.

References `std::__ctype_abstract_base< _CharT >::is()`.

3.11.4.74 `template<typename _CharT> bool std::isgraph (_CharT __c, const locale & __loc) [inline]`

Convenience interface to `ctype.is(ctype_base::graph, __c)`.

Definition at line 2590 of file `locale_facets.h`.

References `std::__ctype_abstract_base< _CharT >::is()`.

3.11.4.75 `template<typename _CharT> bool std::islower (_CharT __c, const locale & __loc) [inline]`

Convenience interface to `ctype.is(ctype_base::lower, __c)`.

Definition at line 2554 of file `locale_facets.h`.

References `std::__ctype_abstract_base< _CharT >::is()`.

3.11.4.76 `template<typename _CharT> bool std::isprint (_CharT __c, const locale & __loc) [inline]`

Convenience interface to `ctype.is(ctype_base::print, __c)`.

Definition at line 2536 of file `locale_facets.h`.

References `std::__ctype_abstract_base< _CharT >::is()`.

3.11.4.77 `template<typename _CharT> bool std::ispunct (_CharT __c, const locale & __loc) [inline]`

Convenience interface to `ctype.is(ctype_base::punct, __c)`.

Definition at line 2572 of file `locale_facets.h`.

References `std::__ctype_abstract_base< _CharT >::is()`.

3.11.4.78 `template<typename _CharT> bool std::isspace (_CharT __c, const locale & __loc) [inline]`

Convenience interface to `ctype.is(ctype_base::space, __c)`.

Definition at line 2530 of file `locale_facets.h`.

References `std::__ctype_abstract_base< _CharT >::is()`.

3.11.4.79 `template<typename _CharT> bool std::isupper (_CharT __c, const locale & __loc) [inline]`

Convenience interface to `ctype.is(ctype_base::upper, __c)`.

Definition at line 2548 of file `locale_facets.h`.

References `std::__ctype_abstract_base< _CharT >::is()`.

3.11.4.80 `template<typename _CharT> bool std::isxdigit (_CharT __c, const locale & __loc) [inline]`

Convenience interface to `ctype.is(ctype_base::xdigit, __c)`.

Definition at line 2578 of file `locale_facets.h`.

References `std::__ctype_abstract_base< _CharT >::is()`.

3.11.4.81 `ios_base& std::left (ios_base & __base) [inline]`

Calls `base.setf(ios_base::left, ios_base::adjustfield)`.

Definition at line 916 of file `ios_base.h`.

References `__gnu_debug::__base()`, `std::ios_base::adjustfield`, `std::ios_base::left`, and `std::ios_base::setf()`.

3.11.4.82 `ios_base& std::noboolalpha (ios_base & __base) [inline]`

Calls `base.unsetf(ios_base::boolalpha)`.

Definition at line 803 of file `ios_base.h`.

References `__gnu_debug::__base()`, `std::ios_base::boolalpha`, and `std::ios_base::unsetf()`.

3.11.4.83 ios_base& std::noshowbase (ios_base & __base) [inline]

Calls base.unsetf(ios_base::showbase).

Definition at line 819 of file ios_base.h.

References __gnu_debug::__base(), std::ios_base::showbase, and std::ios_base::unsetf().

3.11.4.84 ios_base& std::noshowpoint (ios_base & __base) [inline]

Calls base.unsetf(ios_base::showpoint).

Definition at line 835 of file ios_base.h.

References __gnu_debug::__base(), std::ios_base::showpoint, and std::ios_base::unsetf().

3.11.4.85 ios_base& std::noshowpos (ios_base & __base) [inline]

Calls base.unsetf(ios_base::showpos).

Definition at line 851 of file ios_base.h.

References __gnu_debug::__base(), std::ios_base::showpos, and std::ios_base::unsetf().

3.11.4.86 ios_base& std::noskipws (ios_base & __base) [inline]

Calls base.unsetf(ios_base::skipws).

Definition at line 867 of file ios_base.h.

References __gnu_debug::__base(), std::ios_base::skipws, and std::ios_base::unsetf().

3.11.4.87 ios_base& std::nounitbuf (ios_base & __base) [inline]

Calls base.unsetf(ios_base::unitbuf).

Definition at line 899 of file ios_base.h.

References __gnu_debug::__base(), std::ios_base::unitbuf, and std::ios_base::unsetf().

3.11.4.88 ios_base& std::nouppercase (ios_base & __base) [inline]

Calls base.unsetf(ios_base::uppercase).

Definition at line 883 of file ios_base.h.

References __gnu_debug::__base(), std::ios_base::unsetf(), and std::ios_base::uppercase.

3.11.4.89 `ios_base& std::oct (ios_base & __base) [inline]`

Calls `base.setf(ios_base::oct, ios_base::basefield)`.

Definition at line 949 of file `ios_base.h`.

References `__gnu_debug::__base()`, `std::ios_base::basefield`, `std::ios_base::oct`, and `std::ios_base::setf()`.

3.11.4.90 `template<typename _Tp > bool std::operator!= (const _Fwd_list_iterator< _Tp > & __x, const _Fwd_list_const_iterator< _Tp > & __y) [inline], [noexcept]`

Forward list iterator inequality comparison.

Definition at line 266 of file `forward_list.h`.

3.11.4.91 `template<typename _Tp, typename _Seq > bool std::operator!= (const stack< _Tp, _Seq > & __x, const stack< _Tp, _Seq > & __y) [inline]`

Based on `operator==`.

Definition at line 270 of file `stl_stack.h`.

3.11.4.92 `template<typename _Tp, typename _Seq > bool std::operator!= (const queue< _Tp, _Seq > & __x, const queue< _Tp, _Seq > & __y) [inline]`

Based on `operator==`.

Definition at line 295 of file `stl_queue.h`.

3.11.4.93 `template<typename _Key, typename _Compare, typename _Alloc > bool std::operator!= (const multiset< _Key, _Compare, _Alloc > & __x, const multiset< _Key, _Compare, _Alloc > & __y) [inline]`

Returns `!(x == y)`.

Definition at line 801 of file `stl_multiset.h`.

3.11.4.94 `template<typename _Key, typename _Compare, typename _Alloc > bool std::operator!= (const set< _Key, _Compare, _Alloc > & __x, const set< _Key, _Compare, _Alloc > & __y) [inline]`

Returns `!(x == y)`.

Definition at line 817 of file `stl_set.h`.

3.11.4.95 `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc > bool std::operator!= (const multimap< _Key, _Tp, _Compare, _Alloc > & __x, const multimap< _Key, _Tp, _Compare, _Alloc > & __y) [inline]`

Based on `operator==`.

Definition at line 927 of file `stl_multimap.h`.

3.11.4.96 `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc> bool std::operator!=(const map<_Key, _Tp, _Compare, _Alloc> &__x, const map<_Key, _Tp, _Compare, _Alloc> &__y) [inline]`

Based on operator==.

Definition at line 1024 of file `stl_map.h`.

3.11.4.97 `template<typename _Tp, typename _Alloc> bool std::operator!=(const forward_list<_Tp, _Alloc> &__x, const forward_list<_Tp, _Alloc> &__y) [inline]`

Based on operator==.

Definition at line 1373 of file `forward_list.h`.

3.11.4.98 `template<typename _Tp, typename _Alloc> bool std::operator!=(const vector<_Tp, _Alloc> &__x, const vector<_Tp, _Alloc> &__y) [inline]`

Based on operator==.

Definition at line 1535 of file `stl_vector.h`.

3.11.4.99 `template<typename _Tp, typename _Alloc> bool std::operator!=(const list<_Tp, _Alloc> &__x, const list<_Tp, _Alloc> &__y) [inline]`

Based on operator==.

Definition at line 1765 of file `stl_list.h`.

3.11.4.100 `template<typename _Tp, typename _Alloc> bool std::operator!=(const deque<_Tp, _Alloc> &__x, const deque<_Tp, _Alloc> &__y) [inline]`

Based on operator==.

Definition at line 2055 of file `stl_deque.h`.

3.11.4.101 `template<typename _CharT, typename _Traits, typename _Alloc> bool std::operator!=(const basic_string<_CharT, _Traits, _Alloc> &__lhs, const basic_string<_CharT, _Traits, _Alloc> &__rhs) [inline]`

Test difference of two strings.

Parameters

<code>__lhs</code>	First string.
<code>__rhs</code>	Second string.

Returns

True if `__lhs.compare(__rhs) != 0`. False otherwise.

Definition at line 2550 of file `basic_string.h`.

3.11.4.102 `template<typename _CharT, typename _Traits, typename _Alloc> bool std::operator!= (const _CharT * __lhs, const basic_string< _CharT, _Traits, _Alloc > & __rhs) [inline]`

Test difference of C string and string.

Parameters

<code>__lhs</code>	C string.
<code>__rhs</code>	String.

Returns

True if `__rhs.compare(__lhs) != 0`. False otherwise.

Definition at line 2562 of file `basic_string.h`.

3.11.4.103 `template<typename _CharT, typename _Traits, typename _Alloc> bool std::operator!= (const basic_string< _CharT, _Traits, _Alloc > & __lhs, const _CharT * __rhs) [inline]`

Test difference of string and C string.

Parameters

<code>__lhs</code>	String.
<code>__rhs</code>	C string.

Returns

True if `__lhs.compare(__rhs) != 0`. False otherwise.

Definition at line 2574 of file `basic_string.h`.

3.11.4.104 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string< _CharT, _Traits, _Alloc> std::operator+ (const basic_string< _CharT, _Traits, _Alloc > & __lhs, const basic_string< _CharT, _Traits, _Alloc > & __rhs)`

Concatenate two strings.

Parameters

<code>__lhs</code>	First string.
<code>__rhs</code>	Last string.

Returns

New string with value of `__lhs` followed by `__rhs`.

Definition at line 2383 of file `basic_string.h`.

References `std::basic_string<_CharT, _Traits, _Alloc>::append()`.

Referenced by operator+().

3.11.4.105 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string<_CharT, _Traits, _Alloc>
std::operator+ (const _CharT * __lhs, const basic_string<_CharT, _Traits, _Alloc> & __rhs)`

Concatenate C string and string.

Parameters

<code>__lhs</code>	First string.
<code>__rhs</code>	Last string.

Returns

New string with value of `__lhs` followed by `__rhs`.

3.11.4.106 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string<_CharT, _Traits, _Alloc>
std::operator+ (_CharT __lhs, const basic_string<_CharT, _Traits, _Alloc> & __rhs)`

Concatenate character and string.

Parameters

<code>__lhs</code>	First string.
<code>__rhs</code>	Last string.

Returns

New string with `__lhs` followed by `__rhs`.

3.11.4.107 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string<_CharT, _Traits, _Alloc>
std::operator+ (const basic_string<_CharT, _Traits, _Alloc> & __lhs, const _CharT * __rhs) [inline]`

Concatenate string and C string.

Parameters

<code>__lhs</code>	First string.
<code>__rhs</code>	Last string.

Returns

New string with `__lhs` followed by `__rhs`.

Definition at line 2420 of file `basic_string.h`.

References `std::basic_string<_CharT, _Traits, _Alloc>::append()`.

3.11.4.108 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string<_CharT, _Traits, _Alloc>
std::operator+ (const basic_string<_CharT, _Traits, _Alloc> &__lhs, _CharT __rhs) [inline]`

Concatenate string and character.

Parameters

<code>__lhs</code>	First string.
<code>__rhs</code>	Last string.

Returns

New string with `__lhs` followed by `__rhs`.

Definition at line 2436 of file `basic_string.h`.

References `std::basic_string<_CharT, _Traits, _Alloc>::append()`, `std::basic_string<_CharT, _Traits, _Alloc>::capacity()`, `std::basic_string<_CharT, _Traits, _Alloc>::insert()`, `move()`, `operator+()`, and `std::basic_string<_CharT, _Traits, _Alloc>::size()`.

3.11.4.109 `template<typename _Tp, typename _Seq> bool std::operator< (const stack<_Tp, _Seq> &__x, const stack<_Tp, _Seq> &__y) [inline]`

Stack ordering relation.

Parameters

<code>__x</code>	A stack.
<code>__y</code>	A stack of the same type as <code>x</code> .

Returns

True iff `x` is lexicographically less than `__y`.

This is an total ordering relation. Complexity and semantics depend on the underlying sequence type, but the expected rules are: this relation is linear in the size of the sequences, the elements must be comparable with `<`, and `std::lexicographical_compare()` is usually used to make the determination.

Definition at line 264 of file `stl_stack.h`.

3.11.4.110 `template<typename _Tp, typename _Seq> bool std::operator< (const queue< _Tp, _Seq> & __x, const queue< _Tp, _Seq> & __y) [inline]`

Queue ordering relation.

Parameters

<code>__x</code>	A queue.
<code>__y</code>	A queue of the same type as <code>x</code> .

Returns

True iff `__x` is lexicographically less than `__y`.

This is an total ordering relation. Complexity and semantics depend on the underlying sequence type, but the expected rules are: this relation is linear in the size of the sequences, the elements must be comparable with `<`, and `std::lexicographical_compare()` is usually used to make the determination.

Definition at line 289 of file `stl_queue.h`.

References `std::queue< _Tp, _Sequence>::c`.

3.11.4.111 `template<typename _Key, typename _Compare, typename _Alloc> bool std::operator< (const multiset< _Key, _Compare, _Alloc> & __x, const multiset< _Key, _Compare, _Alloc> & __y) [inline]`

Multiset ordering relation.

Parameters

<code>__x</code>	A multiset.
<code>__y</code>	A multiset of the same type as <code>__x</code> .

Returns

True iff `__x` is lexicographically less than `__y`.

This is a total ordering relation. It is linear in the size of the sets. The elements must be comparable with `<`.

See `std::lexicographical_compare()` for how the determination is made.

Definition at line 794 of file `stl_multiset.h`.

3.11.4.112 `template<typename _Key, typename _Compare, typename _Alloc> bool std::operator< (const set< _Key, _Compare, _Alloc> & __x, const set< _Key, _Compare, _Alloc> & __y) [inline]`

Set ordering relation.

Parameters

$_x$	A set.
$_y$	A set of the same type as $_x$.

Returns

True iff $_x$ is lexicographically less than $_y$.

This is a total ordering relation. It is linear in the size of the sets. The elements must be comparable with $<$.

See `std::lexicographical_compare()` for how the determination is made.

Definition at line 810 of file `stl_set.h`.

```
3.11.4.113  template<typename _Key , typename _Tp , typename _Compare , typename _Alloc > bool std::operator< ( const
            multimap< _Key, _Tp, _Compare, _Alloc > & __x, const multimap< _Key, _Tp, _Compare, _Alloc > & __y )
            [inline]
```

Multimap ordering relation.

Parameters

$_x$	A multimap.
$_y$	A multimap of the same type as $_x$.

Returns

True iff x is lexicographically less than y .

This is a total ordering relation. It is linear in the size of the multimaps. The elements must be comparable with $<$.

See `std::lexicographical_compare()` for how the determination is made.

Definition at line 920 of file `stl_multimap.h`.

```
3.11.4.114  template<typename _Key , typename _Tp , typename _Compare , typename _Alloc > bool std::operator< ( const
            map< _Key, _Tp, _Compare, _Alloc > & __x, const map< _Key, _Tp, _Compare, _Alloc > & __y ) [inline]
```

Map ordering relation.

Parameters

$_x$	A map.
$_y$	A map of the same type as x .

Returns

True iff x is lexicographically less than y .

This is a total ordering relation. It is linear in the size of the maps. The elements must be comparable with $<$.

See `std::lexicographical_compare()` for how the determination is made.

Definition at line 1017 of file `stl_map.h`.

3.11.4.115 `template<typename _Tp, typename _Alloc> bool std::operator< (const forward_list< _Tp, _Alloc> & __lx, const forward_list< _Tp, _Alloc> & __ly) [inline]`

Forward list ordering relation.

Parameters

$_lx$	A <code>forward_list</code> .
$_ly$	A <code>forward_list</code> of the same type as $_lx$.

Returns

True iff $_lx$ is lexicographically less than $_ly$.

This is a total ordering relation. It is linear in the number of elements of the forward lists. The elements must be comparable with $<$.

See `std::lexicographical_compare()` for how the determination is made.

Definition at line 1365 of file `forward_list.h`.

References `std::forward_list< _Tp, _Alloc>::cbegin()`, `std::forward_list< _Tp, _Alloc>::cend()`, and `lexicographical_compare()`.

3.11.4.116 `template<typename _Tp, typename _Alloc> bool std::operator< (const vector< _Tp, _Alloc> & __x, const vector< _Tp, _Alloc> & __y) [inline]`

Vector ordering relation.

Parameters

$_x$	A vector.
$_y$	A vector of the same type as $_x$.

Returns

True iff $_x$ is lexicographically less than $_y$.

This is a total ordering relation. It is linear in the size of the vectors. The elements must be comparable with $<$.

See `std::lexicographical_compare()` for how the determination is made.

Definition at line 1528 of file `std_vector.h`.

References `lexicographical_compare()`.

3.11.4.117 `template<typename _Tp, typename _Alloc> bool std::operator< (const list< _Tp, _Alloc > & _x, const list< _Tp, _Alloc > & _y) [inline]`

List ordering relation.

Parameters

$_x$	A list.
$_y$	A list of the same type as $_x$.

Returns

True iff $_x$ is lexicographically less than $_y$.

This is a total ordering relation. It is linear in the size of the lists. The elements must be comparable with $<$.

See `std::lexicographical_compare()` for how the determination is made.

Definition at line 1758 of file `std_list.h`.

References `lexicographical_compare()`.

3.11.4.118 `template<typename _Tp, typename _Alloc> bool std::operator< (const deque< _Tp, _Alloc > & _x, const deque< _Tp, _Alloc > & _y) [inline]`

Deque ordering relation.

Parameters

<code>__x</code>	A deque.
<code>__y</code>	A deque of the same type as <code>__x</code> .

Returns

True iff `x` is lexicographically less than `__y`.

This is a total ordering relation. It is linear in the size of the deques. The elements must be comparable with `<`.

See `std::lexicographical_compare()` for how the determination is made.

Definition at line 2047 of file `std_deque.h`.

References `lexicographical_compare()`.

3.11.4.119 `template<typename _CharT, typename _Traits, typename _Alloc> bool std::operator< (const basic_string<_CharT, _Traits, _Alloc> & __lhs, const basic_string<_CharT, _Traits, _Alloc> & __rhs) [inline]`

Test if string precedes string.

Parameters

<code>__lhs</code>	First string.
<code>__rhs</code>	Second string.

Returns

True if `__lhs` precedes `__rhs`. False otherwise.

Definition at line 2587 of file `basic_string.h`.

References `std::basic_string<_CharT, _Traits, _Alloc>::compare()`.

3.11.4.120 `template<typename _CharT, typename _Traits, typename _Alloc> bool std::operator< (const basic_string<_CharT, _Traits, _Alloc> & __lhs, const _CharT* __rhs) [inline]`

Test if string precedes C string.

Parameters

<code>__lhs</code>	String.
<code>__rhs</code>	C string.

Returns

True if `__lhs` precedes `__rhs`. False otherwise.

Definition at line 2599 of file `basic_string.h`.

References `std::basic_string<_CharT, _Traits, _Alloc>::compare()`.

3.11.4.121 `template<typename _CharT, typename _Traits, typename _Alloc> bool std::operator< (const _CharT * __lhs, const basic_string<_CharT, _Traits, _Alloc> & __rhs) [inline]`

Test if C string precedes string.

Parameters

<code>__lhs</code>	C string.
<code>__rhs</code>	String.

Returns

True if `__lhs` precedes `__rhs`. False otherwise.

Definition at line 2611 of file `basic_string.h`.

References `std::basic_string<_CharT, _Traits, _Alloc>::compare()`.

3.11.4.122 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> basic_ostream<_CharT, _Traits>& std::operator<< (basic_ostream<_CharT, _Traits> & __os, const __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base> & __str) [inline]`

Write string to a stream.

Parameters

<code>__os</code>	Output stream.
<code>__str</code>	String to write out.

Returns

Reference to the output stream.

Output characters of `__str` into `os` following the same rules as for writing a C string.

Definition at line 2630 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::data()`, `getline()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

```
3.11.4.123 template<typename _CharT, typename _Traits, typename _Alloc> basic_ostream<_CharT, _Traits>&
std::operator<<( basic_ostream<_CharT, _Traits> & __os, const basic_string<_CharT, _Traits, _Alloc> & __str
) [inline]
```

Write string to a stream.

Parameters

<code>__os</code>	Output stream.
<code>__str</code>	String to write out.

Returns

Reference to the output stream.

Output characters of `__str` into `os` following the same rules as for writing a C string.

Definition at line 2771 of file `basic_string.h`.

References `std::basic_string<_CharT, _Traits, _Alloc>::data()`, `getline()`, and `std::basic_string<_CharT, _Traits, _Alloc>::size()`.

```
3.11.4.124 template<typename _Tp, typename _Seq> bool std::operator<= ( const stack<_Tp, _Seq> & __x, const stack<
_Tp, _Seq> & __y ) [inline]
```

Based on `operator<`.

Definition at line 282 of file `stl_stack.h`.

```
3.11.4.125 template<typename _Tp, typename _Seq> bool std::operator<= ( const queue<_Tp, _Seq> & __x, const
queue<_Tp, _Seq> & __y ) [inline]
```

Based on `operator<`.

Definition at line 307 of file `stl_queue.h`.

```
3.11.4.126 template<typename _Key, typename _Compare, typename _Alloc> bool std::operator<= ( const multiset<_Key,
_Compare, _Alloc> & __x, const multiset<_Key, _Compare, _Alloc> & __y ) [inline]
```

Returns `!(y < x)`

Definition at line 815 of file `stl_multiset.h`.

```
3.11.4.127 template<typename _Key, typename _Compare, typename _Alloc> bool std::operator<= ( const set<_Key,
_Compare, _Alloc> & __x, const set<_Key, _Compare, _Alloc> & __y ) [inline]
```

Returns `!(y < x)`

Definition at line 831 of file `stl_set.h`.

3.11.4.128 `template<typename _Key , typename _Tp , typename _Compare , typename _Alloc > bool std::operator<= (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y) [inline]`

Based on operator<.

Definition at line 941 of file stl_multimap.h.

3.11.4.129 `template<typename _Key , typename _Tp , typename _Compare , typename _Alloc > bool std::operator<= (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y) [inline]`

Based on operator<.

Definition at line 1038 of file stl_map.h.

3.11.4.130 `template<typename _Tp , typename _Alloc > bool std::operator<= (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly) [inline]`

Based on operator<.

Definition at line 1394 of file forward_list.h.

3.11.4.131 `template<typename _Tp , typename _Alloc > bool std::operator<= (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y) [inline]`

Based on operator<.

Definition at line 1547 of file stl_vector.h.

3.11.4.132 `template<typename _Tp , typename _Alloc > bool std::operator<= (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y) [inline]`

Based on operator<.

Definition at line 1777 of file stl_list.h.

3.11.4.133 `template<typename _Tp , typename _Alloc > bool std::operator<= (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc > &__y) [inline]`

Based on operator<.

Definition at line 2069 of file stl_deque.h.

3.11.4.134 `template<typename _CharT , typename _Traits , typename _Alloc > bool std::operator<= (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs) [inline]`

Test if string doesn't follow string.

Parameters

<code>__lhs</code>	First string.
<code>__rhs</code>	Second string.

Returns

True if `__lhs` doesn't follow `__rhs`. False otherwise.

Definition at line 2661 of file `basic_string.h`.

References `std::basic_string<_CharT, _Traits, _Alloc>::compare()`.

3.11.4.135 `template<typename _CharT, typename _Traits, typename _Alloc> bool std::operator<= (const basic_string<_CharT, _Traits, _Alloc> & __lhs, const _CharT* __rhs) [inline]`

Test if string doesn't follow C string.

Parameters

<code>__lhs</code>	String.
<code>__rhs</code>	C string.

Returns

True if `__lhs` doesn't follow `__rhs`. False otherwise.

Definition at line 2673 of file `basic_string.h`.

References `std::basic_string<_CharT, _Traits, _Alloc>::compare()`.

3.11.4.136 `template<typename _CharT, typename _Traits, typename _Alloc> bool std::operator<= (const _CharT* __lhs, const basic_string<_CharT, _Traits, _Alloc> & __rhs) [inline]`

Test if C string doesn't follow string.

Parameters

<code>__lhs</code>	C string.
<code>__rhs</code>	String.

Returns

True if `__lhs` doesn't follow `__rhs`. False otherwise.

Definition at line 2685 of file `basic_string.h`.

References `std::basic_string<_CharT, _Traits, _Alloc>::compare()`.

3.11.4.137 `template<typename _StateT> bool std::operator==(const fpos<_StateT> & __lhs, const fpos<_StateT> & __rhs) [inline]`

Test if equivalent to another position.

Definition at line 216 of file postypes.h.

3.11.4.138 `template<typename _Tp, typename _Seq> bool std::operator==(const stack<_Tp, _Seq> & __x, const stack<_Tp, _Seq> & __y) [inline]`

Stack equality comparison.

Parameters

<code>__x</code>	A stack.
<code>__y</code>	A stack of the same type as <code>__x</code> .

Returns

True iff the size and elements of the stacks are equal.

This is an equivalence relation. Complexity and semantics depend on the underlying sequence type, but the expected rules are: this relation is linear in the size of the sequences, and stacks are considered equivalent if their sequences compare equal.

Definition at line 246 of file stl_stack.h.

3.11.4.139 `template<typename _Tp> bool std::operator==(const _Fwd_list_iterator<_Tp> & __x, const _Fwd_list_const_iterator<_Tp> & __y) [inline], [noexcept]`

Forward list iterator equality comparison.

Definition at line 257 of file forward_list.h.

3.11.4.140 `template<typename _Tp, typename _Seq> bool std::operator==(const queue<_Tp, _Seq> & __x, const queue<_Tp, _Seq> & __y) [inline]`

Queue equality comparison.

Parameters

<code>__x</code>	A queue.
<code>__y</code>	A queue of the same type as <code>__x</code> .

Returns

True iff the size and elements of the queues are equal.

This is an equivalence relation. Complexity and semantics depend on the underlying sequence type, but the expected rules are: this relation is linear in the size of the sequences, and queues are considered equivalent if their sequences compare equal.

Definition at line 271 of file `stl_queue.h`.

References `std::queue<_Tp, _Sequence>::c`.

```
3.11.4.141 template<typename _Key , typename _Compare , typename _Alloc > bool std::operator==( const multiset<_Key,
    _Compare, _Alloc > &__x, const multiset<_Key, _Compare, _Alloc > &__y ) [inline]
```

Multiset equality comparison.

Parameters

$_x$	A multiset.
$_y$	A multiset of the same type as $_x$.

Returns

True iff the size and elements of the multisets are equal.

This is an equivalence relation. It is linear in the size of the multisets. Multisets are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

Definition at line 777 of file `stl_multiset.h`.

```
3.11.4.142 template<typename _Key , typename _Compare , typename _Alloc > bool std::operator==( const set<_Key,
    _Compare, _Alloc > &__x, const set<_Key, _Compare, _Alloc > &__y ) [inline]
```

Set equality comparison.

Parameters

x	A set.
y	A set of the same type as x .

Returns

True iff the size and elements of the sets are equal.

This is an equivalence relation. It is linear in the size of the sets. Sets are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

Definition at line 793 of file `stl_set.h`.

```
3.11.4.143 template<typename _Key , typename _Tp , typename _Compare , typename _Alloc > bool std::operator== ( const
    multimap< _Key, _Tp, _Compare, _Alloc > & __x, const multimap< _Key, _Tp, _Compare, _Alloc > & __y )
    [inline]
```

Multimap equality comparison.

Parameters

\leftrightarrow __x	A multimap.
\leftrightarrow __y	A multimap of the same type as __x.

Returns

True iff the size and elements of the maps are equal.

This is an equivalence relation. It is linear in the size of the multimaps. Multimaps are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

Definition at line 903 of file `stl_multimap.h`.

Referenced by `std::multimap< _Key, _Tp, _Compare, _Alloc >::equal_range()`.

```
3.11.4.144 template<typename _Key , typename _Tp , typename _Compare , typename _Alloc > bool std::operator== ( const
    map< _Key, _Tp, _Compare, _Alloc > & __x, const map< _Key, _Tp, _Compare, _Alloc > & __y ) [inline]
```

Map equality comparison.

Parameters

\leftrightarrow __x	A map.
\leftrightarrow __y	A map of the same type as x.

Returns

True iff the size and elements of the maps are equal.

This is an equivalence relation. It is linear in the size of the maps. Maps are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

Definition at line 1000 of file `stl_map.h`.

Referenced by `std::map< _Key, _Tp, _Compare, _Alloc >::equal_range()`.

3.11.4.145 `template<typename _Tp, typename _Alloc > bool std::operator==(const forward_list<_Tp, _Alloc > & __x, const forward_list<_Tp, _Alloc > & __y)`

Forward list equality comparison.

Parameters

<code>__x</code>	A <code>forward_list</code>
<code>__y</code>	A <code>forward_list</code> of the same type as <code>__x</code> .

Returns

True iff the elements of the forward lists are equal.

This is an equivalence relation. It is linear in the number of elements of the forward lists. Deques are considered equivalent if corresponding elements compare equal.

3.11.4.146 `template<typename _Tp, typename _Alloc > bool std::operator==(const vector<_Tp, _Alloc > & __x, const vector<_Tp, _Alloc > & __y) [inline]`

Vector equality comparison.

Parameters

<code>__x</code>	A vector.
<code>__y</code>	A vector of the same type as <code>__x</code> .

Returns

True iff the size and elements of the vectors are equal.

This is an equivalence relation. It is linear in the size of the vectors. Vectors are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

Definition at line 1511 of file `stl_vector.h`.

References `std::vector<_Tp, _Alloc >::begin()`, `std::vector<_Tp, _Alloc >::end()`, `equal()`, and `std::vector<_Tp, _Alloc >::size()`.

3.11.4.147 `template<typename _Tp, typename _Alloc > bool std::operator==(const list<_Tp, _Alloc > & __x, const list<_Tp, _Alloc > & __y) [inline]`

List equality comparison.

Parameters

$_x$	A list.
$_y$	A list of the same type as $_x$.

Returns

True iff the size and elements of the lists are equal.

This is an equivalence relation. It is linear in the size of the lists. Lists are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

Definition at line 1729 of file `stl_list.h`.

References `std::list<_Tp, _Alloc>::begin()`, and `std::list<_Tp, _Alloc>::end()`.

3.11.4.148 `template<typename _Tp, typename _Alloc> bool std::operator==(const deque<_Tp, _Alloc> &__x, const deque<_Tp, _Alloc> &__y) [inline]`

Deque equality comparison.

Parameters

$_x$	A deque.
$_y$	A deque of the same type as $_x$.

Returns

True iff the size and elements of the deques are equal.

This is an equivalence relation. It is linear in the size of the deques. Deques are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

Definition at line 2029 of file `stl_deque.h`.

References `std::deque<_Tp, _Alloc>::begin()`, `std::deque<_Tp, _Alloc>::end()`, `equal()`, and `std::deque<_Tp, _Alloc>::size()`.

3.11.4.149 `template<typename _CharT, typename _Traits, typename _Alloc> bool std::operator==(const basic_string<_CharT, _Traits, _Alloc> &__lhs, const basic_string<_CharT, _Traits, _Alloc> &__rhs) [inline]`

Test equivalence of two strings.

Parameters

<code>__lhs</code>	First string.
<code>__rhs</code>	Second string.

Returns

True if `__lhs.compare(__rhs) == 0`. False otherwise.

Definition at line 2504 of file `basic_string.h`.

References `std::basic_string<_CharT, _Traits, _Alloc>::compare()`, `std::basic_string<_CharT, _Traits, _Alloc>::data()`, and `std::basic_string<_CharT, _Traits, _Alloc>::size()`.

3.11.4.150 `template<typename _CharT, typename _Traits, typename _Alloc> bool std::operator==(const _CharT * __lhs, const basic_string<_CharT, _Traits, _Alloc> & __rhs) [inline]`

Test equivalence of C string and string.

Parameters

<code>__lhs</code>	C string.
<code>__rhs</code>	String.

Returns

True if `__rhs.compare(__lhs) == 0`. False otherwise.

Definition at line 2525 of file `basic_string.h`.

References `std::basic_string<_CharT, _Traits, _Alloc>::compare()`.

3.11.4.151 `template<typename _CharT, typename _Traits, typename _Alloc> bool std::operator==(const basic_string<_CharT, _Traits, _Alloc> & __lhs, const _CharT * __rhs) [inline]`

Test equivalence of string and C string.

Parameters

<code>__lhs</code>	String.
<code>__rhs</code>	C string.

Returns

True if `__lhs.compare(__rhs) == 0`. False otherwise.

Definition at line 2537 of file basic_string.h.

References `std::basic_string<_CharT, _Traits, _Alloc>::compare()`.

3.11.4.152 `template<typename _Tp, typename _Seq> bool std::operator> (const stack<_Tp, _Seq> &__x, const stack<_Tp, _Seq> &__y) [inline]`

Based on `operator<`.

Definition at line 276 of file stl_stack.h.

3.11.4.153 `template<typename _Tp, typename _Seq> bool std::operator> (const queue<_Tp, _Seq> &__x, const queue<_Tp, _Seq> &__y) [inline]`

Based on `operator<`.

Definition at line 301 of file stl_queue.h.

3.11.4.154 `template<typename _Key, typename _Compare, typename _Alloc> bool std::operator> (const multiset<_Key, _Compare, _Alloc> &__x, const multiset<_Key, _Compare, _Alloc> &__y) [inline]`

Returns $y < x$.

Definition at line 808 of file stl_multiset.h.

3.11.4.155 `template<typename _Key, typename _Compare, typename _Alloc> bool std::operator> (const set<_Key, _Compare, _Alloc> &__x, const set<_Key, _Compare, _Alloc> &__y) [inline]`

Returns $y < x$.

Definition at line 824 of file stl_set.h.

3.11.4.156 `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc> bool std::operator> (const multimap<_Key, _Tp, _Compare, _Alloc> &__x, const multimap<_Key, _Tp, _Compare, _Alloc> &__y) [inline]`

Based on `operator<`.

Definition at line 934 of file stl_multimap.h.

3.11.4.157 `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc> bool std::operator> (const map<_Key, _Tp, _Compare, _Alloc> &__x, const map<_Key, _Tp, _Compare, _Alloc> &__y) [inline]`

Based on `operator<`.

Definition at line 1031 of file stl_map.h.

3.11.4.158 `template<typename _Tp, typename _Alloc > bool std::operator> (const forward_list<_Tp, _Alloc> & __x, const forward_list<_Tp, _Alloc> & __y) [inline]`

Based on operator<.

Definition at line 1380 of file forward_list.h.

3.11.4.159 `template<typename _Tp, typename _Alloc > bool std::operator> (const vector<_Tp, _Alloc> & __x, const vector<_Tp, _Alloc> & __y) [inline]`

Based on operator<.

Definition at line 1541 of file stl_vector.h.

3.11.4.160 `template<typename _Tp, typename _Alloc > bool std::operator> (const list<_Tp, _Alloc> & __x, const list<_Tp, _Alloc> & __y) [inline]`

Based on operator<.

Definition at line 1771 of file stl_list.h.

3.11.4.161 `template<typename _Tp, typename _Alloc > bool std::operator> (const deque<_Tp, _Alloc> & __x, const deque<_Tp, _Alloc> & __y) [inline]`

Based on operator<.

Definition at line 2062 of file stl_deque.h.

3.11.4.162 `template<typename _CharT, typename _Traits, typename _Alloc > bool std::operator> (const basic_string<_CharT, _Traits, _Alloc> & __lhs, const basic_string<_CharT, _Traits, _Alloc> & __rhs) [inline]`

Test if string follows string.

Parameters

<code>__lhs</code>	First string.
<code>__rhs</code>	Second string.

Returns

True if `__lhs` follows `__rhs`. False otherwise.

Definition at line 2624 of file basic_string.h.

References `std::basic_string<_CharT, _Traits, _Alloc>::compare()`.

Referenced by `std::_Deque_iterator<_Tp, _Tp &, _Tp * >::_M_set_node()`, `inserter()`, `operator==()`, `std::shared_ptr<__detail::_NFA<_Rx_traits> >::shared_ptr()`, and `std::unique_ptr<_Tp[], _Dp>::swap()`.

3.11.4.163 `template<typename _CharT, typename _Traits, typename _Alloc> bool std::operator> (const basic_string<_CharT, _Traits, _Alloc> & __lhs, const _CharT* __rhs) [inline]`

Test if string follows C string.

Parameters

<code>__lhs</code>	String.
<code>__rhs</code>	C string.

Returns

True if `__lhs` follows `__rhs`. False otherwise.

Definition at line 2636 of file `basic_string.h`.

References `std::basic_string<_CharT, _Traits, _Alloc>::compare()`.

3.11.4.164 `template<typename _CharT, typename _Traits, typename _Alloc> bool std::operator> (const _CharT* __lhs, const basic_string<_CharT, _Traits, _Alloc> & __rhs) [inline]`

Test if C string follows string.

Parameters

<code>__lhs</code>	C string.
<code>__rhs</code>	String.

Returns

True if `__lhs` follows `__rhs`. False otherwise.

Definition at line 2648 of file `basic_string.h`.

References `std::basic_string<_CharT, _Traits, _Alloc>::compare()`.

3.11.4.165 `template<typename _Tp, typename _Seq> bool std::operator>= (const stack<_Tp, _Seq> & __x, const stack<_Tp, _Seq> & __y) [inline]`

Based on `operator<`.

Definition at line 288 of file `stl_stack.h`.

3.11.4.166 `template<typename _Tp, typename _Seq> bool std::operator>= (const queue<_Tp, _Seq> & __x, const queue<_Tp, _Seq> & __y) [inline]`

Based on `operator<`.

Definition at line 313 of file `stl_queue.h`.

3.11.4.167 `template<typename _Key , typename _Compare , typename _Alloc > bool std::operator>= (const multiset< _Key, _Compare, _Alloc > & __x, const multiset< _Key, _Compare, _Alloc > & __y) [inline]`

Returns `!(x < y)`

Definition at line 822 of file `stl_multiset.h`.

3.11.4.168 `template<typename _Key , typename _Compare , typename _Alloc > bool std::operator>= (const set< _Key, _Compare, _Alloc > & __x, const set< _Key, _Compare, _Alloc > & __y) [inline]`

Returns `!(x < y)`

Definition at line 838 of file `stl_set.h`.

3.11.4.169 `template<typename _Key , typename _Tp , typename _Compare , typename _Alloc > bool std::operator>= (const multimap< _Key, _Tp, _Compare, _Alloc > & __x, const multimap< _Key, _Tp, _Compare, _Alloc > & __y) [inline]`

Based on `operator<`.

Definition at line 948 of file `stl_multimap.h`.

3.11.4.170 `template<typename _Key , typename _Tp , typename _Compare , typename _Alloc > bool std::operator>= (const map< _Key, _Tp, _Compare, _Alloc > & __x, const map< _Key, _Tp, _Compare, _Alloc > & __y) [inline]`

Based on `operator<`.

Definition at line 1045 of file `stl_map.h`.

3.11.4.171 `template<typename _Tp , typename _Alloc > bool std::operator>= (const forward_list< _Tp, _Alloc > & __x, const forward_list< _Tp, _Alloc > & __y) [inline]`

Based on `operator<`.

Definition at line 1387 of file `forward_list.h`.

3.11.4.172 `template<typename _Tp , typename _Alloc > bool std::operator>= (const vector< _Tp, _Alloc > & __x, const vector< _Tp, _Alloc > & __y) [inline]`

Based on `operator<`.

Definition at line 1553 of file `stl_vector.h`.

3.11.4.173 `template<typename _Tp , typename _Alloc > bool std::operator>= (const list< _Tp, _Alloc > & __x, const list< _Tp, _Alloc > & __y) [inline]`

Based on `operator<`.

Definition at line 1783 of file `stl_list.h`.

3.11.4.174 `template<typename _Tp , typename _Alloc > bool std::operator>= (const deque< _Tp, _Alloc > & __x, const deque< _Tp, _Alloc > & __y) [inline]`

Based on `operator<`.

Definition at line 2076 of file `stl_deque.h`.

3.11.4.175 `template<typename _CharT , typename _Traits , typename _Alloc > bool std::operator>= (const basic_string< _CharT, _Traits, _Alloc > & __lhs, const basic_string< _CharT, _Traits, _Alloc > & __rhs) [inline]`

Test if string doesn't precede string.

Parameters

<code>__lhs</code>	First string.
<code>__rhs</code>	Second string.

Returns

True if `__lhs` doesn't precede `__rhs`. False otherwise.

Definition at line 2698 of file `basic_string.h`.

References `std::basic_string<_CharT, _Traits, _Alloc>::compare()`.

Referenced by `std::_Deque_iterator<_Tp, _Tp&, _Tp*>::M_set_node()`, `inserter()`, `operator==()`, `std::shared_ptr<__detail::_NFA<_Rx_traits>>::shared_ptr()`, and `std::unique_ptr<_Tp[], _Dp>::swap()`.

3.11.4.176 `template<typename _CharT, typename _Traits, typename _Alloc> bool std::operator>= (const basic_string<_CharT, _Traits, _Alloc> &__lhs, const _CharT* __rhs) [inline]`

Test if string doesn't precede C string.

Parameters

<code>__lhs</code>	String.
<code>__rhs</code>	C string.

Returns

True if `__lhs` doesn't precede `__rhs`. False otherwise.

Definition at line 2710 of file `basic_string.h`.

References `std::basic_string<_CharT, _Traits, _Alloc>::compare()`.

3.11.4.177 `template<typename _CharT, typename _Traits, typename _Alloc> bool std::operator>= (const _CharT* __lhs, const basic_string<_CharT, _Traits, _Alloc> &__rhs) [inline]`

Test if C string doesn't precede string.

Parameters

<code>__lhs</code>	C string.
<code>__rhs</code>	String.

Returns

True if `__lhs` doesn't precede `__rhs`. False otherwise.

Definition at line 2722 of file `basic_string.h`.

References `std::basic_string<_CharT, _Traits, _Alloc>::compare()`.

3.11.4.178 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> basic_istream<_CharT, _Traits>& std::operator>> (basic_istream<_CharT, _Traits> & __is, __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base> & __str)`

Read stream into a string.

Parameters

<code>__is</code>	Input stream.
<code>__str</code>	Buffer to store into.

Returns

Reference to the input stream.

Stores characters from `__is` into `__str` until whitespace is found, the end of the stream is encountered, or `str.max_size()` is reached. If `is.width()` is non-zero, that is the limit on the number of characters stored into `__str`. Any previous contents of `__str` are erased.

Referenced by `__gnu_cxx::swap()`.

3.11.4.179 `template<typename _CharT, typename _Traits, typename _Alloc> basic_istream<_CharT, _Traits>& std::operator>> (basic_istream<_CharT, _Traits> & __is, basic_string<_CharT, _Traits, _Alloc> & __str)`

Read stream into a string.

Parameters

<code>__is</code>	Input stream.
<code>__str</code>	Buffer to store into.

Returns

Reference to the input stream.

Stores characters from `__is` into `__str` until whitespace is found, the end of the stream is encountered, or `str.max_size()` is reached. If `is.width()` is non-zero, that is the limit on the number of characters stored into `__str`. Any previous contents of `__str` are erased.

Referenced by `operator!==()`, `std::normal_distribution<result_type>::operator()()`, and `swap()`.

3.11.4.180 `template<typename _InputIterator, typename _OutputIterator > _OutputIterator std::partial_sum (_InputIterator __first, _InputIterator __last, _OutputIterator __result)`

Return list of partial sums.

Accumulates the values in the range [first,last) using the + operator. As each successive input value is added into the total, that partial sum is written to `__result`. Therefore, the first value in `__result` is the first value of the input, the second value in `__result` is the sum of the first and second input values, and so on.

Parameters

<code>__first</code>	Start of input range.
<code>__last</code>	End of input range.
<code>__result</code>	Output sum.

Returns

Iterator pointing just beyond the values written to `__result`.

Definition at line 237 of file `stl_numeric.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs_pu()`, and `__gnu_parallel::__sequential_random_shuffle()`.

3.11.4.181 `template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation > _OutputIterator std::partial_sum (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryOperation __binary_op)`

Return list of partial sums.

Accumulates the values in the range [first,last) using `__binary_op`. As each successive input value is added into the total, that partial sum is written to `__result`. Therefore, the first value in `__result` is the first value of the input, the second value in `__result` is the sum of the first and second input values, and so on.

Parameters

<code>__first</code>	Start of input range.
<code>__last</code>	End of input range.
<code>__result</code>	Output sum.
<code>__binary_op</code>	Function object.

Returns

Iterator pointing just beyond the values written to `__result`.

Definition at line 278 of file `stl_numeric.h`.

```
3.11.4.182 template<typename _InputIterator, typename _OutputIterator, typename _Tp > _OutputIterator std::replace_copy (
    _InputIterator __first, _InputIterator __last, _OutputIterator __result, const _Tp & __old_value, const _Tp & __new_value
) [inline]
```

Copy a sequence, replacing each element of one value with another value.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__result</code>	An output iterator.
<code>__old_value</code>	The value to be replaced.
<code>__new_value</code>	The replacement value.

Returns

The end of the output sequence, `result+(last-first)`.

Copies each element in the input range [`__first`,`__last`) to the output range [`__result`,`__result+(__last-__first)`) replacing elements equal to `__old_value` with `__new_value`.

Definition at line 3135 of file `stl_algo.h`.

```
3.11.4.183 template<typename _Tp > void std::return_temporary_buffer ( _Tp * __p ) [inline]
```

The companion to `get_temporary_buffer()`.

Parameters

<code>__p</code>	A buffer previously allocated by <code>get_temporary_buffer</code> .
------------------	--

Returns

None.

Frees the memory pointed to by `__p`.

Definition at line 112 of file `stl_tempbuf.h`.

Referenced by `std::_Temporary_buffer<_ForwardIterator, _Tp >::_Temporary_buffer()`, and `std::_Temporary_buffer<_ForwardIterator, _Tp >::end()`.

```
3.11.4.184 ios_base& std::right ( ios_base & __base ) [inline]
```

Calls `base.setf(ios_base::right, ios_base::adjustfield)`.

Definition at line 924 of file `ios_base.h`.

References `__gnu_debug::__base()`, `std::ios_base::adjustfield`, `std::ios_base::right`, and `std::ios_base::setf()`.

3.11.4.185 ios_base& std::scientific (ios_base & __base) [inline]

Calls base.setf(ios_base::scientific, ios_base::floatfield).

Definition at line 966 of file ios_base.h.

References __gnu_debug::__base(), std::ios_base::floatfield, std::ios_base::scientific, and std::ios_base::setf().

3.11.4.186 ios_base& std::showbase (ios_base & __base) [inline]

Calls base.setf(ios_base::showbase).

Definition at line 811 of file ios_base.h.

References __gnu_debug::__base(), std::ios_base::setf(), and std::ios_base::showbase.

3.11.4.187 ios_base& std::showpoint (ios_base & __base) [inline]

Calls base.setf(ios_base::showpoint).

Definition at line 827 of file ios_base.h.

References __gnu_debug::__base(), std::ios_base::setf(), and std::ios_base::showpoint.

3.11.4.188 ios_base& std::showpos (ios_base & __base) [inline]

Calls base.setf(ios_base::showpos).

Definition at line 843 of file ios_base.h.

References __gnu_debug::__base(), std::ios_base::setf(), and std::ios_base::showpos.

3.11.4.189 ios_base& std::skipws (ios_base & __base) [inline]

Calls base.setf(ios_base::skipws).

Definition at line 859 of file ios_base.h.

References __gnu_debug::__base(), std::ios_base::setf(), and std::ios_base::skipws.

3.11.4.190 template<typename _Tp, typename _Tp1, _Lock_policy _Lp> __shared_ptr<_Tp, _Lp> std::static_pointer_cast (const __shared_ptr<_Tp1, _Lp> & __r) [inline], [noexcept]

static_pointer_cast

Definition at line 1310 of file shared_ptr_base.h.

3.11.4.191 `template<typename _Key , typename _Compare , typename _Alloc > void std::swap (multiset< _Key, _Compare, _Alloc > & __x, multiset< _Key, _Compare, _Alloc > & __y) [inline]`

See `std::multiset::swap()`.

Definition at line 829 of file `stl_multiset.h`.

References `std::multiset< _Key, _Compare, _Alloc >::swap()`.

3.11.4.192 `template<typename _Key , typename _Compare , typename _Alloc > void std::swap (set< _Key, _Compare, _Alloc > & __x, set< _Key, _Compare, _Alloc > & __y) [inline]`

See `std::set::swap()`.

Definition at line 845 of file `stl_set.h`.

References `std::set< _Key, _Compare, _Alloc >::swap()`.

3.11.4.193 `template<typename _Key , typename _Tp , typename _Compare , typename _Alloc > void std::swap (multimap< _Key, _Tp, _Compare, _Alloc > & __x, multimap< _Key, _Tp, _Compare, _Alloc > & __y) [inline]`

See `std::multimap::swap()`.

Definition at line 955 of file `stl_multimap.h`.

References `std::multimap< _Key, _Tp, _Compare, _Alloc >::swap()`.

3.11.4.194 `template<typename _Key , typename _Tp , typename _Compare , typename _Alloc > void std::swap (map< _Key, _Tp, _Compare, _Alloc > & __x, map< _Key, _Tp, _Compare, _Alloc > & __y) [inline]`

See `std::map::swap()`.

Definition at line 1052 of file `stl_map.h`.

References `std::map< _Key, _Tp, _Compare, _Alloc >::swap()`.

3.11.4.195 `template<typename _Tp , typename _Alloc > void std::swap (forward_list< _Tp, _Alloc > & __lx, forward_list< _Tp, _Alloc > & __ly) [inline]`

See `std::forward_list::swap()`.

Definition at line 1401 of file `forward_list.h`.

References `std::forward_list< _Tp, _Alloc >::swap()`.

3.11.4.196 `template<typename _Tp , typename _Alloc > void std::swap (vector< _Tp, _Alloc > & __x, vector< _Tp, _Alloc > & __y) [inline]`

See `std::vector::swap()`.

Definition at line 1559 of file `stl_vector.h`.

References `std::vector< _Tp, _Alloc >::swap()`.

3.11.4.197 `template<typename _Tp, typename _Alloc > void std::swap (list<_Tp, _Alloc > &__x, list<_Tp, _Alloc > &__y) [inline]`

See `std::list::swap()`.

Definition at line 1789 of file `stl_list.h`.

References `swap()`, and `std::list<_Tp, _Alloc >::swap()`.

3.11.4.198 `template<typename _Tp, typename _Alloc > void std::swap (deque<_Tp, _Alloc > &__x, deque<_Tp, _Alloc > &__y) [inline]`

See `std::deque::swap()`.

Definition at line 2083 of file `stl_deque.h`.

References `std::deque<_Tp, _Alloc >::swap()`.

3.11.4.199 `template<typename _CharT, typename _Traits, typename _Alloc > void std::swap (basic_string<_CharT, _Traits, _Alloc > &__lhs, basic_string<_CharT, _Traits, _Alloc > &__rhs) [inline]`

Swap contents of two strings.

Parameters

<code>__lhs</code>	First string.
<code>__rhs</code>	Second string.

Exchanges the contents of `__lhs` and `__rhs` in constant time.

Definition at line 2735 of file `basic_string.h`.

References `operator>>()`, and `std::basic_string<_CharT, _Traits, _Alloc >::swap()`.

3.11.4.200 `template<typename _CharT > _CharT std::tolower (_CharT __c, const locale &__loc) [inline]`

Convenience interface to `ctype::tolower(__c)`.

Definition at line 2602 of file `locale_facets.h`.

References `std::__ctype_abstract_base<_CharT >::tolower()`.

3.11.4.201 `template<typename _CharT > _CharT std::toupper (_CharT __c, const locale &__loc) [inline]`

Convenience interface to `ctype::toupper(__c)`.

Definition at line 2596 of file `locale_facets.h`.

References `std::__ctype_abstract_base<_CharT >::toupper()`.

3.11.4.202 `template<typename _InputIterator, typename _ForwardIterator > _ForwardIterator std::uninitialized_copy (_InputIterator __first, _InputIterator __last, _ForwardIterator __result) [inline]`

Copies the range `[first,last)` into `result`.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__result</code>	An output iterator.

Returns

`__result + (__first - __last)`

Like `copy()`, but does not require an initialized output range.

Definition at line 107 of file `stl_uninitialized.h`.

References `__addressof()`, `_Construct()`, and `_Destroy()`.

Referenced by `__gnu_parallel::parallel_sort_mwms_pu()`, and `uninitialized_fill_n()`.

3.11.4.203 `template<typename _InputIterator, typename _Size, typename _ForwardIterator> _ForwardIterator
std::uninitialized_copy_n(_InputIterator __first, _Size __n, _ForwardIterator __result) [inline]`

Copies the range `[first,first+n)` into `result`.

Parameters

<code>__first</code>	An input iterator.
<code>__n</code>	The number of elements to copy.
<code>__result</code>	An output iterator.

Returns

`__result + __n`

Like `copy_n()`, but does not require an initialized output range.

Definition at line 674 of file `stl_uninitialized.h`.

References `__iterator_category()`.

3.11.4.204 `template<typename _ForwardIterator, typename _Tp> void std::uninitialized_fill(_ForwardIterator __first,
_ForwardIterator __last, const _Tp & __x) [inline]`

Copies the value `x` into the range `[first,last)`.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__x</code>	The source value.

Returns

Nothing.

Like fill(), but does not require an initialized output range.

Definition at line 173 of file stl_uninitialized.h.

References __addressof(), _Construct(), _Destroy(), and fill_n().

Referenced by uninitialized_fill_n().

3.11.4.205 `template<typename _ForwardIterator, typename _Size, typename _Tp> void std::uninitialized_fill_n (_ForwardIterator
__first, _Size __n, const _Tp & __x) [inline]`

Copies the value x into the range [first,first+n).

Parameters

<code>__first</code>	An input iterator.
<code>__n</code>	The number of copies to make.
<code>__x</code>	The source value.

Returns

Nothing.

Like fill_n(), but does not require an initialized output range.

Definition at line 233 of file stl_uninitialized.h.

References __addressof(), _Construct(), _Destroy(), fill_n(), uninitialized_copy(), and uninitialized_fill().

3.11.4.206 `ios_base& std::unitbuf (ios_base & __base) [inline]`

Calls base.setf(ios_base::unitbuf).

Definition at line 891 of file ios_base.h.

References __gnu_debug::__base(), std::ios_base::setf(), and std::ios_base::unitbuf.

3.11.4.207 `ios_base& std::uppercase (ios_base & __base) [inline]`

Calls base.setf(ios_base::uppercase).

Definition at line 875 of file ios_base.h.

References __gnu_debug::__base(), std::ios_base::setf(), and std::ios_base::uppercase.

3.12 std::__debug Namespace Reference

Classes

- class [map](#)
- class [multimap](#)
- class [multiset](#)
- class [set](#)

Functions

- `template<typename _Key, typename _Compare, typename _Allocator >
bool operator!= (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >
bool operator!= (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >
bool operator!= (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >
bool operator!= (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >
bool operator< (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >
bool operator< (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >
bool operator< (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >
bool operator< (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >
bool operator<= (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >
bool operator<= (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >
bool operator<= (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >
bool operator<= (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >
bool operator== (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`

- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator== (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator== (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator== (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator> (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator> (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator> (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator> (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator>= (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator>= (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator>= (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator>= (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`void swap (multiset< _Key, _Compare, _Allocator > &__x, multiset< _Key, _Compare, _Allocator > &__y)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`void swap (set< _Key, _Compare, _Allocator > &__x, set< _Key, _Compare, _Allocator > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`void swap (multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`void swap (map< _Key, _Tp, _Compare, _Allocator > &__lhs, map< _Key, _Tp, _Compare, _Allocator > &__rhs)`

3.12.1 Detailed Description

GNU debug code, replaces standard behavior with debug behavior.

Macros and namespaces used by the implementation outside of debug wrappers to verify certain properties. The `__glibcxx_requires_xxx` macros are merely wrappers around the `__glibcxx_check_xxx` wrappers when we are compiling with debug mode, but disappear when we are in release mode so that there is no checking performed in, e.g., the standard library algorithms.

3.13 std::__detail Namespace Reference

Classes

- struct [_BracketMatcher](#)
- class [_Compiler](#)
- struct [_Default_ranged_hash](#)
- struct [_Equal_helper](#)
- struct [_Equal_helper< _Key, _Value, _ExtractKey, _Equal, _HashCodeType, false >](#)
- struct [_Equal_helper< _Key, _Value, _ExtractKey, _Equal, _HashCodeType, true >](#)
- struct [_Equality](#)
- struct [_Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false >](#)
- struct [_Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true >](#)
- struct [_Equality_base](#)
- class [_Executor](#)
- struct [_Hash_code_base](#)
- struct [_Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Default_ranged_hash, false >](#)
- struct [_Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Default_ranged_hash, true >](#)
- struct [_Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, false >](#)
- struct [_Hash_node](#)
- struct [_Hash_node< _Value, false >](#)
- struct [_Hash_node< _Value, true >](#)
- struct [_Hash_node_base](#)
- struct [_Hash_node_value_base](#)
- struct [_Hashtable_alloc](#)
- struct [_Hashtable_base](#)
- struct [_Hashtable_ebo_helper](#)
- struct [_Hashtable_ebo_helper< _Nm, _Tp, false >](#)
- struct [_Hashtable_ebo_helper< _Nm, _Tp, true >](#)
- struct [_Hashtable_traits](#)
- struct [_Insert](#)
- struct [_Insert< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false, \[_Unique_keys\]\(#\) >](#)
- struct [_Insert< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true, false >](#)
- struct [_Insert< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true, true >](#)
- struct [_Insert_base](#)
- struct [_List_node_base](#)
- struct [_Local_const_iterator](#)
- struct [_Local_iterator](#)
- struct [_Local_iterator_base](#)
- struct [_Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, true >](#)
- struct [_Map_base](#)
- struct [_Map_base< _Key, _Pair, _Alloc, _Select1st, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false >](#)
- struct [_Map_base< _Key, _Pair, _Alloc, _Select1st, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true >](#)
- struct [_Mod_range_hashing](#)
- struct [_Node_const_iterator](#)
- struct [_Node_iterator](#)
- struct [_Node_iterator_base](#)
- struct [_Prime_rehash_policy](#)
- struct [_Rehash_base](#)
- struct [_Rehash_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _Prime_rehash_policy, \[_Traits\]\(#\) >](#)
- class [_Scanner](#)
- class [_StateSeq](#)

Typedefs

- `template<typename _Key, typename _Value, typename _ExtractKey, typename _H1, typename _H2, typename _Hash >`
`using __hash_code_for_local_iter = _Hash_code_storage< _Hash_code_base< _Key, _Value, _ExtractKey,`
`_H1, _H2, _Hash, false >>`
- `template<typename _CharT >`
`using _Matcher = std::function< bool(_CharT)>`
- `typedef long _StateldT`

Enumerations

- `enum _Opcode : int {`
`_S_opcode_unknown, _S_opcode_alternative, _S_opcode_backref, _S_opcode_line_begin_assertion,`
`_S_opcode_line_end_assertion, _S_opcode_word_boundary, _S_opcode_subexpr_lookahead, _S_↵`
`opcode_subexpr_begin,`
`_S_opcode_subexpr_end, _S_opcode_dummy, _S_opcode_match, _S_opcode_accept }`
- `enum _RegexExecutorPolicy : int { _S_auto, _S_alterate }`

Functions

- `template<typename _TraitsT >`
`std::shared_ptr< _NFA< _TraitsT > > __compile_nfa (const typename _TraitsT::char_type * __first, const type-`
`name _TraitsT::char_type * __last, const _TraitsT & __traits, regex_constants::syntax_option_type __flags)`
- `template<class _Iterator >`
`std::iterator_traits< _Iterator >::difference_type __distance_fw (_Iterator __first, _Iterator __last, std::input_↵`
`iterator_tag)`
- `template<class _Iterator >`
`std::iterator_traits< _Iterator >::difference_type __distance_fw (_Iterator __first, _Iterator __last, std::forward_↵`
`_iterator_tag)`
- `template<class _Iterator >`
`std::iterator_traits< _Iterator >::difference_type __distance_fw (_Iterator __first, _Iterator __last)`
- `template<typename _Bilter, typename _Alloc, typename _CharT, typename _TraitsT, _RegexExecutorPolicy __policy, bool __match_↵`
`mode>`
`bool __regex_algo_impl (_Bilter __s, _Bilter __e, match_results< _Bilter, _Alloc > & __m, const basic_regex<`
`_CharT, _TraitsT > & __re, regex_constants::match_flag_type __flags)`
- `template<typename _Value, bool _Cache_hash_code>`
`bool operator!= (const _Node_iterator_base< _Value, _Cache_hash_code > & __x, const _Node_iterator_↵`
`base< _Value, _Cache_hash_code > & __y) noexcept`
- `template<typename _Key, typename _Value, typename _ExtractKey, typename _H1, typename _H2, typename _Hash, bool __cache>`
`bool operator!= (const _Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache > & __x,`
`const _Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache > & __y)`
- `template<typename _Value, bool _Cache_hash_code>`
`bool operator== (const _Node_iterator_base< _Value, _Cache_hash_code > & __x, const _Node_iterator_↵`
`base< _Value, _Cache_hash_code > & __y) noexcept`
- `template<typename _Key, typename _Value, typename _ExtractKey, typename _H1, typename _H2, typename _Hash, bool __cache>`
`bool operator== (const _Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache > & __x,`
`const _Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache > & __y)`

Variables

- `static const _StateldT _S_invalid_state_id`

3.13.1 Detailed Description

Implementation details not part of the namespace std interface.

3.14 std::__parallel Namespace Reference

Classes

- struct [_CRandNumber](#)

Functions

- `template<typename _Iter, typename _Tp, typename _Tag >
_Tp accumulate_switch (_Iter, _Iter, _Tp, _Tag)`
- `template<typename _Iter, typename _Tp, typename _BinaryOper, typename _Tag >
_Tp accumulate_switch (_Iter, _Iter, _Tp, _BinaryOper, _Tag)`
- `template<typename _RAlter, typename _Tp, typename _BinaryOper >
_Tp accumulate_switch (_RAlter, _RAlter, _Tp, _BinaryOper, random_access_iterator_tag, __gnu_parallel::Parallelism __parallelism=__gnu_parallel::parallel_unbalanced)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper, typename _Tag1, typename _Tag2 >
_OIter adjacent_difference_switch (_Iter, _Iter, _OIter, _BinaryOper, _Tag1, _Tag2)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper >
_OIter adjacent_difference_switch (_Iter, _Iter, _OIter, _BinaryOper, random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::Parallelism __parallelism=__gnu_parallel::parallel_unbalanced)`
- `template<typename _Filter, typename _IterTag >
_Filter adjacent_find_switch (_Filter, _Filter, _IterTag)`
- `template<typename _Filter, typename _BiPredicate, typename _IterTag >
_Filter adjacent_find_switch (_Filter, _Filter, _BiPredicate, _IterTag)`
- `template<typename _RAlter, typename _BiPredicate >
_RAlter adjacent_find_switch (_RAlter, _RAlter, _BiPredicate, random_access_iterator_tag)`
- `template<typename _RAlter >
_RAlter adjacent_find_switch (_RAlter __begin, _RAlter __end, random_access_iterator_tag)`
- `template<typename _FIterator, typename _IteratorTag >
_FIterator adjacent_find_switch (_FIterator __begin, _FIterator __end, _IteratorTag)`
- `template<typename _FIterator, typename _BinaryPredicate, typename _IteratorTag >
_FIterator adjacent_find_switch (_FIterator __begin, _FIterator __end, _BinaryPredicate __pred, _IteratorTag)`
- `template<typename _RAlter, typename _BinaryPredicate >
_RAlter adjacent_find_switch (_RAlter __begin, _RAlter __end, _BinaryPredicate __pred, random_access_iterator_tag)`
- `template<typename _Iter, typename _Predicate, typename _IterTag >
iterator_traits< _Iter >::difference_type count_if_switch (_Iter, _Iter, _Predicate, _IterTag)`
- `template<typename _RAlter, typename _Predicate >
iterator_traits< _RAlter >::difference_type count_if_switch (_RAlter __begin, _RAlter __end, _Predicate __pred, random_access_iterator_tag, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Predicate, typename _IteratorTag >
iterator_traits< _Iter >::difference_type count_if_switch (_Iter __begin, _Iter __end, _Predicate __pred, _IteratorTag)`

- `template<typename _Iter, typename _Tp, typename _IterTag >`
`iterator_traits< _Iter >::difference_type __count_switch (_Iter, _Iter, const _Tp &, _IterTag)`
- `template<typename _RAIter, typename _Tp >`
`iterator_traits< _RAIter >::difference_type __count_switch (_RAIter __begin, _RAIter __end, const _Tp & __value, random_access_iterator_tag, gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Tp, typename _IteratorTag >`
`iterator_traits< _Iter >::difference_type __count_switch (_Iter __begin, _Iter __end, const _Tp & __value, __IteratorTag)`
- `template<typename _Iter, typename _Filter, typename _IterTag1, typename _IterTag2 >`
`_Iter __find_first_of_switch (_Iter, _Iter, _Filter, _Filter, _IterTag1, _IterTag2)`
- `template<typename _RAIter, typename _Filter, typename _BiPredicate, typename _IterTag >`
`_RAIter __find_first_of_switch (_RAIter, _RAIter, _Filter, _Filter, _BiPredicate, random_access_iterator_tag, __IteratorTag)`
- `template<typename _Iter, typename _Filter, typename _BiPredicate, typename _IterTag1, typename _IterTag2 >`
`_Iter __find_first_of_switch (_Iter, _Iter, _Filter, _Filter, _BiPredicate, _IterTag1, _IterTag2)`
- `template<typename _Iter, typename _FIterator, typename _IteratorTag1, typename _IteratorTag2 >`
`_Iter __find_first_of_switch (_Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator __end2, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter, typename _FIterator, typename _BinaryPredicate, typename _IteratorTag >`
`_RAIter __find_first_of_switch (_RAIter __begin1, _RAIter __end1, _FIterator __begin2, _FIterator __end2, _BinaryPredicate __comp, random_access_iterator_tag, _IteratorTag)`
- `template<typename _Iter, typename _FIterator, typename _BinaryPredicate, typename _IteratorTag1, typename _IteratorTag2 >`
`_Iter __find_first_of_switch (_Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator __end2, _BinaryPredicate __comp, _IteratorTag1, _IteratorTag2)`
- `template<typename _Iter, typename _Predicate, typename _IterTag >`
`_Iter __find_if_switch (_Iter, _Iter, _Predicate, _IterTag)`
- `template<typename _Iter, typename _Predicate, typename _IteratorTag >`
`_Iter __find_if_switch (_Iter __begin, _Iter __end, _Predicate __pred, _IteratorTag)`
- `template<typename _RAIter, typename _Predicate >`
`_RAIter __find_if_switch (_RAIter __begin, _RAIter __end, _Predicate __pred, random_access_iterator_tag)`
- `template<typename _Iter, typename _Tp, typename _IteratorTag >`
`_Iter __find_switch (_Iter __begin, _Iter __end, const _Tp & __val, _IteratorTag)`
- `template<typename _RAIter, typename _Tp >`
`_RAIter __find_switch (_RAIter __begin, _RAIter __end, const _Tp & __val, random_access_iterator_tag)`
- `template<typename _Iter, typename _Tp, typename _IterTag >`
`_Iter __find_switch (_Iter, _Iter, const _Tp &, _IterTag)`
- `template<typename _Iter, typename _Function, typename _IteratorTag >`
`_Function __for_each_switch (_Iter __begin, _Iter __end, _Function __f, _IteratorTag)`
- `template<typename _RAIter, typename _Function >`
`_Function __for_each_switch (_RAIter __begin, _RAIter __end, _Function __f, random_access_iterator_tag, gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Function, typename _IterTag >`
`_Function __for_each_switch (_Iter, _Iter, _Function, _IterTag)`
- `template<typename _OIter, typename _Size, typename _Generator, typename _IterTag >`
`_OIter __generate_n_switch (_OIter, _Size, _Generator, _IterTag)`
- `template<typename _OutputIterator, typename _Size, typename _Generator, typename _IteratorTag >`
`_OutputIterator __generate_n_switch (_OutputIterator __begin, _Size __n, _Generator __gen, _IteratorTag)`
- `template<typename _RAIter, typename _Size, typename _Generator >`
`_RAIter __generate_n_switch (_RAIter __begin, _Size __n, _Generator __gen, random_access_iterator_tag, gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Filter, typename _Generator, typename _IterTag >`
`void __generate_switch (_Filter, _Filter, _Generator, _IterTag)`

- `template<typename _FIterator, typename _Generator, typename _IteratorTag >`
`void __generate_switch (_FIterator __begin, _FIterator __end, _Generator __gen, _IteratorTag)`
- `template<typename _RAIter, typename _Generator >`
`void __generate_switch (_RAIter __begin, _RAIter __end, _Generator __gen, random_access_iterator_tag, ↵`
`__gnu_parallel::__Parallelism __parallelism_tag)`
- `template<typename _RAIter1, typename _RAIter2, typename _Tp, typename BinaryFunction1, typename BinaryFunction2 >`
`_Tp __inner_product_switch (_RAIter1, _RAIter1, _RAIter2, _Tp, BinaryFunction1, BinaryFunction2, random_↵`
`_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::__Parallelism=__gnu_parallel::parallel_↵`
`unbalanced)`
- `template<typename _IIter1, typename _IIter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2, typename _Tag1`
`, typename _Tag2 >`
`_Tp __inner_product_switch (_IIter1, _IIter1, _IIter2, _Tp, _BinaryFunction1, _BinaryFunction2, _Tag1, _Tag2)`
- `template<typename _IIter1, typename _IIter2, typename _Predicate, typename _IterTag1, typename _IterTag2 >`
`bool __lexicographical_compare_switch (_IIter1, _IIter1, _IIter2, _IIter2, _Predicate, _IterTag1, _IterTag2)`
- `template<typename _IIter1, typename _IIter2, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2 >`
`bool __lexicographical_compare_switch (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2,`
`_Predicate __pred, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _Predicate >`
`bool __lexicographical_compare_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2`
`__end2, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Filter, typename _Compare, typename _IterTag >`
`_Filter __max_element_switch (_Filter, _Filter, _Compare, _IterTag)`
- `template<typename _FIterator, typename _Compare, typename _IteratorTag >`
`_FIterator __max_element_switch (_FIterator __begin, _FIterator __end, _Compare __comp, _IteratorTag)`
- `template<typename _RAIter, typename _Compare >`
`_RAIter __max_element_switch (_RAIter __begin, _RAIter __end, _Compare __comp, random_access_↵`
`iterator_tag, __gnu_parallel::__Parallelism __parallelism_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OIter, typename _Compare, typename _IterTag1, typename _IterTag2, type-`
`name _IterTag3 >`
`_OIter __merge_switch (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Compare, _IterTag1, _IterTag2, _IterTag3)`
- `template<typename _IIter1, typename _IIter2, typename _OIter, typename _Compare >`
`_OIter __merge_switch (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Compare, random_access_iterator_tag,`
`random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Compare, typename _IteratorTag1, typename _↵`
`IteratorTag2, typename _IteratorTag3 >`
`_OutputIterator __merge_switch (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _Output_↵`
`Iterator __result, _Compare __comp, _IteratorTag1, _IteratorTag2, _IteratorTag3)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Compare >`
`_OutputIterator __merge_switch (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _Output_↵`
`Iterator __result, _Compare __comp, random_access_iterator_tag, random_access_iterator_tag, random_↵`
`access_iterator_tag)`
- `template<typename _Filter, typename _Compare, typename _IterTag >`
`_Filter __min_element_switch (_Filter, _Filter, _Compare, _IterTag)`
- `template<typename _FIterator, typename _Compare, typename _IteratorTag >`
`_FIterator __min_element_switch (_FIterator __begin, _FIterator __end, _Compare __comp, _IteratorTag)`
- `template<typename _RAIter, typename _Compare >`
`_RAIter __min_element_switch (_RAIter __begin, _RAIter __end, _Compare __comp, random_access_↵`
`iterator_tag, __gnu_parallel::__Parallelism __parallelism_tag)`
- `template<typename _IIter1, typename _IIter2, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2 >`
`pair< _IIter1, _IIter2 > __mismatch_switch (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _Predicate __↵`
`_pred, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _Predicate >`
`pair< _RAIter1, _RAIter2 > __mismatch_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2,`
`_Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)`

- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IterTag1, typename _IterTag2 >`
`pair< _Iter1, _Iter2 > __mismatch_switch (_Iter1, _Iter1, _Iter2, _Predicate, _IterTag1, _IterTag2)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper, typename _Tag1, typename _Tag2 >`
`_OIter __partial_sum_switch (_Iter, _Iter, _OIter, _BinaryOper, _Tag1, _Tag2)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper >`
`_OIter __partial_sum_switch (_Iter, _Iter, _OIter, _BinaryOper, random_access_iterator_tag, random_↵`
`access_iterator_tag)`
- `template<typename _Filter, typename _Predicate, typename _IterTag >`
`_Filter __partition_switch (_Filter, _Filter, _Predicate, _IterTag)`
- `template<typename _Filterator, typename _Predicate, typename _IteratorTag >`
`_Filterator __partition_switch (_Filterator __begin, _Filterator __end, _Predicate __pred, _IteratorTag)`
- `template<typename _RAIter, typename _Predicate >`
`_RAIter __partition_switch (_RAIter __begin, _RAIter __end, _Predicate __pred, random_access_iterator_tag)`
- `template<typename _Filter, typename _Predicate, typename _Tp, typename _IterTag >`
`void __replace_if_switch (_Filter, _Filter, _Predicate, const _Tp &, _IterTag)`
- `template<typename _Filterator, typename _Predicate, typename _Tp, typename _IteratorTag >`
`void __replace_if_switch (_Filterator __begin, _Filterator __end, _Predicate __pred, const _Tp & __new_value,`
`_IteratorTag)`
- `template<typename _RAIter, typename _Predicate, typename _Tp >`
`void __replace_if_switch (_RAIter __begin, _RAIter __end, _Predicate __pred, const _Tp & __new_value,`
`random_access_iterator_tag, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Filter, typename _Tp, typename _IterTag >`
`void __replace_switch (_Filter, _Filter, const _Tp &, const _Tp &, _IterTag)`
- `template<typename _Filterator, typename _Tp, typename _IteratorTag >`
`void __replace_switch (_Filterator __begin, _Filterator __end, const _Tp & __old_value, const _Tp & __new_value,`
`_IteratorTag)`
- `template<typename _RAIter, typename _Tp >`
`void __replace_switch (_RAIter __begin, _RAIter __end, const _Tp & __old_value, const _Tp & __new_value,`
`random_access_iterator_tag, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _RAIter, typename _Integer, typename _Tp, typename _BiPredicate >`
`_RAIter __search_n_switch (_RAIter, _RAIter, _Integer, const _Tp &, _BiPredicate, random_access_iterator_↵`
`tag)`
- `template<typename _Filter, typename _Integer, typename _Tp, typename _BiPredicate, typename _IterTag >`
`_Filter __search_n_switch (_Filter, _Filter, _Integer, const _Tp &, _BiPredicate, _IterTag)`
- `template<typename _RAIter, typename _Integer, typename _Tp, typename _BinaryPredicate >`
`_RAIter __search_n_switch (_RAIter __begin, _RAIter __end, _Integer __count, const _Tp & __val, _Binary↵`
`Predicate __binary_pred, random_access_iterator_tag)`
- `template<typename _Filterator, typename _Integer, typename _Tp, typename _BinaryPredicate, typename _IteratorTag >`
`_Filterator __search_n_switch (_Filterator __begin, _Filterator __end, _Integer __count, const _Tp & __val, ↵`
`BinaryPredicate __binary_pred, _IteratorTag)`
- `template<typename _Filter1, typename _Filter2, typename _IterTag1, typename _IterTag2 >`
`_Filter1 __search_switch (_Filter1, _Filter1, _Filter2, _Filter2, _IterTag1, _IterTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _BiPredicate >`
`_RAIter1 __search_switch (_RAIter1, _RAIter1, _RAIter2, _RAIter2, _BiPredicate, random_access_iterator_tag,`
`random_access_iterator_tag)`
- `template<typename _Filter1, typename _Filter2, typename _BiPredicate, typename _IterTag1, typename _IterTag2 >`
`_Filter1 __search_switch (_Filter1, _Filter1, _Filter2, _Filter2, _BiPredicate, _IterTag1, _IterTag2)`
- `template<typename _RAIter1, typename _RAIter2 >`
`_RAIter1 __search_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2,`
`random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Filterator1, typename _Filterator2, typename _IteratorTag1, typename _IteratorTag2 >`
`_Filterator1 __search_switch (_Filterator1 __begin1, _Filterator1 __end1, _Filterator2 __begin2, _Filterator2 ↵`
`end2, _IteratorTag1, _IteratorTag2)`

- `template<typename _RAIter1, typename _RAIter2, typename _BinaryPredicate >`
`_RAIter1 __search_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _↵`
`BinaryPredicate __pred, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _FIterator1, typename _FIterator2, typename _BinaryPredicate, typename _IteratorTag1, typename _IteratorTag2 >`
`_FIterator1 __search_switch (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __↵`
`end2, _BinaryPredicate __pred, _IteratorTag1, _IteratorTag2)`
- `template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename _↵`
`IteratorTag2, typename _IteratorTag3 >`
`_OutputIterator __set_difference_switch (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2,`
`_OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAlter, typename _Predicate >`
`_Output_RAlter __set_difference_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2`
`__end2, _Output_RAlter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_↵`
`tag, random_access_iterator_tag)`
- `template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OIter, typename _IterTag1, typename _IterTag2, type-`
`name _IterTag3 >`
`_OIter __set_difference_switch (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate, _IterTag1, _IterTag2, _Iter↵`
`Tag3)`
- `template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename _↵`
`IteratorTag2, typename _IteratorTag3 >`
`_OutputIterator __set_intersection_switch (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2,`
`_OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAlter, typename _Predicate >`
`_Output_RAlter __set_intersection_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2`
`__end2, _Output_RAlter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_↵`
`tag, random_access_iterator_tag)`
- `template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OIter, typename _IterTag1, typename _IterTag2, type-`
`name _IterTag3 >`
`_OIter __set_intersection_switch (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate, _IterTag1, _IterTag2, _↵`
`IterTag3)`
- `template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename _↵`
`IteratorTag2, typename _IteratorTag3 >`
`_OutputIterator __set_symmetric_difference_switch (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2`
`__end2, _OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAlter, typename _Predicate >`
`_Output_RAlter __set_symmetric_difference_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __↵`
`begin2, _RAIter2 __end2, _Output_RAlter __result, _Predicate __pred, random_access_iterator_tag, random_↵`
`access_iterator_tag, random_access_iterator_tag)`
- `template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OIter, typename _IterTag1, typename _IterTag2, type-`
`name _IterTag3 >`
`_OIter __set_symmetric_difference_switch (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate, _IterTag1, _↵`
`IterTag2, _IterTag3)`
- `template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename _↵`
`IteratorTag2, typename _IteratorTag3 >`
`_OutputIterator __set_union_switch (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _↵`
`OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAlter, typename _Predicate >`
`_Output_RAlter __set_union_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __↵`
`end2, _Output_RAlter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag,`
`random_access_iterator_tag)`
- `template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OIter, typename _IterTag1, typename _IterTag2, type-`
`name _IterTag3 >`
`_OIter __set_union_switch (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate, _IterTag1, _IterTag2, _IterTag3)`

- `template<typename _Iter, typename _OIter, typename _UnaryOperation, typename _IterTag1, typename _IterTag2 >`
`_OIter __transform1_switch (_Iter, _Iter, _OIter, _UnaryOperation, _IterTag1, _IterTag2)`
- `template<typename _RAIter, typename _RAOIter, typename _UnaryOperation >`
`_RAOIter __transform1_switch (_RAIter, _RAIter, _RAOIter, _UnaryOperation, random_access_iterator_tag,
random_access_iterator_tag, __gnu_parallel::Parallelism __parallelism=__gnu_parallel::parallel_balanced)`
- `template<typename _RAIter1, typename _RAIter2, typename _UnaryOperation >`
`_RAIter2 __transform1_switch (_RAIter1 __begin, _RAIter1 __end, _RAIter2 __result, _UnaryOperation
__unary_op, random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::Parallelism __
parallelism_tag)`
- `template<typename _RAIter1, typename _RAIter2, typename _UnaryOperation, typename _IteratorTag1, typename _IteratorTag2 >`
`_RAIter2 __transform1_switch (_RAIter1 __begin, _RAIter1 __end, _RAIter2 __result, _UnaryOperation __
unary_op, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _RAIter3, typename _BiOperation >`
`_RAIter3 __transform2_switch (_RAIter1, _RAIter1, _RAIter2, _RAIter3, _BiOperation, random_access_
iterator_tag, random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::Parallelism __
parallelism=__gnu_parallel::parallel_balanced)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BiOperation, typename _Tag1, typename _Tag2, typename`
`_Tag3 >`
`_OIter __transform2_switch (_Iter1, _Iter1, _Iter2, _OIter, _BiOperation, _Tag1, _Tag2, _Tag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _RAIter3, typename _BinaryOperation >`
`_RAIter3 __transform2_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter3 __result,
_BinaryOperation __binary_op, random_access_iterator_tag, random_access_iterator_tag, random_access_
iterator_tag, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _BinaryOperation, typename _Tag1, typename _Tag2`
`, typename _Tag3 >`
`_OutputIterator __transform2_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _OutputIterator __
result, _BinaryOperation __binary_op, _Tag1, _Tag2, _Tag3)`
- `template<typename _Iter, typename _OutputIterator, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2 >`
`_OutputIterator __unique_copy_switch (_Iter __begin, _Iter __last, _OutputIterator __out, _Predicate __pred,
_IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter, typename RandomAccessOutputIterator, typename _Predicate >`
`RandomAccessOutputIterator __unique_copy_switch (_RAIter __begin, _RAIter __last, RandomAccess_
OutputIterator __out, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter, typename _OIter, typename _Predicate, typename _IterTag1, typename _IterTag2 >`
`_OIter __unique_copy_switch (_Iter, _Iter, _OIter, _Predicate, _IterTag1, _IterTag2)`
- `template<typename _RAIter, typename _RandomAccess_OIter, typename _Predicate >`
`_RandomAccess_OIter __unique_copy_switch (_RAIter, _RAIter, _RandomAccess_OIter, _Predicate,
random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter, typename _Tp >`
`_Tp accumulate (_Iter, _Iter, _Tp)`
- `template<typename _Iter, typename _Tp >`
`_Tp accumulate (_Iter, _Iter, _Tp, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _Tp >`
`_Tp accumulate (_Iter, _Iter, _Tp, __gnu_parallel::Parallelism)`
- `template<typename _Iter, typename _Tp, typename _BinaryOper >`
`_Tp accumulate (_Iter, _Iter, _Tp, _BinaryOper)`
- `template<typename _Iter, typename _Tp, typename _BinaryOper >`
`_Tp accumulate (_Iter, _Iter, _Tp, _BinaryOper, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _Tp, typename _BinaryOper >`
`_Tp accumulate (_Iter, _Iter, _Tp, _BinaryOper, __gnu_parallel::Parallelism)`
- `template<typename _Iter, typename _OIter >`
`_OIter adjacent_difference (_Iter, _Iter, _OIter)`

- `template<typename _Iter, typename _OIter, typename _BinaryOper >`
`_OIter adjacent_difference (_Iter, _Iter, _OIter, _BinaryOper)`
- `template<typename _Iter, typename _OIter >`
`_OIter adjacent_difference (_Iter, _Iter, _OIter, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper >`
`_OIter adjacent_difference (_Iter, _Iter, _OIter, _BinaryOper, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _OIter >`
`_OIter adjacent_difference (_Iter, _Iter, _OIter, __gnu_parallel::Parallelism)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper >`
`_OIter adjacent_difference (_Iter, _Iter, _OIter, _BinaryOper, __gnu_parallel::Parallelism)`
- `template<typename _Filter >`
`_Filter adjacent_find (_Filter, _Filter)`
- `template<typename _Filter >`
`_Filter adjacent_find (_Filter, _Filter, __gnu_parallel::sequential_tag)`
- `template<typename _Filter, typename _BiPredicate >`
`_Filter adjacent_find (_Filter, _Filter, _BiPredicate)`
- `template<typename _Filter, typename _BiPredicate >`
`_Filter adjacent_find (_Filter, _Filter, _BiPredicate, __gnu_parallel::sequential_tag)`
- `template<typename _FIterator >`
`_FIterator adjacent_find (_FIterator __begin, _FIterator __end, __gnu_parallel::sequential_tag)`
- `template<typename _FIterator, typename _BinaryPredicate >`
`_FIterator adjacent_find (_FIterator __begin, _FIterator __end, _BinaryPredicate __binary_pred, __gnu_parallel::sequential_tag)`
- `template<typename _FIterator >`
`_FIterator adjacent_find (_FIterator __begin, _FIterator __end)`
- `template<typename _FIterator, typename _BinaryPredicate >`
`_FIterator adjacent_find (_FIterator __begin, _FIterator __end, _BinaryPredicate __pred)`
- `template<typename _Iter, typename _Tp >`
`iterator_traits< _Iter >::difference_type count (_Iter __begin, _Iter __end, const _Tp &__value, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _Tp >`
`iterator_traits< _Iter >::difference_type count (_Iter __begin, _Iter __end, const _Tp &__value, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Tp >`
`iterator_traits< _Iter >::difference_type count (_Iter __begin, _Iter __end, const _Tp &__value)`
- `template<typename _Iter, typename _Predicate >`
`iterator_traits< _Iter >::difference_type count_if (_Iter __begin, _Iter __end, _Predicate __pred, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _Predicate >`
`iterator_traits< _Iter >::difference_type count_if (_Iter __begin, _Iter __end, _Predicate __pred, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Predicate >`
`iterator_traits< _Iter >::difference_type count_if (_Iter __begin, _Iter __end, _Predicate __pred)`
- `template<typename _Iter1, typename _Iter2 >`
`bool equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`
`bool equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2 >`
`bool equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`
`bool equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred)`

- `template<typename _Iter, typename _Tp >`
`_Iter find (_Iter __begin, _Iter __end, const _Tp &__val, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _Tp >`
`_Iter find (_Iter __begin, _Iter __end, const _Tp &__val)`
- `template<typename _Iter, typename _Filter >`
`_Iter find_first_of (_Iter, _Iter, _Filter, _Filter, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _Filter, typename _BiPredicate >`
`_Iter find_first_of (_Iter, _Iter, _Filter, _Filter, _BiPredicate, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _Filter, typename _BiPredicate >`
`_Iter find_first_of (_Iter, _Iter, _Filter, _Filter, _BiPredicate)`
- `template<typename _Iter, typename _Filter >`
`_Iter find_first_of (_Iter, _Iter, _Filter, _Filter)`
- `template<typename _Iter, typename _FIterator >`
`_Iter find_first_of (_Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator __end2, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _FIterator, typename _BinaryPredicate >`
`_Iter find_first_of (_Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator __end2, _BinaryPredicate __comp, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _FIterator, typename _BinaryPredicate >`
`_Iter find_first_of (_Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator __end2, _BinaryPredicate __comp)`
- `template<typename _Iter, typename _FIterator >`
`_Iter find_first_of (_Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator __end2)`
- `template<typename _Iter, typename _Predicate >`
`_Iter find_if (_Iter __begin, _Iter __end, _Predicate __pred, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _Predicate >`
`_Iter find_if (_Iter __begin, _Iter __end, _Predicate __pred)`
- `template<typename _Iter, typename _Function >`
`_Function for_each (_Iter __begin, _Iter __end, _Function __f, __gnu_parallel::sequential_tag)`
- `template<typename _Iterator, typename _Function >`
`_Function for_each (_Iterator __begin, _Iterator __end, _Function __f, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iterator, typename _Function >`
`_Function for_each (_Iterator __begin, _Iterator __end, _Function __f)`
- `template<typename _Iter, typename _Function >`
`_Function for_each (_Iter, _Iter, _Function)`
- `template<typename _Filter, typename _Generator >`
`void generate (_Filter, _Filter, _Generator)`
- `template<typename _Filter, typename _Generator >`
`void generate (_Filter, _Filter, _Generator, __gnu_parallel::sequential_tag)`
- `template<typename _Filter, typename _Generator >`
`void generate (_Filter, _Filter, _Generator, __gnu_parallel::Parallelism)`
- `template<typename _FIterator, typename _Generator >`
`void generate (_FIterator __begin, _FIterator __end, _Generator __gen, __gnu_parallel::sequential_tag)`
- `template<typename _FIterator, typename _Generator >`
`void generate (_FIterator __begin, _FIterator __end, _Generator __gen, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _FIterator, typename _Generator >`
`void generate (_FIterator __begin, _FIterator __end, _Generator __gen)`
- `template<typename _OIter, typename _Size, typename _Generator >`
`_OIter generate_n (_OIter, _Size, _Generator)`
- `template<typename _OIter, typename _Size, typename _Generator >`
`_OIter generate_n (_OIter, _Size, _Generator, __gnu_parallel::sequential_tag)`

- `template<typename _OIter, typename _Size, typename _Generator >`
`_OIter generate_n (_OIter, _Size, _Generator, __gnu_parallel::Parallelism)`
- `template<typename _OutputIterator, typename _Size, typename _Generator >`
`_OutputIterator generate_n (_OutputIterator __begin, _Size __n, _Generator __gen, __gnu_parallel::sequential_tag)`
- `template<typename _OutputIterator, typename _Size, typename _Generator >`
`_OutputIterator generate_n (_OutputIterator __begin, _Size __n, _Generator __gen, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _OutputIterator, typename _Size, typename _Generator >`
`_OutputIterator generate_n (_OutputIterator __begin, _Size __n, _Generator __gen)`
- `template<typename _Iter1, typename _Iter2, typename _Tp >`
`_Tp inner_product (_Iter1, _Iter1, _Iter2, _Tp)`
- `template<typename _Iter1, typename _Iter2, typename _Tp >`
`_Tp inner_product (_Iter1, _Iter1, _Iter2, _Tp, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Tp >`
`_Tp inner_product (_Iter1, _Iter1, _Iter2, _Tp, __gnu_parallel::Parallelism)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2 >`
`_Tp inner_product (_Iter1, _Iter1, _Iter2, _Tp, _BinaryFunction1, _BinaryFunction2)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2 >`
`_Tp inner_product (_Iter1, _Iter1, _Iter2, _Tp, _BinaryFunction1, _BinaryFunction2, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename BinaryFunction1, typename BinaryFunction2 >`
`_Tp inner_product (_Iter1, _Iter1, _Iter2, _Tp, BinaryFunction1, BinaryFunction2, __gnu_parallel::Parallelism)`
- `template<typename _Iter1, typename _Iter2 >`
`bool lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`
`bool lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _Predicate __pred, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2 >`
`bool lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`
`bool lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _Predicate __pred)`
- `template<typename _Filter >`
`_Filter max_element (_Filter, _Filter)`
- `template<typename _Filter >`
`_Filter max_element (_Filter, _Filter, __gnu_parallel::sequential_tag)`
- `template<typename _Filter >`
`_Filter max_element (_Filter, _Filter, __gnu_parallel::Parallelism)`
- `template<typename _Filter, typename _Compare >`
`_Filter max_element (_Filter, _Filter, _Compare)`
- `template<typename _Filter, typename _Compare >`
`_Filter max_element (_Filter, _Filter, _Compare, __gnu_parallel::sequential_tag)`
- `template<typename _Filter, typename _Compare >`
`_Filter max_element (_Filter, _Filter, _Compare, __gnu_parallel::Parallelism)`
- `template<typename _Filterator >`
`_Filterator max_element (_Filterator __begin, _Filterator __end, __gnu_parallel::sequential_tag)`
- `template<typename _Filterator, typename _Compare >`
`_Filterator max_element (_Filterator __begin, _Filterator __end, _Compare __comp, __gnu_parallel::sequential_tag)`

- `template<typename _FIterator >`
`_FIterator max_element (_FIterator __begin, _FIterator __end, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _FIterator >`
`_FIterator max_element (_FIterator __begin, _FIterator __end)`
- `template<typename _FIterator, typename _Compare >`
`_FIterator max_element (_FIterator __begin, _FIterator __end, _Compare __comp, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _FIterator, typename _Compare >`
`_FIterator max_element (_FIterator __begin, _FIterator __end, _Compare __comp)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`
`_OIter merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`
`_OIter merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`
`_OIter merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`
`_OIter merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator >`
`_OutputIterator merge (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __↵
result, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Compare >`
`_OutputIterator merge (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __↵
result, _Compare __comp, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Compare >`
`_OutputIterator merge (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __↵
result, _Compare __comp)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator >`
`_OutputIterator merge (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __↵
result)`
- `template<typename _Filter >`
`_Filter min_element (_Filter, _Filter)`
- `template<typename _Filter >`
`_Filter min_element (_Filter, _Filter, __gnu_parallel::sequential_tag)`
- `template<typename _Filter >`
`_Filter min_element (_Filter, _Filter, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Filter, typename _Compare >`
`_Filter min_element (_Filter, _Filter, _Compare)`
- `template<typename _Filter, typename _Compare >`
`_Filter min_element (_Filter, _Filter, _Compare, __gnu_parallel::sequential_tag)`
- `template<typename _Filter, typename _Compare >`
`_Filter min_element (_Filter, _Filter, _Compare, __gnu_parallel::Parallelism)`
- `template<typename _FIterator >`
`_FIterator min_element (_FIterator __begin, _FIterator __end, __gnu_parallel::sequential_tag)`
- `template<typename _FIterator, typename _Compare >`
`_FIterator min_element (_FIterator __begin, _FIterator __end, _Compare __comp, __gnu_parallel::sequential↵
_tag)`
- `template<typename _FIterator >`
`_FIterator min_element (_FIterator __begin, _FIterator __end, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _FIterator >`
`_FIterator min_element (_FIterator __begin, _FIterator __end)`
- `template<typename _FIterator, typename _Compare >`
`_FIterator min_element (_FIterator __begin, _FIterator __end, _Compare __comp, __gnu_parallel::Parallelism __parallelism_tag)`

- `template<typename _Filterator, typename _Compare >`
`_Filterator min_element (_Filterator __begin, _Filterator __end, _Compare __comp)`
- `template<typename _Iter1, typename _Iter2 >`
`pair< _Iter1, _Iter2 > mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`
`pair< _Iter1, _Iter2 > mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2 >`
`pair< _Iter1, _Iter2 > mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`
`pair< _Iter1, _Iter2 > mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred)`
- `template<typename _RAlter >`
`void nth_element (_RAlter __begin, _RAlter __nth, _RAlter __end, __gnu_parallel::sequential_tag)`
- `template<typename _RAlter, typename _Compare >`
`void nth_element (_RAlter __begin, _RAlter __nth, _RAlter __end, _Compare __comp, __gnu_parallel::sequential_tag)`
- `template<typename _RAlter, typename _Compare >`
`void nth_element (_RAlter __begin, _RAlter __nth, _RAlter __end, _Compare __comp)`
- `template<typename _RAlter >`
`void nth_element (_RAlter __begin, _RAlter __nth, _RAlter __end)`
- `template<typename _RAlter, typename _Compare >`
`void partial_sort (_RAlter __begin, _RAlter __middle, _RAlter __end, _Compare __comp, __gnu_parallel::sequential_tag)`
- `template<typename _RAlter >`
`void partial_sort (_RAlter __begin, _RAlter __middle, _RAlter __end, __gnu_parallel::sequential_tag)`
- `template<typename _RAlter, typename _Compare >`
`void partial_sort (_RAlter __begin, _RAlter __middle, _RAlter __end, _Compare __comp)`
- `template<typename _RAlter >`
`void partial_sort (_RAlter __begin, _RAlter __middle, _RAlter __end)`
- `template<typename _Iter, typename _OIter >`
`_OIter partial_sum (_Iter, _Iter, _OIter, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper >`
`_OIter partial_sum (_Iter, _Iter, _OIter, _BinaryOper, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _OIter >`
`_OIter partial_sum (_Iter, _Iter, _OIter __result)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper >`
`_OIter partial_sum (_Iter, _Iter, _OIter, _BinaryOper)`
- `template<typename _Filter, typename _Predicate >`
`_Filter partition (_Filter, _Filter, _Predicate, __gnu_parallel::sequential_tag)`
- `template<typename _Filter, typename _Predicate >`
`_Filter partition (_Filter, _Filter, _Predicate)`
- `template<typename _Filterator, typename _Predicate >`
`_Filterator partition (_Filterator __begin, _Filterator __end, _Predicate __pred, __gnu_parallel::sequential_tag)`
- `template<typename _Filterator, typename _Predicate >`
`_Filterator partition (_Filterator __begin, _Filterator __end, _Predicate __pred)`
- `template<typename _RAlter >`
`void random_shuffle (_RAlter __begin, _RAlter __end, __gnu_parallel::sequential_tag)`
- `template<typename _RAlter, typename _RandomNumberGenerator >`
`void random_shuffle (_RAlter __begin, _RAlter __end, _RandomNumberGenerator &__rand, __gnu_parallel::sequential_tag)`
- `template<typename _RAlter >`
`void random_shuffle (_RAlter __begin, _RAlter __end)`

- `template<typename _RAIter, typename _RandomNumberGenerator >`
`void random_shuffle (_RAIter __begin, _RAIter __end, _RandomNumberGenerator &&__rand)`
- `template<typename _Filter, typename _Tp >`
`void replace (_Filter, _Filter, const _Tp &, const _Tp &)`
- `template<typename _Filter, typename _Tp >`
`void replace (_Filter, _Filter, const _Tp &, const _Tp &, __gnu_parallel::sequential_tag)`
- `template<typename _Filter, typename _Tp >`
`void replace (_Filter, _Filter, const _Tp &, const _Tp &, __gnu_parallel::Parallelism)`
- `template<typename _Filterator, typename _Tp >`
`void replace (_Filterator __begin, _Filterator __end, const _Tp &__old_value, const _Tp &__new_value, __gnu_parallel::sequential_tag)`
- `template<typename _Filterator, typename _Tp >`
`void replace (_Filterator __begin, _Filterator __end, const _Tp &__old_value, const _Tp &__new_value, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Filterator, typename _Tp >`
`void replace (_Filterator __begin, _Filterator __end, const _Tp &__old_value, const _Tp &__new_value)`
- `template<typename _Filter, typename _Predicate, typename _Tp >`
`void replace_if (_Filter, _Filter, _Predicate, const _Tp &)`
- `template<typename _Filter, typename _Predicate, typename _Tp >`
`void replace_if (_Filter, _Filter, _Predicate, const _Tp &, __gnu_parallel::sequential_tag)`
- `template<typename _Filter, typename _Predicate, typename _Tp >`
`void replace_if (_Filter, _Filter, _Predicate, const _Tp &, __gnu_parallel::Parallelism)`
- `template<typename _Filterator, typename _Predicate, typename _Tp >`
`void replace_if (_Filterator __begin, _Filterator __end, _Predicate __pred, const _Tp &__new_value, __gnu_parallel::sequential_tag)`
- `template<typename _Filterator, typename _Predicate, typename _Tp >`
`void replace_if (_Filterator __begin, _Filterator __end, _Predicate __pred, const _Tp &__new_value, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Filterator, typename _Predicate, typename _Tp >`
`void replace_if (_Filterator __begin, _Filterator __end, _Predicate __pred, const _Tp &__new_value)`
- `template<typename _Filter1, typename _Filter2 >`
`_Filter1 search (_Filter1, _Filter1, _Filter2, _Filter2, __gnu_parallel::sequential_tag)`
- `template<typename _Filter1, typename _Filter2 >`
`_Filter1 search (_Filter1, _Filter1, _Filter2, _Filter2)`
- `template<typename _Filter1, typename _Filter2, typename _BiPredicate >`
`_Filter1 search (_Filter1, _Filter1, _Filter2, _Filter2, _BiPredicate, __gnu_parallel::sequential_tag)`
- `template<typename _Filter1, typename _Filter2, typename _BiPredicate >`
`_Filter1 search (_Filter1, _Filter1, _Filter2, _Filter2, _BiPredicate)`
- `template<typename _Filterator1, typename _Filterator2 >`
`_Filterator1 search (_Filterator1 __begin1, _Filterator1 __end1, _Filterator2 __begin2, _Filterator2 __end2, __gnu_parallel::sequential_tag)`
- `template<typename _Filterator1, typename _Filterator2 >`
`_Filterator1 search (_Filterator1 __begin1, _Filterator1 __end1, _Filterator2 __begin2, _Filterator2 __end2)`
- `template<typename _Filterator1, typename _Filterator2, typename _BinaryPredicate >`
`_Filterator1 search (_Filterator1 __begin1, _Filterator1 __end1, _Filterator2 __begin2, _Filterator2 __end2, __gnu_parallel::sequential_tag, _BinaryPredicate __pred, __gnu_parallel::sequential_tag)`
- `template<typename _Filterator1, typename _Filterator2, typename _BinaryPredicate >`
`_Filterator1 search (_Filterator1 __begin1, _Filterator1 __end1, _Filterator2 __begin2, _Filterator2 __end2, __gnu_parallel::sequential_tag, _BinaryPredicate __pred)`
- `template<typename _Filter, typename _Integer, typename _Tp >`
`_Filter search_n (_Filter, _Filter, _Integer, const _Tp &, __gnu_parallel::sequential_tag)`
- `template<typename _Filter, typename _Integer, typename _Tp, typename _BiPredicate >`
`_Filter search_n (_Filter, _Filter, _Integer, const _Tp &, _BiPredicate, __gnu_parallel::sequential_tag)`

- `template<typename _Filter, typename _Integer, typename _Tp >`
`_Filter search_n (_Filter, _Filter, _Integer, const _Tp &)`
- `template<typename _Filter, typename _Integer, typename _Tp, typename _BiPredicate >`
`_Filter search_n (_Filter, _Filter, _Integer, const _Tp &, _BiPredicate)`
- `template<typename _FIterator, typename _Integer, typename _Tp >`
`_FIterator search_n (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp & __val, __gnu_parallel::sequential_tag)`
- `template<typename _FIterator, typename _Integer, typename _Tp, typename _BinaryPredicate >`
`_FIterator search_n (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp & __val, _BinaryPredicate __binary_pred, __gnu_parallel::sequential_tag)`
- `template<typename _FIterator, typename _Integer, typename _Tp >`
`_FIterator search_n (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp & __val)`
- `template<typename _FIterator, typename _Integer, typename _Tp, typename _BinaryPredicate >`
`_FIterator search_n (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp & __val, _BinaryPredicate __binary_pred)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator >`
`_OutputIterator set_difference (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, __gnu_parallel::sequential_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Predicate >`
`_OutputIterator set_difference (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, _Predicate __pred, __gnu_parallel::sequential_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator >`
`_OutputIterator set_difference (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Predicate >`
`_OutputIterator set_difference (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, _Predicate __pred)`
- `template<typename _IIter1, typename _IIter2, typename _OIter >`
`_OIter set_difference (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, __gnu_parallel::sequential_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OIter, typename _Predicate >`
`_OIter set_difference (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate, __gnu_parallel::sequential_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OIter >`
`_OIter set_difference (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)`
- `template<typename _IIter1, typename _IIter2, typename _OIter, typename _Predicate >`
`_OIter set_difference (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator >`
`_OutputIterator set_intersection (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, __gnu_parallel::sequential_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Predicate >`
`_OutputIterator set_intersection (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, _Predicate __pred, __gnu_parallel::sequential_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator >`
`_OutputIterator set_intersection (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Predicate >`
`_OutputIterator set_intersection (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, _Predicate __pred)`
- `template<typename _IIter1, typename _IIter2, typename _OIter >`
`_OIter set_intersection (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, __gnu_parallel::sequential_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OIter, typename _Predicate >`
`_OIter set_intersection (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate, __gnu_parallel::sequential_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OIter >`
`_OIter set_intersection (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)`

- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate >`
`_OIter set_intersection (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator >`
`_OutputIterator set_symmetric_difference (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2,`
`_OutputIterator __out, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate >`
`_OutputIterator set_symmetric_difference (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2,`
`_OutputIterator __out, _Predicate __pred, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator >`
`_OutputIterator set_symmetric_difference (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2,`
`_OutputIterator __out)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate >`
`_OutputIterator set_symmetric_difference (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2,`
`_OutputIterator __out, _Predicate __pred)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`
`_OIter set_symmetric_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate >`
`_OIter set_symmetric_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`
`_OIter set_symmetric_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate >`
`_OIter set_symmetric_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator >`
`_OutputIterator set_union (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator`
`__out, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate >`
`_OutputIterator set_union (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator`
`__out, _Predicate __pred, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator >`
`_OutputIterator set_union (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator`
`__out)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate >`
`_OutputIterator set_union (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator`
`__out, _Predicate __pred)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`
`_OIter set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate >`
`_OIter set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`
`_OIter set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate >`
`_OIter set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate)`
- `template<typename _RAIter >`
`void sort (_RAIter __begin, _RAIter __end, __gnu_parallel::sequential_tag)`
- `template<typename _RAIter, typename _Compare >`
`void sort (_RAIter __begin, _RAIter __end, _Compare __comp, __gnu_parallel::sequential_tag)`
- `template<typename _RAIter, typename _Compare, typename _Parallelism >`
`void sort (_RAIter __begin, _RAIter __end, _Compare __comp, _Parallelism __parallelism)`
- `template<typename _RAIter >`
`void sort (_RAIter __begin, _RAIter __end)`
- `template<typename _RAIter >`
`void sort (_RAIter __begin, _RAIter __end, __gnu_parallel::default_parallel_tag __parallelism)`

- `template<typename _RAIter >`
`void sort (_RAIter __begin, _RAIter __end, __gnu_parallel::parallel_tag __parallelism)`
- `template<typename _RAIter >`
`void sort (_RAIter __begin, _RAIter __end, __gnu_parallel::multiway_mergesort_tag __parallelism)`
- `template<typename _RAIter >`
`void sort (_RAIter __begin, _RAIter __end, __gnu_parallel::multiway_mergesort_sampling_tag __parallelism)`
- `template<typename _RAIter >`
`void sort (_RAIter __begin, _RAIter __end, __gnu_parallel::multiway_mergesort_exact_tag __parallelism)`
- `template<typename _RAIter >`
`void sort (_RAIter __begin, _RAIter __end, __gnu_parallel::quicksort_tag __parallelism)`
- `template<typename _RAIter >`
`void sort (_RAIter __begin, _RAIter __end, __gnu_parallel::balanced_quicksort_tag __parallelism)`
- `template<typename _RAIter, typename _Compare >`
`void sort (_RAIter __begin, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter >`
`void stable_sort (_RAIter __begin, _RAIter __end, __gnu_parallel::sequential_tag)`
- `template<typename _RAIter, typename _Compare >`
`void stable_sort (_RAIter __begin, _RAIter __end, _Compare __comp, __gnu_parallel::sequential_tag)`
- `template<typename _RAIter, typename _Compare, typename _Parallelism >`
`void stable_sort (_RAIter __begin, _RAIter __end, _Compare __comp, _Parallelism __parallelism)`
- `template<typename _RAIter >`
`void stable_sort (_RAIter __begin, _RAIter __end)`
- `template<typename _RAIter >`
`void stable_sort (_RAIter __begin, _RAIter __end, __gnu_parallel::default_parallel_tag __parallelism)`
- `template<typename _RAIter >`
`void stable_sort (_RAIter __begin, _RAIter __end, __gnu_parallel::parallel_tag __parallelism)`
- `template<typename _RAIter >`
`void stable_sort (_RAIter __begin, _RAIter __end, __gnu_parallel::multiway_mergesort_tag __parallelism)`
- `template<typename _RAIter >`
`void stable_sort (_RAIter __begin, _RAIter __end, __gnu_parallel::quicksort_tag __parallelism)`
- `template<typename _RAIter >`
`void stable_sort (_RAIter __begin, _RAIter __end, __gnu_parallel::balanced_quicksort_tag __parallelism)`
- `template<typename _RAIter, typename _Compare >`
`void stable_sort (_RAIter __begin, _RAIter __end, _Compare __comp)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation >`
`_OIter transform (_Iter, _Iter, _OIter, _UnaryOperation)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation >`
`_OIter transform (_Iter, _Iter, _OIter, _UnaryOperation, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation >`
`_OIter transform (_Iter, _Iter, _OIter, _UnaryOperation, __gnu_parallel::_Parallelism)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BiOperation >`
`_OIter transform (_Iter1, _Iter1, _Iter2, _OIter, _BiOperation)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BiOperation >`
`_OIter transform (_Iter1, _Iter1, _Iter2, _OIter, _BiOperation, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BiOperation >`
`_OIter transform (_Iter1, _Iter1, _Iter2, _OIter, _BiOperation, __gnu_parallel::_Parallelism)`
- `template<typename _Iter, typename _OutputIterator, typename _UnaryOperation >`
`_OutputIterator transform (_Iter __begin, _Iter __end, _OutputIterator __result, _UnaryOperation __unary_op, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _UnaryOperation >`
`_OutputIterator transform (_Iter __begin, _Iter __end, _OutputIterator __result, _UnaryOperation __unary_op, __gnu_parallel::_Parallelism __parallelism_tag)`

- `template<typename _Iter, typename _OutputIterator, typename _UnaryOperation >`
`_OutputIterator transform (_Iter __begin, _Iter __end, _OutputIterator __result, _UnaryOperation __unary_op)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _BinaryOperation >`
`_OutputIterator transform (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _OutputIterator __result, _↵`
`BinaryOperation __binary_op, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _BinaryOperation >`
`_OutputIterator transform (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _OutputIterator __result, _↵`
`BinaryOperation __binary_op, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _BinaryOperation >`
`_OutputIterator transform (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _OutputIterator __result, _↵`
`BinaryOperation __binary_op)`
- `template<typename _Iter, typename _OutputIterator >`
`_OutputIterator unique_copy (_Iter __begin1, _Iter __end1, _OutputIterator __out, __gnu_parallel::sequential↵`
`_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _Predicate >`
`_OutputIterator unique_copy (_Iter __begin1, _Iter __end1, _OutputIterator __out, _Predicate __pred, __gnu↵`
`_parallel::sequential_tag)`
- `template<typename _Iter, typename _OutputIterator >`
`_OutputIterator unique_copy (_Iter __begin1, _Iter __end1, _OutputIterator __out)`
- `template<typename _Iter, typename _OutputIterator, typename _Predicate >`
`_OutputIterator unique_copy (_Iter __begin1, _Iter __end1, _OutputIterator __out, _Predicate __pred)`
- `template<typename _Iter, typename _OIter >`
`_OIter unique_copy (_Iter, _Iter, _OIter, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _OIter, typename _Predicate >`
`_OIter unique_copy (_Iter, _Iter, _OIter, _Predicate, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _OIter >`
`_OIter unique_copy (_Iter, _Iter, _OIter)`
- `template<typename _Iter, typename _OIter, typename _Predicate >`
`_OIter unique_copy (_Iter, _Iter, _OIter, _Predicate)`

3.14.1 Detailed Description

GNU parallel code, replaces standard behavior with parallel behavior.

3.15 `std::__profile` Namespace Reference

Classes

- class [map](#)
- class [multimap](#)
- class [multiset](#)
- class [set](#)

Functions

- `template<typename _UnorderedCont, typename _Value, bool _Cache_hash_code>`
`bool are_equal (const _UnorderedCont &__uc, const __detail::Hash_node< _Value, _Cache_hash_code >`
`*__lhs, const __detail::Hash_node< _Value, _Cache_hash_code > *__rhs)`
- `template<typename _UnorderedCont, typename _Value, bool _Cache_hash_code>`
`std::size_t get_bucket_index (const _UnorderedCont &__uc, const __detail::Hash_node< _Value, _Cache_`
`__hash_code > *__node)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`
`bool operator!= (const __iterator_tracker< _IteratorL, _Sequence > &__lhs, const __iterator_tracker< _`
`IteratorR, _Sequence > &__rhs) noexcept`
- `template<typename _Iterator, typename _Sequence >`
`bool operator!= (const __iterator_tracker< _Iterator, _Sequence > &__lhs, const __iterator_tracker< _Iterator,`
`_Sequence > &__rhs) noexcept`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator!= (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _`
`Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator!= (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp,`
`_Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator!= (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator >`
`&__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator!= (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare,`
`_Allocator > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`
`__iterator_tracker< _Iterator, _Sequence > operator+ (typename __iterator_tracker< _Iterator, _Sequence >::`
`difference_type __n, const __iterator_tracker< _Iterator, _Sequence > &__i) noexcept`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`
`__iterator_tracker< _IteratorL, _Sequence >::difference_type operator- (const __iterator_tracker< _IteratorL,`
`_Sequence > &__lhs, const __iterator_tracker< _IteratorR, _Sequence > &__rhs) noexcept`
- `template<typename _Iterator, typename _Sequence >`
`__iterator_tracker< _Iterator, _Sequence >::difference_type operator- (const __iterator_tracker< _Iterator, _`
`Sequence > &__lhs, const __iterator_tracker< _Iterator, _Sequence > &__rhs) noexcept`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`
`bool operator< (const __iterator_tracker< _IteratorL, _Sequence > &__lhs, const __iterator_tracker< _`
`IteratorR, _Sequence > &__rhs) noexcept`
- `template<typename _Iterator, typename _Sequence >`
`bool operator< (const __iterator_tracker< _Iterator, _Sequence > &__lhs, const __iterator_tracker< _Iterator,`
`_Sequence > &__rhs) noexcept`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator< (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _`
`Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator< (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp,`
`_Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator< (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator >`
`&__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator< (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare,`
`_Allocator > &__rhs)`

- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`
`bool operator>= (const __iterator_tracker< _IteratorL, _Sequence > &__lhs, const __iterator_tracker< _IteratorR, _Sequence > &__rhs) noexcept`
- `template<typename _Iterator, typename _Sequence >`
`bool operator>= (const __iterator_tracker< _Iterator, _Sequence > &__lhs, const __iterator_tracker< _Iterator, _Sequence > &__rhs) noexcept`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator>= (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator>= (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool operator>= (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool operator>= (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`void swap (multiset< _Key, _Compare, _Allocator > &__x, multiset< _Key, _Compare, _Allocator > &__y)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`void swap (set< _Key, _Compare, _Allocator > &__x, set< _Key, _Compare, _Allocator > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`void swap (multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`void swap (map< _Key, _Tp, _Compare, _Allocator > &__lhs, map< _Key, _Tp, _Compare, _Allocator > &__rhs)`

3.15.1 Detailed Description

GNU profile code, replaces standard behavior with profile behavior.

3.16 std::regex_constants Namespace Reference

5.1 Regular Expression Syntax Options

- `enum __syntax_option {`
`_S_icafe, _S_nosubs, _S_optimize, _S_collate,`
`_S_ECMAScript, _S_basic, _S_extended, _S_awk,`
`_S_grep, _S_egrep, _S_syntax_last }`
- `enum syntax_option_type : unsigned int {`
`icafe, nosubs, optimize, collate,`
`ECMAScript, basic, extended, awk,`
`grep, egrep }`
- `constexpr syntax_option_type operator& (syntax_option_type __a, syntax_option_type __b)`
- `constexpr syntax_option_type operator| (syntax_option_type __a, syntax_option_type __b)`
- `constexpr syntax_option_type operator^ (syntax_option_type __a, syntax_option_type __b)`
- `constexpr syntax_option_type operator~ (syntax_option_type __a)`
- `syntax_option_type & operator&= (syntax_option_type &__a, syntax_option_type __b)`
- `syntax_option_type & operator|= (syntax_option_type &__a, syntax_option_type __b)`
- `syntax_option_type & operator^= (syntax_option_type &__a, syntax_option_type __b)`

5.2 Matching Rules

Matching a regular expression against a sequence of characters [first, last) proceeds according to the rules of the grammar specified for the regular expression object, modified according to the effects listed below for any bitmask elements set.

- enum `__match_flag` {
`_S_not_bol`, `_S_not_eol`, `_S_not_bow`, `_S_not_eow`,
`_S_any`, `_S_not_null`, `_S_continuous`, `_S_prev_avail`,
`_S_sed`, `_S_no_copy`, `_S_first_only`, `_S_match_flag_last` }
- enum `match_flag_type` : unsigned int {
`match_default`, `match_not_bol`, `match_not_eol`, `match_not_bow`,
`match_not_eow`, `match_any`, `match_not_null`, `match_continuous`,
`match_prev_avail`, `format_default`, `format_sed`, `format_no_copy`,
`format_first_only` }
- constexpr `match_flag_type` operator& (`match_flag_type` __a, `match_flag_type` __b)
- constexpr `match_flag_type` operator| (`match_flag_type` __a, `match_flag_type` __b)
- constexpr `match_flag_type` operator^ (`match_flag_type` __a, `match_flag_type` __b)
- constexpr `match_flag_type` operator~ (`match_flag_type` __a)
- `match_flag_type` & operator&= (`match_flag_type` &__a, `match_flag_type` __b)
- `match_flag_type` & operator|= (`match_flag_type` &__a, `match_flag_type` __b)
- `match_flag_type` & operator^= (`match_flag_type` &__a, `match_flag_type` __b)

5.3 Error Types

- enum `error_type` {
`_S_error_collate`, `_S_error_ctype`, `_S_error_escape`, `_S_error_backref`,
`_S_error_brack`, `_S_error_paren`, `_S_error_brace`, `_S_error_badbrace`,
`_S_error_range`, `_S_error_space`, `_S_error_badrepeat`, `_S_error_complexity`,
`_S_error_stack` }
- constexpr `error_type` `error_collate` (_S_error_collate)
- constexpr `error_type` `error_ctype` (_S_error_ctype)
- constexpr `error_type` `error_escape` (_S_error_escape)
- constexpr `error_type` `error_backref` (_S_error_backref)
- constexpr `error_type` `error_brack` (_S_error_brack)
- constexpr `error_type` `error_paren` (_S_error_paren)
- constexpr `error_type` `error_brace` (_S_error_brace)
- constexpr `error_type` `error_badbrace` (_S_error_badbrace)
- constexpr `error_type` `error_range` (_S_error_range)
- constexpr `error_type` `error_space` (_S_error_space)
- constexpr `error_type` `error_badrepeat` (_S_error_badrepeat)
- constexpr `error_type` `error_complexity` (_S_error_complexity)
- constexpr `error_type` `error_stack` (_S_error_stack)

3.16.1 Detailed Description

ISO C++-0x entities sub namespace for regex.

3.16.2 Enumeration Type Documentation

3.16.2.1 enum std::regex_constants::__match_flag

This is a bitmask type indicating regex matching rules.

The `match_flag_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

Definition at line 213 of file `regex_constants.h`.

3.16.2.2 enum std::regex_constants::__syntax_option

This is a bitmask type indicating how to interpret the regex.

The `syntax_option_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

A valid value of type `syntax_option_type` shall have exactly one of the elements `ECMAScript`, `basic`, `extended`, `awk`, `grep`, `egrep` set.

Definition at line 54 of file `regex_constants.h`.

3.16.2.3 enum std::regex_constants::error_type

The expression contained an invalid collating element name.

Definition at line 49 of file `regex_error.h`.

3.16.2.4 enum std::regex_constants::match_flag_type : unsigned int

This is a bitmask type indicating regex matching rules.

The `match_flag_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

Enumerator

match_default The default matching rules.

match_not_bol The first character in the sequence [first, last) is treated as though it is not at the beginning of a line, so the character (^) in the regular expression shall not match [first, first).

match_not_eol The last character in the sequence [first, last) is treated as though it is not at the end of a line, so the character (\$) in the regular expression shall not match [last, last).

match_not_bow The expression \b is not matched against the sub-sequence [first,first).

match_not_eow The expression \b should not be matched against the sub-sequence [last,last).

match_any If more than one match is possible then any match is an acceptable result.

match_not_null The expression does not match an empty sequence.

match_continuous The expression only matches a sub-sequence that begins at first .

match_prev_avail –first is a valid iterator position. When this flag is set then the flags `match_not_bol` and `match_not_bow` are ignored by the regular expression algorithms 28.11 and iterators 28.12.

format_default When a regular expression match is to be replaced by a new string, the new string is constructed using the rules used by the ECMAScript `replace` function in ECMA- 262 [Ecma International, ECMAScript Language Specification, Standard Ecma-262, third edition, 1999], part 15.5.4.11 `String.prototype.replace`. In addition, during search and replace operations all non-overlapping occurrences of the regular expression are located and replaced, and sections of the input that did not match the expression are copied unchanged to the output string.

Format strings (from ECMA-262 [15.5.4.11]):

- `$$` The dollar-sign itself (`$`)
- `$&` The matched substring.
- `$'` The portion of *string* that precedes the matched substring. This would be `match_results::prefix()`.
- `$'` The portion of *string* that follows the matched substring. This would be `match_results::suffix()`.
- `$n` The *n*th capture, where *n* is in [1,9] and `$n` is not followed by a decimal digit. If `n <= match_results::size()` and the *n*th capture is undefined, use the empty string instead. If `n > match_results::size()`, the result is implementation-defined.
- `$nn` The *nn*th capture, where *nn* is a two-digit decimal number on [01, 99]. If `nn <= match_results::size()` and the *n*th capture is undefined, use the empty string instead. If `nn > match_results::size()`, the result is implementation-defined.

format_sed When a regular expression match is to be replaced by a new string, the new string is constructed using the rules used by the POSIX `sed` utility in IEEE Std 1003.1- 2001 [IEEE, Information Technology – Portable Operating System Interface (POSIX), IEEE Standard 1003.1-2001].

format_no_copy During a search and replace operation, sections of the character container sequence being searched that do not match the regular expression shall not be copied to the output string.

format_first_only When specified during a search and replace operation, only the first occurrence of the regular expression shall be replaced.

Definition at line 236 of file `regex_constants.h`.

3.16.2.5 `enum std::regex_constants::syntax_option_type : unsigned int`

This is a bitmask type indicating how to interpret the regex.

The `syntax_option_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

A valid value of type `syntax_option_type` shall have exactly one of the elements `ECMAScript`, `basic`, `extended`, `awk`, `grep`, `egrep` set.

Enumerator

icase Specifies that the matching of regular expressions against a character sequence shall be performed without regard to case.

nosubs Specifies that when a regular expression is matched against a character container sequence, no sub-expression matches are to be stored in the supplied `match_results` structure.

optimize Specifies that the regular expression engine should pay more attention to the speed with which regular expressions are matched, and less to the speed with which regular expression objects are constructed. Otherwise it has no detectable effect on the program output.

collate Specifies that character ranges of the form `[a-b]` should be locale sensitive.

ECMAScript Specifies that the grammar recognized by the regular expression engine is that used by ECMAScript in ECMA-262 [Ecma International, ECMAScript Language Specification, Standard Ecma-262, third edition, 1999], as modified in section [28.13]. This grammar is similar to that defined in the PERL scripting language but extended with elements found in the POSIX regular expression grammar.

basic Specifies that the grammar recognized by the regular expression engine is that used by POSIX basic regular expressions in IEEE Std 1003.1-2001, Portable Operating System Interface (POSIX), Base Definitions and Headers, Section 9, Regular Expressions [IEEE, Information Technology – Portable Operating System Interface (POSIX), IEEE Standard 1003.1-2001].

extended Specifies that the grammar recognized by the regular expression engine is that used by POSIX extended regular expressions in IEEE Std 1003.1-2001, Portable Operating System Interface (POSIX), Base Definitions and Headers, Section 9, Regular Expressions.

awk Specifies that the grammar recognized by the regular expression engine is that used by POSIX utility awk in IEEE Std 1003.1-2001. This option is identical to `syntax_option_type extended`, except that C-style escape sequences are supported. These sequences are: `\\`, `\a`, `\b`, `\f`, `\n`, `\r`, `\t`, `\v`, `\'`, `\'`, and `\ddd` (where ddd is one, two, or three octal digits).

grep Specifies that the grammar recognized by the regular expression engine is that used by POSIX utility grep in IEEE Std 1003.1-2001. This option is identical to `syntax_option_type basic`, except that newlines are treated as whitespace.

egrep Specifies that the grammar recognized by the regular expression engine is that used by POSIX utility grep when given the `-E` option in IEEE Std 1003.1-2001. This option is identical to `syntax_option_type extended`, except that newlines are treated as whitespace.

Definition at line 80 of file `regex_constants.h`.

3.16.3 Function Documentation

3.16.3.1 `constexpr error_type std::regex_constants::error_backref (_S_error_backref)`

The expression contained an invalid back reference.

3.16.3.2 `constexpr error_type std::regex_constants::error_badbrace (_S_error_badbrace)`

The expression contained an invalid range in a `{}` expression.

3.16.3.3 `constexpr error_type std::regex_constants::error_badrepeat (_S_error_badrepeat)`

One of `*?+{` was not preceded by a valid regular expression.

3.16.3.4 `constexpr error_type std::regex_constants::error_brace (_S_error_brace)`

The expression contained mismatched `{` and `}`

3.16.3.5 `constexpr error_type std::regex_constants::error_brack (_S_error_brack)`

The expression contained mismatched `[` and `]`.

3.16.3.6 `constexpr error_type std::regex_constants::error_collate (_S_error_collate)`

The expression contained an invalid collating element name.

3.16.3.7 `constexpr error_type std::regex_constants::error_complexity (_S_error_complexity)`

The complexity of an attempted match against a regular expression exceeded a pre-set level.

3.16.3.8 `constexpr error_type std::regex_constants::error_ctype (_S_error_ctype)`

The expression contained an invalid character class name.

3.16.3.9 `constexpr error_type std::regex_constants::error_escape (_S_error_escape)`

The expression contained an invalid escaped character, or a trailing escape.

3.16.3.10 `constexpr error_type std::regex_constants::error_paren (_S_error_paren)`

The expression contained mismatched (and).

3.16.3.11 `constexpr error_type std::regex_constants::error_range (_S_error_range)`

The expression contained an invalid character range, such as [b-a] in most encodings.

3.16.3.12 `constexpr error_type std::regex_constants::error_space (_S_error_space)`

There was insufficient memory to convert the expression into a finite state machine.

3.16.3.13 `constexpr error_type std::regex_constants::error_stack (_S_error_stack)`

There was insufficient memory to determine whether the regular expression could match the specified character sequence.

3.16.3.14 `constexpr syntax_option_type std::regex_constants::operator& (syntax_option_type __a, syntax_option_type __b) [inline]`

This is a bitmask type indicating how to interpret the regex.

The `syntax_option_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

A valid value of type `syntax_option_type` shall have exactly one of the elements `ECMAScript`, `basic`, `extended`, `awk`, `grep`, `egrep` set.

Definition at line 164 of file `regex_constants.h`.

3.16.3.15 `constexpr match_flag_type std::regex_constants::operator& (match_flag_type __a, match_flag_type __b)`
[inline]

This is a bitmask type indicating regex matching rules.

The `match_flag_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

Definition at line 343 of file `regex_constants.h`.

3.16.3.16 `syntax_option_type& std::regex_constants::operator&= (syntax_option_type & __a, syntax_option_type __b)` [inline]

This is a bitmask type indicating how to interpret the regex.

The `syntax_option_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

A valid value of type `syntax_option_type` shall have exactly one of the elements `ECMAScript`, `basic`, `extended`, `awk`, `grep`, `egrep` set.

Definition at line 189 of file `regex_constants.h`.

3.16.3.17 `match_flag_type& std::regex_constants::operator&= (match_flag_type & __a, match_flag_type __b)`
[inline]

This is a bitmask type indicating regex matching rules.

The `match_flag_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

Definition at line 368 of file `regex_constants.h`.

3.16.3.18 `constexpr syntax_option_type std::regex_constants::operator^ (syntax_option_type __a, syntax_option_type __b)` [inline]

This is a bitmask type indicating how to interpret the regex.

The `syntax_option_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

A valid value of type `syntax_option_type` shall have exactly one of the elements `ECMAScript`, `basic`, `extended`, `awk`, `grep`, `egrep` set.

Definition at line 178 of file `regex_constants.h`.

3.16.3.19 `constexpr match_flag_type std::regex_constants::operator^ (match_flag_type __a, match_flag_type __b)`
[inline]

This is a bitmask type indicating regex matching rules.

The `match_flag_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

Definition at line 357 of file `regex_constants.h`.

```
3.16.3.20 syntax_option_type& std::regex_constants::operator^( syntax_option_type __a, syntax_option_type __b
) [inline]
```

This is a bitmask type indicating how to interpret the regex.

The `syntax_option_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

A valid value of type `syntax_option_type` shall have exactly one of the elements `ECMAScript`, `basic`, `extended`, `awk`, `grep`, `egrep` set.

Definition at line 197 of file `regex_constants.h`.

```
3.16.3.21 match_flag_type& std::regex_constants::operator^( match_flag_type __a, match_flag_type __b )
[inline]
```

This is a bitmask type indicating regex matching rules.

The `match_flag_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

Definition at line 376 of file `regex_constants.h`.

```
3.16.3.22 constexpr syntax_option_type std::regex_constants::operator| ( syntax_option_type __a, syntax_option_type
__b ) [inline]
```

This is a bitmask type indicating how to interpret the regex.

The `syntax_option_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

A valid value of type `syntax_option_type` shall have exactly one of the elements `ECMAScript`, `basic`, `extended`, `awk`, `grep`, `egrep` set.

Definition at line 171 of file `regex_constants.h`.

```
3.16.3.23 constexpr match_flag_type std::regex_constants::operator| ( match_flag_type __a, match_flag_type __b )
[inline]
```

This is a bitmask type indicating regex matching rules.

The `match_flag_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

Definition at line 350 of file `regex_constants.h`.

3.16.3.24 `syntax_option_type& std::regex_constants::operator|=(syntax_option_type __a, syntax_option_type __b) [inline]`

This is a bitmask type indicating how to interpret the regex.

The `syntax_option_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

A valid value of type `syntax_option_type` shall have exactly one of the elements `ECMAScript`, `basic`, `extended`, `awk`, `grep`, `egrep` set.

Definition at line 193 of file `regex_constants.h`.

3.16.3.25 `match_flag_type& std::regex_constants::operator|(match_flag_type __a, match_flag_type __b) [inline]`

This is a bitmask type indicating regex matching rules.

The `match_flag_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

Definition at line 372 of file `regex_constants.h`.

3.16.3.26 `constexpr syntax_option_type std::regex_constants::operator~(syntax_option_type __a) [inline]`

This is a bitmask type indicating how to interpret the regex.

The `syntax_option_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

A valid value of type `syntax_option_type` shall have exactly one of the elements `ECMAScript`, `basic`, `extended`, `awk`, `grep`, `egrep` set.

Definition at line 185 of file `regex_constants.h`.

3.16.3.27 `constexpr match_flag_type std::regex_constants::operator~(match_flag_type __a) [inline]`

This is a bitmask type indicating regex matching rules.

The `match_flag_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

Definition at line 364 of file `regex_constants.h`.

3.17 std::rel_ops Namespace Reference

Functions

- `template<class _Tp >`
`bool operator!=(const _Tp &__x, const _Tp &__y)`
- `template<class _Tp >`
`bool operator<= (const _Tp &__x, const _Tp &__y)`
- `template<class _Tp >`
`bool operator> (const _Tp &__x, const _Tp &__y)`
- `template<class _Tp >`
`bool operator>= (const _Tp &__x, const _Tp &__y)`

3.17.1 Detailed Description

The generated relational operators are sequestered here.

3.17.2 Function Documentation

3.17.2.1 `template<class _Tp> bool std::rel_ops::operator!=(const _Tp &__x, const _Tp &__y) [inline]`

Defines != for arbitrary types, in terms of ==.

Parameters

\longleftrightarrow __x	A thing.
\longleftrightarrow __y	Another thing.

Returns

`__x != __y`

This function uses == to determine its result.

Definition at line 87 of file stl_relops.h.

3.17.2.2 `template<class _Tp> bool std::rel_ops::operator<= (const _Tp &__x, const _Tp &__y) [inline]`

Defines <= for arbitrary types, in terms of <.

Parameters

\longleftrightarrow __x	A thing.
\longleftrightarrow __y	Another thing.

Returns

`__x <= __y`

This function uses < to determine its result.

Definition at line 113 of file stl_relops.h.

3.17.2.3 `template<class _Tp> bool std::rel_ops::operator> (const _Tp &__x, const _Tp &__y) [inline]`

Defines > for arbitrary types, in terms of <.

Parameters

$_x$	A thing.
$_y$	Another thing.

Returns

$_x > _y$

This function uses $<$ to determine its result.

Definition at line 100 of file stl_relops.h.

3.17.2.4 `template<class _Tp> bool std::rel_ops::operator>=(const _Tp & __x, const _Tp & __y) [inline]`

Defines $>=$ for arbitrary types, in terms of $<$.

Parameters

$_x$	A thing.
$_y$	Another thing.

Returns

$_x >= _y$

This function uses $<$ to determine its result.

Definition at line 126 of file stl_relops.h.

3.18 std::tr1 Namespace Reference

Namespaces

- [__detail](#)

3.18.1 Detailed Description

ISO C++ TR1 entities toplevel namespace is std::tr1.

3.19 `std::tr1::__detail` Namespace Reference

3.19.1 Detailed Description

Implementation details not part of the namespace `std::tr1` interface.

3.20 `std::tr2` Namespace Reference

Namespaces

- [__detail](#)

3.20.1 Detailed Description

ISO C++ TR2 entities toplevel namespace is `std::tr2`.

3.21 `std::tr2::__detail` Namespace Reference

3.21.1 Detailed Description

Implementation details not part of the namespace `std::tr2` interface.

4 Class Documentation

4.1 `__cxxabiv1::__forced_unwind` Class Reference

4.1.1 Detailed Description

Thrown as part of forced unwinding.

A magic placeholder class that can be caught by reference to recognize forced unwinding.

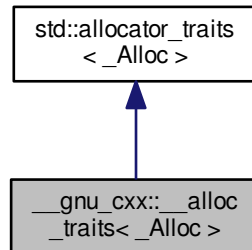
Definition at line 48 of file `cxxabi_forced.h`.

The documentation for this class was generated from the following file:

- [cxxabi_forced.h](#)

4.2 `__gnu_cxx::__alloc_traits<_Alloc>` Struct Template Reference

Inheritance diagram for `__gnu_cxx::__alloc_traits<_Alloc>`:



Public Types

- typedef `std::allocator_traits<_Alloc>` **_Base_type**
- typedef `_Alloc` **allocator_type**
- typedef `_Base_type::const_pointer` **const_pointer**
- typedef `const value_type & const_reference`
- typedef `__const_void_pointer` **const_void_pointer**
- typedef `_Base_type::difference_type` **difference_type**
- typedef `_Base_type::pointer` **pointer**
- typedef `__propagate_on_container_copy_assignment` **propagate_on_container_copy_assignment**
- typedef `__propagate_on_container_move_assignment` **propagate_on_container_move_assignment**
- typedef `__propagate_on_container_swap` **propagate_on_container_swap**
- template<typename `_Tp`>
 - using **rebind_alloc** = typename `__alloc_traits::rebind_alloc<_Alloc, _Tp>`
- template<typename `_Tp`>
 - using **rebind_traits** = `allocator_traits< rebind_alloc<_Tp>>`
- typedef `value_type & reference`
- typedef `_Base_type::size_type` **size_type**
- typedef `_Base_type::value_type` **value_type**
- typedef `__void_pointer` **void_pointer**

Static Public Member Functions

- static constexpr bool **_S_always_equal** ()
- static constexpr bool **_S_nothrow_move** ()
- static constexpr bool **_S_nothrow_swap** ()
- static void **_S_on_swap** (`_Alloc &__a, _Alloc &__b`)
- static constexpr bool **_S_propagate_on_copy_assign** ()
- static constexpr bool **_S_propagate_on_move_assign** ()

- static constexpr bool **_S_propagate_on_swap** ()
- static **_Alloc _S_select_on_copy** (const **_Alloc** &__a)
- static **pointer allocate** (**_Alloc** &__a, **size_type** __n)
- static **pointer allocate** (**_Alloc** &__a, **size_type** __n, **const_void_pointer** __hint)
- template<typename **_Ptr** , typename... **_Args**>
static std::enable_if< __is_custom_pointer< **_Ptr** >::value >::type **construct** (**_Alloc** &__a, **_Ptr** __p, **_Args** &&...__args)
- template<typename **_Tp** , typename... **_Args**>
static auto **construct** (**_Alloc** &__a, **_Tp** *__p, **_Args** &&...__args) -> decltype(**_S_construct**(__a, __p, **std::forward**< **_Args** >(__args)...))
- static void **deallocate** (**_Alloc** &__a, **pointer** __p, **size_type** __n)
- template<typename **_Ptr** >
static std::enable_if< __is_custom_pointer< **_Ptr** >::value >::type **destroy** (**_Alloc** &__a, **_Ptr** __p)
- template<typename **_Tp** >
static void **destroy** (**_Alloc** &__a, **_Tp** *__p)
- static **size_type max_size** (const **_Alloc** &__a) noexcept
- static **_Alloc select_on_container_copy_construction** (const **_Alloc** &__rhs)

4.2.1 Detailed Description

```
template<typename _Alloc>
struct gnu_cxx::__alloc_traits< _Alloc >
```

Uniform interface to C++98 and C++0x allocators.

Definition at line 95 of file ext/alloc_traits.h.

4.2.2 Member Typedef Documentation

4.2.2.1 template<typename **_Alloc**> typedef **__const_void_pointer std::allocator_traits**< **_Alloc** >::**const_void_pointer** [inherited]

The allocator's const void pointer type.

Alloc::const_void_pointer if that type exists, otherwise **pointer_traits**<**pointer**>::rebind<const void>

Definition at line 135 of file bits/alloc_traits.h.

4.2.2.2 template<typename **_Alloc**> typedef **__propagate_on_container_copy_assignment std::allocator_traits**< **_Alloc** >::**propagate_on_container_copy_assignment** [inherited]

How the allocator is propagated on copy assignment.

Alloc::propagate_on_container_copy_assignment if that type exists, otherwise **false_type**

Definition at line 169 of file bits/alloc_traits.h.

4.2.2.3 `template<typename _Alloc> typedef __propagate_on_container_move_assignment std::allocator_traits<_Alloc>::propagate_on_container_move_assignment` [inherited]

How the allocator is propagated on move assignment.

`Alloc::propagate_on_container_move_assignment` if that type exists, otherwise `false_type`

Definition at line 181 of file `bits/alloc_traits.h`.

4.2.2.4 `template<typename _Alloc> typedef __propagate_on_container_swap std::allocator_traits<_Alloc>::propagate_on_container_swap` [inherited]

How the allocator is propagated on swap.

`Alloc::propagate_on_container_swap` if that type exists, otherwise `false_type`

Definition at line 192 of file `bits/alloc_traits.h`.

4.2.2.5 `template<typename _Alloc> typedef __void_pointer std::allocator_traits<_Alloc>::void_pointer` [inherited]

The allocator's void pointer type.

`Alloc::void_pointer` if that type exists, otherwise `pointer_traits<pointer>::rebind<void>`

Definition at line 124 of file `bits/alloc_traits.h`.

4.2.3 Member Function Documentation

4.2.3.1 `template<typename _Alloc> static pointer std::allocator_traits<_Alloc>::allocate (_Alloc &__a, size_type __n)` [inline], [static], [inherited]

Allocate memory.

Parameters

<code>__a</code>	An allocator.
<code>__n</code>	The number of objects to allocate space for.

Calls `a.allocate(n)`

Definition at line 356 of file `bits/alloc_traits.h`.

4.2.3.2 `template<typename _Alloc> static pointer std::allocator_traits<_Alloc>::allocate (_Alloc &__a, size_type __n, const_void_pointer __hint)` [inline], [static], [inherited]

Allocate memory.

Parameters

<code>__a</code>	An allocator.
<code>__n</code>	The number of objects to allocate space for.
<code>__hint</code>	Aid to locality.

Returns

Memory of suitable size and alignment for n objects of type `value_type`

Returns `a.allocate(n, hint)` if that expression is well-formed, otherwise returns `a.allocate(n)`

Definition at line 371 of file `bits/alloc_traits.h`.

4.2.3.3 `template<typename _Alloc> template<typename _Tp, typename... _Args> static auto std::allocator_traits<_Alloc>::construct (_Alloc & __a, _Tp * __p, _Args &&... __args) -> decltype(_S_construct(__a, __p, std::forward<_Args>(__args)...))` `[inline], [static], [inherited]`

Construct an object of type `_Tp`.

Parameters

<code>__a</code>	An allocator.
<code>__p</code>	Pointer to memory of suitable size and alignment for <code>Tp</code>
<code>__args</code>	Constructor arguments.

Calls `__a.construct(__p, std::forward<Args>(__args)...) if that expression is well-formed, otherwise uses placement-new to construct an object of type _Tp at location __p from the arguments __args...`

Definition at line 397 of file `bits/alloc_traits.h`.

4.2.3.4 `template<typename _Alloc> static void std::allocator_traits<_Alloc>::deallocate (_Alloc & __a, pointer __p, size_type __n)` `[inline], [static], [inherited]`

Deallocate memory.

Parameters

<code>__↵ __a</code>	An allocator.
<code>__↵ __p</code>	Pointer to the memory to deallocate.
<code>__↵ __n</code>	The number of objects space was allocated for.

Calls `a.deallocate(p, n)`

Definition at line 382 of file `bits/alloc_traits.h`.

4.2.3.5 `template<typename _Alloc> template<typename _Tp> static void std::allocator_traits<_Alloc>::destroy (_Alloc & __a, _Tp* __p)` `[inline]`, `[static]`, `[inherited]`

Destroy an object of type `_Tp`.

Parameters

<code>__a</code>	An allocator.
<code>__p</code>	Pointer to the object to destroy

Calls `__a.destroy(__p)` if that expression is well-formed, otherwise calls `__p->~Tp()`

Definition at line 410 of file `bits/alloc_traits.h`.

4.2.3.6 `template<typename _Alloc> static size_type std::allocator_traits<_Alloc>::max_size (const _Alloc & __a)` `[inline]`, `[static]`, `[noexcept]`, `[inherited]`

The maximum supported allocation size.

Parameters

<code>__a</code>	An allocator.
------------------	---------------

Returns

`__a.max_size()` or `numeric_limits<size_type>::max()`

Returns `__a.max_size()` if that expression is well-formed, otherwise returns `numeric_limits<size_type>::max()`

Definition at line 421 of file `bits/alloc_traits.h`.

4.2.3.7 `template<typename _Alloc> static _Alloc std::allocator_traits<_Alloc>::select_on_container_copy_construction (const _Alloc & __rhs)` `[inline]`, `[static]`, `[inherited]`

Obtain an allocator to use when copying a container.

Parameters

<code>__rhs</code>	An allocator.
--------------------	---------------

Returns

`__rhs.select_on_container_copy_construction()` or `__rhs`

Returns `__rhs.select_on_container_copy_construction()` if that expression is well-formed, otherwise returns `__rhs`

Definition at line 433 of file `bits/alloc_traits.h`.

The documentation for this struct was generated from the following file:

- [ext/alloc_traits.h](#)

4.3 `__gnu_cxx::__common_pool_policy<_PoolTp, _Thread >` Struct Template Reference

Inherits `__gnu_cxx::__common_pool_base<_PoolTp, _Thread >`.

4.3.1 Detailed Description

```
template<template< bool > class _PoolTp, bool _Thread>
struct __gnu_cxx::__common_pool_policy<_PoolTp, _Thread >
```

Policy for shared `__pool` objects.

Definition at line 460 of file `mt_allocator.h`.

The documentation for this struct was generated from the following file:

- [mt_allocator.h](#)

4.4 `__gnu_cxx::__detail::__mini_vector<_Tp >` Class Template Reference

Public Types

- typedef const `_Tp` & **const_reference**
- typedef ptrdiff_t **difference_type**
- typedef pointer **iterator**
- typedef `_Tp *` **pointer**
- typedef `_Tp &` **reference**
- typedef size_t **size_type**
- typedef `_Tp` **value_type**

Public Member Functions

- reference **back** () const throw ()
- iterator **begin** () const throw ()
- void **clear** () throw ()
- iterator **end** () const throw ()
- void **erase** (iterator `__pos`) throw ()
- void **insert** (iterator `__pos`, const_reference `__x`)
- reference **operator[]** (const size_type `__pos`) const throw ()
- void **pop_back** () throw ()
- void **push_back** (const_reference `__x`)
- size_type **size** () const throw ()

4.4.1 Detailed Description

```
template<typename _Tp>
class __gnu_cxx::__detail::_mini_vector<_Tp>
```

`_mini_vector<>` is a stripped down version of the full-fledged `std::vector<>`.

It is to be used only for built-in types or PODs. Notable differences are:

1. Not all accessor functions are present. 2. Used ONLY for PODs. 3. No Allocator template argument. Uses operator `new()` to get memory, and operator `delete()` to free it. Caveat: The dtor does NOT free the memory allocated, so this a memory-leaking vector!

Definition at line 69 of file `bitmap_allocator.h`.

The documentation for this class was generated from the following file:

- [bitmap_allocator.h](#)

4.5 `__gnu_cxx::__detail::_Bitmap_counter<_Tp>` Class Template Reference

Public Member Functions

- `_Bitmap_counter` ([_BPVector](#) &Rvbp, long __index=-1)
- `pointer` `_M_base` () const throw ()
- `bool` `_M_finished` () const throw ()
- `size_t` * `_M_get` () const throw ()
- `_Index_type` `_M_offset` () const throw ()
- `void` `_M_reset` (long __index=-1) throw ()
- `void` `_M_set_internal_bitmap` (size_t *__new_internal_marker) throw ()
- `_Index_type` `_M_where` () const throw ()
- `_Bitmap_counter` & `operator++` () throw ()

4.5.1 Detailed Description

```
template<typename _Tp>
class __gnu_cxx::__detail::_Bitmap_counter<_Tp>
```

The bitmap counter which acts as the bitmap manipulator, and manages the bit-manipulation functions and the searching and identification functions on the bit-map.

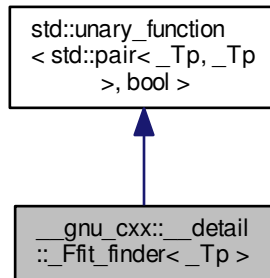
Definition at line 396 of file `bitmap_allocator.h`.

The documentation for this class was generated from the following file:

- [bitmap_allocator.h](#)

4.6 `__gnu_cxx::__detail::_Ffit_finder<_Tp>` Class Template Reference

Inheritance diagram for `__gnu_cxx::__detail::_Ffit_finder<_Tp>`:



Public Types

- typedef `std::pair<_Tp, _Tp>` `argument_type`
- typedef `bool` `result_type`

Public Member Functions

- `size_t * _M_get ()` `const throw ()`
- `_Counter_type _M_offset ()` `const throw ()`
- `bool operator() (_Block_pair __bp)` `throw ()`

4.6.1 Detailed Description

```
template<typename _Tp>
class __gnu_cxx::__detail::_Ffit_finder<_Tp>
```

The class which acts as a predicate for applying the first-fit memory allocation policy for the bitmap allocator.

Definition at line 331 of file `bitmap_allocator.h`.

4.6.2 Member Typedef Documentation

4.6.2.1 typedef `std::pair<_Tp, _Tp>` `std::unary_function<std::pair<_Tp, _Tp>, bool>::argument_type`
[inherited]

`argument_type` is the type of the argument

Definition at line 108 of file `stl_function.h`.

4.6.2.2 `typedef bool std::unary_function< std::pair<_Tp, _Tp>, bool>::result_type` [inherited]

`result_type` is the return type

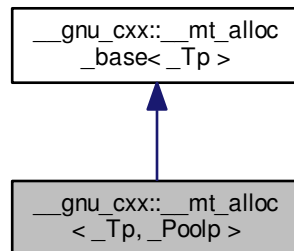
Definition at line 111 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [bitmap_allocator.h](#)

4.7 `__gnu_cxx::__mt_alloc<_Tp, _Poolp>` Class Template Reference

Inheritance diagram for `__gnu_cxx::__mt_alloc<_Tp, _Poolp>`:



Public Types

- `typedef _Poolp __policy_type`
- `typedef _Poolp::pool_type __pool_type`
- `typedef const _Tp * const_pointer`
- `typedef const _Tp & const_reference`
- `typedef ptrdiff_t difference_type`
- `typedef _Tp * pointer`
- `typedef std::true_type propagate_on_container_move_assignment`
- `typedef _Tp & reference`
- `typedef size_t size_type`
- `typedef _Tp value_type`

Public Member Functions

- `__mt_alloc` (const `__mt_alloc` &) noexcept
- `template<typename _Tp1, typename _Poolp1 >`
`__mt_alloc` (const `__mt_alloc`<_Tp1, _Poolp1 > &) noexcept
- `const __pool_base::Tune __M_get_options` ()
- `void __M_set_options` (__pool_base::Tune __t)
- `pointer address` (reference __x) const noexcept
- `const_pointer address` (const_reference __x) const noexcept
- `pointer allocate` (size_type __n, const void *=0)
- `template<typename _Up, typename... _Args>`
`void construct` (_Up *__p, _Args &&... __args)
- `void deallocate` (pointer __p, size_type __n)
- `template<typename _Up >`
`void destroy` (_Up *__p)
- `size_type max_size` () const noexcept

4.7.1 Detailed Description

```
template<typename _Tp, typename _Poolp = __common_pool_policy<__pool, true >>
class __gnu_cxx::__mt_alloc< _Tp, _Poolp >
```

This is a fixed size (power of 2) allocator which - when compiled with thread support - will maintain one freelist per size per thread plus a *global* one. Steps are taken to limit the per thread freelist sizes (by returning excess back to the *global* list).

Further details: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt12ch32.html>.

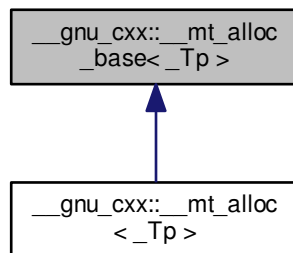
Definition at line 639 of file `mt_allocator.h`.

The documentation for this class was generated from the following file:

- [mt_allocator.h](#)

4.8 `__gnu_cxx::__mt_alloc_base< _Tp >` Class Template Reference

Inheritance diagram for `__gnu_cxx::__mt_alloc_base< _Tp >`:



Public Types

- `typedef const _Tp * const_pointer`
- `typedef const _Tp & const_reference`
- `typedef ptrdiff_t difference_type`
- `typedef _Tp * pointer`
- `typedef std::true_type propagate_on_container_move_assignment`
- `typedef _Tp & reference`
- `typedef size_t size_type`
- `typedef _Tp value_type`

Public Member Functions

- `pointer address (reference __x) const noexcept`
- `const_pointer address (const_reference __x) const noexcept`
- `template<typename _Up, typename... _Args>
void construct (_Up *__p, _Args &&... __args)`
- `template<typename _Up >
void destroy (_Up *__p)`
- `size_type max_size () const noexcept`

4.8.1 Detailed Description

```
template<typename _Tp>
class __gnu_cxx::__mt_alloc_base<_Tp>
```

Base class for `_Tp` dependent member functions.

Definition at line 570 of file `mt_allocator.h`.

The documentation for this class was generated from the following file:

- [mt_allocator.h](#)

4.9 `__gnu_cxx::__per_type_pool_policy<_Tp, _PoolTp, _Thread>` Struct Template Reference

Inherits `__gnu_cxx::__per_type_pool_base<_Tp, _PoolTp, _Thread>`.

4.9.1 Detailed Description

```
template<typename _Tp, template< bool > class _PoolTp, bool _Thread>
struct __gnu_cxx::__per_type_pool_policy<_Tp, _PoolTp, _Thread>
```

Policy for individual `__pool` objects.

Definition at line 555 of file `mt_allocator.h`.

The documentation for this struct was generated from the following file:

- [mt_allocator.h](#)

4.10 `__gnu_cxx::__pool<_Thread>` Class Template Reference

4.10.1 Detailed Description

```
template<bool _Thread>
class __gnu_cxx::__pool<_Thread>
```

Data describing the underlying memory pool, parameterized on threading support.

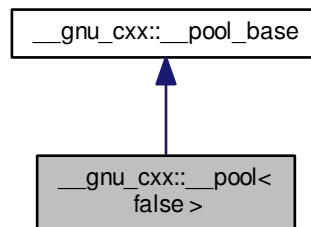
Definition at line 192 of file `mt_allocator.h`.

The documentation for this class was generated from the following file:

- [mt_allocator.h](#)

4.11 `__gnu_cxx::__pool<false>` Class Template Reference

Inheritance diagram for `__gnu_cxx::__pool<false>`:



Public Types

- typedef unsigned short int **`_Binmap_type`**

Public Member Functions

- **`__pool`** (const `__pool_base::_Tune` & `_tune`)
- void **`_M_adjust_freelist`** (const `_Bin_record` &, `_Block_record` *, `size_t`)
- bool **`_M_check_threshold`** (`size_t` `__bytes`)
- void **`_M_destroy`** () throw ()
- `size_t` **`_M_get_align`** ()
- const `_Bin_record` & **`_M_get_bin`** (`size_t` `__which`)
- `size_t` **`_M_get_binmap`** (`size_t` `__bytes`)
- const `_Tune` & **`_M_get_options`** () const
- `size_t` **`_M_get_thread_id`** ()
- void **`_M_initialize_once`** ()
- void **`_M_reclaim_block`** (char * `__p`, `size_t` `__bytes`) throw ()
- char * **`_M_reserve_block`** (`size_t` `__bytes`, const `size_t` `__thread_id`)
- void **`_M_set_options`** (`_Tune` `__t`)

Protected Attributes

- `_Binmap_type * _M_binmap`
- `bool _M_init`
- `_Tune _M_options`

4.11.1 Detailed Description

```
template<>
class __gnu_cxx::__pool< false >
```

Specialization for single thread.

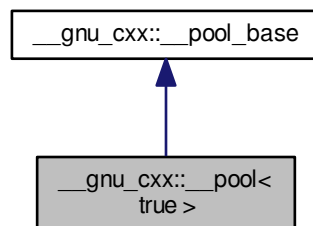
Definition at line 196 of file `mt_allocator.h`.

The documentation for this class was generated from the following file:

- [mt_allocator.h](#)

4.12 __gnu_cxx::__pool< true > Class Template Reference

Inheritance diagram for `__gnu_cxx::__pool< true >`:



Public Types

- `typedef unsigned short int _Binmap_type`

Public Member Functions

- **__pool** (const __pool_base::__Tune &__tune)
- void **_M_adjust_freelist** (const _Bin_record &__bin, _Block_record *__block, size_t __thread_id)
- bool **_M_check_threshold** (size_t __bytes)
- void **_M_destroy** () throw ()
- void **_M_destroy_thread_key** (void *) throw ()
- size_t **_M_get_align** ()
- const _Bin_record & **_M_get_bin** (size_t __which)
- size_t **_M_get_binmap** (size_t __bytes)
- const _Tune & **_M_get_options** () const
- size_t **_M_get_thread_id** ()
- void **_M_initialize** (__destroy_handler)
- void **_M_initialize_once** ()
- void **_M_reclaim_block** (char *__p, size_t __bytes) throw ()
- char * **_M_reserve_block** (size_t __bytes, const size_t __thread_id)
- void **_M_set_options** (_Tune __t)

Protected Attributes

- _Binmap_type * **_M_binmap**
- bool **_M_init**
- _Tune **_M_options**

4.12.1 Detailed Description

```
template<>
class __gnu_cxx::__pool< true >
```

Specialization for thread enabled, via gthreads.h.

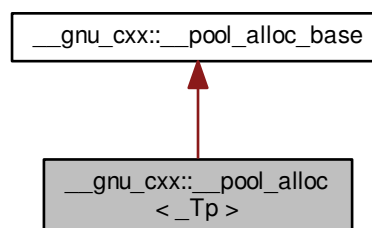
Definition at line 263 of file mt_allocator.h.

The documentation for this class was generated from the following file:

- [mt_allocator.h](#)

4.13 __gnu_cxx::__pool_alloc< _Tp > Class Template Reference

Inheritance diagram for __gnu_cxx::__pool_alloc< _Tp >:



Public Types

- `typedef const _Tp * const_pointer`
- `typedef const _Tp & const_reference`
- `typedef ptrdiff_t difference_type`
- `typedef _Tp * pointer`
- `typedef std::true_type propagate_on_container_move_assignment`
- `typedef _Tp & reference`
- `typedef size_t size_type`
- `typedef _Tp value_type`

Public Member Functions

- `__pool_alloc (const __pool_alloc &) noexcept`
- `template<typename _Tp1 >
 __pool_alloc (const __pool_alloc<_Tp1> &) noexcept`
- `pointer address (reference __x) const noexcept`
- `const_pointer address (const_reference __x) const noexcept`
- `pointer allocate (size_type __n, const void *==0)`
- `template<typename _Up, typename... _Args>
 void construct (_Up * __p, _Args &&... __args)`
- `void deallocate (pointer __p, size_type __n)`
- `template<typename _Up >
 void destroy (_Up * __p)`
- `size_type max_size () const noexcept`

Private Types

- `enum { _S_align }`
- `enum { _S_max_bytes }`
- `enum { _S_free_list_size }`

Private Member Functions

- `char * _M_allocate_chunk (size_t __n, int & __nobjs)`
- `_Obj *volatile * _M_get_free_list (size_t __bytes) throw ()`
- `__mutex & _M_get_mutex () throw ()`
- `void * _M_refill (size_t __n)`
- `size_t _M_round_up (size_t __bytes)`

Static Private Attributes

- `static char * _S_end_free`
- `static _Obj *volatile _S_free_list [_S_free_list_size]`
- `static size_t _S_heap_size`
- `static char * _S_start_free`

4.13.1 Detailed Description

```
template<typename _Tp>
class __gnu_cxx::__pool_alloc<_Tp>
```

Allocator using a memory pool with a single lock.

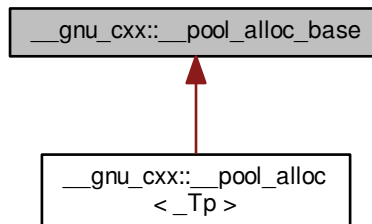
Definition at line 126 of file pool_allocator.h.

The documentation for this class was generated from the following file:

- [pool_allocator.h](#)

4.14 __gnu_cxx::__pool_alloc_base Class Reference

Inheritance diagram for __gnu_cxx::__pool_alloc_base:



Protected Types

- enum { **_S_align** }
- enum { **_S_max_bytes** }
- enum { **_S_free_list_size** }

Protected Member Functions

- char * **_M_allocate_chunk** (size_t __n, int &__nobjs)
- _Obj *volatile * **_M_get_free_list** (size_t __bytes) throw ()
- __mutex & **_M_get_mutex** () throw ()
- void * **_M_refill** (size_t __n)
- size_t **_M_round_up** (size_t __bytes)

Static Protected Attributes

- static char * **_S_end_free**
- static _Obj *volatile **_S_free_list** [_S_free_list_size]
- static size_t **_S_heap_size**
- static char * **_S_start_free**

4.14.1 Detailed Description

Base class for __pool_alloc.

Uses various allocators to fulfill underlying requests (and makes as few requests as possible when in default high-speed pool mode).

Important implementation properties: 0. If globally mandated, then allocate objects from new 1. If the clients request an object of size > _S_max_bytes, the resulting object will be obtained directly from new 2. In all other cases, we allocate an object of size exactly _S_round_up(requested_size). Thus the client has enough size information that we can return the object to the proper free list without permanently losing part of the object.

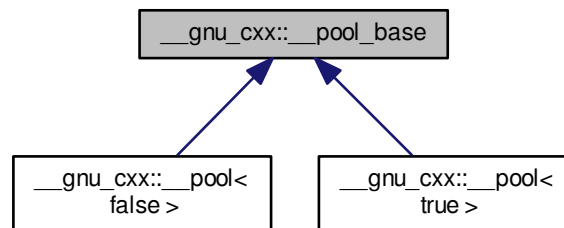
Definition at line 78 of file pool_allocator.h.

The documentation for this class was generated from the following file:

- [pool_allocator.h](#)

4.15 __gnu_cxx::__pool_base Struct Reference

Inheritance diagram for __gnu_cxx::__pool_base:



Public Types

- typedef unsigned short int **_Binmap_type**

Public Member Functions

- **__pool_base** (const _Tune &__options)
- bool **_M_check_threshold** (size_t __bytes)
- size_t **_M_get_align** ()
- size_t **_M_get_binmap** (size_t __bytes)
- const _Tune & **_M_get_options** () const
- void **_M_set_options** (_Tune __t)

Protected Attributes

- _Binmap_type * **_M_binmap**
- bool **_M_init**
- _Tune **_M_options**

4.15.1 Detailed Description

Base class for pool object.

Definition at line 51 of file mt_allocator.h.

The documentation for this struct was generated from the following file:

- [mt_allocator.h](#)

4.16 **__gnu_cxx::__rc_string_base< _CharT, _Traits, _Alloc >** Class Template Reference

Inherits **__gnu_cxx::__vstring_utility< _CharT, _Traits, _Alloc >**.

Public Types

- typedef _Util_Base::_CharT_alloc_type **_CharT_alloc_type**
- typedef __vstring_utility< _CharT, _Traits, _Alloc > **_Util_Base**
- typedef _Alloc **allocator_type**
- typedef _CharT_alloc_type::size_type **size_type**
- typedef _Traits **traits_type**
- typedef _Traits::char_type **value_type**

Public Member Functions

- `__rc_string_base` (const `_Alloc` &__a)
- `__rc_string_base` (const `__rc_string_base` &__rcs)
- `__rc_string_base` (`__rc_string_base` &&__rcs)
- `__rc_string_base` (size_type __n, `_CharT` __c, const `_Alloc` &__a)
- template<typename `_InputIterator` >
`__rc_string_base` (`_InputIterator` __beg, `_InputIterator` __end, const `_Alloc` &__a)
- void `_M_assign` (const `__rc_string_base` &__rcs)
- size_type `_M_capacity` () const
- void `_M_clear` ()
- bool `_M_compare` (const `__rc_string_base` &) const
- template<>
bool `_M_compare` (const `__rc_string_base` &__rcs) const
- template<>
bool `_M_compare` (const `__rc_string_base` &__rcs) const
- `_CharT` * `_M_data` () const
- void `_M_erase` (size_type __pos, size_type __n)
- allocator_type & `_M_get_allocator` ()
- const allocator_type & `_M_get_allocator` () const
- bool `_M_is_shared` () const
- void `_M_leak` ()
- size_type `_M_length` () const
- size_type `_M_max_size` () const
- void `_M_mutate` (size_type __pos, size_type __len1, const `_CharT` * __s, size_type __len2)
- void `_M_reserve` (size_type __res)
- void `_M_set_leaked` ()
- void `_M_set_length` (size_type __n)
- void `_M_swap` (`__rc_string_base` &__rcs)
- template<typename `_InIterator` >
`_CharT` * `_S_construct` (`_InIterator` __beg, `_InIterator` __end, const `_Alloc` &__a, `std::forward_iterator_tag`)

Protected Types

- typedef `__gnu_cxx::__normal_iterator`< const_pointer, `__gnu_cxx::__versa_string`< `_CharT`, `_Traits`, `_Alloc`, `__rc_string_base` > > `__const_rc_iterator`
- typedef `__gnu_cxx::__normal_iterator`< const_pointer, `__gnu_cxx::__versa_string`< `_CharT`, `_Traits`, `_Alloc`, `__sso_string_base` > > `__const_sso_iterator`
- typedef `__gnu_cxx::__normal_iterator`< pointer, `__gnu_cxx::__versa_string`< `_CharT`, `_Traits`, `_Alloc`, `__rc_string_base` > > `__rc_iterator`
- typedef `__gnu_cxx::__normal_iterator`< pointer, `__gnu_cxx::__versa_string`< `_CharT`, `_Traits`, `_Alloc`, `__sso_string_base` > > `__sso_iterator`
- typedef `_CharT_alloc_type::const_pointer` `const_pointer`
- typedef `_CharT_alloc_type::difference_type` `difference_type`
- typedef `_CharT_alloc_type::pointer` `pointer`

Static Protected Member Functions

- static void **_S_assign** (_CharT *__d, size_type __n, _CharT __c)
- static int **_S_compare** (size_type __n1, size_type __n2)
- static void **_S_copy** (_CharT *__d, const _CharT *__s, size_type __n)
- template<typename _Iterator >
static void **_S_copy_chars** (_CharT *__p, _Iterator __k1, _Iterator __k2)
- static void **_S_copy_chars** (_CharT *__p, __sso_iterator __k1, __sso_iterator __k2)
- static void **_S_copy_chars** (_CharT *__p, __const_sso_iterator __k1, __const_sso_iterator __k2)
- static void **_S_copy_chars** (_CharT *__p, __rc_iterator __k1, __rc_iterator __k2)
- static void **_S_copy_chars** (_CharT *__p, __const_rc_iterator __k1, __const_rc_iterator __k2)
- static void **_S_copy_chars** (_CharT *__p, _CharT *__k1, _CharT *__k2)
- static void **_S_copy_chars** (_CharT *__p, const _CharT *__k1, const _CharT *__k2)
- static void **_S_move** (_CharT *__d, const _CharT *__s, size_type __n)

4.16.1 Detailed Description

```
template<typename _CharT, typename _Traits, typename _Alloc>
class __gnu_cxx::__rc_string_base< _CharT, _Traits, _Alloc >
```

Documentation? What's that? Nathan Myers ncm@cantrip.org.

A string looks like this:

```

                                     [_Rep]
                                     _M_length
[ __rc_string_base<char_type>]      _M_capacity
_M_datapointer                    _M_refcount
_M_p ----->                     unnamed array of char_type
```

Where the `_M_p` points to the first character in the string, and you cast it to a pointer-to-`_Rep` and subtract 1 to get a pointer to the header.

This approach has the enormous advantage that a string object requires only one allocation. All the ugliness is confined within a single pair of inline functions, which each compile to a single *add* instruction: `_Rep::_M_refdata()`, and `__rc_string_base::_M_rep()`; and the allocation function which gets a block of raw bytes and with room enough and constructs a `_Rep` object at the front.

The reason you want `_M_data` pointing to the character array and not the `_Rep` is so that the debugger can see the string contents. (Probably we should add a non-inline member to get the `_Rep` for the debugger to use, so users can check the actual string length.)

Note that the `_Rep` object is a POD so that you can have a static *empty string* `_Rep` object already *constructed* before static constructors have run. The reference-count encoding is chosen so that a 0 indicates one reference, so you never try to destroy the empty-string `_Rep` object.

All but the last paragraph is considered pretty conventional for a C++ string implementation.

Definition at line 82 of file `rc_string_base.h`.

The documentation for this class was generated from the following file:

- [rc_string_base.h](#)

4.17 `__gnu_cxx::__scoped_lock` Class Reference

Public Types

- typedef `__mutex` **`__mutex_type`**

Public Member Functions

- **`__scoped_lock`** (`__mutex_type` &`__name`)

4.17.1 Detailed Description

Scoped lock idiom.

Definition at line 231 of file `concurrency.h`.

The documentation for this class was generated from the following file:

- [concurrency.h](#)

4.18 `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >` Class Template Reference

Inherits `_Base<_CharT, _Traits, _Alloc >`.

Public Types

- typedef `_Alloc` **`allocator_type`**
- typedef `__gnu_cxx::__normal_iterator< const_pointer, __versa_string >` **`const_iterator`**
- typedef `_CharT_alloc_type::const_pointer` **`const_pointer`**
- typedef `const value_type &` **`const_reference`**
- typedef `std::reverse_iterator< const_iterator >` **`const_reverse_iterator`**
- typedef `_CharT_alloc_type::difference_type` **`difference_type`**
- typedef `__gnu_cxx::__normal_iterator< pointer, __versa_string >` **`iterator`**
- typedef `_CharT_alloc_type::pointer` **`pointer`**
- typedef `value_type &` **`reference`**
- typedef `std::reverse_iterator< iterator >` **`reverse_iterator`**
- typedef `_CharT_alloc_type::size_type` **`size_type`**
- typedef `_Traits` **`traits_type`**
- typedef `_Traits::char_type` **`value_type`**

Public Member Functions

- [__versa_string](#) (const [_Alloc](#) &__a=[_Alloc\(\)](#)) noexcept
- [__versa_string](#) (const [__versa_string](#) &__str)
- [__versa_string](#) ([__versa_string](#) &&__str) noexcept
- [__versa_string](#) (std::initializer_list< [_CharT](#) > __l, const [_Alloc](#) &__a=[_Alloc\(\)](#))
- [__versa_string](#) (const [__versa_string](#) &__str, size_type __pos, size_type __n=[npos](#))
- [__versa_string](#) (const [__versa_string](#) &__str, size_type __pos, size_type __n, const [_Alloc](#) &__a)
- [__versa_string](#) (const [_CharT](#) *__s, size_type __n, const [_Alloc](#) &__a=[_Alloc\(\)](#))
- [__versa_string](#) (const [_CharT](#) *__s, const [_Alloc](#) &__a=[_Alloc\(\)](#))
- [__versa_string](#) (size_type __n, [_CharT](#) __c, const [_Alloc](#) &__a=[_Alloc\(\)](#))
- template<class [_InputIterator](#) , typename = std:: [RequireInputIter](#)< [_InputIterator](#)>>
 [__versa_string](#) ([_InputIterator](#) __beg, [_InputIterator](#) __end, const [_Alloc](#) &__a=[_Alloc\(\)](#))
- [~__versa_string](#) () noexcept
- [__versa_string](#) & [append](#) (const [__versa_string](#) &__str)
- [__versa_string](#) & [append](#) (const [__versa_string](#) &__str, size_type __pos, size_type __n)
- [__versa_string](#) & [append](#) (const [_CharT](#) *__s, size_type __n)
- [__versa_string](#) & [append](#) (const [_CharT](#) *__s)
- [__versa_string](#) & [append](#) (size_type __n, [_CharT](#) __c)
- [__versa_string](#) & [append](#) (std::initializer_list< [_CharT](#) > __l)
- template<class [_InputIterator](#) , typename = std:: [RequireInputIter](#)< [_InputIterator](#)>>
 [__versa_string](#) & [append](#) ([_InputIterator](#) __first, [_InputIterator](#) __last)
- [__versa_string](#) & [assign](#) (const [__versa_string](#) &__str)
- [__versa_string](#) & [assign](#) ([__versa_string](#) &&__str) noexcept
- [__versa_string](#) & [assign](#) (const [__versa_string](#) &__str, size_type __pos, size_type __n)
- [__versa_string](#) & [assign](#) (const [_CharT](#) *__s, size_type __n)
- [__versa_string](#) & [assign](#) (const [_CharT](#) *__s)
- [__versa_string](#) & [assign](#) (size_type __n, [_CharT](#) __c)
- template<class [_InputIterator](#) , typename = std:: [RequireInputIter](#)< [_InputIterator](#)>>
 [__versa_string](#) & [assign](#) ([_InputIterator](#) __first, [_InputIterator](#) __last)
- [__versa_string](#) & [assign](#) (std::initializer_list< [_CharT](#) > __l)
- const_reference [at](#) (size_type __n) const
- reference [at](#) (size_type __n)
- reference [back](#) () noexcept
- const_reference [back](#) () const noexcept
- iterator [begin](#) () noexcept
- const_iterator [begin](#) () const noexcept
- const [_CharT](#) * [c_str](#) () const noexcept
- size_type [capacity](#) () const noexcept
- const_iterator [cbegin](#) () const noexcept
- const_iterator [cend](#) () const noexcept
- void [clear](#) () noexcept
- int [compare](#) (const [__versa_string](#) &__str) const
- int [compare](#) (size_type __pos, size_type __n, const [__versa_string](#) &__str) const
- int [compare](#) (size_type __pos1, size_type __n1, const [__versa_string](#) &__str, size_type __pos2, size_type __n2) const
- int [compare](#) (const [_CharT](#) *__s) const
- int [compare](#) (size_type __pos, size_type __n1, const [_CharT](#) *__s) const
- int [compare](#) (size_type __pos, size_type __n1, const [_CharT](#) *__s, size_type __n2) const
- size_type [copy](#) ([_CharT](#) *__s, size_type __n, size_type __pos=0) const
- const_reverse_iterator [crbegin](#) () const noexcept

- `const_reverse_iterator` `crend` () const noexcept
- `const _CharT *` `data` () const noexcept
- `bool` `empty` () const noexcept
- `iterator` `end` () noexcept
- `const_iterator` `end` () const noexcept
- `__versa_string` & `erase` (size_type __pos=0, size_type __n=npos)
- `iterator` `erase` (const_iterator __position)
- `iterator` `erase` (const_iterator __first, const_iterator __last)
- size_type `find` (const _CharT *__s, size_type __pos, size_type __n) const
- size_type `find` (const __versa_string & __str, size_type __pos=0) const noexcept
- size_type `find` (const _CharT *__s, size_type __pos=0) const
- size_type `find` (_CharT __c, size_type __pos=0) const noexcept
- size_type `find_first_not_of` (const __versa_string & __str, size_type __pos=0) const noexcept
- size_type `find_first_not_of` (const _CharT *__s, size_type __pos, size_type __n) const
- size_type `find_first_not_of` (const _CharT *__s, size_type __pos=0) const
- size_type `find_first_not_of` (_CharT __c, size_type __pos=0) const noexcept
- size_type `find_first_of` (const __versa_string & __str, size_type __pos=0) const noexcept
- size_type `find_first_of` (const _CharT *__s, size_type __pos, size_type __n) const
- size_type `find_first_of` (const _CharT *__s, size_type __pos=0) const
- size_type `find_first_of` (_CharT __c, size_type __pos=0) const noexcept
- size_type `find_last_not_of` (const __versa_string & __str, size_type __pos=npos) const noexcept
- size_type `find_last_not_of` (const _CharT *__s, size_type __pos, size_type __n) const
- size_type `find_last_not_of` (const _CharT *__s, size_type __pos=npos) const
- size_type `find_last_not_of` (_CharT __c, size_type __pos=npos) const noexcept
- size_type `find_last_of` (const __versa_string & __str, size_type __pos=npos) const noexcept
- size_type `find_last_of` (const _CharT *__s, size_type __pos, size_type __n) const
- size_type `find_last_of` (const _CharT *__s, size_type __pos=npos) const
- size_type `find_last_of` (_CharT __c, size_type __pos=npos) const noexcept
- `reference` `front` () noexcept
- `const_reference` `front` () const noexcept
- `allocator_type` `get_allocator` () const noexcept
- `iterator` `insert` (const_iterator __p, size_type __n, _CharT __c)
- `template<class _InputIterator, typename = std::RequireInputIter<_InputIterator>>`
`iterator` `insert` (const_iterator __p, _InputIterator __beg, _InputIterator __end)
- `iterator` `insert` (const_iterator __p, std::initializer_list<_CharT> __l)
- `__versa_string` & `insert` (size_type __pos1, const __versa_string & __str)
- `__versa_string` & `insert` (size_type __pos1, const __versa_string & __str, size_type __pos2, size_type __n)
- `__versa_string` & `insert` (size_type __pos, const _CharT *__s, size_type __n)
- `__versa_string` & `insert` (size_type __pos, const _CharT *__s)
- `__versa_string` & `insert` (size_type __pos, size_type __n, _CharT __c)
- `iterator` `insert` (const_iterator __p, _CharT __c)
- size_type `length` () const noexcept
- size_type `max_size` () const noexcept
- `__versa_string` & `operator+=` (const __versa_string & __str)
- `__versa_string` & `operator+=` (const _CharT *__s)
- `__versa_string` & `operator+=` (_CharT __c)
- `__versa_string` & `operator+=` (std::initializer_list<_CharT> __l)
- `__versa_string` & `operator=` (const __versa_string & __str)
- `__versa_string` & `operator=` (__versa_string && __str) noexcept
- `__versa_string` & `operator=` (std::initializer_list<_CharT> __l)
- `__versa_string` & `operator=` (const _CharT *__s)

- `__versa_string & operator= (_CharT __c)`
- `const_reference operator[] (size_type __pos) const noexcept`
- `reference operator[] (size_type __pos) noexcept`
- `void pop_back ()`
- `void push_back (_CharT __c)`
- `reverse_iterator rbegin () noexcept`
- `const_reverse_iterator rbegin () const noexcept`
- `reverse_iterator rend () noexcept`
- `const_reverse_iterator rend () const noexcept`
- `__versa_string & replace (size_type __pos, size_type __n, const __versa_string & __str)`
- `__versa_string & replace (size_type __pos1, size_type __n1, const __versa_string & __str, size_type __pos2, size_type __n2)`
- `__versa_string & replace (size_type __pos, size_type __n1, const _CharT * __s, size_type __n2)`
- `__versa_string & replace (size_type __pos, size_type __n1, const _CharT * __s)`
- `__versa_string & replace (size_type __pos, size_type __n1, size_type __n2, _CharT __c)`
- `__versa_string & replace (const_iterator __i1, const_iterator __i2, const __versa_string & __str)`
- `__versa_string & replace (const_iterator __i1, const_iterator __i2, const _CharT * __s, size_type __n)`
- `__versa_string & replace (const_iterator __i1, const_iterator __i2, const _CharT * __s)`
- `__versa_string & replace (const_iterator __i1, const_iterator __i2, size_type __n, _CharT __c)`
- `template<class _InputIterator, typename = std::RequireInputIter<_InputIterator>>>
__versa_string & replace (const_iterator __i1, const_iterator __i2, _InputIterator __k1, _InputIterator __k2)`
- `__versa_string & replace (const_iterator __i1, const_iterator __i2, _CharT * __k1, _CharT * __k2)`
- `__versa_string & replace (const_iterator __i1, const_iterator __i2, const _CharT * __k1, const _CharT * __k2)`
- `__versa_string & replace (const_iterator __i1, const_iterator __i2, iterator __k1, iterator __k2)`
- `__versa_string & replace (const_iterator __i1, const_iterator __i2, const_iterator __k1, const_iterator __k2)`
- `__versa_string & replace (const_iterator __i1, const_iterator __i2, std::initializer_list<_CharT> __l)`
- `void reserve (size_type __res_arg=0)`
- `void resize (size_type __n, _CharT __c)`
- `void resize (size_type __n)`
- `size_type rfind (const __versa_string & __str, size_type __pos=npos) const noexcept`
- `size_type rfind (const _CharT * __s, size_type __pos, size_type __n) const`
- `size_type rfind (const _CharT * __s, size_type __pos=npos) const`
- `size_type rfind (_CharT __c, size_type __pos=npos) const noexcept`
- `void shrink_to_fit () noexcept`
- `size_type size () const noexcept`
- `__versa_string substr (size_type __pos=0, size_type __n=npos) const`
- `void swap (__versa_string & __s) noexcept`

Static Public Attributes

- `static const size_type npos`

4.18.1 Detailed Description

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
class __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >
```

Template class `__versa_string`.

Data structure managing sequences of characters and character-like objects.

Definition at line 56 of file `vstring.h`.

4.18.2 Constructor & Destructor Documentation

4.18.2.1 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::__versa_string (const _Alloc & __a = _Alloc()) [inline], [explicit], [noexcept]`

Construct an empty string using allocator *a*.

Definition at line 137 of file `vstring.h`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::substr()`.

4.18.2.2 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::__versa_string (const __versa_string<_CharT, _Traits, _Alloc, _Base> & __str) [inline]`

Construct string with copy of value of `__str`.

Parameters

<code>__str</code>	Source string.
--------------------	----------------

Definition at line 145 of file `vstring.h`.

4.18.2.3 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::__versa_string (__versa_string<_CharT, _Traits, _Alloc, _Base> && __str) [inline], [noexcept]`

String move constructor.

Parameters

<code>__str</code>	Source string.
--------------------	----------------

The newly-constructed string contains the exact contents of `__str`. The contents of `__str` are a valid, but unspecified string.

Definition at line 157 of file `vstring.h`.

4.18.2.4 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::__versa_string (std::initializer_list<_CharT> __l, const _Alloc & __a = _Alloc()) [inline]`

Construct string from an initializer list.

Parameters

<code>_↔ _l</code>	std::initializer_list of characters.
<code>_↔ _a</code>	Allocator to use (default is default allocator).

Definition at line 165 of file vstring.h.

```
4.18.2.5  template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename >
          class _Base> __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::__versa_string ( const
          __versa_string< _CharT, _Traits, _Alloc, _Base > & __str, size_type __pos, size_type __n = npos ) [inline]
```

Construct string as copy of a substring.

Parameters

<code>__str</code>	Source string.
<code>__pos</code>	Index of first character to copy from.
<code>__n</code>	Number of characters to copy (default remainder).

Definition at line 176 of file vstring.h.

```
4.18.2.6  template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename >
          class _Base> __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::__versa_string ( const
          __versa_string< _CharT, _Traits, _Alloc, _Base > & __str, size_type __pos, size_type __n, const _Alloc & __a )
          [inline]
```

Construct string as copy of a substring.

Parameters

<code>__str</code>	Source string.
<code>__pos</code>	Index of first character to copy from.
<code>__n</code>	Number of characters to copy.
<code>__a</code>	Allocator to use.

Definition at line 191 of file vstring.h.

```
4.18.2.7  template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
          _Base> __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::__versa_string ( const _CharT * __s,
          size_type __n, const _Alloc & __a = _Alloc() ) [inline]
```

Construct string initialized by a character array.

Parameters

<code>__s</code>	Source character array.
<code>__n</code>	Number of characters to copy.
<code>__a</code>	Allocator to use (default is default allocator).

NB: `__s` must have at least `__n` characters, `"0"` has no special meaning.

Definition at line 208 of file `vstring.h`.

```
4.18.2.8 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
_Base> __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::__versa_string ( const _CharT * __s,
const _Alloc & __a = _Alloc() ) [inline]
```

Construct string as copy of a C string.

Parameters

<code>__s</code>	Source C string.
<code>__a</code>	Allocator to use (default is default allocator).

Definition at line 217 of file `vstring.h`.

```
4.18.2.9 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
_Base> __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::__versa_string ( size_type __n, _CharT
__c, const _Alloc & __a = _Alloc() ) [inline]
```

Construct string as multiple characters.

Parameters

<code>__n</code>	Number of characters.
<code>__c</code>	Character to use.
<code>__a</code>	Allocator to use (default is default allocator).

Definition at line 227 of file `vstring.h`.

```
4.18.2.10 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base> template<class _InputIterator , typename = std::_RequireInputIter<_InputIterator>>
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::__versa_string ( _InputIterator __beg,
_InputIterator __end, const _Alloc & __a = _Alloc() ) [inline]
```

Construct string as copy of a range.

Parameters

<code>__beg</code>	Start of range.
<code>__end</code>	End of range.
<code>__a</code>	Allocator to use (default is default allocator).

Definition at line 242 of file `vstring.h`.

```
4.18.2.11 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
_Base> __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::~~__versa_string ( ) [inline],
[noexcept]
```

Destroy the string instance.

Definition at line 249 of file `vstring.h`.

4.18.3 Member Function Documentation

```
4.18.3.1 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
_Base> __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append ( const
__versa_string< _CharT, _Traits, _Alloc, _Base > & __str ) [inline]
```

Append a string to this string.

Parameters

<code>__str</code>	The string to append.
--------------------	-----------------------

Returns

Reference to this string.

Definition at line 692 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size()`.

Referenced by `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare()`, and `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator+=()`.

4.18.3.2 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::append (const __versa_string<_CharT, _Traits, _Alloc, _Base > &__str, size_type __pos, size_type __n) [inline]`

Append a substring.

Parameters

<code>__str</code>	The string to append.
<code>__pos</code>	Index of the first character of <code>str</code> to append.
<code>__n</code>	The number of characters to append.

Returns

Reference to this string.

Exceptions

<code>std::out_of_range</code>	if <code>pos</code> is not a valid index.
--------------------------------	---

This function appends `__n` characters from `__str` starting at `__pos` to this string. If `__n` is larger than the number of available characters in `__str`, the remainder of `__str` is appended.

Definition at line 709 of file `vstring.h`.

4.18.3.3 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::append (const _CharT* __s, size_type __n) [inline]`

Append a C substring.

Parameters

<code>__s</code>	The C string to append.
<code>__n</code>	The number of characters to append.

Returns

Reference to this string.

Definition at line 721 of file `vstring.h`.

4.18.3.4 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
_Base> __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append (const _CharT *
__s) [inline]`

Append a C string.

Parameters

<code>__s</code>	The C string to append.
------------------	-------------------------

Returns

Reference to this string.

Definition at line 734 of file `vstring.h`.

```
4.18.3.5  template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
          _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::append ( size_type __n,
          _CharT __c ) [inline]
```

Append multiple characters.

Parameters

<code>__n</code>	The number of characters to append.
<code>__c</code>	The character to use.

Returns

Reference to this string.

Appends `n` copies of `c` to this string.

Definition at line 751 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

```
4.18.3.6  template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename >
          class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::append (
          std::initializer_list<_CharT> __l ) [inline]
```

Append an `initializer_list` of characters.

Parameters

<code>l</code>	The <code>initializer_list</code> of characters to append.
----------------	--

Returns

Reference to this string.

Definition at line 761 of file vstring.h.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::append()`.

```
4.18.3.7 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
        _Base> template<class _InputIterator , typename = std:: RequireInputIter<_InputIterator>> __versa_string&
        __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append ( _InputIterator __first, _InputIterator __last )
        [inline]
```

Append a range of characters.

Parameters

<code>__first</code>	Iterator referencing the first character to append.
<code>__last</code>	Iterator marking the end of the range.

Returns

Reference to this string.

Appends characters in the range [first,last) to this string.

Definition at line 780 of file vstring.h.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::replace()`.

```
4.18.3.8 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
        _Base> __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::assign ( const
        __versa_string< _CharT, _Traits, _Alloc, _Base > & __str ) [inline]
```

Set value to contents of another string.

Parameters

<code>__str</code>	Source string to use.
--------------------	-----------------------

Returns

Reference to this string.

Definition at line 803 of file vstring.h.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::assign()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::operator=()`.

4.18.3.9 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::assign (__versa_string<_CharT, _Traits, _Alloc, _Base > && __str) [inline], [noexcept]`

Set value to contents of another string.

Parameters

<code>__str</code>	Source string to use.
--------------------	-----------------------

Returns

Reference to this string.

This function sets this string to the exact contents of `__str`. `__str` is a valid, but unspecified string.

Definition at line 819 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::swap()`.

4.18.3.10 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::assign (const __versa_string<_CharT, _Traits, _Alloc, _Base > & __str, size_type __pos, size_type __n) [inline]`

Set value to a substring of a string.

Parameters

<code>__str</code>	The string to use.
<code>__pos</code>	Index of the first character of str.
<code>__n</code>	Number of characters to use.

Returns

Reference to this string.

Exceptions

<code>std::out_of_range</code>	if <code>__pos</code> is not a valid index.
--------------------------------	---

This function sets this string to the substring of `__str` consisting of `__n` characters at `__pos`. If `__n` is larger than the number of available characters in `__str`, the remainder of `__str` is used.

Definition at line 840 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::size()`.

```
4.18.3.11 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
_Base> __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::assign ( const _CharT
*_s, size_type __n ) [inline]
```

Set value to a C substring.

Parameters

<code>__s</code>	The C string to use.
<code>__n</code>	Number of characters to use.

Returns

Reference to this string.

This function sets the value of this string to the first `__n` characters of `__s`. If `__n` is larger than the number of available characters in `__s`, the remainder of `__s` is used.

Definition at line 857 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size()`.

```
4.18.3.12 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
_Base> __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::assign ( const _CharT
*_s ) [inline]
```

Set value to contents of a C string.

Parameters

<code>__s</code>	The C string to use.
------------------	----------------------

Returns

Reference to this string.

This function sets the value of this string to the value of `__s`. The data is copied, so there is no dependence on `__s` once the function returns.

Definition at line 873 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size()`.

```
4.18.3.13 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
_Base> __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::assign ( size_type __n,
_CharT __c ) [inline]
```

Set value to multiple characters.

Parameters

<code>__n</code>	Length of the resulting string.
<code>__c</code>	The character to use.

Returns

Reference to this string.

This function sets the value of this string to `__n` copies of character `__c`.

Definition at line 890 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::size()`.

```
4.18.3.14 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
        _Base> template<class _InputIterator, typename = std::RequireInputIter<_InputIterator>> __versa_string&
        __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::assign ( _InputIterator __first, _InputIterator __last )
        [inline]
```

Set value to a range of characters.

Parameters

<code>__first</code>	Iterator referencing the first character to append.
<code>__last</code>	Iterator marking the end of the range.

Returns

Reference to this string.

Sets value of string to characters in the range `[first,last)`.

Definition at line 909 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::replace()`.

```
4.18.3.15 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename >
        class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::assign (
        std::initializer_list<_CharT> __l ) [inline]
```

Set value to an `initializer_list` of characters.

Parameters

↩	The initializer_list of characters to assign.
↩	
↩	
↩	
/	

Returns

Reference to this string.

Definition at line 919 of file vstring.h.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::assign()`.

4.18.3.16 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> const_reference __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::at (size_type __n) const [inline]`

Provides access to the data contained in the string.

Parameters

↩	The index of the character to access.
<code>__n</code>	

Returns

Read-only (const) reference to the character.

Exceptions

<code>std::out_of_range</code>	If <code>__n</code> is an invalid index.
--------------------------------	--

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws `out_of_range` if the check fails.

Definition at line 577 of file vstring.h.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::size()`.

4.18.3.17 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> reference __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::at (size_type __n) [inline]`

Provides access to the data contained in the string.

Parameters

<code>__↔</code>	The index of the character to access.
<code>__n</code>	

Returns

Read/write reference to the character.

Exceptions

<code>std::out_of_range</code>	If <code>__n</code> is an invalid index.
--------------------------------	--

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws `out_of_range` if the check fails. Success results in unsharing the string.

Definition at line 599 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::size()`.

4.18.3.18 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> reference __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::back () [inline], [noexcept]`

Returns a read/write reference to the data at the last element of the string.

Definition at line 632 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::operator[]()`, and `__gnu_cxx::__versa_↔string<_CharT, _Traits, _Alloc, _Base >::size()`.

4.18.3.19 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> const_reference __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::back () const [inline], [noexcept]`

Returns a read-only (constant) reference to the data at the last element of the string.

Definition at line 640 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::operator[]()`, and `__gnu_cxx::__versa_↔string<_CharT, _Traits, _Alloc, _Base >::size()`.

4.18.3.20 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> iterator __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::begin () [inline], [noexcept]`

Returns a read/write iterator that points to the first character in the string. Unshares the string.

Definition at line 315 of file `vstring.h`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::crend()`, and `__gnu_cxx::__versa_↔string<_CharT, _Traits, _Alloc, _Base >::rend()`.

4.18.3.21 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> const_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::begin () const [inline], [noexcept]`

Returns a read-only (constant) iterator that points to the first character in the string.

Definition at line 326 of file `vstring.h`.

4.18.3.22 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> const _CharT* __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::c_str () const [inline], [noexcept]`

Return const pointer to null-terminated contents.

This is a handle to internal data. Do not modify or dire things may happen.

Definition at line 1647 of file `vstring.h`.

4.18.3.23 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::capacity () const [inline], [noexcept]`

Returns the total number of characters that the string can hold before needing to allocate more memory.

Definition at line 486 of file `vstring.h`.

Referenced by `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::push_back()`, and `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::shrink_to_fit()`.

4.18.3.24 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> const_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::cbegin () const [inline], [noexcept]`

Returns a read-only (constant) iterator that points to the first character in the string.

Definition at line 390 of file `vstring.h`.

4.18.3.25 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> const_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::cend () const [inline], [noexcept]`

Returns a read-only (constant) iterator that points one past the last character in the string.

Definition at line 398 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size()`.

4.18.3.26 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> void __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::clear () [inline], [noexcept]`

Erases the string, making it empty.

Definition at line 514 of file `vstring.h`.

4.18.3.27 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> int __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare (const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str) const [inline]`

Compare to a string.

Parameters

<code>__str</code>	String to compare against.
--------------------	----------------------------

Returns

Integer < 0, 0, or > 0.

Returns an integer < 0 if this string is ordered before `__str`, 0 if their values are equivalent, or > 0 if this string is ordered after `__str`. Determines the effective length `rlen` of the strings to compare as the smallest of `size()` and `str.size()`. The function then compares the two strings by calling `traits::compare(data(), str.data(), rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 2073 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::append()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::capacity()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::data()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::insert()`, `std::min()`, `std::move()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

Referenced by `__gnu_cxx::operator<()`, `__gnu_cxx::operator<=()`, `__gnu_cxx::operator==()`, `__gnu_cxx::operator>()`, and `__gnu_cxx::operator>=()`.

4.18.3.28 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> int __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::compare (size_type __pos, size_type __n, const __versa_string<_CharT, _Traits, _Alloc, _Base> & __str) const`

Compare substring to a string.

Parameters

<code>__pos</code>	Index of first character of substring.
<code>__n</code>	Number of characters in substring.
<code>__str</code>	String to compare against.

Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the `__n` characters starting at `__pos`. Returns an integer < 0 if the substring is ordered before `__str`, 0 if their values are equivalent, or > 0 if the substring is ordered after `__str`. Determines the effective length `rlen` of the strings to compare as the smallest of the length of the substring and `__str.size()`. The function then compares the two strings by calling `traits::compare(substring.data(), str.data(), rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

4.18.3.29 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> int __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::compare (size_type __pos1, size_type __n1, const __versa_string<_CharT, _Traits, _Alloc, _Base> & __str, size_type __pos2, size_type __n2) const`

Compare substring to a substring.

Parameters

<code>__pos1</code>	Index of first character of substring.
<code>__n1</code>	Number of characters in substring.
<code>__str</code>	String to compare against.
<code>__pos2</code>	Index of first character of substring of <code>str</code> .
<code>__n2</code>	Number of characters in substring of <code>str</code> .

Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the `__n1` characters starting at `__pos1`. Form the substring of `__str` from the `__n2` characters starting at `__pos2`. Returns an integer < 0 if this substring is ordered before the substring of `__str`, 0 if their values are equivalent, or > 0 if this substring is ordered after the substring of `__str`. Determines the effective length `rlen` of the strings to compare as the smallest of the lengths of the substrings. The function then compares the two strings by calling `traits::compare(substring.data(),str.substr(pos2,n2).data(),rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

4.18.3.30 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> int __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare (const _CharT * __s) const`

Compare to a C string.

Parameters

<code>__s</code>	C string to compare against.
------------------	------------------------------

Returns

Integer < 0, 0, or > 0.

Returns an integer < 0 if this string is ordered before `__s`, 0 if their values are equivalent, or > 0 if this string is ordered after `__s`. Determines the effective length `rlen` of the strings to compare as the smallest of `size()` and the length of a string constructed from `__s`. The function then compares the two strings by calling `traits::compare(data(),s,rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

4.18.3.31 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> int __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare (size_type __pos, size_type __n1, const _CharT * __s) const`

Compare substring to a C string.

Parameters

<code>__pos</code>	Index of first character of substring.
<code>__n1</code>	Number of characters in substring.
<code>__s</code>	C string to compare against.

Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the `__n1` characters starting at `__pos`. Returns an integer < 0 if the substring is ordered before `__s`, 0 if their values are equivalent, or > 0 if the substring is ordered after `__s`. Determines the effective length `rlen` of the strings to compare as the smallest of the length of the substring and the length of a string constructed from `__s`. The function then compares the two string by calling `traits::compare(substring.data(),s,rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

4.18.332 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> int __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::compare (size_type __pos, size_type __n1, const _CharT * __s, size_type __n2) const`

Compare substring against a character array.

Parameters

<code>__pos</code>	Index of first character of substring.
<code>__n1</code>	Number of characters in substring.
<code>__s</code>	character array to compare against.
<code>__n2</code>	Number of characters of s.

Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the `__n1` characters starting at `__pos`. Form a string from the first `__n2` characters of `__s`. Returns an integer < 0 if this substring is ordered before the string from `__s`, 0 if their values are equivalent, or > 0 if this substring is ordered after the string from `__s`. Determines the effective length `rlen` of the strings to compare as the smallest of the length of the substring and `__n2`. The function then compares the two strings by calling `traits::compare(substring.data(),__s,rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

NB: `__s` must have at least `n2` characters, `l0` has no special meaning.

4.18.333 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::copy (_CharT * __s, size_type __n, size_type __pos = 0) const`

Copy substring into C string.

Parameters

<code>__s</code>	C string to copy value into.
<code>__n</code>	Number of characters to copy.
<code>__pos</code>	Index of first character to copy.

Returns

Number of characters actually copied

Exceptions

<code>std::out_of_range</code>	If <code>pos > size()</code> .
--------------------------------	-----------------------------------

Copies up to `__n` characters starting at `__pos` into the C string `s`. If `__pos` is greater than `size()`, `out_of_range` is thrown.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::replace()`.

4.18.3.34 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> const_reverse_iterator __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::rbegin () const [inline], [noexcept]`

Returns a read-only (constant) reverse iterator that points to the last character in the string. Iteration is done in reverse element order.

Definition at line 407 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::end()`.

4.18.3.35 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> const_reverse_iterator __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::rend () const [inline], [noexcept]`

Returns a read-only (constant) reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order.

Definition at line 416 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::begin()`.

4.18.3.36 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> const _CharT* __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::data () const [inline], [noexcept]`

Return const pointer to contents.

This is a handle to internal data. Do not modify or dire things may happen.

Definition at line 1657 of file `vstring.h`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::compare()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::find()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::find_first_not_of()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::find_first_of()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::find_last_not_of()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::find_last_of()`, `std::operator<()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::rfind()`.

4.18.337 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::empty () const [inline], [noexcept]`

Returns true if the string is empty. Equivalent to `*this == ""`.

Definition at line 522 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

4.18.338 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> iterator __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::end () [inline], [noexcept]`

Returns a read/write iterator that points one past the last character in the string. Unshares the string.

Definition at line 334 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::crbegin()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::rbegin()`.

4.18.339 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> const_iterator __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::end () const [inline], [noexcept]`

Returns a read-only (constant) iterator that points one past the last character in the string.

Definition at line 345 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

4.18.340 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::erase (size_type __pos = 0, size_type __n = npos) [inline]`

Remove characters.

Parameters

<code>__pos</code>	Index of first character to remove (default 0).
<code>__n</code>	Number of characters to remove (default remainder).

Returns

Reference to this string.

Exceptions

<code>std::out_of_range</code>	If <code>__pos</code> is beyond the end of this string.
--------------------------------	---

Removes `__n` characters from this string starting at `__pos`. The length of the string is reduced by `__n`. If there are `< __n` characters to remove, the remainder of the string is truncated. If `__p` is beyond end of string, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1173 of file `vstring.h`.

```
4.18.3.41  template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
           _Base> iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::erase ( const_iterator __position )
           [inline]
```

Remove one character.

Parameters

<code>__position</code>	Iterator referencing the character to remove.
-------------------------	---

Returns

iterator referencing same location after removal.

Removes the character at `__position` from this string. The value of the string doesn't change if an error is thrown.

Definition at line 1190 of file `vstring.h`.

```
4.18.3.42  template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
           _Base> iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::erase ( const_iterator __first,
           const_iterator __last ) [inline]
```

Remove a range of characters.

Parameters

<code>__first</code>	Iterator referencing the first character to remove.
<code>__last</code>	Iterator referencing the end of the range.

Returns

Iterator referencing location of first after removal.

Removes the characters in the range `[first,last)` from this string. The value of the string doesn't change if an error is thrown.

Definition at line 1215 of file `vstring.h`.

4.18.3.43 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find (const _CharT * __s, size_type __pos, size_type __n) const`

Find position of a C substring.

Parameters

<code>__s</code>	C string to locate.
<code>__pos</code>	Index of character to search from.
<code>__n</code>	Number of characters from <code>__s</code> to search for.

Returns

Index of start of first occurrence.

Starting from `__pos`, searches forward for the first `__n` characters in `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_first_of()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::get_allocator()`.

4.18.3.44 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find (const __versa_string<_CharT, _Traits, _Alloc, _Base> & __str, size_type __pos = 0) const [inline], [noexcept]`

Find position of a string.

Parameters

<code>__str</code>	String to locate.
<code>__pos</code>	Index of character to search from (default 0).

Returns

Index of start of first occurrence.

Starting from `__pos`, searches forward for value of `__str` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 1693 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::data()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

4.18.3.45 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find (const _CharT * __s, size_type __pos = 0) const [inline]`

Find position of a C string.

Parameters

<code>__s</code>	C string to locate.
<code>__pos</code>	Index of character to search from (default 0).

Returns

Index of start of first occurrence.

Starting from `__pos`, searches forward for the value of `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 1708 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find()`.

```
4.18.3.46  template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
           _Base> size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base>::find ( _CharT __c, size_type
           __pos = 0 ) const    [noexcept]
```

Find position of a character.

Parameters

<code>__c</code>	Character to locate.
<code>__pos</code>	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from `__pos`, searches forward for `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

```
4.18.3.47  template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename >
           class _Base> size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base>::find_first_not_of (
           const __versa_string< _CharT, _Traits, _Alloc, _Base> & __str, size_type __pos = 0 ) const    [inline],
           [noexcept]
```

Find position of a character not in string.

Parameters

<code>__str</code>	String containing characters to avoid.
<code>__pos</code>	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from `__pos`, searches forward for a character not contained in `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1925 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::data()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::size()`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::find_first_not_of()`.

4.18.3.48 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::find_first_not_of (const _CharT * __s, size_type __pos, size_type __n) const`

Find position of a character not in C substring.

Parameters

<code>__s</code>	C string containing characters to avoid.
<code>__pos</code>	Index of character to search from.
<code>__n</code>	Number of characters from <code>s</code> to consider.

Returns

Index of first occurrence.

Starting from `__pos`, searches forward for a character not contained in the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

4.18.3.49 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::find_first_not_of (const _CharT * __s, size_type __pos = 0) const [inline]`

Find position of a character not in C string.

Parameters

<code>__s</code>	C string containing characters to avoid.
<code>__pos</code>	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from `__pos`, searches forward for a character not contained in `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1956 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_first_not_of()`.

4.18.3.50 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_first_not_of (_CharT __c, size_type __pos = 0) const [noexcept]`

Find position of a different character.

Parameters

<code>__c</code>	Character to avoid.
<code>__pos</code>	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from `__pos`, searches forward for a character other than `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

4.18.3.51 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_first_of (const __versa_string<_CharT, _Traits, _Alloc, _Base> & __str, size_type __pos = 0) const [inline], [noexcept]`

Find position of a character of string.

Parameters

<code>__str</code>	String containing characters to locate.
<code>__pos</code>	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from `__pos`, searches forward for one of the characters of `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1798 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::data()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_first_of()`.

4.18.3.52 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::find_first_of (const _CharT * __s, size_type __pos, size_type __n) const`

Find position of a character of C substring.

Parameters

<code>__s</code>	String containing characters to locate.
<code>__pos</code>	Index of character to search from.
<code>__n</code>	Number of characters from <code>s</code> to search for.

Returns

Index of first occurrence.

Starting from `__pos`, searches forward for one of the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

4.18.3.53 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::find_first_of (const _CharT * __s, size_type __pos = 0) const [inline]`

Find position of a character of C string.

Parameters

<code>__s</code>	String containing characters to locate.
<code>__pos</code>	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from `__pos`, searches forward for one of the characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1828 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::find_first_of()`.

4.18.3.54 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::find_first_of (_CharT __c, size_type __pos = 0) const [inline], [noexcept]`

Find position of a character.

Parameters

<code>__c</code>	Character to locate.
<code>__pos</code>	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from `__pos`, searches forward for the character `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Note: equivalent to `find(c, pos)`.

Definition at line 1847 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find()`.

```
4.18.3.55  template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
           _Base> size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base>::find_last_not_of ( const
           __versa_string< _CharT, _Traits, _Alloc, _Base> & __str, size_type __pos = npow ) const    [inline],
           [noexcept]
```

Find last position of a character not in string.

Parameters

<code>__str</code>	String containing characters to avoid.
<code>__pos</code>	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from `__pos`, searches backward for a character not contained in `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1988 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::data()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_last_not_of()`.

```
4.18.3.56  template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
           _Base> size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base>::find_last_not_of ( const _CharT *
           __s, size_type __pos, size_type __n ) const
```

Find last position of a character not in C substring.

Parameters

<code>__s</code>	C string containing characters to avoid.
<code>__pos</code>	Index of character to search back from.
<code>__n</code>	Number of characters from <code>s</code> to consider.

Returns

Index of last occurrence.

Starting from `__pos`, searches backward for a character not contained in the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

```
4.18.3.57 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
_Base> size_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_last_not_of ( const _CharT *
__s, size_type __pos = npos ) const [inline]
```

Find last position of a character not in C string.

Parameters

<code>__s</code>	C string containing characters to avoid.
<code>__pos</code>	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from `__pos`, searches backward for a character not contained in `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 2019 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_last_not_of()`.

```
4.18.3.58 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
_Base> size_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_last_not_of ( _CharT __c,
size_type __pos = npos ) const [noexcept]
```

Find last position of a different character.

Parameters

<code>__c</code>	Character to avoid.
<code>__pos</code>	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from `__pos`, searches backward for a character other than `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

```
4.18.3.59 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename >
class _Base> size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_last_of ( const
__versa_string< _CharT, _Traits, _Alloc, _Base > & __str, size_type __pos = npos ) const [inline],
[noexcept]
```

Find last position of a character of string.

Parameters

<code>__str</code>	String containing characters to locate.
<code>__pos</code>	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from `__pos`, searches backward for one of the characters of `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1862 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::data()`, and `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size()`.

Referenced by `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_last_of()`.

```
4.18.3.60 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
_Base> size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_last_of ( const _CharT * __s,
size_type __pos, size_type __n ) const
```

Find last position of a character of C substring.

Parameters

<code>__s</code>	C string containing characters to locate.
<code>__pos</code>	Index of character to search back from.
<code>__n</code>	Number of characters from <code>s</code> to search for.

Returns

Index of last occurrence.

Starting from `__pos`, searches backward for one of the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

4.18.3.61 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_last_of(const _CharT* __s, size_type __pos = npos) const [inline]`

Find last position of a character of C string.

Parameters

<code>__s</code>	C string containing characters to locate.
<code>__pos</code>	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from `__pos`, searches backward for one of the characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1892 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_last_of()`.

4.18.3.62 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_last_of(_CharT __c, size_type __pos = npos) const [inline], [noexcept]`

Find last position of a character.

Parameters

<code>__c</code>	Character to locate.
<code>__pos</code>	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from `__pos`, searches backward for `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Note: equivalent to `rfind(c, pos)`.

Definition at line 1911 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::rfind()`.

4.18.3.63 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> reference __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::front () [inline], [noexcept]`

Returns a read/write reference to the data at the first element of the string.

Definition at line 616 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator[]()`.

4.18.3.64 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> const_reference __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::front () const [inline], [noexcept]`

Returns a read-only (constant) reference to the data at the first element of the string.

Definition at line 624 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator[]()`.

4.18.3.65 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> allocator_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::get_allocator () const [inline], [noexcept]`

Return copy of allocator used to construct this string.

Definition at line 1664 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find()`.

4.18.3.66 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::insert (const_iterator __p, size_type __n, _CharT __c) [inline]`

Insert multiple characters.

Parameters

<code>__p</code>	Const_iterator referencing location in string to insert at.
<code>__n</code>	Number of characters to insert
<code>__c</code>	The character to insert.

Returns

Iterator referencing the first inserted char.

Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Inserts `__n` copies of character `__c` starting at the position referenced by iterator `__p`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 940 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::replace()`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::compare()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::insert()`.

4.18.3.67 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> template<class _InputIterator, typename = std::RequireInputIter<_InputIterator>> iterator __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::insert (const_iterator __p, _InputIterator __beg, _InputIterator __end) [inline]`

Insert a range of characters.

Parameters

<code>__p</code>	Const_iterator referencing location in string to insert at.
<code>__beg</code>	Start of range.
<code>__end</code>	End of range.

Returns

Iterator referencing the first inserted char.

Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Inserts characters in range `[beg,end)`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 984 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::insert()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::replace()`.

4.18.3.68 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> iterator __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::insert (const_iterator __p, std::initializer_list<_CharT> __l) [inline]`

Insert an `initializer_list` of characters.

Parameters

<code>_↔ _p</code>	Const_iterator referencing location in string to insert at.
<code>_↔ _l</code>	The initializer_list of characters to insert.

Returns

Iterator referencing the first inserted char.

Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Definition at line 1020 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::insert()`.

4.18.3.69 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::insert (size_type __pos1, const __versa_string<_CharT, _Traits, _Alloc, _Base> & __str) [inline]`

Insert value of a string.

Parameters

<code>__pos1</code>	Iterator referencing location in string to insert at.
<code>__str</code>	The string to insert.

Returns

Reference to this string.

Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Inserts value of `__str` starting at `__pos1`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1037 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::replace()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

4.18.3.70 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::insert (size_type __pos1, const __versa_string<_CharT, _Traits, _Alloc, _Base > & __str, size_type __pos2, size_type __n) [inline]`

Insert a substring.

Parameters

<code>__pos1</code>	Iterator referencing location in string to insert at.
<code>__str</code>	The string to insert.
<code>__pos2</code>	Start of characters in str to insert.
<code>__n</code>	Number of characters to insert.

Returns

Reference to this string.

Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
<code>std::out_of_range</code>	If <code>__pos1 > size()</code> or <code>__pos2 > __str.size()</code> .

Starting at `__pos1`, insert `__n` character of `__str` beginning with `__pos2`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos1` is beyond the end of this string or `__pos2` is beyond the end of `__str`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1060 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::replace()`.

4.18.3.71 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::insert (size_type __pos, const _CharT * __s, size_type __n) [inline]`

Insert a C substring.

Parameters

<code>__pos</code>	Iterator referencing location in string to insert at.
<code>__s</code>	The C string to insert.
<code>__n</code>	The number of characters to insert.

Returns

Reference to this string.

Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
<code>std::out_of_range</code>	If <code>__pos</code> is beyond the end of this string.

Inserts the first `__n` characters of `__s` starting at `__pos`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos` is beyond `end()`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1083 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::replace()`.

4.18.3.72 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::insert (size_type __pos, const _CharT* __s) [inline]`

Insert a C string.

Parameters

<code>__pos</code>	Iterator referencing location in string to insert at.
<code>__s</code>	The C string to insert.

Returns

Reference to this string.

Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
<code>std::out_of_range</code>	If <code>__pos</code> is beyond the end of this string.

Inserts the first `__n` characters of `__s` starting at `__pos`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos` is beyond `end()`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1102 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::replace()`.

4.18.3.73 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::insert (size_type __pos, size_type __n, _CharT __c) [inline]`

Insert multiple characters.

Parameters

<code>__pos</code>	Index in string to insert at.
<code>__n</code>	Number of characters to insert
<code>__c</code>	The character to insert.

Returns

Reference to this string.

Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
<code>std::out_of_range</code>	If <code>__pos</code> is beyond the end of this string.

Inserts `__n` copies of character `__c` starting at index `__pos`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos > length()`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1126 of file `vstring.h`.

```
4.18.3.74 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
    _Base> iterator __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::insert( const_iterator __p, _CharT
    __c ) [inline]
```

Insert one character.

Parameters

<code>__p</code>	Iterator referencing position in string to insert at.
<code>__c</code>	The character to insert.

Returns

Iterator referencing newly inserted char.

Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Inserts character `__c` at position referenced by `__p`. If adding character causes the length to exceed `max_size()`, `length_error` is thrown. If `__p` is beyond end of string, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1145 of file `vstring.h`.

4.18.3.75 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::length () const [inline], [noexcept]`

Returns the number of characters in the string, not including any null-termination.

Definition at line 431 of file `vstring.h`.

4.18.3.76 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::max_size () const [inline], [noexcept]`

Returns the `size()` of the largest possible string.

Definition at line 436 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::resize()`.

4.18.3.77 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator+=(const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str) [inline]`

Append a string to this string.

Parameters

<code>__str</code>	The string to append.
--------------------	-----------------------

Returns

Reference to this string.

Definition at line 651 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append()`.

4.18.3.78 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator+=(const _CharT * __s) [inline]`

Append a C string.

Parameters

<code>__s</code>	The C string to append.
------------------	-------------------------

Returns

Reference to this string.

Definition at line 660 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::append()`.

4.18.3.79 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::operator+=(_CharT _c) [inline]`

Append a character.

Parameters

<code>_c</code>	The character to append.
-----------------	--------------------------

Returns

Reference to this string.

Definition at line 669 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::push_back()`.

4.18.3.80 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::operator+=(std::initializer_list<_CharT > _l) [inline]`

Append an `initializer_list` of characters.

Parameters

<code>_l</code>	The <code>initializer_list</code> of characters to be appended.
-----------------	---

Returns

Reference to this string.

Definition at line 682 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::append()`.

4.18.3.81 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__str) [inline]`

Assign the value of *str* to this string.

Parameters

<code>__str</code>	Source string.
--------------------	----------------

Definition at line 256 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::assign()`.

4.18.3.82 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator= (__versa_string< _CharT, _Traits, _Alloc, _Base > &&__str) [inline], [noexcept]`

String move assignment operator.

Parameters

<code>__str</code>	Source string.
--------------------	----------------

The contents of `__str` are moved into this string (without copying). `__str` is a valid, but unspecified string.

Definition at line 268 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::swap()`.

4.18.3.83 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator= (std::initializer_list< _CharT > __l) [inline]`

Set value to string constructed from initializer list.

Parameters

<code>↵</code>	<code>std::initializer_list.</code>
<code>__↵</code>	
<code>↵</code>	
<code>__↵</code>	
<code>/</code>	

Definition at line 280 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::assign()`.

4.18.3.84 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::operator= (const _CharT * __s) [inline]`

Copy contents of `__s` into this string.

Parameters

<code>__s</code>	Source null-terminated string.
------------------	--------------------------------

Definition at line 292 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::assign()`.

4.18.3.85 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::operator= (_CharT __c) [inline]`

Set value to string of length 1.

Parameters

<code>__c</code>	Source character.
------------------	-------------------

Assigning to a character makes this string length 1 and `(*this)[0] == __c`.

Definition at line 303 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::assign()`.

4.18.3.86 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> const_reference __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::operator[] (size_type __pos) const [inline], [noexcept]`

Subscript access to the data contained in the string.

Parameters

<code>__pos</code>	The index of the character to access.
--------------------	---------------------------------------

Returns

Read-only (constant) reference to the character.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and `out_of_range` lookups are not defined. (For checked lookups see `at()`.)

Definition at line 537 of file vstring.h.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::size()`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::back()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::front()`.

4.18.3.87 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> reference __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::operator[] (size_type __pos) [inline], [noexcept]`

Subscript access to the data contained in the string.

Parameters

<code>__pos</code>	The index of the character to access.
--------------------	---------------------------------------

Returns

Read/write reference to the character.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and `out_of_range` lookups are not defined. (For checked lookups see `at()`.) Unshares the string.

Definition at line 554 of file vstring.h.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::size()`.

4.18.3.88 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> void __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::pop_back () [inline]`

Remove the last character.

The string must be non-empty.

Definition at line 1235 of file vstring.h.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::size()`.

4.18.3.89 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> void __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::push_back (_CharT __c) [inline]`

Append a single character.

Parameters

<code>__c</code>	Character to append.
------------------	----------------------

Definition at line 788 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::capacity()`, and `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size()`.

Referenced by `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator+=()`.

4.18.3.90 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> reverse_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::rbegin ()`
`[inline], [noexcept]`

Returns a read/write reverse iterator that points to the last character in the string. Iteration is done in reverse element order. Unshares the string.

Definition at line 354 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::end()`.

4.18.3.91 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> const_reverse_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::rbegin ()`
`const [inline], [noexcept]`

Returns a read-only (constant) reverse iterator that points to the last character in the string. Iteration is done in reverse element order.

Definition at line 363 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::end()`.

4.18.3.92 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> reverse_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::rend ()`
`[inline], [noexcept]`

Returns a read/write reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order. Unshares the string.

Definition at line 372 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::begin()`.

4.18.3.93 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> const_reverse_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::rend () const`
`[inline], [noexcept]`

Returns a read-only (constant) reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order.

Definition at line 381 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::begin()`.

4.18.3.94 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace (size_type __pos, size_type __n, const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str)` `[inline]`

Replace characters with value from another string.

Parameters

<code>__pos</code>	Index of first character to replace.
<code>__n</code>	Number of characters to be replaced.
<code>__str</code>	String to insert.

Returns

Reference to this string.

Exceptions

<code>std::out_of_range</code>	If <code>__pos</code> is beyond the end of this string.
<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .

Removes the characters in the range `[pos,pos+n)` from this string. In place, the value of `__str` is inserted. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of the result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1257 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::size()`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::append()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::assign()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::insert()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::replace()`.

4.18.3.95 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::replace (size_type __pos1, size_type __n1, const __versa_string<_CharT, _Traits, _Alloc, _Base > & __str, size_type __pos2, size_type __n2) [inline]`

Replace characters with value from another string.

Parameters

<code>__pos1</code>	Index of first character to replace.
<code>__n1</code>	Number of characters to be replaced.
<code>__str</code>	String to insert.
<code>__pos2</code>	Index of first character of <code>str</code> to use.
<code>__n2</code>	Number of characters from <code>str</code> to use.

Returns

Reference to this string.

Exceptions

<code>std::out_of_range</code>	If <code>__pos1 > size()</code> or <code>__pos2 > str.size()</code> .
<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .

Removes the characters in the range `[pos1,pos1 + n)` from this string. In place, the value of `__str` is inserted. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of the result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1280 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT,_Traits,_Alloc,_Base>::replace()`.

4.18.3.96 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT,_Traits,_Alloc,_Base>::replace (size_type __pos, size_type __n1, const _CharT * __s, size_type __n2) [inline]`

Replace characters with value of a C substring.

Parameters

<code>__pos</code>	Index of first character to replace.
<code>__n1</code>	Number of characters to be replaced.
<code>__s</code>	C string to insert.
<code>__n2</code>	Number of characters from <code>__s</code> to use.

Returns

Reference to this string.

Exceptions

<code>std::out_of_range</code>	If <code>__pos1 > size()</code> .
<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .

Removes the characters in the range `[pos,pos + n1)` from this string. In place, the first `__n2` characters of `__s` are inserted, or all of `__s` if `__n2` is too large. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1308 of file `vstring.h`.

4.18.3.97 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT,_Traits,_Alloc,_Base>::replace (size_type __pos, size_type __n1, const _CharT * __s) [inline]`

Replace characters with value of a C string.

Parameters

<code>__pos</code>	Index of first character to replace.
<code>__n1</code>	Number of characters to be replaced.
<code>__s</code>	C string to insert.

Returns

Reference to this string.

Exceptions

<code>std::out_of_range</code>	If <code>__pos > size()</code> .
<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .

Removes the characters in the range `[pos, pos + n1)` from this string. In place, the characters of `__s` are inserted. If `pos` is beyond end of string, `out_of_range` is thrown. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1332 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::replace()`.

4.18.3.98 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::replace (size_type __pos, size_type __n1, size_type __n2, _CharT __c) [inline]`

Replace characters with multiple characters.

Parameters

<code>__pos</code>	Index of first character to replace.
<code>__n1</code>	Number of characters to be replaced.
<code>__n2</code>	Number of characters to insert.
<code>__c</code>	Character to insert.

Returns

Reference to this string.

Exceptions

<code>std::out_of_range</code>	If <code>__pos > size()</code> .
<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .

Removes the characters in the range `[pos, pos + n1)` from this string. In place, `__n2` copies of `__c` are inserted. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1356 of file `vstring.h`.

4.18.3.99 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::replace (const_iterator __i1, const_iterator __i2, const __versa_string<_CharT, _Traits, _Alloc, _Base > & __str) [inline]`

Replace range of characters with string.

Parameters

<code>__i1</code>	Iterator referencing start of range to replace.
<code>__i2</code>	Iterator referencing end of range to replace.
<code>__str</code>	String value to insert.

Returns

Reference to this string.

Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Removes the characters in the range `[i1, i2)`. In place, the value of `__str` is inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1375 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::replace()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::size()`.

4.18.3.100 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::replace (const_iterator __i1, const_iterator __i2, const _CharT * __s, size_type __n) [inline]`

Replace range of characters with C substring.

Parameters

<code>__i1</code>	Iterator referencing start of range to replace.
<code>__i2</code>	Iterator referencing end of range to replace.
<code>__s</code>	C string value to insert.
<code>__n</code>	Number of characters from <code>s</code> to insert.

Returns

Reference to this string.

Exceptions

<i>std::length_error</i>	If new length exceeds <code>max_size()</code> .
--------------------------	---

Removes the characters in the range `[i1,i2)`. In place, the first *n* characters of `__s` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1398 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::replace()`.

4.18.3.101 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::replace (const_iterator __i1, const_iterator __i2, const _CharT* __s) [inline]`

Replace range of characters with C string.

Parameters

<code>__i1</code>	Iterator referencing start of range to replace.
<code>__i2</code>	Iterator referencing end of range to replace.
<code>__s</code>	C string value to insert.

Returns

Reference to this string.

Exceptions

<i>std::length_error</i>	If new length exceeds <code>max_size()</code> .
--------------------------	---

Removes the characters in the range `[i1,i2)`. In place, the characters of `__s` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1424 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::replace()`.


```
4.18.3.102 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename >
class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::replace (
    const_iterator __i1, const_iterator __i2, size_type __n, _CharT __c ) [inline]
```

Replace range of characters with multiple characters.

Parameters

<code>__i1</code>	Iterator referencing start of range to replace.
<code>__i2</code>	Iterator referencing end of range to replace.
<code>__n</code>	Number of characters to insert.
<code>__c</code>	Character to insert.

Returns

Reference to this string.

Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Removes the characters in the range `[i1,i2)`. In place, `__n` copies of `__c` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1449 of file `vstring.h`.

```
4.18.3.103 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
    _Base> template<class _InputIterator, typename = std:: RequireInputIter<_InputIterator>>> __versa_string&
    __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace( const_iterator __i1, const_iterator __i2,
    _InputIterator __k1, _InputIterator __k2 ) [inline]
```

Replace range of characters with range.

Parameters

<code>__i1</code>	Iterator referencing start of range to replace.
<code>__i2</code>	Iterator referencing end of range to replace.
<code>__k1</code>	Iterator referencing start of range to insert.
<code>__k2</code>	Iterator referencing end of range to insert.

Returns

Reference to this string.

Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Removes the characters in the range `[i1,i2)`. In place, characters in the range `[k1,k2)` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1478 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::replace()`.

```
4.18.3.104 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename >
class _Base> __versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::replace (
const_iterator __i1, const_iterator __i2, std::initializer_list<_CharT> __l ) [inline]
```

Replace range of characters with `initializer_list`.

Parameters

<code>__i1</code>	Iterator referencing start of range to replace.
<code>__i2</code>	Iterator referencing end of range to replace.
<code>__l</code>	The <code>initializer_list</code> of characters to insert.

Returns

Reference to this string.

Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Removes the characters in the range `[i1,i2)`. In place, characters in the range `[k1,k2)` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1582 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::copy()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::replace()`.

```
4.18.3.105 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
_Base> void __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::reserve ( size_type __res_arg = 0 )
[inline]
```

Attempt to preallocate enough memory for specified number of characters.

Parameters

<code>__res_arg</code>	Number of characters required.
------------------------	--------------------------------

Exceptions

<code>std::length_error</code>	If <code>__res_arg</code> exceeds <code>max_size()</code> .
--------------------------------	---

This function attempts to reserve enough memory for the string to hold the specified number of characters. If the number requested is more than `max_size()`, `length_error` is thrown.

The advantage of this function is that if optimal code is a necessity and the user can determine the string length that will be required, the user can reserve the memory in advance, and thus prevent a possible reallocation of memory and copying of string data.

Definition at line 507 of file `vstring.h`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::shrink_to_fit()`.

4.18.3.106 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> void __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::resize (size_type __n, _CharT __c)`

Resizes the string to the specified number of characters.

Parameters

<code>__n</code>	Number of characters the string should contain.
<code>__c</code>	Character to fill any new elements.

This function will resize the string to the specified number of characters. If the number is smaller than the string's current size the string is truncated, otherwise the string is extended and new elements are set to `__c`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::max_size()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::resize()`.

4.18.3.107 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> void __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::resize (size_type __n)`
`[inline]`

Resizes the string to the specified number of characters.

Parameters

<code>__n</code>	Number of characters the string should contain.
------------------	---

This function will resize the string to the specified length. If the new size is smaller than the string's current size the string is truncated, otherwise the string is extended and new characters are default-constructed. For basic types such as `char`, this means setting them to 0.

Definition at line 463 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::resize()`.

4.18.3.108 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::rfind (const __versa_string<_CharT, _Traits, _Alloc, _Base > & __str, size_type __pos = npos) const` `[inline], [noexcept]`

Find last position of a string.

Parameters

<code>__str</code>	String to locate.
<code>__pos</code>	Index of character to search back from (default end).

Returns

Index of start of last occurrence.

Starting from `__pos`, searches backward for value of `__str` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 1738 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::data()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::size()`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::find_last_of()`, and `__gnu_cxx::__↵versa_string<_CharT, _Traits, _Alloc, _Base >::rfind()`.

4.18.3.109 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::rfind (const _CharT * __s, size_type __pos, size_type __n) const`

Find last position of a C substring.

Parameters

<code>__s</code>	C string to locate.
<code>__pos</code>	Index of character to search back from.
<code>__n</code>	Number of characters from <code>s</code> to search for.

Returns

Index of start of last occurrence.

Starting from `__pos`, searches backward for the first `__n` characters in `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

4.18.3.110 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::rfind (const _CharT * __s, size_type __pos = npos) const [inline]`

Find last position of a C string.

Parameters

<code>__s</code>	C string to locate.
<code>__pos</code>	Index of character to start search at (default end).

Returns

Index of start of last occurrence.

Starting from `__pos`, searches backward for the value of `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 1768 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::rfind()`.

4.18.3.111 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::rfind (_CharT __c, size_type __pos = npos) const [noexcept]`

Find last position of a character.

Parameters

<code>__c</code>	Character to locate.
<code>__pos</code>	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from `__pos`, searches backward for `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

4.18.3.112 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> void __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::shrink_to_fit () [inline], [noexcept]`

A non-binding request to reduce capacity() to size().

Definition at line 469 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::capacity()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::reserve()`, and `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size()`.

4.18.3.113 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size() const [inline], [noexcept]`

Returns the number of characters in the string, not including any null-termination.

Definition at line 425 of file `vstring.h`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::append()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::assign()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::at()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::back()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::cend()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::compare()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::empty()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::end()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_first_not_of()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_first_of()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_last_not_of()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_last_of()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::insert()`, `std::operator<<()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::operator[]()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::pop_back()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::push_back()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::replace()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::rfind()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::shrink_to_fit()`.

4.18.3.114 `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::substr(size_type __pos = 0, size_type __n = npos) const [inline]`

Get a substring.

Parameters

<code>__pos</code>	Index of first character (default 0).
<code>__n</code>	Number of characters in substring (default remainder).

Returns

The new string.

Exceptions

<code>std::out_of_range</code>	If <code>pos > size()</code> .
--------------------------------	-----------------------------------

Construct and return a new string using the `__n` characters starting at `__pos`. If the string is too short, use the remainder of the characters. If `__pos` is beyond the end of the string, `out_of_range` is thrown.

Definition at line 2052 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::__versa_string()`.

```
4.18.3.115 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
_Base> void __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::swap ( __versa_string< _CharT,
_Traits, _Alloc, _Base > & __s ) [inline], [noexcept]
```

Swap contents with another string.

Parameters

<code>__↔</code>	String to swap with.
<code>__s</code>	

Exchanges the contents of this string with that of `__s` in constant time.

Definition at line 1636 of file `vstring.h`.

Referenced by `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::assign()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator=()`, and `__gnu_cxx::swap()`.

4.18.4 Member Data Documentation

```
4.18.4.1 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class
_Base> const size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::npos [static]
```

Value returned by various member functions when they fail.

Definition at line 81 of file `vstring.h`.

The documentation for this class was generated from the following file:

- [vstring.h](#)

4.19 __gnu_cxx::Caster<_ToType> Struct Template Reference

Public Types

- `typedef _ToType::element_type * type`

4.19.1 Detailed Description

```
template<typename _ToType>
struct __gnu_cxx::Caster<_ToType>
```

These functions are here to allow containers to support non standard pointer types. For normal pointers, these resolve to the use of the standard cast operation. For other types the functions will perform the appropriate cast to/from the custom pointer class so long as that class meets the following conditions: 1) has a typedef `element_type` which names the type it points to. 2) has a `get()` const method which returns `element_type*`. 3) has a constructor which can take one `element_type*` argument. This type supports the semantics of the pointer cast operators (below.)

Definition at line 52 of file `cast.h`.

The documentation for this struct was generated from the following file:

- [cast.h](#)

4.20 `__gnu_cxx::Char_types<_CharT>` Struct Template Reference

Public Types

- typedef unsigned long **int_type**
- typedef [std::streamoff](#) **off_type**
- typedef [std::streampos](#) **pos_type**
- typedef [std::mbstate_t](#) **state_type**

4.20.1 Detailed Description

```
template<typename _CharT>  
struct __gnu_cxx::Char_types<_CharT>
```

Mapping from character type to associated types.

Note

This is an implementation class for the generic version of `char_traits`. It defines `int_type`, `off_type`, `pos_type`, and `state_type`. By default these are unsigned long, `streamoff`, `streampos`, and `mbstate_t`. Users who need a different set of types, but who don't need to change the definitions of any function defined in `char_traits`, can specialize `__gnu_cxx::Char_types` while leaving `__gnu_cxx::char_traits` alone.

Definition at line 58 of file `char_traits.h`.

The documentation for this struct was generated from the following file:

- [char_traits.h](#)

4.21 `__gnu_cxx::ExtPtr_allocator<_Tp>` Class Template Reference

Public Types

- typedef [_Pointer_adapter<_Relative_pointer_impl<const _Tp>>](#) **const_pointer**
- typedef const _Tp & **const_reference**
- typedef [std::ptrdiff_t](#) **difference_type**
- typedef [_Pointer_adapter<_Relative_pointer_impl<_Tp>>](#) **pointer**
- typedef _Tp & **reference**
- typedef [std::size_t](#) **size_type**
- typedef _Tp **value_type**

Public Member Functions

- **_ExtPtr_allocator** (const [_ExtPtr_allocator](#) &__rarg) noexcept
- template<typename _Up >
 _ExtPtr_allocator (const [_ExtPtr_allocator](#)< _Up > &__rarg) noexcept
- const [std::allocator](#)< _Tp > & **_M_getUnderlyingImp** () const
- [pointer](#) **address** (reference __x) const noexcept
- [const_pointer](#) **address** (const_reference __x) const noexcept
- [pointer](#) **allocate** (size_type __n, void *__hint=0)
- template<typename _Up, typename... _Args>
 void **construct** (_Up *__p, _Args &&...__args)
- template<typename... _Args>
 void **construct** ([pointer](#) __p, _Args &&...__args)
- void **deallocate** ([pointer](#) __p, size_type __n)
- template<typename _Up >
 void **destroy** (_Up *__p)
- void **destroy** ([pointer](#) __p)
- size_type **max_size** () const noexcept
- template<typename _Up >
 bool **operator!=** (const [_ExtPtr_allocator](#)< _Up > &__rarg)
- bool **operator!=** (const [_ExtPtr_allocator](#) &__rarg)
- template<typename _Up >
 bool **operator==** (const [_ExtPtr_allocator](#)< _Up > &__rarg)
- bool **operator==** (const [_ExtPtr_allocator](#) &__rarg)

Friends

- template<typename _Up >
 void **swap** ([_ExtPtr_allocator](#)< _Up > &, [_ExtPtr_allocator](#)< _Up > &)

4.21.1 Detailed Description

```
template<typename _Tp>
class __gnu_cxx::_ExtPtr_allocator< _Tp >
```

An example allocator which uses a non-standard pointer type.

This allocator specifies that containers use a 'relative pointer' as it's pointer type. (See `ext/pointer.h`) Memory allocation in this example is still performed using `std::allocator`.

Definition at line 56 of file `extptr_allocator.h`.

The documentation for this class was generated from the following file:

- [extptr_allocator.h](#)

4.22 `__gnu_cxx::_Invalid_type` Struct Reference

4.22.1 Detailed Description

The specialization on this type helps resolve the problem of reference to void, and eliminates the need to specialize `_Pointer_adapter` for cases of `void*`, `const void*`, and so on.

Definition at line 213 of file `pointer.h`.

The documentation for this struct was generated from the following file:

- [pointer.h](#)

4.23 `__gnu_cxx::_Pointer_adapter<_Storage_policy>` Class Template Reference

Inherits `_Storage_policy`.

Public Types

- typedef `std::ptrdiff_t` **difference_type**
- typedef `_Storage_policy::element_type` **element_type**
- typedef `std::random_access_iterator_tag` **iterator_category**
- typedef `_Pointer_adapter` **pointer**
- typedef `_Reference_type< element_type >::reference` **reference**
- typedef `_Unqualified_type< element_type >::type` **value_type**

Public Member Functions

- `_Pointer_adapter` (`element_type *` __arg=0)
- `_Pointer_adapter` (const `_Pointer_adapter` &__arg)
- template<typename `_Up` >
 `_Pointer_adapter` (`_Up *` __arg)
- template<typename `_Up` >
 `_Pointer_adapter` (const `_Pointer_adapter`< `_Up` > &__arg)
- **operator __unspecified_bool_type** () const
- bool **operator!** () const
- reference **operator*** () const
- `_Pointer_adapter` & **operator++** ()
- `_Pointer_adapter` **operator++** (int)
- `_Pointer_adapter` & **operator+=** (short __offset)
- `_Pointer_adapter` & **operator+=** (unsigned short __offset)
- `_Pointer_adapter` & **operator+=** (int __offset)
- `_Pointer_adapter` & **operator+=** (unsigned int __offset)
- `_Pointer_adapter` & **operator+=** (long __offset)
- `_Pointer_adapter` & **operator+=** (unsigned long __offset)
- template<typename `_Up` >
 `std::ptrdiff_t` **operator-** (const `_Pointer_adapter`< `_Up` > &__rhs) const

- [_Pointer_adapter](#) & **operator--** ()
- [_Pointer_adapter](#) **operator--** (int)
- [_Pointer_adapter](#) & **operator-=** (short __offset)
- [_Pointer_adapter](#) & **operator-=** (unsigned short __offset)
- [_Pointer_adapter](#) & **operator-=** (int __offset)
- [_Pointer_adapter](#) & **operator-=** (unsigned int __offset)
- [_Pointer_adapter](#) & **operator-=** (long __offset)
- [_Pointer_adapter](#) & **operator-=** (unsigned long __offset)
- element_type * **operator->** () const
- [_Pointer_adapter](#) & **operator=** (const [_Pointer_adapter](#) & __arg)
- template<typename _Up >
 [_Pointer_adapter](#) & **operator=** (const [_Pointer_adapter](#)< _Up > & __arg)
- template<typename _Up >
 [_Pointer_adapter](#) & **operator=** (_Up * __arg)
- reference **operator[]** (std::ptrdiff_t __index) const

Friends

- [_Pointer_adapter](#) **operator+** (const [_Pointer_adapter](#) & __lhs, short __offset)
- [_Pointer_adapter](#) **operator+** (short __offset, const [_Pointer_adapter](#) & __rhs)
- [_Pointer_adapter](#) **operator+** (const [_Pointer_adapter](#) & __lhs, unsigned short __offset)
- [_Pointer_adapter](#) **operator+** (unsigned short __offset, const [_Pointer_adapter](#) & __rhs)
- [_Pointer_adapter](#) **operator+** (const [_Pointer_adapter](#) & __lhs, int __offset)
- [_Pointer_adapter](#) **operator+** (int __offset, const [_Pointer_adapter](#) & __rhs)
- [_Pointer_adapter](#) **operator+** (const [_Pointer_adapter](#) & __lhs, unsigned int __offset)
- [_Pointer_adapter](#) **operator+** (unsigned int __offset, const [_Pointer_adapter](#) & __rhs)
- [_Pointer_adapter](#) **operator+** (const [_Pointer_adapter](#) & __lhs, long __offset)
- [_Pointer_adapter](#) **operator+** (long __offset, const [_Pointer_adapter](#) & __rhs)
- [_Pointer_adapter](#) **operator+** (unsigned long __offset, const [_Pointer_adapter](#) & __rhs)
- [_Pointer_adapter](#) **operator+** (const [_Pointer_adapter](#) & __lhs, unsigned long __offset)
- std::ptrdiff_t **operator-** (const [_Pointer_adapter](#) & __lhs, element_type * __rhs)
- std::ptrdiff_t **operator-** (element_type * __lhs, const [_Pointer_adapter](#) & __rhs)
- template<typename _Up >
 std::ptrdiff_t **operator-** (const [_Pointer_adapter](#) & __lhs, _Up * __rhs)
- template<typename _Up >
 std::ptrdiff_t **operator-** (_Up * __lhs, const [_Pointer_adapter](#) & __rhs)
- [_Pointer_adapter](#) **operator-** (const [_Pointer_adapter](#) & __lhs, short __offset)
- [_Pointer_adapter](#) **operator-** (const [_Pointer_adapter](#) & __lhs, unsigned short __offset)
- [_Pointer_adapter](#) **operator-** (const [_Pointer_adapter](#) & __lhs, int __offset)
- [_Pointer_adapter](#) **operator-** (const [_Pointer_adapter](#) & __lhs, unsigned int __offset)
- [_Pointer_adapter](#) **operator-** (const [_Pointer_adapter](#) & __lhs, long __offset)
- [_Pointer_adapter](#) **operator-** (const [_Pointer_adapter](#) & __lhs, unsigned long __offset)

4.23.1 Detailed Description

```
template<typename _Storage_policy>
class __gnu_cxx::Pointer_adapter<_Storage_policy>
```

The following provides an 'alternative pointer' that works with the containers when specified as the pointer typedef of the allocator.

The pointer type used with the containers doesn't have to be this class, but it must support the implicit conversions, pointer arithmetic, comparison operators, etc. that are supported by this class, and avoid raising compile-time ambiguities. Because creating a working pointer can be challenging, this pointer template was designed to wrapper an easier storage policy type, so that it becomes reusable for creating other pointer types.

A key point of this class is also that it allows container writers to 'assume' `Allocator::pointer` is a typedef for a normal pointer. This class supports most of the conventions of a true pointer, and can, for instance handle implicit conversion to `const` and base class pointer types. The only impositions on container writers to support extended pointers are: 1) use the `Allocator::pointer` typedef appropriately for pointer types. 2) if you need pointer casting, use the `__pointer_cast<>` functions from `ext/cast.h`. This allows pointer cast operations to be overloaded as necessary by custom pointers.

Note: The `const` qualifier works with this pointer adapter as follows:

```
_Tp* == _Pointer_adapter<_Std_pointer_impl<_Tp>>; const _Tp* == _Pointer_adapter<_Std_pointer_impl<const
_Tp>>; _Tp* const == const _Pointer_adapter<_Std_pointer_impl<_Tp>>; const _Tp* const == const _Pointer_
adapter<_Std_pointer_impl<const _Tp>>;
```

Definition at line 281 of file `pointer.h`.

The documentation for this class was generated from the following file:

- [pointer.h](#)

4.24 `__gnu_cxx::Relative_pointer_impl<_Tp>` Class Template Reference

Public Types

- typedef `_Tp` **element_type**

Public Member Functions

- `_Tp *` **get** () const
- bool **operator**< (const [_Relative_pointer_impl](#) &__rarg) const
- bool **operator**== (const [_Relative_pointer_impl](#) &__rarg) const
- void **set** (_Tp *__arg)

4.24.1 Detailed Description

```
template<typename _Tp>
class __gnu_cxx::_Relative_pointer_impl< _Tp >
```

A storage policy for use with `_Pointer_adapter<>` which stores the pointer's address as an offset value which is relative to its own address.

This is intended for pointers within shared memory regions which might be mapped at different addresses by different processes. For null pointers, a value of 1 is used. (0 is legitimate sometimes for nodes in circularly linked lists) This value was chosen as the least likely to generate an incorrect null, As there is no reason why any normal pointer would point 1 byte into its own pointer address.

Definition at line 109 of file `pointer.h`.

The documentation for this class was generated from the following file:

- [pointer.h](#)

4.25 `__gnu_cxx::_Relative_pointer_impl< const _Tp >` Class Template Reference

Public Types

- typedef const `_Tp` **element_type**

Public Member Functions

- const `_Tp` * **get** () const
- bool **operator**< (const `_Relative_pointer_impl` &__rarg) const
- bool **operator**== (const `_Relative_pointer_impl` &__rarg) const
- void **set** (const `_Tp` *__arg)

4.25.1 Detailed Description

```
template<typename _Tp>
class __gnu_cxx::_Relative_pointer_impl< const _Tp >
```

`Relative_pointer_impl` needs a specialization for const T because of the casting done during pointer arithmetic.

Definition at line 161 of file `pointer.h`.

The documentation for this class was generated from the following file:

- [pointer.h](#)

4.26 `__gnu_cxx::Std_pointer_impl<_Tp>` Class Template Reference

Public Types

- typedef `_Tp` **element_type**

Public Member Functions

- `_Tp * get ()` const
- bool **operator**< (const `_Std_pointer_impl` &__rarg) const
- bool **operator**== (const `_Std_pointer_impl` &__rarg) const
- void **set** (element_type *__arg)

4.26.1 Detailed Description

```
template<typename _Tp>
class __gnu_cxx::Std_pointer_impl<_Tp>
```

A storage policy for use with `_Pointer_adapter<>` which yields a standard pointer.

A `_Storage_policy` is required to provide 4 things: 1) A `get()` API for returning the stored pointer value. 2) An `set()` API for storing a pointer value. 3) An `element_type` typedef to define the type this points to. 4) An `operator<()` to support pointer comparison. 5) An `operator==()` to support pointer comparison.

Definition at line 66 of file `pointer.h`.

The documentation for this class was generated from the following file:

- [pointer.h](#)

4.27 `__gnu_cxx::Unqualified_type<_Tp>` Struct Template Reference

Public Types

- typedef `_Tp` **type**

4.27.1 Detailed Description

```
template<typename _Tp>
struct __gnu_cxx::Unqualified_type<_Tp>
```

This structure accommodates the way in which `std::iterator_traits<>` is normally specialized for `const T*`, so that `value_of_type` is still `T`.

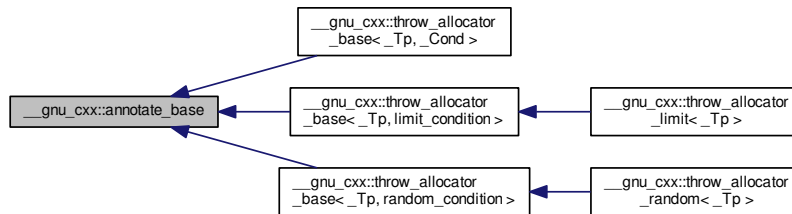
Definition at line 241 of file `pointer.h`.

The documentation for this struct was generated from the following file:

- [pointer.h](#)

4.28 `__gnu_cxx::annotate_base` Struct Reference

Inheritance diagram for `__gnu_cxx::annotate_base`:



Public Member Functions

- void **check** (size_t label)
- void **check_allocated** (void *p, size_t size)
- void **check_constructed** (void *p)
- void **check_constructed** (size_t label)
- void **erase** (void *p, size_t size)
- void **erase_construct** (void *p)
- void **insert** (void *p, size_t size)
- void **insert_construct** (void *p)

Static Public Member Functions

- static void **check** ()
- static size_t **get_label** ()
- static void **set_label** (size_t l)

Friends

- std::ostream & **operator**<< (std::ostream &, const [annotate_base](#) &)

4.28.1 Detailed Description

Base class for checking address and label information about allocations. Create a `std::map` between the allocated address (void*) and a datum for annotations, which are a pair of numbers corresponding to label and allocated size.

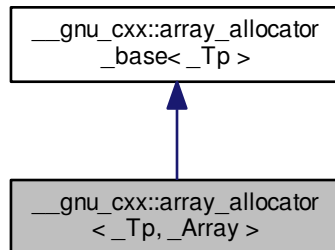
Definition at line 88 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw_allocator.h](#)

4.29 `__gnu_cxx::array_allocator<_Tp, _Array>` Class Template Reference

Inheritance diagram for `__gnu_cxx::array_allocator<_Tp, _Array>`:



Public Types

- typedef `_Array` **array_type**
- typedef const `_Tp` * **const_pointer**
- typedef const `_Tp` & **const_reference**
- typedef ptrdiff_t **difference_type**
- typedef `_Tp` * **pointer**
- typedef std::true_type **propagate_on_container_move_assignment**
- typedef `_Tp` & **reference**
- typedef size_t **size_type**
- typedef `_Tp` **value_type**

Public Member Functions

- **array_allocator** (`array_type` * __array=0) noexcept
- **array_allocator** (const [array_allocator](#) & __o) noexcept
- template<typename `_Tp1` , typename `_Array1` >
array_allocator (const [array_allocator](#)< `_Tp1`, `_Array1` > &) noexcept
- pointer **address** (reference __x) const noexcept
- const_pointer **address** (const_reference __x) const noexcept
- pointer **allocate** (size_type __n, const void *=0)
- template<typename `_Up` , typename... `_Args`>
void **construct** (`_Up` * __p, `_Args` &&... __args)
- void **deallocate** (pointer, size_type)
- template<typename `_Up` >
void **destroy** (`_Up` * __p)
- size_type **max_size** () const noexcept

4.29.1 Detailed Description

```
template<typename _Tp, typename _Array = std::tr1::array<_Tp, 1>>
class __gnu_cxx::array_allocator<_Tp, _Array>
```

An allocator that uses previously allocated memory. This memory can be externally, globally, or otherwise allocated.

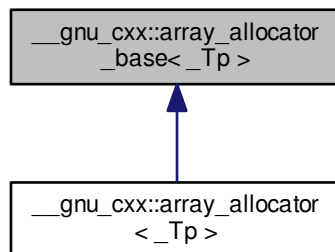
Definition at line 53 of file ext/alloc_traits.h.

The documentation for this class was generated from the following files:

- [ext/alloc_traits.h](#)
- [array_allocator.h](#)

4.30 __gnu_cxx::array_allocator_base<_Tp> Class Template Reference

Inheritance diagram for __gnu_cxx::array_allocator_base<_Tp>:



Public Types

- typedef const _Tp * **const_pointer**
- typedef const _Tp & **const_reference**
- typedef ptrdiff_t **difference_type**
- typedef _Tp * **pointer**
- typedef _Tp & **reference**
- typedef size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- pointer **address** (reference `__x`) const noexcept
- const_pointer **address** (const_reference `__x`) const noexcept
- template<typename `_Up` , typename... `_Args`>
void **construct** (`_Up` *`__p`, `_Args` &&... `__args`)
- void **deallocate** (pointer, size_type)
- template<typename `_Up` >
void **destroy** (`_Up` *`__p`)
- size_type **max_size** () const noexcept

4.30.1 Detailed Description

```
template<typename _Tp>
class __gnu_cxx::array_allocator_base<_Tp>
```

Base class.

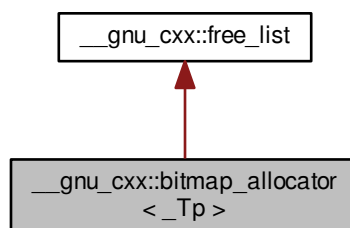
Definition at line 50 of file `array_allocator.h`.

The documentation for this class was generated from the following file:

- [array_allocator.h](#)

4.31 `__gnu_cxx::bitmap_allocator<_Tp>` Class Template Reference

Inheritance diagram for `__gnu_cxx::bitmap_allocator<_Tp>`:



Public Types

- typedef free_list::__mutex_type **__mutex_type**
- typedef const _Tp * **const_pointer**
- typedef const _Tp & **const_reference**
- typedef ptrdiff_t **difference_type**
- typedef _Tp * **pointer**
- typedef std::true_type **propagate_on_container_move_assignment**
- typedef _Tp & **reference**
- typedef size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- **bitmap_allocator** (const [bitmap_allocator](#) &) noexcept
- template<typename _Tp1 >
 bitmap_allocator (const [bitmap_allocator](#)< _Tp1 > &) noexcept
- pointer [_M_allocate_single_object](#) () throw (std::bad_alloc)
- void [_M_deallocate_single_object](#) (pointer __p) throw ()
- pointer **address** (reference __r) const noexcept
- const_pointer **address** (const_reference __r) const noexcept
- pointer **allocate** (size_type __n)
- pointer **allocate** (size_type __n, typename [bitmap_allocator](#)< void >::const_pointer)
- template<typename _Up, typename... _Args>
 void **construct** (_Up * __p, _Args &&... __args)
- void **deallocate** (pointer __p, size_type __n) throw ()
- template<typename _Up >
 void **destroy** (_Up * __p)
- size_type **max_size** () const noexcept

Private Types

- typedef vector_type::iterator **iterator**
- typedef [__detail::__mini_vector](#)< value_type > **vector_type**

Private Member Functions

- void [_M_clear](#) ()
- size_t * [_M_get](#) (size_t __sz) throw (std::bad_alloc)
- void [_M_insert](#) (size_t * __addr) throw ()

4.31.1 Detailed Description

```
template<typename _Tp>
class __gnu_cxx::bitmap_allocator< _Tp >
```

Bitmap Allocator, primary template.

Definition at line 59 of file ext/alloc_traits.h.

4.31.2 Member Function Documentation

4.31.2.1 `template<typename _Tp> pointer __gnu_cxx::bitmap_allocator<_Tp>::_M_allocate_single_object () throw std::bad_alloc) [inline]`

Allocates memory for a single object of size `sizeof(_Tp)`.

Exceptions

<code>std::bad_alloc.</code>	If memory can not be allocated.
------------------------------	---------------------------------

Complexity: Worst case complexity is $O(N)$, but that is hardly ever hit. If and when this particular case is encountered, the next few cases are guaranteed to have a worst case complexity of $O(1)$! That's why this function performs very well on average. You can consider this function to have a complexity referred to commonly as: Amortized Constant time.

Definition at line 827 of file `bitmap_allocator.h`.

References `__gnu_cxx::__detail::__bit_allocate()`, `__gnu_cxx::__detail::__num_bitmaps()`, and `__gnu_cxx::_Bit_scan←_forward()`.

4.31.2.2 `template<typename _Tp> void __gnu_cxx::bitmap_allocator<_Tp>::_M_deallocate_single_object (pointer __p) throw) [inline]`

Deallocates memory that belongs to a single object of size `sizeof(_Tp)`.

Complexity: $O(\lg(N))$, but the worst case is not hit often! This is because containers usually deallocate memory close to each other and this case is handled in $O(1)$ time by the `deallocate` function.

Definition at line 917 of file `bitmap_allocator.h`.

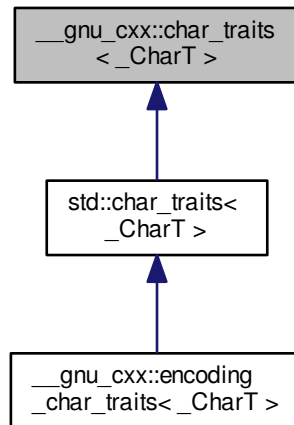
References `std::__addressof()`, `__gnu_cxx::__detail::__bit_free()`, `__gnu_cxx::__detail::__num_bitmaps()`, and `std::__←_rotate()`.

The documentation for this class was generated from the following files:

- [ext/alloc_traits.h](#)
- [bitmap_allocator.h](#)

4.32 `__gnu_cxx::char_traits<_CharT>` Struct Template Reference

Inheritance diagram for `__gnu_cxx::char_traits<_CharT>`:



Public Types

- typedef `_CharT char_type`
- typedef `_Char_types<_CharT>::int_type int_type`
- typedef `_Char_types<_CharT>::off_type off_type`
- typedef `_Char_types<_CharT>::pos_type pos_type`
- typedef `_Char_types<_CharT>::state_type state_type`

Static Public Member Functions

- static void **assign** (`char_type &__c1`, `const char_type &__c2`)
- static `char_type *` **assign** (`char_type *__s`, `std::size_t __n`, `char_type __a`)
- static int **compare** (`const char_type *__s1`, `const char_type *__s2`, `std::size_t __n`)
- static `char_type *` **copy** (`char_type *__s1`, `const char_type *__s2`, `std::size_t __n`)
- static constexpr int_type **eof** ()
- static constexpr bool **eq** (`const char_type &__c1`, `const char_type &__c2`)
- static constexpr bool **eq_int_type** (`const int_type &__c1`, `const int_type &__c2`)
- static const `char_type *` **find** (`const char_type *__s`, `std::size_t __n`, `const char_type &__a`)
- static `std::size_t` **length** (`const char_type *__s`)
- static constexpr bool **lt** (`const char_type &__c1`, `const char_type &__c2`)
- static `char_type *` **move** (`char_type *__s1`, `const char_type *__s2`, `std::size_t __n`)
- static constexpr int_type **not_eof** (`const int_type &__c`)
- static constexpr `char_type` **to_char_type** (`const int_type &__c`)
- static constexpr int_type **to_int_type** (`const char_type &__c`)

4.32.1 Detailed Description

```
template<typename _CharT>
struct __gnu_cxx::char_traits<_CharT>
```

Base class used to implement `std::char_traits`.

Note

For any given actual character type, this definition is probably wrong. (Most of the member functions are likely to be right, but the `int_type` and `state_type` typedefs, and the `eof()` member function, are likely to be wrong.) The reason this class exists is so users can specialize it. Classes in namespace `std` may not be specialized for fundamental types, but classes in namespace `__gnu_cxx` may be.

See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt05ch13s03.html> for advice on how to make use of this class for *unusual* character types. Also, check out `include/ext/pod_char_traits.h`.

Definition at line 83 of file `char_traits.h`.

The documentation for this struct was generated from the following file:

- [char_traits.h](#)

4.33 `__gnu_cxx::character<_Value, _Int, _St>` Struct Template Reference

Public Types

- typedef `character<_Value, _Int, _St>` **char_type**
- typedef `_Int` **int_type**
- typedef `_St` **state_type**
- typedef `_Value` **value_type**

Static Public Member Functions

- template<typename V2 >
static `char_type` **from** (const V2 &v)
- template<typename V2 >
static V2 **to** (const `char_type` &c)

Public Attributes

- `value_type` **value**

4.33.1 Detailed Description

```
template<typename _Value, typename _Int, typename _St = std::mbstate_t>
struct __gnu_cxx::character< _Value, _Int, _St >
```

A POD class that serves as a character abstraction class.

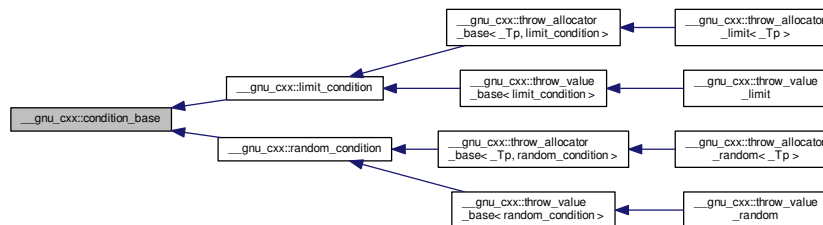
Definition at line 49 of file pod_char_traits.h.

The documentation for this struct was generated from the following file:

- [pod_char_traits.h](#)

4.34 __gnu_cxx::condition_base Struct Reference

Inheritance diagram for __gnu_cxx::condition_base:



4.34.1 Detailed Description

Base struct for condition policy.

Requires a public member function with the signature void throw_conditionally()

Definition at line 403 of file throw_allocator.h.

The documentation for this struct was generated from the following file:

- [throw_allocator.h](#)

4.35 __gnu_cxx::debug_allocator< _Alloc > Class Template Reference

Public Types

- typedef _Traits::const_pointer **const_pointer**
- typedef _Traits::const_reference **const_reference**
- typedef _Traits::difference_type **difference_type**
- typedef _Traits::pointer **pointer**
- typedef _Traits::reference **reference**
- typedef _Traits::size_type **size_type**
- typedef _Traits::value_type **value_type**

Public Member Functions

- `template<typename _Alloc2 >`
`debug_allocator` (const `debug_allocator<_Alloc2>` &__a2, typename `__convertible<_Alloc2>::__type=0`)
- `debug_allocator` (const `_Alloc` &__a)
- pointer `allocate` (size_type __n)
- pointer `allocate` (size_type __n, const void * __hint)
- void `construct` (pointer __p, const value_type &__val)
- `template<typename _Tp, typename... _Args>`
void `construct` (_Tp * __p, _Args &&... __args)
- void `deallocate` (pointer __p, size_type __n)
- `template<typename _Tp >`
void `destroy` (_Tp * __p)
- size_type `max_size` () const throw ()

Friends

- `template<typename >`
class `debug_allocator`
- bool `operator==` (const `debug_allocator` &__lhs, const `debug_allocator` &__rhs)

4.35.1 Detailed Description

```
template<typename _Alloc>
class __gnu_cxx::debug_allocator<_Alloc>
```

A meta-allocator with debugging bits.

This is precisely the allocator defined in the C++03 Standard.

Definition at line 62 of file `debug_allocator.h`.

The documentation for this class was generated from the following file:

- [debug_allocator.h](#)

4.36 `__gnu_cxx::enc_filebuf<_CharT>` Class Template Reference

Inherits `basic_filebuf<_CharT, encoding_char_traits<_CharT>>`.

Public Types

- typedef `traits_type::pos_type` `pos_type`
- typedef `traits_type::state_type` `state_type`
- typedef `encoding_char_traits<_CharT>` `traits_type`

Public Member Functions

- **enc_filebuf** ([state_type](#) &__state)

4.36.1 Detailed Description

```
template<typename _CharT>
class __gnu_cxx::enc_filebuf< _CharT >
```

class enc_filebuf.

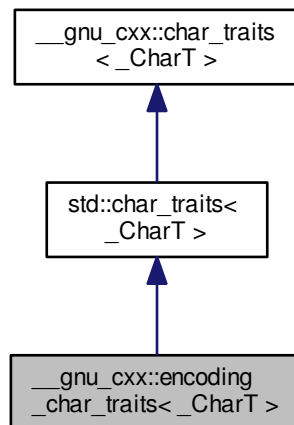
Definition at line 42 of file enc_filebuf.h.

The documentation for this class was generated from the following file:

- [enc_filebuf.h](#)

4.37 __gnu_cxx::encoding_char_traits< _CharT > Struct Template Reference

Inheritance diagram for __gnu_cxx::encoding_char_traits< _CharT >:



Public Types

- typedef `_CharT` **char_type**
- typedef `_Char_types< _CharT >::int_type` **int_type**
- typedef `_Char_types< _CharT >::off_type` **off_type**
- typedef `std::fpos< state_type >` **pos_type**
- typedef `encoding_state` **state_type**

Static Public Member Functions

- static void **assign** (char_type &__c1, const char_type &__c2)
- static char_type * **assign** (char_type *__s, std::size_t __n, char_type __a)
- static int **compare** (const char_type *__s1, const char_type *__s2, std::size_t __n)
- static char_type * **copy** (char_type *__s1, const char_type *__s2, std::size_t __n)
- static constexpr int_type **eof** ()
- static constexpr bool **eq** (const char_type &__c1, const char_type &__c2)
- static constexpr bool **eq_int_type** (const int_type &__c1, const int_type &__c2)
- static const char_type * **find** (const char_type *__s, std::size_t __n, const char_type &__a)
- static std::size_t **length** (const char_type *__s)
- static constexpr bool **lt** (const char_type &__c1, const char_type &__c2)
- static char_type * **move** (char_type *__s1, const char_type *__s2, std::size_t __n)
- static constexpr int_type **not_eof** (const int_type &__c)
- static constexpr char_type **to_char_type** (const int_type &__c)
- static constexpr int_type **to_int_type** (const char_type &__c)

4.37.1 Detailed Description

```
template<typename _CharT>
struct __gnu_cxx::encoding_char_traits< _CharT >
```

encoding_char_traits

Definition at line 210 of file codecvt_specializations.h.

The documentation for this struct was generated from the following file:

- [codecvt_specializations.h](#)

4.38 `__gnu_cxx::encoding_state` Class Reference

Public Types

- typedef iconv_t **descriptor_type**

Public Member Functions

- **encoding_state** (const char *__int, const char *__ext, int __ibom=0, int __ebom=0, int __bytes=1)
- **encoding_state** (const [encoding_state](#) &__obj)
- int **character_ratio** () const
- int **external_bom** () const
- const [std::string](#) **external_encoding** () const
- bool **good** () const throw ()
- const descriptor_type & **in_descriptor** () const
- int **internal_bom** () const
- const [std::string](#) **internal_encoding** () const
- [encoding_state](#) & **operator=** (const [encoding_state](#) &__obj)
- const descriptor_type & **out_descriptor** () const

Protected Member Functions

- void **construct** (const [encoding_state](#) &__obj)
- void **destroy** () throw ()
- void **init** ()

Protected Attributes

- int **_M_bytes**
- int **_M_ext_bom**
- [std::string](#) **_M_ext_enc**
- descriptor_type **_M_in_desc**
- int **_M_int_bom**
- [std::string](#) **_M_int_enc**
- descriptor_type **_M_out_desc**

4.38.1 Detailed Description

Extension to use iconv for dealing with character encodings.

Definition at line 50 of file `codecvt_specializations.h`.

The documentation for this class was generated from the following file:

- [codecvt_specializations.h](#)

4.39 `__gnu_cxx::forced_error` Struct Reference

Inherits exception.

4.39.1 Detailed Description

Thrown by exception safety machinery.

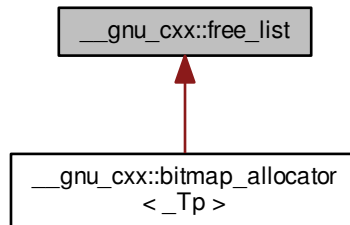
Definition at line 74 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw_allocator.h](#)

4.40 `__gnu_cxx::free_list` Class Reference

Inheritance diagram for `__gnu_cxx::free_list`:



Public Types

- typedef `__mutex` **`__mutex_type`**
- typedef `vector_type::iterator` **`iterator`**
- typedef `size_t * value_type`
- typedef `__detail::__mini_vector< value_type >` **`vector_type`**

Public Member Functions

- void `_M_clear` ()
- `size_t * _M_get` (size_t `__sz`) throw (std::bad_alloc)
- void `_M_insert` (size_t * `__addr`) throw ()

4.40.1 Detailed Description

The free list class for managing chunks of memory to be given to and returned by the `bitmap_allocator`.

Definition at line 521 of file `bitmap_allocator.h`.

4.40.2 Member Function Documentation

4.40.2.1 void `__gnu_cxx::free_list::_M_clear` ()

This function just clears the internal Free List, and gives back all the memory to the OS.

4.40.2.2 `size_t * __gnu_cxx::free_list::_M_get` (size_t `__sz`) throw std::bad_alloc

This function gets a block of memory of the specified size from the free list.

Parameters

<code>__sz</code>	The size in bytes of the memory required.
-------------------	---

Returns

A pointer to the new memory block of size at least equal to that requested.

4.40.2.3 `void __gnu_cxx::free_list::M_insert (size_t * __addr) throw () [inline]`

This function returns the block of memory to the internal free list.

Parameters

<code>__addr</code>	The pointer to the memory block that was given by a call to the <code>_M_get</code> function.
---------------------	---

Definition at line 631 of file `bitmap_allocator.h`.

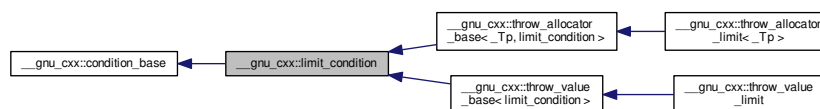
References `__gnu_cxx::__detail::__num_bitmaps()`, `_BALLOC_ALIGN_BYTES`, and `std::make_pair()`.

The documentation for this class was generated from the following file:

- [bitmap_allocator.h](#)

4.41 __gnu_cxx::limit_condition Struct Reference

Inheritance diagram for `__gnu_cxx::limit_condition`:



Classes

- struct [always_adjustor](#)
- struct [limit_adjustor](#)
- struct [never_adjustor](#)

Static Public Member Functions

- static `size_t & count ()`
- static `size_t & limit ()`
- static void `set_limit (const size_t __l)`
- static void `throw_conditionally ()`

4.41.1 Detailed Description

Base class for incremental control and throw.

Definition at line 412 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw_allocator.h](#)

4.42 `__gnu_cxx::limit_condition::always_adjustor` Struct Reference

Inherits `__gnu_cxx::limit_condition::adjustor_base`.

4.42.1 Detailed Description

Always enter the condition.

Definition at line 436 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw_allocator.h](#)

4.43 `__gnu_cxx::limit_condition::limit_adjustor` Struct Reference

Inherits `__gnu_cxx::limit_condition::adjustor_base`.

Public Member Functions

- `limit_adjustor (const size_t __l)`

4.43.1 Detailed Description

Enter the nth condition.

Definition at line 442 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw_allocator.h](#)

4.44 `__gnu_cxx::limit_condition::never_adjustor` Struct Reference

Inherits `__gnu_cxx::limit_condition::adjustor_base`.

4.44.1 Detailed Description

Never enter the condition.

Definition at line 430 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw_allocator.h](#)

4.45 `__gnu_cxx::malloc_allocator< _Tp >` Class Template Reference

Public Types

- typedef const `_Tp` * **const_pointer**
- typedef const `_Tp` & **const_reference**
- typedef ptrdiff_t **difference_type**
- typedef `_Tp` * **pointer**
- typedef std::true_type **propagate_on_container_move_assignment**
- typedef `_Tp` & **reference**
- typedef size_t **size_type**
- typedef `_Tp` **value_type**

Public Member Functions

- **malloc_allocator** (const [malloc_allocator](#) &) noexcept
- template<typename `_Tp1` >
 malloc_allocator (const [malloc_allocator](#)< `_Tp1` > &) noexcept
- pointer **address** (reference __x) const noexcept
- const_pointer **address** (const_reference __x) const noexcept
- pointer **allocate** (size_type __n, const void *==0)
- template<typename `_Up` , typename... `_Args`>
 void **construct** (`_Up` *__p, `_Args` &&... __args)
- void **deallocate** (pointer __p, size_type)
- template<typename `_Up` >
 void **destroy** (`_Up` *__p)
- size_type **max_size** () const noexcept

4.45.1 Detailed Description

```
template<typename _Tp>
class __gnu_cxx::malloc_allocator<_Tp>
```

An allocator that uses malloc.

This is precisely the allocator defined in the C++ Standard.

- all allocation calls malloc
- all deallocation calls free

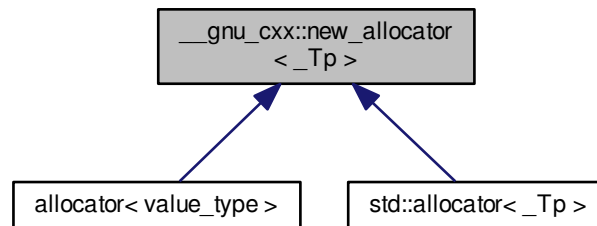
Definition at line 65 of file ext/alloc_traits.h.

The documentation for this class was generated from the following files:

- [ext/alloc_traits.h](#)
- [malloc_allocator.h](#)

4.46 __gnu_cxx::new_allocator<_Tp> Class Template Reference

Inheritance diagram for __gnu_cxx::new_allocator<_Tp>:



Public Types

- typedef const _Tp * **const_pointer**
- typedef const _Tp & **const_reference**
- typedef ptrdiff_t **difference_type**
- typedef _Tp * **pointer**
- typedef std::true_type **propagate_on_container_move_assignment**
- typedef _Tp & **reference**
- typedef size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- **new_allocator** (const [new_allocator](#) &) noexcept
- template<typename _Tp1 >
 new_allocator (const [new_allocator](#)< _Tp1 > &) noexcept
- pointer **address** (reference __x) const noexcept
- const_pointer **address** (const_reference __x) const noexcept
- pointer **allocate** (size_type __n, const void *=0)
- template<typename _Up , typename... _Args>
 void **construct** (_Up *__p, _Args &&... __args)
- void **deallocate** (pointer __p, size_type)
- template<typename _Up >
 void **destroy** (_Up *__p)
- size_type **max_size** () const noexcept

4.46.1 Detailed Description

```
template<typename _Tp>
class __gnu_cxx::new_allocator< _Tp >
```

An allocator that uses global new, as per [20.4].

This is precisely the allocator defined in the C++ Standard.

- all allocation calls operator new
- all deallocation calls operator delete

Template Parameters

<code>_Tp</code>	Type of allocated object.
------------------	---------------------------

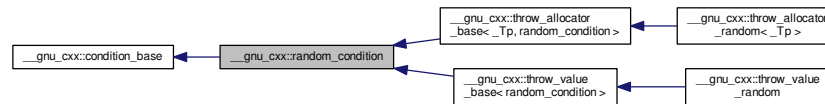
Definition at line 77 of file ext/alloc_traits.h.

The documentation for this class was generated from the following files:

- [ext/alloc_traits.h](#)
- [new_allocator.h](#)

4.47 `__gnu_cxx::random_condition` Struct Reference

Inheritance diagram for `__gnu_cxx::random_condition`:



Classes

- struct [always_adjustor](#)
- struct [group_adjustor](#)
- struct [never_adjustor](#)

Public Member Functions

- void **seed** (unsigned long __s)

Static Public Member Functions

- static void **set_probability** (double __p)
- static void **throw_conditionally** ()

4.47.1 Detailed Description

Base class for random probability control and throw.

Definition at line 484 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw_allocator.h](#)

4.48 `__gnu_cxx::random_condition::always_adjustor` Struct Reference

Inherits `__gnu_cxx::random_condition::adjustor_base`.

4.48.1 Detailed Description

Always enter the condition.

Definition at line 517 of file throw_allocator.h.

The documentation for this struct was generated from the following file:

- [throw_allocator.h](#)

4.49 __gnu_cxx::random_condition::group_adjustor Struct Reference

Inherits __gnu_cxx::random_condition::adjustor_base.

Public Member Functions

- **group_adjustor** (size_t size)

4.49.1 Detailed Description

Group condition.

Definition at line 502 of file throw_allocator.h.

The documentation for this struct was generated from the following file:

- [throw_allocator.h](#)

4.50 __gnu_cxx::random_condition::never_adjustor Struct Reference

Inherits __gnu_cxx::random_condition::adjustor_base.

4.50.1 Detailed Description

Never enter the condition.

Definition at line 511 of file throw_allocator.h.

The documentation for this struct was generated from the following file:

- [throw_allocator.h](#)

4.51 `__gnu_cxx::recursive_init_error` Class Reference

Inherits exception.

4.51.1 Detailed Description

Exception thrown by `__cxa_guard_acquire`.

6.7[stmt.dcl]/4: If control re-enters the declaration (recursively) while the object is being initialized, the behavior is undefined.

Since we already have a library function to handle locking, we might as well check for this situation and throw an exception. We use the second byte of the guard variable to remember that we're in the middle of an initialization.

Definition at line 694 of file `cxxabi.h`.

The documentation for this class was generated from the following file:

- [cxxabi.h](#)

4.52 `__gnu_cxx::stdio_filebuf<_CharT, _Traits>` Class Template Reference

Inherits `basic_filebuf<_CharT, _Traits>`.

Public Types

- typedef `_CharT` **char_type**
- typedef `traits_type::int_type` **int_type**
- typedef `traits_type::off_type` **off_type**
- typedef `traits_type::pos_type` **pos_type**
- typedef `std::size_t` **size_t**
- typedef `_Traits` **traits_type**

Public Member Functions

- [stdio_filebuf](#) ()
- [stdio_filebuf](#) (int `__fd`, [std::ios_base::openmode](#) `__mode`, `size_t` `__size`=`static_cast<size_t>(BUFSIZ)`)
- [stdio_filebuf](#) (`std::__c_file *` `__f`, [std::ios_base::openmode](#) `__mode`, `size_t` `__size`=`static_cast<size_t>(BUFSIZ)`)
- virtual [~stdio_filebuf](#) ()
- int [fd](#) ()
- `std::__c_file *` [file](#) ()

4.52.1 Detailed Description

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>>
class __gnu_cxx::stdio_filebuf< _CharT, _Traits >
```

Provides a layer of compatibility for C/POSIX.

This GNU extension provides extensions for working with standard C FILE*'s and POSIX file descriptors. It must be instantiated by the user with the type of character used in the file stream, e.g., `stdio_filebuf<char>`.

Definition at line 50 of file `stdio_filebuf.h`.

4.52.2 Constructor & Destructor Documentation

4.52.2.1 `template<typename _CharT, typename _Traits = std::char_traits<_CharT>> __gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf ()` `[inline]`

deferred initialization

Definition at line 65 of file `stdio_filebuf.h`.

References `__gnu_cxx::stdio_filebuf< _CharT, _Traits >::~~stdio_filebuf()`.

Referenced by `__gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf()`, and `__gnu_cxx::stdio_filebuf< _CharT, _↵ Traits >::~~stdio_filebuf()`.

4.52.2.2 `template<typename _CharT, typename _Traits > __gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf (int __fd, std::ios_base::openmode __mode, size_t __size = static_cast<size_t>(BUFSIZ))`

Parameters

<code>__fd</code>	An open file descriptor.
<code>__mode</code>	Same meaning as in a standard filebuf.
<code>__size</code>	Optimal or preferred size of internal buffer, in chars.

This constructor associates a file stream buffer with an open POSIX file descriptor. The file descriptor will be automatically closed when the `stdio_filebuf` is closed/destroyed.

Definition at line 128 of file `stdio_filebuf.h`.

References `__gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf()`.

4.52.2.3 `template<typename _CharT, typename _Traits > __gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf (std::_c_file * __f, std::ios_base::openmode __mode, size_t __size = static_cast<size_t>(BUFSIZ))`

Parameters

<code>__f</code>	An open <code>FILE*</code> .
<code>__mode</code>	Same meaning as in a standard <code>filebuf</code> .
<code>__size</code>	Optimal or preferred size of internal buffer, in chars. Defaults to system's <code>BUFSIZ</code> .

This constructor associates a file stream buffer with an open C `FILE*`. The `FILE*` will not be automatically closed when the `stdio_filebuf` is closed/destroyed.

Definition at line 144 of file `stdio_filebuf.h`.

4.52.2.4 `template<typename _CharT, typename _Traits> __gnu_cxx::stdio_filebuf<_CharT, _Traits>::~~stdio_filebuf ()`
`[virtual]`

Closes the external data stream if the file descriptor constructor was used.

Definition at line 123 of file `stdio_filebuf.h`.

References `__gnu_cxx::stdio_filebuf<_CharT, _Traits>::stdio_filebuf()`.

Referenced by `__gnu_cxx::stdio_filebuf<_CharT, _Traits>::stdio_filebuf()`.

4.52.3 Member Function Documentation

4.52.3.1 `template<typename _CharT, typename _Traits = std::char_traits<_CharT>> int __gnu_cxx::stdio_filebuf<_CharT, _Traits>::fd ()` `[inline]`

Returns

The underlying file descriptor.

Once associated with an external data stream, this function can be used to access the underlying POSIX file descriptor. Note that there is no way for the library to track what you do with the descriptor, so be careful.

Definition at line 109 of file `stdio_filebuf.h`.

4.52.3.2 `template<typename _CharT, typename _Traits = std::char_traits<_CharT>> std::_c_file* __gnu_cxx::stdio_filebuf<_CharT, _Traits>::file ()` `[inline]`

Returns

The underlying `FILE*`.

This function can be used to access the underlying "C" file pointer. Note that there is no way for the library to track what you do with the file, so be careful.

Definition at line 119 of file `stdio_filebuf.h`.

The documentation for this class was generated from the following file:

- [stdio_filebuf.h](#)

4.53 `__gnu_cxx::stdio_sync_filebuf<_CharT,_Traits>` Class Template Reference

Inherits `basic_streambuf<_CharT,_Traits>`.

Public Types

- typedef `_CharT` **char_type**
- typedef `traits_type::int_type` **int_type**
- typedef `traits_type::off_type` **off_type**
- typedef `traits_type::pos_type` **pos_type**
- typedef `_Traits` **traits_type**

Public Member Functions

- **stdio_sync_filebuf** (`std::__c_file * __f`)
- `std::__c_file * const` **file** ()

Protected Member Functions

- virtual `int_type` **overflow** (`int_type __c=traits_type::eof()`)
- virtual `int_type` **pbackfail** (`int_type __c=traits_type::eof()`)
- virtual `std::streampos` **seekoff** (`std::streamoff __off, std::ios_base::seekdir __dir, std::ios_base::openmode=std::ios_base::in|std::ios_base::out`)
- virtual `std::streampos` **seekpos** (`std::streampos __pos, std::ios_base::openmode __mode=std::ios_base::in|std::ios_base::out`)
- virtual `int` **sync** ()
- `int_type` **syncgetc** ()
- template<>
`stdio_sync_filebuf< char >::int_type` **syncgetc** ()
- template<>
`stdio_sync_filebuf< wchar_t >::int_type` **syncgetc** ()
- `int_type` **syncputc** (`int_type __c`)
- template<>
`stdio_sync_filebuf< char >::int_type` **syncputc** (`int_type __c`)
- template<>
`stdio_sync_filebuf< wchar_t >::int_type` **syncputc** (`int_type __c`)
- `int_type` **syncungetc** (`int_type __c`)
- template<>
`stdio_sync_filebuf< char >::int_type` **syncungetc** (`int_type __c`)
- template<>
`stdio_sync_filebuf< wchar_t >::int_type` **syncungetc** (`int_type __c`)
- virtual `int_type` **uflow** ()
- virtual `int_type` **underflow** ()
- virtual `std::streamsize` **xsgetn** (`char_type * __s, std::streamsize __n`)
- template<>
`std::streamsize` **xsgetn** (`char * __s, std::streamsize __n`)
- template<>
`std::streamsize` **xsgetn** (`wchar_t * __s, std::streamsize __n`)
- virtual `std::streamsize` **xspu**tn (`const char_type * __s, std::streamsize __n`)
- template<>
`std::streamsize` **xspu**tn (`const char * __s, std::streamsize __n`)
- template<>
`std::streamsize` **xspu**tn (`const wchar_t * __s, std::streamsize __n`)

4.53.1 Detailed Description

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>>
class __gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>
```

Provides a layer of compatibility for C.

This GNU extension provides extensions for working with standard C FILE*'s. It must be instantiated by the user with the type of character used in the file stream, e.g., `stdio_filebuf<char>`.

Definition at line 56 of file `stdio_sync_filebuf.h`.

4.53.2 Member Function Documentation

```
4.53.2.1 template<typename _CharT, typename _Traits = std::char_traits<_CharT>> std::__c_file* const
        __gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>::file ( ) [inline]
```

Returns

The underlying FILE*.

This function can be used to access the underlying C file pointer. Note that there is no way for the library to track what you do with the file, so be careful.

Definition at line 88 of file `stdio_sync_filebuf.h`.

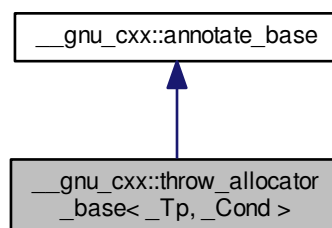
References `std::ios_base::beg`, `std::ios_base::cur`, `std::ios_base::in`, and `std::ios_base::out`.

The documentation for this class was generated from the following file:

- [stdio_sync_filebuf.h](#)

4.54 `__gnu_cxx::throw_allocator_base<_Tp, _Cond>` Class Template Reference

Inheritance diagram for `__gnu_cxx::throw_allocator_base<_Tp, _Cond>`:



Public Types

- typedef const value_type * **const_pointer**
- typedef const value_type & **const_reference**
- typedef ptrdiff_t **difference_type**
- typedef value_type * **pointer**
- typedef std::true_type **propagate_on_container_move_assignment**
- typedef value_type & **reference**
- typedef size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- pointer **address** (reference __x) const noexcept
- const_pointer **address** (const_reference __x) const noexcept
- pointer **allocate** (size_type __n, [std::allocator](#)< void >::const_pointer hint=0)
- void **check** (size_type __n)
- void **check_allocated** (void *p, size_t size)
- void **check_allocated** (pointer __p, size_type __n)
- void **check_constructed** (void *p)
- void **check_constructed** (size_t label)
- template<typename _Up, typename... _Args>
void **construct** (_Up *__p, _Args &&... __args)
- void **deallocate** (pointer __p, size_type __n)
- template<typename _Up >
void **destroy** (_Up *__p)
- void **erase** (void *p, size_t size)
- void **erase_construct** (void *p)
- void **insert** (void *p, size_t size)
- void **insert_construct** (void *p)
- size_type **max_size** () const noexcept

Static Public Member Functions

- static void **check** ()
- static size_t **get_label** ()
- static void **set_label** (size_t l)

4.54.1 Detailed Description

```
template<typename _Tp, typename _Cond>
class __gnu_cxx::throw_allocator_base< _Tp, _Cond >
```

Allocator class with logging and exception generation control. Intended to be used as an allocator_type in templated code.

Note: Deallocate not allowed to throw.

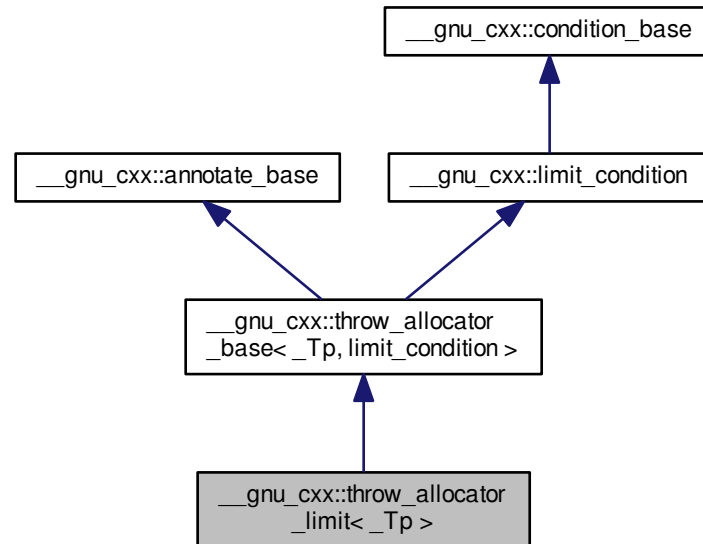
Definition at line 790 of file throw_allocator.h.

The documentation for this class was generated from the following file:

- [throw_allocator.h](#)

4.55 `__gnu_cxx::throw_allocator_limit<_Tp>` Struct Template Reference

Inheritance diagram for `__gnu_cxx::throw_allocator_limit<_Tp>`:



Public Types

- typedef const value_type * **const_pointer**
- typedef const value_type & **const_reference**
- typedef ptrdiff_t **difference_type**
- typedef value_type * **pointer**
- typedef std::true_type **propagate_on_container_move_assignment**
- typedef value_type & **reference**
- typedef size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- **throw_allocator_limit** (const [throw_allocator_limit](#) &) noexcept
- template<typename _Tp1 >
 throw_allocator_limit (const [throw_allocator_limit](#)<_Tp1> &) noexcept
- pointer **address** (reference __x) const noexcept
- const_pointer **address** (const_reference __x) const noexcept
- pointer **allocate** (size_type __n, [std::allocator](#)< void >::const_pointer hint=0)
- void **check** (size_type __n)
- void **check_allocated** (void *p, size_t size)

- void **check_allocated** (pointer __p, size_type __n)
- void **check_constructed** (void *p)
- void **check_constructed** (size_t label)
- void **construct** (_Up *__p, _Args &&... __args)
- void **deallocate** (pointer __p, size_type __n)
- void **destroy** (_Up *__p)
- void **erase** (void *p, size_t size)
- void **erase_construct** (void *p)
- void **insert** (void *p, size_t size)
- void **insert_construct** (void *p)
- size_type **max_size** () const noexcept

Static Public Member Functions

- static void **check** ()
- static size_t & **count** ()
- static size_t **get_label** ()
- static size_t & **limit** ()
- static void **set_label** (size_t l)
- static void **set_limit** (const size_t __l)
- static void **throw_conditionally** ()

4.55.1 Detailed Description

```
template<typename _Tp>
struct __gnu_cxx::throw_allocator_limit< _Tp >
```

Allocator throwing via limit condition.

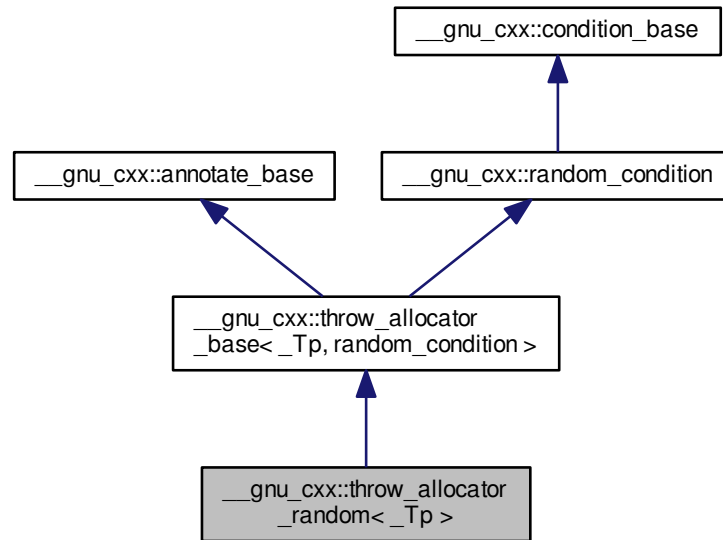
Definition at line 899 of file throw_allocator.h.

The documentation for this struct was generated from the following file:

- [throw_allocator.h](#)

4.56 `__gnu_cxx::throw_allocator_random<_Tp>` Struct Template Reference

Inheritance diagram for `__gnu_cxx::throw_allocator_random<_Tp>`:



Public Types

- typedef const value_type * **const_pointer**
- typedef const value_type & **const_reference**
- typedef ptrdiff_t **difference_type**
- typedef value_type * **pointer**
- typedef std::true_type **propagate_on_container_move_assignment**
- typedef value_type & **reference**
- typedef size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- **throw_allocator_random** (const [throw_allocator_random](#) &) noexcept
- template<typename _Tp1 >
 throw_allocator_random (const [throw_allocator_random](#)<_Tp1 > &) noexcept
- pointer **address** (reference __x) const noexcept
- const_pointer **address** (const_reference __x) const noexcept
- pointer **allocate** (size_type __n, [std::allocator](#)< void >::const_pointer hint=0)
- void **check** (size_type __n)
- void **check_allocated** (void *p, size_t size)

- void **check_allocated** (pointer __p, size_type __n)
- void **check_constructed** (void *p)
- void **check_constructed** (size_t label)
- void **construct** (_Up *__p, _Args &&... __args)
- void **deallocate** (pointer __p, size_type __n)
- void **destroy** (_Up *__p)
- void **erase** (void *p, size_t size)
- void **erase_construct** (void *p)
- void **insert** (void *p, size_t size)
- void **insert_construct** (void *p)
- size_type **max_size** () const noexcept
- void **seed** (unsigned long __s)

Static Public Member Functions

- static void **check** ()
- static size_t **get_label** ()
- static void **set_label** (size_t l)
- static void **set_probability** (double __p)
- static void **throw_conditionally** ()

4.56.1 Detailed Description

```
template<typename _Tp>
struct __gnu_cxx::throw_allocator_random< _Tp >
```

Allocator throwing via random condition.

Definition at line 920 of file throw_allocator.h.

The documentation for this struct was generated from the following file:

- [throw_allocator.h](#)

4.57 __gnu_cxx::throw_value_base< _Cond > Struct Template Reference

Inherits `_Cond`.

Public Types

- typedef `_Cond` **condition_type**

Public Member Functions

- `throw_value_base` (const [throw_value_base](#) &__v)
- `throw_value_base` ([throw_value_base](#) &&)=default
- `throw_value_base` (const std::size_t __i)
- [throw_value_base](#) & `operator++` ()
- `throw_value_base` & `operator=` (const [throw_value_base](#) &__v)
- `throw_value_base` & `operator=` ([throw_value_base](#) &&)=default

Public Attributes

- std::size_t `_M_i`

4.57.1 Detailed Description

```
template<typename _Cond>
struct __gnu_cxx::throw_value_base<_Cond>
```

Class with exception generation control. Intended to be used as a `value_type` in templated code.

Note: Destructor not allowed to throw.

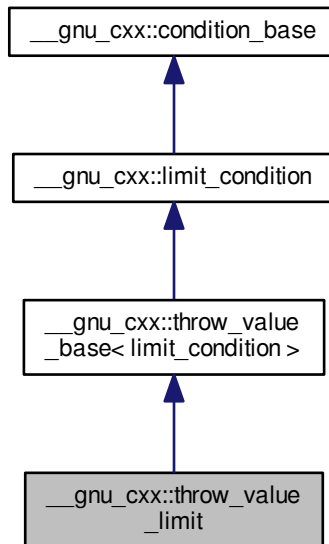
Definition at line 603 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw_allocator.h](#)

4.58 `__gnu_cxx::throw_value_limit` Struct Reference

Inheritance diagram for `__gnu_cxx::throw_value_limit`:



Public Types

- typedef `throw_value_base< limit_condition >` **base_type**
- typedef `limit_condition` **condition_type**

Public Member Functions

- **throw_value_limit** (const `throw_value_limit` &__other)
- **throw_value_limit** (`throw_value_limit` &&)=default
- **throw_value_limit** (const std::size_t __i)
- `throw_value_base` & **operator++** ()
- `throw_value_limit` & **operator=** (const `throw_value_limit` &__other)
- `throw_value_limit` & **operator=** (`throw_value_limit` &&)=default

Static Public Member Functions

- static size_t & **count** ()
- static size_t & **limit** ()
- static void **set_limit** (const size_t __i)
- static void **throw_conditionally** ()

Public Attributes

- `std::size_t _M_i`

4.58.1 Detailed Description

Type throwing via limit condition.

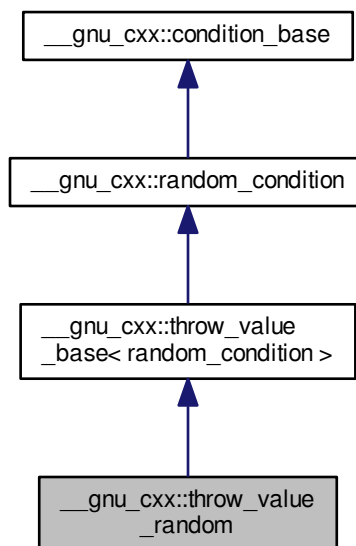
Definition at line 720 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw_allocator.h](#)

4.59 __gnu_cxx::throw_value_random Struct Reference

Inheritance diagram for `__gnu_cxx::throw_value_random`:



Public Types

- typedef `throw_value_base< random_condition >` **base_type**
- typedef `random_condition` **condition_type**

Public Member Functions

- **throw_value_random** (const [throw_value_random](#) &__other)
- **throw_value_random** ([throw_value_random](#) &&)=default
- **throw_value_random** (const std::size_t __i)
- [throw_value_base](#) & **operator++** ()
- [throw_value_random](#) & **operator=** (const [throw_value_random](#) &__other)
- [throw_value_random](#) & **operator=** ([throw_value_random](#) &&)=default
- void **seed** (unsigned long __s)

Static Public Member Functions

- static void **set_probability** (double __p)
- static void **throw_conditionally** ()

Public Attributes

- std::size_t **M_i**

4.59.1 Detailed Description

Type throwing via random condition.

Definition at line 751 of file [throw_allocator.h](#).

The documentation for this struct was generated from the following file:

- [throw_allocator.h](#)

4.60 [__gnu_debug::_After_nth_from<_Iterator>](#) Class Template Reference

Public Member Functions

- **_After_nth_from** (const difference_type &__n, const [_Iterator](#) &[__base](#))
- bool **operator()** (const [_Iterator](#) &__x) const

4.60.1 Detailed Description

```
template<typename _Iterator>
class \_\_gnu\_debug::\_After\_nth\_from<\_Iterator> {
```

A function object that returns true when the given random access iterator is at least *n* steps away from the given iterator.

Definition at line 77 of file [safe_sequence.h](#).

The documentation for this class was generated from the following file:

- [safe_sequence.h](#)

4.61 `__gnu_debug::_BeforeBeginHelper<_Sequence>` Struct Template Reference

Static Public Member Functions

- `template<typename _Iterator>`
`static bool _S_Is (const _Safe_iterator<_Iterator, _Sequence> &)`
- `template<typename _Iterator>`
`static bool _S_Is_Beginnest (const _Safe_iterator<_Iterator, _Sequence> &__it)`

4.61.1 Detailed Description

```
template<typename _Sequence>
struct __gnu_debug::_BeforeBeginHelper<_Sequence>
```

Helper struct to deal with sequence offering a `before_begin` iterator.

Definition at line 45 of file `safe_iterator.h`.

The documentation for this struct was generated from the following file:

- [safe_iterator.h](#)

4.62 `__gnu_debug::_Equal_to<_Type>` Class Template Reference

Public Member Functions

- `_Equal_to (const _Type &__v)`
- `bool operator() (const _Type &__x) const`

4.62.1 Detailed Description

```
template<typename _Type>
class __gnu_debug::_Equal_to<_Type>
```

A simple function object that returns true if the passed-in value is equal to the stored value.

Definition at line 62 of file `safe_sequence.h`.

The documentation for this class was generated from the following file:

- [safe_sequence.h](#)

4.63 `__gnu_debug::_Not_equal_to<_Type>` Class Template Reference

Public Member Functions

- `_Not_equal_to` (const _Type &__v)
- `bool operator()` (const _Type &__x) const

4.63.1 Detailed Description

```
template<typename _Type>
class __gnu_debug::_Not_equal_to<_Type>
```

A simple function object that returns true if the passed-in value is not equal to the stored value. It saves typing over using both `bind1st` and `not_equal`.

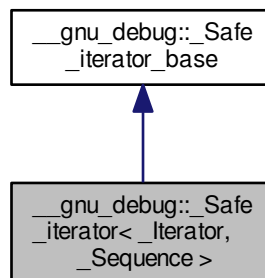
Definition at line 47 of file `safe_sequence.h`.

The documentation for this class was generated from the following file:

- [safe_sequence.h](#)

4.64 `__gnu_debug::_Safe_iterator<_Iterator, _Sequence>` Class Template Reference

Inheritance diagram for `__gnu_debug::_Safe_iterator<_Iterator, _Sequence>`:



Public Types

- `typedef _Traits::difference_type` **difference_type**
- `typedef _Traits::iterator_category` **iterator_category**
- `typedef _Iterator` **iterator_type**
- `typedef _Traits::pointer` **pointer**
- `typedef _Traits::reference` **reference**
- `typedef _Traits::value_type` **value_type**

Public Member Functions

- `_Safe_iterator` () noexcept
- `_Safe_iterator` (const `_Iterator` &__i, const `_Sequence` *__seq) noexcept
- `_Safe_iterator` (const `_Safe_iterator` &__x) noexcept
- `_Safe_iterator` (`_Safe_iterator` &&__x) noexcept
- template<typename `_MutableIterator` >
`_Safe_iterator` (const `_Safe_iterator`< `_MutableIterator`, typename `__gnu_cxx::__enable_if`<(std::__are_same<
`_MutableIterator`, typename `_Sequence::iterator::iterator_type` >::__value), `_Sequence` >::__type > &__x) noex-
cept
- void `_M_attach` (`_Safe_sequence_base` *__seq, bool __constant)
- void `_M_attach` (`_Safe_sequence_base` *__seq)
- void `_M_attach_single` (`_Safe_sequence_base` *__seq, bool __constant) throw ()
- void `_M_attach_single` (`_Safe_sequence_base` *__seq)
- bool `_M_attached_to` (const `_Safe_sequence_base` *__seq) const
- bool `_M_before_dereferenceable` () const
- bool `_M_can_advance` (const `difference_type` &__n) const
- bool `_M_can_compare` (const `_Safe_iterator_base` &__x) const throw ()
- bool `_M_decrementable` () const
- bool `_M_dereferenceable` () const
- void `_M_detach` ()
- void `_M_detach_single` () throw ()
- `__gnu_cxx::__conditional_type`< std::__are_same< `_Const_iterator`, `_Safe_iterator` >::__value, const `_Sequence` *,
`_Sequence` * >::__type `_M_get_sequence` () const
- bool `_M_incrementable` () const
- void `_M_invalidate` ()
- bool `_M_is_before_begin` () const
- bool `_M_is_begin` () const
- bool `_M_is_beginnest` () const
- bool `_M_is_end` () const
- void `_M_reset` () throw ()
- bool `_M_singular` () const throw ()
- void `_M_unlink` () throw ()
- bool `_M_valid_range` (const `_Safe_iterator` &__rhs) const
- `_Iterator base` () const noexcept
- `operator _Iterator` () const noexcept
- reference `operator*` () const noexcept
- `_Safe_iterator operator+` (const `difference_type` &__n) const noexcept
- `_Safe_iterator & operator++` () noexcept
- `_Safe_iterator operator++` (int) noexcept
- `_Safe_iterator & operator+=` (const `difference_type` &__n) noexcept
- `_Safe_iterator operator-` (const `difference_type` &__n) const noexcept
- `_Safe_iterator & operator--` () noexcept
- `_Safe_iterator operator--` (int) noexcept
- `_Safe_iterator & operator-=` (const `difference_type` &__n) noexcept
- pointer `operator->` () const noexcept
- `_Safe_iterator & operator=` (const `_Safe_iterator` &__x) noexcept
- `_Safe_iterator & operator=` (`_Safe_iterator` &&__x) noexcept
- reference `operator[]` (const `difference_type` &__n) const noexcept

Public Attributes

- [_Safe_iterator_base](#) * [_M_next](#)
- [_Safe_iterator_base](#) * [_M_prior](#)
- [_Safe_sequence_base](#) * [_M_sequence](#)
- unsigned int [_M_version](#)

Protected Member Functions

- [__gnu_cxx::__mutex](#) & [_M_get_mutex](#) () throw ()

4.64.1 Detailed Description

```
template<typename _Iterator, typename _Sequence>
class __gnu_debug::Safe_iterator< _Iterator, _Sequence >
```

Safe iterator wrapper.

The class template `_Safe_iterator` is a wrapper around an iterator that tracks the iterator's movement among sequences and checks that operations performed on the "safe" iterator are legal. In addition to the basic iterator operations (which are validated, and then passed to the underlying iterator), `_Safe_iterator` has member functions for iterator invalidation, attaching/detaching the iterator from sequences, and querying the iterator's state.

Definition at line 46 of file `formatter.h`.

4.64.2 Constructor & Destructor Documentation

```
4.64.2.1 template<typename _Iterator, typename _Sequence> __gnu_debug::Safe_iterator< _Iterator, _Sequence
>::Safe_iterator( ) [inline], [noexcept]
```

Postcondition

the iterator is singular and unattached

Definition at line 138 of file `safe_iterator.h`.

```
4.64.2.2 template<typename _Iterator, typename _Sequence> __gnu_debug::Safe_iterator< _Iterator, _Sequence
>::Safe_iterator( const _Iterator & __i, const _Sequence * __seq ) [inline], [noexcept]
```

Safe iterator construction from an unsafe iterator and its sequence.

Precondition

`seq` is not NULL

Postcondition

this is not singular

Definition at line 147 of file `safe_iterator.h`.

4.64.2.3 `template<typename _Iterator, typename _Sequence> __gnu_debug:: Safe_iterator< _Iterator, _Sequence >:: Safe_iterator (const _Safe_iterator< _Iterator, _Sequence > &__x) [inline], [noexcept]`

Copy construction.

Definition at line 159 of file `safe_iterator.h`.

4.64.2.4 `template<typename _Iterator, typename _Sequence> __gnu_debug:: Safe_iterator< _Iterator, _Sequence >:: Safe_iterator (_Safe_iterator< _Iterator, _Sequence > &&__x) [inline], [noexcept]`

Move construction.

Postcondition

__x is singular and unattached

Definition at line 176 of file `safe_iterator.h`.

References `std::swap()`.

4.64.2.5 `template<typename _Iterator, typename _Sequence> template<typename _MutableIterator > __gnu_debug:: Safe_iterator< _Iterator, _Sequence >:: Safe_iterator (const _Safe_iterator< _MutableIterator, typename __gnu_cxx:: enable_if<(std:: are_same< _MutableIterator, typename _Sequence::iterator::iterator_type >:: value), _Sequence >:: type > &__x) [inline], [noexcept]`

Converting constructor from a mutable iterator to a constant iterator.

Definition at line 194 of file `safe_iterator.h`.

4.64.3 Member Function Documentation

4.64.3.1 `void __gnu_debug:: Safe_iterator_base:: M_attach (_Safe_sequence_base * __seq, bool __constant) [inherited]`

Attaches this iterator to the given sequence, detaching it from whatever sequence it was attached to originally. If the new sequence is the NULL pointer, the iterator is left unattached.

Referenced by `__gnu_debug:: Safe_local_iterator< _Iterator, _Sequence >:: M_attach()`, `__gnu_debug:: Safe_iterator< _Iterator, _Sequence >:: M_attach()`, `__gnu_debug:: Safe_sequence_base:: M_invalidate_all()`, and `__gnu_debug:: Safe_iterator_base:: Safe_iterator_base()`.

4.64.3.2 `template<typename _Iterator, typename _Sequence> void __gnu_debug:: Safe_iterator< _Iterator, _Sequence >:: M_attach (_Safe_sequence_base * __seq) [inline]`

Attach iterator to the given sequence.

Definition at line 402 of file `safe_iterator.h`.

References `__gnu_debug:: Safe_iterator_base:: M_attach()`.

4.64.3.3 `void __gnu_debug::Safe_iterator_base::M_attach_single (_Safe_sequence_base * __seq, bool __constant) throw`
`[inherited]`

Likewise, but not thread-safe.

Referenced by `__gnu_debug::Safe_local_iterator< _Iterator, _Sequence >::M_attach_single()`, `__gnu_debug::Safe_iterator< _Iterator, _Sequence >::M_attach_single()`, `__gnu_debug::Safe_sequence_base::M_invalidate_all()`, and `__gnu_debug::Safe_iterator_base::Safe_iterator_base()`.

4.64.3.4 `template<typename _Iterator, typename _Sequence> void __gnu_debug::Safe_iterator< _Iterator, _Sequence >::M_attach_single (_Safe_sequence_base * __seq)` `[inline]`

Likewise, but not thread-safe.

Definition at line 409 of file `safe_iterator.h`.

References `__gnu_debug::Safe_iterator_base::M_attach_single()`.

4.64.3.5 `bool __gnu_debug::Safe_iterator_base::M_attached_to (const _Safe_sequence_base * __seq) const`
`[inline], [inherited]`

Determines if we are attached to the given sequence.

Definition at line 129 of file `safe_base.h`.

References `__gnu_debug::Safe_iterator_base::M_can_compare()`, and `__gnu_debug::Safe_iterator_base::M_singular()`.

4.64.3.6 `template<typename _Iterator, typename _Sequence> bool __gnu_debug::Safe_iterator< _Iterator, _Sequence >::M_before_dereferenceable () const` `[inline]`

Is the iterator before a dereferenceable one?

Definition at line 421 of file `safe_iterator.h`.

References `__gnu_debug::base()`.

4.64.3.7 `bool __gnu_debug::Safe_iterator_base::M_can_compare (const _Safe_iterator_base & __x) const throw`
`[inherited]`

Can we compare this iterator to the given iterator `__x`? Returns true if both iterators are nonsingular and reference the same sequence.

Referenced by `__gnu_debug::Safe_iterator_base::M_attached_to()`, and `__gnu_debug::Safe_iterator< _Iterator, _Sequence >::M_is_beginnest()`.

4.64.3.8 `template<typename _Iterator, typename _Sequence> bool __gnu_debug::Safe_iterator< _Iterator, _Sequence >::M_dereferenceable () const` `[inline]`

Is the iterator dereferenceable?

Definition at line 416 of file `safe_iterator.h`.

Referenced by `__gnu_debug::check_dereferenceable()`.

4.64.3.9 `void __gnu_debug::Safe_iterator_base::M_detach ()` [inherited]

Detach the iterator for whatever sequence it is attached to, if any.

Referenced by `__gnu_debug::Safe_sequence_base::M_invalidate_all()`, and `__gnu_debug::Safe_iterator_base::↵_Safe_iterator_base()`.

4.64.3.10 `void __gnu_debug::Safe_iterator_base::M_detach_single () throw` [inherited]

Likewise, but not thread-safe.

Referenced by `__gnu_debug::Safe_sequence_base::M_invalidate_all()`, and `__gnu_debug::Safe_iterator_base::↵_Safe_iterator_base()`.

4.64.3.11 `__gnu_cxx::mutex& __gnu_debug::Safe_iterator_base::M_get_mutex () throw` [protected],
[inherited]

For use in `_Safe_iterator`.

Referenced by `__gnu_debug::Safe_iterator_base::Safe_iterator_base()`, and `__gnu_debug::Safe_sequence_↵base::~~Safe_sequence_base()`.

4.64.3.12 `template<typename _Iterator, typename _Sequence> bool __gnu_debug::Safe_iterator<_Iterator, _Sequence>::M_incrementable () const` [inline]

Is the iterator incrementable?

Definition at line 433 of file `safe_iterator.h`.

4.64.3.13 `void __gnu_debug::Safe_iterator_base::M_invalidate ()` [inline], [inherited]

Invalidate the iterator, making it singular.

Definition at line 142 of file `safe_base.h`.

References `__gnu_debug::Safe_iterator_base::M_reset()`.

4.64.3.14 `template<typename _Iterator, typename _Sequence> bool __gnu_debug::Safe_iterator<_Iterator, _Sequence>::M_is_before_begin () const` [inline]

Is this iterator equal to the sequence's `before_begin()` iterator if any?

Definition at line 470 of file `safe_iterator.h`.

4.64.3.15 `template<typename _Iterator, typename _Sequence> bool __gnu_debug::Safe_iterator<_Iterator, _Sequence>::M_is_begin () const` [inline]

Is this iterator equal to the sequence's `begin()` iterator?

Definition at line 459 of file `safe_iterator.h`.

4.64.3.16 `template<typename _Iterator, typename _Sequence> bool __gnu_debug::_Safe_iterator<_Iterator, _Sequence>::_M_is_beginnest () const [inline]`

Is this iterator equal to the sequence's before_begin() iterator if any or begin() otherwise?

Definition at line 476 of file safe_iterator.h.

References `__gnu_debug::_Safe_iterator_base::_M_can_compare()`, `__gnu_debug::_Safe_iterator_base::_M_is_singular()`, and `__gnu_debug::_Safe_iterator<_Iterator, _Sequence>::base()`.

4.64.3.17 `template<typename _Iterator, typename _Sequence> bool __gnu_debug::_Safe_iterator<_Iterator, _Sequence>::_M_is_end () const [inline]`

Is this iterator equal to the sequence's end() iterator?

Definition at line 464 of file safe_iterator.h.

4.64.3.18 `void __gnu_debug::_Safe_iterator_base::_M_reset () throw () [inherited]`

Reset all member variables

Referenced by `__gnu_debug::_Safe_iterator_base::_M_invalidate()`.

4.64.3.19 `bool __gnu_debug::_Safe_iterator_base::_M_singular () const throw () [inherited]`

Is this iterator singular?

Referenced by `__gnu_debug::_check_singular_aux()`, `__gnu_debug::_Safe_iterator_base::_M_attached_to()`, `__gnu_debug::_Safe_local_iterator<_Iterator, _Sequence>::_M_dereferenceable()`, `__gnu_debug::_Safe_local_iterator<_Iterator, _Sequence>::_M_incrementable()`, `__gnu_debug::_Safe_iterator<_Iterator, _Sequence>::_M_is_beginnest()`, and `__gnu_debug::_Safe_local_iterator<_Iterator, _Sequence>::_Safe_local_iterator()`.

4.64.3.20 `void __gnu_debug::_Safe_iterator_base::_M_unlink () throw () [inline], [inherited]`

Unlink itself

Definition at line 151 of file safe_base.h.

References `__gnu_debug::_Safe_iterator_base::_M_next`, and `__gnu_debug::_Safe_iterator_base::_M_prior`.

4.64.3.21 `template<typename _Iterator, typename _Sequence> _Iterator __gnu_debug::_Safe_iterator<_Iterator, _Sequence>::base () const [inline], [noexcept]`

Return the underlying iterator.

Definition at line 392 of file safe_iterator.h.

Referenced by `__gnu_debug::_Safe_iterator<_Iterator, _Sequence>::_M_is_beginnest()`.

4.64.3.22 `template<typename _Iterator, typename _Sequence> __gnu_debug::_Safe_iterator<_Iterator, _Sequence>::operator _Iterator () const` `[inline], [noexcept]`

Conversion to underlying non-debug iterator to allow better interaction with non-debug containers.

Definition at line 398 of file `safe_iterator.h`.

4.64.3.23 `template<typename _Iterator, typename _Sequence> reference __gnu_debug::_Safe_iterator<_Iterator, _Sequence>::operator* () const` `[inline], [noexcept]`

Iterator dereference.

Precondition

iterator is dereferenceable

Definition at line 257 of file `safe_iterator.h`.

4.64.3.24 `template<typename _Iterator, typename _Sequence> _Safe_iterator& __gnu_debug::_Safe_iterator<_Iterator, _Sequence>::operator++ ()` `[inline], [noexcept]`

Iterator preincrement.

Precondition

iterator is incrementable

Definition at line 285 of file `safe_iterator.h`.

4.64.3.25 `template<typename _Iterator, typename _Sequence> _Safe_iterator __gnu_debug::_Safe_iterator<_Iterator, _Sequence>::operator++ (int)` `[inline], [noexcept]`

Iterator postincrement.

Precondition

iterator is incrementable

Definition at line 299 of file `safe_iterator.h`.

4.64.3.26 `template<typename _Iterator, typename _Sequence> _Safe_iterator& __gnu_debug::_Safe_iterator<_Iterator, _Sequence>::operator-- ()` `[inline], [noexcept]`

Iterator predecrement.

Precondition

iterator is decrementable

Definition at line 315 of file `safe_iterator.h`.

4.64.3.27 `template<typename _Iterator, typename _Sequence> _Safe_iterator __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator--(int) [inline], [noexcept]`

Iterator postdecrement.

Precondition

iterator is decrementable

Definition at line 329 of file `safe_iterator.h`.

4.64.3.28 `template<typename _Iterator, typename _Sequence> pointer __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator->() const [inline], [noexcept]`

Iterator dereference.

Precondition

iterator is dereferenceable

Todo Make this correct w.r.t. iterators that return proxies

Definition at line 271 of file `safe_iterator.h`.

References `std::__addressof()`.

4.64.3.29 `template<typename _Iterator, typename _Sequence> _Safe_iterator& __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator=(const _Safe_iterator< _Iterator, _Sequence > &__x) [inline], [noexcept]`

Copy assignment.

Definition at line 214 of file `safe_iterator.h`.

4.64.3.30 `template<typename _Iterator, typename _Sequence> _Safe_iterator& __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator=(_Safe_iterator< _Iterator, _Sequence > &&__x) [inline], [noexcept]`

Move assignment.

Postcondition

`__x` is singular and unattached

Definition at line 234 of file `safe_iterator.h`.

4.64.4 Member Data Documentation

4.64.4.1 _Safe_iterator_base* __gnu_debug::_Safe_iterator_base::_M_next [inherited]

Pointer to the next iterator in the sequence's list of iterators. Only valid when _M_sequence != NULL.

Definition at line 72 of file safe_base.h.

Referenced by __gnu_debug::_Safe_iterator_base::_M_unlink().

4.64.4.2 _Safe_iterator_base* __gnu_debug::_Safe_iterator_base::_M_prior [inherited]

Pointer to the previous iterator in the sequence's list of iterators. Only valid when _M_sequence != NULL.

Definition at line 68 of file safe_base.h.

Referenced by __gnu_debug::_Safe_iterator_base::_M_unlink().

4.64.4.3 _Safe_sequence_base* __gnu_debug::_Safe_iterator_base::_M_sequence [inherited]

The sequence this iterator references; may be NULL to indicate a singular iterator.

Definition at line 55 of file safe_base.h.

Referenced by __gnu_debug::_Safe_local_iterator<_Iterator, _Sequence>::_M_incrementable(), __gnu_debug::_Safe_iterator_base::_Safe_iterator_base(), and __gnu_debug::_Safe_local_iterator_base::_Safe_local_iterator_base().

4.64.4.4 unsigned int __gnu_debug::_Safe_iterator_base::_M_version [inherited]

The version number of this iterator. The sentinel value 0 is used to indicate an invalidated iterator (i.e., one that is singular because of an operation on the container). This version number must equal the version number in the sequence referenced by _M_sequence for the iterator to be non-singular.

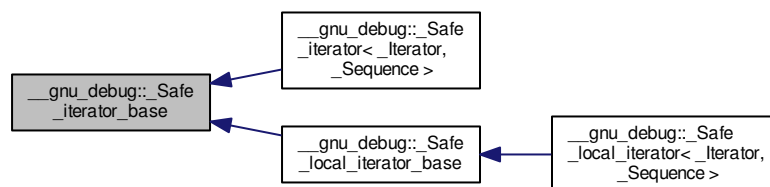
Definition at line 64 of file safe_base.h.

The documentation for this class was generated from the following files:

- [formatter.h](#)
- [safe_iterator.h](#)

4.65 __gnu_debug::_Safe_iterator_base Class Reference

Inheritance diagram for __gnu_debug::_Safe_iterator_base:



Public Member Functions

- void `_M_attach` (`_Safe_sequence_base * __seq`, `bool __constant`)
- void `_M_attach_single` (`_Safe_sequence_base * __seq`, `bool __constant`) throw ()
- bool `_M_attached_to` (`const _Safe_sequence_base * __seq`) const
- bool `_M_can_compare` (`const _Safe_iterator_base & __x`) const throw ()
- void `_M_detach` ()
- void `_M_detach_single` () throw ()
- void `_M_invalidate` ()
- void `_M_reset` () throw ()
- bool `_M_singular` () const throw ()
- void `_M_unlink` () throw ()

Public Attributes

- `_Safe_iterator_base * _M_next`
- `_Safe_iterator_base * _M_prior`
- `_Safe_sequence_base * _M_sequence`
- unsigned int `_M_version`

Protected Member Functions

- `_Safe_iterator_base` ()
- `_Safe_iterator_base` (`const _Safe_sequence_base * __seq`, `bool __constant`)
- `_Safe_iterator_base` (`const _Safe_iterator_base & __x`, `bool __constant`)
- `_Safe_iterator_base` (`const _Safe_iterator_base &`)
- `__gnu_cxx::__mutex & _M_get_mutex` () throw ()
- `_Safe_iterator_base & operator=` (`const _Safe_iterator_base &`)

4.65.1 Detailed Description

Basic functionality for a *safe* iterator.

The `_Safe_iterator_base` base class implements the functionality of a safe iterator that is not specific to a particular iterator type. It contains a pointer back to the sequence it references along with iterator version information and pointers to form a doubly-linked list of iterators referenced by the container.

This class must not perform any operations that can throw an exception, or the exception guarantees of derived iterators will be broken.

Definition at line 50 of file `safe_base.h`.

4.65.2 Constructor & Destructor Documentation**4.65.2.1 `__gnu_debug::_Safe_iterator_base::_Safe_iterator_base ()` `[inline]`, `[protected]`**

Initializes the iterator and makes it singular.

Definition at line 76 of file `safe_base.h`.

Referenced by `_Safe_iterator_base()`.

4.65.2.2 `__gnu_debug::Safe_iterator_base::Safe_iterator_base (const _Safe_sequence_base * __seq, bool __constant)`
`[inline], [protected]`

Initialize the iterator to reference the sequence pointed to by `__seq`. `__constant` is true when we are initializing a constant iterator, and false if it is a mutable iterator. Note that `__seq` may be NULL, in which case the iterator will be singular. Otherwise, the iterator will reference `__seq` and be nonsingular.

Definition at line 87 of file `safe_base.h`.

References `_M_attach()`.

4.65.2.3 `__gnu_debug::Safe_iterator_base::Safe_iterator_base (const _Safe_iterator_base & __x, bool __constant)`
`[inline], [protected]`

Initializes the iterator to reference the same sequence that `__x` does. `__constant` is true if this is a constant iterator, and false if it is mutable.

Definition at line 94 of file `safe_base.h`.

References `_M_attach()`, `_M_attach_single()`, `_M_detach()`, `_M_detach_single()`, `_M_get_mutex()`, `_M_sequence`, and `_Safe_iterator_base()`.

4.65.3 Member Function Documentation

4.65.3.1 `void __gnu_debug::Safe_iterator_base::M_attach (_Safe_sequence_base * __seq, bool __constant)`

Attaches this iterator to the given sequence, detaching it from whatever sequence it was attached to originally. If the new sequence is the NULL pointer, the iterator is left unattached.

Referenced by `__gnu_debug::Safe_local_iterator< _Iterator, _Sequence >::M_attach()`, `__gnu_debug::Safe_iterator< _Iterator, _Sequence >::M_attach()`, `__gnu_debug::Safe_sequence_base::M_invalidate_all()`, and `_Safe_iterator_base()`.

4.65.3.2 `void __gnu_debug::Safe_iterator_base::M_attach_single (_Safe_sequence_base * __seq, bool __constant) throw ()`

Likewise, but not thread-safe.

Referenced by `__gnu_debug::Safe_local_iterator< _Iterator, _Sequence >::M_attach_single()`, `__gnu_debug::Safe_iterator< _Iterator, _Sequence >::M_attach_single()`, `__gnu_debug::Safe_sequence_base::M_invalidate_all()`, and `_Safe_iterator_base()`.

4.65.3.3 `bool __gnu_debug::Safe_iterator_base::M_attached_to (const _Safe_sequence_base * __seq) const`
`[inline]`

Determines if we are attached to the given sequence.

Definition at line 129 of file `safe_base.h`.

References `_M_can_compare()`, and `_M_singular()`.

4.65.3.4 `bool __gnu_debug::_Safe_iterator_base::_M_can_compare (const _Safe_iterator_base & __x) const throw ()`

Can we compare this iterator to the given iterator `__x`? Returns true if both iterators are nonsingular and reference the same sequence.

Referenced by `_M_attached_to()`, and `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_is_beginnest()`.

4.65.3.5 `void __gnu_debug::_Safe_iterator_base::_M_detach ()`

Detach the iterator for whatever sequence it is attached to, if any.

Referenced by `__gnu_debug::_Safe_sequence_base::_M_invalidate_all()`, and `_Safe_iterator_base()`.

4.65.3.6 `void __gnu_debug::_Safe_iterator_base::_M_detach_single () throw ()`

Likewise, but not thread-safe.

Referenced by `__gnu_debug::_Safe_sequence_base::_M_invalidate_all()`, and `_Safe_iterator_base()`.

4.65.3.7 `__gnu_cxx::mutex& __gnu_debug::_Safe_iterator_base::_M_get_mutex () throw ()` `[protected]`

For use in `_Safe_iterator`.

Referenced by `_Safe_iterator_base()`, and `__gnu_debug::_Safe_sequence_base::~~_Safe_sequence_base()`.

4.65.3.8 `void __gnu_debug::_Safe_iterator_base::_M_invalidate ()` `[inline]`

Invalidate the iterator, making it singular.

Definition at line 142 of file `safe_base.h`.

References `_M_reset()`.

4.65.3.9 `void __gnu_debug::_Safe_iterator_base::_M_reset () throw ()`

Reset all member variables

Referenced by `_M_invalidate()`.

4.65.3.10 `bool __gnu_debug::_Safe_iterator_base::_M_singular () const throw ()`

Is this iterator singular?

Referenced by `__gnu_debug::_check_singular_aux()`, `_M_attached_to()`, `__gnu_debug::_Safe_local_iterator< _↵ Iterator, _Sequence >::_M_dereferenceable()`, `__gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::_M_↵ incrementable()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_is_beginnest()`, and `__gnu_debug::_↵ Safe_local_iterator< _Iterator, _Sequence >::_Safe_local_iterator()`.

4.65.3.11 `void __gnu_debug::Safe_iterator_base::_M_unlink () throw ()` `[inline]`

Unlink itself

Definition at line 151 of file `safe_base.h`.

References `_M_next`, and `_M_prior`.

4.65.4 Member Data Documentation

4.65.4.1 `_Safe_iterator_base* __gnu_debug::Safe_iterator_base::_M_next`

Pointer to the next iterator in the sequence's list of iterators. Only valid when `_M_sequence != NULL`.

Definition at line 72 of file `safe_base.h`.

Referenced by `_M_unlink()`.

4.65.4.2 `_Safe_iterator_base* __gnu_debug::Safe_iterator_base::_M_prior`

Pointer to the previous iterator in the sequence's list of iterators. Only valid when `_M_sequence != NULL`.

Definition at line 68 of file `safe_base.h`.

Referenced by `_M_unlink()`.

4.65.4.3 `_Safe_sequence_base* __gnu_debug::Safe_iterator_base::_M_sequence`

The sequence this iterator references; may be `NULL` to indicate a singular iterator.

Definition at line 55 of file `safe_base.h`.

Referenced by `__gnu_debug::Safe_local_iterator<_Iterator, _Sequence>::_M_incrementable()`, `_Safe_iterator_base::base()`, and `__gnu_debug::Safe_local_iterator_base::_Safe_local_iterator_base()`.

4.65.4.4 `unsigned int __gnu_debug::Safe_iterator_base::_M_version`

The version number of this iterator. The sentinel value 0 is used to indicate an invalidated iterator (i.e., one that is singular because of an operation on the container). This version number must equal the version number in the sequence referenced by `_M_sequence` for the iterator to be non-singular.

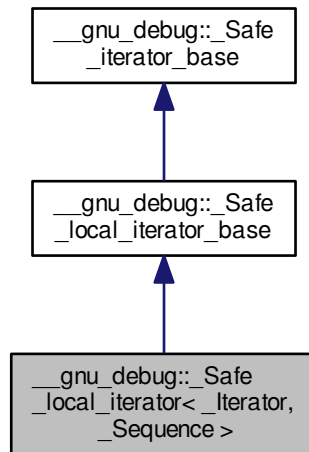
Definition at line 64 of file `safe_base.h`.

The documentation for this class was generated from the following file:

- [safe_base.h](#)

4.66 `__gnu_debug::Safe_local_iterator<_Iterator, _Sequence>` Class Template Reference

Inheritance diagram for `__gnu_debug::Safe_local_iterator<_Iterator, _Sequence>`:



Public Types

- typedef `_Traits::difference_type` **difference_type**
- typedef `_Traits::iterator_category` **iterator_category**
- typedef `_Iterator` **iterator_type**
- typedef `_Traits::pointer` **pointer**
- typedef `_Traits::reference` **reference**
- typedef `_Traits::value_type` **value_type**

Public Member Functions

- [_Safe_local_iterator](#) ()
- [_Safe_local_iterator](#) (const `_Iterator` &__i, const `_Sequence` * __seq)
- [_Safe_local_iterator](#) (const [_Safe_local_iterator](#) &__x)
- template<typename `_MutableIterator` >
[_Safe_local_iterator](#) (const [_Safe_local_iterator](#) < `_MutableIterator`, typename `__gnu_cxx::enable_if< std::is_same< _MutableIterator, typename _Sequence::local_iterator::iterator_type >::value, _Sequence >::value_type > &__x`)
- void [_M_attach](#) ([_Safe_sequence_base](#) * __seq, bool __constant)
- void [_M_attach](#) ([_Safe_sequence_base](#) * __seq)
- void [_M_attach_single](#) ([_Safe_sequence_base](#) * __seq, bool __constant) throw ()
- void [_M_attach_single](#) ([_Safe_sequence_base](#) * __seq)
- bool [_M_attached_to](#) (const [_Safe_sequence_base](#) * __seq) const

- `bool _M_can_compare (const _Safe_iterator_base &__x) const` throw ()
- `bool _M_dereferenceable () const`
- `void _M_detach ()`
- `void _M_detach_single ()` throw ()
- `__gnu_cxx::__conditional_type< std::__are_same< _Const_local_iterator, _Safe_local_iterator >::__value, const _Sequence *, _Sequence * >::__type _M_get_sequence () const`
- `template<typename _Other > bool _M_in_same_bucket (const _Safe_local_iterator< _Other, _Sequence > &__other) const`
- `bool _M_incrementable () const`
- `void _M_invalidate ()`
- `bool _M_is_begin () const`
- `bool _M_is_end () const`
- `void _M_reset ()` throw ()
- `bool _M_singular () const` throw ()
- `void _M_unlink ()` throw ()
- `bool _M_valid_range (const _Safe_local_iterator &__rhs) const`
- `_Iterator base () const`
- `size_type bucket () const`
- `operator _Iterator () const`
- `reference operator* () const`
- `_Safe_local_iterator & operator++ ()`
- `_Safe_local_iterator operator++ (int)`
- `pointer operator-> () const`
- `_Safe_local_iterator & operator= (const _Safe_local_iterator &__x)`

Public Attributes

- `_Safe_iterator_base * _M_next`
- `_Safe_iterator_base * _M_prior`
- `_Safe_sequence_base * _M_sequence`
- `unsigned int _M_version`

Protected Member Functions

- `_Safe_unordered_container_base * _M_get_container () const` noexcept
- `__gnu_cxx::__mutex & _M_get_mutex ()` throw ()

4.66.1 Detailed Description

```
template<typename _Iterator, typename _Sequence>
class __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >
```

Safe iterator wrapper.

The class template `_Safe_local_iterator` is a wrapper around an iterator that tracks the iterator's movement among sequences and checks that operations performed on the "safe" iterator are legal. In addition to the basic iterator operations (which are validated, and then passed to the underlying iterator), `_Safe_local_iterator` has member functions for iterator invalidation, attaching/detaching the iterator from sequences, and querying the iterator's state.

Definition at line 49 of file `formatter.h`.

4.66.2 Constructor & Destructor Documentation

4.66.2.1 `template<typename _Iterator, typename _Sequence> __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::_Safe_local_iterator () [inline]`

Postcondition

the iterator is singular and unattached

Definition at line 80 of file `safe_local_iterator.h`.

4.66.2.2 `template<typename _Iterator, typename _Sequence> __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::_Safe_local_iterator (const _Iterator & __i, const _Sequence * __seq) [inline]`

Safe iterator construction from an unsafe iterator and its sequence.

Precondition

`seq` is not NULL

Postcondition

this is not singular

Definition at line 89 of file `safe_local_iterator.h`.

References `__gnu_debug::_Safe_iterator_base::_M_singular()`.

4.66.2.3 `template<typename _Iterator, typename _Sequence> __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::_Safe_local_iterator (const _Safe_local_iterator< _Iterator, _Sequence > & __x) [inline]`

Copy construction.

Definition at line 100 of file `safe_local_iterator.h`.

4.66.2.4 `template<typename _Iterator, typename _Sequence> template<typename _MutableIterator > __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::_Safe_local_iterator (const _Safe_local_iterator< _MutableIterator, typename __gnu_cxx::__enable_if< std::__are_same< _MutableIterator, typename _Sequence::local_iterator::iterator_type >::_value, _Sequence >::_type > & __x) [inline]`

Converting constructor from a mutable iterator to a constant iterator.

Definition at line 118 of file `safe_local_iterator.h`.

4.66.3 Member Function Documentation

4.66.3.1 `void __gnu_debug::Safe_local_iterator_base::M_attach (_Safe_sequence_base * __seq, bool __constant)`
`[inherited]`

Attaches this iterator to the given container, detaching it from whatever container it was attached to originally. If the new container is the NULL pointer, the iterator is left unattached.

Referenced by `__gnu_debug::Safe_local_iterator_base::Safe_local_iterator_base()`.

4.66.3.2 `template<typename _Iterator, typename _Sequence> void __gnu_debug::Safe_local_iterator<_Iterator, _Sequence>::M_attach (_Safe_sequence_base * __seq)` `[inline]`

Attach iterator to the given sequence.

Definition at line 232 of file `safe_local_iterator.h`.

References `__gnu_debug::Safe_iterator_base::M_attach()`.

Referenced by `__gnu_debug::Safe_local_iterator<_Iterator, _Sequence>::operator=()`.

4.66.3.3 `void __gnu_debug::Safe_local_iterator_base::M_attach_single (_Safe_sequence_base * __seq, bool __constant)`
`throw)` `[inherited]`

Likewise, but not thread-safe.

Referenced by `__gnu_debug::Safe_local_iterator_base::Safe_local_iterator_base()`.

4.66.3.4 `template<typename _Iterator, typename _Sequence> void __gnu_debug::Safe_local_iterator<_Iterator, _Sequence>::M_attach_single (_Safe_sequence_base * __seq)` `[inline]`

Likewise, but not thread-safe.

Definition at line 237 of file `safe_local_iterator.h`.

References `__gnu_debug::Safe_iterator_base::M_attach_single()`.

4.66.3.5 `bool __gnu_debug::Safe_iterator_base::M_attached_to (const _Safe_sequence_base * __seq) const`
`[inline], [inherited]`

Determines if we are attached to the given sequence.

Definition at line 129 of file `safe_base.h`.

References `__gnu_debug::Safe_iterator_base::M_can_compare()`, and `__gnu_debug::Safe_iterator_base::M_↔singular()`.

4.66.3.6 `bool __gnu_debug::_Safe_iterator_base::_M_can_compare (const _Safe_iterator_base & __x) const throw)`
`[inherited]`

Can we compare this iterator to the given iterator `__x`? Returns true if both iterators are nonsingular and reference the same sequence.

Referenced by `__gnu_debug::_Safe_iterator_base::_M_attached_to()`, and `__gnu_debug::_Safe_iterator<_Iterator, _Sequence>::_M_is_beginnest()`.

4.66.3.7 `template<typename _Iterator, typename _Sequence> bool __gnu_debug::_Safe_local_iterator<_Iterator, _Sequence>::_M_dereferenceable () const` `[inline]`

Is the iterator dereferenceable?

Definition at line 242 of file `safe_local_iterator.h`.

References `__gnu_debug::_Safe_local_iterator<_Iterator, _Sequence>::_M_is_end()`, and `__gnu_debug::_Safe_iterator_base::_M_singular()`.

Referenced by `__gnu_debug::_Safe_local_iterator<_Iterator, _Sequence>::operator*()`, and `__gnu_debug::_Safe_local_iterator<_Iterator, _Sequence>::operator->()`.

4.66.3.8 `void __gnu_debug::_Safe_local_iterator_base::_M_detach ()` `[inherited]`

Detach the iterator for whatever container it is attached to, if any.

Referenced by `__gnu_debug::_Safe_local_iterator_base::_Safe_local_iterator_base()`.

4.66.3.9 `void __gnu_debug::_Safe_local_iterator_base::_M_detach_single () throw)` `[inherited]`

Likewise, but not thread-safe.

Referenced by `__gnu_debug::_Safe_local_iterator_base::_Safe_local_iterator_base()`.

4.66.3.10 `__gnu_cxx::mutex& __gnu_debug::_Safe_iterator_base::_M_get_mutex () throw)` `[protected]`,
`[inherited]`

For use in `_Safe_iterator`.

Referenced by `__gnu_debug::_Safe_iterator_base::_Safe_iterator_base()`, and `__gnu_debug::_Safe_sequence_base::~~Safe_sequence_base()`.

4.66.3.11 `template<typename _Iterator, typename _Sequence> template<typename _Other> bool __gnu_debug::_Safe_local_iterator<_Iterator, _Sequence>::_M_in_same_bucket (const _Safe_local_iterator<_Other, _Sequence> & __other) const` `[inline]`

Is this iterator part of the same bucket as the other one?

Definition at line 274 of file `safe_local_iterator.h`.

References `__gnu_debug::_Safe_local_iterator<_Iterator, _Sequence>::bucket()`.

4.66.3.12 `template<typename _Iterator, typename _Sequence> bool __gnu_debug::Safe_local_iterator<_Iterator, _Sequence>::_M_incrementable () const [inline]`

Is the iterator incrementable?

Definition at line 247 of file `safe_local_iterator.h`.

References `__gnu_debug::Safe_local_iterator<_Iterator, _Sequence>::_M_is_end()`, `__gnu_debug::Safe_iterator_base::_M_sequence`, and `__gnu_debug::Safe_iterator_base::_M_singular()`.

Referenced by `__gnu_debug::Safe_local_iterator<_Iterator, _Sequence>::operator++()`.

4.66.3.13 `void __gnu_debug::Safe_iterator_base::_M_invalidate () [inline], [inherited]`

Invalidate the iterator, making it singular.

Definition at line 142 of file `safe_base.h`.

References `__gnu_debug::Safe_iterator_base::_M_reset()`.

4.66.3.14 `template<typename _Iterator, typename _Sequence> bool __gnu_debug::Safe_local_iterator<_Iterator, _Sequence>::_M_is_begin () const [inline]`

Is this iterator equal to the sequence's begin(bucket) iterator?

Definition at line 264 of file `safe_local_iterator.h`.

References `__gnu_debug::Safe_local_iterator<_Iterator, _Sequence>::base()`, and `__gnu_debug::Safe_local_iterator<_Iterator, _Sequence>::bucket()`.

4.66.3.15 `template<typename _Iterator, typename _Sequence> bool __gnu_debug::Safe_local_iterator<_Iterator, _Sequence>::_M_is_end () const [inline]`

Is this iterator equal to the sequence's end(bucket) iterator?

Definition at line 268 of file `safe_local_iterator.h`.

References `__gnu_debug::Safe_local_iterator<_Iterator, _Sequence>::base()`, and `__gnu_debug::Safe_local_iterator<_Iterator, _Sequence>::bucket()`.

Referenced by `__gnu_debug::Safe_local_iterator<_Iterator, _Sequence>::_M_dereferenceable()`, and `__gnu_debug::Safe_local_iterator<_Iterator, _Sequence>::_M_incrementable()`.

4.66.3.16 `void __gnu_debug::Safe_iterator_base::_M_reset () throw () [inherited]`

Reset all member variables

Referenced by `__gnu_debug::Safe_iterator_base::_M_invalidate()`.

4.66.3.17 `bool __gnu_debug::_Safe_iterator_base::_M_singular () const throw)` `[inherited]`

Is this iterator singular?

Referenced by `__gnu_debug::_check_singular_aux()`, `__gnu_debug::_Safe_iterator_base::_M_attached_to()`, `__gnu_debug::_Safe_local_iterator<_Iterator, _Sequence>::_M_dereferenceable()`, `__gnu_debug::_Safe_local_iterator<_Iterator, _Sequence>::_M_incrementable()`, `__gnu_debug::_Safe_iterator<_Iterator, _Sequence>::_M_is_beginnest()`, and `__gnu_debug::_Safe_local_iterator<_Iterator, _Sequence>::_Safe_local_iterator()`.

4.66.3.18 `void __gnu_debug::_Safe_iterator_base::_M_unlink () throw)` `[inline]`, `[inherited]`

Unlink itself

Definition at line 151 of file `safe_base.h`.

References `__gnu_debug::_Safe_iterator_base::_M_next`, and `__gnu_debug::_Safe_iterator_base::_M_prior`.

4.66.3.19 `template<typename _Iterator, typename _Sequence> _Iterator __gnu_debug::_Safe_local_iterator<_Iterator, _Sequence>::base () const` `[inline]`

Return the underlying iterator.

Definition at line 216 of file `safe_local_iterator.h`.

Referenced by `__gnu_debug::_Safe_local_iterator<_Iterator, _Sequence>::_M_is_begin()`, and `__gnu_debug::_Safe_local_iterator<_Iterator, _Sequence>::_M_is_end()`.

4.66.3.20 `template<typename _Iterator, typename _Sequence> size_type __gnu_debug::_Safe_local_iterator<_Iterator, _Sequence>::bucket () const` `[inline]`

Return the bucket.

Definition at line 222 of file `safe_local_iterator.h`.

Referenced by `__gnu_debug::_Safe_local_iterator<_Iterator, _Sequence>::_M_in_same_bucket()`, `__gnu_debug::_Safe_local_iterator<_Iterator, _Sequence>::_M_is_begin()`, and `__gnu_debug::_Safe_local_iterator<_Iterator, _Sequence>::_M_is_end()`.

4.66.3.21 `template<typename _Iterator, typename _Sequence> __gnu_debug::_Safe_local_iterator<_Iterator, _Sequence>::operator _Iterator () const` `[inline]`

Conversion to underlying non-debug iterator to allow better interaction with non-debug containers.

Definition at line 228 of file `safe_local_iterator.h`.

4.66.3.22 `template<typename _Iterator, typename _Sequence> reference __gnu_debug::_Safe_local_iterator<_Iterator, _Sequence>::operator*() const [inline]`

Iterator dereference.

Precondition

iterator is dereferenceable

Definition at line 159 of file `safe_local_iterator.h`.

References `__gnu_debug::_Safe_local_iterator<_Iterator, _Sequence>::M_dereferenceable()`.

4.66.3.23 `template<typename _Iterator, typename _Sequence> _Safe_local_iterator& __gnu_debug::_Safe_local_iterator<_Iterator, _Sequence>::operator++() [inline]`

Iterator preincrement.

Precondition

iterator is incrementable

Definition at line 187 of file `safe_local_iterator.h`.

References `__gnu_debug::_Safe_local_iterator<_Iterator, _Sequence>::M_incrementable()`.

4.66.3.24 `template<typename _Iterator, typename _Sequence> _Safe_local_iterator __gnu_debug::_Safe_local_iterator<_Iterator, _Sequence>::operator++(int) [inline]`

Iterator postincrement.

Precondition

iterator is incrementable

Definition at line 201 of file `safe_local_iterator.h`.

References `__gnu_debug::_Safe_local_iterator<_Iterator, _Sequence>::M_incrementable()`.

4.66.3.25 `template<typename _Iterator, typename _Sequence> pointer __gnu_debug::_Safe_local_iterator<_Iterator, _Sequence>::operator->() const [inline]`

Iterator dereference.

Precondition

iterator is dereferenceable

Todo Make this correct w.r.t. iterators that return proxies

Definition at line 173 of file `safe_local_iterator.h`.

References `std::__addressof()`, and `__gnu_debug::_Safe_local_iterator<_Iterator, _Sequence>::M_dereferenceable()`.

```
4.66.3.26 template<typename _Iterator, typename _Sequence> _Safe_local_iterator& __gnu_debug::_Safe_local_↵
        iterator< _Iterator, _Sequence >::operator= ( const _Safe_local_iterator< _Iterator, _Sequence > & _x )
        [inline]
```

Copy assignment.

Definition at line 140 of file `safe_local_iterator.h`.

References `__gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::_M_attach()`.

4.66.4 Member Data Documentation

```
4.66.4.1 _Safe_iterator_base* __gnu_debug::_Safe_iterator_base::_M_next [inherited]
```

Pointer to the next iterator in the sequence's list of iterators. Only valid when `_M_sequence != NULL`.

Definition at line 72 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_iterator_base::_M_unlink()`.

```
4.66.4.2 _Safe_iterator_base* __gnu_debug::_Safe_iterator_base::_M_prior [inherited]
```

Pointer to the previous iterator in the sequence's list of iterators. Only valid when `_M_sequence != NULL`.

Definition at line 68 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_iterator_base::_M_unlink()`.

```
4.66.4.3 _Safe_sequence_base* __gnu_debug::_Safe_iterator_base::_M_sequence [inherited]
```

The sequence this iterator references; may be `NULL` to indicate a singular iterator.

Definition at line 55 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::_M_incrementable()`, `__gnu_debug::_↵`
`Safe_iterator_base::_Safe_iterator_base()`, and `__gnu_debug::_Safe_local_iterator_base::_Safe_local_iterator_base()`.

```
4.66.4.4 unsigned int __gnu_debug::_Safe_iterator_base::_M_version [inherited]
```

The version number of this iterator. The sentinel value 0 is used to indicate an invalidated iterator (i.e., one that is singular because of an operation on the container). This version number must equal the version number in the sequence referenced by `_M_sequence` for the iterator to be non-singular.

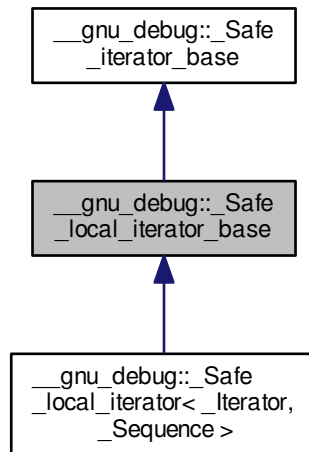
Definition at line 64 of file `safe_base.h`.

The documentation for this class was generated from the following files:

- [formatter.h](#)
- [safe_local_iterator.h](#)

4.67 __gnu_debug::_Safe_local_iterator_base Class Reference

Inheritance diagram for __gnu_debug::_Safe_local_iterator_base:



Public Member Functions

- void [_M_attach](#) ([_Safe_sequence_base](#) * __seq, bool __constant)
- void [_M_attach_single](#) ([_Safe_sequence_base](#) * __seq, bool __constant) throw ()
- bool [_M_attached_to](#) (const [_Safe_sequence_base](#) * __seq) const
- bool [_M_can_compare](#) (const [_Safe_iterator_base](#) & __x) const throw ()
- void [_M_detach](#) ()
- void [_M_detach_single](#) () throw ()
- void [_M_invalidate](#) ()
- void [_M_reset](#) () throw ()
- bool [_M_singular](#) () const throw ()
- void [_M_unlink](#) () throw ()

Public Attributes

- [_Safe_iterator_base](#) * [_M_next](#)
- [_Safe_iterator_base](#) * [_M_prior](#)
- [_Safe_sequence_base](#) * [_M_sequence](#)
- unsigned int [_M_version](#)

Protected Member Functions

- `_Safe_local_iterator_base()`
- `_Safe_local_iterator_base(const _Safe_sequence_base * __seq, bool __constant)`
- `_Safe_local_iterator_base(const _Safe_local_iterator_base & __x, bool __constant)`
- `_Safe_local_iterator_base(const _Safe_local_iterator_base &)`
- `_Safe_unordered_container_base * _M_get_container() const noexcept`
- `__gnu_cxx::__mutex & _M_get_mutex() throw()`
- `_Safe_local_iterator_base & operator= (const _Safe_local_iterator_base &)`

4.67.1 Detailed Description

Basic functionality for a *safe* iterator.

The `_Safe_local_iterator_base` base class implements the functionality of a safe local iterator that is not specific to a particular iterator type. It contains a pointer back to the container it references along with iterator version information and pointers to form a doubly-linked list of local iterators referenced by the container.

This class must not perform any operations that can throw an exception, or the exception guarantees of derived iterators will be broken.

Definition at line 50 of file `safe_unordered_base.h`.

4.67.2 Constructor & Destructor Documentation

4.67.2.1 `__gnu_debug::_Safe_local_iterator_base::_Safe_local_iterator_base() [inline], [protected]`

Initializes the iterator and makes it singular.

Definition at line 54 of file `safe_unordered_base.h`.

Referenced by `_Safe_local_iterator_base()`.

4.67.2.2 `__gnu_debug::_Safe_local_iterator_base::_Safe_local_iterator_base(const _Safe_sequence_base * __seq, bool __constant) [inline], [protected]`

Initialize the iterator to reference the container pointed to by `__seq`. `__constant` is true when we are initializing a constant local iterator, and false if it is a mutable local iterator. Note that `__seq` may be NULL, in which case the iterator will be singular. Otherwise, the iterator will reference `__seq` and be nonsingular.

Definition at line 64 of file `safe_unordered_base.h`.

References `_M_attach()`.

4.67.2.3 `__gnu_debug::Safe_local_iterator_base::Safe_local_iterator_base (const _Safe_local_iterator_base & __x, bool __constant) [inline], [protected]`

Initializes the iterator to reference the same container that `__x` does. `__constant` is true if this is a constant iterator, and false if it is mutable.

Definition at line 70 of file `safe_unordered_base.h`.

References `_M_attach()`, `_M_attach_single()`, `_M_detach()`, `_M_detach_single()`, `__gnu_debug::Safe_iterator_base::_M_sequence`, and `_Safe_local_iterator_base()`.

4.67.3 Member Function Documentation

4.67.3.1 `void __gnu_debug::Safe_local_iterator_base::_M_attach (_Safe_sequence_base * __seq, bool __constant)`

Attaches this iterator to the given container, detaching it from whatever container it was attached to originally. If the new container is the NULL pointer, the iterator is left unattached.

Referenced by `_Safe_local_iterator_base()`.

4.67.3.2 `void __gnu_debug::Safe_local_iterator_base::_M_attach_single (_Safe_sequence_base * __seq, bool __constant) throw`

Likewise, but not thread-safe.

Referenced by `_Safe_local_iterator_base()`.

4.67.3.3 `bool __gnu_debug::Safe_iterator_base::_M_attached_to (const _Safe_sequence_base * __seq) const [inline], [inherited]`

Determines if we are attached to the given sequence.

Definition at line 129 of file `safe_base.h`.

References `__gnu_debug::Safe_iterator_base::_M_can_compare()`, and `__gnu_debug::Safe_iterator_base::_M_is_singular()`.

4.67.3.4 `bool __gnu_debug::Safe_iterator_base::_M_can_compare (const _Safe_iterator_base & __x) const throw [inherited]`

Can we compare this iterator to the given iterator `__x`? Returns true if both iterators are nonsingular and reference the same sequence.

Referenced by `__gnu_debug::Safe_iterator_base::_M_attached_to()`, and `__gnu_debug::Safe_iterator<_Iterator, _Sequence>::_M_is_beginnest()`.

4.67.3.5 `void __gnu_debug::Safe_local_iterator_base::_M_detach ()`

Detach the iterator for whatever container it is attached to, if any.

Referenced by `_Safe_local_iterator_base()`.

4.67.3.6 `void __gnu_debug::_Safe_local_iterator_base::M_detach_single () throw`

Likewise, but not thread-safe.

Referenced by `_Safe_local_iterator_base()`.

4.67.3.7 `__gnu_cxx::mutex& __gnu_debug::_Safe_iterator_base::M_get_mutex () throw` `[protected]`,
`[inherited]`

For use in `_Safe_iterator`.

Referenced by `__gnu_debug::_Safe_iterator_base::_Safe_iterator_base()`, and `__gnu_debug::_Safe_sequence_base::~~_Safe_sequence_base()`.

4.67.3.8 `void __gnu_debug::_Safe_iterator_base::M_invalidate ()` `[inline]`, `[inherited]`

Invalidate the iterator, making it singular.

Definition at line 142 of file `safe_base.h`.

References `__gnu_debug::_Safe_iterator_base::M_reset()`.

4.67.3.9 `void __gnu_debug::_Safe_iterator_base::M_reset () throw` `[inherited]`

Reset all member variables

Referenced by `__gnu_debug::_Safe_iterator_base::M_invalidate()`.

4.67.3.10 `bool __gnu_debug::_Safe_iterator_base::M_singular () const throw` `[inherited]`

Is this iterator singular?

Referenced by `__gnu_debug::_check_singular_aux()`, `__gnu_debug::_Safe_iterator_base::M_attached_to()`, `__gnu_debug::_Safe_local_iterator<_Iterator, _Sequence>::_M_dereferenceable()`, `__gnu_debug::_Safe_local_iterator<_Iterator, _Sequence>::_M_incrementable()`, `__gnu_debug::_Safe_iterator<_Iterator, _Sequence>::_M_is_beginnest()`, and `__gnu_debug::_Safe_local_iterator<_Iterator, _Sequence>::_Safe_local_iterator()`.

4.67.3.11 `void __gnu_debug::_Safe_iterator_base::M_unlink () throw` `[inline]`, `[inherited]`

Unlink itself

Definition at line 151 of file `safe_base.h`.

References `__gnu_debug::_Safe_iterator_base::M_next`, and `__gnu_debug::_Safe_iterator_base::M_prior`.

4.67.4 Member Data Documentation

4.67.4.1 __Safe_iterator_base* __gnu_debug::_Safe_iterator_base::_M_next [inherited]

Pointer to the next iterator in the sequence's list of iterators. Only valid when _M_sequence != NULL.

Definition at line 72 of file safe_base.h.

Referenced by __gnu_debug::_Safe_iterator_base::_M_unlink().

4.67.4.2 __Safe_iterator_base* __gnu_debug::_Safe_iterator_base::_M_prior [inherited]

Pointer to the previous iterator in the sequence's list of iterators. Only valid when _M_sequence != NULL.

Definition at line 68 of file safe_base.h.

Referenced by __gnu_debug::_Safe_iterator_base::_M_unlink().

4.67.4.3 __Safe_sequence_base* __gnu_debug::_Safe_iterator_base::_M_sequence [inherited]

The sequence this iterator references; may be NULL to indicate a singular iterator.

Definition at line 55 of file safe_base.h.

Referenced by __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::_M_incrementable(), __gnu_debug::_Safe_iterator_base::_Safe_iterator_base(), and _Safe_local_iterator_base().

4.67.4.4 unsigned int __gnu_debug::_Safe_iterator_base::_M_version [inherited]

The version number of this iterator. The sentinel value 0 is used to indicate an invalidated iterator (i.e., one that is singular because of an operation on the container). This version number must equal the version number in the sequence referenced by _M_sequence for the iterator to be non-singular.

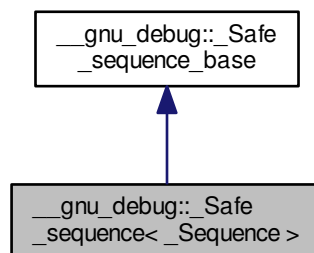
Definition at line 64 of file safe_base.h.

The documentation for this class was generated from the following file:

- [safe_unordered_base.h](#)

4.68 __gnu_debug::_Safe_sequence< _Sequence > Class Template Reference

Inheritance diagram for __gnu_debug::_Safe_sequence< _Sequence >:



Public Member Functions

- void `_M_attach` (`_Safe_iterator_base *``__it`, bool `__constant`)
- void `_M_attach_single` (`_Safe_iterator_base *``__it`, bool `__constant`) throw ()
- void `_M_detach` (`_Safe_iterator_base *``__it`)
- void `_M_detach_single` (`_Safe_iterator_base *``__it`) throw ()
- void `_M_invalidate_all` () const
- template<typename `_Predicate` >
void `_M_invalidate_if` (`_Predicate` `__pred`)
- template<typename `_Predicate` >
void `_M_transfer_from_if` (`_Safe_sequence` &`__from`, `_Predicate` `__pred`)

Public Attributes

- `_Safe_iterator_base *``_M_const_iterators`
- `_Safe_iterator_base *``_M_iterators`
- unsigned int `_M_version`

Protected Member Functions

- void `_M_detach_all` ()
- void `_M_detach_singular` ()
- `__gnu_cxx::__mutex` & `_M_get_mutex` () throw ()
- void `_M_revalidate_singular` ()
- void `_M_swap` (`_Safe_sequence_base` &`__x`)

4.68.1 Detailed Description

```
template<typename _Sequence>
class __gnu_debug::_Safe_sequence<_Sequence >
```

Base class for constructing a *safe* sequence type that tracks iterators that reference it.

The class template `_Safe_sequence` simplifies the construction of *safe* sequences that track the iterators that reference the sequence, so that the iterators are notified of changes in the sequence that may affect their operation, e.g., if the container invalidates its iterators or is destructed. This class template may only be used by deriving from it and passing the name of the derived class as its template parameter via the curiously recurring template pattern. The derived class must have `iterator` and `const_iterator` types that are instantiations of class template `_Safe_iterator` for this sequence. Iterators will then be tracked automatically.

Definition at line 52 of file `formatter.h`.

4.68.2 Member Function Documentation

4.68.2.1 void `__gnu_debug::_Safe_sequence_base::_M_attach` (`_Safe_iterator_base *``__it`, bool `__constant`)
[*inherited*]

Attach an iterator to this sequence.

4.68.2.2 void __gnu_debug::_Safe_sequence_base::_M_attach_single (_Safe_iterator_base * __it, bool __constant) throw)
[inherited]

Likewise but not thread safe.

4.68.2.3 void __gnu_debug::_Safe_sequence_base::_M_detach (_Safe_iterator_base * __it) [inherited]

Detach an iterator from this sequence

4.68.2.4 void __gnu_debug::_Safe_sequence_base::_M_detach_all () [protected],[inherited]

Detach all iterators, leaving them singular.

4.68.2.5 void __gnu_debug::_Safe_sequence_base::_M_detach_single (_Safe_iterator_base * __it) throw)
[inherited]

Likewise but not thread safe.

4.68.2.6 void __gnu_debug::_Safe_sequence_base::_M_detach_singular () [protected],[inherited]

Detach all singular iterators.

Postcondition

for all iterators i attached to this sequence, i->_M_version == _M_version.

4.68.2.7 __gnu_cxx::mutex& __gnu_debug::_Safe_sequence_base::_M_get_mutex () throw) [protected],
[inherited]

For use in _Safe_sequence.

4.68.2.8 void __gnu_debug::_Safe_sequence_base::_M_invalidate_all () const [inline],[inherited]

Invalidates all iterators.

Definition at line 242 of file safe_base.h.

References __gnu_debug::_Safe_iterator_base::_M_attach(), __gnu_debug::_Safe_iterator_base::_M_attach_single(), __gnu_debug::_Safe_iterator_base::_M_detach(), and __gnu_debug::_Safe_iterator_base::_M_detach_single().

4.68.2.9 template<typename _Sequence> template<typename _Predicate> void __gnu_debug::_Safe_sequence<_Sequence>::_M_invalidate_if (_Predicate __pred)

Invalidates all iterators x that reference this sequence, are not singular, and for which __pred(x) returns true. __pred will be invoked with the normal iterators nested in the safe ones.

4.68.2.10 `void __gnu_debug::Safe_sequence_base::_M_revalidate_singular ()` [protected],[inherited]

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

4.68.2.11 `void __gnu_debug::Safe_sequence_base::_M_swap (_Safe_sequence_base & __x)` [protected],[inherited]

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

4.68.2.12 `template<typename _Sequence> template<typename _Predicate > void __gnu_debug::Safe_sequence<_Sequence>::_M_transfer_from_if (_Safe_sequence<_Sequence> & __from, _Predicate __pred)`

Transfers all iterators `x` that reference `from` sequence, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

4.68.3 Member Data Documentation

4.68.3.1 `_Safe_iterator_base* __gnu_debug::Safe_sequence_base::_M_const_iterators` [inherited]

The list of constant iterators that reference this container.

Definition at line 184 of file `safe_base.h`.

4.68.3.2 `_Safe_iterator_base* __gnu_debug::Safe_sequence_base::_M_iterators` [inherited]

The list of mutable iterators that reference this container.

Definition at line 181 of file `safe_base.h`.

4.68.3.3 `unsigned int __gnu_debug::Safe_sequence_base::_M_version` [mutable],[inherited]

The container version number. This number may never be 0.

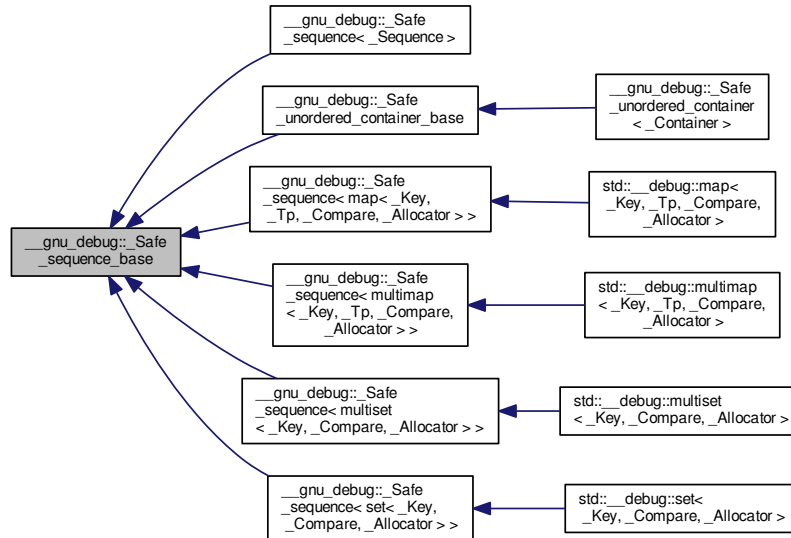
Definition at line 187 of file `safe_base.h`.

The documentation for this class was generated from the following files:

- [formatter.h](#)
- [safe_sequence.h](#)

4.69 __gnu_debug::__Safe_sequence_base Class Reference

Inheritance diagram for __gnu_debug::__Safe_sequence_base:



Public Member Functions

- void [_M_attach](#) ([_Safe_iterator_base](#) * __it, bool __constant)
- void [_M_attach_single](#) ([_Safe_iterator_base](#) * __it, bool __constant) throw ()
- void [_M_detach](#) ([_Safe_iterator_base](#) * __it)
- void [_M_detach_single](#) ([_Safe_iterator_base](#) * __it) throw ()
- void [_M_invalidate_all](#) () const

Public Attributes

- [_Safe_iterator_base](#) * [_M_const_iterators](#)
- [_Safe_iterator_base](#) * [_M_iterators](#)
- unsigned int [_M_version](#)

Protected Member Functions

- [_Safe_sequence_base](#) (const [_Safe_sequence_base](#) &) noexcept
- [_Safe_sequence_base](#) ([_Safe_sequence_base](#) && __x) noexcept
- [~_Safe_sequence_base](#) ()
- void [_M_detach_all](#) ()
- void [_M_detach_singular](#) ()
- [__gnu_cxx::__mutex](#) & [_M_get_mutex](#) () throw ()
- void [_M_revalidate_singular](#) ()
- void [_M_swap](#) ([_Safe_sequence_base](#) & __x)

4.69.1 Detailed Description

Base class that supports tracking of iterators that reference a sequence.

The `_Safe_sequence_base` class provides basic support for tracking iterators into a sequence. Sequences that track iterators must derived from `_Safe_sequence_base` publicly, so that safe iterators (which inherit `_Safe_iterator_base`) can attach to them. This class contains two linked lists of iterators, one for constant iterators and one for mutable iterators, and a version number that allows very fast invalidation of all iterators that reference the container.

This class must ensure that no operation on it may throw an exception, otherwise *safe* sequences may fail to provide the exception-safety guarantees required by the C++ standard.

Definition at line 177 of file `safe_base.h`.

4.69.2 Constructor & Destructor Documentation

4.69.2.1 `__gnu_debug::_Safe_sequence_base::~~_Safe_sequence_base () [inline], [protected]`

Notify all iterators that reference this sequence that the sequence is being destroyed.

Definition at line 206 of file `safe_base.h`.

References `__gnu_debug::_Safe_iterator_base::_M_get_mutex()`.

4.69.3 Member Function Documentation

4.69.3.1 `void __gnu_debug::_Safe_sequence_base::_M_attach (_Safe_iterator_base * __it, bool __constant)`

Attach an iterator to this sequence.

4.69.3.2 `void __gnu_debug::_Safe_sequence_base::_M_attach_single (_Safe_iterator_base * __it, bool __constant) throw)`

Likewise but not thread safe.

4.69.3.3 `void __gnu_debug::_Safe_sequence_base::_M_detach (_Safe_iterator_base * __it)`

Detach an iterator from this sequence

4.69.3.4 `void __gnu_debug::_Safe_sequence_base::_M_detach_all () [protected]`

Detach all iterators, leaving them singular.

4.69.3.5 `void __gnu_debug::_Safe_sequence_base::_M_detach_single (_Safe_iterator_base * __it) throw)`

Likewise but not thread safe.

4.69.3.6 `void __gnu_debug::Safe_sequence_base::M_detach_singular ()` [protected]

Detach all singular iterators.

Postcondition

for all iterators *i* attached to this sequence, *i*->_M_version == _M_version.

4.69.3.7 `__gnu_cxx::mutex& __gnu_debug::Safe_sequence_base::M_get_mutex () throw` [protected]

For use in `_Safe_sequence`.

4.69.3.8 `void __gnu_debug::Safe_sequence_base::M_invalidate_all () const` [inline]

Invalidates all iterators.

Definition at line 242 of file `safe_base.h`.

References `__gnu_debug::Safe_iterator_base::M_attach()`, `__gnu_debug::Safe_iterator_base::M_attach_single()`, `__gnu_debug::Safe_iterator_base::M_detach()`, and `__gnu_debug::Safe_iterator_base::M_detach_single()`.

4.69.3.9 `void __gnu_debug::Safe_sequence_base::M_revalidate_singular ()` [protected]

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

4.69.3.10 `void __gnu_debug::Safe_sequence_base::M_swap (_Safe_sequence_base & __x)` [protected]

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

4.69.4 Member Data Documentation

4.69.4.1 `_Safe_iterator_base* __gnu_debug::Safe_sequence_base::M_const_iterators`

The list of constant iterators that reference this container.

Definition at line 184 of file `safe_base.h`.

4.69.4.2 `_Safe_iterator_base* __gnu_debug::Safe_sequence_base::M_iterators`

The list of mutable iterators that reference this container.

Definition at line 181 of file `safe_base.h`.

4.69.4.3 unsigned int __gnu_debug::_Safe_sequence_base::_M_version [mutable]

The container version number. This number may never be 0.

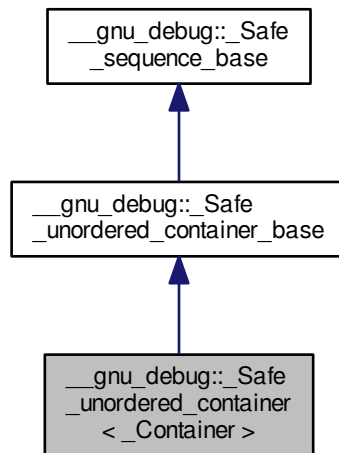
Definition at line 187 of file safe_base.h.

The documentation for this class was generated from the following file:

- [safe_base.h](#)

4.70 __gnu_debug::_Safe_unordered_container<_Container> Class Template Reference

Inheritance diagram for __gnu_debug::_Safe_unordered_container<_Container>:



Public Member Functions

- void [_M_attach](#) ([_Safe_iterator_base](#) * __it, bool __constant)
- void [_M_attach_local](#) ([_Safe_iterator_base](#) * __it, bool __constant)
- void [_M_attach_local_single](#) ([_Safe_iterator_base](#) * __it, bool __constant) throw ()
- void [_M_attach_single](#) ([_Safe_iterator_base](#) * __it, bool __constant) throw ()
- void [_M_detach](#) ([_Safe_iterator_base](#) * __it)
- void [_M_detach_local](#) ([_Safe_iterator_base](#) * __it)
- void [_M_detach_local_single](#) ([_Safe_iterator_base](#) * __it) throw ()
- void [_M_detach_single](#) ([_Safe_iterator_base](#) * __it) throw ()
- void [_M_invalidate_all](#) () const
- template<typename _Predicate>
void [_M_invalidate_if](#) (_Predicate __pred)
- template<typename _Predicate>
void [_M_invalidate_local_if](#) (_Predicate __pred)

Public Attributes

- `_Safe_iterator_base * _M_const_iterators`
- `_Safe_iterator_base * _M_const_local_iterators`
- `_Safe_iterator_base * _M_iterators`
- `_Safe_iterator_base * _M_local_iterators`
- `unsigned int _M_version`

Protected Member Functions

- `void _M_detach_all ()`
- `void _M_detach_singular ()`
- `__gnu_cxx::__mutex & _M_get_mutex () throw ()`
- `void _M_revalidate_singular ()`
- `void _M_swap (_Safe_unordered_container_base & __x)`
- `void _M_swap (_Safe_sequence_base & __x)`

4.70.1 Detailed Description

```
template<typename _Container>
class __gnu_debug::_Safe_unordered_container<_Container>
```

Base class for constructing a *safe* unordered container type that tracks iterators that reference it.

The class template `_Safe_unordered_container` simplifies the construction of *safe* unordered containers that track the iterators that reference the container, so that the iterators are notified of changes in the container that may affect their operation, e.g., if the container invalidates its iterators or is destructed. This class template may only be used by deriving from it and passing the name of the derived class as its template parameter via the curiously recurring template pattern. The derived class must have `iterator` and `const_iterator` types that are instantiations of class template `_Safe_iterator` for this container and `local_iterator` and `const_local_iterator` types that are instantiations of class template `_Safe_local_iterator` for this container. Iterators will then be tracked automatically.

Definition at line 58 of file `safe_unordered_container.h`.

4.70.2 Member Function Documentation

4.70.2.1 `void __gnu_debug::_Safe_sequence_base::_M_attach (_Safe_iterator_base * __it, bool __constant)`
[inherited]

Attach an iterator to this sequence.

4.70.2.2 `void __gnu_debug::_Safe_unordered_container_base::_M_attach_local (_Safe_iterator_base * __it, bool __constant)`
[inherited]

Attach an iterator to this container.

4.70.2.3 `void __gnu_debug::_Safe_unordered_container_base::_M_attach_local_single (_Safe_iterator_base * __it, bool __constant) throw` `[inherited]`

Likewise but not thread safe.

4.70.2.4 `void __gnu_debug::_Safe_sequence_base::_M_attach_single (_Safe_iterator_base * __it, bool __constant) throw` `[inherited]`

Likewise but not thread safe.

4.70.2.5 `void __gnu_debug::_Safe_sequence_base::_M_detach (_Safe_iterator_base * __it)` `[inherited]`

Detach an iterator from this sequence

4.70.2.6 `void __gnu_debug::_Safe_unordered_container_base::_M_detach_all ()` `[protected]`, `[inherited]`

Detach all iterators, leaving them singular.

4.70.2.7 `void __gnu_debug::_Safe_unordered_container_base::_M_detach_local (_Safe_iterator_base * __it)` `[inherited]`

Detach an iterator from this container

4.70.2.8 `void __gnu_debug::_Safe_unordered_container_base::_M_detach_local_single (_Safe_iterator_base * __it) throw` `[inherited]`

Likewise but not thread safe.

4.70.2.9 `void __gnu_debug::_Safe_sequence_base::_M_detach_single (_Safe_iterator_base * __it) throw` `[inherited]`

Likewise but not thread safe.

4.70.2.10 `void __gnu_debug::_Safe_sequence_base::_M_detach_singular ()` `[protected]`, `[inherited]`

Detach all singular iterators.

Postcondition

for all iterators *i* attached to this sequence, `i->_M_version == _M_version`.

4.70.2.11 `__gnu_cxx::__mutex& __gnu_debug::_Safe_sequence_base::_M_get_mutex () throw` `[protected]`, `[inherited]`

For use in `_Safe_sequence`.

4.70.2.12 `void __gnu_debug::Safe_sequence_base::M_invalidate_all() const` `[inline]`, `[inherited]`

Invalidates all iterators.

Definition at line 242 of file `safe_base.h`.

References `__gnu_debug::Safe_iterator_base::M_attach()`, `__gnu_debug::Safe_iterator_base::M_attach_single()`, `__gnu_debug::Safe_iterator_base::M_detach()`, and `__gnu_debug::Safe_iterator_base::M_detach_single()`.

4.70.2.13 `template<typename _Container> template<typename _Predicate> void __gnu_debug::Safe_unordered_container<_Container>::M_invalidate_if(_Predicate __pred)`

Invalidates all iterators `x` that reference this container, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

4.70.2.14 `template<typename _Container> template<typename _Predicate> void __gnu_debug::Safe_unordered_container<_Container>::M_invalidate_local_if(_Predicate __pred)`

Invalidates all local iterators `x` that reference this container, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal local iterators nested in the safe ones.

4.70.2.15 `void __gnu_debug::Safe_sequence_base::M_revalidate_singular()` `[protected]`, `[inherited]`

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

4.70.2.16 `void __gnu_debug::Safe_unordered_container_base::M_swap(_Safe_unordered_container_base & __x)` `[protected]`, `[inherited]`

Swap this container with the given container. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

4.70.2.17 `void __gnu_debug::Safe_sequence_base::M_swap(_Safe_sequence_base & __x)` `[protected]`, `[inherited]`

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

4.70.3 Member Data Documentation

4.70.3.1 `_Safe_iterator_base* __gnu_debug::Safe_sequence_base::M_const_iterators` `[inherited]`

The list of constant iterators that reference this container.

Definition at line 184 of file `safe_base.h`.

4.70.3.2 `_Safe_iterator_base* __gnu_debug::Safe_unordered_container_base::M_const_local_iterators` [inherited]

The list of constant local iterators that reference this container.

Definition at line 131 of file `safe_unordered_base.h`.

4.70.3.3 `_Safe_iterator_base* __gnu_debug::Safe_sequence_base::M_iterators` [inherited]

The list of mutable iterators that reference this container.

Definition at line 181 of file `safe_base.h`.

4.70.3.4 `_Safe_iterator_base* __gnu_debug::Safe_unordered_container_base::M_local_iterators` [inherited]

The list of mutable local iterators that reference this container.

Definition at line 128 of file `safe_unordered_base.h`.

4.70.3.5 `unsigned int __gnu_debug::Safe_sequence_base::M_version` [mutable], [inherited]

The container version number. This number may never be 0.

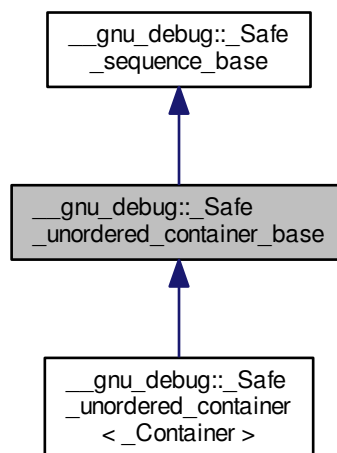
Definition at line 187 of file `safe_base.h`.

The documentation for this class was generated from the following file:

- [safe_unordered_container.h](#)

4.71 `__gnu_debug::Safe_unordered_container_base` Class Reference

Inheritance diagram for `__gnu_debug::Safe_unordered_container_base`:



Public Member Functions

- `void _M_attach (_Safe_iterator_base * __it, bool __constant)`
- `void _M_attach_local (_Safe_iterator_base * __it, bool __constant)`
- `void _M_attach_local_single (_Safe_iterator_base * __it, bool __constant) throw ()`
- `void _M_attach_single (_Safe_iterator_base * __it, bool __constant) throw ()`
- `void _M_detach (_Safe_iterator_base * __it)`
- `void _M_detach_local (_Safe_iterator_base * __it)`
- `void _M_detach_local_single (_Safe_iterator_base * __it) throw ()`
- `void _M_detach_single (_Safe_iterator_base * __it) throw ()`
- `void _M_invalidate_all () const`

Public Attributes

- `_Safe_iterator_base * _M_const_iterators`
- `_Safe_iterator_base * _M_const_local_iterators`
- `_Safe_iterator_base * _M_iterators`
- `_Safe_iterator_base * _M_local_iterators`
- `unsigned int _M_version`

Protected Member Functions

- `_Safe_unordered_container_base (const _Safe_unordered_container_base &) noexcept`
- `_Safe_unordered_container_base (_Safe_unordered_container_base && __x) noexcept`
- `~_Safe_unordered_container_base ()`
- `void _M_detach_all ()`
- `void _M_detach_singular ()`
- `__gnu_cxx::__mutex & _M_get_mutex () throw ()`
- `void _M_revalidate_singular ()`
- `void _M_swap (_Safe_unordered_container_base & __x)`
- `void _M_swap (_Safe_sequence_base & __x)`

4.71.1 Detailed Description

Base class that supports tracking of local iterators that reference an unordered container.

The `_Safe_unordered_container_base` class provides basic support for tracking iterators into an unordered container. Containers that track iterators must derived from `_Safe_unordered_container_base` publicly, so that safe iterators (which inherit `_Safe_iterator_base`) can attach to them. This class contains four linked lists of iterators, one for constant iterators, one for mutable iterators, one for constant local iterators, one for mutable local iterators and a version number that allows very fast invalidation of all iterators that reference the container.

This class must ensure that no operation on it may throw an exception, otherwise *safe* containers may fail to provide the exception-safety guarantees required by the C++ standard.

Definition at line 123 of file `safe_unordered_base.h`.

4.71.2 Constructor & Destructor Documentation

4.71.2.1 `__gnu_debug::_Safe_unordered_container_base::~~Safe_unordered_container_base () [inline], [protected]`

Notify all iterators that reference this container that the container is being destroyed.

Definition at line 151 of file `safe_unordered_base.h`.

4.71.3 Member Function Documentation

4.71.3.1 `void __gnu_debug::_Safe_sequence_base::_M_attach (_Safe_iterator_base * __it, bool __constant) [inherited]`

Attach an iterator to this sequence.

4.71.3.2 `void __gnu_debug::_Safe_unordered_container_base::_M_attach_local (_Safe_iterator_base * __it, bool __constant)`

Attach an iterator to this container.

4.71.3.3 `void __gnu_debug::_Safe_unordered_container_base::_M_attach_local_single (_Safe_iterator_base * __it, bool __constant) throw)`

Likewise but not thread safe.

4.71.3.4 `void __gnu_debug::_Safe_sequence_base::_M_attach_single (_Safe_iterator_base * __it, bool __constant) throw) [inherited]`

Likewise but not thread safe.

4.71.3.5 `void __gnu_debug::_Safe_sequence_base::_M_detach (_Safe_iterator_base * __it) [inherited]`

Detach an iterator from this sequence

4.71.3.6 `void __gnu_debug::_Safe_unordered_container_base::_M_detach_all () [protected]`

Detach all iterators, leaving them singular.

4.71.3.7 `void __gnu_debug::_Safe_unordered_container_base::_M_detach_local (_Safe_iterator_base * __it)`

Detach an iterator from this container

4.71.3.8 `void __gnu_debug::_Safe_unordered_container_base::_M_detach_local_single (_Safe_iterator_base * __it) throw)`

Likewise but not thread safe.

4.71.3.9 `void __gnu_debug::Safe_sequence_base::M_detach_single (_Safe_iterator_base * __it) throw (`
`[inherited]`

Likewise but not thread safe.

4.71.3.10 `void __gnu_debug::Safe_sequence_base::M_detach_singular () [protected],[inherited]`

Detach all singular iterators.

Postcondition

for all iterators *i* attached to this sequence, *i*->_M_version == _M_version.

4.71.3.11 `__gnu_cxx::mutex& __gnu_debug::Safe_sequence_base::M_get_mutex () throw (` `[protected],`
`[inherited]`

For use in `_Safe_sequence`.

4.71.3.12 `void __gnu_debug::Safe_sequence_base::M_invalidate_all () const [inline],[inherited]`

Invalidates all iterators.

Definition at line 242 of file `safe_base.h`.

References `__gnu_debug::Safe_iterator_base::M_attach()`, `__gnu_debug::Safe_iterator_base::M_attach_single()`, `__gnu_debug::Safe_iterator_base::M_detach()`, and `__gnu_debug::Safe_iterator_base::M_detach_single()`.

4.71.3.13 `void __gnu_debug::Safe_sequence_base::M_revalidate_singular () [protected],[inherited]`

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

4.71.3.14 `void __gnu_debug::Safe_unordered_container_base::M_swap (_Safe_unordered_container_base & __x)`
`[protected]`

Swap this container with the given container. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

4.71.3.15 `void __gnu_debug::Safe_sequence_base::M_swap (_Safe_sequence_base & __x) [protected],`
`[inherited]`

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

4.71.4 Member Data Documentation

4.71.4.1 `_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators` [inherited]

The list of constant iterators that reference this container.

Definition at line 184 of file `safe_base.h`.

4.71.4.2 `_Safe_iterator_base* __gnu_debug::_Safe_unordered_container_base::_M_const_local_iterators`

The list of constant local iterators that reference this container.

Definition at line 131 of file `safe_unordered_base.h`.

4.71.4.3 `_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators` [inherited]

The list of mutable iterators that reference this container.

Definition at line 181 of file `safe_base.h`.

4.71.4.4 `_Safe_iterator_base* __gnu_debug::_Safe_unordered_container_base::_M_local_iterators`

The list of mutable local iterators that reference this container.

Definition at line 128 of file `safe_unordered_base.h`.

4.71.4.5 `unsigned int __gnu_debug::_Safe_sequence_base::_M_version` [mutable], [inherited]

The container version number. This number may never be 0.

Definition at line 187 of file `safe_base.h`.

The documentation for this class was generated from the following file:

- [safe_unordered_base.h](#)

4.72 `__gnu_parallel::__accumulate_binop_reduct<_BinOp>` Struct Template Reference

Public Member Functions

- `__accumulate_binop_reduct` (`_BinOp` & `b`)
- `template<typename _Result, typename _Addend>`
`_Result operator()` (`const _Result` & `__x`, `const _Addend` & `__y`)

Public Attributes

- `_BinOp` & `__binop`

4.72.1 Detailed Description

```
template<typename _BinOp>
struct __gnu_parallel::__accumulate_binop_reduct<_BinOp>
```

General reduction, using a binary operator.

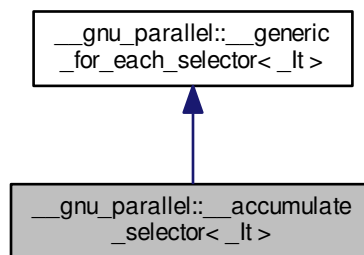
Definition at line 335 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for_each_selectors.h](#)

4.73 `__gnu_parallel::__accumulate_selector<_It>` Struct Template Reference

Inheritance diagram for `__gnu_parallel::__accumulate_selector<_It>`:



Public Member Functions

- `template<typename _Op>`
`std::iterator_traits<_It>::value_type operator() (_Op __o, _It __i)`

Public Attributes

- `_It` [_M_finish_iterator](#)

4.73.1 Detailed Description

```
template<typename _It>
struct __gnu_parallel::__accumulate_selector<_It>
```

`std::accumulate()` selector.

Definition at line 208 of file `for_each_selectors.h`.

4.73.2 Member Function Documentation

4.73.2.1 `template<typename _It > template<typename _Op > std::iterator_traits<_It>::value_type
__gnu_parallel::__accumulate_selector<_It>::operator()(_Op __o, _It __i) [inline]`

Functor execution.

Parameters

<code>__o</code>	Operator (unused).
<code>__i</code>	iterator referencing object.

Returns

The current value.

Definition at line 216 of file `for_each_selectors.h`.

4.73.3 Member Data Documentation

4.73.3.1 `template<typename _It > _It __gnu_parallel::__generic_for_each_selector<_It>::M_finish_iterator
[inherited]`

Iterator on last element processed; needed for some algorithms (e. g. `std::transform()`).

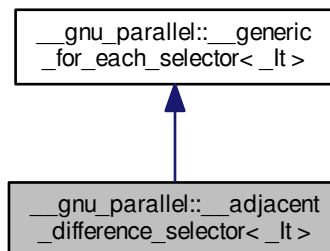
Definition at line 47 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for_each_selectors.h](#)

4.74 `__gnu_parallel::__adjacent_difference_selector<_It>` Struct Template Reference

Inheritance diagram for `__gnu_parallel::__adjacent_difference_selector<_It>`:



Public Member Functions

- `template<typename _Op >`
`bool operator() (_Op &__o, _It __i)`

Public Attributes

- `_It _M_finish_iterator`

4.74.1 Detailed Description

```
template<typename _It>
struct __gnu_parallel::__adjacent_difference_selector< _It >
```

Selector that returns the difference between two adjacent `__elements`.

Definition at line 269 of file `for_each_selectors.h`.

4.74.2 Member Data Documentation

4.74.2.1 `template<typename _It > _It __gnu_parallel::__generic_for_each_selector< _It >::_M_finish_iterator`
[*inherited*]

`_Iterator` on last element processed; needed for some algorithms (e. g. `std::transform()`).

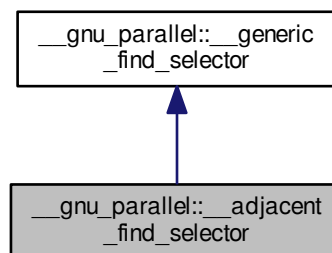
Definition at line 47 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- `for_each_selectors.h`

4.75 `__gnu_parallel::__adjacent_find_selector` Struct Reference

Inheritance diagram for `__gnu_parallel::__adjacent_find_selector`:



Public Member Functions

- `template<typename _RAIter1, typename _RAIter2, typename _Pred >`
`std::pair< _RAIter1, _RAIter2 > _M_sequential_algorithm (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred)`
- `template<typename _RAIter1, typename _RAIter2, typename _Pred >`
`bool operator() (_RAIter1 __i1, _RAIter2 __i2, _Pred __pred)`

4.75.1 Detailed Description

Test predicate on two adjacent elements.

Definition at line 80 of file `find_selectors.h`.

4.75.2 Member Function Documentation

4.75.2.1 `template<typename _RAIter1, typename _RAIter2, typename _Pred > std::pair<_RAIter1, _RAIter2>`
`__gnu_parallel::__adjacent_find_selector::__M_sequential_algorithm (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2`
`__begin2, _Pred __pred) [inline]`

Corresponding sequential algorithm on a sequence.

Parameters

<code>__begin1</code>	Begin iterator of first sequence.
<code>__end1</code>	End iterator of first sequence.
<code>__begin2</code>	Begin iterator of second sequence.
<code>__pred</code>	Find predicate.

Definition at line 105 of file `find_selectors.h`.

References `std::make_pair()`.

4.75.2.2 `template<typename _RAIter1, typename _RAIter2, typename _Pred > bool __gnu_parallel::__adjacent_find_selector`
`::operator() (_RAIter1 __i1, _RAIter2 __i2, _Pred __pred) [inline]`

Test on one position.

Parameters

<code>__i1</code>	Iterator on first sequence.
<code>__i2</code>	Iterator on second sequence (unused).
<code>__pred</code>	Find predicate.

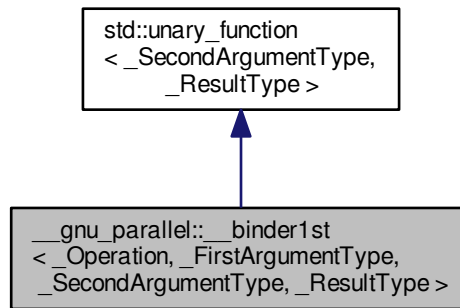
Definition at line 90 of file `find_selectors.h`.

The documentation for this struct was generated from the following file:

- [find_selectors.h](#)

4.76 `__gnu_parallel::__binder1st< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >` Class Template Reference

Inheritance diagram for `__gnu_parallel::__binder1st< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >`:



Public Types

- typedef `_SecondArgumentType` [argument_type](#)
- typedef `_ResultType` [result_type](#)

Public Member Functions

- **`__binder1st`** (`const _Operation &__x, const _FirstArgumentType &__y`)
- `_ResultType` **`operator()`** (`const _SecondArgumentType &__x`)
- `_ResultType` **`operator()`** (`_SecondArgumentType &__x`) `const`

Protected Attributes

- `_Operation` **`_M_op`**
- `_FirstArgumentType` **`_M_value`**

4.76.1 Detailed Description

```
template<typename _Operation, typename _FirstArgumentType, typename _SecondArgumentType, typename _ResultType>
class __gnu_parallel::__binder1st< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >
```

Similar to `std::binder1st`, but giving the argument types explicitly.

Definition at line 192 of file `parallel/base.h`.

4.76.2 Member Typedef Documentation

4.76.2.1 `typedef _SecondArgumentType std::unary_function< _SecondArgumentType, _ResultType >::argument_type` [inherited]

`argument_type` is the type of the argument

Definition at line 108 of file `stl_function.h`.

4.76.2.2 `typedef _ResultType std::unary_function< _SecondArgumentType, _ResultType >::result_type` [inherited]

`result_type` is the return type

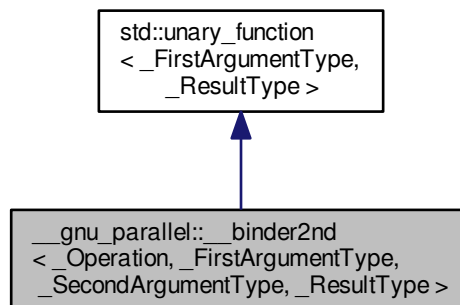
Definition at line 111 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [parallel/base.h](#)

4.77 `__gnu_parallel::__binder2nd< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >` Class Template Reference

Inheritance diagram for `__gnu_parallel::__binder2nd< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >`:



Public Types

- typedef `_FirstArgumentType` [argument_type](#)
- typedef `_ResultType` [result_type](#)

Public Member Functions

- `__binder2nd` (`const _Operation &__x`, `const _SecondArgumentType &__y`)
- `_ResultType operator()` (`const _FirstArgumentType &__x`) `const`
- `_ResultType operator()` (`_FirstArgumentType &__x`)

Protected Attributes

- `_Operation` `_M_op`
- `_SecondArgumentType` `_M_value`

4.77.1 Detailed Description

```
template<typename _Operation, typename _FirstArgumentType, typename _SecondArgumentType, typename _ResultType>
class __gnu_parallel::__binder2nd<_Operation, _FirstArgumentType, _SecondArgumentType, _ResultType>
```

Similar to `std::binder2nd`, but giving the argument types explicitly.

Definition at line 220 of file `parallel/base.h`.

4.77.2 Member Typedef Documentation

4.77.2.1 `typedef _FirstArgumentType std::unary_function<_FirstArgumentType, _ResultType>::argument_type`
 [inherited]

`argument_type` is the type of the argument

Definition at line 108 of file `stl_function.h`.

4.77.2.2 `typedef _ResultType std::unary_function<_FirstArgumentType, _ResultType>::result_type` [inherited]

`result_type` is the return type

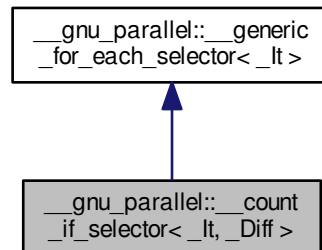
Definition at line 111 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [parallel/base.h](#)

4.78 `__gnu_parallel::__count_if_selector<_It, _Diff>` Struct Template Reference

Inheritance diagram for `__gnu_parallel::__count_if_selector<_It, _Diff>`:



Public Member Functions

- `template<typename _Op> _Diff operator\(\) (_Op &__o, _It __i)`

Public Attributes

- `_It _M_finish_iterator`

4.78.1 Detailed Description

```
template<typename _It, typename _Diff>
struct __gnu_parallel::__count_if_selector<_It, _Diff>
```

`std::count_if()` selector.

Definition at line 194 of file `for_each_selectors.h`.

4.78.2 Member Function Documentation

4.78.2.1 `template<typename _It, typename _Diff> template<typename _Op> _Diff __gnu_parallel::__count_if_selector<_It, _Diff>::operator() (_Op &__o, _It __i) [inline]`

Functor execution.

Parameters

<code>_↔ _o</code>	Operator.
<code>_↔ _i</code>	iterator referencing object.

Returns

1 if count, 0 if does not count.

Definition at line 202 of file `for_each_selectors.h`.

4.78.3 Member Data Documentation

4.78.3.1 `template<typename _It> _It __gnu_parallel::__generic_for_each_selector<_It>::_M_finish_iterator` [inherited]

`_Iterator` on last element processed; needed for some algorithms (e. g. `std::transform()`).

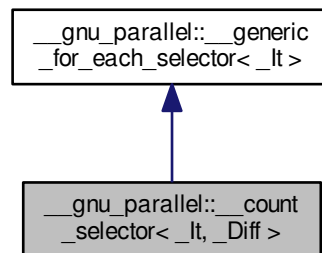
Definition at line 47 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for_each_selectors.h](#)

4.79 `__gnu_parallel::__count_selector<_It, _Diff>` Struct Template Reference

Inheritance diagram for `__gnu_parallel::__count_selector<_It, _Diff>`:



Public Member Functions

- `template<typename _ValueType >
_Diff operator() (_ValueType &__v, _It __i)`

Public Attributes

- `_It _M_finish_iterator`

4.79.1 Detailed Description

```
template<typename _It, typename _Diff>
struct __gnu_parallel::__count_selector< _It, _Diff >
```

`std::count()` selector.

Definition at line 180 of file `for_each_selectors.h`.

4.79.2 Member Function Documentation

4.79.2.1 `template<typename _It, typename _Diff> template<typename _ValueType > _Diff __gnu_parallel::__count_↔
selector< _It, _Diff >::operator() (_ValueType &__v, _It __i) [inline]`

Functor execution.

Parameters

<code>_↔ __v</code>	Current value.
<code>_↔ __i</code>	iterator referencing object.

Returns

1 if count, 0 if does not count.

Definition at line 188 of file `for_each_selectors.h`.

4.79.3 Member Data Documentation

4.79.3.1 `template<typename _It > _It __gnu_parallel::__generic_for_each_selector< _It >::M_finish_iterator
[inherited]`

`_Iterator` on last element processed; needed for some algorithms (e. g. `std::transform()`).

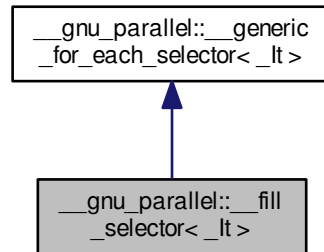
Definition at line 47 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for_each_selectors.h](#)

4.80 `__gnu_parallel::__fill_selector<_It>` Struct Template Reference

Inheritance diagram for `__gnu_parallel::__fill_selector<_It>`:



Public Member Functions

- `template<typename _ValueType > bool operator() (_ValueType &__v, _It __i)`

Public Attributes

- `_It _M_finish_iterator`

4.80.1 Detailed Description

```
template<typename _It>
struct __gnu_parallel::__fill_selector<_It>
```

`std::fill()` selector.

Definition at line 84 of file `for_each_selectors.h`.

4.80.2 Member Function Documentation

4.80.2.1 `template<typename _It> template<typename _ValueType> bool __gnu_parallel::__fill_selector<_It>::operator() (_ValueType &__v, _It __i) [inline]`

Functor execution.

Parameters

<code>_↔ _v</code>	Current value.
<code>_↔ _i</code>	iterator referencing object.

Definition at line 91 of file `for_each_selectors.h`.

4.80.3 Member Data Documentation

4.80.3.1 `template<typename _It> _It __gnu_parallel::__generic_for_each_selector<_It>::__M_finish_iterator` [inherited]

`_Iterator` on last element processed; needed for some algorithms (e. g. `std::transform()`).

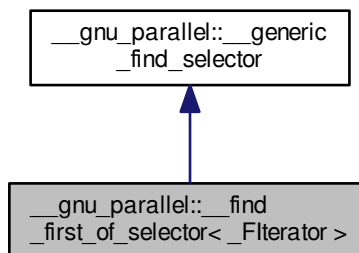
Definition at line 47 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for_each_selectors.h](#)

4.81 `__gnu_parallel::__find_first_of_selector<_FIterator>` Struct Template Reference

Inheritance diagram for `__gnu_parallel::__find_first_of_selector<_FIterator>`:



Public Member Functions

- `__find_first_of_selector` (`_FIterator __begin`, `_FIterator __end`)
- `template<typename _RAIter1, typename _RAIter2, typename _Pred>`
`std::pair<_RAIter1, _RAIter2> _M_sequential_algorithm (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __↔
begin2, _Pred __pred)`
- `template<typename _RAIter1, typename _RAIter2, typename _Pred>`
`bool operator\(\) (_RAIter1 __i1, _RAIter2 __i2, _Pred __pred)`

Public Attributes

- `_FIterator _M_begin`
- `_FIterator _M_end`

4.81.1 Detailed Description

```
template<typename _FIterator>
struct __gnu_parallel::__find_first_of_selector<_FIterator>
```

Test predicate on several elements.

Definition at line 153 of file `find_selectors.h`.

4.81.2 Member Function Documentation

```
4.81.2.1 template<typename _FIterator> template<typename _RAlter1, typename _RAlter2, typename _Pred>
std::pair<_RAlter1, _RAlter2> __gnu_parallel::__find_first_of_selector<_FIterator>::_M_sequential_algorithm
( _RAlter1 __begin1, _RAlter1 __end1, _RAlter2 __begin2, _Pred __pred ) [inline]
```

Corresponding sequential algorithm on a sequence.

Parameters

<code>__begin1</code>	Begin iterator of first sequence.
<code>__end1</code>	End iterator of first sequence.
<code>__begin2</code>	Begin iterator of second sequence.
<code>__pred</code>	Find predicate.

Definition at line 186 of file `find_selectors.h`.

References `std::make_pair()`.

```
4.81.2.2 template<typename _FIterator> template<typename _RAlter1, typename _RAlter2, typename _Pred> bool
__gnu_parallel::__find_first_of_selector<_FIterator>::operator()( _RAlter1 __i1, _RAlter2 __i2, _Pred __pred )
[inline]
```

Test on one position.

Parameters

<code>__i1</code>	_Iterator on first sequence.
<code>__i2</code>	_Iterator on second sequence (unused).
<code>__pred</code>	Find predicate.

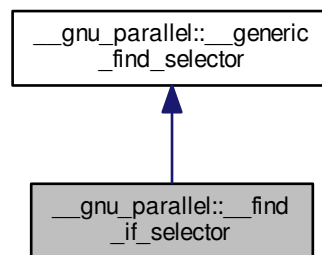
Definition at line 169 of file find_selectors.h.

The documentation for this struct was generated from the following file:

- [find_selectors.h](#)

4.82 __gnu_parallel::__find_if_selector Struct Reference

Inheritance diagram for __gnu_parallel::__find_if_selector:



Public Member Functions

- `template<typename _RAIter1, typename _RAIter2, typename _Pred >`
`std::pair<_RAIter1, _RAIter2 > M_sequential_algorithm (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred)`
- `template<typename _RAIter1, typename _RAIter2, typename _Pred >`
`bool operator\(\) (_RAIter1 __i1, _RAIter2 __i2, _Pred __pred)`

4.82.1 Detailed Description

Test predicate on a single element, used for `std::find()` and `std::find_if()`.

Definition at line 50 of file find_selectors.h.

4.82.2 Member Function Documentation

- 4.82.2.1 `template<typename _RAIter1, typename _RAIter2, typename _Pred > std::pair<_RAIter1, _RAIter2>`
`__gnu_parallel::__find_if_selector::M_sequential_algorithm (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred) [inline]`

Corresponding sequential algorithm on a sequence.

Parameters

<code>__begin1</code>	Begin iterator of first sequence.
<code>__end1</code>	End iterator of first sequence.
<code>__begin2</code>	Begin iterator of second sequence.
<code>__pred</code>	Find predicate.

Definition at line 72 of file `find_selectors.h`.

References `std::make_pair()`.

```
4.82.2.2 template<typename _RAIter1, typename _RAIter2, typename _Pred> bool __gnu_parallel::__find_if_selector::operator() (
    _RAIter1 __i1, _RAIter2 __i2, _Pred __pred ) [inline]
```

Test on one position.

Parameters

<code>__i1</code>	_Iterator on first sequence.
<code>__i2</code>	_Iterator on second sequence (unused).
<code>__pred</code>	Find predicate.

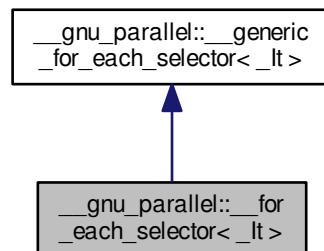
Definition at line 60 of file `find_selectors.h`.

The documentation for this struct was generated from the following file:

- [find_selectors.h](#)

4.83 __gnu_parallel::__for_each_selector<_It> Struct Template Reference

Inheritance diagram for `__gnu_parallel::__for_each_selector<_It>`:



Public Member Functions

- `template<typename _Op >`
`bool operator() (_Op &__o, _It __i)`

Public Attributes

- `_It _M_finish_iterator`

4.83.1 Detailed Description

```
template<typename _It>
struct __gnu_parallel::__for_each_selector< _It >
```

`std::for_each()` selector.

Definition at line 52 of file `for_each_selectors.h`.

4.83.2 Member Function Documentation

4.83.2.1 `template<typename _It> template<typename _Op> bool __gnu_parallel::__for_each_selector< _It >::operator()`
`(_Op &__o, _It __i) [inline]`

Functor execution.

Parameters

<code>__o</code>	Operator.
<code>__i</code>	iterator referencing object.

Definition at line 59 of file `for_each_selectors.h`.

4.83.3 Member Data Documentation

4.83.3.1 `template<typename _It> _It __gnu_parallel::__generic_for_each_selector< _It >::M_finish_iterator`
`[inherited]`

`_It` iterator on last element processed; needed for some algorithms (e. g. `std::transform()`).

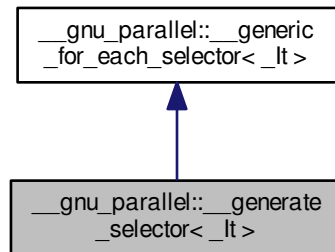
Definition at line 47 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- `for_each_selectors.h`

4.84 `__gnu_parallel::__generate_selector<_It>` Struct Template Reference

Inheritance diagram for `__gnu_parallel::__generate_selector<_It>`:



Public Member Functions

- `template<typename _Op>`
`bool operator() (_Op &__o, _It __i)`

Public Attributes

- `_It _M_finish_iterator`

4.84.1 Detailed Description

```
template<typename _It>
struct __gnu_parallel::__generate_selector<_It>
```

`std::generate()` selector.

Definition at line 68 of file `for_each_selectors.h`.

4.84.2 Member Function Documentation

4.84.2.1 `template<typename _It> template<typename _Op> bool __gnu_parallel::__generate_selector<_It>::operator()`
`(_Op &__o, _It __i) [inline]`

Functor execution.

Parameters

<code>_↔ _o</code>	Operator.
<code>_↔ _i</code>	iterator referencing object.

Definition at line 75 of file `for_each_selectors.h`.

4.84.3 Member Data Documentation

4.84.3.1 `template<typename _It> _It __gnu_parallel::__generic_for_each_selector<_It>::__M_finish_iterator` [inherited]

`_Iterator` on last element processed; needed for some algorithms (e. g. `std::transform()`).

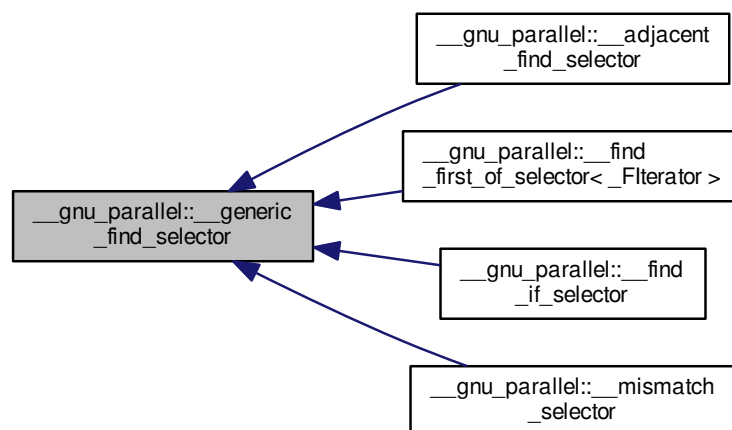
Definition at line 47 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for_each_selectors.h](#)

4.85 `__gnu_parallel::__generic_find_selector` Struct Reference

Inheritance diagram for `__gnu_parallel::__generic_find_selector`:



4.85.1 Detailed Description

Base class of all `__gnu_parallel::__find_template` selectors.

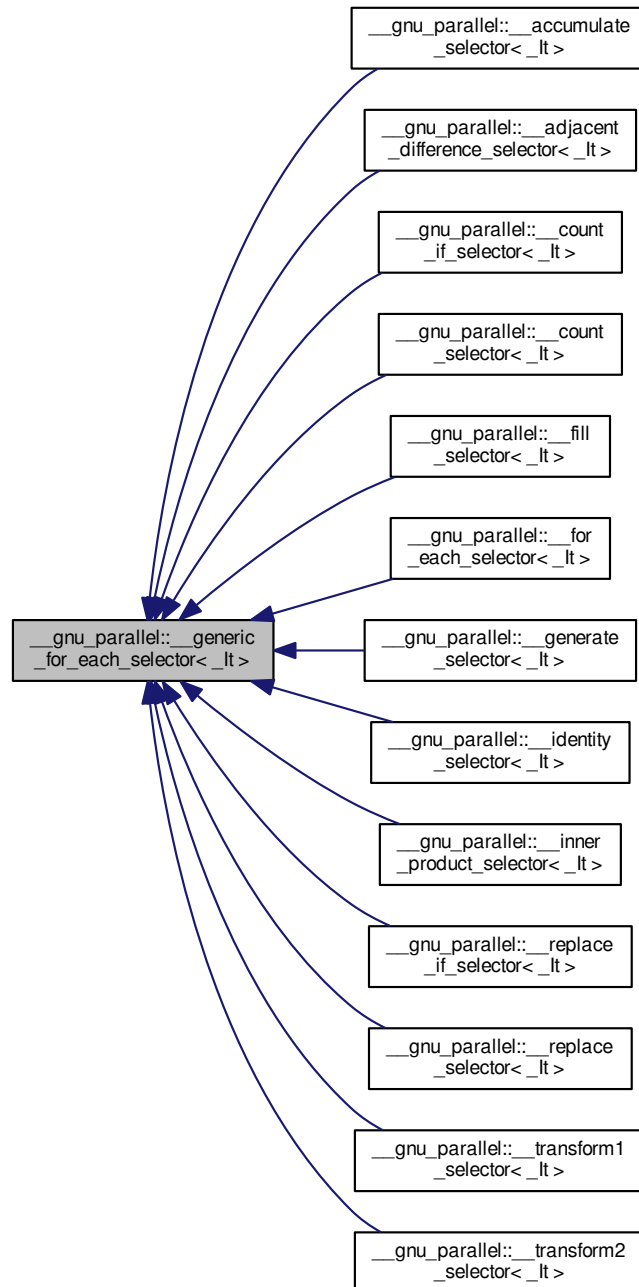
Definition at line 43 of file `find_selectors.h`.

The documentation for this struct was generated from the following file:

- [find_selectors.h](#)

4.86 `__gnu_parallel::__generic_for_each_selector<_It>` Struct Template Reference

Inheritance diagram for `__gnu_parallel::__generic_for_each_selector<_It>`:



Public Attributes

- [_It_M_finish_iterator](#)

4.86.1 Detailed Description

```
template<typename _It>
struct __gnu_parallel::__generic_for_each_selector<_It>
```

Generic `__selector` for embarrassingly parallel functions.

Definition at line 42 of file `for_each_selectors.h`.

4.86.2 Member Data Documentation

4.86.2.1 `template<typename _It> _It __gnu_parallel::__generic_for_each_selector<_It>::M_finish_iterator`

`_Iterator` on last element processed; needed for some algorithms (e. g. `std::transform()`).

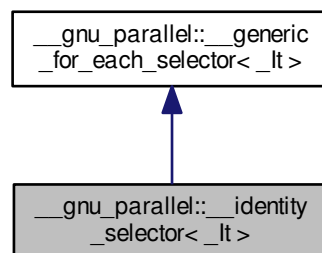
Definition at line 47 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for_each_selectors.h](#)

4.87 `__gnu_parallel::__identity_selector<_It>` Struct Template Reference

Inheritance diagram for `__gnu_parallel::__identity_selector<_It>`:



Public Member Functions

- `template<typename _Op >
_It operator() (_Op __o, _It __i)`

Public Attributes

- `_It _M_finish_iterator`

4.87.1 Detailed Description

```
template<typename _It>
struct __gnu_parallel::__identity_selector< _It >
```

Selector that just returns the passed iterator.

Definition at line 253 of file `for_each_selectors.h`.

4.87.2 Member Function Documentation

4.87.2.1 `template<typename _It> template<typename _Op> _It __gnu_parallel::__identity_selector< _It >::operator() (`
`_Op __o, _It __i) [inline]`

Functor execution.

Parameters

<code>__o</code>	Operator (unused).
<code>__i</code>	iterator referencing object.

Returns

Passed iterator.

Definition at line 261 of file `for_each_selectors.h`.

4.87.3 Member Data Documentation

4.87.3.1 `template<typename _It> _It __gnu_parallel::__generic_for_each_selector< _It >::M_finish_iterator`
`[inherited]`

Iterator on last element processed; needed for some algorithms (e. g. `std::transform()`).

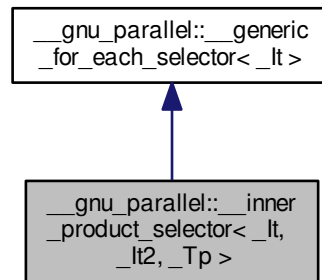
Definition at line 47 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for_each_selectors.h](#)

4.88 `__gnu_parallel::__inner_product_selector<_It, _It2, _Tp>` Struct Template Reference

Inheritance diagram for `__gnu_parallel::__inner_product_selector<_It, _It2, _Tp>`:



Public Member Functions

- [__inner_product_selector](#) (`_It __b1, _It2 __b2`)
- `template<typename _Op>`
`_Tp operator()` (`_Op __mult, _It __current`)

Public Attributes

- `_It __begin1_iterator`
- `_It2 __begin2_iterator`
- `_It __M_finish_iterator`

4.88.1 Detailed Description

```
template<typename _It, typename _It2, typename _Tp>
struct __gnu_parallel::__inner_product_selector<_It, _It2, _Tp>
```

`std::inner_product()` selector.

Definition at line 222 of file `for_each_selectors.h`.

4.88.2 Constructor & Destructor Documentation

4.88.2.1 `template<typename _It, typename _It2, typename _Tp> __gnu_parallel::__inner_product_selector<_It, _It2, _Tp>::__inner_product_selector (_It __b1, _It2 __b2)` `[inline]`, `[explicit]`

Constructor.

Parameters

<code>__b1</code>	Begin iterator of first sequence.
<code>__b2</code>	Begin iterator of second sequence.

Definition at line 234 of file `for_each_selectors.h`.

4.88.3 Member Function Documentation

4.88.3.1 `template<typename _It, typename _It2, typename _Tp> template<typename _Op> _Tp
__gnu_parallel::__inner_product_selector<_It, _It2, _Tp>::operator() (_Op __mult, _It __current)
[inline]`

Functor execution.

Parameters

<code>__mult</code>	Multiplication functor.
<code>__current</code>	iterator referencing object.

Returns

Inner product elemental `__result`.

Definition at line 243 of file `for_each_selectors.h`.

4.88.4 Member Data Documentation

4.88.4.1 `template<typename _It, typename _It2, typename _Tp> _It __gnu_parallel::__inner_product_selector<_It, _It2,
_Tp>::__begin1_iterator`

Begin iterator of first sequence.

Definition at line 225 of file `for_each_selectors.h`.

4.88.4.2 `template<typename _It, typename _It2, typename _Tp> _It2 __gnu_parallel::__inner_product_selector<_It, _It2,
_Tp>::__begin2_iterator`

Begin iterator of second sequence.

Definition at line 228 of file `for_each_selectors.h`.

4.88.4.3 `template<typename _It> _It __gnu_parallel::__generic_for_each_selector<_It>::_M_finish_iterator`
[inherited]

`_Iterator` on last element processed; needed for some algorithms (e. g. `std::transform()`).

Definition at line 47 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for_each_selectors.h](#)

4.89 `__gnu_parallel::__max_element_reduct<_Compare, _It>` Struct Template Reference

Public Member Functions

- `__max_element_reduct` (`_Compare &__c`)
- `_It operator()` (`_It __x, _It __y`)

Public Attributes

- `_Compare & __comp`

4.89.1 Detailed Description

```
template<typename _Compare, typename _It>
struct __gnu_parallel::__max_element_reduct<_Compare, _It>
```

Reduction for finding the maximum element, using a comparator.

Definition at line 321 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for_each_selectors.h](#)

4.90 `__gnu_parallel::__min_element_reduct<_Compare, _It>` Struct Template Reference

Public Member Functions

- `__min_element_reduct` (`_Compare &__c`)
- `_It operator()` (`_It __x, _It __y`)

Public Attributes

- `_Compare` & `__comp`

4.90.1 Detailed Description

```
template<typename _Compare, typename _It>
struct __gnu_parallel::__min_element_reduct< _Compare, _It >
```

Reduction for finding the maximum element, using a comparator.

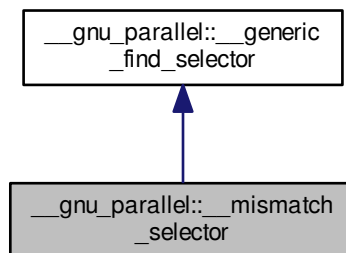
Definition at line 307 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for_each_selectors.h](#)

4.91 `__gnu_parallel::__mismatch_selector` Struct Reference

Inheritance diagram for `__gnu_parallel::__mismatch_selector`:



Public Member Functions

- `template<typename _RAIter1, typename _RAIter2, typename _Pred >`
`std::pair<_RAIter1, _RAIter2 > _M_sequential_algorithm (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred)`
- `template<typename _RAIter1, typename _RAIter2, typename _Pred >`
`bool operator\(\) (_RAIter1 __i1, _RAIter2 __i2, _Pred __pred)`

4.91.1 Detailed Description

Test inverted predicate on a single element.

Definition at line 119 of file `find_selectors.h`.

4.91.2 Member Function Documentation

4.91.2.1 `template<typename _RAIter1, typename _RAIter2, typename _Pred > std::pair<_RAIter1, _RAIter2> __gnu_parallel::__mismatch_selector::M_sequential_algorithm (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred) [inline]`

Corresponding sequential algorithm on a sequence.

Parameters

<code>__begin1</code>	Begin iterator of first sequence.
<code>__end1</code>	End iterator of first sequence.
<code>__begin2</code>	Begin iterator of second sequence.
<code>__pred</code>	Find predicate.

Definition at line 143 of file `find_selectors.h`.

4.91.2.2 `template<typename _RAIter1, typename _RAIter2, typename _Pred > bool __gnu_parallel::__mismatch_selector<_RAIter1, _RAIter2, _Pred>::operator() (_RAIter1 __i1, _RAIter2 __i2, _Pred __pred) [inline]`

Test on one position.

Parameters

<code>__i1</code>	_Iterator on first sequence.
<code>__i2</code>	_Iterator on second sequence (unused).
<code>__pred</code>	Find predicate.

Definition at line 130 of file `find_selectors.h`.

The documentation for this struct was generated from the following file:

- [find_selectors.h](#)

4.92 `__gnu_parallel::__multiway_merge_3_variant_sentinel_switch< __sentinels, _RAIterlterator, _RAIter3, _DifferenceTp, _Compare >` Struct Template Reference

Public Member Functions

- `_RAIter3 operator() (_RAIterlterator __seqs_begin, _RAIterlterator __seqs_end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp)`

4.92.1 Detailed Description

```
template<bool __sentinels, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>
struct __gnu_parallel::__multiway_merge_3_variant_sentinel_switch< __sentinels, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare
>
```

Switch for 3-way merging with __sentinels turned off.

Note that 3-way merging is always stable!

Definition at line 752 of file multiway_merge.h.

The documentation for this struct was generated from the following file:

- [multiway_merge.h](#)

4.93 __gnu_parallel::__multiway_merge_3_variant_sentinel_switch< true, _RAIterIterator, _RAIter3, _↵ _DifferenceTp, _Compare > Struct Template Reference

Public Member Functions

- **_RAIter3 operator()** (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _↵
_DifferenceTp __length, _Compare __comp)

4.93.1 Detailed Description

```
template<typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>
struct __gnu_parallel::__multiway_merge_3_variant_sentinel_switch< true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >
```

Switch for 3-way merging with __sentinels turned on.

Note that 3-way merging is always stable!

Definition at line 772 of file multiway_merge.h.

The documentation for this struct was generated from the following file:

- [multiway_merge.h](#)

4.94 __gnu_parallel::__multiway_merge_4_variant_sentinel_switch< __sentinels, _RAIterIterator, _RAIter3, ↵ _DifferenceTp, _Compare > Struct Template Reference

Public Member Functions

- **_RAIter3 operator()** (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _↵
_DifferenceTp __length, _Compare __comp)

4.94.1 Detailed Description

```
template<bool __sentinels, typename _RAIterlterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>
struct __gnu_parallel::__multiway_merge_4_variant_sentinel_switch< __sentinels, _RAIterlterator, _RAIter3, _DifferenceTp, _Compare
>
```

Switch for 4-way merging with `__sentinels` turned off.

Note that 4-way merging is always stable!

Definition at line 795 of file `multiway_merge.h`.

The documentation for this struct was generated from the following file:

- [multiway_merge.h](#)

4.95 `__gnu_parallel::__multiway_merge_4_variant_sentinel_switch< true, _RAIterlterator, _RAIter3, _DifferenceTp, _Compare > Struct Template Reference`

Public Member Functions

- `_RAIter3 operator() (_RAIterlterator __seqs_begin, _RAIterlterator __seqs_end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp)`

4.95.1 Detailed Description

```
template<typename _RAIterlterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>
struct __gnu_parallel::__multiway_merge_4_variant_sentinel_switch< true, _RAIterlterator, _RAIter3, _DifferenceTp, _Compare >
```

Switch for 4-way merging with `__sentinels` turned on.

Note that 4-way merging is always stable!

Definition at line 815 of file `multiway_merge.h`.

The documentation for this struct was generated from the following file:

- [multiway_merge.h](#)

4.96 `__gnu_parallel::__multiway_merge_k_variant_sentinel_switch< __sentinels, __stable, _RAIterlterator, _RAIter3, _DifferenceTp, _Compare > Struct Template Reference`

Public Member Functions

- `_RAIter3 operator() (_RAIterlterator __seqs_begin, _RAIterlterator __seqs_end, _RAIter3 __target, const typename std::iterator_traits< typename std::iterator_traits< _RAIterlterator >::value_type::first_type >::value_type & __sentinel, _DifferenceTp __length, _Compare __comp)`

4.96.1 Detailed Description

```
template<bool __sentinels, bool __stable, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>
struct __gnu_parallel::__multiway_merge_k_variant_sentinel_switch< __sentinels, __stable, _RAIterIterator, _RAIter3, _DifferenceTp,
    _Compare >
```

Switch for k-way merging with `__sentinels` turned on.

Definition at line 837 of file `multiway_merge.h`.

The documentation for this struct was generated from the following file:

- [multiway_merge.h](#)

4.97 `__gnu_parallel::__multiway_merge_k_variant_sentinel_switch< false, __stable, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >` Struct Template Reference

Public Member Functions

- `_RAIter3 operator() (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator >::value_type::first_type >::value_type & __sentinel, _DifferenceTp __length, _Compare __comp)`

4.97.1 Detailed Description

```
template<bool __stable, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>
struct __gnu_parallel::__multiway_merge_k_variant_sentinel_switch< false, __stable, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >
```

Switch for k-way merging with `__sentinels` turned off.

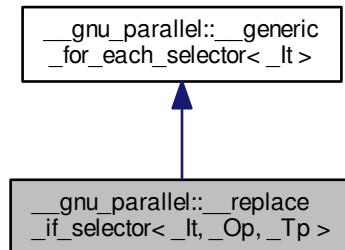
Definition at line 872 of file `multiway_merge.h`.

The documentation for this struct was generated from the following file:

- [multiway_merge.h](#)

4.98 `__gnu_parallel::__replace_if_selector<_It,_Op,_Tp>` Struct Template Reference

Inheritance diagram for `__gnu_parallel::__replace_if_selector<_It,_Op,_Tp>`:



Public Member Functions

- `__replace_if_selector` (const `_Tp` & `__new_val`)
- bool `operator()` (`_Op` & `__o`, `_It` `__i`)

Public Attributes

- const `_Tp` & `__new_val`
- `_It` `_M_finish_iterator`

4.98.1 Detailed Description

```
template<typename _It, typename _Op, typename _Tp>
struct __gnu_parallel::__replace_if_selector<_It,_Op,_Tp>
```

`std::replace()` selector.

Definition at line 156 of file `for_each_selectors.h`.

4.98.2 Constructor & Destructor Documentation

4.98.2.1 `template<typename _It, typename _Op, typename _Tp> __gnu_parallel::__replace_if_selector<_It,_Op,_Tp>::__replace_if_selector (const _Tp & __new_val) [inline],[explicit]`

Constructor.

Parameters

<code>__new_val</code>	Value to replace with.
------------------------	------------------------

Definition at line 164 of file `for_each_selectors.h`.

4.98.3 Member Function Documentation

4.98.3.1 `template<typename _It, typename _Op, typename _Tp> bool __gnu_parallel::__replace_if_selector<_It, _Op, _Tp>::operator() (_Op & __o, _It __i) [inline]`

Functor execution.

Parameters

<code>__o</code>	Operator.
<code>__i</code>	iterator referencing object.

Definition at line 170 of file `for_each_selectors.h`.

4.98.4 Member Data Documentation

4.98.4.1 `template<typename _It, typename _Op, typename _Tp> const _Tp& __gnu_parallel::__replace_if_selector<_It, _Op, _Tp>::__new_val`

Value to replace with.

Definition at line 159 of file `for_each_selectors.h`.

4.98.4.2 `template<typename _It > _It __gnu_parallel::__generic_for_each_selector<_It >::__M_finish_iterator [inherited]`

`_Iterator` on last element processed; needed for some algorithms (e. g. `std::transform()`).

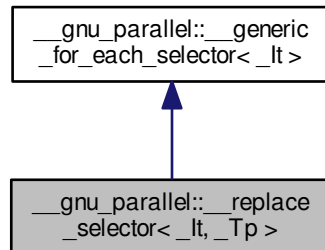
Definition at line 47 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for_each_selectors.h](#)

4.99 __gnu_parallel::__replace_selector<_It,_Tp> Struct Template Reference

Inheritance diagram for __gnu_parallel::__replace_selector<_It,_Tp>:



Public Member Functions

- [__replace_selector](#) (const _Tp & [__new_val](#))
- bool [operator\(\)](#) (_Tp &__v, _It __i)

Public Attributes

- const _Tp & [__new_val](#)
- _It [_M_finish_iterator](#)

4.99.1 Detailed Description

```
template<typename _It, typename _Tp>
struct __gnu_parallel::__replace_selector<_It,_Tp>
```

std::replace() selector.

Definition at line 132 of file `for_each_selectors.h`.

4.99.2 Constructor & Destructor Documentation

4.99.2.1 `template<typename _It, typename _Tp> __gnu_parallel::__replace_selector<_It,_Tp>::__replace_selector (const _Tp & __new_val) [inline],[explicit]`

Constructor.

Parameters

<code>__new_val</code>	Value to replace with.
------------------------	------------------------

Definition at line 140 of file `for_each_selectors.h`.

4.99.3 Member Function Documentation

4.99.3.1 `template<typename _It, typename _Tp> bool __gnu_parallel::__replace_selector<_It, _Tp>::operator() (_Tp & __v, _It __i) [inline]`

Functor execution.

Parameters

<code>__v</code>	Current value.
<code>__i</code>	iterator referencing object.

Definition at line 146 of file `for_each_selectors.h`.

4.99.4 Member Data Documentation

4.99.4.1 `template<typename _It, typename _Tp> const _Tp& __gnu_parallel::__replace_selector<_It, _Tp>::__new_val`

Value to replace with.

Definition at line 135 of file `for_each_selectors.h`.

4.99.4.2 `template<typename _It> _It __gnu_parallel::__generic_for_each_selector<_It>::__M_finish_iterator [inherited]`

`_Iterator` on last element processed; needed for some algorithms (e. g. `std::transform()`).

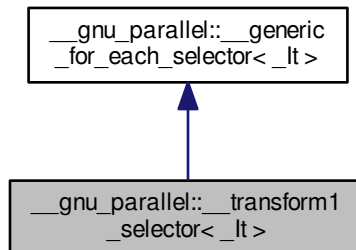
Definition at line 47 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for_each_selectors.h](#)

4.100 `__gnu_parallel::__transform1_selector<_It>` Struct Template Reference

Inheritance diagram for `__gnu_parallel::__transform1_selector<_It>`:



Public Member Functions

- `template<typename _Op>`
`bool operator() (_Op &__o, _It __i)`

Public Attributes

- `_It _M_finish_iterator`

4.100.1 Detailed Description

```
template<typename _It>
struct __gnu_parallel::__transform1_selector<_It>
```

`std::transform()` `__selector`, one input sequence variant.

Definition at line 100 of file `for_each_selectors.h`.

4.100.2 Member Function Documentation

4.100.2.1 `template<typename _It> template<typename _Op> bool __gnu_parallel::__transform1_selector<_It>::operator() (_Op &__o, _It __i) [inline]`

Functor execution.

Parameters

<code>_↔ _o</code>	Operator.
<code>_↔ _i</code>	iterator referencing object.

Definition at line 107 of file `for_each_selectors.h`.

4.100.3 Member Data Documentation
4.100.3.1 `template<typename _It > _It __gnu_parallel::__generic_for_each_selector< _It >::M_finish_iterator`
 [inherited]

`_Iterator` on last element processed; needed for some algorithms (e. g. `std::transform()`).

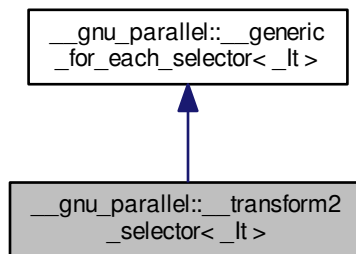
Definition at line 47 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for_each_selectors.h](#)

4.101 __gnu_parallel::__transform2_selector< _It > Struct Template Reference

Inheritance diagram for `__gnu_parallel::__transform2_selector< _It >`:

**Public Member Functions**

- `template<typename _Op >`
`bool operator() (_Op &__o, _It __i)`

Public Attributes

- [_It _M_finish_iterator](#)

4.101.1 Detailed Description

```
template<typename _It>
struct __gnu_parallel::__transform2_selector<_It>
```

`std::transform()` __selector, two input sequences variant.

Definition at line 116 of file `for_each_selectors.h`.

4.101.2 Member Function Documentation

4.101.2.1 `template<typename _It> template<typename _Op> bool __gnu_parallel::__transform2_selector<_It>::operator()(_Op &_o, _It _i) [inline]`

Functor execution.

Parameters

<code>_o</code>	Operator.
<code>_i</code>	iterator referencing object.

Definition at line 123 of file `for_each_selectors.h`.

4.101.3 Member Data Documentation

4.101.3.1 `template<typename _It> _It __gnu_parallel::__generic_for_each_selector<_It>::__M_finish_iterator [inherited]`

__iterator on last element processed; needed for some algorithms (e. g. `std::transform()`).

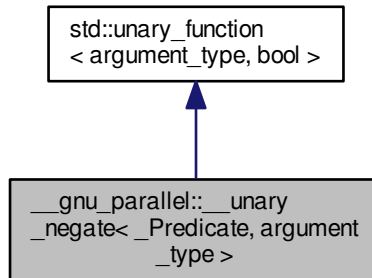
Definition at line 47 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for_each_selectors.h](#)

4.102 `__gnu_parallel::__unary_negate<_Predicate, argument_type >` Class Template Reference

Inheritance diagram for `__gnu_parallel::__unary_negate<_Predicate, argument_type >`:



Public Types

- typedef `argument_type` `argument_type`
- typedef `bool` `result_type`

Public Member Functions

- `__unary_negate` (const `_Predicate` &`__x`)
- `bool operator()` (const `argument_type` &`__x`)

Protected Attributes

- `_Predicate` `_M_pred`

4.102.1 Detailed Description

```
template<typename _Predicate, typename argument_type>
class __gnu_parallel::__unary_negate<_Predicate, argument_type >
```

Similar to `std::unary_negate`, but giving the argument types explicitly.

Definition at line 173 of file `parallel/base.h`.

4.102.2 Member Typedef Documentation

4.102.2.1 `typedef argument_type std::unary_function< argument_type, bool >::argument_type` [inherited]

`argument_type` is the type of the argument

Definition at line 108 of file `stl_function.h`.

4.102.2.2 `typedef bool std::unary_function< argument_type, bool >::result_type` [inherited]

`result_type` is the return type

Definition at line 111 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [parallel/base.h](#)

4.103 `__gnu_parallel::__DRandomShufflingGlobalData<_RAIter>` Struct Template Reference

Public Types

- `typedef _TraitsType::difference_type` **DifferenceType**
- `typedef std::iterator_traits<_RAIter>` **TraitsType**
- `typedef _TraitsType::value_type` **ValueType**

Public Member Functions

- [_DRandomShufflingGlobalData](#) (`_RAIter &__source`)

Public Attributes

- [_ThreadIndex](#) * [_M_bin_proc](#)
- `DifferenceType` ** [_M_dist](#)
- `int` [_M_num_bins](#)
- `int` [_M_num_bits](#)
- `_RAIter &` [_M_source](#)
- `DifferenceType` * [_M_starts](#)
- `ValueType` ** [_M_temporaries](#)

4.103.1 Detailed Description

```
template<typename _RAIter>
struct __gnu_parallel::__DRandomShufflingGlobalData<_RAIter>
```

Data known to every thread participating in `__gnu_parallel::__parallel_random_shuffle()`.

Definition at line 52 of file `random_shuffle.h`.

4.103.2 Constructor & Destructor Documentation

4.103.2.1 `template<typename _RAIter> __gnu_parallel::_DRandomShufflingGlobalData< _RAIter
>::_DRandomShufflingGlobalData (_RAIter & __source) [inline]`

Constructor.

Definition at line 83 of file `random_shuffle.h`.

4.103.3 Member Data Documentation

4.103.3.1 `template<typename _RAIter> _ThreadIndex* __gnu_parallel::_DRandomShufflingGlobalData< _RAIter
>::_M_bin_proc`

Number of the thread that will further process the corresponding bin.

Definition at line 74 of file `random_shuffle.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`.

4.103.3.2 `template<typename _RAIter> _DifferenceType** __gnu_parallel::_DRandomShufflingGlobalData< _RAIter
>::_M_dist`

Two-dimensional array to hold the thread-bin distribution.

Dimensions `(_M_num_threads + 1) __x (_M_num_bins + 1)`.

Definition at line 67 of file `random_shuffle.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`, and `__gnu_parallel::__parallel_random_shuffle_drs↵
_pu()`.

4.103.3.3 `template<typename _RAIter> int __gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M_num_bins`

Number of bins to distribute to.

Definition at line 77 of file `random_shuffle.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`, and `__gnu_parallel::__parallel_random_shuffle_drs↵
_pu()`.

4.103.3.4 `template<typename _RAIter> int __gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M_num_bits`

Number of bits needed to address the bins.

Definition at line 80 of file `random_shuffle.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`, and `__gnu_parallel::__parallel_random_shuffle_drs↵
_pu()`.

4.103.3.5 `template<typename _RAIter> _RAIter& __gnu_parallel::_DRandomShufflingGlobalData<_RAIter>::_M_source`

Begin iterator of the `__source`.

Definition at line 59 of file `random_shuffle.h`.

4.103.3.6 `template<typename _RAIter> _DifferenceType* __gnu_parallel::_DRandomShufflingGlobalData<_RAIter>::_M_starts`

Start indexes of the threads' `__chunks`.

Definition at line 70 of file `random_shuffle.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`, and `__gnu_parallel::__parallel_random_shuffle_drs↔_pu()`.

4.103.3.7 `template<typename _RAIter> _ValueType** __gnu_parallel::_DRandomShufflingGlobalData<_RAIter>::_M_temporaries`

Temporary arrays for each thread.

Definition at line 62 of file `random_shuffle.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`.

The documentation for this struct was generated from the following file:

- [random_shuffle.h](#)

4.104 `__gnu_parallel::_DRSSorterPU<_RAIter, _RandomNumberGenerator>` Struct Template Reference

Public Attributes

- [_BinIndex __bins_end](#)
- [_BinIndex _M_bins_begin](#)
- [int _M_num_threads](#)
- [_DRandomShufflingGlobalData<_RAIter> * _M_sd](#)
- [uint32_t _M_seed](#)

4.104.1 Detailed Description

```
template<typename _RAIter, typename _RandomNumberGenerator>
struct __gnu_parallel::_DRSSorterPU<_RAIter, _RandomNumberGenerator>
```

Local data for a thread participating in `__gnu_parallel::__parallel_random_shuffle()`.

Definition at line 91 of file `random_shuffle.h`.

4.104.2 Member Data Documentation

4.104.2.1 `template<typename _RAIter, typename _RandomNumberGenerator> _BinIndex __gnu_parallel::_DRSSorterPU<_RAIter, _RandomNumberGenerator>::_bins_end`

End index for bins taken care of by this thread.

Definition at line 100 of file `random_shuffle.h`.

Referenced by `__gnu_parallel::_parallel_random_shuffle_drs()`.

4.104.2.2 `template<typename _RAIter, typename _RandomNumberGenerator> _BinIndex __gnu_parallel::_DRSSorterPU<_RAIter, _RandomNumberGenerator>::_M_bins_begin`

Begin index for bins taken care of by this thread.

Definition at line 97 of file `random_shuffle.h`.

Referenced by `__gnu_parallel::_parallel_random_shuffle_drs()`.

4.104.2.3 `template<typename _RAIter, typename _RandomNumberGenerator> int __gnu_parallel::_DRSSorterPU<_RAIter, _RandomNumberGenerator>::_M_num_threads`

Number of threads participating in total.

Definition at line 94 of file `random_shuffle.h`.

Referenced by `__gnu_parallel::_parallel_random_shuffle_drs()`, and `__gnu_parallel::_parallel_random_shuffle_drs↵_pu()`.

4.104.2.4 `template<typename _RAIter, typename _RandomNumberGenerator> _DRandomShufflingGlobalData<_RAIter>* __gnu_parallel::_DRSSorterPU<_RAIter, _RandomNumberGenerator>::_M_sd`

Pointer to global data.

Definition at line 106 of file `random_shuffle.h`.

Referenced by `__gnu_parallel::_parallel_random_shuffle_drs()`, and `__gnu_parallel::_parallel_random_shuffle_drs↵_pu()`.

4.104.2.5 `template<typename _RAIter, typename _RandomNumberGenerator> uint32_t __gnu_parallel::_DRSSorterPU<_RAIter, _RandomNumberGenerator>::_M_seed`

Random `_M_seed` for this thread.

Definition at line 103 of file `random_shuffle.h`.

Referenced by `__gnu_parallel::_parallel_random_shuffle_drs()`, and `__gnu_parallel::_parallel_random_shuffle_drs↵_pu()`.

The documentation for this struct was generated from the following file:

- [random_shuffle.h](#)

4.105 `__gnu_parallel::_DummyReduct` Struct Reference

Public Member Functions

- `bool operator()` (`bool`, `bool`) `const`

4.105.1 Detailed Description

Reduction function doing nothing.

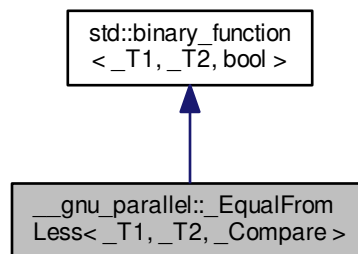
Definition at line 298 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for_each_selectors.h](#)

4.106 `__gnu_parallel::_EqualFromLess<_T1, _T2, _Compare>` Class Template Reference

Inheritance diagram for `__gnu_parallel::_EqualFromLess<_T1, _T2, _Compare>`:



Public Types

- `typedef _T1` [first_argument_type](#)
- `typedef bool` [result_type](#)
- `typedef _T2` [second_argument_type](#)

Public Member Functions

- `_EqualFromLess` (`_Compare` &`__comp`)
- `bool operator()` (`const _T1` &`__a`, `const _T2` &`__b`)

4.106.1 Detailed Description

```
template<typename _T1, typename _T2, typename _Compare>
class __gnu_parallel::_EqualFromLess< _T1, _T2, _Compare >
```

Constructs predicate for equality from strict weak ordering predicate.

Definition at line 157 of file parallel/base.h.

4.106.2 Member Typedef Documentation

4.106.2.1 `typedef _T1 std::binary_function< _T1, _T2, bool >::first_argument_type` [inherited]

`first_argument_type` is the type of the first argument

Definition at line 121 of file stl_function.h.

4.106.2.2 `typedef bool std::binary_function< _T1, _T2, bool >::result_type` [inherited]

`result_type` is the return type

Definition at line 127 of file stl_function.h.

4.106.2.3 `typedef _T2 std::binary_function< _T1, _T2, bool >::second_argument_type` [inherited]

`second_argument_type` is the type of the second argument

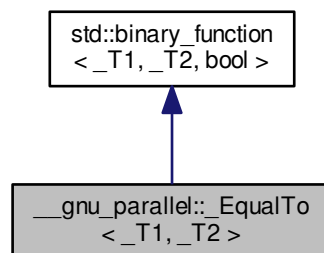
Definition at line 124 of file stl_function.h.

The documentation for this class was generated from the following file:

- [parallel/base.h](#)

4.107 `__gnu_parallel::_EqualTo< _T1, _T2 >` Struct Template Reference

Inheritance diagram for `__gnu_parallel::_EqualTo< _T1, _T2 >`:



Public Types

- typedef `_T1` [first_argument_type](#)
- typedef `bool` [result_type](#)
- typedef `_T2` [second_argument_type](#)

Public Member Functions

- `bool operator()` (`const _T1 &__t1, const _T2 &__t2`) `const`

4.107.1 Detailed Description

```
template<typename _T1, typename _T2>
struct __gnu_parallel::_EqualTo<_T1, _T2>
```

Similar to `std::equal_to`, but allows two different types.

Definition at line 244 of file `parallel/base.h`.

4.107.2 Member Typedef Documentation

4.107.2.1 `typedef _T1 std::binary_function<_T1, _T2, bool>::first_argument_type` `[inherited]`

`first_argument_type` is the type of the first argument

Definition at line 121 of file `stl_function.h`.

4.107.2.2 `typedef bool std::binary_function<_T1, _T2, bool>::result_type` `[inherited]`

`result_type` is the return type

Definition at line 127 of file `stl_function.h`.

4.107.2.3 `typedef _T2 std::binary_function<_T1, _T2, bool>::second_argument_type` `[inherited]`

`second_argument_type` is the type of the second argument

Definition at line 124 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [parallel/base.h](#)

4.108 `__gnu_parallel::_GuardedIterator<_RAIter, _Compare >` Class Template Reference

Public Member Functions

- `_GuardedIterator` (`_RAIter __begin`, `_RAIter __end`, `_Compare &__comp`)
- `operator _RAIter` ()
- `std::iterator_traits<_RAIter >::value_type & operator*` ()
- `_GuardedIterator<_RAIter, _Compare > & operator++` ()

Friends

- `bool operator<` (`_GuardedIterator<_RAIter, _Compare > &__bi1`, `_GuardedIterator<_RAIter, _Compare > &__bi2`)
- `bool operator<=` (`_GuardedIterator<_RAIter, _Compare > &__bi1`, `_GuardedIterator<_RAIter, _Compare > &__bi2`)

4.108.1 Detailed Description

```
template<typename _RAIter, typename _Compare>
class __gnu_parallel::_GuardedIterator<_RAIter, _Compare >
```

`_Iterator` wrapper supporting an implicit supremum at the end of the sequence, dominating all comparisons.

The implicit supremum comes with a performance cost.

Deriving from `_RAIter` is not possible since `_RAIter` need not be a class.

Definition at line 73 of file `multiway_merge.h`.

4.108.2 Constructor & Destructor Documentation

4.108.2.1 `template<typename _RAIter, typename _Compare > __gnu_parallel::_GuardedIterator<_RAIter, _Compare >::_GuardedIterator (_RAIter __begin, _RAIter __end, _Compare &__comp)` `[inline]`

Constructor. Sets iterator to beginning of sequence.

Parameters

<code>__begin</code>	Begin iterator of sequence.
<code>__end</code>	End iterator of sequence.
<code>__comp</code>	Comparator provided for associated overloaded compare operators.

Definition at line 91 of file `multiway_merge.h`.

4.108.3 Member Function Documentation

4.108.3.1 `template<typename _RAIter, typename _Compare> __gnu_parallel::_GuardedIterator<_RAIter, _Compare>::operator _RAIter () [inline]`

Convert to wrapped iterator.

Returns

Wrapped iterator.

Definition at line 112 of file `multiway_merge.h`.

4.108.3.2 `template<typename _RAIter, typename _Compare> std::iterator_traits<_RAIter>::value_type& __gnu_parallel::_GuardedIterator<_RAIter, _Compare>::operator* () [inline]`

Dereference operator.

Returns

Referenced element.

Definition at line 107 of file `multiway_merge.h`.

4.108.3.3 `template<typename _RAIter, typename _Compare> _GuardedIterator<_RAIter, _Compare>& __gnu_parallel::_GuardedIterator<_RAIter, _Compare>::operator++ () [inline]`

Pre-increment operator.

Returns

This.

Definition at line 98 of file `multiway_merge.h`.

4.108.4 Friends And Related Function Documentation

4.108.4.1 `template<typename _RAIter, typename _Compare> bool operator< (_GuardedIterator<_RAIter, _Compare> & __b1, _GuardedIterator<_RAIter, _Compare> & __b2) [friend]`

Compare two elements referenced by guarded iterators.

Parameters

<code>__b1</code>	First iterator.
<code>__b2</code>	Second iterator.

Returns

`true` if less.

Definition at line 120 of file `multiway_merge.h`.

4.108.4.2 `template<typename _RAIter, typename _Compare > bool operator<= (_GuardedIterator< _RAIter, _Compare > & __bi1, _GuardedIterator< _RAIter, _Compare > & __bi2) [friend]`

Compare two elements referenced by guarded iterators.

Parameters

<code>__bi1</code>	First iterator.
<code>__bi2</code>	Second iterator.

Returns

`True` if less equal.

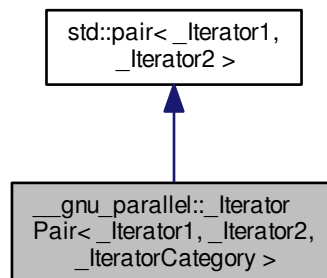
Definition at line 135 of file `multiway_merge.h`.

The documentation for this class was generated from the following file:

- [multiway_merge.h](#)

4.109 `__gnu_parallel::_IteratorPair< _Iterator1, _Iterator2, _IteratorCategory >` Class Template Reference

Inheritance diagram for `__gnu_parallel::_IteratorPair< _Iterator1, _Iterator2, _IteratorCategory >`:



Public Types

- `typedef std::iterator_traits<_Iterator1> _TraitsType`
- `typedef _TraitsType::difference_type difference_type`
- `typedef _Iterator1 first_type`
- `typedef _IteratorCategory iterator_category`
- `typedef _IteratorPair * pointer`
- `typedef _IteratorPair & reference`
- `typedef _Iterator2 second_type`
- `typedef void value_type`

Public Member Functions

- `_IteratorPair (const _Iterator1 &__first, const _Iterator2 &__second)`
- `operator _Iterator2 () const`
- `_IteratorPair operator+ (difference_type __delta) const`
- `_IteratorPair & operator++ ()`
- `const _IteratorPair operator++ (int)`
- `difference_type operator- (const _IteratorPair &__other) const`
- `_IteratorPair & operator-- ()`
- `const _IteratorPair operator-- (int)`
- `_IteratorPair & operator= (const _IteratorPair &__other)`
- `void swap (pair &__p) noexcept(noexcept(swap(first, __p.first)) &&noexcept(swap(second, __p.second)))`

Public Attributes

- `_Iterator1 first`
- `_Iterator2 second`

4.109.1 Detailed Description

```
template<typename _Iterator1, typename _Iterator2, typename _IteratorCategory>
class __gnu_parallel::_IteratorPair<_Iterator1, _Iterator2, _IteratorCategory>
```

A pair of iterators. The usual iterator operations are applied to both child iterators.

Definition at line 45 of file `iterator.h`.

4.109.2 Member Typedef Documentation

4.109.2.1 `typedef _Iterator2 std::pair<_Iterator1, _Iterator2>::second_type` [inherited]

`first_type` is the first bound type

Definition at line 99 of file `stl_pair.h`.

4.109.3 Member Data Documentation

4.109.3.1 `_Iterator1 std::pair<_Iterator1, _Iterator2>::first` [inherited]

`second_type` is the second bound type

Definition at line 101 of file `stl_pair.h`.

4.109.3.2 `_Iterator2 std::pair<_Iterator1, _Iterator2>::second` [inherited]

`first` is a copy of the first object

Definition at line 102 of file `stl_pair.h`.

The documentation for this class was generated from the following file:

- [iterator.h](#)

4.110 `__gnu_parallel::_IteratorTriple<_Iterator1, _Iterator2, _Iterator3, _IteratorCategory>` Class Template Reference

Public Types

- `typedef std::iterator_traits<_Iterator1>::difference_type difference_type`
- `typedef _IteratorCategory iterator_category`
- `typedef _IteratorTriple * pointer`
- `typedef _IteratorTriple & reference`
- `typedef void value_type`

Public Member Functions

- `_IteratorTriple (const _Iterator1 &__first, const _Iterator2 &__second, const _Iterator3 &__third)`
- `operator _Iterator3 () const`
- `_IteratorTriple operator+ (difference_type __delta) const`
- `_IteratorTriple & operator++ ()`
- `const _IteratorTriple operator++ (int)`
- `difference_type operator- (const _IteratorTriple &__other) const`
- `_IteratorTriple & operator-- ()`
- `const _IteratorTriple operator-- (int)`
- `_IteratorTriple & operator= (const _IteratorTriple &__other)`

Public Attributes

- `_Iterator1 _M_first`
- `_Iterator2 _M_second`
- `_Iterator3 _M_third`

4.110.1 Detailed Description

```
template<typename _Iterator1, typename _Iterator2, typename _Iterator3, typename _IteratorCategory>
class __gnu_parallel::IteratorTriple<_Iterator1, _Iterator2, _Iterator3, _IteratorCategory>
```

A triple of iterators. The usual iterator operations are applied to all three child iterators.

Definition at line 120 of file `iterator.h`.

The documentation for this class was generated from the following file:

- [iterator.h](#)

4.111 `__gnu_parallel::__Job<_DifferenceTp>` Struct Template Reference

Public Types

- typedef `_DifferenceTp` **`_DifferenceType`**

Public Attributes

- volatile `_DifferenceType` [_M_first](#)
- volatile `_DifferenceType` [_M_last](#)
- volatile `_DifferenceType` [_M_load](#)

4.111.1 Detailed Description

```
template<typename _DifferenceTp>
struct __gnu_parallel::__Job<_DifferenceTp>
```

One `__job` for a certain thread.

Definition at line 54 of file `workstealing.h`.

4.111.2 Member Data Documentation

4.111.2.1 `template<typename _DifferenceTp> volatile _DifferenceType __gnu_parallel::__Job<_DifferenceTp>::__M_first`

First element.

Changed by owning and stealing thread. By stealing thread, always incremented.

Definition at line 62 of file `workstealing.h`.

Referenced by `__gnu_parallel::__for_each_template_random_access_workstealing()`.

4.111.2.2 `template<typename _DifferenceTp> volatile _DifferenceType __gnu_parallel::_Job<_DifferenceTp>::_M_last`

Last element.

Changed by owning thread only.

Definition at line 67 of file `workstealing.h`.

Referenced by `__gnu_parallel::_for_each_template_random_access_workstealing()`.

4.111.2.3 `template<typename _DifferenceTp> volatile _DifferenceType __gnu_parallel::_Job<_DifferenceTp>::_M_load`

Number of elements, i.e. `_M_last-_M_first+1`.

Changed by owning thread only.

Definition at line 72 of file `workstealing.h`.

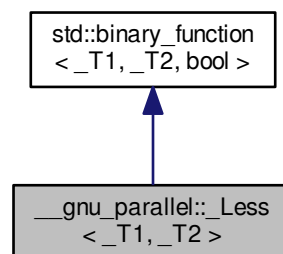
Referenced by `__gnu_parallel::_for_each_template_random_access_workstealing()`.

The documentation for this struct was generated from the following file:

- [workstealing.h](#)

4.112 `__gnu_parallel::_Less<_T1, _T2>` Struct Template Reference

Inheritance diagram for `__gnu_parallel::_Less<_T1, _T2>`:



Public Types

- typedef `_T1` [first_argument_type](#)
- typedef `bool` [result_type](#)
- typedef `_T2` [second_argument_type](#)

Public Member Functions

- `bool operator() (const _T1 &__t1, const _T2 &__t2) const`
- `bool operator() (const _T2 &__t2, const _T1 &__t1) const`

4.112.1 Detailed Description

```
template<typename _T1, typename _T2>
struct __gnu_parallel::_Less<_T1, _T2>
```

Similar to `std::less`, but allows two different types.

Definition at line 252 of file `parallel/base.h`.

4.112.2 Member Typedef Documentation

4.112.2.1 `typedef _T1 std::binary_function<_T1, _T2, bool >::first_argument_type` [inherited]

`first_argument_type` is the type of the first argument

Definition at line 121 of file `stl_function.h`.

4.112.2.2 `typedef bool std::binary_function<_T1, _T2, bool >::result_type` [inherited]

`result_type` is the return type

Definition at line 127 of file `stl_function.h`.

4.112.2.3 `typedef _T2 std::binary_function<_T1, _T2, bool >::second_argument_type` [inherited]

`second_argument_type` is the type of the second argument

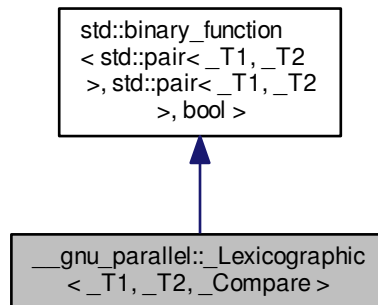
Definition at line 124 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [parallel/base.h](#)

4.113 `__gnu_parallel::_Lexicographic<_T1, _T2, _Compare >` Class Template Reference

Inheritance diagram for `__gnu_parallel::_Lexicographic<_T1, _T2, _Compare >`:



Public Types

- typedef `std::pair<_T1, _T2 >` `first_argument_type`
- typedef `bool` `result_type`
- typedef `std::pair<_T1, _T2 >` `second_argument_type`

Public Member Functions

- `_Lexicographic` (`_Compare &__comp`)
- `bool operator()` (`const std::pair<_T1, _T2 > &__p1, const std::pair<_T1, _T2 > &__p2`) `const`

4.113.1 Detailed Description

```
template<typename _T1, typename _T2, typename _Compare>
class __gnu_parallel::_Lexicographic<_T1, _T2, _Compare >
```

Compare __a pair of types lexicographically, ascending.

Definition at line 53 of file `multiseq_selection.h`.

4.113.2 Member Typedef Documentation

4.113.2.1 typedef `std::pair<_T1, _T2 >` `std::binary_function<std::pair<_T1, _T2 >, std::pair<_T1, _T2 >, bool>::first_argument_type` [inherited]

`first_argument_type` is the type of the first argument

Definition at line 121 of file `stl_function.h`.

4.113.2.2 `typedef bool std::binary_function< std::pair<_T1, _T2>, std::pair<_T1, _T2>, bool>::result_type`
`[inherited]`

`result_type` is the return type

Definition at line 127 of file `stl_function.h`.

4.113.2.3 `typedef std::pair<_T1, _T2> std::binary_function< std::pair<_T1, _T2>, std::pair<_T1, _T2>, bool>::second_argument_type`
`[inherited]`

`second_argument_type` is the type of the second argument

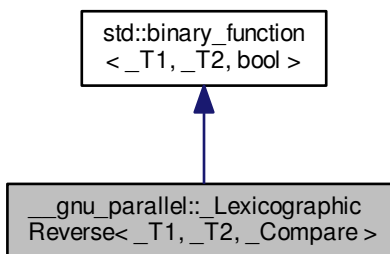
Definition at line 124 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [multiseq_selection.h](#)

4.114 `__gnu_parallel::_LexicographicReverse<_T1, _T2, _Compare>` Class Template Reference

Inheritance diagram for `__gnu_parallel::_LexicographicReverse<_T1, _T2, _Compare>`:



Public Types

- `typedef _T1 first_argument_type`
- `typedef bool result_type`
- `typedef _T2 second_argument_type`

Public Member Functions

- `_LexicographicReverse` (`_Compare` &`__comp`)
- `bool operator()` (const `std::pair<_T1, _T2>` &`__p1`, const `std::pair<_T1, _T2>` &`__p2`) const

4.114.1 Detailed Description

```
template<typename _T1, typename _T2, typename _Compare>
class __gnu_parallel::_LexicographicReverse< _T1, _T2, _Compare >
```

Compare __a pair of types lexicographically, descending.

Definition at line 80 of file multiseq_selection.h.

4.114.2 Member Typedef Documentation

4.114.2.1 `typedef _T1 std::binary_function< _T1, _T2, bool >::first_argument_type` [inherited]

`first_argument_type` is the type of the first argument

Definition at line 121 of file stl_function.h.

4.114.2.2 `typedef bool std::binary_function< _T1, _T2, bool >::result_type` [inherited]

`result_type` is the return type

Definition at line 127 of file stl_function.h.

4.114.2.3 `typedef _T2 std::binary_function< _T1, _T2, bool >::second_argument_type` [inherited]

`second_argument_type` is the type of the second argument

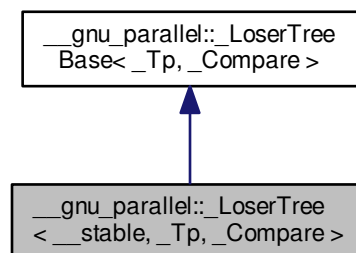
Definition at line 124 of file stl_function.h.

The documentation for this class was generated from the following file:

- [multiseq_selection.h](#)

4.115 `__gnu_parallel::_LoserTree< __stable, _Tp, _Compare >` Class Template Reference

Inheritance diagram for `__gnu_parallel::_LoserTree< __stable, _Tp, _Compare >`:



Public Member Functions

- `_LoserTree` (unsigned int __k, _Compare __comp)
- void `__delete_min_insert` (_Tp __key, bool __sup)
- int `__get_min_source` ()
- void `__init` ()
- unsigned int `__init_winner` (unsigned int __root)
- void `__insert_start` (const _Tp &__key, int __source, bool __sup)

Protected Attributes

- _Compare `_M_comp`
- bool `_M_first_insert`
- unsigned int `_M_ik`
- unsigned int `_M_k`
- unsigned int `_M_log_k`
- `_Loser` * `_M_losers`
- unsigned int `_M_offset`

4.115.1 Detailed Description

```
template<bool __stable, typename _Tp, typename _Compare>
class __gnu_parallel::_LoserTree< __stable, _Tp, _Compare >
```

Stable `_LoserTree` variant.

Provides the stable implementations of `insert_start`, `__init_winner`, `__init` and `__delete_min_insert`.

Unstable variant is done using partial specialisation below.

Definition at line 169 of file `losertree.h`.

4.115.2 Member Function Documentation

4.115.2.1 `template<bool __stable, typename _Tp, typename _Compare > void __gnu_parallel::_LoserTree< __stable, _Tp, _Compare >::__delete_min_insert (_Tp __key, bool __sup) [inline]`

Delete the smallest element and insert a new element from the previously smallest element's sequence.

This implementation is stable.

Definition at line 222 of file `losertree.h`.

References `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_comp`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_key`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_losers`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_source`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_sup`, and `std::swap()`.

4.115.2.2 `template<typename _Tp, typename _Compare > int __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__get_min_source() [inline],[inherited]`

Returns

the index of the sequence with the smallest element.

Definition at line 155 of file losertree.h.

References `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__Loser::__M_source`.

4.115.2.3 `template<typename _Tp, typename _Compare > void __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__insert_start(const _Tp & __key, int __source, bool __sup) [inline],[inherited]`

Initializes the sequence "`_M_source`" with the element "`__key`".

Parameters

<code>__key</code>	the element to insert
<code>__source</code>	<code>__index</code> of the <code>__source</code> sequence
<code>__sup</code>	flag that determines whether the value to insert is an explicit <code>__supremum</code> .

Definition at line 134 of file losertree.h.

References `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__Loser::__M_key`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__Loser::__M_source`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__Loser::__M_sup`.

4.115.3 Member Data Documentation

4.115.3.1 `template<typename _Tp, typename _Compare > _Compare __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__M_comp [protected],[inherited]`

`_Compare` to use.

Definition at line 78 of file losertree.h.

Referenced by `__gnu_parallel::_LoserTree< __stable, _Tp, _Compare >::__delete_min_insert()`, `__gnu_parallel::_LoserTree< false, _Tp, _Compare >::__delete_min_insert()`, and `__gnu_parallel::_LoserTree< false, _Tp, _Compare >::__init_winner()`.

4.115.3.2 `template<typename _Tp, typename _Compare > bool __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__M_first_insert [protected],[inherited]`

State flag that determines whether the `_LoserTree` is empty.

Only used for building the `_LoserTree`.

Definition at line 85 of file losertree.h.

4.115.3.3 `template<typename _Tp, typename _Compare > unsigned int __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_log_k` `[protected]`, `[inherited]`

`log_2{_M_k}`

Definition at line 72 of file `losertree.h`.

Referenced by `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_LoserTreeBase()`.

4.115.3.4 `template<typename _Tp, typename _Compare > _Loser* __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_losers` `[protected]`, `[inherited]`

`_LoserTree` `__elements`.

Definition at line 75 of file `losertree.h`.

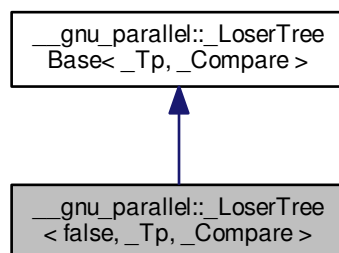
Referenced by `__gnu_parallel::_LoserTree< __stable, _Tp, _Compare >::_delete_min_insert()`, `__gnu_parallel::_LoserTree< false, _Tp, _Compare >::_delete_min_insert()`, `__gnu_parallel::_LoserTree< false, _Tp, _Compare >::_init_winner()`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::~~LoserTreeBase()`.

The documentation for this class was generated from the following file:

- [losertree.h](#)

4.116 `__gnu_parallel::_LoserTree< false, _Tp, _Compare >` Class Template Reference

Inheritance diagram for `__gnu_parallel::_LoserTree< false, _Tp, _Compare >`:



Public Member Functions

- **`_LoserTree`** (`unsigned int __k, _Compare __comp`)
- `void __delete_min_insert` (`_Tp __key, bool __sup`)
- `int __get_min_source` ()
- `void __init` ()
- `unsigned int __init_winner` (`unsigned int __root`)
- `void __insert_start` (`const _Tp &__key, int __source, bool __sup`)

Protected Attributes

- [_Compare](#) [_M_comp](#)
- [bool](#) [_M_first_insert](#)
- [unsigned int](#) [_M_ik](#)
- [unsigned int](#) [_M_k](#)
- [unsigned int](#) [_M_log_k](#)
- [_Loser](#) * [_M_losers](#)
- [unsigned int](#) [_M_offset](#)

4.116.1 Detailed Description

```
template<typename _Tp, typename _Compare>
class __gnu_parallel::LoserTree< false, _Tp, _Compare >
```

Unstable `_LoserTree` variant.

Stability (non-stable here) is selected with partial specialization.

Definition at line 261 of file `losertree.h`.

4.116.2 Member Function Documentation

4.116.2.1 `template<typename _Tp, typename _Compare > void __gnu_parallel::LoserTree< false, _Tp, _Compare >::__delete_min_insert(_Tp __key, bool __sup) [inline]`

Delete the `_M_key` smallest element and insert the element `__key` instead.

Parameters

<code>__key</code>	the <code>_M_key</code> to insert
<code>__sup</code>	true iff <code>__key</code> is an explicitly marked supremum

Definition at line 324 of file `losertree.h`.

References `__gnu_parallel::LoserTreeBase< _Tp, _Compare >::__M_comp`, `__gnu_parallel::LoserTreeBase< _Tp, _Compare >::__Loser::__M_key`, `__gnu_parallel::LoserTreeBase< _Tp, _Compare >::__M_losers`, `__gnu_parallel::LoserTreeBase< _Tp, _Compare >::__Loser::__M_source`, `__gnu_parallel::LoserTreeBase< _Tp, _Compare >::__Loser::__M_sup`, and `std::swap()`.

4.116.2.2 `template<typename _Tp, typename _Compare > int __gnu_parallel::LoserTreeBase< _Tp, _Compare >::__get_min_source() [inline],[inherited]`

Returns

the index of the sequence with the smallest element.

Definition at line 155 of file `losertree.h`.

References `__gnu_parallel::LoserTreeBase< _Tp, _Compare >::__Loser::__M_source`.

4.116.2.3 `template<typename _Tp, typename _Compare > unsigned int __gnu_parallel::_LoserTree< false, _Tp, _Compare >::_init_winner(unsigned int __root) [inline]`

Computes the winner of the competition at position "`__root`".

Called recursively (starting at 0) to build the initial tree.

Parameters

<code>__root</code>	__index of the "game" to start.
---------------------	---------------------------------

Definition at line 284 of file `losertree.h`.

References `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_comp`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_key`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_losers`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_sup`.

4.116.2.4 `template<typename _Tp, typename _Compare > void __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_insert_start(const _Tp & __key, int __source, bool __sup) [inline],[inherited]`

Initializes the sequence "`_M_source`" with the element "`__key`".

Parameters

<code>__key</code>	the element to insert
<code>__source</code>	__index of the <code>__source</code> sequence
<code>__sup</code>	flag that determines whether the value to insert is an explicit <code>__supremum</code> .

Definition at line 134 of file `losertree.h`.

References `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_key`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_source`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_sup`.

4.116.3 Member Data Documentation

4.116.3.1 `template<typename _Tp, typename _Compare > _Compare __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_comp [protected],[inherited]`

`_Compare` to use.

Definition at line 78 of file `losertree.h`.

Referenced by `__gnu_parallel::_LoserTree< __stable, _Tp, _Compare >::_delete_min_insert()`, `__delete_min_insert()`, and `__init_winner()`.

4.116.3.2 `template<typename _Tp, typename _Compare > bool __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_first_insert [protected], [inherited]`

State flag that determines whether the `_LoserTree` is empty.

Only used for building the `_LoserTree`.

Definition at line 85 of file `losertree.h`.

4.116.3.3 `template<typename _Tp, typename _Compare > unsigned int __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_log_k [protected], [inherited]`

`log_2{ _M_k }`

Definition at line 72 of file `losertree.h`.

Referenced by `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_LoserTreeBase()`.

4.116.3.4 `template<typename _Tp, typename _Compare > _Loser* __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_losers [protected], [inherited]`

`_LoserTree __elements`.

Definition at line 75 of file `losertree.h`.

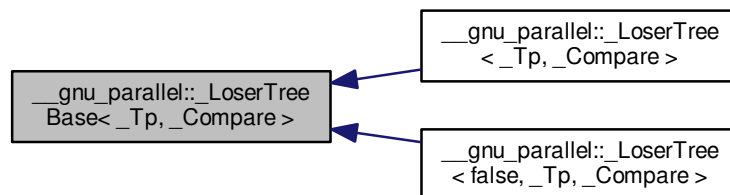
Referenced by `__gnu_parallel::_LoserTree< __stable, _Tp, _Compare >::_delete_min_insert()`, `__delete_min_↵insert()`, `__init_winner()`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::~~LoserTreeBase()`.

The documentation for this class was generated from the following file:

- [losertree.h](#)

4.117 `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >` Class Template Reference

Inheritance diagram for `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >`:



Classes

- struct `_Loser`

Public Member Functions

- `_LoserTreeBase` (unsigned int `__k`, `_Compare` `__comp`)
- `~_LoserTreeBase` ()
- int `__get_min_source` ()
- void `__insert_start` (const `_Tp` &`__key`, int `__source`, bool `__sup`)

Protected Attributes

- `_Compare` `_M_comp`
- bool `_M_first_insert`
- unsigned int `_M_ik`
- unsigned int `_M_k`
- unsigned int `_M_log_k`
- `_Loser` * `_M_losers`
- unsigned int `_M_offset`

4.117.1 Detailed Description

```
template<typename _Tp, typename _Compare>
class __gnu_parallel::_LoserTreeBase<_Tp, _Compare>
```

Guarded loser/tournament tree.

The smallest element is at the top.

Guarding is done explicitly through one flag `_M_sup` per element, `inf` is not needed due to a better initialization routine. This is a well-performing variant.

Parameters

<code>_Tp</code>	the element type
<code>_Compare</code>	the comparator to use, defaults to <code>std::less<_Tp></code>

Definition at line 55 of file `losertree.h`.

4.117.2 Constructor & Destructor Documentation

```
4.117.2.1 template<typename _Tp, typename _Compare> __gnu_parallel::_LoserTreeBase<_Tp, _Compare>
         >::_LoserTreeBase ( unsigned int __k, _Compare __comp ) [inline]
```

The constructor.

Parameters

<code>__k</code>	The number of sequences to merge.
<code>__comp</code>	The comparator to use.

Definition at line 94 of file `losertree.h`.

References `__gnu_parallel::__rd_log2()`, `__gnu_parallel::_LoserTreeBase<_Tp, _Compare>::_M_log_k`, and `__gnu_parallel::_LoserTreeBase<_Tp, _Compare>::_Loser::_M_sup`.

4.117.2.2 `template<typename _Tp, typename _Compare> __gnu_parallel::_LoserTreeBase<_Tp, _Compare>::~~_LoserTreeBase() [inline]`

The destructor.

Definition at line 118 of file `losertree.h`.

References `__gnu_parallel::_LoserTreeBase<_Tp, _Compare>::_M_losers`.

4.117.3 Member Function Documentation

4.117.3.1 `template<typename _Tp, typename _Compare> int __gnu_parallel::_LoserTreeBase<_Tp, _Compare>::_get_min_source() [inline]`

Returns

the index of the sequence with the smallest element.

Definition at line 155 of file `losertree.h`.

References `__gnu_parallel::_LoserTreeBase<_Tp, _Compare>::_Loser::_M_source`.

4.117.3.2 `template<typename _Tp, typename _Compare> void __gnu_parallel::_LoserTreeBase<_Tp, _Compare>::_insert_start(const _Tp & __key, int __source, bool __sup) [inline]`

Initializes the sequence "`_M_source`" with the element "`__key`".

Parameters

<code>__key</code>	the element to insert
<code>__source</code>	<code>__index</code> of the <code>__source</code> sequence
<code>__sup</code>	flag that determines whether the value to insert is an explicit <code>__supremum</code> .

Definition at line 134 of file `losertree.h`.

References `__gnu_parallel::_LoserTreeBase<_Tp, _Compare>::_Loser::_M_key`, `__gnu_parallel::_LoserTreeBase<_Tp, _Compare>::_Loser::_M_source`, and `__gnu_parallel::_LoserTreeBase<_Tp, _Compare>::_Loser::_M_sup`.

4.117.4 Member Data Documentation

4.117.4.1 `template<typename _Tp, typename _Compare> _Compare __gnu_parallel::_LoserTreeBase<_Tp, _Compare>::__M_comp` `[protected]`

`_Compare` to use.

Definition at line 78 of file `losertree.h`.

Referenced by `__gnu_parallel::_LoserTree<__stable, _Tp, _Compare>::__delete_min_insert()`, `__gnu_parallel::_LoserTree<false, _Tp, _Compare>::__delete_min_insert()`, and `__gnu_parallel::_LoserTree<false, _Tp, _Compare>::__init_winner()`.

4.117.4.2 `template<typename _Tp, typename _Compare> bool __gnu_parallel::_LoserTreeBase<_Tp, _Compare>::__M_first_insert` `[protected]`

State flag that determines whether the `_LoserTree` is empty.

Only used for building the `_LoserTree`.

Definition at line 85 of file `losertree.h`.

4.117.4.3 `template<typename _Tp, typename _Compare> unsigned int __gnu_parallel::_LoserTreeBase<_Tp, _Compare>::__M_log_k` `[protected]`

`log_2{M_k}`

Definition at line 72 of file `losertree.h`.

Referenced by `__gnu_parallel::_LoserTreeBase<_Tp, _Compare>::_LoserTreeBase()`.

4.117.4.4 `template<typename _Tp, typename _Compare> _Loser* __gnu_parallel::_LoserTreeBase<_Tp, _Compare>::__M_losers` `[protected]`

`_LoserTree` `__elements`.

Definition at line 75 of file `losertree.h`.

Referenced by `__gnu_parallel::_LoserTree<__stable, _Tp, _Compare>::__delete_min_insert()`, `__gnu_parallel::_LoserTree<false, _Tp, _Compare>::__delete_min_insert()`, `__gnu_parallel::_LoserTree<false, _Tp, _Compare>::__init_winner()`, and `__gnu_parallel::_LoserTreeBase<_Tp, _Compare>::~~LoserTreeBase()`.

The documentation for this class was generated from the following file:

- [losertree.h](#)

4.118 `__gnu_parallel::_LoserTreeBase<_Tp, _Compare>::_Loser` Struct Reference

Public Attributes

- `_Tp _M_key`
- `int _M_source`
- `bool _M_sup`

4.118.1 Detailed Description

```
template<typename _Tp, typename _Compare>
struct __gnu_parallel::_LoserTreeBase<_Tp, _Compare>::_Loser
```

Internal representation of a `_LoserTree` element.

Definition at line 59 of file `losertree.h`.

4.118.2 Member Data Documentation

4.118.2.1 `template<typename _Tp, typename _Compare> _Tp __gnu_parallel::_LoserTreeBase<_Tp, _Compare>::_Loser::_M_key`

`_M_key` of the element in the `_LoserTree`.

Definition at line 66 of file `losertree.h`.

Referenced by `__gnu_parallel::_LoserTree<__stable, _Tp, _Compare>::_delete_min_insert()`, `__gnu_parallel::_LoserTree<false, _Tp, _Compare>::_delete_min_insert()`, `__gnu_parallel::_LoserTree<false, _Tp, _Compare>::_init_winner()`, and `__gnu_parallel::_LoserTreeBase<_Tp, _Compare>::_insert_start()`.

4.118.2.2 `template<typename _Tp, typename _Compare> int __gnu_parallel::_LoserTreeBase<_Tp, _Compare>::_Loser::_M_source`

`__index` of the `__source` sequence.

Definition at line 64 of file `losertree.h`.

Referenced by `__gnu_parallel::_LoserTree<__stable, _Tp, _Compare>::_delete_min_insert()`, `__gnu_parallel::_LoserTree<false, _Tp, _Compare>::_delete_min_insert()`, `__gnu_parallel::_LoserTreeBase<_Tp, _Compare>::_get_min_source()`, and `__gnu_parallel::_LoserTreeBase<_Tp, _Compare>::_insert_start()`.

4.118.2.3 `template<typename _Tp, typename _Compare > bool __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_sup`

flag, true iff this is a "maximum" `__sentinel`.

Definition at line 62 of file `losertree.h`.

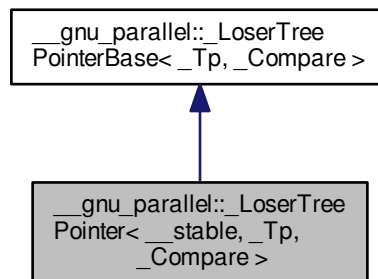
Referenced by `__gnu_parallel::_LoserTree< __stable, _Tp, _Compare >::_delete_min_insert()`, `__gnu_parallel::_LoserTree< false, _Tp, _Compare >::_delete_min_insert()`, `__gnu_parallel::_LoserTree< false, _Tp, _Compare >::_init_winner()`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_insert_start()`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_LoserTreeBase()`.

The documentation for this struct was generated from the following file:

- [losertree.h](#)

4.119 `__gnu_parallel::_LoserTreePointer< __stable, _Tp, _Compare >` Class Template Reference

Inheritance diagram for `__gnu_parallel::_LoserTreePointer< __stable, _Tp, _Compare >`:



Public Member Functions

- **`_LoserTreePointer`** (unsigned int `__k`, `_Compare` `__comp=std::less<_Tp>()`)
- void **`__delete_min_insert`** (const `_Tp` &`__key`, bool `__sup`)
- int **`__get_min_source`** ()
- void **`__init`** ()
- unsigned int **`__init_winner`** (unsigned int `__root`)
- void **`__insert_start`** (const `_Tp` &`__key`, int `__source`, bool `__sup`)

Protected Attributes

- `_Compare` **`_M_comp`**
- `unsigned int` **`_M_ik`**
- `unsigned int` **`_M_k`**
- `_Loser` * **`_M_losers`**
- `unsigned int` **`_M_offset`**

4.119.1 Detailed Description

```
template<bool __stable, typename _Tp, typename _Compare>
class __gnu_parallel::_LoserTreePointer< __stable, _Tp, _Compare >
```

Stable `_LoserTree` implementation.

The unstable variant is implemented using partial instantiation below.

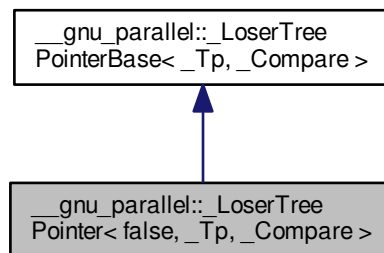
Definition at line 409 of file `losertree.h`.

The documentation for this class was generated from the following file:

- [losertree.h](#)

4.120 `__gnu_parallel::_LoserTreePointer< false, _Tp, _Compare >` Class Template Reference

Inheritance diagram for `__gnu_parallel::_LoserTreePointer< false, _Tp, _Compare >`:



Public Member Functions

- **`_LoserTreePointer`** (`unsigned int` `__k`, `_Compare` `__comp`=`std::less< _Tp >()`)
- `void` **`__delete_min_insert`** (`const _Tp` & `__key`, `bool` `__sup`)
- `int` **`__get_min_source`** ()
- `void` **`__init`** ()
- `unsigned int` **`__init_winner`** (`unsigned int` `__root`)
- `void` **`__insert_start`** (`const _Tp` & `__key`, `int` `__source`, `bool` `__sup`)

Protected Attributes

- `_Compare` **`_M_comp`**
- unsigned int **`_M_ik`**
- unsigned int **`_M_k`**
- [_Loser](#) * **`_M_losers`**
- unsigned int **`_M_offset`**

4.120.1 Detailed Description

```
template<typename _Tp, typename _Compare>
class __gnu_parallel::_LoserTreePointer< false, _Tp, _Compare >
```

Unstable `_LoserTree` implementation.

The stable variant is above.

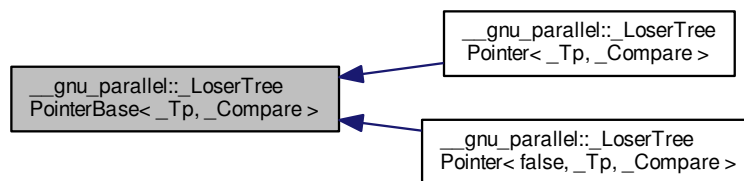
Definition at line 491 of file `losertree.h`.

The documentation for this class was generated from the following file:

- [losertree.h](#)

4.121 `__gnu_parallel::_LoserTreePointerBase<_Tp, _Compare>` Class Template Reference

Inheritance diagram for `__gnu_parallel::_LoserTreePointerBase<_Tp, _Compare>`:



Classes

- struct [_Loser](#)

Public Member Functions

- **_LoserTreePointerBase** (unsigned int __k, _Compare __comp=[std::less](#)< _Tp >())
- int **__get_min_source** ()
- void **__insert_start** (const _Tp &__key, int __source, bool __sup)

Protected Attributes

- _Compare **_M_comp**
- unsigned int **_M_ik**
- unsigned int **_M_k**
- [_Loser](#) * **_M_losers**
- unsigned int **_M_offset**

4.121.1 Detailed Description

```
template<typename _Tp, typename _Compare>
class __gnu_parallel::_LoserTreePointerBase< _Tp, _Compare >
```

Base class of _Loser Tree implementation using pointers.

Definition at line 357 of file losertree.h.

The documentation for this class was generated from the following file:

- [losertree.h](#)

4.122 __gnu_parallel::_LoserTreePointerBase< _Tp, _Compare >::_Loser Struct Reference

Public Attributes

- const _Tp * **_M_keyp**
- int **_M_source**
- bool **_M_sup**

4.122.1 Detailed Description

```
template<typename _Tp, typename _Compare>
struct __gnu_parallel::_LoserTreePointerBase< _Tp, _Compare >::_Loser
```

Internal representation of _LoserTree __elements.

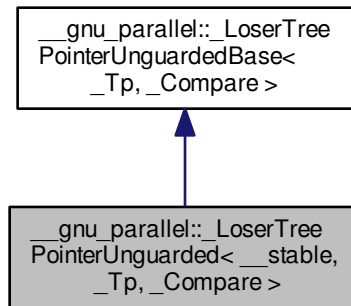
Definition at line 361 of file losertree.h.

The documentation for this struct was generated from the following file:

- [losertree.h](#)

4.123 `__gnu_parallel::_LoserTreePointerUnguarded< __stable, _Tp, _Compare >` Class Template Reference

Inheritance diagram for `__gnu_parallel::_LoserTreePointerUnguarded< __stable, _Tp, _Compare >`:



Public Member Functions

- **`_LoserTreePointerUnguarded`** (unsigned int `__k`, const `_Tp` & `__sentinel`, `_Compare` `__comp`=`std::less`< `_Tp` >())
- void **`__delete_min_insert`** (const `_Tp` & `__key`, bool `__sup`)
- int **`__get_min_source`** ()
- void **`__init`** ()
- unsigned int **`__init_winner`** (unsigned int `__root`)
- void **`__insert_start`** (const `_Tp` & `__key`, int `__source`, bool)

Protected Attributes

- `_Compare` **`_M_comp`**
- unsigned int **`_M_ik`**
- unsigned int **`_M_k`**
- `_Loser` * **`_M_losers`**
- unsigned int **`_M_offset`**

4.123.1 Detailed Description

```

template<bool __stable, typename _Tp, typename _Compare>
class __gnu_parallel::_LoserTreePointerUnguarded< __stable, _Tp, _Compare >

```

Stable unguarded `_LoserTree` variant storing pointers.

Unstable variant is implemented below using partial specialization.

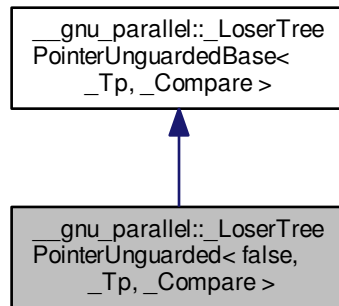
Definition at line 891 of file `losertree.h`.

The documentation for this class was generated from the following file:

- [losertree.h](#)

4.124 `__gnu_parallel::_LoserTreePointerUnguarded< false, _Tp, _Compare >` Class Template Reference

Inheritance diagram for `__gnu_parallel::_LoserTreePointerUnguarded< false, _Tp, _Compare >`:



Public Member Functions

- **`_LoserTreePointerUnguarded`** (unsigned int __k, const _Tp &__sentinel, _Compare __comp=[std::less](#)< _Tp >())
- void **`__delete_min_insert`** (const _Tp &__key, bool __sup)
- int **`__get_min_source`** ()
- void **`__init`** ()
- unsigned int **`__init_winner`** (unsigned int __root)
- void **`__insert_start`** (const _Tp &__key, int __source, bool)

Protected Attributes

- `_Compare` **`_M_comp`**
- unsigned int **`_M_ik`**
- unsigned int **`_M_k`**
- `_Loser *` **`_M_losers`**
- unsigned int **`_M_offset`**

4.124.1 Detailed Description

```

template<typename _Tp, typename _Compare>
class __gnu_parallel::_LoserTreePointerUnguarded< false, _Tp, _Compare >

```

Unstable unguarded `_LoserTree` variant storing pointers.

Stable variant is above.

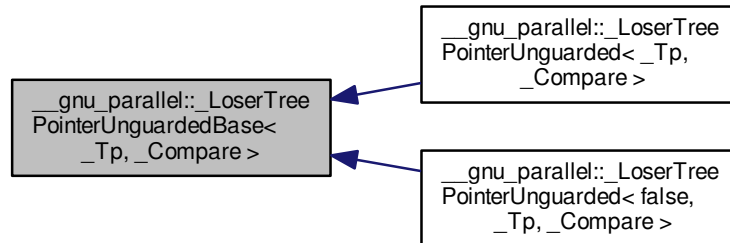
Definition at line 977 of file `losertree.h`.

The documentation for this class was generated from the following file:

- [losertree.h](#)

4.125 `__gnu_parallel::_LoserTreePointerUnguardedBase<_Tp, _Compare>` Class Template Reference

Inheritance diagram for `__gnu_parallel::_LoserTreePointerUnguardedBase<_Tp, _Compare>`:



Public Member Functions

- **`_LoserTreePointerUnguardedBase`** (unsigned int __k, const _Tp &__sentinel, _Compare __comp=[std::less<_Tp>](#)())
- int **`__get_min_source`** ()
- void **`__insert_start`** (const _Tp &__key, int __source, bool)

Protected Attributes

- _Compare **`_M_comp`**
- unsigned int **`_M_ik`**
- unsigned int **`_M_k`**
- _Loser * **`_M_losers`**
- unsigned int **`_M_offset`**

4.125.1 Detailed Description

```

template<typename _Tp, typename _Compare>
class __gnu_parallel::_LoserTreePointerUnguardedBase<_Tp, _Compare>

```

Unguarded loser tree, keeping only pointers to the elements in the tree structure.

No guarding is done, therefore not a single input sequence must run empty. This is a very fast variant.

Definition at line 828 of file `losertree.h`.

The documentation for this class was generated from the following file:

- [losertree.h](#)

4.126 `__gnu_parallel::_LoserTreeTraits<_Tp>` Struct Template Reference

Static Public Attributes

- static const bool [_M_use_pointer](#)

4.126.1 Detailed Description

```
template<typename _Tp>
struct __gnu_parallel::_LoserTreeTraits<_Tp>
```

Traits for determining whether the loser tree should use pointers or copies.

The field "`_M_use_pointer`" is used to determine whether to use pointers in the loser trees or whether to copy the values into the loser tree.

The default behavior is to use pointers if the data type is 4 times as big as the pointer to it.

Specialize for your data type to customize the behavior.

Example:

```
template<> struct _LoserTreeTraits<int> { static const bool _M_use_pointer = false; };

template<> struct _LoserTreeTraits<heavyweight_type> { static const bool _M_use_pointer = true; };
```

Parameters

<code>_Tp</code>	type to give the loser tree traits for.
------------------	---

Definition at line 731 of file `multiway_merge.h`.

4.126.2 Member Data Documentation

4.126.2.1 `template<typename _Tp> const bool __gnu_parallel::_LoserTreeTraits<_Tp>::_M_use_pointer` `[static]`

True iff to use pointers instead of values in loser trees.

The default behavior is to use pointers if the data type is four times as big as the pointer to it.

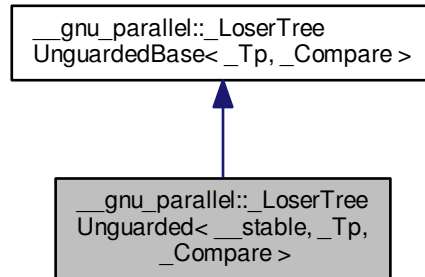
Definition at line 739 of file `multiway_merge.h`.

The documentation for this struct was generated from the following file:

- [multiway_merge.h](#)

4.127 `__gnu_parallel::_LoserTreeUnguarded< __stable, _Tp, _Compare >` Class Template Reference

Inheritance diagram for `__gnu_parallel::_LoserTreeUnguarded< __stable, _Tp, _Compare >`:



Public Member Functions

- **`_LoserTreeUnguarded`** (unsigned int __k, const _Tp &__sentinel, _Compare __comp=[std::less](#)< _Tp >())
- void **`_delete_min_insert`** (_Tp __key, bool)
- int **`_get_min_source`** ()
- void **`_init`** ()
- unsigned int **`_init_winner`** (unsigned int __root)
- void **`_insert_start`** (const _Tp &__key, int __source, bool)

Protected Attributes

- _Compare **`_M_comp`**
- unsigned int **`_M_ik`**
- unsigned int **`_M_k`**
- _Loser * **`_M_losers`**
- unsigned int **`_M_offset`**

4.127.1 Detailed Description

```
template<bool __stable, typename _Tp, typename _Compare>
class __gnu_parallel::_LoserTreeUnguarded< __stable, _Tp, _Compare >
```

Stable implementation of unguarded `_LoserTree`.

Unstable variant is selected below with partial specialization.

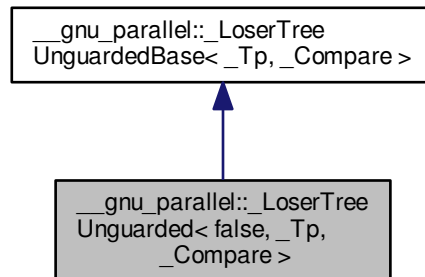
Definition at line 646 of file `losertree.h`.

The documentation for this class was generated from the following file:

- [losertree.h](#)

4.128 `__gnu_parallel::_LoserTreeUnguarded< false, _Tp, _Compare >` Class Template Reference

Inheritance diagram for `__gnu_parallel::_LoserTreeUnguarded< false, _Tp, _Compare >`:



Public Member Functions

- **`_LoserTreeUnguarded`** (unsigned int __k, const _Tp &__sentinel, _Compare __comp=[std::less](#)< _Tp >())
- void **`__delete_min_insert`** (_Tp __key, bool)
- int **`__get_min_source`** ()
- void **`__init`** ()
- unsigned int **`__init_winner`** (unsigned int __root)
- void **`__insert_start`** (const _Tp &__key, int __source, bool)

Protected Attributes

- _Compare **`_M_comp`**
- unsigned int **`_M_ik`**
- unsigned int **`_M_k`**
- _Loser * **`_M_losers`**
- unsigned int **`_M_offset`**

4.128.1 Detailed Description

```
template<typename _Tp, typename _Compare>
class __gnu_parallel::_LoserTreeUnguarded< false, _Tp, _Compare >
```

Non-Stable implementation of unguarded `_LoserTree`.

Stable implementation is above.

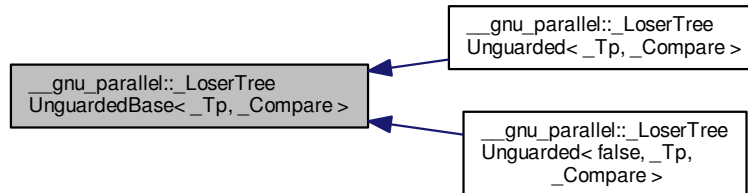
Definition at line 734 of file `losertree.h`.

The documentation for this class was generated from the following file:

- [losertree.h](#)

4.129 `__gnu_parallel::_LoserTreeUnguardedBase<_Tp, _Compare>` Class Template Reference

Inheritance diagram for `__gnu_parallel::_LoserTreeUnguardedBase<_Tp, _Compare>`:



Public Member Functions

- `_LoserTreeUnguardedBase` (unsigned int __k, const _Tp &__sentinel, _Compare __comp=[std::less](#)<_Tp>())
- int `__get_min_source` ()
- void `__insert_start` (const _Tp &__key, int __source, bool)

Protected Attributes

- _Compare `_M_comp`
- unsigned int `_M_ik`
- unsigned int `_M_k`
- _Loser * `_M_losers`
- unsigned int `_M_offset`

4.129.1 Detailed Description

```
template<typename _Tp, typename _Compare>
class __gnu_parallel::_LoserTreeUnguardedBase<_Tp, _Compare>
```

Base class for unguarded `_LoserTree` implementation.

The whole element is copied into the tree structure.

No guarding is done, therefore not a single input sequence must run empty. Unused `__sequence` heads are marked with a sentinel which is `>` all elements that are to be merged.

This is a very fast variant.

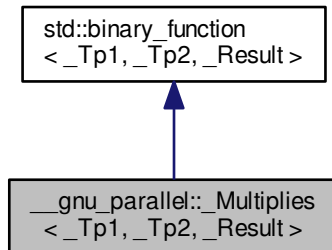
Definition at line 574 of file `losertree.h`.

The documentation for this class was generated from the following file:

- [losertree.h](#)

4.130 `__gnu_parallel::_Multiplies<_Tp1, _Tp2, _Result>` Struct Template Reference

Inheritance diagram for `__gnu_parallel::_Multiplies<_Tp1, _Tp2, _Result>`:



Public Types

- typedef `_Tp1` `first_argument_type`
- typedef `_Result` `result_type`
- typedef `_Tp2` `second_argument_type`

Public Member Functions

- `_Result` **operator()** (const `_Tp1` &`__x`, const `_Tp2` &`__y`) const

4.130.1 Detailed Description

```
template<typename _Tp1, typename _Tp2, typename _Result = __typeof__(*static_cast<_Tp1*>(0) * *static_cast<_Tp2*>(0))>
struct __gnu_parallel::_Multiplies<_Tp1, _Tp2, _Result>
```

Similar to `std::multiplies`, but allows two different types.

Definition at line 288 of file `parallel/base.h`.

4.130.2 Member Typedef Documentation

4.130.2.1 typedef `_Tp1` `std::binary_function<_Tp1, _Tp2, _Result>::first_argument_type` [inherited]

`first_argument_type` is the type of the first argument

Definition at line 121 of file `stl_function.h`.

4.130.2.2 `typedef _Result std::binary_function<_Tp1, _Tp2, _Result>::result_type` `[inherited]`

`result_type` is the return type

Definition at line 127 of file `stl_function.h`.

4.130.2.3 `typedef _Tp2 std::binary_function<_Tp1, _Tp2, _Result>::second_argument_type` `[inherited]`

`second_argument_type` is the type of the second argument

Definition at line 124 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [parallel/base.h](#)

4.131 __gnu_parallel::_Nothing Struct Reference

Public Member Functions

- `template<typename _It>`
`void operator() (_It __i)`

4.131.1 Detailed Description

Functor doing nothing.

For some `__reduction` tasks (this is not a function object, but is passed as `__selector` `__dummy` parameter.

Definition at line 288 of file `for_each_selectors.h`.

4.131.2 Member Function Documentation

4.131.2.1 `template<typename _It> void __gnu_parallel::_Nothing::operator() (_It __i)` `[inline]`

Functor execution.

Parameters

<code>↔</code>	iterator referencing object.
<code>__↔</code>	
<code>↔</code>	
<code>__↔</code>	
<code>i</code>	

Definition at line 294 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for_each_selectors.h](#)

4.132 `__gnu_parallel::_Piece<_DifferenceTp>` Struct Template Reference

Public Types

- `typedef _DifferenceTp _DifferenceType`

Public Attributes

- `_DifferenceType _M_begin`
- `_DifferenceType _M_end`

4.132.1 Detailed Description

```
template<typename _DifferenceTp>
struct __gnu_parallel::_Piece<_DifferenceTp >
```

Subsequence description.

Definition at line 46 of file `multiway_mergesort.h`.

4.132.2 Member Data Documentation

4.132.2.1 `template<typename _DifferenceTp> _DifferenceType __gnu_parallel::_Piece<_DifferenceTp>::_M_begin`

Begin of subsequence.

Definition at line 51 of file `multiway_mergesort.h`.

4.132.2.2 `template<typename _DifferenceTp> _DifferenceType __gnu_parallel::_Piece<_DifferenceTp>::_M_end`

End of subsequence.

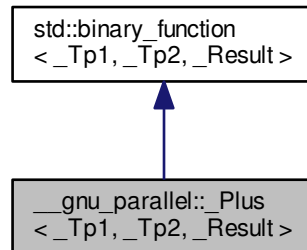
Definition at line 54 of file `multiway_mergesort.h`.

The documentation for this struct was generated from the following file:

- [multiway_mergesort.h](#)

4.133 `__gnu_parallel::_Plus<_Tp1, _Tp2, _Result >` Struct Template Reference

Inheritance diagram for `__gnu_parallel::_Plus<_Tp1, _Tp2, _Result >`:



Public Types

- `typedef _Tp1 first_argument_type`
- `typedef _Result result_type`
- `typedef _Tp2 second_argument_type`

Public Member Functions

- `_Result operator() (const _Tp1 &__x, const _Tp2 &__y) const`

4.133.1 Detailed Description

```
template<typename _Tp1, typename _Tp2, typename _Result = __typeof__(*static_cast<_Tp1*>(0) + *static_cast<_Tp2*>(0))>
struct __gnu_parallel::_Plus<_Tp1, _Tp2, _Result >
```

Similar to `std::plus`, but allows two different types.

Definition at line 272 of file `parallel/base.h`.

4.133.2 Member Typedef Documentation

4.133.2.1 `typedef _Tp1 std::binary_function<_Tp1, _Tp2, _Result>::first_argument_type` [inherited]

`first_argument_type` is the type of the first argument

Definition at line 121 of file `stl_function.h`.

4.133.2.2 `typedef _Result std::binary_function<_Tp1, _Tp2, _Result>::result_type` [inherited]

`result_type` is the return type

Definition at line 127 of file `stl_function.h`.

4.133.2.3 `typedef _Tp2 std::binary_function<_Tp1, _Tp2, _Result>::second_argument_type` [inherited]

`second_argument_type` is the type of the second argument

Definition at line 124 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [parallel/base.h](#)

4.134 `__gnu_parallel::PMWMSortingData<_RAIter>` Struct Template Reference

Public Types

- `typedef _TraitsType::difference_type` **_DifferenceType**
- `typedef std::iterator_traits<_RAIter>` **_TraitsType**
- `typedef _TraitsType::value_type` **_ValueType**

Public Attributes

- [_ThreadIndex](#) [_M_num_threads](#)
- `_DifferenceType *` [_M_offsets](#)
- `std::vector<_Piece<_DifferenceType>> *` [_M_pieces](#)
- `_ValueType *` [_M_samples](#)
- `_RAIter` [_M_source](#)
- `_DifferenceType *` [_M_starts](#)
- `_ValueType **` [_M_temporary](#)

4.134.1 Detailed Description

```
template<typename _RAIter>
struct __gnu_parallel::PMWMSortingData<_RAIter>
```

Data accessed by all threads.

PMWMS = parallel multiway mergesort

Definition at line 61 of file `multiway_mergesort.h`.

4.134.2 Member Data Documentation

4.134.2.1 `template<typename _RAIter> _ThreadIndex __gnu_parallel::PMWSSortingData<_RAIter>::M_num_threads`

Number of threads involved.

Definition at line 68 of file `multiway_mergesort.h`.

Referenced by `__gnu_parallel::parallel_sort_mwms()`, and `__gnu_parallel::parallel_sort_mwms_pu()`.

4.134.2.2 `template<typename _RAIter> _DifferenceType* __gnu_parallel::PMWSSortingData<_RAIter>::M_offsets`

Offsets to add to the found positions.

Definition at line 83 of file `multiway_mergesort.h`.

Referenced by `__gnu_parallel::parallel_sort_mwms()`.

4.134.2.3 `template<typename _RAIter> std::vector<_Piece<_DifferenceType>>* __gnu_parallel::PMWSSortingData<_RAIter>::M_pieces`

Pieces of data to merge [thread][__sequence].

Definition at line 86 of file `multiway_mergesort.h`.

Referenced by `__gnu_parallel::parallel_sort_mwms()`, and `__gnu_parallel::parallel_sort_mwms_pu()`.

4.134.2.4 `template<typename _RAIter> _ValueType* __gnu_parallel::PMWSSortingData<_RAIter>::M_samples`

Samples.

Definition at line 80 of file `multiway_mergesort.h`.

Referenced by `__gnu_parallel::determine_samples()`, and `__gnu_parallel::parallel_sort_mwms()`.

4.134.2.5 `template<typename _RAIter> _RAIter __gnu_parallel::PMWSSortingData<_RAIter>::M_source`

Input __begin.

Definition at line 71 of file `multiway_mergesort.h`.

Referenced by `__gnu_parallel::determine_samples()`, `__gnu_parallel::parallel_sort_mwms()`, and `__gnu_parallel::parallel_sort_mwms_pu()`.

4.134.2.6 `template<typename _RAIter> _DifferenceType* __gnu_parallel::PMWSSortingData<_RAIter>::M_starts`

Start indices, per thread.

Definition at line 74 of file `multiway_mergesort.h`.

Referenced by `__gnu_parallel::determine_samples()`, `__gnu_parallel::parallel_sort_mwms()`, and `__gnu_parallel::parallel_sort_mwms_pu()`.

4.134.2.7 `template<typename _RAIter> _ValueType** __gnu_parallel::_PMWMSSortingData<_RAIter>::_M_temporary`

Storage in which to sort.

Definition at line 77 of file `multiway_mergesort.h`.

Referenced by `__gnu_parallel::parallel_sort_mwms()`, and `__gnu_parallel::parallel_sort_mwms_pu()`.

The documentation for this struct was generated from the following file:

- [multiway_mergesort.h](#)

4.135 `__gnu_parallel::_PseudoSequence<_Tp, _DifferenceTp>` Class Template Reference

Public Types

- typedef `_DifferenceTp` **`_DifferenceType`**
- typedef `_PseudoSequenceIterator<_Tp, uint64_t>` **`iterator`**

Public Member Functions

- `_PseudoSequence` (`const _Tp &__val, _DifferenceType __count`)
- `iterator begin` () const
- `iterator end` () const

4.135.1 Detailed Description

```
template<typename _Tp, typename _DifferenceTp>
class __gnu_parallel::_PseudoSequence<_Tp, _DifferenceTp>
```

Sequence that conceptually consists of multiple copies of the same element. The copies are not stored explicitly, of course.

Parameters

<code>_Tp</code>	Sequence <code>_M_value</code> type.
<code>_DifferenceTp</code>	Sequence difference type.

Definition at line 359 of file `parallel/base.h`.

4.135.2 Constructor & Destructor Documentation

4.135.2.1 `template<typename _Tp, typename _DifferenceTp> __gnu_parallel::_PseudoSequence<_Tp, _DifferenceTp>::_PseudoSequence (const _Tp & __val, _DifferenceType __count) [inline]`

Constructor.

Parameters

<code>__val</code>	Element of the sequence.
<code>__count</code>	Number of (virtual) copies.

Definition at line 371 of file `parallel/base.h`.

4.135.3 Member Function Documentation

4.135.3.1 `template<typename _Tp, typename _DifferenceTp> iterator __gnu_parallel::_PseudoSequence<_Tp, _DifferenceTp>::begin () const [inline]`

Begin iterator.

Definition at line 376 of file `parallel/base.h`.

4.135.3.2 `template<typename _Tp, typename _DifferenceTp> iterator __gnu_parallel::_PseudoSequence<_Tp, _DifferenceTp>::end () const [inline]`

End iterator.

Definition at line 381 of file `parallel/base.h`.

The documentation for this class was generated from the following file:

- [parallel/base.h](#)

4.136 `__gnu_parallel::_PseudoSequencer<_Tp, _DifferenceTp>` Class Template Reference

Public Types

- `typedef _DifferenceTp _DifferenceType`

Public Member Functions

- `_PseudoSequencer (const _Tp &__val, _DifferenceType __pos)`
- `bool operator!= (const _PseudoSequencer &__i2)`
- `const _Tp & operator* () const`
- `_PseudoSequencer & operator++ ()`
- `_PseudoSequencer operator++ (int)`
- `_DifferenceType operator- (const _PseudoSequencer &__i2)`
- `bool operator== (const _PseudoSequencer &__i2)`
- `const _Tp & operator[] (_DifferenceType) const`

4.136.1 Detailed Description

```
template<typename _Tp, typename _DifferenceTp>  
class __gnu_parallel::_PseudoSequenceIterator< _Tp, _DifferenceTp >
```

_Iterator associated with __gnu_parallel::_PseudoSequence. It features the usual random-access iterator functionality.

Parameters

<code>_Tp</code>	Sequence <code>_M_value</code> type.
<code>_DifferenceTp</code>	Sequence difference type.

Definition at line 306 of file `parallel/base.h`.

The documentation for this class was generated from the following file:

- [parallel/base.h](#)

4.137 `__gnu_parallel::__QSBThreadLocal<_RAIter>` Struct Template Reference

Public Types

- typedef `_TraitsType::difference_type` **`_DifferenceType`**
- typedef `std::pair<_RAIter, _RAIter>` [_Piece](#)
- typedef `std::iterator_traits<_RAIter>` **`_TraitsType`**

Public Member Functions

- [_QSBThreadLocal](#) (int `__queue_size`)

Public Attributes

- volatile `_DifferenceType *` [_M_elements_leftover](#)
- [_Piece](#) [_M_global](#)
- [_Piece](#) [_M_initial](#)
- [_RestrictedBoundedConcurrentQueue<_Piece>](#) [_M_leftover_parts](#)
- [_ThreadIndex](#) [_M_num_threads](#)

4.137.1 Detailed Description

```
template<typename _RAIter>
struct __gnu_parallel::__QSBThreadLocal<_RAIter>
```

Information local to one thread in the parallel quicksort run.

Definition at line 62 of file `balanced_quicksort.h`.

4.137.2 Member Typedef Documentation

4.137.2.1 `template<typename _RAIter> typedef std::pair<_RAIter, _RAIter> __gnu_parallel::_QSBThreadLocal<_RAIter>::_Piece`

Continuous part of the sequence, described by an iterator pair.

Definition at line 69 of file `balanced_quicksort.h`.

4.137.3 Constructor & Destructor Documentation

4.137.3.1 `template<typename _RAIter> __gnu_parallel::_QSBThreadLocal<_RAIter>::_QSBThreadLocal (int __queue_size) [inline]`

Constructor.

Parameters

<code>__queue_size</code>	size of the work-stealing queue.
---------------------------	----------------------------------

Definition at line 88 of file `balanced_quicksort.h`.

4.137.4 Member Data Documentation

4.137.4.1 `template<typename _RAIter> volatile _DifferenceType* __gnu_parallel::__QSBThreadLocal<_RAIter>::__M_elements_leftover`

Pointer to a counter of elements left over to sort.

Definition at line 81 of file `balanced_quicksort.h`.

Referenced by `__gnu_parallel::__parallel_sort_qsb()`, `__gnu_parallel::__qsb_conquer()`, and `__gnu_parallel::__qsb_local_sort_with_helping()`.

4.137.4.2 `template<typename _RAIter> _Piece __gnu_parallel::__QSBThreadLocal<_RAIter>::__M_global`

The complete sequence to sort.

Definition at line 84 of file `balanced_quicksort.h`.

4.137.4.3 `template<typename _RAIter> _Piece __gnu_parallel::__QSBThreadLocal<_RAIter>::__M_initial`

Initial piece to work on.

Definition at line 72 of file `balanced_quicksort.h`.

Referenced by `__gnu_parallel::__qsb_conquer()`, and `__gnu_parallel::__qsb_local_sort_with_helping()`.

4.137.4.4 `template<typename _RAIter> _RestrictedBoundedConcurrentQueue<_Piece> __gnu_parallel::__QSBThreadLocal<_RAIter>::__M_leftover_parts`

Work-stealing queue.

Definition at line 75 of file `balanced_quicksort.h`.

Referenced by `__gnu_parallel::__parallel_sort_qsb()`, and `__gnu_parallel::__qsb_local_sort_with_helping()`.

4.137.4.5 `template<typename _RAIter> _ThreadIndex __gnu_parallel::__QSBThreadLocal<_RAIter>::__M_num_threads`

Number of threads involved in this algorithm.

Definition at line 78 of file `balanced_quicksort.h`.

Referenced by `__gnu_parallel::__qsb_local_sort_with_helping()`.

The documentation for this struct was generated from the following file:

- [balanced_quicksort.h](#)

4.138 __gnu_parallel::_RandomNumber Class Reference

Public Member Functions

- [_RandomNumber](#) ()
- [_RandomNumber](#) (uint32_t __seed, uint64_t _M_supremum=0x100000000ULL)
- unsigned long [__genrand_bits](#) (int __bits)
- uint32_t [operator\(\)](#) ()
- uint32_t [operator\(\)](#) (uint64_t local_supremum)

4.138.1 Detailed Description

Random number generator, based on the Mersenne twister.

Definition at line 42 of file random_number.h.

4.138.2 Constructor & Destructor Documentation

4.138.2.1 __gnu_parallel::_RandomNumber::_RandomNumber () [inline]

Default constructor. Seed with 0.

Definition at line 74 of file random_number.h.

4.138.2.2 __gnu_parallel::_RandomNumber::_RandomNumber (uint32_t __seed, uint64_t _M_supremum = 0x100000000ULL) [inline]

Constructor.

Parameters

<i>__seed</i>	Random __seed.
<i>_M_supremum</i>	Generate integer random numbers in the interval [0, _M_supremum).

Definition at line 85 of file random_number.h.

4.138.3 Member Function Documentation

4.138.3.1 unsigned long __gnu_parallel::_RandomNumber::_genrand_bits (int __bits) [inline]

Generate a number of random bits, run-time parameter.

Parameters

<code>__bits</code>	Number of bits to generate.
---------------------	-----------------------------

Definition at line 109 of file `random_number.h`.

4.138.3.2 `uint32_t __gnu_parallel::RandomNumber::operator() ()` `[inline]`

Generate unsigned random 32-bit integer.

Definition at line 94 of file `random_number.h`.

4.138.3.3 `uint32_t __gnu_parallel::RandomNumber::operator() (uint64_t local_supremum)` `[inline]`

Generate unsigned random 32-bit integer in the interval `[0,local_supremum)`.

Definition at line 100 of file `random_number.h`.

The documentation for this class was generated from the following file:

- [random_number.h](#)

4.139 `__gnu_parallel::_RestrictedBoundedConcurrentQueue<_Tp>` Class Template Reference

Public Member Functions

- [_RestrictedBoundedConcurrentQueue](#) ([_SequenceIndex](#) __max_size)
- [~_RestrictedBoundedConcurrentQueue](#) ()
- bool [pop_back](#) ([_Tp](#) &__t)
- bool [pop_front](#) ([_Tp](#) &__t)
- void [push_front](#) (const [_Tp](#) &__t)

4.139.1 Detailed Description

```
template<typename _Tp>
class __gnu_parallel::_RestrictedBoundedConcurrentQueue<_Tp>
```

Double-ended queue of bounded size, allowing lock-free atomic access. `push_front()` and `pop_front()` must not be called concurrently to each other, while `pop_back()` can be called concurrently at all times. `empty()`, `size()`, and `top()` are intentionally not provided. Calling them would not make sense in a concurrent setting.

Parameters

<code>_Tp</code>	Contained element type.
------------------	-------------------------

Definition at line 52 of file queue.h.

4.139.2 Constructor & Destructor Documentation

4.139.2.1 `template<typename _Tp> __gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp
>::_RestrictedBoundedConcurrentQueue (_SequenceIndex __max_size) [inline]`

Constructor. Not to be called concurrent, of course.

Parameters

<code>__max_size</code>	Maximal number of elements to be contained.
-------------------------	---

Definition at line 68 of file queue.h.

4.139.2.2 `template<typename _Tp> __gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp
>::~~_RestrictedBoundedConcurrentQueue () [inline]`

Destructor. Not to be called concurrent, of course.

Definition at line 77 of file queue.h.

4.139.3 Member Function Documentation

4.139.3.1 `template<typename _Tp> bool __gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp >::pop_back (
_Tp &__t) [inline]`

Pops one element from the queue at the front end. Must not be called concurrently with pop_front().

Definition at line 127 of file queue.h.

4.139.3.2 `template<typename _Tp> bool __gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp >::pop_front (
_Tp &__t) [inline]`

Pops one element from the queue at the front end. Must not be called concurrently with pop_front().

Definition at line 100 of file queue.h.

4.139.3.3 `template<typename _Tp> void __gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp >::push_front (
const _Tp &__t) [inline]`

Pushes one element into the queue at the front end. Must not be called concurrently with pop_front().

Definition at line 83 of file queue.h.

The documentation for this class was generated from the following file:

- [queue.h](#)

4.140 `__gnu_parallel::_SamplingSorter< __stable, _RAIter, _StrictWeakOrdering >` Struct Template Reference

Public Member Functions

- void **operator()** (_RAIter __first, _RAIter __last, _StrictWeakOrdering __comp)

4.140.1 Detailed Description

```
template<bool __stable, class _RAIter, class _StrictWeakOrdering>
struct __gnu_parallel::_SamplingSorter< __stable, _RAIter, _StrictWeakOrdering >
```

Stable sorting functor.

Used to reduce code instantiation in `multiway_merge_sampling_splitting`.

Definition at line 1007 of file `multiway_merge.h`.

The documentation for this struct was generated from the following file:

- [multiway_merge.h](#)

4.141 `__gnu_parallel::_SamplingSorter< false, _RAIter, _StrictWeakOrdering >` Struct Template Reference

Public Member Functions

- void **operator()** (_RAIter __first, _RAIter __last, _StrictWeakOrdering __comp)

4.141.1 Detailed Description

```
template<class _RAIter, class _StrictWeakOrdering>
struct __gnu_parallel::_SamplingSorter< false, _RAIter, _StrictWeakOrdering >
```

Non-`__stable` sorting functor.

Used to reduce code instantiation in `multiway_merge_sampling_splitting`.

Definition at line 1020 of file `multiway_merge.h`.

The documentation for this struct was generated from the following file:

- [multiway_merge.h](#)

4.142 `__gnu_parallel::_Settings` Struct Reference

Static Public Member Functions

- static const `_Settings` & `get` () throw ()
- static void `set` (`_Settings` &) throw ()

Public Attributes

- `_SequenceIndex` `accumulate_minimal_n`
- unsigned int `adjacent_difference_minimal_n`
- `_AlgorithmStrategy` `algorithm_strategy`
- unsigned int `cache_line_size`
- `_SequenceIndex` `count_minimal_n`
- `_SequenceIndex` `fill_minimal_n`
- `_FindAlgorithm` `find_algorithm`
- double `find_increasing_factor`
- `_SequenceIndex` `find_initial_block_size`
- `_SequenceIndex` `find_maximum_block_size`
- float `find_scale_factor`
- `_SequenceIndex` `find_sequential_search_size`
- `_SequenceIndex` `for_each_minimal_n`
- `_SequenceIndex` `generate_minimal_n`
- unsigned long long `L1_cache_size`
- unsigned long long `L2_cache_size`
- `_SequenceIndex` `max_element_minimal_n`
- `_SequenceIndex` `merge_minimal_n`
- unsigned int `merge_oversampling`
- `_SplittingAlgorithm` `merge_splitting`
- `_SequenceIndex` `min_element_minimal_n`
- `_MultiwayMergeAlgorithm` `multiway_merge_algorithm`
- int `multiway_merge_minimal_k`
- `_SequenceIndex` `multiway_merge_minimal_n`
- unsigned int `multiway_merge_oversampling`
- `_SplittingAlgorithm` `multiway_merge_splitting`
- `_SequenceIndex` `nth_element_minimal_n`
- `_SequenceIndex` `partial_sort_minimal_n`
- `_PartialSumAlgorithm` `partial_sum_algorithm`
- float `partial_sum_dilation`
- unsigned int `partial_sum_minimal_n`
- double `partition_chunk_share`
- `_SequenceIndex` `partition_chunk_size`
- `_SequenceIndex` `partition_minimal_n`
- `_SequenceIndex` `qsb_steals`
- unsigned int `random_shuffle_minimal_n`
- `_SequenceIndex` `replace_minimal_n`
- `_SequenceIndex` `search_minimal_n`
- `_SequenceIndex` `set_difference_minimal_n`
- `_SequenceIndex` `set_intersection_minimal_n`
- `_SequenceIndex` `set_symmetric_difference_minimal_n`

- [_SequenceIndex](#) `set_union_minimal_n`
- [_SortAlgorithm](#) `sort_algorithm`
- [_SequenceIndex](#) `sort_minimal_n`
- unsigned int `sort_mwms_oversampling`
- unsigned int `sort_qs_num_samples_preset`
- [_SequenceIndex](#) `sort_qsb_base_case_maximal_n`
- [_SplittingAlgorithm](#) `sort_splitting`
- unsigned int `TLB_size`
- [_SequenceIndex](#) `transform_minimal_n`
- [_SequenceIndex](#) `unique_copy_minimal_n`
- [_SequenceIndex](#) `workstealing_chunk_size`

4.142.1 Detailed Description

class `_Settings` Run-time settings for the parallel mode including all tunable parameters.

Definition at line 123 of file `settings.h`.

4.142.2 Member Function Documentation

4.142.2.1 `static const _Settings& __gnu_parallel::Settings::get() throw` [static]

Get the global settings.

Referenced by `__gnu_parallel::__find_template()`, `__gnu_parallel::__for_each_template_random_access_workstealing()`, `__gnu_parallel::__parallel_nth_element()`, `__gnu_parallel::__parallel_partial_sum()`, `__gnu_parallel::__parallel_partial_sum_linear()`, `__gnu_parallel::__parallel_partition()`, `__gnu_parallel::__parallel_random_shuffle_drs()`, `__gnu_parallel::__parallel_sort()`, `__gnu_parallel::__parallel_sort_qs_conquer()`, `__gnu_parallel::__qsb_local_sort_with_helping()`, `__gnu_parallel::__sequential_random_shuffle()`, `__gnu_parallel::multiway_merge()`, `__gnu_parallel::multiway_merge_sampling_splitting()`, `__gnu_parallel::multiway_merge_sentinels()`, `__gnu_parallel::parallel_multiway_merge()`, `__gnu_parallel::parallel_sort_mwms()`, and `__gnu_parallel::parallel_sort_mwms_pu()`.

4.142.2.2 `static void __gnu_parallel::Settings::set (_Settings &) throw` [static]

Set the global settings.

4.142.3 Member Data Documentation

4.142.3.1 `_SequenceIndex __gnu_parallel::Settings::accumulate_minimal_n`

Minimal input size for accumulate.

Definition at line 139 of file `settings.h`.

4.142.3.2 `unsigned int __gnu_parallel::_Settings::adjacent_difference_minimal_n`

Minimal input size for `adjacent_difference`.

Definition at line 142 of file `settings.h`.

4.142.3.3 `unsigned int __gnu_parallel::_Settings::cache_line_size`

Overestimation of cache line size. Used to avoid false sharing, i.e. elements of different threads are at least this amount apart.

Definition at line 265 of file `settings.h`.

Referenced by `__gnu_parallel::_for_each_template_random_access_workstealing()`.

4.142.3.4 `_SequenceIndex __gnu_parallel::_Settings::count_minimal_n`

Minimal input size for `count` and `count_if`.

Definition at line 145 of file `settings.h`.

4.142.3.5 `_SequenceIndex __gnu_parallel::_Settings::fill_minimal_n`

Minimal input size for `fill`.

Definition at line 148 of file `settings.h`.

4.142.3.6 `double __gnu_parallel::_Settings::find_increasing_factor`

Block size increase factor for `find`.

Definition at line 151 of file `settings.h`.

4.142.3.7 `_SequenceIndex __gnu_parallel::_Settings::find_initial_block_size`

Initial block size for `find`.

Definition at line 154 of file `settings.h`.

Referenced by `__gnu_parallel::_find_template()`.

4.142.3.8 `_SequenceIndex __gnu_parallel::_Settings::find_maximum_block_size`

Maximal block size for `find`.

Definition at line 157 of file `settings.h`.

4.142.3.9 `float __gnu_parallel::_Settings::find_scale_factor`

Block size scale-down factor with respect to current position.

Definition at line 276 of file `settings.h`.

Referenced by `__gnu_parallel::_find_template()`.

4.142.3.10 `_SequenceIndex __gnu_parallel::_Settings::find_sequential_search_size`

Start with looking for this many elements sequentially, for `find`.

Definition at line 160 of file `settings.h`.

Referenced by `__gnu_parallel::_find_template()`.

4.142.3.11 `_SequenceIndex __gnu_parallel::_Settings::for_each_minimal_n`

Minimal input size for `for_each`.

Definition at line 163 of file `settings.h`.

4.142.3.12 `_SequenceIndex __gnu_parallel::_Settings::generate_minimal_n`

Minimal input size for `generate`.

Definition at line 166 of file `settings.h`.

4.142.3.13 `unsigned long long __gnu_parallel::_Settings::L1_cache_size`

size of the L1 cache in bytes (underestimation).

Definition at line 254 of file `settings.h`.

4.142.3.14 `unsigned long long __gnu_parallel::_Settings::L2_cache_size`

size of the L2 cache in bytes (underestimation).

Definition at line 257 of file `settings.h`.

Referenced by `__gnu_parallel::_parallel_random_shuffle_drs()`, and `__gnu_parallel::_sequential_random_shuffle()`.

4.142.3.15 `_SequenceIndex __gnu_parallel::_Settings::max_element_minimal_n`

Minimal input size for `max_element`.

Definition at line 169 of file `settings.h`.

4.142.3.16 `_SequenceIndex __gnu_parallel::_Settings::merge_minimal_n`

Minimal input size for merge.

Definition at line 172 of file settings.h.

4.142.3.17 `unsigned int __gnu_parallel::_Settings::merge_oversampling`

Oversampling factor for merge.

Definition at line 175 of file settings.h.

Referenced by `__gnu_parallel::multiway_merge_sampling_splitting()`, and `__gnu_parallel::parallel_multiway_merge()`.

4.142.3.18 `_SequenceIndex __gnu_parallel::_Settings::min_element_minimal_n`

Minimal input size for `min_element`.

Definition at line 178 of file settings.h.

4.142.3.19 `int __gnu_parallel::_Settings::multiway_merge_minimal_k`

Oversampling factor for `multiway_merge`.

Definition at line 184 of file settings.h.

4.142.3.20 `_SequenceIndex __gnu_parallel::_Settings::multiway_merge_minimal_n`

Minimal input size for `multiway_merge`.

Definition at line 181 of file settings.h.

4.142.3.21 `unsigned int __gnu_parallel::_Settings::multiway_merge_oversampling`

Oversampling factor for `multiway_merge`.

Definition at line 187 of file settings.h.

4.142.3.22 `_SequenceIndex __gnu_parallel::_Settings::nth_element_minimal_n`

Minimal input size for `nth_element`.

Definition at line 190 of file settings.h.

Referenced by `__gnu_parallel::__parallel_nth_element()`.

4.142.3.23 `_SequenceIndex __gnu_parallel::_Settings::partial_sort_minimal_n`

Minimal input size for `partial_sort`.

Definition at line 203 of file settings.h.

4.142.3.24 float __gnu_parallel::_Settings::partial_sum_dilation

Ratio for partial_sum. Assume "sum and write result" to be this factor slower than just "sum".

Definition at line 207 of file settings.h.

Referenced by __gnu_parallel::__parallel_partial_sum_linear().

4.142.3.25 unsigned int __gnu_parallel::_Settings::partial_sum_minimal_n

Minimal input size for partial_sum.

Definition at line 210 of file settings.h.

4.142.3.26 double __gnu_parallel::_Settings::partition_chunk_share

Chunk size for partition, relative to input size. If > 0.0, this value overrides partition_chunk_size.

Definition at line 197 of file settings.h.

Referenced by __gnu_parallel::__parallel_partition().

4.142.3.27 _SequenceIndex __gnu_parallel::_Settings::partition_chunk_size

Chunk size for partition.

Definition at line 193 of file settings.h.

Referenced by __gnu_parallel::__parallel_partition().

4.142.3.28 _SequenceIndex __gnu_parallel::_Settings::partition_minimal_n

Minimal input size for partition.

Definition at line 200 of file settings.h.

Referenced by __gnu_parallel::__parallel_nth_element().

4.142.3.29 _SequenceIndex __gnu_parallel::_Settings::qsb_steals

The number of stolen ranges in load-balanced quicksort.

Definition at line 270 of file settings.h.

4.142.3.30 unsigned int __gnu_parallel::_Settings::random_shuffle_minimal_n

Minimal input size for random_shuffle.

Definition at line 213 of file settings.h.

4.142.3.31 `_SequenceIndex __gnu_parallel::_Settings::replace_minimal_n`

Minimal input size for `replace` and `replace_if`.

Definition at line 216 of file `settings.h`.

4.142.3.32 `_SequenceIndex __gnu_parallel::_Settings::search_minimal_n`

Minimal input size for `search` and `search_n`.

Definition at line 273 of file `settings.h`.

4.142.3.33 `_SequenceIndex __gnu_parallel::_Settings::set_difference_minimal_n`

Minimal input size for `set_difference`.

Definition at line 219 of file `settings.h`.

4.142.3.34 `_SequenceIndex __gnu_parallel::_Settings::set_intersection_minimal_n`

Minimal input size for `set_intersection`.

Definition at line 222 of file `settings.h`.

4.142.3.35 `_SequenceIndex __gnu_parallel::_Settings::set_symmetric_difference_minimal_n`

Minimal input size for `set_symmetric_difference`.

Definition at line 225 of file `settings.h`.

4.142.3.36 `_SequenceIndex __gnu_parallel::_Settings::set_union_minimal_n`

Minimal input size for `set_union`.

Definition at line 228 of file `settings.h`.

4.142.3.37 `_SequenceIndex __gnu_parallel::_Settings::sort_minimal_n`

Minimal input size for parallel sorting.

Definition at line 231 of file `settings.h`.

4.142.3.38 `unsigned int __gnu_parallel::_Settings::sort_mwms_oversampling`

Oversampling factor for parallel `std::sort` (MWMS).

Definition at line 234 of file `settings.h`.

Referenced by `__gnu_parallel::parallel_sort_mwms()`, and `__gnu_parallel::parallel_sort_mwms_pu()`.

4.142.3.39 `unsigned int __gnu_parallel::_Settings::sort_qs_num_samples_preset`

Such many samples to take to find a good pivot (quicksort).

Definition at line 237 of file `settings.h`.

4.142.3.40 `_SequenceIndex __gnu_parallel::_Settings::sort_qsb_base_case_maximal_n`

Maximal subsequence `__length` to switch to unbalanced `__base` case. Applies to `std::sort` with dynamically load-balanced quicksort.

Definition at line 241 of file `settings.h`.

Referenced by `__gnu_parallel::__qsb_local_sort_with_helping()`.

4.142.3.41 `unsigned int __gnu_parallel::_Settings::TLB_size`

size of the Translation Lookaside Buffer (underestimation).

Definition at line 260 of file `settings.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`, and `__gnu_parallel::__sequential_random_shuffle()`.

4.142.3.42 `_SequenceIndex __gnu_parallel::_Settings::transform_minimal_n`

Minimal input size for parallel `std::transform`.

Definition at line 244 of file `settings.h`.

4.142.3.43 `_SequenceIndex __gnu_parallel::_Settings::unique_copy_minimal_n`

Minimal input size for `unique_copy`.

Definition at line 247 of file `settings.h`.

The documentation for this struct was generated from the following file:

- [settings.h](#)

4.143 `__gnu_parallel::_SplitConsistently< __exact, _RAIter, _Compare, _SortingPlacesIterator > Struct` Template Reference

4.143.1 Detailed Description

```
template<bool __exact, typename _RAIter, typename _Compare, typename _SortingPlacesIterator>
struct __gnu_parallel::_SplitConsistently< __exact, _RAIter, _Compare, _SortingPlacesIterator >
```

Split consistently.

Definition at line 122 of file `multiway_mergesort.h`.

The documentation for this struct was generated from the following file:

- [multiway_mergesort.h](#)

4.144 `__gnu_parallel::_SplitConsistently< false, _RAIter, _Compare, _SortingPlacesIterator >` Struct Template Reference

Public Member Functions

- void **operator()** (const [_ThreadIndex](#) __iam, [_PMWSSortingData](#)< _RAIter > *__sd, _Compare &__comp, const typename std::iterator_traits< _RAIter >::difference_type __num_samples) const

4.144.1 Detailed Description

```
template<typename _RAIter, typename _Compare, typename _SortingPlacesIterator>
struct __gnu_parallel::_SplitConsistently< false, _RAIter, _Compare, _SortingPlacesIterator >
```

Split by sampling.

Definition at line 187 of file `multiway_mergesort.h`.

The documentation for this struct was generated from the following file:

- [multiway_mergesort.h](#)

4.145 `__gnu_parallel::_SplitConsistently< true, _RAIter, _Compare, _SortingPlacesIterator >` Struct Template Reference

Public Member Functions

- void **operator()** (const [_ThreadIndex](#) __iam, [_PMWSSortingData](#)< _RAIter > *__sd, _Compare &__comp, const typename std::iterator_traits< _RAIter >::difference_type __num_samples) const

4.145.1 Detailed Description

```
template<typename _RAIter, typename _Compare, typename _SortingPlacesIterator>
struct __gnu_parallel::_SplitConsistently< true, _RAIter, _Compare, _SortingPlacesIterator >
```

Split by exact splitting.

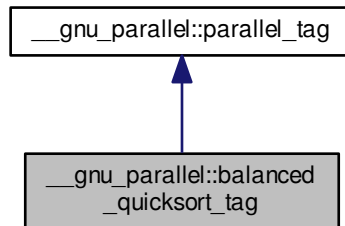
Definition at line 128 of file `multiway_mergesort.h`.

The documentation for this struct was generated from the following file:

- [multiway_mergesort.h](#)

4.146 `__gnu_parallel::balanced_quicksort_tag` Struct Reference

Inheritance diagram for `__gnu_parallel::balanced_quicksort_tag`:



Public Member Functions

- **`balanced_quicksort_tag`** (`_ThreadIndex` `__num_threads`)
- `_ThreadIndex` `__get_num_threads` ()
- void `set_num_threads` (`_ThreadIndex` `__num_threads`)

4.146.1 Detailed Description

Forces parallel sorting using balanced quicksort at compile time.

Definition at line 164 of file `tags.h`.

4.146.2 Member Function Documentation

4.146.2.1 `_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads` () `[inline]`, `[inherited]`

Find out desired number of threads.

Returns

Desired number of threads.

Definition at line 63 of file `tags.h`.

Referenced by `__gnu_parallel::__parallel_sort`(), `__gnu_parallel::multiway_merge`(), and `__gnu_parallel::multiway_merge_sentinels`().

4.146.2.2 void `__gnu_parallel::parallel_tag::set_num_threads` (`_ThreadIndex` `__num_threads`) `[inline]`, `[inherited]`

Set the desired number of threads.

Parameters

<code>__num_threads</code>	Desired number of threads.
----------------------------	----------------------------

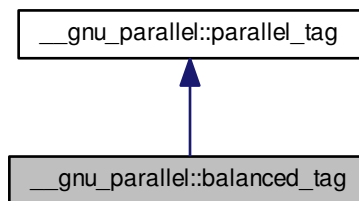
Definition at line 73 of file tags.h.

The documentation for this struct was generated from the following file:

- [tags.h](#)

4.147 `__gnu_parallel::balanced_tag` Struct Reference

Inheritance diagram for `__gnu_parallel::balanced_tag`:



Public Member Functions

- [_ThreadIndex __get_num_threads\(\)](#)
- void [set_num_threads\(_ThreadIndex __num_threads\)](#)

4.147.1 Detailed Description

Recommends parallel execution using dynamic load-balancing at compile time.

Definition at line 88 of file tags.h.

4.147.2 Member Function Documentation

4.147.2.1 `_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads()` `[inline]`, `[inherited]`

Find out desired number of threads.

Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by `__gnu_parallel::__parallel_sort()`, `__gnu_parallel::multiway_merge()`, and `__gnu_parallel::multiway_merge_sentinels()`.

4.147.2.2 `void __gnu_parallel::parallel_tag::set_num_threads (_ThreadIndex __num_threads) [inline], [inherited]`

Set the desired number of threads.

Parameters

<code>__num_threads</code>	Desired number of threads.
----------------------------	----------------------------

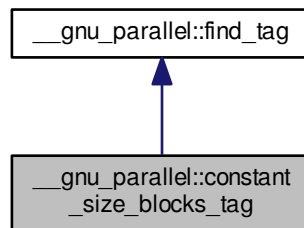
Definition at line 73 of file `tags.h`.

The documentation for this struct was generated from the following file:

- [tags.h](#)

4.148 `__gnu_parallel::constant_size_blocks_tag` Struct Reference

Inheritance diagram for `__gnu_parallel::constant_size_blocks_tag`:



4.148.1 Detailed Description

Selects the constant block size variant for `std::find()`.

See also

`_GLIBCXX_FIND_CONSTANT_SIZE_BLOCKS`

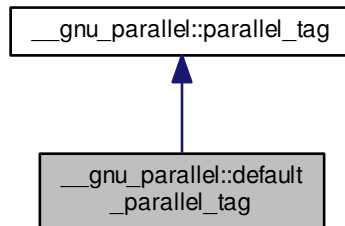
Definition at line 178 of file `tags.h`.

The documentation for this struct was generated from the following file:

- [tags.h](#)

4.149 `__gnu_parallel::default_parallel_tag` Struct Reference

Inheritance diagram for `__gnu_parallel::default_parallel_tag`:



Public Member Functions

- **`default_parallel_tag`** (`_ThreadIndex` `__num_threads`)
- `_ThreadIndex` `__get_num_threads` ()
- void `set_num_threads` (`_ThreadIndex` `__num_threads`)

4.149.1 Detailed Description

Recommends parallel execution using the default parallel algorithm.

Definition at line 79 of file `tags.h`.

4.149.2 Member Function Documentation

4.149.2.1 `_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads` () `[inline]`, `[inherited]`

Find out desired number of threads.

Returns

Desired number of threads.

Definition at line 63 of file `tags.h`.

Referenced by `__gnu_parallel::__parallel_sort`(), `__gnu_parallel::multiway_merge`(), and `__gnu_parallel::multiway_merge_sentinels`().

4.149.2.2 void `__gnu_parallel::parallel_tag::set_num_threads` (`_ThreadIndex` `__num_threads`) `[inline]`, `[inherited]`

Set the desired number of threads.

Parameters

<code>__num_threads</code>	Desired number of threads.
----------------------------	----------------------------

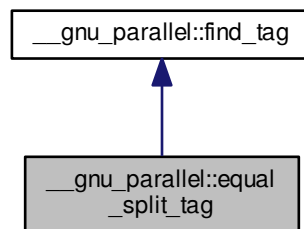
Definition at line 73 of file `tags.h`.

The documentation for this struct was generated from the following file:

- [tags.h](#)

4.150 `__gnu_parallel::equal_split_tag` Struct Reference

Inheritance diagram for `__gnu_parallel::equal_split_tag`:



4.150.1 Detailed Description

Selects the equal splitting variant for `std::find()`.

See also

`_GLIBCXX_FIND_EQUAL_SPLIT`

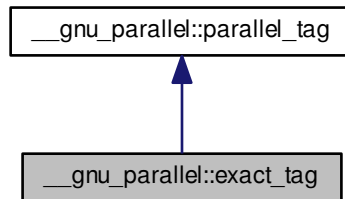
Definition at line 182 of file `tags.h`.

The documentation for this struct was generated from the following file:

- [tags.h](#)

4.151 `__gnu_parallel::exact_tag` Struct Reference

Inheritance diagram for `__gnu_parallel::exact_tag`:



Public Member Functions

- **`exact_tag`** (`_ThreadIndex` `__num_threads`)
- `_ThreadIndex` `__get_num_threads` ()
- void `set_num_threads` (`_ThreadIndex` `__num_threads`)

4.151.1 Detailed Description

Forces parallel merging with exact splitting, at compile time.

Definition at line 109 of file tags.h.

4.151.2 Member Function Documentation

4.151.2.1 `_ThreadIndex` `__gnu_parallel::parallel_tag::__get_num_threads` () `[inline]`, `[inherited]`

Find out desired number of threads.

Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by `__gnu_parallel::__parallel_sort`(), `__gnu_parallel::multiway_merge`(), and `__gnu_parallel::multiway_merge_sentinels`().

4.151.2.2 void `__gnu_parallel::parallel_tag::set_num_threads` (`_ThreadIndex` `__num_threads`) `[inline]`, `[inherited]`

Set the desired number of threads.

Parameters

<code>__num_threads</code>	Desired number of threads.
----------------------------	----------------------------

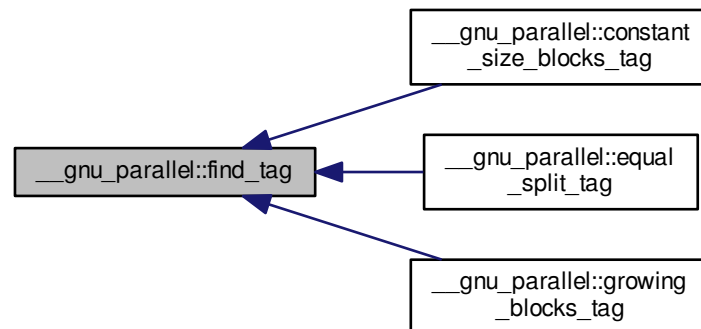
Definition at line 73 of file tags.h.

The documentation for this struct was generated from the following file:

- [tags.h](#)

4.152 `__gnu_parallel::find_tag` Struct Reference

Inheritance diagram for `__gnu_parallel::find_tag`:



4.152.1 Detailed Description

Base class for for `std::find()` variants.

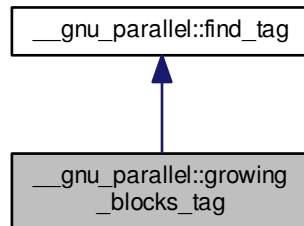
Definition at line 104 of file tags.h.

The documentation for this struct was generated from the following file:

- [tags.h](#)

4.153 `__gnu_parallel::growing_blocks_tag` Struct Reference

Inheritance diagram for `__gnu_parallel::growing_blocks_tag`:



4.153.1 Detailed Description

Selects the growing block size variant for `std::find()`.

See also

`_GLIBCXX_FIND_GROWING_BLOCKS`

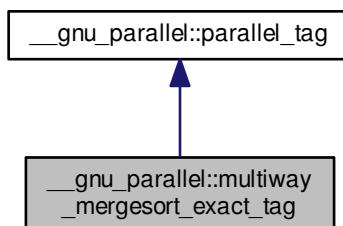
Definition at line 174 of file `tags.h`.

The documentation for this struct was generated from the following file:

- [tags.h](#)

4.154 `__gnu_parallel::multiway_mergesort_exact_tag` Struct Reference

Inheritance diagram for `__gnu_parallel::multiway_mergesort_exact_tag`:



Public Member Functions

- **`multiway_mergesort_exact_tag`** ([_ThreadIndex](#) __num_threads)
- [_ThreadIndex](#) `__get_num_threads` ()
- void `set_num_threads` ([_ThreadIndex](#) __num_threads)

4.154.1 Detailed Description

Forces parallel sorting using multiway mergesort with exact splitting at compile time.

Definition at line 137 of file tags.h.

4.154.2 Member Function Documentation

4.154.2.1 `_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads ()` `[inline]`, `[inherited]`

Find out desired number of threads.

Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by `__gnu_parallel::__parallel_sort()`, `__gnu_parallel::multiway_merge()`, and `__gnu_parallel::multiway_merge_sentinels()`.

4.154.2.2 `void __gnu_parallel::parallel_tag::set_num_threads (_ThreadIndex __num_threads)` `[inline]`, `[inherited]`

Set the desired number of threads.

Parameters

<code>__num_threads</code>	Desired number of threads.
----------------------------	----------------------------

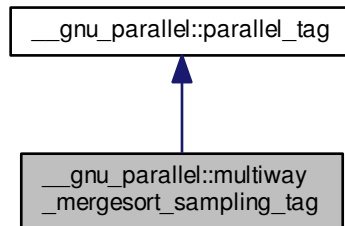
Definition at line 73 of file tags.h.

The documentation for this struct was generated from the following file:

- [tags.h](#)

4.155 `__gnu_parallel::multiway_mergesort_sampling_tag` Struct Reference

Inheritance diagram for `__gnu_parallel::multiway_mergesort_sampling_tag`:



Public Member Functions

- **`multiway_mergesort_sampling_tag`** (`_ThreadIndex` `__num_threads`)
- `_ThreadIndex` `__get_num_threads` ()
- void `set_num_threads` (`_ThreadIndex` `__num_threads`)

4.155.1 Detailed Description

Forces parallel sorting using multiway mergesort with splitting by sampling at compile time.

Definition at line 146 of file `tags.h`.

4.155.2 Member Function Documentation

4.155.2.1 `_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads` () `[inline]`, `[inherited]`

Find out desired number of threads.

Returns

Desired number of threads.

Definition at line 63 of file `tags.h`.

Referenced by `__gnu_parallel::__parallel_sort`(), `__gnu_parallel::multiway_merge`(), and `__gnu_parallel::multiway_merge_sentinels`().

4.155.2.2 void `__gnu_parallel::parallel_tag::set_num_threads` (`_ThreadIndex` `__num_threads`) `[inline]`, `[inherited]`

Set the desired number of threads.

Parameters

<code>__num_threads</code>	Desired number of threads.
----------------------------	----------------------------

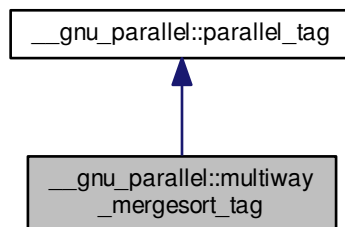
Definition at line 73 of file `tags.h`.

The documentation for this struct was generated from the following file:

- [tags.h](#)

4.156 `__gnu_parallel::multiway_mergesort_tag` Struct Reference

Inheritance diagram for `__gnu_parallel::multiway_mergesort_tag`:



Public Member Functions

- **`multiway_mergesort_tag`** ([_ThreadIndex](#) `__num_threads`)
- [_ThreadIndex](#) `__get_num_threads` ()
- void `set_num_threads` ([_ThreadIndex](#) `__num_threads`)

4.156.1 Detailed Description

Forces parallel sorting using multiway mergesort at compile time.

Definition at line 128 of file `tags.h`.

4.156.2 Member Function Documentation

4.156.2.1 `_ThreadIndex __gnu_parallel::parallel_tag::get_num_threads ()` `[inline]`, `[inherited]`

Find out desired number of threads.

Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by `__gnu_parallel::__parallel_sort()`, `__gnu_parallel::multiway_merge()`, and `__gnu_parallel::multiway_merge_sentinels()`.

4.156.2.2 `void __gnu_parallel::parallel_tag::set_num_threads (_ThreadIndex __num_threads)` `[inline]`, `[inherited]`

Set the desired number of threads.

Parameters

<code>__num_threads</code>	Desired number of threads.
----------------------------	----------------------------

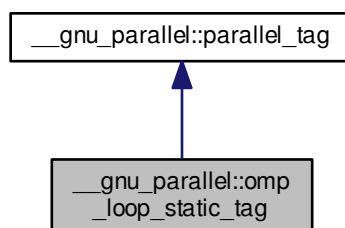
Definition at line 73 of file tags.h.

The documentation for this struct was generated from the following file:

- [tags.h](#)

4.157 `__gnu_parallel::omp_loop_static_tag` Struct Reference

Inheritance diagram for `__gnu_parallel::omp_loop_static_tag`:



Public Member Functions

- [_ThreadIndex __get_num_threads\(\)](#)
- void [set_num_threads\(_ThreadIndex __num_threads\)](#)

4.157.1 Detailed Description

Recommends parallel execution using OpenMP static load-balancing at compile time.

Definition at line 100 of file tags.h.

4.157.2 Member Function Documentation

4.157.2.1 `_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads()` `[inline]`, `[inherited]`

Find out desired number of threads.

Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by `__gnu_parallel::__parallel_sort()`, `__gnu_parallel::multiway_merge()`, and `__gnu_parallel::multiway_merge_sentinels()`.

4.157.2.2 `void __gnu_parallel::parallel_tag::set_num_threads(_ThreadIndex __num_threads)` `[inline]`, `[inherited]`

Set the desired number of threads.

Parameters

<code>__num_threads</code>	Desired number of threads.
----------------------------	----------------------------

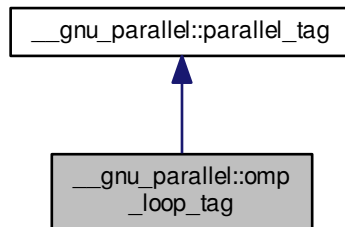
Definition at line 73 of file tags.h.

The documentation for this struct was generated from the following file:

- [tags.h](#)

4.158 `__gnu_parallel::omp_loop_tag` Struct Reference

Inheritance diagram for `__gnu_parallel::omp_loop_tag`:



Public Member Functions

- [_ThreadIndex __get_num_threads\(\)](#)
- void [set_num_threads\(_ThreadIndex __num_threads\)](#)

4.158.1 Detailed Description

Recommends parallel execution using OpenMP dynamic load-balancing at compile time.

Definition at line 96 of file tags.h.

4.158.2 Member Function Documentation

4.158.2.1 `_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads()` `[inline]`, `[inherited]`

Find out desired number of threads.

Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by `__gnu_parallel::__parallel_sort()`, `__gnu_parallel::multiway_merge()`, and `__gnu_parallel::multiway_merge_sentinels()`.

4.158.2.2 `void __gnu_parallel::parallel_tag::set_num_threads(_ThreadIndex __num_threads)` `[inline]`, `[inherited]`

Set the desired number of threads.

Parameters

<code>__num_threads</code>	Desired number of threads.
----------------------------	----------------------------

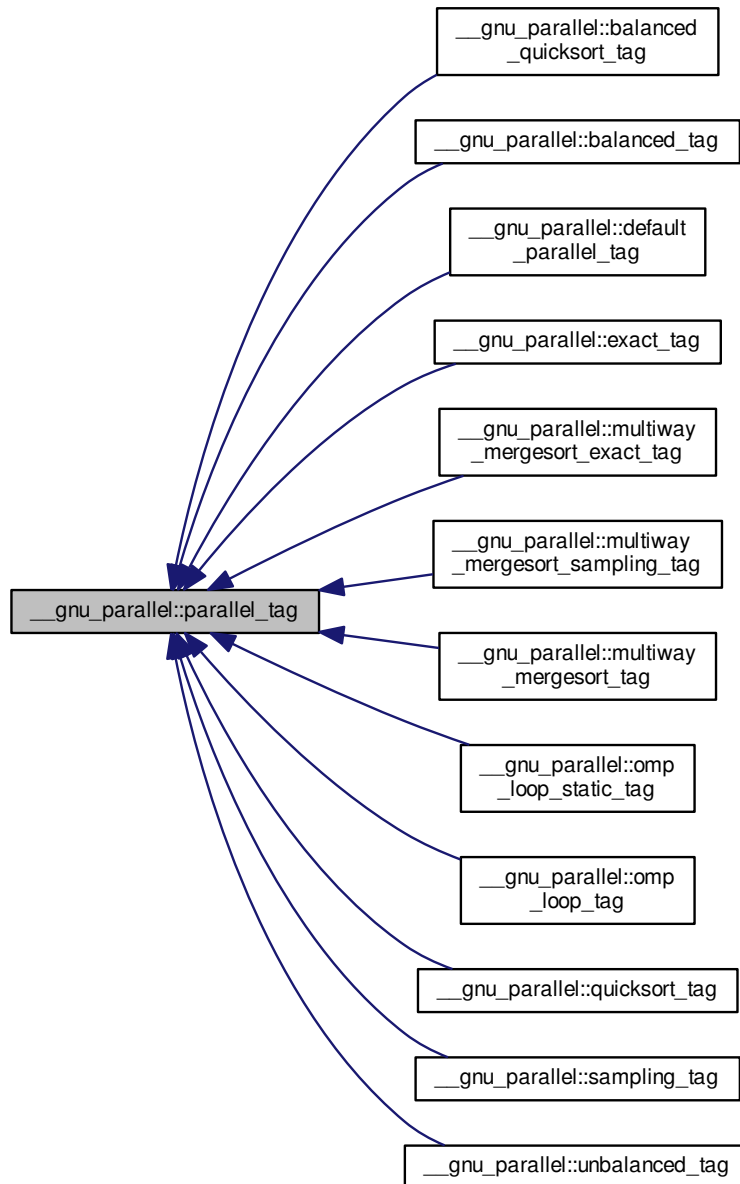
Definition at line 73 of file `tags.h`.

The documentation for this struct was generated from the following file:

- [tags.h](#)

4.159 __gnu_parallel::parallel_tag Struct Reference

Inheritance diagram for __gnu_parallel::parallel_tag:



Public Member Functions

- [parallel_tag](#) ()

- `parallel_tag` (`_ThreadIndex __num_threads`)
- `_ThreadIndex __get_num_threads` ()
- void `set_num_threads` (`_ThreadIndex __num_threads`)

4.159.1 Detailed Description

Recommends parallel execution at compile time, optionally using a user-specified number of threads.

Definition at line 46 of file `tags.h`.

4.159.2 Constructor & Destructor Documentation

4.159.2.1 `__gnu_parallel::parallel_tag::parallel_tag` () `[inline]`

Default constructor. Use default number of threads.

Definition at line 53 of file `tags.h`.

4.159.2.2 `__gnu_parallel::parallel_tag::parallel_tag` (`_ThreadIndex __num_threads`) `[inline]`

Default constructor. Recommend number of threads to use.

Parameters

<code>__num_threads</code>	Desired number of threads.
----------------------------	----------------------------

Definition at line 58 of file `tags.h`.

4.159.3 Member Function Documentation

4.159.3.1 `_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads` () `[inline]`

Find out desired number of threads.

Returns

Desired number of threads.

Definition at line 63 of file `tags.h`.

Referenced by `__gnu_parallel::__parallel_sort`(), `__gnu_parallel::multiway_merge`(), and `__gnu_parallel::multiway_merge_sentinels`().

4.159.3.2 void `__gnu_parallel::parallel_tag::set_num_threads` (`_ThreadIndex __num_threads`) `[inline]`

Set the desired number of threads.

Parameters

<code>__num_threads</code>	Desired number of threads.
----------------------------	----------------------------

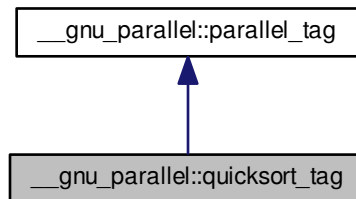
Definition at line 73 of file tags.h.

The documentation for this struct was generated from the following file:

- [tags.h](#)

4.160 `__gnu_parallel::quicksort_tag` Struct Reference

Inheritance diagram for `__gnu_parallel::quicksort_tag`:



Public Member Functions

- **`quicksort_tag`** (`_ThreadIndex` __num_threads)
- `_ThreadIndex` `__get_num_threads` ()
- void `set_num_threads` (`_ThreadIndex` __num_threads)

4.160.1 Detailed Description

Forces parallel sorting using unbalanced quicksort at compile time.

Definition at line 155 of file tags.h.

4.160.2 Member Function Documentation

4.160.2.1 _ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads () [inline],[inherited]

Find out desired number of threads.

Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by __gnu_parallel::__parallel_sort(), __gnu_parallel::multiway_merge(), and __gnu_parallel::multiway_merge_sentinels().

4.160.2.2 void __gnu_parallel::parallel_tag::set_num_threads (_ThreadIndex __num_threads) [inline],[inherited]

Set the desired number of threads.

Parameters

<code>__num_threads</code>	Desired number of threads.
----------------------------	----------------------------

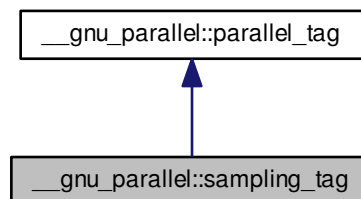
Definition at line 73 of file tags.h.

The documentation for this struct was generated from the following file:

- [tags.h](#)

4.161 __gnu_parallel::sampling_tag Struct Reference

Inheritance diagram for __gnu_parallel::sampling_tag:



Public Member Functions

- **sampling_tag** ([_ThreadIndex](#) __num_threads)
- [_ThreadIndex](#) **__get_num_threads** ()
- void **set_num_threads** ([_ThreadIndex](#) __num_threads)

4.161.1 Detailed Description

Forces parallel merging with exact splitting, at compile time.

Definition at line 118 of file tags.h.

4.161.2 Member Function Documentation

4.161.2.1 [_ThreadIndex](#) **__gnu_parallel::parallel_tag::__get_num_threads** () [\[inline\]](#), [\[inherited\]](#)

Find out desired number of threads.

Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by [__gnu_parallel::__parallel_sort\(\)](#), [__gnu_parallel::multiway_merge\(\)](#), and [__gnu_parallel::multiway_merge_sentinels\(\)](#).

4.161.2.2 void **__gnu_parallel::parallel_tag::set_num_threads** ([_ThreadIndex](#) __num_threads) [\[inline\]](#), [\[inherited\]](#)

Set the desired number of threads.

Parameters

__num_threads	Desired number of threads.
-------------------------------	----------------------------

Definition at line 73 of file tags.h.

The documentation for this struct was generated from the following file:

- [tags.h](#)

4.162 [__gnu_parallel::sequential_tag](#) Struct Reference

4.162.1 Detailed Description

Forces sequential execution at compile time.

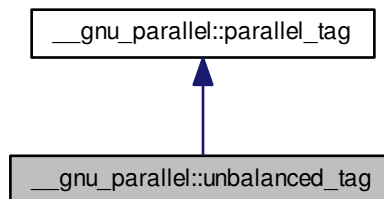
Definition at line 42 of file tags.h.

The documentation for this struct was generated from the following file:

- [tags.h](#)

4.163 `__gnu_parallel::unbalanced_tag` Struct Reference

Inheritance diagram for `__gnu_parallel::unbalanced_tag`:



Public Member Functions

- [_ThreadIndex __get_num_threads\(\)](#)
- [void set_num_threads\(_ThreadIndex __num_threads\)](#)

4.163.1 Detailed Description

Recommends parallel execution using static load-balancing at compile time.

Definition at line 92 of file tags.h.

4.163.2 Member Function Documentation

4.163.2.1 `_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads()` `[inline]`, `[inherited]`

Find out desired number of threads.

Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by `__gnu_parallel::__parallel_sort()`, `__gnu_parallel::multiway_merge()`, and `__gnu_parallel::multiway_merge_sentinels()`.

4.163.2.2 void __gnu_parallel::parallel_tag::set_num_threads (_ThreadIndex __num_threads) [inline],
[inherited]

Set the desired number of threads.

Parameters

<code>__num_threads</code>	Desired number of threads.
----------------------------	----------------------------

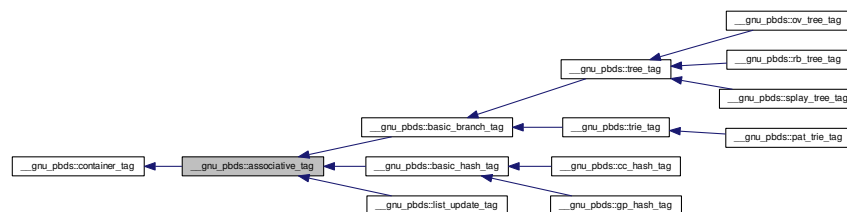
Definition at line 73 of file tags.h.

The documentation for this struct was generated from the following file:

- [tags.h](#)

4.164 __gnu_pbds::associative_tag Struct Reference

Inheritance diagram for __gnu_pbds::associative_tag:



4.164.1 Detailed Description

Basic associative-container.

Definition at line 135 of file tag_and_trait.hpp.

The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

4.165 __gnu_pbds::basic_branch< Key, Mapped, Tag, Node_Update, Policy_Tl, _Alloc > Class Template Reference

Inherits type< Key, Mapped, _Alloc, Tag, Policy_Tl >.

Public Types

- typedef Node_Update **node_update**

Protected Member Functions

- **basic_branch** (const [basic_branch](#) &other)
- template<typename T0 >
basic_branch (T0 t0)
- template<typename T0 , typename T1 >
basic_branch (T0 t0, T1 t1)
- template<typename T0 , typename T1 , typename T2 >
basic_branch (T0 t0, T1 t1, T2 t2)
- template<typename T0 , typename T1 , typename T2 , typename T3 >
basic_branch (T0 t0, T1 t1, T2 t2, T3 t3)
- template<typename T0 , typename T1 , typename T2 , typename T3 , typename T4 >
basic_branch (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4)
- template<typename T0 , typename T1 , typename T2 , typename T3 , typename T4 , typename T5 >
basic_branch (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4, T5 t5)
- template<typename T0 , typename T1 , typename T2 , typename T3 , typename T4 , typename T5 , typename T6 >
basic_branch (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4, T5 t5, T6 t6)

4.165.1 Detailed Description

```
template<typename Key, typename Mapped, typename Tag, typename Node_Update, typename Policy_Tl, typename _Alloc>
class __gnu_pbds::basic_branch< Key, Mapped, Tag, Node_Update, Policy_Tl, _Alloc >
```

A branched, tree-like (tree, trie) container abstraction.

Template Parameters

<i>Key</i>	Key type.
<i>Mapped</i>	Map type.
<i>Tag</i>	Instantiating data structure type, see <code>container_tag</code> .
<i>Node_Update</i>	Updates nodes, restores invariants.
<i>Policy_Tl</i>	Policy typelist.
<i>_Alloc</i>	Allocator type.

Base is dispatched at compile time via Tag, from the following choices: `tree_tag`, `trie_tag`, and their descendants.

Base choices are: `detail::ov_tree_map`, `detail::rb_tree_map`, `detail::splay_tree_map`, and `detail::pat_trie_map`.

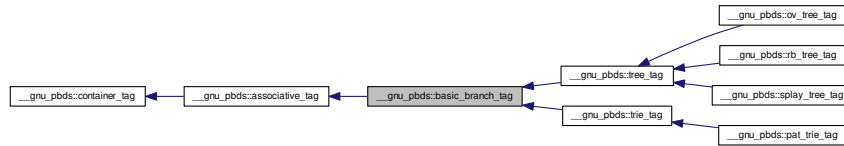
Definition at line 555 of file `assoc_container.hpp`.

The documentation for this class was generated from the following file:

- [assoc_container.hpp](#)

4.166 `__gnu_pbds::basic_branch_tag` Struct Reference

Inheritance diagram for `__gnu_pbds::basic_branch_tag`:



4.166.1 Detailed Description

Basic branch structure.

Definition at line 147 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

4.167 `__gnu_pbds::basic_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Resize_Policy, Store_Hash, Tag, Policy_Tl, _Alloc >` Class Template Reference

Inherits `type< Key, Mapped, _Alloc, Tag, __gnu_cxx::typelist::append< __gnu_cxx::typelist::create4< Hash_Fn, Eq_Fn, Resize_Policy, detail::integral_constant< int, Store_Hash > >::type, Policy_Tl >::type >`.

Protected Member Functions

- **`basic_hash_table`** (const [basic_hash_table](#) &other)
- `template<typename T0 >`
`basic_hash_table` (T0 t0)
- `template<typename T0 , typename T1 >`
`basic_hash_table` (T0 t0, T1 t1)
- `template<typename T0 , typename T1 , typename T2 >`
`basic_hash_table` (T0 t0, T1 t1, T2 t2)
- `template<typename T0 , typename T1 , typename T2 , typename T3 >`
`basic_hash_table` (T0 t0, T1 t1, T2 t2, T3 t3)
- `template<typename T0 , typename T1 , typename T2 , typename T3 , typename T4 >`
`basic_hash_table` (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4)
- `template<typename T0 , typename T1 , typename T2 , typename T3 , typename T4 , typename T5 >`
`basic_hash_table` (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4, T5 t5)
- `template<typename T0 , typename T1 , typename T2 , typename T3 , typename T4 , typename T5 , typename T6 >`
`basic_hash_table` (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4, T5 t5, T6 t6)
- `template<typename T0 , typename T1 , typename T2 , typename T3 , typename T4 , typename T5 , typename T6 , typename T7 >`
`basic_hash_table` (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4, T5 t5, T6 t6, T7 t7)
- `template<typename T0 , typename T1 , typename T2 , typename T3 , typename T4 , typename T5 , typename T6 , typename T7 , typename T8 >`
`basic_hash_table` (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4, T5 t5, T6 t6, T7 t7, T8 t8)

4.167.1 Detailed Description

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename Resize_Policy, bool Store_Hash, type-
name Tag, typename Policy_Tl, typename _Alloc>
class __gnu_pbds::basic_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Resize_Policy, Store_Hash, Tag, Policy_Tl, _Alloc >
```

A hashed container abstraction.

Template Parameters

<i>Key</i>	Key type.
<i>Mapped</i>	Map type.
<i>Hash_Fn</i>	Hashing functor.
<i>Eq_Fn</i>	Equal functor.
<i>Resize_Policy</i>	Resizes hash.
<i>Store_Hash</i>	Indicates whether the hash value will be stored along with each key.
<i>Tag</i>	Instantiating data structure type, see <code>container_tag</code> .
<i>Policy_Tl</i>	Policy typelist.
<i>_Alloc</i>	Allocator type.

Base is dispatched at compile time via `Tag`, from the following choices: `cc_hash_tag`, `gp_hash_tag`, and descendants of `basic_hash_tag`.

Base choices are: `detail::cc_ht_map`, `detail::gp_ht_map`

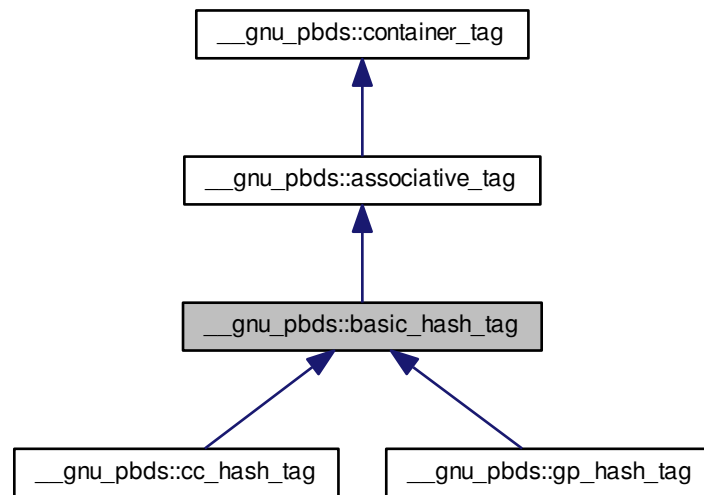
Definition at line 104 of file `assoc_container.hpp`.

The documentation for this class was generated from the following file:

- [assoc_container.hpp](#)

4.168 `__gnu_pbds::basic_hash_tag` Struct Reference

Inheritance diagram for `__gnu_pbds::basic_hash_tag`:



4.168.1 Detailed Description

Basic hash structure.

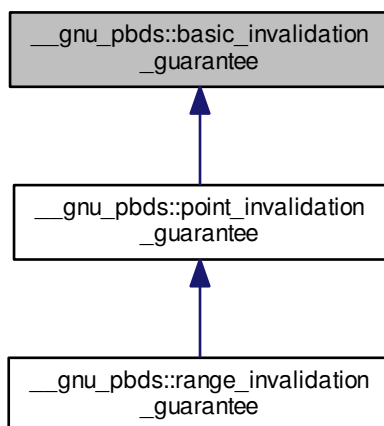
Definition at line 138 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

4.169 `__gnu_pbds::basic_invalidation_guarantee` Struct Reference

Inheritance diagram for `__gnu_pbds::basic_invalidation_guarantee`:



4.169.1 Detailed Description

Signifies a basic invalidation guarantee that any iterator, pointer, or reference to a container object's mapped value type is valid as long as the container is not modified.

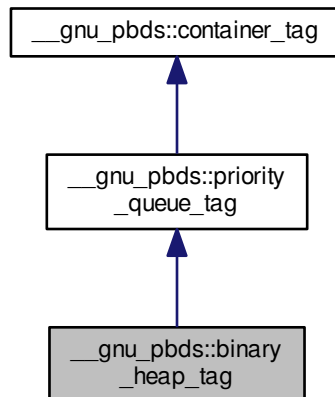
Definition at line 93 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

4.170 `__gnu_pbds::binary_heap_tag` Struct Reference

Inheritance diagram for `__gnu_pbds::binary_heap_tag`:



4.170.1 Detailed Description

Binary-heap (array-based).

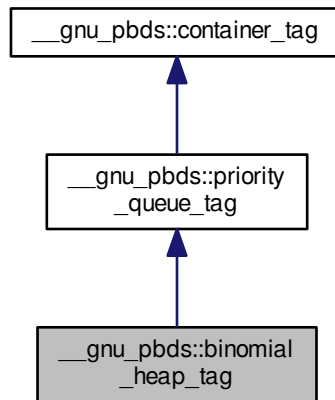
Definition at line 183 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

4.171 `__gnu_pbds::binomial_heap_tag` Struct Reference

Inheritance diagram for `__gnu_pbds::binomial_heap_tag`:



4.171.1 Detailed Description

Binomial-heap.

Definition at line 177 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

4.172 `__gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >`
Class Template Reference

Public Types

- enum { [external_load_access](#) }
- typedef `Size_Type` **size_type**

Public Member Functions

- [cc_hash_max_collision_check_resize_trigger](#) (float load=0.5)
- float [get_load](#) () const
- void [set_load](#) (float load)
- void **swap** ([cc_hash_max_collision_check_resize_trigger](#)< `External_Load_Access`, `Size_Type` > &other)

Protected Member Functions

- bool `is_grow_needed` (size_type size, size_type num_entries) const
- bool `is_resize_needed` () const
- void `notify_cleared` ()
- void `notify_erase_search_collision` ()
- void `notify_erase_search_end` ()
- void `notify_erase_search_start` ()
- void `notify_erased` (size_type num_entries)
- void `notify_externally_resized` (size_type new_size)
- void `notify_find_search_collision` ()
- void `notify_find_search_end` ()
- void `notify_find_search_start` ()
- void `notify_insert_search_collision` ()
- void `notify_insert_search_end` ()
- void `notify_insert_search_start` ()
- void `notify_inserted` (size_type num_entries)
- void `notify_resized` (size_type new_size)

4.172.1 Detailed Description

```
template<bool External_Load_Access = false, typename Size_Type = std::size_t>
class __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >
```

A resize trigger policy based on collision checks. It keeps the simulated load factor lower than some given load factor.

Definition at line 293 of file hash_policy.hpp.

4.172.2 Member Enumeration Documentation

4.172.2.1 `template<bool External_Load_Access = false, typename Size_Type = std::size_t> anonymous enum`

Enumerator

`external_load_access` Specifies whether the load factor can be accessed externally. The two options have different trade-offs in terms of flexibility, genericity, and encapsulation.

Definition at line 298 of file hash_policy.hpp.

4.172.3 Constructor & Destructor Documentation

4.172.3.1 `template<bool External_Load_Access, typename Size_Type > __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >::cc_hash_max_collision_check_resize_trigger (float load = 0.5)`

Default constructor, or constructor taking load, a `__load` factor which it will attempt to maintain.

Definition at line 44 of file hash_policy.hpp.

4.172.4 Member Function Documentation

4.172.4.1 `template<bool External_Load_Access, typename Size_Type > float __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >::get_load () const`
[inline]

Returns the current load.

Definition at line 190 of file hash_policy.hpp.

4.172.4.2 `template<bool External_Load_Access, typename Size_Type > bool __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >::is_grow_needed (size_type size, size_type num_entries) const` [inline],[protected]

Queries whether a grow is needed. This method is called only if this object indicated is needed.

Definition at line 133 of file hash_policy.hpp.

4.172.4.3 `template<bool External_Load_Access, typename Size_Type > bool __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >::is_resize_needed () const` [inline],[protected]

Queries whether a resize is needed.

Definition at line 127 of file hash_policy.hpp.

4.172.4.4 `template<bool External_Load_Access, typename Size_Type > void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >::notify_cleared ()`
[protected]

Notifies the table was cleared.

Definition at line 121 of file hash_policy.hpp.

4.172.4.5 `template<bool External_Load_Access, typename Size_Type > void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >::notify_erase_search_collision ()` [inline],[protected]

Notifies a search encountered a collision.

Definition at line 97 of file hash_policy.hpp.

4.172.4.6 `template<bool External_Load_Access, typename Size_Type > void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >::notify_erase_search_end ()` [inline],[protected]

Notifies a search ended.

Definition at line 103 of file hash_policy.hpp.

4.172.4.7 `template<bool External_Load_Access, typename Size_Type > void __gnu_pbds::cc_hash_max_collision_↵
check_resize_trigger< External_Load_Access, Size_Type >::notify_erase_search_start () [inline],
[protected]`

Notifies an erase search started.

Definition at line 91 of file hash_policy.hpp.

4.172.4.8 `template<bool External_Load_Access, typename Size_Type > void __gnu_pbds::cc_hash_max_collision_↵
_check_resize_trigger< External_Load_Access, Size_Type >::notify_erased (size_type num_entries)
[inline], [protected]`

Notifies an element was erased.

Definition at line 115 of file hash_policy.hpp.

4.172.4.9 `template<bool External_Load_Access, typename Size_Type > void __gnu_pbds::cc_hash_max_collision_↵
_check_resize_trigger< External_Load_Access, Size_Type >::notify_externally_resized (size_type new_size)
[protected]`

Notifies the table was resized externally.

Definition at line 172 of file hash_policy.hpp.

4.172.4.10 `template<bool External_Load_Access, typename Size_Type > void __gnu_pbds::cc_hash_max_collision_↵
check_resize_trigger< External_Load_Access, Size_Type >::notify_find_search_collision () [inline],
[protected]`

Notifies a search encountered a collision.

Definition at line 61 of file hash_policy.hpp.

4.172.4.11 `template<bool External_Load_Access, typename Size_Type > void __gnu_pbds::cc_hash_max_collision_↵
check_resize_trigger< External_Load_Access, Size_Type >::notify_find_search_end () [inline],
[protected]`

Notifies a search ended.

Definition at line 67 of file hash_policy.hpp.

4.172.4.12 `template<bool External_Load_Access, typename Size_Type > void __gnu_pbds::cc_hash_max_collision_↵
check_resize_trigger< External_Load_Access, Size_Type >::notify_find_search_start () [inline],
[protected]`

Notifies a find search started.

Definition at line 55 of file hash_policy.hpp.

4.172.4.13 `template<bool External_Load_Access, typename Size_Type > void __gnu_pbds::cc_hash_max_collision_↵
_check_resize_trigger< External_Load_Access, Size_Type >::notify_insert_search_collision () [inline],
[protected]`

Notifies a search encountered a collision.

Definition at line 79 of file `hash_policy.hpp`.

4.172.4.14 `template<bool External_Load_Access, typename Size_Type > void __gnu_pbds::cc_hash_max_collision_↵
_check_resize_trigger< External_Load_Access, Size_Type >::notify_insert_search_end () [inline],
[protected]`

Notifies a search ended.

Definition at line 85 of file `hash_policy.hpp`.

4.172.4.15 `template<bool External_Load_Access, typename Size_Type > void __gnu_pbds::cc_hash_max_collision_↵
_check_resize_trigger< External_Load_Access, Size_Type >::notify_insert_search_start () [inline],
[protected]`

Notifies an insert search started.

Definition at line 73 of file `hash_policy.hpp`.

4.172.4.16 `template<bool External_Load_Access, typename Size_Type > void __gnu_pbds::cc_hash_max_collision_↵
_check_resize_trigger< External_Load_Access, Size_Type >::notify_inserted (size_type num_entries)
[inline], [protected]`

Notifies an element was inserted.

Definition at line 109 of file `hash_policy.hpp`.

4.172.4.17 `template<bool External_Load_Access, typename Size_Type > void __gnu_pbds::cc_hash_max_collision_↵
_check_resize_trigger< External_Load_Access, Size_Type >::notify_resized (size_type new_size)
[protected]`

Notifies the table was resized as a result of this object's signifying that a resize is needed.

Definition at line 139 of file `hash_policy.hpp`.

4.172.4.18 `template<bool External_Load_Access, typename Size_Type > void __gnu_pbds::cc_hash_max_↵
_collision_check_resize_trigger< External_Load_Access, Size_Type >::set_load (float load
)`

Sets the load; does not resize the container.

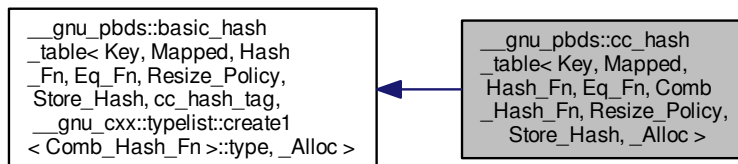
Definition at line 205 of file `hash_policy.hpp`.

The documentation for this class was generated from the following file:

- [hash_policy.hpp](#)

4.173 `__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc >`:



Public Types

- typedef Comb_Hash_Fn **comb_hash_fn**
- typedef `cc_hash_tag` **container_category**
- typedef Eq_Fn **eq_fn**
- typedef Hash_Fn **hash_fn**
- typedef Resize_Policy **resize_policy**

Public Member Functions

- `cc_hash_table` ()
- `cc_hash_table` (const hash_fn &h)
- `cc_hash_table` (const hash_fn &h, const eq_fn &e)
- `cc_hash_table` (const hash_fn &h, const eq_fn &e, const comb_hash_fn &ch)
- `cc_hash_table` (const hash_fn &h, const eq_fn &e, const comb_hash_fn &ch, const resize_policy &rp)
- template<typename It >
`cc_hash_table` (It first, It last)
- template<typename It >
`cc_hash_table` (It first, It last, const hash_fn &h)
- template<typename It >
`cc_hash_table` (It first, It last, const hash_fn &h, const eq_fn &e)
- template<typename It >
`cc_hash_table` (It first, It last, const hash_fn &h, const eq_fn &e, const comb_hash_fn &ch)
- template<typename It >
`cc_hash_table` (It first, It last, const hash_fn &h, const eq_fn &e, const comb_hash_fn &ch, const resize_policy &rp)
- **cc_hash_table** (const `cc_hash_table` &other)
- `cc_hash_table` & **operator=** (const `cc_hash_table` &other)
- void **swap** (`cc_hash_table` &other)

4.173.1 Detailed Description

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn
= typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn = detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash, typename
_Alloc = std::allocator<char>>
class __gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc >
```

A collision-chaining hash-based associative container.

Template Parameters

<i>Key</i>	Key type.
<i>Mapped</i>	Map type.
<i>Hash_Fn</i>	Hashing functor.
<i>Eq_Fn</i>	Equal functor.
<i>Comb_Hash_Fn</i>	Combining hash functor. If <i>Hash_Fn</i> is not null_type, then this is the ranged-hash functor; otherwise, this is the range-hashing functor. XXX(See Design::Hash-Based Containers::Hash Policies.)
<i>Resize_Policy</i>	Resizes hash.
<i>Store_Hash</i>	Indicates whether the hash value will be stored along with each key. If <i>Hash_Fn</i> is null_type, then the container will not compile if this value is true
<i>_Alloc</i>	Allocator type.

Base tag choices are: `cc_hash_tag`.

Base is `basic_hash_table`.

Definition at line 204 of file `assoc_container.hpp`.

4.173.2 Constructor & Destructor Documentation

4.173.2.1 `template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn = detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> __gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc >::cc_hash_table () [inline]`

Default constructor.

Definition at line 217 of file `assoc_container.hpp`.

4.173.2.2 `template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn = detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> __gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc >::cc_hash_table (const hash_fn & h) [inline]`

Constructor taking some policy objects. `r_hash_fn` will be copied by the `Hash_Fn` object of the container object.

Definition at line 221 of file `assoc_container.hpp`.

4.173.2.3 `template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn = detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> __gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc >::cc_hash_table (const hash_fn & h, const eq_fn & e) [inline]`

Constructor taking some policy objects. `r_hash_fn` will be copied by the `hash_fn` object of the container object, and `r_eq_fn` will be copied by the `eq_fn` object of the container object.

Definition at line 228 of file `assoc_container.hpp`.

4.173.2.4 `template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn = detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> __gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc >::cc_hash_table (const hash_fn & h, const eq_fn & e, const comb_hash_fn & ch) [inline]`

Constructor taking some policy objects. `r_hash_fn` will be copied by the `hash_fn` object of the container object, `r_eq_fn` will be copied by the `eq_fn` object of the container object, and `r_comb_hash_fn` will be copied by the `comb_hash_fn` object of the container object.

Definition at line 236 of file `assoc_container.hpp`.

4.173.2.5 `template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn = detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> __gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc >::cc_hash_table (const hash_fn & h, const eq_fn & e, const comb_hash_fn & ch, const resize_policy & rp) [inline]`

Constructor taking some policy objects. `r_hash_fn` will be copied by the `hash_fn` object of the container object, `r_eq_fn` will be copied by the `eq_fn` object of the container object, `r_comb_hash_fn` will be copied by the `comb_hash_fn` object of the container object, and `r_resize_policy` will be copied by the `resize_policy` object of the container object.

Definition at line 245 of file `assoc_container.hpp`.

4.173.2.6 `template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn = detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> template<typename It > __gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc >::cc_hash_table (It first, It last) [inline]`

Constructor taking `__iterators` to a range of `value_types`. The `value_types` between `first_it` and `last_it` will be inserted into the container object.

Definition at line 253 of file `assoc_container.hpp`.


```
4.173.2.7 template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type,
typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn = detail::default_comb_
_hash_fn::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Hash_Fn>::type, bool
Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> template<typename It >
__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc
>::cc_hash_table ( It first, It last, const hash_fn & h ) [inline]
```

Constructor taking `__` iterators to a range of `value_types` and some policy objects. The `value_types` between `first_it` and `last_it` will be inserted into the container object.

Definition at line 260 of file `assoc_container.hpp`.

```
4.173.2.8 template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type,
typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn = detail::default_comb_
_hash_fn::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Hash_Fn>::type, bool
Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> template<typename It >
__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc
>::cc_hash_table ( It first, It last, const hash_fn & h, const eq_fn & e ) [inline]
```

Constructor taking `__` iterators to a range of `value_types` and some policy objects. The `value_types` between `first_it` and `last_it` will be inserted into the container object. `r_hash_fn` will be copied by the `hash_fn` object of the container object, and `r_eq_fn` will be copied by the `eq_fn` object of the container object.

Definition at line 271 of file `assoc_container.hpp`.

```
4.173.2.9 template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type,
typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn = detail::default_comb_
_hash_fn::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Hash_Fn>::type, bool
Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> template<typename It >
__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc
>::cc_hash_table ( It first, It last, const hash_fn & h, const eq_fn & e, const comb_hash_fn & ch ) [inline]
```

Constructor taking `__` iterators to a range of `value_types` and some policy objects. The `value_types` between `first_it` and `last_it` will be inserted into the container object. `r_hash_fn` will be copied by the `hash_fn` object of the container object, `r_eq_fn` will be copied by the `eq_fn` object of the container object, and `r_comb_hash_fn` will be copied by the `comb_hash_fn` object of the container object.

Definition at line 283 of file `assoc_container.hpp`.

```
4.173.2.10 template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type,
typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn = detail::default_
comb_hash_fn::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Hash_Fn>::type,
bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> template<typename It >
__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash,
_Alloc >::cc_hash_table ( It first, It last, const hash_fn & h, const eq_fn & e, const comb_hash_fn & ch, const
resize_policy & rp ) [inline]
```

Constructor taking `__` iterators to a range of `value_types` and some policy objects. The `value_types` between `first_it` and `last_it` will be inserted into the container object. `r_hash_fn` will be copied by the `hash_fn` object of the container object, `r_eq_fn` will be copied by the `eq_fn` object of the container object, `r_comb_hash_fn` will be copied by the `comb_hash_fn` object of the container object, and `r_resize_policy` will be copied by the `resize_policy` object of the container object.

Definition at line 297 of file `assoc_container.hpp`.

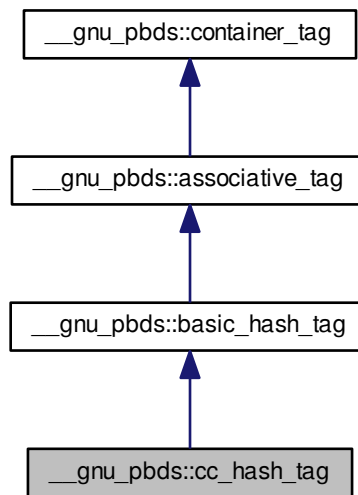
References `std::swap()`.

The documentation for this class was generated from the following file:

- [assoc_container.hpp](#)

4.174 `__gnu_pbds::cc_hash_tag` Struct Reference

Inheritance diagram for `__gnu_pbds::cc_hash_tag`:



4.174.1 Detailed Description

Collision-chaining hash.

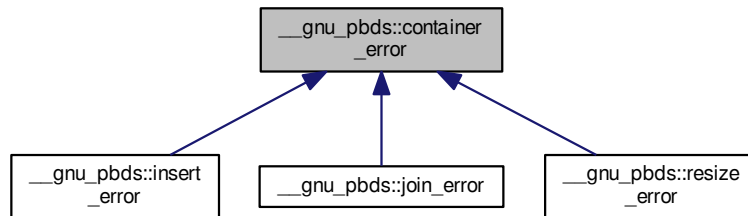
Definition at line 141 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

4.175 `__gnu_pbds::container_error` Struct Reference

Inheritance diagram for `__gnu_pbds::container_error`:



4.175.1 Detailed Description

Base class for exceptions.

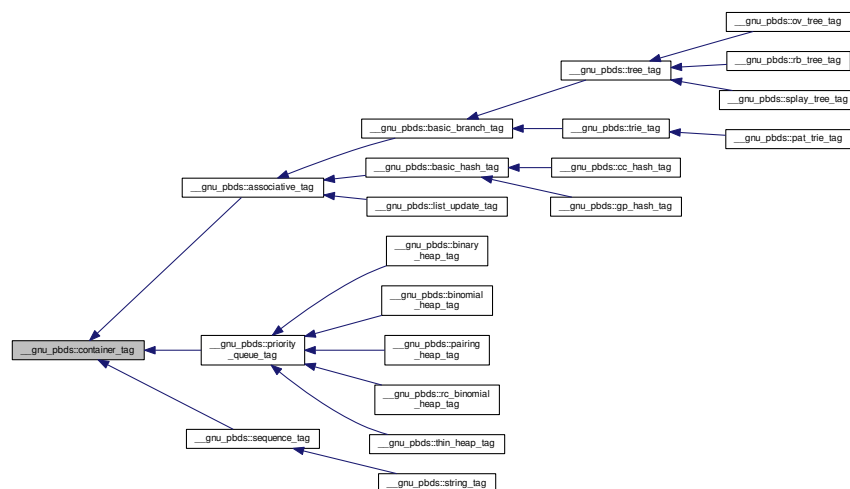
Definition at line 57 of file `exception.hpp`.

The documentation for this struct was generated from the following file:

- [exception.hpp](#)

4.176 `__gnu_pbds::container_tag` Struct Reference

Inheritance diagram for `__gnu_pbds::container_tag`:



4.176.1 Detailed Description

Base data structure tag.

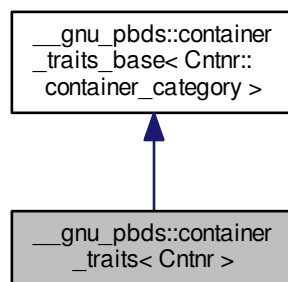
Definition at line 125 of file tag_and_trait.hpp.

The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

4.177 `__gnu_pbds::container_traits< Cntnr >` Struct Template Reference

Inheritance diagram for `__gnu_pbds::container_traits< Cntnr >`:



Public Types

- enum { [order_preserving](#), [erase_can_throw](#), [split_join_can_throw](#), [reverse_iteration](#) }
- typedef [container_traits_base< container_category >](#) **base_type**
- typedef `Cntnr::container_category` **container_category**
- typedef `Cntnr` **container_type**
- typedef `base_type::invalidation_guarantee` **invalidation_guarantee**

4.177.1 Detailed Description

```
template<typename Cntnr>
struct __gnu_pbds::container_traits< Cntnr >
```

Container traits.

Definition at line 418 of file tag_and_trait.hpp.

4.177.2 Member Enumeration Documentation

4.177.2.1 `template<typename Cntnr> anonymous enum`

Enumerator

- `order_preserving`** True only if Cntnr objects guarantee storing keys by order.
- `erase_can_throw`** True only if erasing a key can throw.
- `split_join_can_throw`** True only if split or join operations can throw.
- `reverse_iteration`** True only reverse iterators are supported.

Definition at line 426 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

4.178 `__gnu_pbds::container_traits_base<_Tag>` Struct Template Reference

4.178.1 Detailed Description

```
template<typename _Tag>
struct __gnu_pbds::container_traits_base<_Tag>
```

Primary template, container traits base.

Definition at line 220 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

4.179 `__gnu_pbds::container_traits_base<binary_heap_tag>` Struct Template Reference

Public Types

- enum { **`order_preserving`**, **`erase_can_throw`**, **`split_join_can_throw`**, **`reverse_iteration`** }
- typedef [binary_heap_tag](#) **`container_category`**
- typedef [basic_invalidation_guarantee](#) **`invalidation_guarantee`**

4.179.1 Detailed Description

```
template<>
struct __gnu_pbds::container_traits_base< binary_heap_tag >
```

Specialization, binary heap.

Definition at line 400 of file tag_and_trait.hpp.

The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

4.180 __gnu_pbds::container_traits_base< binomial_heap_tag > Struct Template Reference

Public Types

- enum { **order_preserving**, **erase_can_throw**, **split_join_can_throw**, **reverse_iteration** }
- typedef [binomial_heap_tag](#) **container_category**
- typedef [point_invalidation_guarantee](#) **invalidation_guarantee**

4.180.1 Detailed Description

```
template<>
struct __gnu_pbds::container_traits_base< binomial_heap_tag >
```

Specialization, binomial heap.

Definition at line 368 of file tag_and_trait.hpp.

The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

4.181 __gnu_pbds::container_traits_base< cc_hash_tag > Struct Template Reference

Public Types

- enum { **order_preserving**, **erase_can_throw**, **split_join_can_throw**, **reverse_iteration** }
- typedef [cc_hash_tag](#) **container_category**
- typedef [point_invalidation_guarantee](#) **invalidation_guarantee**

4.181.1 Detailed Description

```
template<>
struct __gnu_pbds::container_traits_base< cc_hash_tag >
```

Specialization, cc hash.

Definition at line 224 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

4.182 `__gnu_pbds::container_traits_base< gp_hash_tag >` Struct Template Reference

Public Types

- enum { `order_preserving`, `erase_can_throw`, `split_join_can_throw`, `reverse_iteration` }
- typedef `gp_hash_tag` `container_category`
- typedef `basic_invalidation_guarantee` `invalidation_guarantee`

4.182.1 Detailed Description

```
template<>
struct __gnu_pbds::container_traits_base< gp_hash_tag >
```

Specialization, gp hash.

Definition at line 240 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

4.183 `__gnu_pbds::container_traits_base< list_update_tag >` Struct Template Reference

Public Types

- enum { `order_preserving`, `erase_can_throw`, `split_join_can_throw`, `reverse_iteration` }
- typedef `list_update_tag` `container_category`
- typedef `point_invalidation_guarantee` `invalidation_guarantee`

4.183.1 Detailed Description

```
template<>
struct __gnu_pbds::container_traits_base< list_update_tag >
```

Specialization, list update.

Definition at line 320 of file tag_and_trait.hpp.

The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

4.184 __gnu_pbds::container_traits_base< ov_tree_tag > Struct Template Reference

Public Types

- enum { **order_preserving**, **erase_can_throw**, **split_join_can_throw**, **reverse_iteration** }
- typedef [ov_tree_tag](#) **container_category**
- typedef [basic_invalidation_guarantee](#) **invalidation_guarantee**

4.184.1 Detailed Description

```
template<>
struct __gnu_pbds::container_traits_base< ov_tree_tag >
```

Specialization, ov tree.

Definition at line 288 of file tag_and_trait.hpp.

The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

4.185 __gnu_pbds::container_traits_base< pairing_heap_tag > Struct Template Reference

Public Types

- enum { **order_preserving**, **erase_can_throw**, **split_join_can_throw**, **reverse_iteration** }
- typedef [pairing_heap_tag](#) **container_category**
- typedef [point_invalidation_guarantee](#) **invalidation_guarantee**

4.185.1 Detailed Description

```
template<>
struct __gnu_pbds::container_traits_base< pairing_heap_tag >
```

Specialization, pairing heap.

Definition at line 336 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

4.186 `__gnu_pbds::container_traits_base< pat_trie_tag >` Struct Template Reference

Public Types

- enum { `order_preserving`, `erase_can_throw`, `split_join_can_throw`, `reverse_iteration` }
- typedef [pat_trie_tag](#) `container_category`
- typedef [range_invalidation_guarantee](#) `invalidation_guarantee`

4.186.1 Detailed Description

```
template<>
struct __gnu_pbds::container_traits_base< pat_trie_tag >
```

Specialization, pat trie.

Definition at line 304 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

4.187 `__gnu_pbds::container_traits_base< rb_tree_tag >` Struct Template Reference

Public Types

- enum { `order_preserving`, `erase_can_throw`, `split_join_can_throw`, `reverse_iteration` }
- typedef [rb_tree_tag](#) `container_category`
- typedef [range_invalidation_guarantee](#) `invalidation_guarantee`

4.187.1 Detailed Description

```
template<>
struct __gnu_pbds::container_traits_base< rb_tree_tag >
```

Specialization, rb tree.

Definition at line 256 of file tag_and_trait.hpp.

The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

4.188 __gnu_pbds::container_traits_base< rc_binomial_heap_tag > Struct Template Reference

Public Types

- enum { **order_preserving**, **erase_can_throw**, **split_join_can_throw**, **reverse_iteration** }
- typedef [rc_binomial_heap_tag](#) **container_category**
- typedef [point_invalidation_guarantee](#) **invalidation_guarantee**

4.188.1 Detailed Description

```
template<>
struct __gnu_pbds::container_traits_base< rc_binomial_heap_tag >
```

Specialization, rc binomial heap.

Definition at line 384 of file tag_and_trait.hpp.

The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

4.189 __gnu_pbds::container_traits_base< splay_tree_tag > Struct Template Reference

Public Types

- enum { **order_preserving**, **erase_can_throw**, **split_join_can_throw**, **reverse_iteration** }
- typedef [splay_tree_tag](#) **container_category**
- typedef [range_invalidation_guarantee](#) **invalidation_guarantee**

4.189.1 Detailed Description

```
template<>
struct __gnu_pbds::container_traits_base< splay_tree_tag >
```

Specialization, splay tree.

Definition at line 272 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

4.190 `__gnu_pbds::container_traits_base< thin_heap_tag >` Struct Template Reference

Public Types

- enum { **order_preserving**, **erase_can_throw**, **split_join_can_throw**, **reverse_iteration** }
- typedef [thin_heap_tag](#) **container_category**
- typedef [point_invalidation_guarantee](#) **invalidation_guarantee**

4.190.1 Detailed Description

```
template<>
struct __gnu_pbds::container_traits_base< thin_heap_tag >
```

Specialization, thin heap.

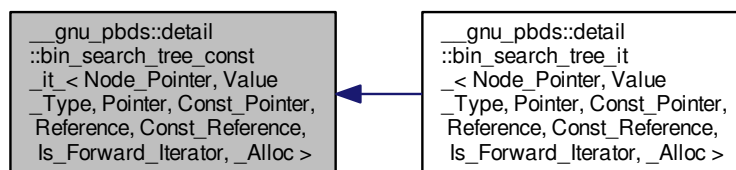
Definition at line 352 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

4.191 `__gnu_pbds::detail::bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc >`:



Public Types

- typedef Const_Pointer **const_pointer**
- typedef Const_Reference **const_reference**
- typedef _Alloc::difference_type **difference_type**
- typedef [std::bidirectional_iterator_tag](#) **iterator_category**
- typedef Pointer **pointer**
- typedef Reference **reference**
- typedef Value_Type **value_type**

Public Member Functions

- **bin_search_tree_const_it_** (const Node_Pointer p_nd=0)
- **bin_search_tree_const_it_** (const [bin_search_tree_const_it_](#) < Node_Pointer, Value_Type, Pointer, Const_↔ Pointer, Reference, Const_Reference, !Is_Forward_Iterator, _Alloc > &other)
- bool **operator!=** (const [bin_search_tree_const_it_](#) < Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > &other) const
- bool **operator!=** (const [bin_search_tree_const_it_](#) < Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, !Is_Forward_Iterator, _Alloc > &other) const
- const_reference **operator*** () const
- [bin_search_tree_const_it_](#) < Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > & **operator++** ()
- [bin_search_tree_const_it_](#) < Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > **operator++** (int)
- [bin_search_tree_const_it_](#) < Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > & **operator--** ()
- [bin_search_tree_const_it_](#) < Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > **operator--** (int)
- const_pointer **operator->** () const
- [bin_search_tree_const_it_](#) < Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > & **operator=** (const [bin_search_tree_const_it_](#) < Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > &other)
- [bin_search_tree_const_it_](#) < Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > & **operator=** (const [bin_search_tree_const_it_](#) < Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, !Is_Forward_Iterator, _Alloc > &other)
- bool **operator==** (const [bin_search_tree_const_it_](#) < Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > &other) const
- bool **operator==** (const [bin_search_tree_const_it_](#) < Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, !Is_Forward_Iterator, _Alloc > &other) const

Public Attributes

- Node_Pointer **m_p_nd**

Protected Member Functions

- void **dec** (false_type)
- void **dec** (true_type)
- void **inc** (false_type)
- void **inc** (true_type)

4.191.1 Detailed Description

```
template<typename Node_Pointer, typename Value_Type, typename Pointer, typename Const_Pointer, typename Reference, typename Const_Reference, bool Is_Forward_Iterator, typename _Alloc>
class __gnu_pbds::detail::bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_
Reference, Is_Forward_Iterator, _Alloc >
```

Const iterator.

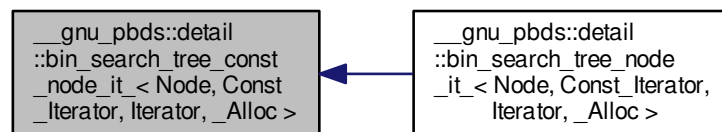
Definition at line 105 of file `point_iterators.hpp`.

The documentation for this class was generated from the following file:

- [point_iterators.hpp](#)

4.192 `__gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >`:



Public Types

- typedef Const_Iterator [const_reference](#)
- typedef [trivial_iterator_difference_type](#) `difference_type`
- typedef [trivial_iterator_tag](#) `iterator_category`
- typedef `_Alloc::template rebind< metadata_type >::other::const_reference` [metadata_const_reference](#)
- typedef `Node::metadata_type` [metadata_type](#)
- typedef Const_Iterator [reference](#)
- typedef Const_Iterator [value_type](#)

Public Member Functions

- **bin_search_tree_const_node_it_** (const node_pointer p_nd=0)
- **bin_search_tree_const_node_it_** < Node, Const_Iterator, Iterator, _Alloc > **get_l_child** () const
- **metadata_const_reference** **get_metadata** () const
- **bin_search_tree_const_node_it_** < Node, Const_Iterator, Iterator, _Alloc > **get_r_child** () const
- bool **operator!=** (const **bin_search_tree_const_node_it_** < Node, Const_Iterator, Iterator, _Alloc > &other) const
- **const_reference** **operator*** () const
- bool **operator==** (const **bin_search_tree_const_node_it_** < Node, Const_Iterator, Iterator, _Alloc > &other) const

Public Attributes

- node_pointer **m_p_nd**

4.192.1 Detailed Description

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
class __gnu_pbds::detail::bin_search_tree_const_node_it_ < Node, Const_Iterator, Iterator, _Alloc >
```

Const node iterator.

Definition at line 58 of file bin_search_tree_/node_iterators.hpp.

4.192.2 Member Typedef Documentation

4.192.2.1 `template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > typedef Const_Iterator
__gnu_pbds::detail::bin_search_tree_const_node_it_ < Node, Const_Iterator, Iterator, _Alloc
>::const_reference`

Iterator's __const reference type.

Definition at line 80 of file bin_search_tree_/node_iterators.hpp.

4.192.2.2 `template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > typedef
trivial_iterator_difference_type __gnu_pbds::detail::bin_search_tree_const_node_it_ < Node,
Const_Iterator, Iterator, _Alloc >::difference_type`

Difference type.

Definition at line 71 of file bin_search_tree_/node_iterators.hpp.

4.192.2.3 `template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > typedef trivial_iterator_tag
__gnu_pbds::detail::bin_search_tree_const_node_it_ < Node, Const_Iterator, Iterator, _Alloc
>::iterator_category`

Category.

Definition at line 68 of file bin_search_tree_/node_iterators.hpp.

4.192.2.4 `template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > typedef _Alloc::template
rebind<metadata_type>::other::const_reference __gnu_pbds::detail::bin_search_tree_const_node_it_<
Node, Const_Iterator, Iterator, _Alloc >::metadata_const_reference`

Const metadata reference type.

Definition at line 88 of file `bin_search_tree_/node_iterators.hpp`.

4.192.2.5 `template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > typedef Node::metadata_type
__gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc
>::metadata_type`

Metadata type.

Definition at line 83 of file `bin_search_tree_/node_iterators.hpp`.

4.192.2.6 `template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > typedef Const_Iterator
__gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >::reference`

Iterator's reference type.

Definition at line 77 of file `bin_search_tree_/node_iterators.hpp`.

4.192.2.7 `template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > typedef Const_Iterator
__gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >::value_type`

Iterator's value type.

Definition at line 74 of file `bin_search_tree_/node_iterators.hpp`.

4.192.3 Member Function Documentation

4.192.3.1 `template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > bin_search_tree_const_node_
it<Node, Const_Iterator, Iterator, _Alloc> __gnu_pbds::detail::bin_search_tree_const_node_it_< Node,
Const_Iterator, Iterator, _Alloc >::get_l_child () const [inline]`

Returns the `__const` node iterator associated with the left node.

Definition at line 107 of file `bin_search_tree_/node_iterators.hpp`.

4.192.3.2 `template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > metadata_const_reference
__gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >::get_metadata
() const [inline]`

Metadata access.

Definition at line 102 of file `bin_search_tree_/node_iterators.hpp`.

4.192.3.3 `template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > bin_search_tree_const_node_it_<Node, Const_Iterator, Iterator, _Alloc> __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >::get_r_child () const [inline]`

Returns the __const node iterator associated with the right node.

Definition at line 112 of file `bin_search_tree_/node_iterators.hpp`.

4.192.3.4 `template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > bool __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >::operator!= (const bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc > & other) const [inline]`

Compares (negatively) to a different iterator object.

Definition at line 122 of file `bin_search_tree_/node_iterators.hpp`.

4.192.3.5 `template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > const_reference __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >::operator* () const [inline]`

Access.

Definition at line 97 of file `bin_search_tree_/node_iterators.hpp`.

4.192.3.6 `template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > bool __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >::operator== (const bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc > & other) const [inline]`

Compares to a different iterator object.

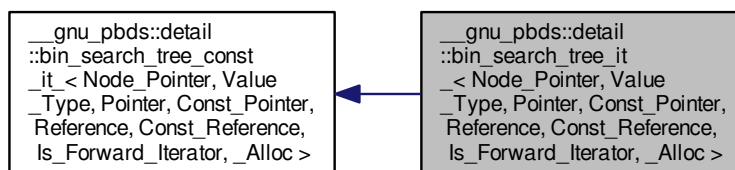
Definition at line 117 of file `bin_search_tree_/node_iterators.hpp`.

The documentation for this class was generated from the following file:

- [bin_search_tree_/node_iterators.hpp](#)

4.193 `__gnu_pbds::detail::bin_search_tree_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::bin_search_tree_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc >`:



Public Types

- typedef Const_Pointer **const_pointer**
- typedef Const_Reference **const_reference**
- typedef _Alloc::difference_type **difference_type**
- typedef [std::bidirectional_iterator_tag](#) **iterator_category**
- typedef Pointer **pointer**
- typedef Reference **reference**
- typedef Value_Type **value_type**

Public Member Functions

- **bin_search_tree_it** (const Node_Pointer p_nd=0)
- **bin_search_tree_it** (const [bin_search_tree_it](#) < Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > &other)
- bool **operator!=** (const [bin_search_tree_const_it](#) < Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > &other) const
- bool **operator!=** (const [bin_search_tree_const_it](#) < Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > &other) const
- [bin_search_tree_const_it](#) < Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc >::reference **operator*** () const
- [bin_search_tree_it](#) < Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > & **operator++** ()
- [bin_search_tree_it](#) < Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > **operator++** (int)
- [bin_search_tree_it](#) < Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > & **operator--** ()
- [bin_search_tree_it](#) < Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > **operator--** (int)
- [bin_search_tree_const_it](#) < Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc >::pointer **operator->** () const
- [bin_search_tree_it](#) < Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > & **operator=** (const [bin_search_tree_it](#) < Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > &other)
- [bin_search_tree_it](#) < Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > & **operator=** (const [bin_search_tree_it](#) < Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > &other)
- bool **operator==** (const [bin_search_tree_const_it](#) < Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > &other) const
- bool **operator==** (const [bin_search_tree_const_it](#) < Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > &other) const

Public Attributes

- Node_Pointer **m_p_nd**

Protected Types

- typedef [bin_search_tree_const_it](#) < Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > **base_it_type**

Protected Member Functions

- void **dec** (false_type)
- void **dec** (true_type)
- void **inc** (false_type)
- void **inc** (true_type)

4.193.1 Detailed Description

```
template<typename Node_Pointer, typename Value_Type, typename Pointer, typename Const_Pointer, typename Reference, typename
Const_Reference, bool Is_Forward_Iterator, typename _Alloc>
class __gnu_pbds::detail::bin_search_tree_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_
_Forward_Iterator, _Alloc >
```

Iterator.

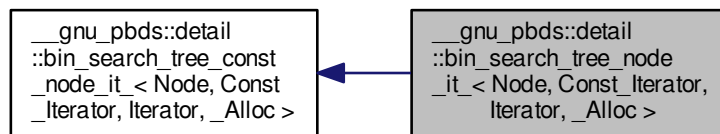
Definition at line 282 of file point_iterators.hpp.

The documentation for this class was generated from the following file:

- [point_iterators.hpp](#)

4.194 __gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc > Class Template Reference

Inheritance diagram for __gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc >:



Public Types

- typedef Iterator [const_reference](#)
- typedef [trivial_iterator_difference_type](#) difference_type
- typedef [trivial_iterator_tag](#) iterator_category
- typedef _Alloc::template rebind< [metadata_type](#) >::other::const_reference [metadata_const_reference](#)
- typedef Node::metadata_type [metadata_type](#)
- typedef Iterator [reference](#)
- typedef Iterator [value_type](#)

Public Member Functions

- `bin_search_tree_node_it_` (const node_pointer p_nd=0)
- `bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc >` `get_l_child` () const
- `metadata_const_reference` `get_metadata` () const
- `bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc >` `get_r_child` () const
- bool `operator!=` (const `bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >` &other) const
- Iterator `operator*` () const
- bool `operator==` (const `bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >` &other) const

Public Attributes

- node_pointer `m_p_nd`

4.194.1 Detailed Description

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
class __gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc >
```

Node iterator.

Definition at line 136 of file `bin_search_tree_/node_iterators.hpp`.

4.194.2 Member Typedef Documentation

4.194.2.1 `template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > typedef Iterator`
`__gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc >::const_reference`

Iterator's `__const` reference type.

Definition at line 153 of file `bin_search_tree_/node_iterators.hpp`.

4.194.2.2 `template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > typedef`
`trivial_iterator_difference_type __gnu_pbds::detail::bin_search_tree_const_node_it_< Node,`
`Const_Iterator, Iterator, _Alloc >::difference_type` `[inherited]`

Difference type.

Definition at line 71 of file `bin_search_tree_/node_iterators.hpp`.

4.194.2.3 `template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > typedef trivial_iterator_tag`
`__gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc`
`>::iterator_category` `[inherited]`

Category.

Definition at line 68 of file `bin_search_tree_/node_iterators.hpp`.

```
4.194.2.4  template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > typedef _Alloc::template
rebind<metadata_type>::other::const_reference __gnu_pbds::detail::bin_search_tree_const_node_it_<
Node, Const_Iterator, Iterator, _Alloc >::metadata_const_reference  [inherited]
```

Const metadata reference type.

Definition at line 88 of file bin_search_tree_/node_iterators.hpp.

```
4.194.2.5  template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > typedef Node::metadata_type
__gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc
>::metadata_type  [inherited]
```

Metadata type.

Definition at line 83 of file bin_search_tree_/node_iterators.hpp.

```
4.194.2.6  template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > typedef Iterator
__gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc >::reference
```

Iterator's reference type.

Definition at line 150 of file bin_search_tree_/node_iterators.hpp.

```
4.194.2.7  template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > typedef Iterator
__gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc >::value_type
```

Iterator's value type.

Definition at line 147 of file bin_search_tree_/node_iterators.hpp.

4.194.3 Member Function Documentation

```
4.194.3.1  template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > bin_search_tree_node_it_↵
<Node, Const_Iterator, Iterator, _Alloc> __gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator,
Iterator, _Alloc >::get_l_child( ) const  [inline]
```

Returns the node iterator associated with the left node.

Definition at line 167 of file bin_search_tree_/node_iterators.hpp.

```
4.194.3.2  template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > metadata_const_reference
__gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >::get_metadata
( ) const  [inline], [inherited]
```

Metadata access.

Definition at line 102 of file bin_search_tree_/node_iterators.hpp.

4.194.3.3 `template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > bin_search_tree_node_it_↵
 <Node, Const_Iterator, Iterator, _Alloc> __gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator,
 Iterator, _Alloc >::get_r_child () const [inline]`

Returns the node iterator associated with the right node.

Definition at line 175 of file `bin_search_tree_/node_iterators.hpp`.

4.194.3.4 `template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > bool
 __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >::operator!= (
 const bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc > & other) const [inline],
 [inherited]`

Compares (negatively) to a different iterator object.

Definition at line 122 of file `bin_search_tree_/node_iterators.hpp`.

4.194.3.5 `template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > Iterator
 __gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc >::operator* () const
 [inline]`

Access.

Definition at line 162 of file `bin_search_tree_/node_iterators.hpp`.

4.194.3.6 `template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > bool
 __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >::operator== (
 const bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc > & other) const [inline],
 [inherited]`

Compares to a different iterator object.

Definition at line 117 of file `bin_search_tree_/node_iterators.hpp`.

The documentation for this class was generated from the following file:

- [bin_search_tree_/node_iterators.hpp](#)

4.195 `__gnu_pbds::detail::bin_search_tree_traits< Key, Mapped, Cmp_Fn, Node_Update, Node, _Alloc >` Struct Template Reference

Public Types

- typedef `bin_search_tree_const_it_< typename _Alloc::template rebind< node >::other::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_↵
 traits::reference, typename type_traits::const_reference, false, _Alloc > const_reverse_iterator`
- typedef `Node node`
- typedef `bin_search_tree_const_node_it_< Node, point_const_iterator, point_iterator, _Alloc > node_const_↵
 iterator`

- typedef [bin_search_tree_node_it](#) < Node, [point_const_iterator](#), [point_iterator](#), _Alloc > **node_iterator**
- typedef Node_Update< [node_const_iterator](#), [node_iterator](#), Cmp_Fn, _Alloc > **node_update**
- typedef [__gnu_pbds::null_node_update](#)< [node_const_iterator](#), [node_iterator](#), Cmp_Fn, _Alloc > * **null_node_update_pointer**
- typedef [bin_search_tree_const_it](#) < typename _Alloc::template rebind< node >::other::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, true, _Alloc > **point_const_iterator**
- typedef [bin_search_tree_it](#) < typename _Alloc::template rebind< node >::other::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, true, _Alloc > **point_iterator**
- typedef [bin_search_tree_it](#) < typename _Alloc::template rebind< node >::other::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, false, _Alloc > **reverse_iterator**

4.195.1 Detailed Description

```
template<typename Key, typename Mapped, class Cmp_Fn, template< typename Node_Cltr, class Node_Itr, class _Cmp_Fn, typename
_Alloc > class Node_Update, class Node, typename _Alloc>
struct __gnu_pbds::detail::bin_search_tree_traits< Key, Mapped, Cmp_Fn, Node_Update, Node, _Alloc >
```

Binary search tree traits, primary template.

Definition at line 63 of file [bin_search_tree_/traits.hpp](#).

4.195.2 Member Typedef Documentation

4.195.2.1 `template<typename Key, typename Mapped, class Cmp_Fn, template< typename Node_Cltr, class Node_Itr, class _Cmp_Fn, typename _Alloc > class Node_Update, class Node, typename _Alloc> typedef bin_search_tree_const_node_it < Node, point_const_iterator, point_iterator, _Alloc> __gnu_pbds::detail::bin_search_tree_traits< Key, Mapped, Cmp_Fn, Node_Update, Node, _Alloc >::node_const_iterator`

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

Definition at line 131 of file [bin_search_tree_/traits.hpp](#).

The documentation for this struct was generated from the following file:

- [bin_search_tree_/traits.hpp](#)

4.196 `__gnu_pbds::detail::bin_search_tree_traits< Key, null_type, Cmp_Fn, Node_Update, Node, _Alloc >` Struct Template Reference

Public Types

- typedef `bin_search_tree_const_it` `< typename _Alloc::template rebind< node >::other::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, false, _Alloc >` **const_reverse_iterator**
- typedef `Node` **node**
- typedef `bin_search_tree_const_node_it` `< Node, point_const_iterator, point_iterator, _Alloc >` **node_const_iterator**
- typedef `node_const_iterator` **node_iterator**
- typedef `Node_Update` `< node_const_iterator, node_iterator, Cmp_Fn, _Alloc >` **node_update**
- typedef `__gnu_pbds::null_node_update` `< node_const_iterator, node_iterator, Cmp_Fn, _Alloc >` *** null_node_update_pointer**
- typedef `bin_search_tree_const_it` `< typename _Alloc::template rebind< node >::other::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, true, _Alloc >` **point_const_iterator**
- typedef `point_const_iterator` **point_iterator**
- typedef `const_reverse_iterator` **reverse_iterator**

4.196.1 Detailed Description

```
template<typename Key, class Cmp_Fn, template< typename Node_Cltr, class Node_Itr, class _Cmp_Fn, typename _Alloc > class
Node_Update, class Node, typename _Alloc>
```

```
struct __gnu_pbds::detail::bin_search_tree_traits< Key, null_type, Cmp_Fn, Node_Update, Node, _Alloc >
```

Specialization.

Definition at line 169 of file `bin_search_tree_/traits.hpp`.

4.196.2 Member Typedef Documentation

4.196.2.1 `template<typename Key , class Cmp_Fn , template< typename Node_Cltr, class Node_Itr, class _Cmp_Fn, typename _Alloc > class Node_Update, class Node , typename _Alloc >` typedef `bin_search_tree_const_node_it` `< Node, point_const_iterator, point_iterator, _Alloc>` `__gnu_pbds::detail::bin_search_tree_traits< Key, null_type, Cmp_Fn, Node_Update, Node, _Alloc >::node_const_iterator`

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

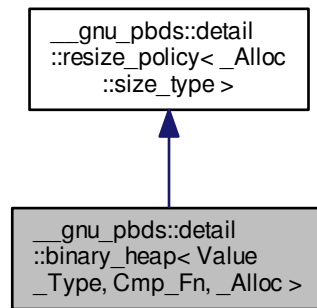
Definition at line 221 of file `bin_search_tree_/traits.hpp`.

The documentation for this struct was generated from the following file:

- [bin_search_tree_/traits.hpp](#)

4.197 `__gnu_pbds::detail::binary_heap< Value_Type, Cmp_Fn, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::binary_heap< Value_Type, Cmp_Fn, _Alloc >`:



Public Types

- typedef `_Alloc` **allocator_type**
- typedef `Cmp_Fn` **cmp_fn**
- typedef `cond_dealtor< value_type, _Alloc >` **cond_dealtor_t**
- typedef `binary_heap_const_iterator< value_type, entry, simple_value, _Alloc >` **const_iterator**
- typedef `value_allocator::const_pointer` **const_pointer**
- typedef `value_allocator::const_reference` **const_reference**
- typedef `_Alloc::difference_type` **difference_type**
- typedef `__conditional_type< simple_value, value_type, pointer >::__type` **entry**
- typedef `_Alloc::template rebind< entry >::other` **entry_allocator**
- typedef `entry_cmp< Value_Type, Cmp_Fn, _Alloc, is_simple< Value_Type >::value >::type` **entry_cmp**
- typedef `entry_allocator::pointer` **entry_pointer**
- typedef `const_iterator` **iterator**
- typedef `binary_heap_point_const_iterator< value_type, entry, simple_value, _Alloc >` **point_const_iterator**
- typedef `point_const_iterator` **point_iterator**
- typedef `value_allocator::pointer` **pointer**
- typedef `value_allocator::reference` **reference**
- typedef `__gnu_pbds::detail::resize_policy< typename _Alloc::size_type >` **resize_policy**
- typedef `_Alloc::size_type` **size_type**
- typedef `Value_Type` **value_type**

Public Member Functions

- **binary_heap** (const cmp_fn &)
- **binary_heap** (const [binary_heap](#) &)
- [iterator](#) **begin** ()
- [const_iterator](#) **begin** () const
- void **clear** ()
- bool **empty** () const
- [iterator](#) **end** ()
- [const_iterator](#) **end** () const
- void **erase** ([point_iterator](#))
- void **erase_at** (entry_pointer, size_type, false_type)
- void **erase_at** (entry_pointer, size_type, true_type)
- template<typename Pred >
size_type **erase_if** (Pred)
- Cmp_Fn & **get_cmp_fn** ()
- const Cmp_Fn & **get_cmp_fn** () const
- size_type **get_new_size_for_arbitrary** (size_type) const
- size_type **get_new_size_for_grow** () const
- size_type **get_new_size_for_shrink** () const
- bool **grow_needed** (size_type) const
- void **join** ([binary_heap](#) &)
- size_type **max_size** () const
- void **modify** ([point_iterator](#), const_reference)
- void **notify_arbitrary** (size_type)
- void **notify_grow_resize** ()
- void **notify_shrink_resize** ()
- void **pop** ()
- [point_iterator](#) **push** (const_reference)
- bool **resize_needed_for_grow** (size_type) const
- bool **resize_needed_for_shrink** (size_type) const
- bool **shrink_needed** (size_type) const
- size_type **size** () const
- template<typename Pred >
void **split** (Pred, [binary_heap](#) &)
- void **swap** ([resize_policy](#)< _Alloc::size_type > &)
- void **swap** ([binary_heap](#) &)
- const_reference **top** () const

Static Public Attributes

- static const _Alloc::size_type **min_size**

Protected Member Functions

- template<typename It >
void **copy_from_range** (It, It)

4.197.1 Detailed Description

```
template<typename Value_Type, typename Cmp_Fn, typename _Alloc>
class __gnu_pbds::detail::binary_heap< Value_Type, Cmp_Fn, _Alloc >
```

Binary heaps composed of resize and compare policies.

Based on CLRS.

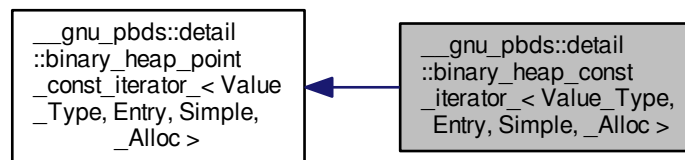
Definition at line 84 of file `binary_heap.hpp`.

The documentation for this class was generated from the following file:

- [binary_heap.hpp](#)

4.198 `__gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >`:



Public Types

- typedef `base_type::const_pointer` `const_pointer`
- typedef `base_type::const_reference` `const_reference`
- typedef `_Alloc::difference_type` `difference_type`
- typedef `std::forward_iterator_tag` `iterator_category`
- typedef `base_type::pointer` `pointer`
- typedef `base_type::reference` `reference`
- typedef `base_type::value_type` `value_type`

Public Member Functions

- `binary_heap_const_iterator_` (entry_pointer p_e)
- `binary_heap_const_iterator_` ()
- `binary_heap_const_iterator_` (const `binary_heap_const_iterator_` &other)
- `bool operator!=` (const `binary_heap_const_iterator_` &other) const
- `bool operator!=` (const `binary_heap_point_const_iterator_` &other) const
- `const_reference operator*` () const
- `binary_heap_const_iterator_ & operator++` ()
- `binary_heap_const_iterator_ operator++` (int)
- `const_pointer operator->` () const
- `bool operator==` (const `binary_heap_const_iterator_` &other) const
- `bool operator==` (const `binary_heap_point_const_iterator_` &other) const

Public Attributes

- entry_pointer `m_p_e`

4.198.1 Detailed Description

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
class __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >
```

Const point-type iterator.

Definition at line 60 of file `binary_heap_/const_iterator.hpp`.

4.198.2 Member Typedef Documentation

4.198.2.1 `template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > typedef base_type::const_pointer __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >::const_pointer`

Iterator's const pointer type.

Definition at line 80 of file `binary_heap_/const_iterator.hpp`.

4.198.2.2 `template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > typedef base_type::const_reference __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >::const_reference`

Iterator's const reference type.

Definition at line 86 of file `binary_heap_/const_iterator.hpp`.

4.198.2.3 `template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > typedef _Alloc::difference_type
__gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >::difference_type`

Difference type.

Definition at line 71 of file `binary_heap_/const_iterator.hpp`.

4.198.2.4 `template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > typedef std::forward_iterator_tag
__gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >::iterator_category`

Category.

Definition at line 68 of file `binary_heap_/const_iterator.hpp`.

4.198.2.5 `template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > typedef base_type::pointer
__gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >::pointer`

Iterator's pointer type.

Definition at line 77 of file `binary_heap_/const_iterator.hpp`.

4.198.2.6 `template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > typedef base_type::reference
__gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >::reference`

Iterator's reference type.

Definition at line 83 of file `binary_heap_/const_iterator.hpp`.

4.198.2.7 `template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > typedef base_type::value_type
__gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >::value_type`

Iterator's value type.

Definition at line 74 of file `binary_heap_/const_iterator.hpp`.

4.198.3 Constructor & Destructor Documentation

4.198.3.1 `template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > __gnu_pbds::detail::
::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >::binary_heap_const_iterator_ ()
[inline]`

Default constructor.

Definition at line 94 of file `binary_heap_/const_iterator.hpp`.

4.198.3.2 `template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >::binary_heap_const_iterator_ (const binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc > & other) [inline]`

Copy constructor.

Definition at line 99 of file `binary_heap_/const_iterator.hpp`.

4.198.4 Member Function Documentation

4.198.4.1 `template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > bool __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >::operator!= (const binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc > & other) const [inline]`

Compares content (negatively) to a different iterator object.

Definition at line 110 of file `binary_heap_/const_iterator.hpp`.

4.198.4.2 `template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > bool __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >::operator!= (const binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc > & other) const [inline], [inherited]`

Compares content (negatively) to a different iterator object.

Definition at line 126 of file `binary_heap_/point_const_iterator.hpp`.

4.198.4.3 `template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > const_reference __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >::operator* () const [inline], [inherited]`

Access.

Definition at line 113 of file `binary_heap_/point_const_iterator.hpp`.

4.198.4.4 `template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > const_pointer __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >::operator-> () const [inline], [inherited]`

Access.

Definition at line 105 of file `binary_heap_/point_const_iterator.hpp`.

4.198.4.5 `template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > bool __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >::operator== (const binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc > & other) const [inline]`

Compares content to a different iterator object.

Definition at line 105 of file `binary_heap_/const_iterator.hpp`.

```
4.198.4.6 template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > bool
    __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >::operator== (
        const binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc > & other ) const    [inline],
        [inherited]
```

Compares content to a different iterator object.

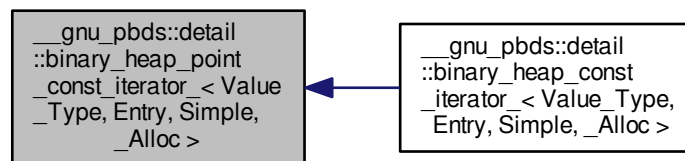
Definition at line 121 of file `binary_heap_/point_const_iterator.hpp`.

The documentation for this class was generated from the following file:

- [binary_heap_/const_iterator.hpp](#)

4.199 __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc > Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >`:



Public Types

- `typedef _Alloc::template rebind< value_type >::other::const_pointer const_pointer`
- `typedef _Alloc::template rebind< value_type >::other::const_reference const_reference`
- `typedef trivial_iterator_difference_type difference_type`
- `typedef trivial_iterator_tag iterator_category`
- `typedef _Alloc::template rebind< value_type >::other::pointer pointer`
- `typedef _Alloc::template rebind< value_type >::other::reference reference`
- `typedef Value_Type value_type`

Public Member Functions

- `binary_heap_point_const_iterator_ (entry_pointer p_e)`
- `binary_heap_point_const_iterator_ ()`
- `binary_heap_point_const_iterator_ (const binary_heap_point_const_iterator_ &other)`
- `bool operator!= (const binary_heap_point_const_iterator_ &other) const`
- `const_reference operator* () const`
- `const_pointer operator-> () const`
- `bool operator== (const binary_heap_point_const_iterator_ &other) const`

Public Attributes

- entry_pointer `m_p_e`

Protected Types

- `typedef _Alloc::template rebind< Entry >::other::pointer` `entry_pointer`

4.199.1 Detailed Description

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
class __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >
```

Const point-type iterator.

Definition at line 55 of file `binary_heap_/point_const_iterator.hpp`.

4.199.2 Member Typedef Documentation

4.199.2.1 `template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > typedef _Alloc::template rebind<value_type>::other::const_pointer __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >::const_pointer`

Iterator's const pointer type.

Definition at line 77 of file `binary_heap_/point_const_iterator.hpp`.

4.199.2.2 `template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > typedef _Alloc::template rebind<value_type>::other::const_reference __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >::const_reference`

Iterator's const reference type.

Definition at line 87 of file `binary_heap_/point_const_iterator.hpp`.

4.199.2.3 `template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > typedef trivial_iterator_difference_type __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >::difference_type`

Difference type.

Definition at line 65 of file `binary_heap_/point_const_iterator.hpp`.

```
4.199.2.4  template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > typedef trivial_iterator_tag
__gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc
>::iterator_category
```

Category.

Definition at line 62 of file binary_heap_/point_const_iterator.hpp.

```
4.199.2.5  template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > typedef _Alloc::template
rebind<value_type>::other::pointer __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type,
Entry, Simple, _Alloc >::pointer
```

Iterator's pointer type.

Definition at line 72 of file binary_heap_/point_const_iterator.hpp.

```
4.199.2.6  template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > typedef _Alloc::template
rebind<value_type>::other::reference __gnu_pbds::detail::binary_heap_point_const_iterator_<
Value_Type, Entry, Simple, _Alloc >::reference
```

Iterator's reference type.

Definition at line 82 of file binary_heap_/point_const_iterator.hpp.

```
4.199.2.7  template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > typedef Value_Type
__gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >::value_type
```

Iterator's value type.

Definition at line 68 of file binary_heap_/point_const_iterator.hpp.

4.199.3 Constructor & Destructor Documentation

```
4.199.3.1  template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > __gnu_pbds::detail::binary_↵
heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >::binary_heap_point_const_iterator_( )
[inline]
```

Default constructor.

Definition at line 95 of file binary_heap_/point_const_iterator.hpp.

```
4.199.3.2  template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > __gnu_pbds::detail::binary_↵
heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >::binary_heap_point_const_iterator_(
const binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc > & other ) [inline]
```

Copy constructor.

Definition at line 99 of file binary_heap_/point_const_iterator.hpp.

4.199.4 Member Function Documentation

4.199.4.1 `template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > bool
__gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >::operator!=(
const binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc > & other) const [inline]`

Compares content (negatively) to a different iterator object.

Definition at line 126 of file `binary_heap_/point_const_iterator.hpp`.

4.199.4.2 `template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > const_reference
__gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >::operator*()
const [inline]`

Access.

Definition at line 113 of file `binary_heap_/point_const_iterator.hpp`.

4.199.4.3 `template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > const_pointer
__gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >::operator->() const [inline]`

Access.

Definition at line 105 of file `binary_heap_/point_const_iterator.hpp`.

4.199.4.4 `template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > bool
__gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >::operator==(
const binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc > & other) const [inline]`

Compares content to a different iterator object.

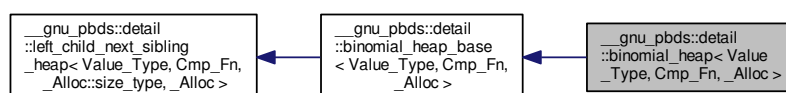
Definition at line 121 of file `binary_heap_/point_const_iterator.hpp`.

The documentation for this class was generated from the following file:

- [binary_heap_/point_const_iterator.hpp](#)

4.200 `__gnu_pbds::detail::binomial_heap< Value_Type, Cmp_Fn, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::binomial_heap< Value_Type, Cmp_Fn, _Alloc >`:



Public Types

- typedef base_type::allocator_type **allocator_type**
- typedef base_type::cmp_fn **cmp_fn**
- typedef base_type::const_iterator **const_iterator**
- typedef base_type::const_pointer **const_pointer**
- typedef base_type::const_reference **const_reference**
- typedef _Alloc::difference_type **difference_type**
- typedef base_type::iterator **iterator**
- typedef base_type::point_const_iterator **point_const_iterator**
- typedef base_type::point_iterator **point_iterator**
- typedef base_type::pointer **pointer**
- typedef base_type::reference **reference**
- typedef _Alloc::size_type **size_type**
- typedef Value_Type **value_type**

Public Member Functions

- **binomial_heap** (const Cmp_Fn &)
- **binomial_heap** (const binomial_heap &)
- **iterator begin** ()
- **const_iterator begin** () const
- void **clear** ()
- bool **empty** () const
- **iterator end** ()
- **const_iterator end** () const
- void **erase** (point_iterator)
- template<typename Pred >
size_type **erase_if** (Pred)
- Cmp_Fn & **get_cmp_fn** ()
- const Cmp_Fn & **get_cmp_fn** () const
- void **join** (binomial_heap_base< Value_Type, Cmp_Fn, _Alloc > &)
- size_type **max_size** () const
- void **modify** (point_iterator, const_reference)
- void **pop** ()
- **point_iterator push** (const_reference)
- size_type **size** () const
- template<typename Pred >
void **split** (Pred, binomial_heap_base< Value_Type, Cmp_Fn, _Alloc > &)
- void **swap** (left_child_next_sibling_heap< Value_Type, Cmp_Fn, _Alloc::size_type, _Alloc > &)
- const_reference **top** () const

Protected Types

- typedef base_type::node **node**
- typedef _Alloc::template rebind< left_child_next_sibling_heap_node_ < Value_Type, _Alloc::size_type, _Alloc > ::other **node_allocator**
- typedef _Alloc::size_type **node_metadata**
- typedef std::pair< node_pointer, node_pointer > **node_pointer_pair**

Protected Member Functions

- void **actual_erase_node** (node_pointer)
- void **bubble_to_top** (node_pointer)
- void **clear_imp** (node_pointer)
- template<typename It >
void **copy_from_range** (It, It)
- void **find_max** ()
- node_pointer **get_new_node_for_insert** (const_reference)
- node_pointer **prune** (Pred)
- void **swap** ([binomial_heap_base](#)< Value_Type, Cmp_Fn, _Alloc > &)
- void **swap_with_parent** (node_pointer, node_pointer)
- void **to_linked_list** ()
- void **value_swap** ([left_child_next_sibling_heap](#) &)

Static Protected Member Functions

- static void **make_child_of** (node_pointer, node_pointer)
- static node_pointer **parent** (node_pointer)

Protected Attributes

- node_pointer **m_p_max**
- node_pointer **m_p_root**
- size_type **m_size**

4.200.1 Detailed Description

```
template<typename Value_Type, typename Cmp_Fn, typename _Alloc>
class __gnu_pbds::detail::binomial_heap< Value_Type, Cmp_Fn, _Alloc >
```

Binomial heap.

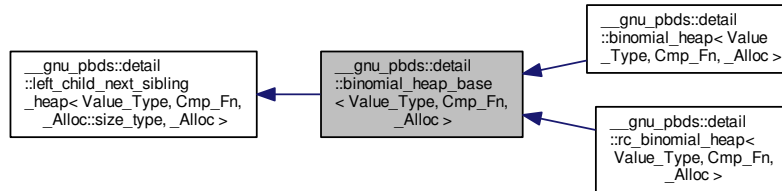
Definition at line 68 of file `binomial_heap_.hpp`.

The documentation for this class was generated from the following file:

- [binomial_heap_.hpp](#)

4.201 `__gnu_pbds::detail::binomial_heap_base< Value_Type, Cmp_Fn, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::binomial_heap_base< Value_Type, Cmp_Fn, _Alloc >`:



Public Types

- typedef `_Alloc` **allocator_type**
- typedef `Cmp_Fn` **cmp_fn**
- typedef `base_type::const_iterator` **const_iterator**
- typedef `__rebind_v::const_pointer` **const_pointer**
- typedef `__rebind_v::const_reference` **const_reference**
- typedef `_Alloc::difference_type` **difference_type**
- typedef `base_type::iterator` **iterator**
- typedef `base_type::point_const_iterator` **point_const_iterator**
- typedef `base_type::point_iterator` **point_iterator**
- typedef `__rebind_v::pointer` **pointer**
- typedef `__rebind_v::reference` **reference**
- typedef `_Alloc::size_type` **size_type**
- typedef `Value_Type` **value_type**

Public Member Functions

- `iterator` **begin** ()
- `const_iterator` **begin** () const
- void **clear** ()
- bool **empty** () const
- `iterator` **end** ()
- `const_iterator` **end** () const
- void **erase** (`point_iterator`)
- template<typename Pred >
size_type **erase_if** (Pred)
- `Cmp_Fn` & **get_cmp_fn** ()
- const `Cmp_Fn` & **get_cmp_fn** () const
- void **join** (`binomial_heap_base< Value_Type, Cmp_Fn, _Alloc >` &)
- size_type **max_size** () const
- void **modify** (`point_iterator`, const_reference)
- void **pop** ()

- [point_iterator](#) **push** (const_reference)
- size_type **size** () const
- template<typename Pred >
void **split** (Pred, [binomial_heap_base](#)< Value_Type, Cmp_Fn, _Alloc > &)
- void **swap** ([left_child_next_sibling_heap](#)< Value_Type, Cmp_Fn, _Alloc::size_type, _Alloc > &)
- const_reference **top** () const

Protected Types

- typedef base_type::node **node**
- typedef _Alloc::template rebind< [left_child_next_sibling_heap_node](#)< Value_Type, _Alloc::size_type, _Alloc >>::other **node_allocator**
- typedef base_type::node_const_pointer **node_const_pointer**
- typedef _Alloc::size_type **node_metadata**
- typedef base_type::node_pointer **node_pointer**
- typedef [std::pair](#)< node_pointer, node_pointer > **node_pointer_pair**

Protected Member Functions

- **binomial_heap_base** (const Cmp_Fn &)
- **binomial_heap_base** (const [binomial_heap_base](#)< Value_Type, Cmp_Fn, _Alloc > &)
- void **actual_erase_node** (node_pointer)
- void **bubble_to_top** (node_pointer)
- void **clear_imp** (node_pointer)
- template<typename It >
void **copy_from_range** (It, It)
- void **find_max** ()
- node_pointer **get_new_node_for_insert** (const_reference)
- node_pointer **prune** (Pred)
- void **swap** ([binomial_heap_base](#)< Value_Type, Cmp_Fn, _Alloc > &)
- void **swap_with_parent** (node_pointer, node_pointer)
- void **to_linked_list** ()
- void **value_swap** ([left_child_next_sibling_heap](#) &)

Static Protected Member Functions

- static void **make_child_of** (node_pointer, node_pointer)
- static node_pointer **parent** (node_pointer)

Protected Attributes

- node_pointer **m_p_max**
- node_pointer **m_p_root**
- size_type **m_size**

4.201.1 Detailed Description

```
template<typename Value_Type, typename Cmp_Fn, typename _Alloc>
class __gnu_pbds::detail::binomial_heap_base< Value_Type, Cmp_Fn, _Alloc >
```

Base class for binomial heap.

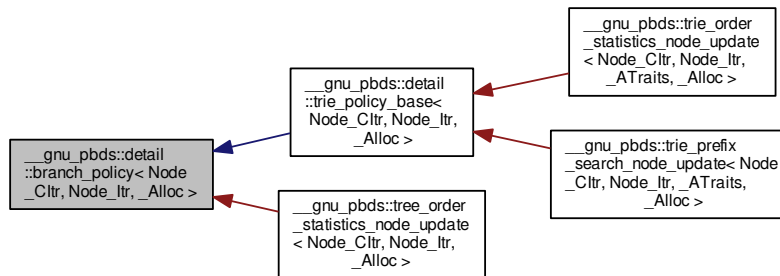
Definition at line 77 of file binomial_heap_base_.hpp.

The documentation for this class was generated from the following file:

- [binomial_heap_base_.hpp](#)

4.202 __gnu_pbds::detail::branch_policy< Node_Cltr, Node_Itr, _Alloc > Struct Template Reference

Inheritance diagram for __gnu_pbds::detail::branch_policy< Node_Cltr, Node_Itr, _Alloc >:



Protected Types

- typedef `rebind_v::const_pointer` **const_pointer**
- typedef `rebind_v::const_reference` **const_reference**
- typedef `Node_Itr::value_type` **it_type**
- typedef `rebind_k::const_reference` **key_const_reference**
- typedef `value_type::first_type` **key_type**
- typedef `remove_const< key_type >::type` **rkey_type**
- typedef `remove_const< value_type >::type` **rcvalue_type**
- typedef `_Alloc::template rebind< rkey_type >::other` **rebind_k**
- typedef `_Alloc::template rebind< rcvalue_type >::other` **rebind_v**
- typedef `rebind_v::reference` **reference**
- typedef `std::iterator_traits< it_type >::value_type` **value_type**

Protected Member Functions

- virtual `it_type` **end** ()=0
- `it_type` **end_iterator** () const

Static Protected Member Functions

- static `key_const_reference` **extract_key** (const_reference r_val)

4.202.1 Detailed Description

```
template<typename Node_Cltr, typename Node_Itr, typename _Alloc>
struct __gnu_pbds::detail::branch_policy< Node_Cltr, Node_Itr, _Alloc >
```

Primary template, base class for branch structure policies.

Definition at line 52 of file `branch_policy.hpp`.

The documentation for this struct was generated from the following file:

- [branch_policy.hpp](#)

4.203 `__gnu_pbds::detail::branch_policy< Node_Cltr, Node_Cltr, _Alloc >` Struct Template Reference

Protected Types

- typedef `rebind_v::const_pointer` **const_pointer**
- typedef `rebind_v::const_reference` **const_reference**
- typedef `Node_Cltr::value_type` **it_type**
- typedef `rebind_v::const_reference` **key_const_reference**
- typedef `value_type` **key_type**
- typedef `remove_const< value_type >::type` **rcvalue_type**
- typedef `_Alloc::template rebind< rcvalue_type >::other` **rebind_v**
- typedef `rebind_v::reference` **reference**
- typedef `std::iterator_traits< it_type >::value_type` **value_type**

Protected Member Functions

- virtual `it_type` **end** () const =0
- `it_type` **end_iterator** () const

Static Protected Member Functions

- static `key_const_reference` **extract_key** (const_reference r_val)

4.203.1 Detailed Description

```
template<typename Node_Cltr, typename _Alloc>
struct __gnu_pbds::detail::branch_policy< Node_Cltr, Node_Cltr, _Alloc >
```

Specialization for const iterators.

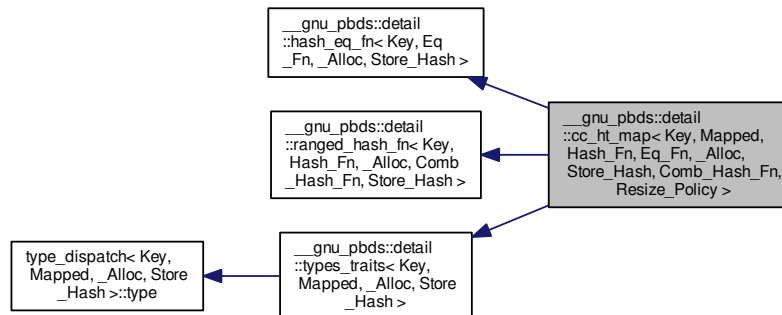
Definition at line 88 of file branch_policy.hpp.

The documentation for this struct was generated from the following file:

- [branch_policy.hpp](#)

4.204 __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy > Class Template Reference

Inheritance diagram for __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >:



Public Types

- enum { **store_hash** }
- typedef _Alloc **allocator_type**
- typedef Comb_Hash_Fn **comb_hash_fn**
- typedef [const_iterator](#) **const_iterator**
- typedef traits_base::const_pointer **const_pointer**
- typedef traits_base::const_reference **const_reference**
- typedef _Alloc::difference_type **difference_type**
- typedef Eq_Fn **eq_fn**
- typedef Hash_Fn **hash_fn**
- typedef [iterator](#) **iterator**
- typedef traits_base::key_const_pointer **key_const_pointer**

- typedef traits_base::key_const_reference **key_const_reference**
- typedef traits_base::key_pointer **key_pointer**
- typedef traits_base::key_reference **key_reference**
- typedef traits_base::key_type **key_type**
- typedef traits_base::mapped_const_pointer **mapped_const_pointer**
- typedef traits_base::mapped_const_reference **mapped_const_reference**
- typedef traits_base::mapped_pointer **mapped_pointer**
- typedef traits_base::mapped_reference **mapped_reference**
- typedef traits_base::mapped_type **mapped_type**
- typedef __nothrowcopy::indicator **no_throw_indicator**
- typedef [point_const_iterator](#) **point_const_iterator**
- typedef [point_iterator](#) **point_iterator**
- typedef traits_base::pointer **pointer**
- typedef traits_base::reference **reference**
- typedef Resize_Policy **resize_policy**
- typedef _Alloc::size_type **size_type**
- typedef integral_constant< int, Store_Hash > **store_extra**
- typedef traits_base::value_type **value_type**

Public Member Functions

- **cc_ht_map** (const Hash_Fn &)
- **cc_ht_map** (const Hash_Fn &, const Eq_Fn &)
- **cc_ht_map** (const Hash_Fn &, const Eq_Fn &, const Comb_Hash_Fn &)
- **cc_ht_map** (const Hash_Fn &, const Eq_Fn &, const Comb_Hash_Fn &, const Resize_Policy &)
- **cc_ht_map** (const [cc_ht_map](#)< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy > &)
- iterator **begin** ()
- const_iterator **begin** () const
- void **clear** ()
- template<typename It >
void **copy_from_range** (It, It)
- bool **empty** () const
- iterator **end** ()
- const_iterator **end** () const
- bool **erase** (key_const_reference)
- template<typename Pred >
size_type **erase_if** (Pred)
- point_iterator **find** (key_const_reference)
- point_const_iterator **find** (key_const_reference) const
- point_iterator **find_end** ()
- point_const_iterator **find_end** () const
- Comb_Hash_Fn & [get_comb_hash_fn](#) ()
- const Comb_Hash_Fn & [get_comb_hash_fn](#) () const
- Eq_Fn & [get_eq_fn](#) ()
- const Eq_Fn & [get_eq_fn](#) () const
- Hash_Fn & [get_hash_fn](#) ()
- const Hash_Fn & [get_hash_fn](#) () const
- Resize_Policy & [get_resize_policy](#) ()
- const Resize_Policy & [get_resize_policy](#) () const

- void **initialize** ()
- `std::pair`< point_iterator, bool > **insert** (const_reference r_val)
- size_type **max_size** () const
- mapped_reference **operator[]** (key_const_reference r_key)
- size_type **size** () const
- void **swap** (`cc_ht_map`< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy > &)

Public Attributes

- no_throw_indicator **m_no_throw_copies_indicator**
- store_extra **m_store_extra_indicator**

Friends

- class **const_iterator_**
- class **iterator_**

4.204.1 Detailed Description

template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool Store_Hash, typename Comb_Hash_Fn, typename Resize_Policy>

class __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >

A collision-chaining hash-based container.

Template Parameters

<i>Key</i>	Key type.
<i>Mapped</i>	Map type.
<i>Hash_Fn</i>	Hashing functor. Default is __gnu_cxx::hash.
<i>Eq_Fn</i>	Equal functor. Default std::equal_to<Key>
<i>_Alloc</i>	Allocator type.
<i>Store_Hash</i>	If key type stores extra metadata. Defaults to false.
<i>Comb_Hash_Fn</i>	Combining hash functor. If Hash_Fn is not null_type, then this is the ranged-hash functor; otherwise, this is the range-hashing functor. XXX(See Design::Hash-Based Containers::Hash Policies.) Default direct_mask_range_hashing.
<i>Resize_Policy</i>	Resizes hash. Defaults to hash_standard_resize_policy, using hash_exponential_size_policy and hash_load_check_resize_trigger.

Bases are: detail::hash_eq_fn, Resize_Policy, detail::ranged_hash_fn, detail::types_traits. (Optional: detail::debug_map_base.)

Definition at line 139 of file cc_ht_map_.hpp.

4.204.2 Member Enumeration Documentation

4.204.2.1 `template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool Store_Hash, typename Comb_Hash_Fn, typename Resize_Policy> anonymous enum`

Value stores hash, true or false.

Definition at line 200 of file `cc_ht_map.hpp`.

4.204.3 Member Function Documentation

4.204.3.1 `template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc , bool Store_Hash, typename Comb_Hash_Fn , typename Resize_Policy > bool __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >::empty () const [inline]`

True if `size() == 0`.

Definition at line 52 of file `cc_ht_map.hpp`.

4.204.3.2 `template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc , bool Store_Hash, typename Comb_Hash_Fn , typename Resize_Policy > Comb_Hash_Fn & __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >::get_comb_hash_fn ()`

Return current `comb_hash_fn`.

Definition at line 70 of file `cc_ht_map.hpp`.

4.204.3.3 `template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc , bool Store_Hash, typename Comb_Hash_Fn , typename Resize_Policy > const Comb_Hash_Fn & __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >::get_comb_hash_fn () const`

Return current `const comb_hash_fn`.

Definition at line 76 of file `cc_ht_map.hpp`.

4.204.3.4 `template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc , bool Store_Hash, typename Comb_Hash_Fn , typename Resize_Policy > Eq_Fn & __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >::get_eq_fn ()`

Return current `eq_fn`.

Definition at line 58 of file `cc_ht_map.hpp`.

4.204.3.5 `template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc , bool Store_Hash, typename Comb_Hash_Fn , typename Resize_Policy > const Eq_Fn & __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >::get_eq_fn () const`

Return current `const eq_fn`.

Definition at line 64 of file `cc_ht_map.hpp`.

4.204.3.6 `template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc , bool Store_Hash, typename Comb_Hash_Fn , typename Resize_Policy > Hash_Fn & __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >::get_hash_fn ()`

Return current hash_fn.

Definition at line 46 of file cc_ht_map_.hpp.

4.204.3.7 `template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc , bool Store_Hash, typename Comb_Hash_Fn , typename Resize_Policy > const Hash_Fn & __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >::get_hash_fn () const`

Return current const hash_fn.

Definition at line 52 of file cc_ht_map_.hpp.

4.204.3.8 `template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc , bool Store_Hash, typename Comb_Hash_Fn , typename Resize_Policy > Resize_Policy & __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >::get_resize_policy ()`

Return current resize_policy.

Definition at line 82 of file cc_ht_map_.hpp.

4.204.3.9 `template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc , bool Store_Hash, typename Comb_Hash_Fn , typename Resize_Policy > const Resize_Policy & __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >::get_resize_policy () const`

Return current const resize_policy.

Definition at line 88 of file cc_ht_map_.hpp.

The documentation for this class was generated from the following file:

- [cc_ht_map_.hpp](#)

4.205 `__gnu_pbds::detail::cond_dealtor< Entry, _Alloc >` Class Template Reference

Public Types

- `typedef HT_Map::entry entry`
- `typedef HT_Map::entry_allocator entry_allocator`
- `typedef __rebind_e::other entry_allocator`
- `typedef entry_allocator::pointer entry_pointer`
- `typedef HT_Map::key_type key_type`

Public Member Functions

- **cond_dealtor** (entry_allocator *p_a, entry *p_e)
- **cond_dealtor** (entry_pointer p_e)
- void **set_key_destruct** ()
- void **set_no_action** ()
- void **set_no_action_destructor** ()

Protected Attributes

- bool **m_key_destruct**
- entry_allocator *const **m_p_a**
- entry *const **m_p_e**

4.205.1 Detailed Description

```
template<typename Entry, typename _Alloc>
class __gnu_pbds::detail::cond_dealtor< Entry, _Alloc >
```

Conditional deallocate constructor argument.

Conditional key destructor, cc_hash.

Definition at line 50 of file cond_dealtor.hpp.

The documentation for this class was generated from the following files:

- [cond_dealtor.hpp](#)
- [cond_key_dtor_entry_dealtor.hpp](#)

4.206 `__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, Tag, Policy_TI >` Struct Template Reference

4.206.1 Detailed Description

```
template<typename Key, typename Mapped, typename _Alloc, typename Tag, typename Policy_TI = null_type>
struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, Tag, Policy_TI >
```

Dispatch mechanism, primary template for associative types.

Definition at line 449 of file tag_and_trait.hpp.

The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

4.207 `__gnu_pbds::detail::container_base_dispatch<_VTp, Cmp_Fn, _Alloc, binary_heap_tag, null_type >` Struct Template Reference

Public Types

- typedef [binary_heap](#)<_VTp, Cmp_Fn, _Alloc > [type](#)

4.207.1 Detailed Description

```
template<typename _VTp, typename Cmp_Fn, typename _Alloc>
struct __gnu_pbds::detail::container_base_dispatch<_VTp, Cmp_Fn, _Alloc, binary_heap_tag, null_type >
```

Specialization for `binary_heap`.

Definition at line 95 of file `priority_queue_base_dispatch.hpp`.

4.207.2 Member Typedef Documentation

4.207.2.1 `template<typename _VTp, typename Cmp_Fn, typename _Alloc > typedef binary_heap<_VTp, Cmp_Fn, _Alloc> __gnu_pbds::detail::container_base_dispatch<_VTp, Cmp_Fn, _Alloc, binary_heap_tag, null_type >::type`

Dispatched type.

Definition at line 99 of file `priority_queue_base_dispatch.hpp`.

The documentation for this struct was generated from the following file:

- [priority_queue_base_dispatch.hpp](#)

4.208 `__gnu_pbds::detail::container_base_dispatch<_VTp, Cmp_Fn, _Alloc, binomial_heap_tag, null_type >` Struct Template Reference

Public Types

- typedef [binomial_heap](#)<_VTp, Cmp_Fn, _Alloc > [type](#)

4.208.1 Detailed Description

```
template<typename _VTp, typename Cmp_Fn, typename _Alloc>
struct __gnu_pbds::detail::container_base_dispatch<_VTp, Cmp_Fn, _Alloc, binomial_heap_tag, null_type >
```

Specialization for `binomial_heap`.

Definition at line 77 of file `priority_queue_base_dispatch.hpp`.

4.208.2 Member Typedef Documentation

4.208.2.1 `template<typename _VTp, typename Cmp_Fn, typename _Alloc> typedef binomial_heap<_VTp, Cmp_Fn, _Alloc>
__gnu_pbds::detail::container_base_dispatch<_VTp, Cmp_Fn, _Alloc, binomial_heap_tag, null_type
>::type`

Dispatched type.

Definition at line 81 of file `priority_queue_base_dispatch.hpp`.

The documentation for this struct was generated from the following file:

- [priority_queue_base_dispatch.hpp](#)

4.209 `__gnu_pbds::detail::container_base_dispatch<_VTp, Cmp_Fn, _Alloc, pairing_heap_tag, null_type>` Struct Template Reference

Public Types

- typedef `pairing_heap<_VTp, Cmp_Fn, _Alloc>` `type`

4.209.1 Detailed Description

```
template<typename _VTp, typename Cmp_Fn, typename _Alloc>
struct __gnu_pbds::detail::container_base_dispatch<_VTp, Cmp_Fn, _Alloc, pairing_heap_tag, null_type>
```

Specialization for `pairing_heap`.

Definition at line 68 of file `priority_queue_base_dispatch.hpp`.

4.209.2 Member Typedef Documentation

4.209.2.1 `template<typename _VTp, typename Cmp_Fn, typename _Alloc> typedef pairing_heap<_VTp, Cmp_Fn, _Alloc>
__gnu_pbds::detail::container_base_dispatch<_VTp, Cmp_Fn, _Alloc, pairing_heap_tag, null_type
>::type`

Dispatched type.

Definition at line 72 of file `priority_queue_base_dispatch.hpp`.

The documentation for this struct was generated from the following file:

- [priority_queue_base_dispatch.hpp](#)

4.210 `__gnu_pbds::detail::container_base_dispatch<_VTp, Cmp_Fn, _Alloc, rc_binomial_heap_tag, null_type>` > Struct Template Reference

Public Types

- typedef [rc_binomial_heap](#)<_VTp, Cmp_Fn, _Alloc > [type](#)

4.210.1 Detailed Description

```
template<typename _VTp, typename Cmp_Fn, typename _Alloc>
struct __gnu_pbds::detail::container_base_dispatch<_VTp, Cmp_Fn, _Alloc, rc_binomial_heap_tag, null_type >
```

Specialization for rc_binary_heap.

Definition at line 86 of file priority_queue_base_dispatch.hpp.

4.210.2 Member Typedef Documentation

4.210.2.1 `template<typename _VTp , typename Cmp_Fn , typename _Alloc > typedef rc_binomial_heap<_VTp, Cmp_Fn, _Alloc> __gnu_pbds::detail::container_base_dispatch<_VTp, Cmp_Fn, _Alloc, rc_binomial_heap_tag, null_type >::type`

Dispatched type.

Definition at line 90 of file priority_queue_base_dispatch.hpp.

The documentation for this struct was generated from the following file:

- [priority_queue_base_dispatch.hpp](#)

4.211 `__gnu_pbds::detail::container_base_dispatch<_VTp, Cmp_Fn, _Alloc, thin_heap_tag, null_type>` > Struct Template Reference

Public Types

- typedef [thin_heap](#)<_VTp, Cmp_Fn, _Alloc > [type](#)

4.211.1 Detailed Description

```
template<typename _VTp, typename Cmp_Fn, typename _Alloc>
struct __gnu_pbds::detail::container_base_dispatch<_VTp, Cmp_Fn, _Alloc, thin_heap_tag, null_type >
```

Specialization for thin_heap.

Definition at line 104 of file priority_queue_base_dispatch.hpp.

4.211.2 Member Typedef Documentation

4.211.2.1 `template<typename VTp, typename Cmp_Fn, typename _Alloc > typedef thin_heap<VTp, Cmp_Fn, _Alloc>
__gnu_pbds::detail::container_base_dispatch<VTp, Cmp_Fn, _Alloc, thin_heap_tag, null_type>::type`

Dispatched type.

Definition at line 108 of file `priority_queue_base_dispatch.hpp`.

The documentation for this struct was generated from the following file:

- [priority_queue_base_dispatch.hpp](#)

4.212 `__gnu_pbds::detail::container_base_dispatch`< Key, Mapped, _Alloc, cc_hash_tag, Policy_TI > Struct Template Reference

Public Types

- typedef `cc_ht_map`< Key, Mapped, at0t, at1t, _Alloc, at3t::value, at4t, at2t > `type`

4.212.1 Detailed Description

```
template<typename Key, typename Mapped, typename _Alloc, typename Policy_TI>  
struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, cc_hash_tag, Policy_TI >
```

Specialization collision-chaining hash map.

Definition at line 258 of file `container_base_dispatch.hpp`.

4.212.2 Member Typedef Documentation

4.212.2.1 `template<typename Key, typename Mapped, typename _Alloc, typename Policy_TI > typedef cc_ht_map<Key,
Mapped, at0t, at1t, _Alloc, at3t::value, at4t, at2t> __gnu_pbds::detail::container_base_dispatch< Key, Mapped,
_Alloc, cc_hash_tag, Policy_TI >::type`

Dispatched type.

Definition at line 275 of file `container_base_dispatch.hpp`.

The documentation for this struct was generated from the following file:

- [container_base_dispatch.hpp](#)

4.213 `__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, gp_hash_tag, Policy_Tl >` Struct Template Reference

Public Types

- typedef [gp_ht_map](#)< Key, Mapped, at0t, at1t, _Alloc, at3t::value, at4t, at5t, at2t > [type](#)

4.213.1 Detailed Description

```
template<typename Key, typename Mapped, typename _Alloc, typename Policy_Tl>
struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, gp_hash_tag, Policy_Tl >
```

Specialization general-probe hash map.

Definition at line 303 of file `container_base_dispatch.hpp`.

4.213.2 Member Typedef Documentation

```
4.213.2.1 template<typename Key , typename Mapped , typename _Alloc , typename Policy_Tl > typedef gp_ht_map<Key,
Mapped, at0t, at1t, _Alloc, at3t::value, at4t, at5t, at2t> __gnu_pbds::detail::container_base_dispatch< Key,
Mapped, _Alloc, gp_hash_tag, Policy_Tl >::type
```

Dispatched type.

Definition at line 322 of file `container_base_dispatch.hpp`.

The documentation for this struct was generated from the following file:

- [container_base_dispatch.hpp](#)

4.214 `__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, list_update_tag, Policy_Tl >` Struct Template Reference

Public Types

- typedef [lu_map](#)< Key, Mapped, at0t, _Alloc, at1t > [type](#)

4.214.1 Detailed Description

```
template<typename Key, typename Mapped, typename _Alloc, typename Policy_Tl>
struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, list_update_tag, Policy_Tl >
```

Specialization for list-update map.

Definition at line 107 of file `container_base_dispatch.hpp`.

4.214.2 Member Typedef Documentation

4.214.2.1 `template<typename Key , typename Mapped , typename _Alloc , typename Policy_Tl > typedef lu_map<Key, Mapped, at0t, _Alloc, at1t> __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, list_update_tag, Policy_Tl >::type`

Dispatched type.

Definition at line 118 of file `container_base_dispatch.hpp`.

The documentation for this struct was generated from the following file:

- [container_base_dispatch.hpp](#)

4.215 `__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, ov_tree_tag, Policy_Tl >` Struct Template Reference

Public Types

- typedef `ov_tree_map< Key, Mapped, at0t, at1t, _Alloc >` `type`

4.215.1 Detailed Description

```
template<typename Key, typename Mapped, typename _Alloc, typename Policy_Tl>
struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, ov_tree_tag, Policy_Tl >
```

Specialization ordered-vector tree map.

Definition at line 227 of file `container_base_dispatch.hpp`.

4.215.2 Member Typedef Documentation

4.215.2.1 `template<typename Key , typename Mapped , typename _Alloc , typename Policy_Tl > typedef ov_tree_map<Key, Mapped, at0t, at1t, _Alloc> __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, ov_tree_tag, Policy_Tl >::type`

Dispatched type.

Definition at line 237 of file `container_base_dispatch.hpp`.

The documentation for this struct was generated from the following file:

- [container_base_dispatch.hpp](#)

4.216 `__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, pat_trie_tag, Policy_Tl >` Struct Template Reference

Public Types

- typedef `pat_trie_map< Key, Mapped, at1t, _Alloc >` **type**

4.216.1 Detailed Description

```
template<typename Key, typename Mapped, typename _Alloc, typename Policy_Tl>
struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, pat_trie_tag, Policy_Tl >
```

Specialization for PATRICIA trie map.

Definition at line 139 of file `container_base_dispatch.hpp`.

The documentation for this struct was generated from the following file:

- [container_base_dispatch.hpp](#)

4.217 `__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, rb_tree_tag, Policy_Tl >` Struct Template Reference

Public Types

- typedef `rb_tree_map< Key, Mapped, at0t, at1t, _Alloc >` **type**

4.217.1 Detailed Description

```
template<typename Key, typename Mapped, typename _Alloc, typename Policy_Tl>
struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, rb_tree_tag, Policy_Tl >
```

Specialization for R-B tree map.

Definition at line 165 of file `container_base_dispatch.hpp`.

4.217.2 Member Typedef Documentation

```
4.217.2.1 template<typename Key , typename Mapped , typename _Alloc , typename Policy_Tl > typedef rb_tree_map<Key,
Mapped, at0t, at1t, _Alloc> __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, rb_tree_tag,
Policy_Tl >::type
```

Dispatched type.

Definition at line 175 of file `container_base_dispatch.hpp`.

The documentation for this struct was generated from the following file:

- [container_base_dispatch.hpp](#)

4.218 `__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, splay_tree_tag, Policy_Tl >` Struct Template Reference

Public Types

- typedef `splay_tree_map< Key, Mapped, at0t, at1t, _Alloc >` `type`

4.218.1 Detailed Description

```
template<typename Key, typename Mapped, typename _Alloc, typename Policy_Tl>
struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, splay_tree_tag, Policy_Tl >
```

Specialization splay tree map.

Definition at line 195 of file `container_base_dispatch.hpp`.

4.218.2 Member Typedef Documentation

4.218.2.1 `template<typename Key , typename Mapped , typename _Alloc , typename Policy_Tl > typedef splay_tree_map<Key, Mapped, at0t, at1t, _Alloc> __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, splay_tree_tag, Policy_Tl >::type`

Dispatched type.

Definition at line 206 of file `container_base_dispatch.hpp`.

The documentation for this struct was generated from the following file:

- `container_base_dispatch.hpp`

4.219 `__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, cc_hash_tag, Policy_Tl >` Struct Template Reference

Public Types

- typedef `cc_ht_set< Key, null_type, at0t, at1t, _Alloc, at3t::value, at4t, at2t >` `type`

4.219.1 Detailed Description

```
template<typename Key, typename _Alloc, typename Policy_Tl>
struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, cc_hash_tag, Policy_Tl >
```

Specialization collision-chaining hash set.

Definition at line 280 of file `container_base_dispatch.hpp`.

4.219.2 Member Typedef Documentation

4.219.2.1 `template<typename Key , typename _Alloc , typename Policy_TI > typedef cc_ht_set<Key, null_type, at0t, at1t, _Alloc, at3t::value, at4t, at2t> __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, cc_hash_tag, Policy_TI >::type`

Dispatched type.

Definition at line 298 of file `container_base_dispatch.hpp`.

The documentation for this struct was generated from the following file:

- [container_base_dispatch.hpp](#)

4.220 __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, gp_hash_tag, Policy_TI > Struct Template Reference

Public Types

- `typedef gp_ht_set< Key, null_type, at0t, at1t, _Alloc, at3t::value, at4t, at5t, at2t > type`

4.220.1 Detailed Description

```
template<typename Key, typename _Alloc, typename Policy_TI>
struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, gp_hash_tag, Policy_TI >
```

Specialization general-probe hash set.

Definition at line 327 of file `container_base_dispatch.hpp`.

4.220.2 Member Typedef Documentation

4.220.2.1 `template<typename Key , typename _Alloc , typename Policy_TI > typedef gp_ht_set<Key, null_type, at0t, at1t, _Alloc, at3t::value, at4t, at5t, at2t> __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, gp_hash_tag, Policy_TI >::type`

Dispatched type.

Definition at line 347 of file `container_base_dispatch.hpp`.

The documentation for this struct was generated from the following file:

- [container_base_dispatch.hpp](#)

4.221 `__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, list_update_tag, Policy_TI >` Struct Template Reference

Public Types

- `typedef lu_set< Key, null_type, at0t, _Alloc, at1t > type`

4.221.1 Detailed Description

```
template<typename Key, typename _Alloc, typename Policy_TI>
struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, list_update_tag, Policy_TI >
```

Specialization for list-update set.

Definition at line 123 of file `container_base_dispatch.hpp`.

4.221.2 Member Typedef Documentation

4.221.2.1 `template<typename Key , typename _Alloc , typename Policy_TI > typedef lu_set<Key, null_type, at0t, _Alloc, at1t> __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, list_update_tag, Policy_TI >::type`

Dispatched type.

Definition at line 134 of file `container_base_dispatch.hpp`.

The documentation for this struct was generated from the following file:

- [container_base_dispatch.hpp](#)

4.222 `__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, ov_tree_tag, Policy_TI >` Struct Template Reference

Public Types

- `typedef ov_tree_set< Key, null_type, at0t, at1t, _Alloc > type`

4.222.1 Detailed Description

```
template<typename Key, typename _Alloc, typename Policy_TI>
struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, ov_tree_tag, Policy_TI >
```

Specialization ordered-vector tree set.

Definition at line 242 of file `container_base_dispatch.hpp`.

4.222.2 Member Typedef Documentation

4.222.2.1 `template<typename Key , typename _Alloc , typename Policy_TI > typedef ov_tree_set<Key, null_type, at0t, at1t, _Alloc> __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, ov_tree_tag, Policy_TI >::type`

Dispatched type.

Definition at line 253 of file `container_base_dispatch.hpp`.

The documentation for this struct was generated from the following file:

- [container_base_dispatch.hpp](#)

4.223 __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, pat_trie_tag, Policy_TI > Struct Template Reference

Public Types

- `typedef pat_trie_set< Key, null_type, at1t, _Alloc > type`

4.223.1 Detailed Description

```
template<typename Key, typename _Alloc, typename Policy_TI>
struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, pat_trie_tag, Policy_TI >
```

Specialization for PATRICIA trie set.

Definition at line 151 of file `container_base_dispatch.hpp`.

4.223.2 Member Typedef Documentation

4.223.2.1 `template<typename Key , typename _Alloc , typename Policy_TI > typedef pat_trie_set<Key, null_type, at1t, _Alloc> __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, pat_trie_tag, Policy_TI >::type`

Dispatched type.

Definition at line 160 of file `container_base_dispatch.hpp`.

The documentation for this struct was generated from the following file:

- [container_base_dispatch.hpp](#)

4.224 `__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, rb_tree_tag, Policy_Tl >` Struct Template Reference

Public Types

- `typedef rb_tree_set< Key, null_type, at0t, at1t, _Alloc >` **type**

4.224.1 Detailed Description

```
template<typename Key, typename _Alloc, typename Policy_Tl>
struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, rb_tree_tag, Policy_Tl >
```

Specialization for R-B tree set.

Definition at line 180 of file `container_base_dispatch.hpp`.

The documentation for this struct was generated from the following file:

- [container_base_dispatch.hpp](#)

4.225 `__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, splay_tree_tag, Policy_Tl >` Struct Template Reference

Public Types

- `typedef splay_tree_set< Key, null_type, at0t, at1t, _Alloc >` [type](#)

4.225.1 Detailed Description

```
template<typename Key, typename _Alloc, typename Policy_Tl>
struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, splay_tree_tag, Policy_Tl >
```

Specialization splay tree set.

Definition at line 211 of file `container_base_dispatch.hpp`.

4.225.2 Member Typedef Documentation

4.225.2.1 `template<typename Key , typename _Alloc , typename Policy_Tl > typedef splay_tree_set<Key, null_type, at0t, at1t, _Alloc> __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, splay_tree_tag, Policy_Tl >::type`

Dispatched type.

Definition at line 222 of file `container_base_dispatch.hpp`.

The documentation for this struct was generated from the following file:

- [container_base_dispatch.hpp](#)

4.226 `__gnu_pbds::detail::default_comb_hash_fn` Struct Reference

Public Types

- typedef [direct_mask_range_hashing](#) type

4.226.1 Detailed Description

Primary template, `default_comb_hash_fn`.

Definition at line 80 of file `standard_policies.hpp`.

4.226.2 Member Typedef Documentation

4.226.2.1 `typedef direct_mask_range_hashing __gnu_pbds::detail::default_comb_hash_fn::type`

Dispatched type.

Definition at line 83 of file `standard_policies.hpp`.

The documentation for this struct was generated from the following file:

- [standard_policies.hpp](#)

4.227 `__gnu_pbds::detail::default_eq_fn< Key >` Struct Template Reference

Public Types

- typedef [std::equal_to< Key >](#) type

4.227.1 Detailed Description

```
template<typename Key>
struct __gnu_pbds::detail::default_eq_fn< Key >
```

Primary template, `default_eq_fn`.

Definition at line 67 of file `standard_policies.hpp`.

4.227.2 Member Typedef Documentation

4.227.2.1 `template<typename Key> typedef std::equal_to<Key> __gnu_pbds::detail::default_eq_fn< Key >::type`

Dispatched type.

Definition at line 70 of file `standard_policies.hpp`.

The documentation for this struct was generated from the following file:

- [standard_policies.hpp](#)

4.228 `__gnu_pbds::detail::default_hash_fn< Key >` Struct Template Reference

Public Types

- `typedef std::tr1::hash< Key > type`

4.228.1 Detailed Description

```
template<typename Key>
struct __gnu_pbds::detail::default_hash_fn< Key >
```

Primary template, `default_hash_fn`.

Definition at line 59 of file `standard_policies.hpp`.

4.228.2 Member Typedef Documentation

4.228.2.1 `template<typename Key> typedef std::tr1::hash<Key> __gnu_pbds::detail::default_hash_fn< Key >::type`

Dispatched type.

Definition at line 62 of file `standard_policies.hpp`.

The documentation for this struct was generated from the following file:

- [standard_policies.hpp](#)

4.229 `__gnu_pbds::detail::default_probe_fn< Comb_Probe_Fn >` Struct Template Reference

Public Types

- `typedef cond_type::__type type`

4.229.1 Detailed Description

```
template<typename Comb_Probe_Fn>
struct __gnu_pbds::detail::default_probe_fn< Comb_Probe_Fn >
```

Primary template, default_probe_fn.

Definition at line 117 of file standard_policies.hpp.

4.229.2 Member Typedef Documentation

4.229.2.1 `template<typename Comb_Probe_Fn> typedef cond_type::__type __gnu_pbds::detail::default_probe_fn< Comb_Probe_Fn >::type`

Dispatched type.

Definition at line 129 of file standard_policies.hpp.

The documentation for this struct was generated from the following file:

- [standard_policies.hpp](#)

4.230 __gnu_pbds::detail::default_resize_policy< Comb_Hash_Fn > Struct Template Reference

Public Types

- typedef [hash_standard_resize_policy](#)< size_policy_type, [trigger](#), false, size_type > [type](#)

4.230.1 Detailed Description

```
template<typename Comb_Hash_Fn>
struct __gnu_pbds::detail::default_resize_policy< Comb_Hash_Fn >
```

Primary template, default_resize_policy.

Definition at line 88 of file standard_policies.hpp.

4.230.2 Member Typedef Documentation

4.230.2.1 `template<typename Comb_Hash_Fn> typedef hash_standard_resize_policy<size_policy_type, trigger, false, size_type> __gnu_pbds::detail::default_resize_policy< Comb_Hash_Fn >::type`

Dispatched type.

Definition at line 105 of file standard_policies.hpp.

The documentation for this struct was generated from the following file:

- [standard_policies.hpp](#)

4.231 `__gnu_pbds::detail::default_trie_access_traits< Key >` Struct Template Reference

4.231.1 Detailed Description

```
template<typename Key>
struct __gnu_pbds::detail::default_trie_access_traits< Key >
```

Primary template, `default_trie_access_traits`.

Definition at line 135 of file `standard_policies.hpp`.

The documentation for this struct was generated from the following file:

- [standard_policies.hpp](#)

4.232 `__gnu_pbds::detail::default_trie_access_traits< std::basic_string< Char, Char_Traits, std::allocator< char > > >` Struct Template Reference

Public Types

- typedef [trie_string_access_traits< string_type >](#) `type`

4.232.1 Detailed Description

```
template<typename Char, typename Char_Traits>
struct __gnu_pbds::detail::default_trie_access_traits< std::basic_string< Char, Char_Traits, std::allocator< char > > >
```

Partial specialization, `default_trie_access_traits`.

Definition at line 142 of file `standard_policies.hpp`.

4.232.2 Member Typedef Documentation

```
4.232.2.1 template<typename Char , typename Char_Traits > typedef trie_string_access_traits<string_type>
__gnu_pbds::detail::default_trie_access_traits< std::basic_string< Char, Char_Traits, std::allocator<
char > > >::type
```

Dispatched type.

Definition at line 149 of file `standard_policies.hpp`.

The documentation for this struct was generated from the following file:

- [standard_policies.hpp](#)

4.233 `__gnu_pbds::detail::default_update_policy` Struct Reference

Public Types

- typedef [lu_move_to_front_policy](#) `type`

4.233.1 Detailed Description

Default update policy.

Definition at line 109 of file `standard_policies.hpp`.

4.233.2 Member Typedef Documentation

4.233.2.1 typedef `lu_move_to_front_policy __gnu_pbds::detail::default_update_policy::type`

Dispatched type.

Definition at line 112 of file `standard_policies.hpp`.

The documentation for this struct was generated from the following file:

- [standard_policies.hpp](#)

4.234 `__gnu_pbds::detail::dumnode_const_iterator< Key, Data, _Alloc >` Struct Template Reference

Public Types

- typedef `const_iterator` **const_reference**
- typedef `const_reference` **reference**
- typedef `const_iterator` **value_type**

4.234.1 Detailed Description

```
template<typename Key, typename Data, typename _Alloc>
struct __gnu_pbds::detail::dumnode_const_iterator< Key, Data, _Alloc >
```

Constant node iterator.

Definition at line 52 of file `null_node_metadata.hpp`.

The documentation for this struct was generated from the following file:

- [null_node_metadata.hpp](#)

4.235 `__gnu_pbds::detail::entry_cmp<_VTp, Cmp_Fn, _Alloc, No_Throw>` Struct Template Reference

4.235.1 Detailed Description

```
template<typename _VTp, typename Cmp_Fn, typename _Alloc, bool No_Throw>
struct __gnu_pbds::detail::entry_cmp<_VTp, Cmp_Fn, _Alloc, No_Throw>
```

Entry compare, primary template.

Definition at line 50 of file `entry_cmp.hpp`.

The documentation for this struct was generated from the following file:

- [entry_cmp.hpp](#)

4.236 `__gnu_pbds::detail::entry_cmp<_VTp, Cmp_Fn, _Alloc, false>` Struct Template Reference

Classes

- struct [type](#)

Public Types

- typedef `__rebind_v::other::const_pointer` **entry**

4.236.1 Detailed Description

```
template<typename _VTp, typename Cmp_Fn, typename _Alloc>
struct __gnu_pbds::detail::entry_cmp<_VTp, Cmp_Fn, _Alloc, false>
```

Specialization, false.

Definition at line 62 of file `entry_cmp.hpp`.

The documentation for this struct was generated from the following file:

- [entry_cmp.hpp](#)

4.237 `__gnu_pbds::detail::entry_cmp<_VTp, Cmp_Fn, _Alloc, false>::type` Struct Reference

Inherits `Cmp_Fn`.

Public Member Functions

- **type** (const Cmp_Fn &other)
- bool **operator()** (entry lhs, entry rhs) const

4.237.1 Detailed Description

```
template<typename _VTp, typename Cmp_Fn, typename _Alloc>
struct __gnu_pbds::detail::entry_cmp<_VTp, Cmp_Fn, _Alloc, false >::type
```

Compare plus entry.

Definition at line 71 of file entry_cmp.hpp.

The documentation for this struct was generated from the following file:

- [entry_cmp.hpp](#)

4.238 __gnu_pbds::detail::entry_cmp<_VTp, Cmp_Fn, _Alloc, true > Struct Template Reference

Public Types

- typedef Cmp_Fn [type](#)

4.238.1 Detailed Description

```
template<typename _VTp, typename Cmp_Fn, typename _Alloc>
struct __gnu_pbds::detail::entry_cmp<_VTp, Cmp_Fn, _Alloc, true >
```

Specialization, true.

Definition at line 54 of file entry_cmp.hpp.

4.238.2 Member Typedef Documentation

4.238.2.1 `template<typename _VTp , typename Cmp_Fn , typename _Alloc > typedef Cmp_Fn
__gnu_pbds::detail::entry_cmp<_VTp, Cmp_Fn, _Alloc, true >::type`

Compare.

Definition at line 57 of file entry_cmp.hpp.

The documentation for this struct was generated from the following file:

- [entry_cmp.hpp](#)

4.239 `__gnu_pbds::detail::entry_pred<_VTp, Pred, _Alloc, No_Throw >` Struct Template Reference

4.239.1 Detailed Description

```
template<typename _VTp, typename Pred, typename _Alloc, bool No_Throw>
struct __gnu_pbds::detail::entry_pred<_VTp, Pred, _Alloc, No_Throw >
```

Entry predicate primary class template.

Definition at line 50 of file `entry_pred.hpp`.

The documentation for this struct was generated from the following file:

- [entry_pred.hpp](#)

4.240 `__gnu_pbds::detail::entry_pred<_VTp, Pred, _Alloc, false >` Struct Template Reference

Public Types

- typedef `__rebind_v::other::const_pointer` **entry**

4.240.1 Detailed Description

```
template<typename _VTp, typename Pred, typename _Alloc>
struct __gnu_pbds::detail::entry_pred<_VTp, Pred, _Alloc, false >
```

Specialization, `false`.

Definition at line 61 of file `entry_pred.hpp`.

The documentation for this struct was generated from the following file:

- [entry_pred.hpp](#)

4.241 `__gnu_pbds::detail::entry_pred<_VTp, Pred, _Alloc, true >` Struct Template Reference

Public Types

- typedef `Pred` **type**

4.241.1 Detailed Description

```
template<typename _VTp, typename Pred, typename _Alloc>
struct __gnu_pbds::detail::entry_pred< _VTp, Pred, _Alloc, true >
```

Specialization, true.

Definition at line 54 of file entry_pred.hpp.

The documentation for this struct was generated from the following file:

- [entry_pred.hpp](#)

4.242 __gnu_pbds::detail::eq_by_less< Key, Cmp_Fn > Struct Template Reference

Inherits Cmp_Fn.

Public Member Functions

- bool **operator()** (const Key &r_lhs, const Key &r_rhs) const

4.242.1 Detailed Description

```
template<typename Key, class Cmp_Fn>
struct __gnu_pbds::detail::eq_by_less< Key, Cmp_Fn >
```

Equivalence function.

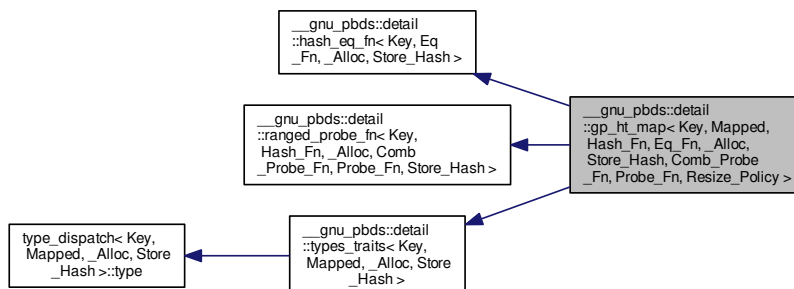
Definition at line 56 of file eq_by_less.hpp.

The documentation for this struct was generated from the following file:

- [eq_by_less.hpp](#)

4.243 __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy > Class Template Reference

Inheritance diagram for __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >:



Public Types

- enum { **store_hash** }
- typedef `_Alloc` **allocator_type**
- typedef `Comb_Probe_Fn` **comb_probe_fn**
- typedef `const_iterator` **const_iterator**
- typedef `traits_base::const_pointer` **const_pointer**
- typedef `traits_base::const_reference` **const_reference**
- typedef `_Alloc::difference_type` **difference_type**
- typedef `Eq_Fn` **eq_fn**
- typedef `Hash_Fn` **hash_fn**
- typedef `iterator` **iterator**
- typedef `traits_base::key_const_pointer` **key_const_pointer**
- typedef `traits_base::key_const_reference` **key_const_reference**
- typedef `traits_base::key_pointer` **key_pointer**
- typedef `traits_base::key_reference` **key_reference**
- typedef `traits_base::key_type` **key_type**
- typedef `traits_base::mapped_const_pointer` **mapped_const_pointer**
- typedef `traits_base::mapped_const_reference` **mapped_const_reference**
- typedef `traits_base::mapped_pointer` **mapped_pointer**
- typedef `traits_base::mapped_reference` **mapped_reference**
- typedef `traits_base::mapped_type` **mapped_type**
- typedef `__nothrowcopy::indicator` **no_throw_indicator**
- typedef `point_const_iterator` **point_const_iterator**
- typedef `point_iterator` **point_iterator**
- typedef `traits_base::pointer` **pointer**
- typedef `Probe_Fn` **probe_fn**
- typedef `traits_base::reference` **reference**
- typedef `Resize_Policy` **resize_policy**
- typedef `_Alloc::size_type` **size_type**
- typedef `integral_constant< int, Store_Hash >` **store_extra**
- typedef `traits_base::value_type` **value_type**

Public Member Functions

- **gp_ht_map** (const `gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >` &)
- **gp_ht_map** (const `Hash_Fn` &)
- **gp_ht_map** (const `Hash_Fn` &, const `Eq_Fn` &)
- **gp_ht_map** (const `Hash_Fn` &, const `Eq_Fn` &, const `Comb_Probe_Fn` &)
- **gp_ht_map** (const `Hash_Fn` &, const `Eq_Fn` &, const `Comb_Probe_Fn` &, const `Probe_Fn` &)
- **gp_ht_map** (const `Hash_Fn` &, const `Eq_Fn` &, const `Comb_Probe_Fn` &, const `Probe_Fn` &, const `Resize_Policy` &)
- iterator **begin** ()
- const_iterator **begin** () const
- void **clear** ()
- template<typename It >
void **copy_from_range** (It, It)
- bool **empty** () const
- iterator **end** ()

- `const_iterator` **end** () const
- `bool` **erase** (key_const_reference)
- `template<typename Pred >`
`size_type` **erase_if** (Pred)
- `point_iterator` **find** (key_const_reference)
- `point_const_iterator` **find** (key_const_reference) const
- `point_iterator` **find_end** ()
- `point_const_iterator` **find_end** () const
- `Comb_Probe_Fn` & [get_comb_probe_fn](#) ()
- `const Comb_Probe_Fn` & [get_comb_probe_fn](#) () const
- `Eq_Fn` & [get_eq_fn](#) ()
- `const Eq_Fn` & [get_eq_fn](#) () const
- `Hash_Fn` & [get_hash_fn](#) ()
- `const Hash_Fn` & [get_hash_fn](#) () const
- `Probe_Fn` & [get_probe_fn](#) ()
- `const Probe_Fn` & [get_probe_fn](#) () const
- `Resize_Policy` & [get_resize_policy](#) ()
- `const Resize_Policy` & [get_resize_policy](#) () const
- `std::pair< point_iterator, bool >` **insert** (const_reference r_val)
- `size_type` **max_size** () const
- `mapped_reference` **operator[]** (key_const_reference r_key)
- `size_type` **size** () const
- `void` **swap** ([gp_ht_map](#)< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy > &)

Public Attributes

- `no_throw_indicator` **m_no_throw_copies_indicator**
- `store_extra` **m_store_extra_indicator**

Friends

- `class` **const_iterator_**
- `class` **iterator_**

4.243.1 Detailed Description

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool Store_Hash, typename
Comb_Probe_Fn, typename Probe_Fn, typename Resize_Policy>
class __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy
>
```

A general-probing hash-based container.

Template Parameters

<i>Key</i>	Key type.
<i>Mapped</i>	Map type.

Template Parameters

<i>Hash_Fn</i>	Hashing functor. Default is <code>__gnu_cxx::hash</code> .
<i>Eq_Fn</i>	Equal functor. Default <code>std::equal_to<Key></code>
<i>_Alloc</i>	Allocator type.
<i>Store_Hash</i>	If key type stores extra metadata. Defaults to false.
<i>Comb_Probe_Fn</i>	Combining probe functor. If <i>Hash_Fn</i> is not null_type, then this is the ranged-probe functor; otherwise, this is the range-hashing functor. XXX See Design::Hash-Based Containers::Hash Policies. Default <code>direct_mask_range_hashing</code> .
<i>Probe_Fn</i>	Probe functor. Defaults to <code>linear_probe_fn</code> , also <code>quadratic_probe_fn</code> .
<i>Resize_Policy</i>	Resizes hash. Defaults to <code>hash_standard_resize_policy</code> , using <code>hash_exponential_size_policy</code> and <code>hash_load_check_resize_trigger</code> .

Bases are: `detail::hash_eq_fn`, `Resize_Policy`, `detail::ranged_probe_fn`, `detail::types_traits`. (Optional: `detail::debug_map_base`.)

Definition at line 142 of file `gp_ht_map.hpp`.

4.243.2 Member Enumeration Documentation

4.243.2.1 `template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool Store_Hash, typename Comb_Probe_Fn, typename Probe_Fn, typename Resize_Policy>` anonymous enum

Value stores hash, true or false.

Definition at line 208 of file `gp_ht_map.hpp`.

4.243.3 Member Function Documentation

4.243.3.1 `template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc , bool Store_Hash, typename Comb_Probe_Fn , typename Probe_Fn , typename Resize_Policy > bool __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >::empty () const` `[inline]`

True if `size() == 0`.

Definition at line 58 of file `gp_ht_map.hpp`.

4.243.3.2 `template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc , bool Store_Hash, typename Comb_Probe_Fn , typename Probe_Fn , typename Resize_Policy > Comb_Probe_Fn & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >::get_comb_probe_fn ()`

Return current `comb_probe_fn`.

Definition at line 82 of file `gp_ht_map.hpp`.

4.243.3.3 `template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc , bool Store_Hash, typename Comb_Probe_Fn , typename Probe_Fn , typename Resize_Policy > const Comb_Probe_Fn & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >::get_comb_probe_fn () const`

Return current const comb_probe_fn.

Definition at line 88 of file gp_ht_map_.hpp.

4.243.3.4 `template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc , bool Store_Hash, typename Comb_Probe_Fn , typename Probe_Fn , typename Resize_Policy > Eq_Fn & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >::get_eq_fn ()`

Return current eq_fn.

Definition at line 58 of file gp_ht_map_.hpp.

4.243.3.5 `template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc , bool Store_Hash, typename Comb_Probe_Fn , typename Probe_Fn , typename Resize_Policy > const Eq_Fn & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >::get_eq_fn () const`

Return current const eq_fn.

Definition at line 64 of file gp_ht_map_.hpp.

4.243.3.6 `template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc , bool Store_Hash, typename Comb_Probe_Fn , typename Probe_Fn , typename Resize_Policy > Hash_Fn & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >::get_hash_fn ()`

Return current hash_fn.

Definition at line 46 of file gp_ht_map_.hpp.

4.243.3.7 `template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc , bool Store_Hash, typename Comb_Probe_Fn , typename Probe_Fn , typename Resize_Policy > const Hash_Fn & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >::get_hash_fn () const`

Return current const hash_fn.

Definition at line 52 of file gp_ht_map_.hpp.

4.243.3.8 `template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc , bool Store_Hash, typename Comb_Probe_Fn , typename Probe_Fn , typename Resize_Policy > Probe_Fn & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >::get_probe_fn ()`

Return current probe_fn.

Definition at line 70 of file gp_ht_map_.hpp.

4.243.3.9 `template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc , bool Store_Hash, typename Comb_Probe_Fn , typename Probe_Fn , typename Resize_Policy > const Probe_Fn & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >::get_probe_fn () const`

Return current const probe_fn.

Definition at line 76 of file `gp_ht_map.hpp`.

4.243.3.10 `template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc , bool Store_Hash, typename Comb_Probe_Fn , typename Probe_Fn , typename Resize_Policy > Resize_Policy & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >::get_resize_policy () const`

Return current resize_policy.

Definition at line 94 of file `gp_ht_map.hpp`.

4.243.3.11 `template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc , bool Store_Hash, typename Comb_Probe_Fn , typename Probe_Fn , typename Resize_Policy > const Resize_Policy & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >::get_resize_policy () const`

Return current const resize_policy.

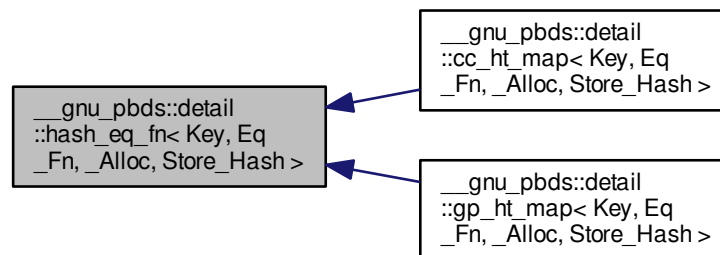
Definition at line 100 of file `gp_ht_map.hpp`.

The documentation for this class was generated from the following file:

- [gp_ht_map.hpp](#)

4.244 `__gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, Store_Hash >` Struct Template Reference

Inheritance diagram for `__gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, Store_Hash >`:



4.244.1 Detailed Description

```
template<typename Key, typename Eq_Fn, typename _Alloc, bool Store_Hash>
struct __gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, Store_Hash >
```

Primary template.

Definition at line 54 of file hash_eq_fn.hpp.

The documentation for this struct was generated from the following file:

- [hash_eq_fn.hpp](#)

4.245 __gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, false > Struct Template Reference

Inherits Eq_Fn.

Public Types

- typedef Eq_Fn **eq_fn_base**
- typedef _Alloc::template rebind< Key >::other **key_allocator**
- typedef key_allocator::const_reference **key_const_reference**

Public Member Functions

- **hash_eq_fn** (const Eq_Fn &r_eq_fn)
- bool **operator()** (key_const_reference r_lhs_key, key_const_reference r_rhs_key) const
- void **swap** (const [hash_eq_fn](#) &other)

4.245.1 Detailed Description

```
template<typename Key, typename Eq_Fn, typename _Alloc>
struct __gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, false >
```

Specialization 1 - The client requests that hash values not be stored.

Definition at line 58 of file hash_eq_fn.hpp.

The documentation for this struct was generated from the following file:

- [hash_eq_fn.hpp](#)

4.246 `__gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, true >` Struct Template Reference

Inherits `Eq_Fn`.

Public Types

- typedef `Eq_Fn` **eq_fn_base**
- typedef `_Alloc::template rebind< Key >::other` **key_allocator**
- typedef `key_allocator::const_reference` **key_const_reference**
- typedef `_Alloc::size_type` **size_type**

Public Member Functions

- **hash_eq_fn** (`const Eq_Fn &r_eq_fn`)
- bool **operator()** (`key_const_reference r_lhs_key, size_type lhs_hash, key_const_reference r_rhs_key, size_type rhs_hash`) const
- void **swap** (`const hash_eq_fn &other`)

4.246.1 Detailed Description

```
template<typename Key, class Eq_Fn, class _Alloc>
struct __gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, true >
```

Specialization 2 - The client requests that hash values be stored.

Definition at line 81 of file `hash_eq_fn.hpp`.

The documentation for this struct was generated from the following file:

- [hash_eq_fn.hpp](#)

4.247 `__gnu_pbds::detail::hash_load_check_resize_trigger_size_base< Size_Type, Hold_Size >` Class Template Reference

4.247.1 Detailed Description

```
template<typename Size_Type, bool Hold_Size>
class __gnu_pbds::detail::hash_load_check_resize_trigger_size_base< Size_Type, Hold_Size >
```

Primary template.

Definition at line 50 of file `hash_load_check_resize_trigger_size_base.hpp`.

The documentation for this class was generated from the following file:

- [hash_load_check_resize_trigger_size_base.hpp](#)

4.248 `__gnu_pbds::detail::hash_load_check_resize_trigger_size_base< Size_Type, true >` Class Template Reference

Protected Types

- typedef `Size_Type` **size_type**

Protected Member Functions

- `size_type` **get_size** () const
- void **set_size** (`size_type` size)
- void **swap** ([hash_load_check_resize_trigger_size_base](#) &other)

4.248.1 Detailed Description

```
template<typename Size_Type>
class __gnu_pbds::detail::hash_load_check_resize_trigger_size_base< Size_Type, true >
```

Specializations.

Definition at line 54 of file `hash_load_check_resize_trigger_size_base.hpp`.

The documentation for this class was generated from the following file:

- [hash_load_check_resize_trigger_size_base.hpp](#)

4.249 `__gnu_pbds::detail::left_child_next_sibling_heap< Value_Type, Cmp_Fn, Node_Metadata, _Alloc >` Class Template Reference

Inherits `Cmp_Fn`.

Public Types

- typedef `_Alloc` **allocator_type**
- typedef `Cmp_Fn` **cmp_fn**
- typedef [left_child_next_sibling_heap_const_iterator_](#)< `node`, `_Alloc` > **const_iterator**
- typedef `__rebind_v::other::const_pointer` **const_pointer**
- typedef `__rebind_v::other::const_reference` **const_reference**
- typedef `_Alloc::difference_type` **difference_type**
- typedef [const_iterator](#) **iterator**
- typedef [left_child_next_sibling_heap_node_point_const_iterator_](#)< `node`, `_Alloc` > **point_const_iterator**
- typedef [point_const_iterator](#) **point_iterator**
- typedef `__rebind_v::other::pointer` **pointer**
- typedef `__rebind_v::other::reference` **reference**
- typedef `_Alloc::size_type` **size_type**
- typedef `Value_Type` **value_type**

Public Member Functions

- `left_child_next_sibling_heap` (const Cmp_Fn &)
- `left_child_next_sibling_heap` (const `left_child_next_sibling_heap` &)
- `iterator begin` ()
- `const_iterator begin` () const
- void `clear` ()
- bool `empty` () const
- `iterator end` ()
- `const_iterator end` () const
- Cmp_Fn & `get_cmp_fn` ()
- const Cmp_Fn & `get_cmp_fn` () const
- size_type `max_size` () const
- size_type `size` () const
- void `swap` (`left_child_next_sibling_heap`< Value_Type, Cmp_Fn, Node_Metadata, _Alloc > &)

Protected Types

- typedef node_allocator::value_type `node`
- typedef _Alloc::template rebind< `left_child_next_sibling_heap_node`< Value_Type, Node_Metadata, _Alloc >::other `node_allocator`
- typedef node_allocator::const_pointer `node_const_pointer`
- typedef Node_Metadata `node_metadata`
- typedef node_allocator::pointer `node_pointer`
- typedef `std::pair`< node_pointer, node_pointer > `node_pointer_pair`

Protected Member Functions

- void `actual_erase_node` (node_pointer)
- void `bubble_to_top` (node_pointer)
- void `clear_imp` (node_pointer)
- node_pointer `get_new_node_for_insert` (const_reference)
- template<typename Pred >
node_pointer `prune` (Pred)
- void `swap_with_parent` (node_pointer, node_pointer)
- void `to_linked_list` ()
- void `value_swap` (`left_child_next_sibling_heap` &)

Static Protected Member Functions

- static void `make_child_of` (node_pointer, node_pointer)
- static node_pointer `parent` (node_pointer)

Protected Attributes

- node_pointer `m_p_root`
- size_type `m_size`

4.249.1 Detailed Description

```
template<typename Value_Type, typename Cmp_Fn, typename Node_Metadata, typename _Alloc>
class __gnu_pbds::detail::left_child_next_sibling_heap< Value_Type, Cmp_Fn, Node_Metadata, _Alloc >
```

Base class for a basic heap.

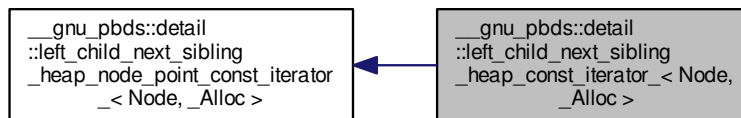
Definition at line 90 of file `left_child_next_sibling_heap.hpp`.

The documentation for this class was generated from the following file:

- [left_child_next_sibling_heap.hpp](#)

4.250 `__gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >`:



Public Types

- typedef `base_type::const_pointer` `const_pointer`
- typedef `base_type::const_reference` `const_reference`
- typedef `_Alloc::difference_type` `difference_type`
- typedef `std::forward_iterator_tag` `iterator_category`
- typedef `base_type::pointer` `pointer`
- typedef `base_type::reference` `reference`
- typedef `base_type::value_type` `value_type`

Public Member Functions

- `left_child_next_sibling_heap_const_iterator_` (node_pointer p_nd)
- `left_child_next_sibling_heap_const_iterator_` ()
- `left_child_next_sibling_heap_const_iterator_` (const `left_child_next_sibling_heap_const_iterator_< Node, _Alloc >` &other)
- `bool operator!=` (const `left_child_next_sibling_heap_const_iterator_< Node, _Alloc >` &other) const
- `bool operator!=` (const `left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >` &other) const
- `const_reference operator*` () const
- `left_child_next_sibling_heap_const_iterator_< Node, _Alloc >` & `operator++` ()
- `left_child_next_sibling_heap_const_iterator_< Node, _Alloc >` `operator++` (int)
- `const_pointer operator->` () const
- `bool operator==` (const `left_child_next_sibling_heap_const_iterator_< Node, _Alloc >` &other) const
- `bool operator==` (const `left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >` &other) const

Public Attributes

- node_pointer `m_p_nd`

4.250.1 Detailed Description

```
template<typename Node, typename _Alloc>
class __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >
```

Const point-type iterator.

Definition at line 60 of file `left_child_next_sibling_heap_/const_iterator.hpp`.

4.250.2 Member Typedef Documentation

```
4.250.2.1 template<typename Node , typename _Alloc > typedef base_type::const_pointer
        __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >::const_pointer
```

Iterator's const pointer type.

Definition at line 81 of file `left_child_next_sibling_heap_/const_iterator.hpp`.

```
4.250.2.2 template<typename Node , typename _Alloc > typedef base_type::const_reference
        __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >::const_reference
```

Iterator's const reference type.

Definition at line 87 of file `left_child_next_sibling_heap_/const_iterator.hpp`.

```
4.250.2.3 template<typename Node , typename _Alloc > typedef _Alloc::difference_type __gnu_pbds::detail::left_child_↔
        next_sibling_heap_const_iterator_< Node, _Alloc >::difference_type
```

Difference type.

Definition at line 72 of file `left_child_next_sibling_heap_/const_iterator.hpp`.

```
4.250.2.4 template<typename Node , typename _Alloc > typedef std::forward_iterator_tag
        __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >::iterator_category
```

Category.

Definition at line 69 of file `left_child_next_sibling_heap_/const_iterator.hpp`.

```
4.250.2.5 template<typename Node , typename _Alloc > typedef base_type::pointer __gnu_pbds::detail::left_child_↔
        next_sibling_heap_const_iterator_< Node, _Alloc >::pointer
```

Iterator's pointer type.

Definition at line 78 of file `left_child_next_sibling_heap_/const_iterator.hpp`.

4.250.2.6 `template<typename Node , typename _Alloc > typedef base_type::reference __gnu_pbds::detail::left_child_↵
next_sibling_heap_const_iterator_< Node, _Alloc >::reference`

Iterator's reference type.

Definition at line 84 of file `left_child_next_sibling_heap_/const_iterator.hpp`.

4.250.2.7 `template<typename Node , typename _Alloc > typedef base_type::value_type
__gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >::value_type`

Iterator's value type.

Definition at line 75 of file `left_child_next_sibling_heap_/const_iterator.hpp`.

4.250.3 Constructor & Destructor Documentation

4.250.3.1 `template<typename Node , typename _Alloc > __gnu_pbds::detail::left_child_next_sibling_↵
heap_const_iterator_< Node, _Alloc >::left_child_next_sibling_heap_const_iterator_ ()
[inline]`

Default constructor.

Definition at line 96 of file `left_child_next_sibling_heap_/const_iterator.hpp`.

4.250.3.2 `template<typename Node , typename _Alloc > __gnu_pbds::detail::left_child_next_sibling_↵
heap_const_iterator_< Node, _Alloc >::left_child_next_sibling_heap_const_iterator_ (const
left_child_next_sibling_heap_const_iterator_< Node, _Alloc > & other) [inline]`

Copy constructor.

Definition at line 101 of file `left_child_next_sibling_heap_/const_iterator.hpp`.

4.250.4 Member Function Documentation

4.250.4.1 `template<typename Node , typename _Alloc > bool __gnu_pbds::detail::left_child_next_sibling_heap_↵
const_iterator_< Node, _Alloc >::operator!= (const left_child_next_sibling_heap_const_iterator_< Node,
_Alloc > & other) const [inline]`

Compares content (negatively) to a different iterator object.

Definition at line 111 of file `left_child_next_sibling_heap_/const_iterator.hpp`.

4.250.4.2 `template<typename Node , typename _Alloc > bool __gnu_pbds::detail::left_child_↵
next_sibling_heap_node_point_const_iterator_< Node, _Alloc >::operator!= (const
left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > & other) const [inline],
[inherited]`

Compares content (negatively) to a different iterator object.

Definition at line 137 of file `left_child_next_sibling_heap_/point_const_iterator.hpp`.

4.250.4.3 `template<typename Node , typename _Alloc > const_reference __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >::operator*() const` [inline], [inherited]

Access.

Definition at line 124 of file `left_child_next_sibling_heap_/point_const_iterator.hpp`.

4.250.4.4 `template<typename Node , typename _Alloc > const_pointer __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >::operator-> () const` [inline], [inherited]

Access.

Definition at line 116 of file `left_child_next_sibling_heap_/point_const_iterator.hpp`.

4.250.4.5 `template<typename Node , typename _Alloc > bool __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >::operator==(const left_child_next_sibling_heap_const_iterator_< Node, _Alloc > & other) const` [inline]

Compares content to a different iterator object.

Definition at line 106 of file `left_child_next_sibling_heap_/const_iterator.hpp`.

4.250.4.6 `template<typename Node , typename _Alloc > bool __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >::operator==(const left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > & other) const` [inline], [inherited]

Compares content to a different iterator object.

Definition at line 132 of file `left_child_next_sibling_heap_/point_const_iterator.hpp`.

The documentation for this class was generated from the following file:

- [left_child_next_sibling_heap_/const_iterator.hpp](#)

4.251 `__gnu_pbds::detail::left_child_next_sibling_heap_node_<_Value,_Metadata,_Alloc>` Struct Template Reference

Public Types

- typedef `_Metadata` **metadata_type**
- typedef `_Alloc::template rebind< this_type >::other::pointer` **node_pointer**
- typedef `_Alloc::size_type` **size_type**
- typedef `_Value` **value_type**

Public Attributes

- metadata_type **m_metadata**
- node_pointer **m_p_l_child**
- node_pointer **m_p_next_sibling**
- node_pointer **m_p_prev_or_parent**
- value_type **m_value**

4.251.1 Detailed Description

```
template<typename _Value, typename _Metadata, typename _Alloc>
struct __gnu_pbds::detail::left_child_next_sibling_heap_node_< _Value, _Metadata, _Alloc >
```

Node.

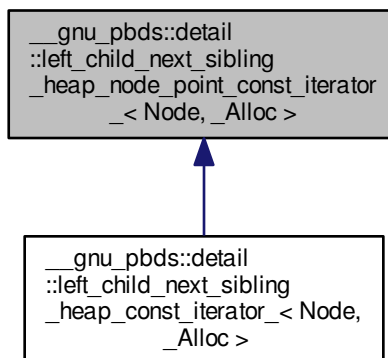
Definition at line 50 of file left_child_next_sibling_heap_/node.hpp.

The documentation for this struct was generated from the following file:

- [left_child_next_sibling_heap_/node.hpp](#)

4.252 __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > Class Template Reference

Inheritance diagram for __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >:



Public Types

- typedef `_Alloc::template rebind< value_type >::other::const_pointer` `const_pointer`
- typedef `_Alloc::template rebind< value_type >::other::const_reference` `const_reference`
- typedef `trivial_iterator_difference_type` `difference_type`
- typedef `trivial_iterator_tag` `iterator_category`
- typedef `_Alloc::template rebind< value_type >::other::pointer` `pointer`
- typedef `_Alloc::template rebind< value_type >::other::reference` `reference`
- typedef `Node::value_type` `value_type`

Public Member Functions

- `left_child_next_sibling_heap_node_point_const_iterator_` (node_pointer p_nd)
- `left_child_next_sibling_heap_node_point_const_iterator_` ()
- `left_child_next_sibling_heap_node_point_const_iterator_` (const `left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >` &other)
- `bool operator!=` (const `left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >` &other) const
- `const_reference operator*` () const
- `const_pointer operator->` () const
- `bool operator==` (const `left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >` &other) const

Public Attributes

- node_pointer `m_p_nd`

Protected Types

- typedef `_Alloc::template rebind< Node >::other::pointer` `node_pointer`

4.252.1 Detailed Description

```
template<typename Node, typename _Alloc>
class __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >
```

Const point-type iterator.

Definition at line 61 of file `left_child_next_sibling_heap_/point_const_iterator.hpp`.

4.252.2 Member Typedef Documentation

4.252.2.1 `template<typename Node , typename _Alloc > typedef _Alloc::template rebind< value_type>::other::const_pointer __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >::const_pointer`

Iterator's const pointer type.

Definition at line 86 of file `left_child_next_sibling_heap_/point_const_iterator.hpp`.

```
4.252.2.2  template<typename Node , typename _Alloc > typedef _Alloc::template rebind< value_type>::other::const_reference
__gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc
>::const_reference
```

Iterator's const reference type.

Definition at line 98 of file left_child_next_sibling_heap_/point_const_iterator.hpp.

```
4.252.2.3  template<typename Node , typename _Alloc > typedef trivial_iterator_difference_type
__gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc
>::difference_type
```

Difference type.

Definition at line 71 of file left_child_next_sibling_heap_/point_const_iterator.hpp.

```
4.252.2.4  template<typename Node , typename _Alloc > typedef trivial_iterator_tag __gnu_pbds::
::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc
>::iterator_category
```

Category.

Definition at line 68 of file left_child_next_sibling_heap_/point_const_iterator.hpp.

```
4.252.2.5  template<typename Node , typename _Alloc > typedef _Alloc::template rebind< value_type>::other::pointer
__gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc
>::pointer
```

Iterator's pointer type.

Definition at line 80 of file left_child_next_sibling_heap_/point_const_iterator.hpp.

```
4.252.2.6  template<typename Node , typename _Alloc > typedef _Alloc::template rebind< value_type>::other::reference
__gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc
>::reference
```

Iterator's reference type.

Definition at line 92 of file left_child_next_sibling_heap_/point_const_iterator.hpp.

```
4.252.2.7  template<typename Node , typename _Alloc > typedef Node::value_type __gnu_pbds::
::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc
>::value_type
```

Iterator's value type.

Definition at line 74 of file left_child_next_sibling_heap_/point_const_iterator.hpp.

4.252.3 Constructor & Destructor Documentation

4.252.3.1 `template<typename Node , typename _Alloc > __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >::left_child_next_sibling_heap_node_point_const_iterator_ ()`
[inline]

Default constructor.

Definition at line 106 of file `left_child_next_sibling_heap_/point_const_iterator.hpp`.

4.252.3.2 `template<typename Node , typename _Alloc > __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >::left_child_next_sibling_heap_node_point_const_iterator_ (const left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > & other)` [inline]

Copy constructor.

Definition at line 111 of file `left_child_next_sibling_heap_/point_const_iterator.hpp`.

4.252.4 Member Function Documentation

4.252.4.1 `template<typename Node , typename _Alloc > bool __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >::operator!= (const left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > & other) const` [inline]

Compares content (negatively) to a different iterator object.

Definition at line 137 of file `left_child_next_sibling_heap_/point_const_iterator.hpp`.

4.252.4.2 `template<typename Node , typename _Alloc > const_reference __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >::operator* () const`
[inline]

Access.

Definition at line 124 of file `left_child_next_sibling_heap_/point_const_iterator.hpp`.

4.252.4.3 `template<typename Node , typename _Alloc > const_pointer __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >::operator-> () const`
[inline]

Access.

Definition at line 116 of file `left_child_next_sibling_heap_/point_const_iterator.hpp`.

```
4.252.4.4  template<typename Node , typename _Alloc > bool __gnu_pbds::detail::left_child_↵
            next_sibling_heap_node_point_const_iterator_< Node, _Alloc >::operator==( const
            left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > & other ) const  [inline]
```

Compares content to a different iterator object.

Definition at line 132 of file `left_child_next_sibling_heap_/point_const_iterator.hpp`.

The documentation for this class was generated from the following file:

- [left_child_next_sibling_heap_/point_const_iterator.hpp](#)

4.253 __gnu_pbds::detail::lu_counter_metadata< Size_Type > Class Template Reference

Public Types

- typedef Size_Type **size_type**

Friends

- class **lu_counter_policy_base**< **size_type** >

4.253.1 Detailed Description

```
template<typename Size_Type = std::size_t>
class __gnu_pbds::detail::lu_counter_metadata< Size_Type >
```

A list-update metadata type that moves elements to the front of the list based on the counter algorithm.

Definition at line 51 of file `lu_counter_metadata.hpp`.

The documentation for this class was generated from the following file:

- [lu_counter_metadata.hpp](#)

4.254 __gnu_pbds::detail::lu_counter_policy_base< Size_Type > Class Template Reference

Protected Types

- typedef Size_Type **size_type**

Protected Member Functions

- [lu_counter_metadata](#)< size_type > **operator()** (size_type max_size) const
- template<typename Metadata_Reference >
bool **operator()** (Metadata_Reference r_data, size_type m_max_count) const

4.254.1 Detailed Description

```
template<typename Size_Type>
class __gnu_pbds::detail::lu_counter_policy_base< Size_Type >
```

Base class for list-update counter policy.

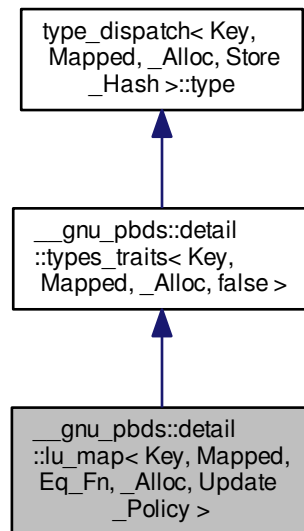
Definition at line 46 of file `lu_counter_metadata.hpp`.

The documentation for this class was generated from the following file:

- [lu_counter_metadata.hpp](#)

4.255 `__gnu_pbds::detail::lu_map< Key, Mapped, Eq_Fn, _Alloc, Update_Policy >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::lu_map< Key, Mapped, Eq_Fn, _Alloc, Update_Policy >`:



Public Types

- typedef `_Alloc` **allocator_type**
- typedef `std::pair< size_type, size_type >` **comp_hash**
- typedef `const_iterator` **const_iterator**
- typedef `traits_base::const_pointer` **const_pointer**
- typedef `traits_base::const_reference` **const_reference**

- typedef _Alloc::difference_type **difference_type**
- typedef Eq_Fn **eq_fn**
- typedef iterator **iterator**
- typedef traits_base::key_const_pointer **key_const_pointer**
- typedef traits_base::key_const_reference **key_const_reference**
- typedef traits_base::key_pointer **key_pointer**
- typedef traits_base::key_reference **key_reference**
- typedef traits_base::key_type **key_type**
- typedef traits_base::mapped_const_pointer **mapped_const_pointer**
- typedef traits_base::mapped_const_reference **mapped_const_reference**
- typedef traits_base::mapped_pointer **mapped_pointer**
- typedef traits_base::mapped_reference **mapped_reference**
- typedef traits_base::mapped_type **mapped_type**
- typedef __nothrowcopy::indicator **no_throw_indicator**
- typedef point_const_iterator **point_const_iterator**
- typedef point_iterator **point_iterator**
- typedef traits_base::pointer **pointer**
- typedef traits_base::reference **reference**
- typedef _Alloc::size_type **size_type**
- typedef integral_constant< int, Store_Hash > **store_extra**
- typedef Update_Policy::metadata_type **update_metadata**
- typedef Update_Policy **update_policy**
- typedef traits_base::value_type **value_type**

Public Member Functions

- **lu_map** (const **lu_map**< Key, Mapped, Eq_Fn, _Alloc, Update_Policy > &)
- template<typename It >
 lu_map (It, It)
- iterator **begin** ()
- const_iterator **begin** () const
- void **clear** ()
- bool **empty** () const
- iterator **end** ()
- const_iterator **end** () const
- bool **erase** (key_const_reference)
- template<typename Pred >
 size_type **erase_if** (Pred)
- point_iterator **find** (key_const_reference r_key)
- point_const_iterator **find** (key_const_reference r_key) const
- **std::pair**< point_iterator, bool > **insert** (const_reference)
- size_type **max_size** () const
- mapped_reference **operator[]** (key_const_reference r_key)
- size_type **size** () const
- void **swap** (**lu_map**< Key, Mapped, Eq_Fn, _Alloc, Update_Policy > &)

Public Attributes

- no_throw_indicator **m_no_throw_copies_indicator**
- store_extra **m_store_extra_indicator**

Protected Member Functions

- `template<typename It >`
`void copy_from_range (It, It)`

Friends

- class `const_iterator_`
- class `iterator_`

4.255.1 Detailed Description

```
template<typename Key, typename Mapped, typename Eq_Fn, typename _Alloc, typename Update_Policy>
class __gnu_pbds::detail::lu_map< Key, Mapped, Eq_Fn, _Alloc, Update_Policy >
```

list-based (with updates) associative container. Skip to the lu, my darling.

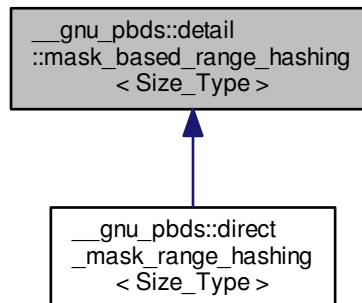
Definition at line 91 of file `lu_map.hpp`.

The documentation for this class was generated from the following file:

- [lu_map.hpp](#)

4.256 `__gnu_pbds::detail::mask_based_range_hashing< Size_Type >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::mask_based_range_hashing< Size_Type >`:



Protected Types

- typedef `Size_Type` **size_type**

Protected Member Functions

- void **notify_resized** (size_type size)
- size_type **range_hash** (size_type hash) const
- void **swap** ([mask_based_range_hashing](#) &other)

4.256.1 Detailed Description

```
template<typename Size_Type>
class __gnu_pbds::detail::mask_based_range_hashing< Size_Type >
```

Range hashing policy.

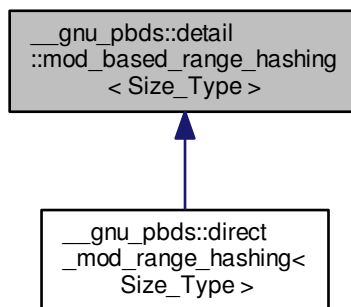
Definition at line 50 of file mask_based_range_hashing.hpp.

The documentation for this class was generated from the following file:

- [mask_based_range_hashing.hpp](#)

4.257 __gnu_pbds::detail::mod_based_range_hashing< Size_Type > Class Template Reference

Inheritance diagram for __gnu_pbds::detail::mod_based_range_hashing< Size_Type >:



Protected Types

- typedef Size_Type **size_type**

Protected Member Functions

- void **notify_resized** (size_type s)
- size_type **range_hash** (size_type s) const
- void **swap** ([mod_based_range_hashing](#) &other)

4.257.1 Detailed Description

```
template<typename Size_Type>
class __gnu_pbds::detail::mod_based_range_hashing< Size_Type >
```

Mod based range hashing.

Definition at line 50 of file `mod_based_range_hashing.hpp`.

The documentation for this class was generated from the following file:

- [mod_based_range_hashing.hpp](#)

4.258 `__gnu_pbds::detail::no_throw_copies< Key, Mapped >` Struct Template Reference

Public Types

- typedef integral_constant< int, __simple > **indicator**

Static Public Attributes

- static const bool **__simple**

4.258.1 Detailed Description

```
template<typename Key, typename Mapped>
struct __gnu_pbds::detail::no_throw_copies< Key, Mapped >
```

Primary template.

Definition at line 61 of file `types_traits.hpp`.

The documentation for this struct was generated from the following file:

- [types_traits.hpp](#)

4.259 `__gnu_pbds::detail::no_throw_copies< Key, null_type >` Struct Template Reference

Public Types

- `typedef integral_constant< int, is_simple< Key >::value > indicator`

4.259.1 Detailed Description

```
template<typename Key>
struct __gnu_pbds::detail::no_throw_copies< Key, null_type >
```

Specialization.

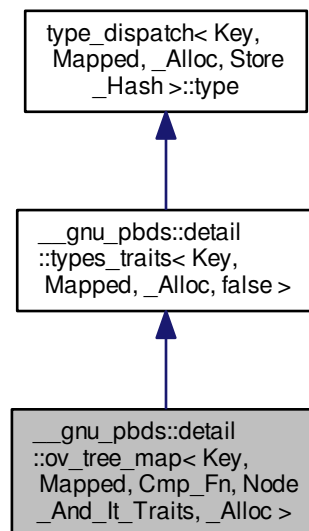
Definition at line 70 of file `types_traits.hpp`.

The documentation for this struct was generated from the following file:

- [types_traits.hpp](#)

4.260 `__gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >`:



Classes

- class [cond_dtor](#)

Public Types

- typedef `_Alloc` **allocator_type**
- typedef `Cmp_Fn` **cmp_fn**
- typedef `std::pair< size_type, size_type >` **comp_hash**
- typedef `point_const_iterator` **const_iterator**
- typedef `traits_base::const_pointer` **const_pointer**
- typedef `traits_base::const_reference` **const_reference**
- typedef `ov_tree_tag` **container_category**
- typedef `_Alloc::difference_type` **difference_type**
- typedef `point_iterator` **iterator**
- typedef `traits_base::key_const_pointer` **key_const_pointer**
- typedef `traits_base::key_const_reference` **key_const_reference**
- typedef `traits_base::key_pointer` **key_pointer**
- typedef `traits_base::key_reference` **key_reference**
- typedef `traits_base::key_type` **key_type**
- typedef `traits_base::mapped_const_pointer` **mapped_const_pointer**
- typedef `traits_base::mapped_const_reference` **mapped_const_reference**
- typedef `traits_base::mapped_pointer` **mapped_pointer**
- typedef `traits_base::mapped_reference` **mapped_reference**
- typedef `traits_base::mapped_type` **mapped_type**
- typedef `__nothrowcopy::indicator` **no_throw_indicator**
- typedef `traits_type::node_const_iterator` **node_const_iterator**
- typedef `traits_type::node_iterator` **node_iterator**
- typedef `traits_type::node_update` **node_update**
- typedef `const_pointer` **point_const_iterator**
- typedef `pointer` **point_iterator**
- typedef `traits_base::pointer` **pointer**
- typedef `traits_base::reference` **reference**
- typedef `_Alloc::size_type` **size_type**
- typedef `integral_constant< int, Store_Hash >` **store_extra**
- typedef `traits_base::value_type` **value_type**

Public Member Functions

- **ov_tree_map** (const `Cmp_Fn` &)
- **ov_tree_map** (const `Cmp_Fn` &, const `node_update` &)
- **ov_tree_map** (const [ov_tree_map](#)< `Key`, `Mapped`, `Cmp_Fn`, `Node_And_It_Traits`, `_Alloc` > &)
- iterator **begin** ()
- const_iterator **begin** () const
- void **clear** ()
- template<typename `It` >
void **copy_from_range** (`It`, `It`)
- bool **empty** () const
- iterator **end** ()

- `const_iterator` **end** () const
- `bool` **erase** (key_const_reference)
- `iterator` **erase** (iterator it)
- `template<typename Pred >`
`size_type` **erase_if** (Pred)
- `point_iterator` **find** (key_const_reference r_key)
- `point_const_iterator` **find** (key_const_reference r_key) const
- `Cmp_Fn` & **get_cmp_fn** ()
- `const Cmp_Fn` & **get_cmp_fn** () const
- `std::pair< point_iterator, bool >` **insert** (const_reference r_value)
- `void` **join** (`ov_tree_map`< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- `point_iterator` **lower_bound** (key_const_reference r_key)
- `point_const_iterator` **lower_bound** (key_const_reference r_key) const
- `size_type` **max_size** () const
- `node_const_iterator` **node_begin** () const
- `node_iterator` **node_begin** ()
- `node_const_iterator` **node_end** () const
- `node_iterator` **node_end** ()
- `mapped_reference` **operator[]** (key_const_reference r_key)
- `size_type` **size** () const
- `void` **split** (key_const_reference, `ov_tree_map`< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- `void` **swap** (`ov_tree_map`< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- `point_iterator` **upper_bound** (key_const_reference r_key)
- `point_const_iterator` **upper_bound** (key_const_reference r_key) const

Public Attributes

- `no_throw_indicator` **m_no_throw_copies_indicator**
- `store_extra` **m_store_extra_indicator**

4.260.1 Detailed Description

```
template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _Alloc>
class __gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >
```

Ordered-vector tree associative-container.

Definition at line 106 of file `ov_tree_map.hpp`.

4.260.2 Member Function Documentation

4.260.2.1 `template<typename Key , typename Mapped , typename Cmp_Fn , typename Node_And_It_Traits , typename _Alloc >` `ov_tree_map`< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::`node_const_iterator`
`__gnu_pbds::detail::ov_tree_map`< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::`node_begin` () const
`[inline]`

Returns a `const node_iterator` corresponding to the node at the root of the tree.

Definition at line 45 of file `ov_tree_map.hpp`.

```
4.260.2.2  template<typename Key , typename Mapped , typename Cmp_Fn , typename Node_And_It_Traits ,
            typename _Alloc > ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_iterator
            __gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_begin ( )
            [inline]
```

Returns a `node_iterator` corresponding to the node at the root of the tree.

Definition at line 57 of file `ov_tree_map.hpp`.

```
4.260.2.3  template<typename Key , typename Mapped , typename Cmp_Fn , typename Node_And_It_Traits , typename
            _Alloc > ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_const_iterator
            __gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_end ( ) const
            [inline]
```

Returns a `const node_iterator` corresponding to a node just after a leaf of the tree.

Definition at line 51 of file `ov_tree_map.hpp`.

```
4.260.2.4  template<typename Key , typename Mapped , typename Cmp_Fn , typename Node_And_It_Traits ,
            typename _Alloc > ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_iterator
            __gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_end ( )
            [inline]
```

Returns a `node_iterator` corresponding to a node just after a leaf of the tree.

Definition at line 63 of file `ov_tree_map.hpp`.

The documentation for this class was generated from the following file:

- [ov_tree_map.hpp](#)

4.261 `__gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::cond_dtor< Size_Type >` Class Template Reference

Public Member Functions

- **cond_dtor** (value_vector `a_vec`, iterator `&r_last_it`, `Size_Type` `total_size`)
- void **set_no_action** ()

Protected Attributes

- value_vector **m_a_vec**
- const `Size_Type` **m_max_size**
- bool **m_no_action**
- iterator & **m_r_last_it**

4.261.1 Detailed Description

```
template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _Alloc>
template<typename Size_Type>
class __gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::cond_dtor< Size_Type >
```

Conditional destructor.

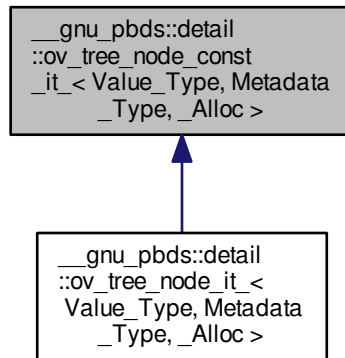
Definition at line 182 of file `ov_tree_map.hpp`.

The documentation for this class was generated from the following file:

- [ov_tree_map.hpp](#)

4.262 __gnu_pbds::detail::ov_tree_node_const_it< Value_Type, Metadata_Type, _Alloc > Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::ov_tree_node_const_it< Value_Type, Metadata_Type, _Alloc >`:



Public Types

- typedef `_Alloc::template rebind< typename remove_const< Value_Type >::type >::other::const_pointer` **const_reference**
- typedef `trivial_iterator_difference_type` **difference_type**
- typedef `trivial_iterator_tag` **iterator_category**
- typedef `_Alloc::template rebind< metadata_type >::other::const_reference` **metadata_const_reference**
- typedef `Metadata_Type` **metadata_type**
- typedef `_Alloc::template rebind< typename remove_const< Value_Type >::type >::other::const_pointer` **reference**
- typedef `_Alloc::template rebind< Value_Type >::other::const_pointer` **value_type**

Public Member Functions

- `ov_tree_node_const_it_` (const_pointer p_nd=0, const_pointer p_begin_nd=0, const_pointer p_end_nd=0, const_metadata_pointer p_metadata=0)
- `this_type get_l_child ()` const
- metadata_const_reference `get_metadata ()` const
- `this_type get_r_child ()` const
- bool `operator!=` (const `this_type` &other) const
- const_reference `operator*` () const
- bool `operator==` (const `this_type` &other) const

Public Attributes

- pointer `m_p_begin_value`
- pointer `m_p_end_value`
- const_metadata_pointer `m_p_metadata`
- pointer `m_p_value`

Protected Types

- typedef `_Alloc::template rebind< Metadata_Type >::other::const_pointer` `const_metadata_pointer`
- typedef `_Alloc::template rebind< Value_Type >::other::const_pointer` `const_pointer`
- typedef `_Alloc::template rebind< Value_Type >::other::pointer` `pointer`
- typedef `ov_tree_node_const_it_< Value_Type, Metadata_Type, _Alloc >` `this_type`

Static Protected Member Functions

- template<typename Ptr >
static Ptr `mid_pointer` (Ptr p_begin, Ptr p_end)

4.262.1 Detailed Description

```
template<typename Value_Type, typename Metadata_Type, typename _Alloc>
class __gnu_pbds::detail::ov_tree_node_const_it_< Value_Type, Metadata_Type, _Alloc >
```

Const node reference.

Definition at line 57 of file `ov_tree_map_/node_iterators.hpp`.

4.262.2 Member Function Documentation

4.262.2.1 `template<typename Value_Type , typename Metadata_Type , typename _Alloc > this_type`
`__gnu_pbds::detail::ov_tree_node_const_it_< Value_Type, Metadata_Type, _Alloc >::get_l_child ()` const
`[inline]`

Returns the node iterator associated with the left node.

Definition at line 142 of file `ov_tree_map_/node_iterators.hpp`.

4.262.2.2 `template<typename Value_Type , typename Metadata_Type , typename _Alloc > this_type
__gnu_pbds::detail::ov_tree_node_const_it_< Value_Type, Metadata_Type, _Alloc >::get_r_child () const
[inline]`

Returns the node iterator associated with the right node.

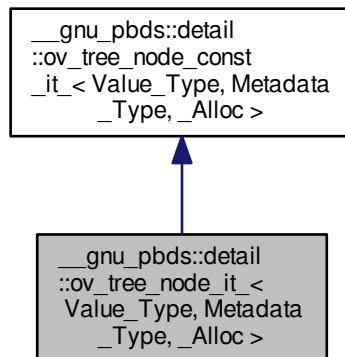
Definition at line 158 of file `ov_tree_map_/node_iterators.hpp`.

The documentation for this class was generated from the following file:

- [ov_tree_map_/node_iterators.hpp](#)

4.263 `__gnu_pbds::detail::ov_tree_node_it_< Value_Type, Metadata_Type, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::ov_tree_node_it_< Value_Type, Metadata_Type, _Alloc >`:



Public Types

- `typedef _Alloc::template rebind< typename remove_const< Value_Type >::type >::other::pointer const_` ← **reference**
- `typedef trivial_iterator_difference_type difference_type`
- `typedef trivial_iterator_tag iterator_category`
- `typedef _Alloc::template rebind< metadata_type >::other::const_reference metadata_const_reference`
- `typedef Metadata_Type metadata_type`
- `typedef _Alloc::template rebind< typename remove_const< Value_Type >::type >::other::pointer reference`
- `typedef _Alloc::template rebind< Value_Type >::other::pointer value_type`

Public Member Functions

- `ov_tree_node_it_` (const_pointer p_nd=0, const_pointer p_begin_nd=0, const_pointer p_end_nd=0, const_metadata_pointer p_metadata=0)
- `ov_tree_node_it_` `get_l_child` () const
- metadata_const_reference `get_metadata` () const
- `ov_tree_node_it_` `get_r_child` () const
- bool `operator!=` (const `this_type` &other) const
- reference `operator*` () const
- bool `operator==` (const `this_type` &other) const

Public Attributes

- pointer `m_p_begin_value`
- pointer `m_p_end_value`
- const_metadata_pointer `m_p_metadata`
- pointer `m_p_value`

Static Protected Member Functions

- template<typename Ptr >
static Ptr `mid_pointer` (Ptr p_begin, Ptr p_end)

4.263.1 Detailed Description

```
template<typename Value_Type, typename Metadata_Type, typename _Alloc>
class __gnu_pbds::detail::ov_tree_node_it_< Value_Type, Metadata_Type, _Alloc >
```

Node reference.

Definition at line 204 of file `ov_tree_map_/node_iterators.hpp`.

4.263.2 Member Function Documentation

4.263.2.1 `template<typename Value_Type , typename Metadata_Type , typename _Alloc > ov_tree_node_it_`
`__gnu_pbds::detail::ov_tree_node_it_< Value_Type, Metadata_Type, _Alloc >::get_l_child ()` const
`[inline]`

Returns the node reference associated with the left node.

Definition at line 252 of file `ov_tree_map_/node_iterators.hpp`.

4.263.2.2 `template<typename Value_Type , typename Metadata_Type , typename _Alloc > ov_tree_node_it_`
`__gnu_pbds::detail::ov_tree_node_it_< Value_Type, Metadata_Type, _Alloc >::get_r_child ()` const
`[inline]`

Returns the node reference associated with the right node.

Definition at line 268 of file `ov_tree_map_/node_iterators.hpp`.

4.263.2.3 `template<typename Value_Type , typename Metadata_Type , typename _Alloc > reference
__gnu_pbds::detail::ov_tree_node_it_< Value_Type, Metadata_Type, _Alloc >::operator* () const
[inline]`

Access.

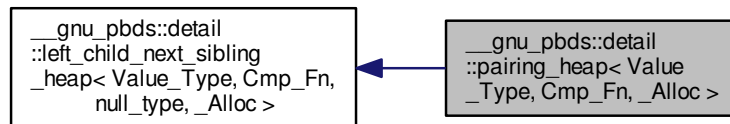
Definition at line 247 of file `ov_tree_map_/node_iterators.hpp`.

The documentation for this class was generated from the following file:

- [ov_tree_map_/node_iterators.hpp](#)

4.264 `__gnu_pbds::detail::pairing_heap< Value_Type, Cmp_Fn, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::pairing_heap< Value_Type, Cmp_Fn, _Alloc >`:



Public Types

- typedef `_Alloc` **allocator_type**
- typedef `Cmp_Fn` **cmp_fn**
- typedef `base_type::const_iterator` **const_iterator**
- typedef `__rebind_a::const_pointer` **const_pointer**
- typedef `__rebind_a::const_reference` **const_reference**
- typedef `_Alloc::difference_type` **difference_type**
- typedef `base_type::iterator` **iterator**
- typedef `base_type::point_const_iterator` **point_const_iterator**
- typedef `base_type::point_iterator` **point_iterator**
- typedef `__rebind_a::pointer` **pointer**
- typedef `__rebind_a::reference` **reference**
- typedef `_Alloc::size_type` **size_type**
- typedef `Value_Type` **value_type**

Public Member Functions

- **pairing_heap** (const Cmp_Fn &)
- **pairing_heap** (const [pairing_heap](#) &)
- **iterator begin** ()
- **const_iterator begin** () const
- void **clear** ()
- bool **empty** () const
- **iterator end** ()
- **const_iterator end** () const
- void **erase** ([point_iterator](#))
- template<typename Pred >
size_type **erase_if** (Pred)
- Cmp_Fn & **get_cmp_fn** ()
- const Cmp_Fn & **get_cmp_fn** () const
- void **join** ([pairing_heap](#) &)
- size_type **max_size** () const
- void **modify** ([point_iterator](#), const_reference)
- void **pop** ()
- [point_iterator](#) **push** (const_reference)
- size_type **size** () const
- template<typename Pred >
void **split** (Pred, [pairing_heap](#) &)
- void **swap** ([pairing_heap](#) &)
- void **swap** ([left_child_next_sibling_heap](#)< Value_Type, Cmp_Fn, [null_type](#), _Alloc > &)
- const_reference **top** () const

Protected Types

- typedef node_allocator::value_type **node**
- typedef _Alloc::template rebind< [left_child_next_sibling_heap_node](#)< Value_Type, [null_type](#), _Alloc > >::other
node_allocator
- typedef node_allocator::const_pointer **node_const_pointer**
- typedef [null_type](#) **node_metadata**
- typedef [std::pair](#)< node_pointer, node_pointer > **node_pointer_pair**

Protected Member Functions

- void **actual_erase_node** (node_pointer)
- void **bubble_to_top** (node_pointer)
- void **clear_imp** (node_pointer)
- template<typename It >
void **copy_from_range** (It, It)
- node_pointer **get_new_node_for_insert** (const_reference)
- node_pointer **prune** (Pred)
- void **swap_with_parent** (node_pointer, node_pointer)
- void **to_linked_list** ()
- void **value_swap** ([left_child_next_sibling_heap](#) &)

Static Protected Member Functions

- static void **make_child_of** (node_pointer, node_pointer)
- static node_pointer **parent** (node_pointer)

Protected Attributes

- node_pointer **m_p_root**
- size_type **m_size**

4.264.1 Detailed Description

```
template<typename Value_Type, typename Cmp_Fn, typename _Alloc>
class __gnu_pbds::detail::pairing_heap< Value_Type, Cmp_Fn, _Alloc >
```

Pairing heap.

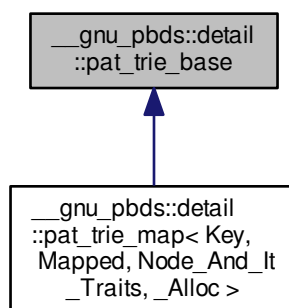
Definition at line 77 of file pairing_heap.hpp.

The documentation for this class was generated from the following file:

- [pairing_heap.hpp](#)

4.265 __gnu_pbds::detail::pat_trie_base Struct Reference

Inheritance diagram for __gnu_pbds::detail::pat_trie_base:



Classes

- class [_Clter](#)
- struct [_Head](#)
- struct [_Inode](#)
- class [_Iter](#)
- struct [_Leaf](#)
- struct [_Metadata](#)
- struct [_Metadata< null_type, _Alloc >](#)
- struct [_Node_base](#)
- class [_Node_citer](#)
- class [_Node_iter](#)

Public Types

- enum [node_type](#) { [i_node](#), [leaf_node](#), [head_node](#) }

4.265.1 Detailed Description

Base type for PATRICIA trees.

Definition at line 51 of file `pat_trie_base.hpp`.

4.265.2 Member Enumeration Documentation

4.265.2.1 enum `__gnu_pbds::detail::pat_trie_base::node_type`

Three types of nodes.

`i_node` is used by `_Inode`, `leaf_node` by `_Leaf`, and `head_node` by `_Head`.

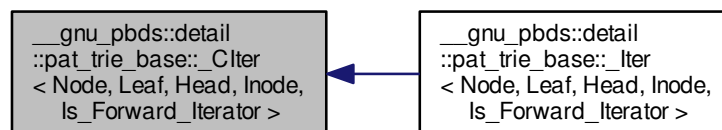
Definition at line 58 of file `pat_trie_base.hpp`.

The documentation for this struct was generated from the following file:

- [pat_trie_base.hpp](#)

4.266 `__gnu_pbds::detail::pat_trie_base::_Clter< Node, Leaf, Head, Inode, Is_Forward_Iterator >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::pat_trie_base::_Clter< Node, Leaf, Head, Inode, Is_Forward_Iterator >`:



Public Types

- typedef __Alloc::template rebind< Head > **__rebind_h**
- typedef __Alloc::template rebind< Inode > **__rebind_in**
- typedef __Alloc::template rebind< Leaf > **__rebind_l**
- typedef __Alloc::template rebind< Node > **__rebind_n**
- typedef allocator_type **_Alloc**
- typedef Node::allocator_type **allocator_type**
- typedef type_traits::const_pointer **const_pointer**
- typedef type_traits::const_reference **const_reference**
- typedef allocator_type::difference_type **difference_type**
- typedef __rebind_h::other::pointer **head_pointer**
- typedef Inode::iterator **inode_iterator**
- typedef __rebind_in::other::pointer **inode_pointer**
- typedef [std::bidirectional_iterator_tag](#) **iterator_category**
- typedef __rebind_l::other::const_pointer **leaf_const_pointer**
- typedef __rebind_l::other::pointer **leaf_pointer**
- typedef __rebind_n::other::pointer **node_pointer**
- typedef type_traits::pointer **pointer**
- typedef type_traits::reference **reference**
- typedef Node::type_traits **type_traits**
- typedef type_traits::value_type **value_type**

Public Member Functions

- **_Clter** (node_pointer p_nd=0)
- **_Clter** (const [_Clter](#)< Node, Leaf, Head, Inode,!Is_Forward_Iterator > &other)
- bool **operator!=** (const [_Clter](#) &other) const
- bool **operator!=** (const [_Clter](#)< Node, Leaf, Head, Inode,!Is_Forward_Iterator > &other) const
- const_reference **operator*** () const
- [_Clter](#) & **operator++** ()
- [_Clter](#) **operator++** (int)
- [_Clter](#) & **operator--** ()
- [_Clter](#) **operator--** (int)
- const_pointer **operator->** () const
- [_Clter](#) & **operator=** (const [_Clter](#) &other)
- [_Clter](#) & **operator=** (const [_Clter](#)< Node, Leaf, Head, Inode,!Is_Forward_Iterator > &other)
- bool **operator==** (const [_Clter](#) &other) const
- bool **operator==** (const [_Clter](#)< Node, Leaf, Head, Inode,!Is_Forward_Iterator > &other) const

Public Attributes

- node_pointer **m_p_nd**

Protected Member Functions

- void **dec** (false_type)
- void **dec** (true_type)
- void **inc** (false_type)
- void **inc** (true_type)

Static Protected Member Functions

- static node_pointer **get_larger_sibling** (node_pointer p_nd)
- static node_pointer **get_smaller_sibling** (node_pointer p_nd)
- static leaf_pointer **leftmost_descendant** (node_pointer p_nd)
- static leaf_pointer **rightmost_descendant** (node_pointer p_nd)

4.266.1 Detailed Description

```
template<typename Node, typename Leaf, typename Head, typename Inode, bool Is_Forward_Iterator>
class __gnu_pbds::detail::pat_trie_base::_CIter< Node, Leaf, Head, Inode, Is_Forward_Iterator >
```

Const iterator.

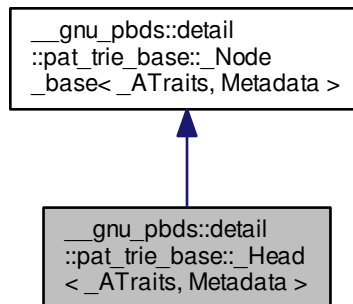
Definition at line 487 of file `pat_trie_base.hpp`.

The documentation for this class was generated from the following file:

- [pat_trie_base.hpp](#)

4.267 `__gnu_pbds::detail::pat_trie_base::_Head<_ATraits, Metadata>` Struct Template Reference

Inheritance diagram for `__gnu_pbds::detail::pat_trie_base::_Head<_ATraits, Metadata>`:



Public Types

- typedef `_Alloc::template rebind<_ATraits>` **__rebind_at**
- typedef `_Alloc::template rebind<_Node_base>` **__rebind_n**
- typedef `_ATraits::const_iterator` **a_const_iterator**
- typedef `_rebind_at::other::const_pointer` **a_const_pointer**
- typedef `_ATraits` **access_traits**
- typedef `_Alloc` **allocator_type**
- typedef `_Node_base<_ATraits, Metadata>` **base_type**
- typedef `base_type::node_pointer` **node_pointer**
- typedef `base_type::type_traits` **type_traits**

Public Attributes

- node_pointer **m_p_max**
- node_pointer **m_p_min**
- node_pointer **m_p_parent**
- const [node_type](#) **m_type**

4.267.1 Detailed Description

```
template<typename ATraits, typename Metadata>
struct __gnu_pbds::detail::pat_trie_base::_Head< ATraits, Metadata >
```

Head node for PATRICIA tree.

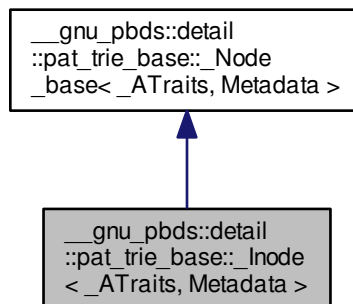
Definition at line 131 of file pat_trie_base.hpp.

The documentation for this struct was generated from the following file:

- [pat_trie_base.hpp](#)

4.268 __gnu_pbds::detail::pat_trie_base::_Inode< ATraits, Metadata > Struct Template Reference

Inheritance diagram for __gnu_pbds::detail::pat_trie_base::_Inode< ATraits, Metadata >:



Classes

- struct [const_iterator](#)
- struct [iterator](#)

Public Types

- enum { **arr_size** }
- typedef `_Alloc::template rebind<_ATraits>` **__rebind_at**
- typedef `_Alloc::template rebind<node_pointer>::other` **__rebind_np**
- typedef `base_type::allocator_type` **_Alloc**
- typedef `base_type::access_traits` **access_traits**
- typedef `_Alloc` **allocator_type**
- typedef `_Node_base<_ATraits, Metadata>` **base_type**
- typedef `__rebind_np::pointer` **node_pointer_pointer**
- typedef `__rebind_np::reference` **node_pointer_reference**
- typedef `_Alloc::size_type` **size_type**
- typedef `base_type::type_traits` **type_traits**
- typedef `type_traits::value_type` **value_type**

Public Member Functions

- **_Inode** (size_type, const a_const_iterator)
- node_pointer **add_child** (node_pointer, a_const_iterator, a_const_iterator, a_const_pointer)
- **const_iterator begin** () const
- **iterator begin** ()
- **const_iterator end** () const
- **iterator end** ()
- **iterator get_child_it** (a_const_iterator, a_const_iterator, a_const_pointer)
- node_pointer **get_child_node** (a_const_iterator, a_const_iterator, a_const_pointer)
- node_const_pointer **get_child_node** (a_const_iterator, a_const_iterator, a_const_pointer) const
- size_type **get_e_ind** () const
- node_const_pointer **get_join_child** (node_const_pointer, a_const_pointer) const
- node_pointer **get_join_child** (node_pointer, a_const_pointer)
- node_pointer **get_lower_bound_child_node** (a_const_iterator, a_const_iterator, size_type, a_const_pointer)
- leaf_pointer **leftmost_descendant** ()
- leaf_const_pointer **leftmost_descendant** () const
- a_const_iterator **pref_b_it** () const
- a_const_iterator **pref_e_it** () const
- void **remove_child** (node_pointer)
- void **remove_child** (iterator)
- void **replace_child** (node_pointer, a_const_iterator, a_const_iterator, a_const_pointer)
- leaf_pointer **rightmost_descendant** ()
- leaf_const_pointer **rightmost_descendant** () const
- bool **should_be_mine** (a_const_iterator, a_const_iterator, size_type, a_const_pointer) const
- void **update_prefixes** (a_const_pointer)

Public Attributes

- node_pointer **m_p_parent**
- const **node_type m_type**

4.268.1 Detailed Description

```
template<typename ATraits, typename Metadata>
struct __gnu_pbds::detail::pat_trie_base::_Inode< ATraits, Metadata >
```

Internal node type, PATRICIA tree.

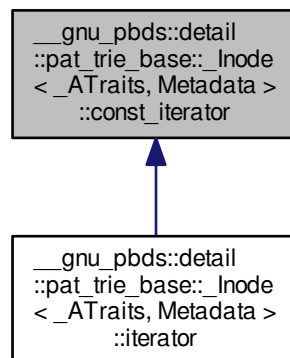
Definition at line 211 of file pat_trie_base.hpp.

The documentation for this struct was generated from the following file:

- [pat_trie_base.hpp](#)

4.269 __gnu_pbds::detail::pat_trie_base::_Inode< ATraits, Metadata >::const_iterator Struct Reference

Inheritance diagram for __gnu_pbds::detail::pat_trie_base::_Inode< ATraits, Metadata >::const_iterator:



Public Types

- typedef `_Alloc::difference_type` **difference_type**
- typedef `std::forward_iterator_tag` **iterator_category**
- typedef `node_pointer_pointer` **pointer**
- typedef `node_pointer_reference` **reference**
- typedef `node_pointer` **value_type**

Public Member Functions

- **const_iterator** (node_pointer_pointer p_p_cur=0, node_pointer_pointer p_p_end=0)
- bool **operator!=** (const [const_iterator](#) &other) const
- node_const_pointer **operator*** () const
- [const_iterator](#) & **operator++** ()
- [const_iterator](#) **operator++** (int)
- const node_pointer_pointer **operator->** () const
- bool **operator==** (const [const_iterator](#) &other) const

Public Attributes

- node_pointer_pointer **m_p_p_cur**
- node_pointer_pointer **m_p_p_end**

4.269.1 Detailed Description

```
template<typename _ATraits, typename Metadata>
struct __gnu_pbds::detail::pat_trie_base::_Inode<_ATraits, Metadata>::const_iterator
```

Constant child iterator.

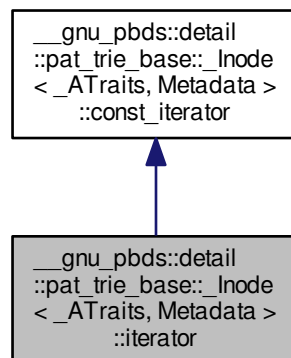
Definition at line 255 of file `pat_trie_base.hpp`.

The documentation for this struct was generated from the following file:

- [pat_trie_base.hpp](#)

4.270 `__gnu_pbds::detail::pat_trie_base::_Inode<_ATraits, Metadata>::iterator` Struct Reference

Inheritance diagram for `__gnu_pbds::detail::pat_trie_base::_Inode<_ATraits, Metadata>::iterator`:



Public Types

- typedef _Alloc::difference_type **difference_type**
- typedef [std::forward_iterator_tag](#) **iterator_category**
- typedef node_pointer_pointer **pointer**
- typedef node_pointer_reference **reference**
- typedef node_pointer **value_type**

Public Member Functions

- **iterator** (node_pointer_pointer p_p_cur=0, node_pointer_pointer p_p_end=0)
- bool **operator!=** (const [const_iterator](#) &other) const
- bool **operator!=** (const [iterator](#) &other) const
- node_const_pointer **operator*** () const
- node_pointer **operator*** ()
- [iterator](#) & **operator++** ()
- [iterator](#) **operator++** (int)
- const node_pointer_pointer **operator->** () const
- node_pointer_pointer **operator->** ()
- bool **operator==** (const [const_iterator](#) &other) const
- bool **operator==** (const [iterator](#) &other) const

Public Attributes

- node_pointer_pointer **m_p_p_cur**
- node_pointer_pointer **m_p_p_end**

4.270.1 Detailed Description

```
template<typename ATraits, typename Metadata>
struct __gnu_pbds::detail::pat_trie_base::_Inode< ATraits, Metadata >::iterator
```

Child iterator.

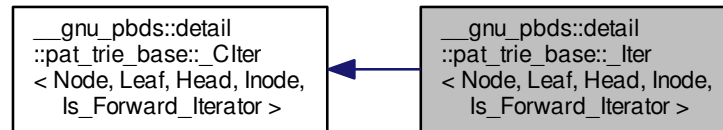
Definition at line 320 of file pat_trie_base.hpp.

The documentation for this struct was generated from the following file:

- [pat_trie_base.hpp](#)

4.271 `__gnu_pbds::detail::pat_trie_base::_Iter< Node, Leaf, Head, Inode, Is_Forward_Iterator >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::pat_trie_base::_Iter< Node, Leaf, Head, Inode, Is_Forward_Iterator >`:



Public Types

- `typedef _Alloc::template rebind< Head > __rebind_h`
- `typedef _Alloc::template rebind< Inode > __rebind_in`
- `typedef _Alloc::template rebind< Leaf > __rebind_l`
- `typedef _Alloc::template rebind< Node > __rebind_n`
- `typedef allocator_type _Alloc`
- `typedef base_type::allocator_type allocator_type`
- `typedef _CIter< Node, Leaf, Head, Inode, Is_Forward_Iterator > base_type`
- `typedef type_traits::const_pointer const_pointer`
- `typedef type_traits::const_reference const_reference`
- `typedef allocator_type::difference_type difference_type`
- `typedef base_type::head_pointer head_pointer`
- `typedef Inode::iterator inode_iterator`
- `typedef base_type::inode_pointer inode_pointer`
- `typedef std::bidirectional_iterator_tag iterator_category`
- `typedef base_type::leaf_const_pointer leaf_const_pointer`
- `typedef base_type::leaf_pointer leaf_pointer`
- `typedef base_type::node_pointer node_pointer`
- `typedef type_traits::pointer pointer`
- `typedef type_traits::reference reference`
- `typedef base_type::type_traits type_traits`
- `typedef type_traits::value_type value_type`

Public Member Functions

- `_Iter (node_pointer p_nd=0)`
- `_Iter (const _Iter< Node, Leaf, Head, Inode,!Is_Forward_Iterator > &other)`
- `bool operator!= (const _CIter &other) const`
- `bool operator!= (const _CIter< Node, Leaf, Head, Inode,!Is_Forward_Iterator > &other) const`
- `reference operator* () const`
- `_Iter & operator++ ()`

- [_Iter](#) **operator++** (int)
- [_Iter](#) & **operator--** ()
- [_Iter](#) **operator--** (int)
- pointer **operator->** () const
- [_Iter](#) & **operator=** (const [_Iter](#) &other)
- [_Iter](#) & **operator=** (const [_Iter](#)< Node, Leaf, Head, Inode, !Is_Forward_Iterator > &other)
- bool **operator==** (const [_CIter](#) &other) const
- bool **operator==** (const [_CIter](#)< Node, Leaf, Head, Inode, !Is_Forward_Iterator > &other) const

Public Attributes

- node_pointer **m_p_nd**

Protected Member Functions

- void **dec** (false_type)
- void **dec** (true_type)
- void **inc** (false_type)
- void **inc** (true_type)

Static Protected Member Functions

- static node_pointer **get_larger_sibling** (node_pointer p_nd)
- static node_pointer **get_smaller_sibling** (node_pointer p_nd)
- static leaf_pointer **leftmost_descendant** (node_pointer p_nd)
- static leaf_pointer **rightmost_descendant** (node_pointer p_nd)

4.271.1 Detailed Description

```
template<typename Node, typename Leaf, typename Head, typename Inode, bool Is_Forward_Iterator>
class __gnu_pbds::detail::pat_trie_base::_Iter< Node, Leaf, Head, Inode, Is_Forward_Iterator >
```

Iterator.

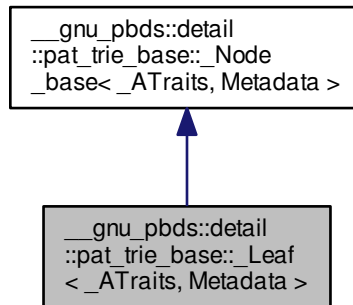
Definition at line 713 of file pat_trie_base.hpp.

The documentation for this class was generated from the following file:

- [pat_trie_base.hpp](#)

4.272 `__gnu_pbds::detail::pat_trie_base::_Leaf<_ATraits, Metadata>` Struct Template Reference

Inheritance diagram for `__gnu_pbds::detail::pat_trie_base::_Leaf<_ATraits, Metadata>`:



Public Types

- `typedef __Alloc::template rebind<_ATraits> __rebind_at`
- `typedef __Alloc::template rebind<_Node_base> __rebind_n`
- `typedef _ATraits::const_iterator a_const_iterator`
- `typedef __rebind_at::other::const_pointer a_const_pointer`
- `typedef _ATraits access_traits`
- `typedef __Alloc allocator_type`
- `typedef _Node_base<_ATraits, Metadata> base_type`
- `typedef type_traits::const_reference const_reference`
- `typedef __rebind_n::other::pointer node_pointer`
- `typedef type_traits::reference reference`
- `typedef base_type::type_traits type_traits`
- `typedef type_traits::value_type value_type`

Public Member Functions

- `_Leaf` (const_reference other)
- reference `value` ()
- const_reference `value` () const

Public Attributes

- node_pointer `m_p_parent`
- const `node_type` `m_type`

4.272.1 Detailed Description

```
template<typename ATraits, typename Metadata>
struct __gnu_pbds::detail::pat_trie_base::_Leaf< ATraits, Metadata >
```

Leaf node for PATRICIA tree.

Definition at line 162 of file pat_trie_base.hpp.

The documentation for this struct was generated from the following file:

- [pat_trie_base.hpp](#)

4.273 __gnu_pbds::detail::pat_trie_base::_Metadata< Metadata, _Alloc > Struct Template Reference

Public Types

- typedef _Alloc::template rebind< Metadata > __rebind_m
- typedef _Alloc **allocator_type**
- typedef __rebind_m::other::const_reference **const_reference**
- typedef Metadata **metadata_type**

Public Member Functions

- const_reference **get_metadata** () const

Public Attributes

- metadata_type **m_metadata**

4.273.1 Detailed Description

```
template<typename Metadata, typename _Alloc>
struct __gnu_pbds::detail::pat_trie_base::_Metadata< Metadata, _Alloc >
```

Metadata base primary template.

Definition at line 67 of file pat_trie_base.hpp.

The documentation for this struct was generated from the following file:

- [pat_trie_base.hpp](#)

4.274 `__gnu_pbds::detail::pat_trie_base::_Metadata< null_type, _Alloc >` Struct Template Reference

Public Types

- typedef `_Alloc` **allocator_type**
- typedef `null_type` **metadata_type**

4.274.1 Detailed Description

```
template<typename _Alloc>
struct __gnu_pbds::detail::pat_trie_base::_Metadata< null_type, _Alloc >
```

Specialization for null metadata.

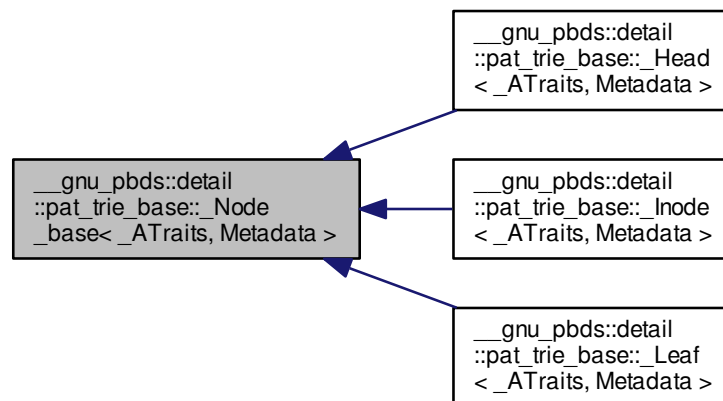
Definition at line 83 of file `pat_trie_base.hpp`.

The documentation for this struct was generated from the following file:

- [pat_trie_base.hpp](#)

4.275 `__gnu_pbds::detail::pat_trie_base::_Node_base< _ATraits, Metadata >` Struct Template Reference

Inheritance diagram for `__gnu_pbds::detail::pat_trie_base::_Node_base< _ATraits, Metadata >`:



Public Types

- typedef __Alloc::template rebind< _ATraits > **__rebind_at**
- typedef __Alloc::template rebind< [_Node_base](#) > **__rebind_n**
- typedef _ATraits::const_iterator **a_const_iterator**
- typedef __rebind_at::other::const_pointer **a_const_pointer**
- typedef _ATraits **access_traits**
- typedef __Alloc **allocator_type**
- typedef __rebind_n::other::pointer **node_pointer**
- typedef _ATraits::type_traits **type_traits**

Public Member Functions

- [_Node_base](#) ([node_type](#) type)

Public Attributes

- node_pointer **m_p_parent**
- const [node_type](#) **m_type**

4.275.1 Detailed Description

```
template<typename _ATraits, typename Metadata>
struct __gnu_pbds::detail::pat_trie_base::_Node_base< _ATraits, Metadata >
```

Node base.

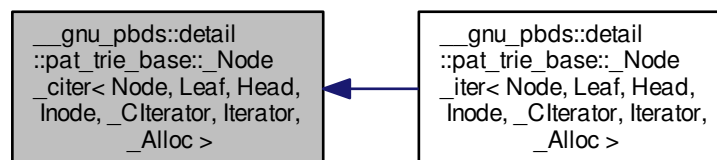
Definition at line 92 of file pat_trie_base.hpp.

The documentation for this struct was generated from the following file:

- [pat_trie_base.hpp](#)

4.276 [__gnu_pbds::detail::pat_trie_base::_Node_citer](#)< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc > Class Template Reference

Inheritance diagram for [__gnu_pbds::detail::pat_trie_base::_Node_citer](#)< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >:



Public Types

- typedef `_Alloc::template rebind< metadata_type > __rebind_m`
- typedef `__rebind_m::other __rebind_ma`
- typedef `value_type const_reference`
- typedef `trivial_iterator_difference_type difference_type`
- typedef `trivial_iterator_tag iterator_category`
- typedef `__rebind_ma::const_reference metadata_const_reference`
- typedef `Node::metadata_type metadata_type`
- typedef `value_type reference`
- typedef `_Alloc::size_type size_type`
- typedef `_Citerator value_type`

Public Member Functions

- `_Node_citer` (`node_pointer p_nd=0, a_const_pointer p_traits=0`)
- `_Node_citer get_child` (`size_type i`) `const`
- `metadata_const_reference get_metadata` () `const`
- `size_type num_children` () `const`
- `bool operator!=` (`const _Node_citer &other`) `const`
- `const_reference operator*` () `const`
- `bool operator==` (`const _Node_citer &other`) `const`
- `std::pair< a_const_iterator, a_const_iterator > valid_prefix` () `const`

Public Attributes

- `node_pointer m_p_nd`
- `a_const_pointer m_p_traits`

Protected Types

- typedef `_Alloc::template rebind< Inode > __rebind_in`
- typedef `_Alloc::template rebind< Leaf > __rebind_l`
- typedef `_Alloc::template rebind< Node > __rebind_n`
- typedef `Node::a_const_iterator a_const_iterator`
- typedef `Node::a_const_pointer a_const_pointer`
- typedef `__rebind_in::other::const_pointer inode_const_pointer`
- typedef `__rebind_in::other::pointer inode_pointer`
- typedef `__rebind_l::other::const_pointer leaf_const_pointer`
- typedef `__rebind_l::other::pointer leaf_pointer`
- typedef `__rebind_n::other::pointer node_pointer`

4.276.1 Detailed Description

```
template<typename Node, typename Leaf, typename Head, typename Inode, typename _Citerator, typename Iterator, typename _↵
Alloc>
class __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _Citerator, Iterator, _Alloc >
```

Node const iterator.

Definition at line 814 of file `pat_trie_base.hpp`.

4.276.2 Member Typedef Documentation

4.276.2.1 `template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator , typename Iterator , typename _Alloc > typedef _Alloc::template rebind<metadata_type> __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >::_rebind_m`

Const metadata reference type.

Definition at line 869 of file `pat_trie_base.hpp`.

4.276.2.2 `template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator , typename Iterator , typename _Alloc > typedef Node::metadata_type __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >::_metadata_type`

Metadata type.

Definition at line 866 of file `pat_trie_base.hpp`.

4.276.3 Member Function Documentation

4.276.3.1 `template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator , typename Iterator , typename _Alloc > _Node_citer __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >::get_child (size_type i) const [inline]`

Returns a `__const` node `__iterator` to the corresponding node's `i`-th child.

Definition at line 911 of file `pat_trie_base.hpp`.

References `std::advance()`.

4.276.3.2 `template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator , typename Iterator , typename _Alloc > metadata_const_reference __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >::get_metadata () const [inline]`

Metadata access.

Definition at line 894 of file `pat_trie_base.hpp`.

4.276.3.3 `template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator , typename Iterator , typename _Alloc > size_type __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >::num_children () const [inline]`

Returns the number of children in the corresponding node.

Definition at line 899 of file `pat_trie_base.hpp`.

References `std::distance()`.

4.276.3.4 `template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator , typename Iterator ,
typename _Alloc > bool __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator,
Iterator, _Alloc >::operator!=(const _Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc > & other)
const [inline]`

Compares content (negatively) to a different iterator object.

Definition at line 927 of file `pat_trie_base.hpp`.

4.276.3.5 `template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator , typename Iterator ,
typename _Alloc > const_reference __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode,
_CIterator, Iterator, _Alloc >::operator*() const [inline]`

Const access; returns the `__const_iterator*` associated with the current leaf.

Definition at line 886 of file `pat_trie_base.hpp`.

4.276.3.6 `template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator , typename Iterator ,
typename _Alloc > bool __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator,
Iterator, _Alloc >::operator==(const _Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc > & other)
const [inline]`

Compares content to a different iterator object.

Definition at line 922 of file `pat_trie_base.hpp`.

4.276.3.7 `template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator , typename
Iterator , typename _Alloc > std::pair<a_const_iterator, a_const_iterator> __gnu_pbds::detail::pat←
_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >::valid_prefix () const
[inline]`

Subtree valid prefix.

Definition at line 880 of file `pat_trie_base.hpp`.

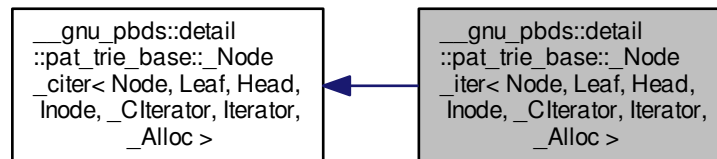
References `std::make_pair()`.

The documentation for this class was generated from the following file:

- [pat_trie_base.hpp](#)

4.277 `__gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >` **Class Template Reference**

Inheritance diagram for `__gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >`:



Public Types

- typedef `_Alloc::template rebind< metadata_type > __rebind_m`
- typedef `__rebind_m::other __rebind_ma`
- typedef `value_type const_reference`
- typedef `trivial_iterator_difference_type difference_type`
- typedef `trivial_iterator_tag iterator_category`
- typedef `__rebind_ma::const_reference metadata_const_reference`
- typedef `Node::metadata_type metadata_type`
- typedef `value_type reference`
- typedef `base_type::size_type size_type`
- typedef `Iterator value_type`

Public Member Functions

- `_Node_iter (node_pointer p_nd=0, a_const_pointer p_traits=0)`
- `_Node_iter get_child (size_type i) const`
- `metadata_const_reference get_metadata () const`
- `size_type num_children () const`
- `bool operator!= (const _Node_citer &other) const`
- `reference operator* () const`
- `bool operator== (const _Node_citer &other) const`
- `std::pair< a_const_iterator, a_const_iterator > valid_prefix () const`

Public Attributes

- node_pointer **m_p_nd**
- a_const_pointer **m_p_traits**

Protected Types

- typedef `_Alloc::template rebind< Inode > __rebind_in`
- typedef `_Alloc::template rebind< Leaf > __rebind_l`
- typedef `Node::a_const_iterator a_const_iterator`
- typedef `__rebind_in::other::const_pointer inode_const_pointer`
- typedef `__rebind_l::other::const_pointer leaf_const_pointer`
- typedef `__rebind_l::other::pointer leaf_pointer`

4.277.1 Detailed Description

```
template<typename Node, typename Leaf, typename Head, typename Inode, typename _Citerator, typename Iterator, typename _↵
Alloc>
class __gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode, _Citerator, Iterator, _Alloc >
```

Node iterator.

Definition at line 943 of file `pat_trie_base.hpp`.

4.277.2 Member Typedef Documentation

4.277.2.1 `template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator
 , typename Iterator , typename _Alloc > typedef _Alloc::template rebind<metadata_type>
 __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc
>::_rebind_m [inherited]`

Const metadata reference type.

Definition at line 869 of file `pat_trie_base.hpp`.

4.277.2.2 `template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator , typename Iterator ,
 typename _Alloc > typedef Node::metadata_type __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf,
 Head, Inode, _CIterator, Iterator, _Alloc >::metadata_type [inherited]`

Metadata type.

Definition at line 866 of file `pat_trie_base.hpp`.

4.277.3 Member Function Documentation

4.277.3.1 `template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator , typename Iterator ,
 typename _Alloc > _Node_iter __gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode,
 _CIterator, Iterator, _Alloc >::get_child (size_type i) const [inline]`

Returns a node __iterator to the corresponding node's i-th child.

Definition at line 976 of file `pat_trie_base.hpp`.

References `std::advance()`, `std::begin()`, `std::distance()`, `std::end()`, and `std::make_pair()`.

4.277.3.2 `template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator , typename Iterator ,
 typename _Alloc > metadata_const_reference __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf,
 Head, Inode, _CIterator, Iterator, _Alloc >::get_metadata () const [inline],[inherited]`

Metadata access.

Definition at line 894 of file `pat_trie_base.hpp`.

4.277.3.3 `template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator , typename Iterator
 , typename _Alloc > size_type __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode,
 _CIterator, Iterator, _Alloc >::num_children () const [inline],[inherited]`

Returns the number of children in the corresponding node.

Definition at line 899 of file `pat_trie_base.hpp`.

References `std::distance()`.

4.277.3.4 `template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator , typename Iterator ,
typename _Alloc > bool __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator,
Iterator, _Alloc >::operator!=(const _Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc > & other)
const [inline],[inherited]`

Compares content (negatively) to a different iterator object.

Definition at line 927 of file pat_trie_base.hpp.

4.277.3.5 `template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator , typename Iterator ,
typename _Alloc > reference __gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode,
_CIterator, Iterator, _Alloc >::operator*() const [inline]`

Access; returns the iterator* associated with the current leaf.

Definition at line 968 of file pat_trie_base.hpp.

4.277.3.6 `template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator , typename Iterator ,
typename _Alloc > bool __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator,
Iterator, _Alloc >::operator==(const _Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc > & other)
const [inline],[inherited]`

Compares content to a different iterator object.

Definition at line 922 of file pat_trie_base.hpp.

4.277.3.7 `template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator , typename
Iterator , typename _Alloc > std::pair<a_const_iterator, a_const_iterator> __gnu_pbds::detail::pat_trie_↵
base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >::valid_prefix () const [inline],
[inherited]`

Subtree valid prefix.

Definition at line 880 of file pat_trie_base.hpp.

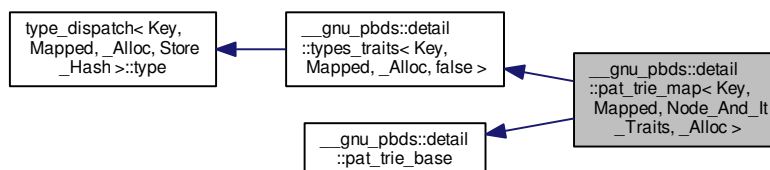
References `std::make_pair()`.

The documentation for this class was generated from the following file:

- [pat_trie_base.hpp](#)

4.278 `__gnu_pbds::detail::pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >`:



Public Types

- typedef traits_type::access_traits **access_traits**
- typedef _Alloc **allocator_type**
- typedef [std::pair](#)< size_type, size_type > **comp_hash**
- typedef point_const_iterator **const_iterator**
- typedef traits_base::const_pointer **const_pointer**
- typedef traits_base::const_reference **const_reference**
- typedef traits_type::const_reverse_iterator **const_reverse_iterator**
- typedef [pat_trie_tag](#) **container_category**
- typedef _Alloc::difference_type **difference_type**
- typedef point_iterator **iterator**
- typedef traits_base::key_const_pointer **key_const_pointer**
- typedef traits_base::key_const_reference **key_const_reference**
- typedef traits_base::key_pointer **key_pointer**
- typedef traits_base::key_reference **key_reference**
- typedef traits_base::key_type **key_type**
- typedef traits_base::mapped_const_pointer **mapped_const_pointer**
- typedef traits_base::mapped_const_reference **mapped_const_reference**
- typedef traits_base::mapped_pointer **mapped_pointer**
- typedef traits_base::mapped_reference **mapped_reference**
- typedef traits_base::mapped_type **mapped_type**
- typedef __nothrowcopy::indicator **no_throw_indicator**
- typedef traits_type::node_const_iterator **node_const_iterator**
- typedef traits_type::node_iterator **node_iterator**
- enum [node_type](#) { **i_node**, **leaf_node**, **head_node** }
- typedef traits_type::node_update **node_update**
- typedef traits_type::const_iterator **point_const_iterator**
- typedef traits_type::iterator **point_iterator**
- typedef traits_base::pointer **pointer**
- typedef traits_base::reference **reference**
- typedef traits_type::reverse_iterator **reverse_iterator**
- typedef _Alloc::size_type **size_type**
- typedef integral_constant< int, Store_Hash > **store_extra**
- typedef traits_base::value_type **value_type**

Public Member Functions

- **pat_trie_map** (const access_traits &)
- **pat_trie_map** (const [pat_trie_map](#)< Key, Mapped, Node_And_It_Traits, _Alloc > &)
- iterator **begin** ()
- const_iterator **begin** () const
- void **clear** ()
- bool **empty** () const
- iterator **end** ()
- const_iterator **end** () const
- bool **erase** (key_const_reference)
- const_iterator **erase** (const_iterator)
- iterator **erase** (iterator)
- const_reverse_iterator **erase** (const_reverse_iterator)

- reverse_iterator **erase** (reverse_iterator)
- template<typename Pred >
size_type **erase_if** (Pred)
- point_iterator **find** (key_const_reference)
- point_const_iterator **find** (key_const_reference) const
- access_traits & **get_access_traits** ()
- const access_traits & **get_access_traits** () const
- node_update & **get_node_update** ()
- const node_update & **get_node_update** () const
- std::pair< point_iterator, bool > **insert** (const_reference)
- void **join** (pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc > &)
- point_iterator **lower_bound** (key_const_reference)
- point_const_iterator **lower_bound** (key_const_reference) const
- size_type **max_size** () const
- node_const_iterator **node_begin** () const
- node_iterator **node_begin** ()
- node_const_iterator **node_end** () const
- node_iterator **node_end** ()
- mapped_reference **operator[]** (key_const_reference r_key)
- reverse_iterator **rbegin** ()
- const_reverse_iterator **rbegin** () const
- reverse_iterator **rend** ()
- const_reverse_iterator **rend** () const
- size_type **size** () const
- void **split** (key_const_reference, pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc > &)
- void **swap** (pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc > &)
- point_iterator **upper_bound** (key_const_reference)
- point_const_iterator **upper_bound** (key_const_reference) const

Public Attributes

- no_throw_indicator **m_no_throw_copies_indicator**
- store_extra **m_store_extra_indicator**

Protected Member Functions

- template<typename It >
void **copy_from_range** (It, It)
- node_pointer **recursive_copy_node** (node_const_pointer)
- void **value_swap** (pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc > &)

4.278.1 Detailed Description

```
template<typename Key, typename Mapped, typename Node_And_It_Traits, typename _Alloc>
class __gnu_pbds::detail::pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >
```

PATRICIA trie.

This implementation loosely borrows ideas from: 1) Fast Mergeable Integer Maps, Okasaki, Gill 1998 2) Ptsset: Sets of integers implemented as Patricia trees, Jean-Christophe Filliatr, 2000.

Definition at line 101 of file pat_trie_.hpp.

4.278.2 Member Enumeration Documentation

4.278.2.1 `enum __gnu_pbds::detail::pat_trie_base::node_type` [inherited]

Three types of nodes.

`i_node` is used by `_Inode`, `leaf_node` by `_Leaf`, and `head_node` by `_Head`.

Definition at line 58 of file `pat_trie_base.hpp`.

4.278.3 Member Function Documentation

4.278.3.1 `template<typename Key , typename Mapped , typename Node_And_It_Traits , typename _Alloc > pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >::node_const_iterator __gnu_pbds::detail::pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >::node_begin () const` [inline]

Returns a `const node_iterator` corresponding to the node at the root of the tree.

Definition at line 101 of file `pat_trie_.hpp`.

4.278.3.2 `template<typename Key , typename Mapped , typename Node_And_It_Traits , typename _Alloc > pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >::node_iterator __gnu_pbds::detail::pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >::node_begin ()` [inline]

Returns a `node_iterator` corresponding to the node at the root of the tree.

Definition at line 107 of file `pat_trie_.hpp`.

4.278.3.3 `template<typename Key , typename Mapped , typename Node_And_It_Traits , typename _Alloc > pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >::node_const_iterator __gnu_pbds::detail::pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >::node_end () const` [inline]

Returns a `const node_iterator` corresponding to a node just after a leaf of the tree.

Definition at line 113 of file `pat_trie_.hpp`.

4.278.3.4 `template<typename Key , typename Mapped , typename Node_And_It_Traits , typename _Alloc > pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >::node_iterator __gnu_pbds::detail::pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >::node_end ()` [inline]

Returns a `node_iterator` corresponding to a node just after a leaf of the tree.

Definition at line 119 of file `pat_trie_.hpp`.

References `std::begin()`, `std::end()`, `std::pair< _T1, _T2 >::first`, `std::make_pair()`, `__gnu_pbds::detail::pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >::node_end()`, and `std::swap()`.

Referenced by `__gnu_pbds::detail::pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >::node_end()`.

The documentation for this class was generated from the following file:

- [pat_trie_.hpp](#)

4.279 `__gnu_pbds::detail::probe_fn_base<_Alloc>` Class Template Reference

4.279.1 Detailed Description

```
template<typename _Alloc>
class __gnu_pbds::detail::probe_fn_base<_Alloc>
```

Probe functor base.

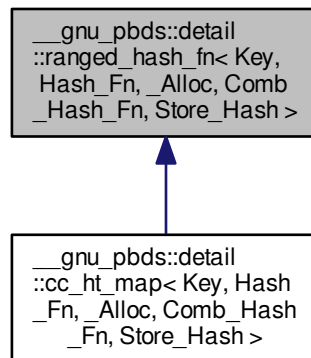
Definition at line 52 of file `probe_fn_base.hpp`.

The documentation for this class was generated from the following file:

- [probe_fn_base.hpp](#)

4.280 `__gnu_pbds::detail::ranged_hash_fn<Key, Hash_Fn, _Alloc, Comb_Hash_Fn, Store_Hash>` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::ranged_hash_fn<Key, Hash_Fn, _Alloc, Comb_Hash_Fn, Store_Hash>`:



4.280.1 Detailed Description

```
template<typename Key, typename Hash_Fn, typename _Alloc, typename Comb_Hash_Fn, bool Store_Hash>
class __gnu_pbds::detail::ranged_hash_fn<Key, Hash_Fn, _Alloc, Comb_Hash_Fn, Store_Hash>
```

Primary template.

Definition at line 55 of file `ranged_hash_fn.hpp`.

The documentation for this class was generated from the following file:

- [ranged_hash_fn.hpp](#)

4.281 `__gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, false >` Class Template Reference

Inherits `Hash_Fn`, and `Comb_Hash_Fn`.

Protected Types

- typedef `Comb_Hash_Fn` **`comb_hash_fn_base`**
- typedef `Hash_Fn` **`hash_fn_base`**
- typedef `_Alloc::template rebind< Key >::other` **`key_allocator`**
- typedef `key_allocator::const_reference` **`key_const_reference`**
- typedef `_Alloc::size_type` **`size_type`**

Protected Member Functions

- **`ranged_hash_fn`** (`size_type`)
- **`ranged_hash_fn`** (`size_type`, `const Hash_Fn &`)
- **`ranged_hash_fn`** (`size_type`, `const Hash_Fn &`, `const Comb_Hash_Fn &`)
- void **`notify_resized`** (`size_type`)
- `size_type` **`operator()`** (`key_const_reference`) `const`
- void **`swap`** ([ranged_hash_fn](#)< `Key`, `Hash_Fn`, `_Alloc`, `Comb_Hash_Fn`, `false` > &)

4.281.1 Detailed Description

```
template<typename Key, typename Hash_Fn, typename _Alloc, typename Comb_Hash_Fn>
class __gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, false >
```

Specialization 1 The client supplies a hash function and a ranged hash function, and requests that hash values not be stored.

Definition at line 71 of file [ranged_hash_fn.hpp](#).

The documentation for this class was generated from the following file:

- [ranged_hash_fn.hpp](#)

4.282 `__gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, true >` Class Template Reference

Inherits `Hash_Fn`, and `Comb_Hash_Fn`.

Protected Types

- typedef Comb_Hash_Fn **comb_hash_fn_base**
- typedef [std::pair](#)< size_type, size_type > **comp_hash**
- typedef Hash_Fn **hash_fn_base**
- typedef _Alloc::template rebind< Key >::other **key_allocator**
- typedef key_allocator::const_reference **key_const_reference**
- typedef _Alloc::size_type **size_type**

Protected Member Functions

- **ranged_hash_fn** (size_type)
- **ranged_hash_fn** (size_type, const Hash_Fn &)
- **ranged_hash_fn** (size_type, const Hash_Fn &, const Comb_Hash_Fn &)
- void **notify_resized** (size_type)
- [comp_hash operator\(\)](#) (key_const_reference) const
- [comp_hash operator\(\)](#) (key_const_reference, size_type) const
- void **swap** ([ranged_hash_fn](#)< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, true > &)

4.282.1 Detailed Description

```
template<typename Key, typename Hash_Fn, typename _Alloc, typename Comb_Hash_Fn>
class __gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, true >
```

Specialization 2 The client supplies a hash function and a ranged hash function, and requests that hash values be stored.

Definition at line 153 of file ranged_hash_fn.hpp.

The documentation for this class was generated from the following file:

- [ranged_hash_fn.hpp](#)

4.283 __gnu_pbds::detail::ranged_hash_fn< Key, null_type, _Alloc, Comb_Hash_Fn, false > Class Template Reference

Inherits Comb_Hash_Fn.

Protected Types

- typedef Comb_Hash_Fn **comb_hash_fn_base**
- typedef _Alloc::size_type **size_type**

Protected Member Functions

- **ranged_hash_fn** (size_type)
- **ranged_hash_fn** (size_type, const Comb_Hash_Fn &)
- **ranged_hash_fn** (size_type, const [null_type](#) &, const Comb_Hash_Fn &)
- void **swap** ([ranged_hash_fn](#)< Key, [null_type](#), _Alloc, Comb_Hash_Fn, false > &)

4.283.1 Detailed Description

```
template<typename Key, typename _Alloc, typename Comb_Hash_Fn>
class __gnu_pbds::detail::ranged_hash_fn< Key, null_type, _Alloc, Comb_Hash_Fn, false >
```

Specialization 3 The client does not supply a hash function (by specifying `null_type` as the `Hash_Fn` parameter), and requests that hash values not be stored.

Definition at line 255 of file `ranged_hash_fn.hpp`.

The documentation for this class was generated from the following file:

- [ranged_hash_fn.hpp](#)

4.284 `__gnu_pbds::detail::ranged_hash_fn< Key, null_type, _Alloc, Comb_Hash_Fn, true >` Class Template Reference

Inherits `Comb_Hash_Fn`.

Protected Types

- typedef `Comb_Hash_Fn` **comb_hash_fn_base**
- typedef `_Alloc::size_type` **size_type**

Protected Member Functions

- **ranged_hash_fn** (size_type)
- **ranged_hash_fn** (size_type, const Comb_Hash_Fn &)
- **ranged_hash_fn** (size_type, const [null_type](#) &, const Comb_Hash_Fn &)
- void **swap** ([ranged_hash_fn](#)< Key, [null_type](#), _Alloc, Comb_Hash_Fn, true > &)

4.284.1 Detailed Description

```
template<typename Key, typename _Alloc, typename Comb_Hash_Fn>
class __gnu_pbds::detail::ranged_hash_fn< Key, null_type, _Alloc, Comb_Hash_Fn, true >
```

Specialization 4 The client does not supply a hash function (by specifying `null_type` as the `Hash_Fn` parameter), and requests that hash values be stored.

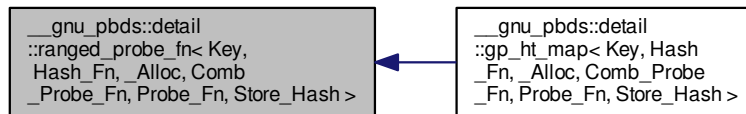
Definition at line 312 of file `ranged_hash_fn.hpp`.

The documentation for this class was generated from the following file:

- [ranged_hash_fn.hpp](#)

4.285 `__gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, Store_Hash >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, Store_Hash >`:



4.285.1 Detailed Description

```
template<typename Key, typename Hash_Fn, typename _Alloc, typename Comb_Probe_Fn, typename Probe_Fn, bool Store_Hash>
class __gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, Store_Hash >
```

Primary template.

Definition at line 55 of file `ranged_probe_fn.hpp`.

The documentation for this class was generated from the following file:

- [ranged_probe_fn.hpp](#)

4.286 `__gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, false >` Class Template Reference

Inherits `Hash_Fn`, `Comb_Probe_Fn`, and `Probe_Fn`.

Protected Types

- typedef `Comb_Probe_Fn` **comb_probe_fn_base**
- typedef `Hash_Fn` **hash_fn_base**
- typedef `_Alloc::template rebind< Key >::other` **key_allocator**
- typedef `key_allocator::const_reference` **key_const_reference**
- typedef `Probe_Fn` **probe_fn_base**
- typedef `_Alloc::size_type` **size_type**

Protected Member Functions

- `ranged_probe_fn` (size_type)
- `ranged_probe_fn` (size_type, const Hash_Fn &)
- `ranged_probe_fn` (size_type, const Hash_Fn &, const Comb_Probe_Fn &)
- `ranged_probe_fn` (size_type, const Hash_Fn &, const Comb_Probe_Fn &, const Probe_Fn &)
- void `notify_resized` (size_type)
- size_type `operator()` (key_const_reference) const
- size_type `operator()` (key_const_reference, size_type, size_type) const
- void `swap` (`ranged_probe_fn`< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, false > &)

4.286.1 Detailed Description

```
template<typename Key, typename Hash_Fn, typename _Alloc, typename Comb_Probe_Fn, typename Probe_Fn>
class __gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, false >
```

Specialization 1 The client supplies a probe function and a ranged probe function, and requests that hash values not be stored.

Definition at line 71 of file `ranged_probe_fn.hpp`.

The documentation for this class was generated from the following file:

- [ranged_probe_fn.hpp](#)

4.287 `__gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, true >` Class Template Reference

Inherits `Hash_Fn`, `Comb_Probe_Fn`, and `Probe_Fn`.

Protected Types

- typedef Comb_Probe_Fn `comb_probe_fn_base`
- typedef `std::pair`< size_type, size_type > `comp_hash`
- typedef Hash_Fn `hash_fn_base`
- typedef _Alloc::template rebind< Key >::other `key_allocator`
- typedef key_allocator::const_reference `key_const_reference`
- typedef Probe_Fn `probe_fn_base`
- typedef _Alloc::size_type `size_type`

Protected Member Functions

- `ranged_probe_fn` (size_type)
- `ranged_probe_fn` (size_type, const Hash_Fn &)
- `ranged_probe_fn` (size_type, const Hash_Fn &, const Comb_Probe_Fn &)
- `ranged_probe_fn` (size_type, const Hash_Fn &, const Comb_Probe_Fn &, const Probe_Fn &)
- void `notify_resized` (size_type)
- `comp_hash operator()` (key_const_reference) const
- size_type `operator()` (key_const_reference, size_type, size_type) const
- size_type `operator()` (key_const_reference, size_type) const
- void `swap` (`ranged_probe_fn`< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, true > &)

4.287.1 Detailed Description

```
template<typename Key, typename Hash_Fn, typename _Alloc, typename Comb_Probe_Fn, typename Probe_Fn>
class __gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, true >
```

Specialization 2- The client supplies a probe function and a ranged probe function, and requests that hash values not be stored.

Definition at line 176 of file ranged_probe_fn.hpp.

The documentation for this class was generated from the following file:

- [ranged_probe_fn.hpp](#)

4.288 __gnu_pbds::detail::ranged_probe_fn< Key, null_type, _Alloc, Comb_Probe_Fn, null_type, false > Class Template Reference

Inherits Comb_Probe_Fn.

Protected Types

- typedef Comb_Probe_Fn **comb_probe_fn_base**
- typedef _Alloc::template rebind< Key >::other **key_allocator**
- typedef key_allocator::const_reference **key_const_reference**
- typedef _Alloc::size_type **size_type**

Protected Member Functions

- **ranged_probe_fn** (size_type size)
- **ranged_probe_fn** (size_type, const Comb_Probe_Fn &r_comb_probe_fn)
- **ranged_probe_fn** (size_type, const [null_type](#) &, const Comb_Probe_Fn &r_comb_probe_fn, const [null_type](#) &)
- void **swap** ([ranged_probe_fn](#) &other)

4.288.1 Detailed Description

```
template<typename Key, typename _Alloc, typename Comb_Probe_Fn>
class __gnu_pbds::detail::ranged_probe_fn< Key, null_type, _Alloc, Comb_Probe_Fn, null_type, false >
```

Specialization 3 and 4 The client does not supply a hash function or probe function, and requests that hash values not be stored.

Definition at line 296 of file ranged_probe_fn.hpp.

The documentation for this class was generated from the following file:

- [ranged_probe_fn.hpp](#)

4.289 `__gnu_pbds::detail::rb_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >` Class Template Reference

Inherits `__gnu_pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >`.

Public Types

- typedef `_Alloc` **allocator_type**
- typedef `Cmp_Fn` **cmp_fn**
- typedef `std::pair< size_type, size_type >` **comp_hash**
- typedef `base_type::const_iterator` **const_iterator**
- typedef `base_type::const_pointer` **const_pointer**
- typedef `base_type::const_reference` **const_reference**
- typedef `base_type::const_reverse_iterator` **const_reverse_iterator**
- typedef `rb_tree_tag` **container_category**
- typedef `_Alloc::difference_type` **difference_type**
- typedef `base_type::iterator` **iterator**
- typedef `base_type::key_const_pointer` **key_const_pointer**
- typedef `base_type::key_const_reference` **key_const_reference**
- typedef `base_type::key_pointer` **key_pointer**
- typedef `base_type::key_reference` **key_reference**
- typedef `base_type::key_type` **key_type**
- typedef `base_type::mapped_const_pointer` **mapped_const_pointer**
- typedef `base_type::mapped_const_reference` **mapped_const_reference**
- typedef `base_type::mapped_pointer` **mapped_pointer**
- typedef `base_type::mapped_reference` **mapped_reference**
- typedef `base_type::mapped_type` **mapped_type**
- typedef `__nothrowcopy::indicator` **no_throw_indicator**
- typedef `traits_type::node_const_iterator` **node_const_iterator**
- typedef `traits_type::node_iterator` **node_iterator**
- typedef `base_type::node_update` **node_update**
- typedef `base_type::const_iterator` **point_const_iterator**
- typedef `base_type::point_iterator` **point_iterator**
- typedef `base_type::pointer` **pointer**
- typedef `base_type::reference` **reference**
- typedef `base_type::reverse_iterator` **reverse_iterator**
- typedef `_Alloc::size_type` **size_type**
- typedef `integral_constant< int, Store_Hash >` **store_extra**
- typedef `base_type::value_type` **value_type**

Public Member Functions

- **rb_tree_map** (const `Cmp_Fn` &)
- **rb_tree_map** (const `Cmp_Fn` &, const `node_update` &)
- **rb_tree_map** (const `rb_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >` &)
- **begin** ()
- **const_iterator begin** () const
- **clear** ()

- `template<typename It >`
 `void copy_from_range (It, It)`
- `bool empty () const`
- `iterator end ()`
- `const_iterator end () const`
- `bool erase (key_const_reference)`
- `iterator erase (iterator)`
- `reverse_iterator erase (reverse_iterator)`
- `template<typename Pred >`
 `size_type erase_if (Pred)`
- `point_iterator find (key_const_reference)`
- `point_const_iterator find (key_const_reference) const`
- `Cmp_Fn & get_cmp_fn ()`
- `const Cmp_Fn & get_cmp_fn () const`
- `std::pair< point_iterator, bool > insert (const_reference)`
- `void join (rb_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)`
- `point_iterator lower_bound (key_const_reference)`
- `point_const_iterator lower_bound (key_const_reference) const`
- `size_type max_size () const`
- `node_const_iterator node_begin () const`
- `node_iterator node_begin ()`
- `node_const_iterator node_end () const`
- `node_iterator node_end ()`
- `mapped_reference operator[] (key_const_reference r_key)`
- `reverse_iterator rbegin ()`
- `const_reverse_iterator rbegin () const`
- `reverse_iterator rend ()`
- `const_reverse_iterator rend () const`
- `size_type size () const`
- `void split (key_const_reference, rb_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)`
- `void swap (rb_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)`
- `void swap (bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)`
- `point_iterator upper_bound (key_const_reference)`
- `point_const_iterator upper_bound (key_const_reference) const`

Public Attributes

- `no_throw_indicator m_no_throw_copies_indicator`
- `store_extra m_store_extra_indicator`

Protected Types

- `typedef node_allocator::value_type node`
- `typedef _Alloc::template rebind< typename traits_type::node >::other node_allocator`
- `typedef traits_type::null_node_update_pointer null_node_update_pointer`
- `typedef types_traits< Key, Mapped, _Alloc, false > traits_base`

Protected Member Functions

- void **actual_erase_node** (node_pointer)
- void **apply_update** (node_pointer, null_node_update_pointer)
- template<typename Node_Update_>
void **apply_update** (node_pointer, Node_Update_*)
- [std::pair](#)< node_pointer, bool > **erase** (node_pointer)
- node_pointer **get_new_node_for_leaf_insert** (const_reference, false_type)
- node_pointer **get_new_node_for_leaf_insert** (const_reference, true_type)
- void **initialize_min_max** ()
- iterator **insert_imp_empty** (const_reference)
- [std::pair](#)< point_iterator, bool > **insert_leaf** (const_reference)
- iterator **insert_leaf_new** (const_reference, node_pointer, bool)
- void **join_finish** (bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- bool **join_prep** (bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- size_type **recursive_count** (node_pointer) const
- void **rotate_left** (node_pointer)
- void **rotate_parent** (node_pointer)
- void **rotate_right** (node_pointer)
- void **split_finish** (bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- bool **split_prep** (key_const_reference, bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- void **update_min_max_for_erased_node** (node_pointer)
- void **update_to_top** (node_pointer, null_node_update_pointer)
- template<typename Node_Update_>
void **update_to_top** (node_pointer, Node_Update_*)
- void **value_swap** (bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)

Static Protected Member Functions

- static void **clear_imp** (node_pointer)

Protected Attributes

- node_pointer **m_p_head**
- size_type **m_size**

Static Protected Attributes

- static node_allocator **s_node_allocator**

4.289.1 Detailed Description

```
template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _Alloc>  
class __gnu_pbds::detail::rb_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >
```

Red-Black tree.

This implementation uses an idea from the SGI STL (using a *header* node which is needed for efficient iteration).

Definition at line 84 of file `rb_tree_.hpp`.

4.289.2 Member Function Documentation

4.289.2.1 `template<typename Key , typename Mapped , typename Cmp_Fn , typename Node_And_It_Traits , typename _Alloc > bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_const_iterator
__gnu_pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_begin () const
[inline], [inherited]`

Returns a const node_iterator corresponding to the node at the root of the tree.

Definition at line 109 of file bin_search_tree_.hpp.

4.289.2.2 `template<typename Key , typename Mapped , typename Cmp_Fn , typename Node_And_It_Traits , typename _Alloc > bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_iterator
__gnu_pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_begin ()
[inline], [inherited]`

Returns a node_iterator corresponding to the node at the root of the tree.

Definition at line 117 of file bin_search_tree_.hpp.

4.289.2.3 `template<typename Key , typename Mapped , typename Cmp_Fn , typename Node_And_It_Traits , typename _Alloc > bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_const_iterator
__gnu_pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_end () const
[inline], [inherited]`

Returns a const node_iterator corresponding to a node just after a leaf of the tree.

Definition at line 125 of file bin_search_tree_.hpp.

4.289.2.4 `template<typename Key , typename Mapped , typename Cmp_Fn , typename Node_And_It_Traits , typename _Alloc > bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_iterator
__gnu_pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_end ()
[inline], [inherited]`

Returns a node_iterator corresponding to a node just after a leaf of the tree.

Definition at line 133 of file bin_search_tree_.hpp.

The documentation for this class was generated from the following file:

- [rb_tree_.hpp](#)

4.290 __gnu_pbds::detail::rb_tree_node_< Value_Type, Metadata, _Alloc > Struct Template Reference

Public Types

- typedef _Alloc::template rebind< metadata_type >::other::const_reference **metadata_const_reference**
- typedef _Alloc::template rebind< metadata_type >::other::reference **metadata_reference**
- typedef Metadata **metadata_type**
- typedef _Alloc::template rebind< [rb_tree_node_< Value_Type, Metadata, _Alloc >](#) >::other::pointer **node_←
pointer**
- typedef Value_Type **value_type**

Public Member Functions

- `metadata_const_reference` **get_metadata** () const
- `metadata_reference` **get_metadata** ()
- `bool` **special** () const

Public Attributes

- `metadata_type` **m_metadata**
- `node_pointer` **m_p_left**
- `node_pointer` **m_p_parent**
- `node_pointer` **m_p_right**
- `bool` **m_red**
- `value_type` **m_value**

4.290.1 Detailed Description

```
template<typename Value_Type, class Metadata, typename _Alloc>
struct __gnu_pbds::detail::rb_tree_node_< Value_Type, Metadata, _Alloc >
```

Node for Red-Black trees.

Definition at line 52 of file `rb_tree_map_/node.hpp`.

The documentation for this struct was generated from the following file:

- [rb_tree_map_/node.hpp](#)

4.291 `__gnu_pbds::detail::rc< _Node, _Alloc >` Class Template Reference

Public Types

- `typedef entry_const_pointer` **const_iterator**
- `typedef node_pointer` **entry**

Public Member Functions

- `rc` (const [rc](#) &)
- `const const_iterator` **begin** () const
- `void` **clear** ()
- `bool` **empty** () const
- `const const_iterator` **end** () const
- `void` **pop** ()
- `void` **push** (entry)
- `size_type` **size** () const
- `void` **swap** ([rc](#) &)
- `node_pointer` **top** () const

4.291.1 Detailed Description

```
template<typename _Node, typename _Alloc>
class __gnu_pbds::detail::rc< _Node, _Alloc >
```

Redundant binary counter.

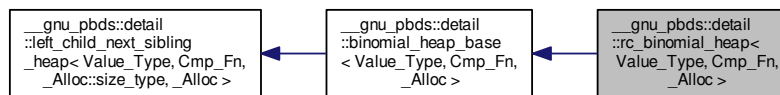
Definition at line 50 of file rc.hpp.

The documentation for this class was generated from the following file:

- [rc.hpp](#)

4.292 __gnu_pbds::detail::rc_binomial_heap< Value_Type, Cmp_Fn, _Alloc > Class Template Reference

Inheritance diagram for __gnu_pbds::detail::rc_binomial_heap< Value_Type, Cmp_Fn, _Alloc >:



Public Types

- typedef base_type::allocator_type **allocator_type**
- typedef base_type::cmp_fn **cmp_fn**
- typedef [base_type::const_iterator](#) **const_iterator**
- typedef base_type::const_pointer **const_pointer**
- typedef base_type::const_reference **const_reference**
- typedef _Alloc::difference_type **difference_type**
- typedef [base_type::iterator](#) **iterator**
- typedef [base_type::point_const_iterator](#) **point_const_iterator**
- typedef [base_type::point_iterator](#) **point_iterator**
- typedef base_type::pointer **pointer**
- typedef base_type::reference **reference**
- typedef _Alloc::size_type **size_type**
- typedef Value_Type **value_type**

Public Member Functions

- `rc_binomial_heap` (const Cmp_Fn &)
- `rc_binomial_heap` (const `rc_binomial_heap`< Value_Type, Cmp_Fn, _Alloc > &)
- `iterator begin` ()
- `const_iterator begin` () const
- void `clear` ()
- bool `empty` () const
- `iterator end` ()
- `const_iterator end` () const
- void `erase` (`point_iterator`)
- template<typename Pred >
size_type `erase_if` (Pred)
- Cmp_Fn & `get_cmp_fn` ()
- const Cmp_Fn & `get_cmp_fn` () const
- void `join` (`rc_binomial_heap`< Value_Type, Cmp_Fn, _Alloc > &)
- void `join` (`binomial_heap_base`< Value_Type, Cmp_Fn, _Alloc > &)
- size_type `max_size` () const
- void `modify` (`point_iterator`, const_reference)
- void `pop` ()
- `point_iterator push` (const_reference)
- size_type `size` () const
- template<typename Pred >
void `split` (Pred, `rc_binomial_heap`< Value_Type, Cmp_Fn, _Alloc > &)
- template<typename Pred >
void `split` (Pred, `binomial_heap_base`< Value_Type, Cmp_Fn, _Alloc > &)
- void `swap` (`rc_binomial_heap`< Value_Type, Cmp_Fn, _Alloc > &)
- void `swap` (`left_child_next_sibling_heap`< Value_Type, Cmp_Fn, _Alloc::size_type, _Alloc > &)
- const_reference `top` () const

Protected Types

- typedef base_type::node `node`
- typedef _Alloc::template rebind< `left_child_next_sibling_heap_node`< Value_Type, _Alloc::size_type, _Alloc >::other `node_allocator`
- typedef _Alloc::size_type `node_metadata`
- typedef `std::pair`< node_pointer, node_pointer > `node_pointer_pair`

Protected Member Functions

- void `actual_erase_node` (node_pointer)
- void `bubble_to_top` (node_pointer)
- void `clear_imp` (node_pointer)
- template<typename It >
void `copy_from_range` (It, It)
- void `find_max` ()
- node_pointer `get_new_node_for_insert` (const_reference)
- node_pointer `prune` (Pred)
- void `swap` (`binomial_heap_base`< Value_Type, Cmp_Fn, _Alloc > &)
- void `swap_with_parent` (node_pointer, node_pointer)
- void `to_linked_list` ()
- void `value_swap` (`left_child_next_sibling_heap` &)

Static Protected Member Functions

- static void **make_child_of** (node_pointer, node_pointer)
- static node_pointer **parent** (node_pointer)

Protected Attributes

- node_pointer **m_p_max**
- node_pointer **m_p_root**
- size_type **m_size**

4.292.1 Detailed Description

```
template<typename Value_Type, typename Cmp_Fn, typename _Alloc>
class __gnu_pbds::detail::rc_binomial_heap< Value_Type, Cmp_Fn, _Alloc >
```

Redundant-counter binomial heap.

Definition at line 66 of file rc_binomial_heap_.hpp.

The documentation for this class was generated from the following file:

- [rc_binomial_heap_.hpp](#)

4.293 __gnu_pbds::detail::resize_policy< _Tp > Class Template Reference

Public Types

- typedef _Tp **size_type**

Public Member Functions

- **resize_policy** (const [resize_policy](#) &other)
- size_type **get_new_size_for_arbitrary** (size_type) const
- size_type **get_new_size_for_grow** () const
- size_type **get_new_size_for_shrink** () const
- bool **grow_needed** (size_type) const
- void **notify_arbitrary** (size_type)
- void **notify_grow_resize** ()
- void **notify_shrink_resize** ()
- bool **resize_needed_for_grow** (size_type) const
- bool **resize_needed_for_shrink** (size_type) const
- bool **shrink_needed** (size_type) const
- void **swap** ([resize_policy](#)< _Tp > &)

Static Public Attributes

- static const `_Tp` **min_size**

4.293.1 Detailed Description

```
template<typename _Tp>
class __gnu_pbds::detail::resize_policy< _Tp >
```

Resize policy for binary heap.

Definition at line 52 of file `resize_policy.hpp`.

The documentation for this class was generated from the following file:

- [resize_policy.hpp](#)

4.294 `__gnu_pbds::detail::splay_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >` Class Template Reference

Inherits `__gnu_pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >`.

Public Types

- typedef `_Alloc` **allocator_type**
- typedef `Cmp_Fn` **cmp_fn**
- typedef `std::pair< size_type, size_type >` **comp_hash**
- typedef `base_type::const_iterator` **const_iterator**
- typedef `base_type::const_pointer` **const_pointer**
- typedef `base_type::const_reference` **const_reference**
- typedef `base_type::const_reverse_iterator` **const_reverse_iterator**
- typedef `splay_tree_tag` **container_category**
- typedef `_Alloc::difference_type` **difference_type**
- typedef `base_type::iterator` **iterator**
- typedef `base_type::key_const_pointer` **key_const_pointer**
- typedef `base_type::key_const_reference` **key_const_reference**
- typedef `base_type::key_pointer` **key_pointer**
- typedef `base_type::key_reference` **key_reference**
- typedef `base_type::key_type` **key_type**
- typedef `base_type::mapped_const_pointer` **mapped_const_pointer**
- typedef `base_type::mapped_const_reference` **mapped_const_reference**
- typedef `base_type::mapped_pointer` **mapped_pointer**
- typedef `base_type::mapped_reference` **mapped_reference**
- typedef `base_type::mapped_type` **mapped_type**
- typedef `__nothrowcopy::indicator` **no_throw_indicator**
- typedef `traits_type::node_const_iterator` **node_const_iterator**
- typedef `traits_type::node_iterator` **node_iterator**

- typedef base_type::node_update **node_update**
- typedef base_type::const_iterator **point_const_iterator**
- typedef base_type::point_iterator **point_iterator**
- typedef base_type::pointer **pointer**
- typedef base_type::reference **reference**
- typedef base_type::reverse_iterator **reverse_iterator**
- typedef _Alloc::size_type **size_type**
- typedef integral_constant< int, Store_Hash > **store_extra**
- typedef base_type::value_type **value_type**

Public Member Functions

- **splay_tree_map** (const Cmp_Fn &)
- **splay_tree_map** (const Cmp_Fn &, const node_update &)
- **splay_tree_map** (const [splay_tree_map](#)< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- iterator **begin** ()
- const_iterator **begin** () const
- void **clear** ()
- template<typename It >
void **copy_from_range** (It, It)
- bool **empty** () const
- iterator **end** ()
- const_iterator **end** () const
- bool **erase** (key_const_reference)
- iterator **erase** (iterator it)
- reverse_iterator **erase** (reverse_iterator)
- template<typename Pred >
size_type **erase_if** (Pred)
- point_iterator **find** (key_const_reference)
- point_const_iterator **find** (key_const_reference) const
- Cmp_Fn & **get_cmp_fn** ()
- const Cmp_Fn & **get_cmp_fn** () const
- void **initialize** ()
- [std::pair](#)< point_iterator, bool > **insert** (const_reference r_value)
- void **join** ([splay_tree_map](#)< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- point_iterator **lower_bound** (key_const_reference)
- point_const_iterator **lower_bound** (key_const_reference) const
- size_type **max_size** () const
- node_const_iterator **node_begin** () const
- node_iterator **node_begin** ()
- node_const_iterator **node_end** () const
- node_iterator **node_end** ()
- mapped_reference **operator[]** (key_const_reference r_key)
- reverse_iterator **rbegin** ()
- const_reverse_iterator **rbegin** () const
- reverse_iterator **rend** ()
- const_reverse_iterator **rend** () const
- size_type **size** () const
- void **split** (key_const_reference, [splay_tree_map](#)< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- void **swap** ([splay_tree_map](#)< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- void **swap** (bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- point_iterator **upper_bound** (key_const_reference)
- point_const_iterator **upper_bound** (key_const_reference) const

Public Attributes

- no_throw_indicator **m_no_throw_copies_indicator**
- store_extra **m_store_extra_indicator**

Protected Types

- typedef node_allocator::value_type **node**
- typedef _Alloc::template rebind< typename traits_type::node >::other **node_allocator**
- typedef traits_type::null_node_update_pointer **null_node_update_pointer**
- typedef [types_traits](#)< Key, Mapped, _Alloc, false > **traits_base**

Protected Member Functions

- void **actual_erase_node** (node_pointer)
- void **apply_update** (node_pointer, null_node_update_pointer)
- template<typename Node_Update_>
void **apply_update** (node_pointer, Node_Update_*)
- [std::pair](#)< node_pointer, bool > **erase** (node_pointer)
- node_pointer **get_new_node_for_leaf_insert** (const_reference, false_type)
- node_pointer **get_new_node_for_leaf_insert** (const_reference, true_type)
- void **initialize_min_max** ()
- iterator **insert_imp_empty** (const_reference)
- [std::pair](#)< point_iterator, bool > **insert_leaf** (const_reference)
- iterator **insert_leaf_new** (const_reference, node_pointer, bool)
- void **join_finish** (bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- bool **join_prep** (bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- size_type **recursive_count** (node_pointer) const
- void **rotate_left** (node_pointer)
- void **rotate_parent** (node_pointer)
- void **rotate_right** (node_pointer)
- void **split_finish** (bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- bool **split_prep** (key_const_reference, bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- void **update_min_max_for_erased_node** (node_pointer)
- void **update_to_top** (node_pointer, null_node_update_pointer)
- template<typename Node_Update_>
void **update_to_top** (node_pointer, Node_Update_*)
- void **value_swap** (bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)

Static Protected Member Functions

- static void **clear_imp** (node_pointer)

Protected Attributes

- node_pointer **m_p_head**
- size_type **m_size**

Static Protected Attributes

- static node_allocator **s_node_allocator**

4.294.1 Detailed Description

```
template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _Alloc>
class __gnu_pbds::detail::splay_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >
```

Splay tree.

Definition at line 107 of file splay_tree_.hpp.

4.294.2 Member Function Documentation

```
4.294.2.1 template<typename Key , typename Mapped , typename Cmp_Fn , typename Node_And_It_Traits , typename
    _Alloc > bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_const_iterator
    __gnu_pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_begin ( ) const
    [inline], [inherited]
```

Returns a const node_iterator corresponding to the node at the root of the tree.

Definition at line 109 of file bin_search_tree_.hpp.

```
4.294.2.2 template<typename Key , typename Mapped , typename Cmp_Fn , typename Node_And_It_Traits , typename
    _Alloc > bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_iterator
    __gnu_pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_begin ( )
    [inline], [inherited]
```

Returns a node_iterator corresponding to the node at the root of the tree.

Definition at line 117 of file bin_search_tree_.hpp.

```
4.294.2.3 template<typename Key , typename Mapped , typename Cmp_Fn , typename Node_And_It_Traits , typename
    _Alloc > bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_const_iterator
    __gnu_pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_end ( ) const
    [inline], [inherited]
```

Returns a const node_iterator corresponding to a node just after a leaf of the tree.

Definition at line 125 of file bin_search_tree_.hpp.

4.294.2.4 `template<typename Key , typename Mapped , typename Cmp_Fn , typename Node_And_It_Traits , typename _Alloc > bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_iterator
__gnu_pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_end ()
[inline], [inherited]`

Returns a `node_iterator` corresponding to a node just after a leaf of the tree.

Definition at line 133 of file `bin_search_tree_.hpp`.

The documentation for this class was generated from the following file:

- [splay_tree_.hpp](#)

4.295 `__gnu_pbds::detail::splay_tree_node_ < Value_Type, Metadata, _Alloc >` Struct Template Reference

Public Types

- `typedef _Alloc::template rebind< metadata_type >::other::const_reference metadata_const_reference`
- `typedef _Alloc::template rebind< metadata_type >::other::reference metadata_reference`
- `typedef Metadata metadata_type`
- `typedef _Alloc::template rebind< splay_tree_node_ < Value_Type, Metadata, _Alloc > >::other::pointer node_pointer`
- `typedef Value_Type value_type`

Public Member Functions

- `metadata_const_reference get_metadata () const`
- `metadata_reference get_metadata ()`
- `bool special () const`

Public Attributes

- `metadata_type m_metadata`
- `node_pointer m_p_left`
- `node_pointer m_p_parent`
- `node_pointer m_p_right`
- `bool m_special`
- `value_type m_value`

4.295.1 Detailed Description

```
template<typename Value_Type, class Metadata, typename _Alloc>
struct __gnu_pbds::detail::splay_tree_node_ < Value_Type, Metadata, _Alloc >
```

Node for splay tree.

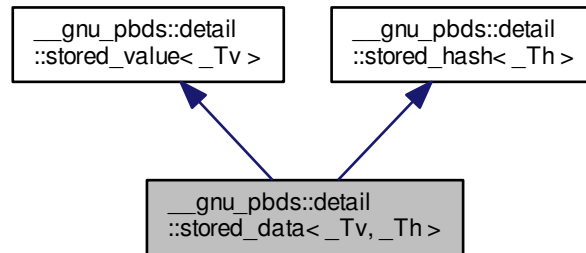
Definition at line 50 of file `splay_tree_/node.hpp`.

The documentation for this struct was generated from the following file:

- [splay_tree_/node.hpp](#)

4.296 `__gnu_pbds::detail::stored_data<_Tv,_Th>` Struct Template Reference

Inheritance diagram for `__gnu_pbds::detail::stored_data<_Tv,_Th>`:



Public Types

- typedef `_Th` **hash_type**
- typedef `_Tv` **value_type**

Public Attributes

- hash_type **m_hash**
- value_type **m_value**

4.296.1 Detailed Description

```
template<typename _Tv, typename _Th>
struct __gnu_pbds::detail::stored_data<_Tv,_Th>
```

Primary template for representation of stored data. Two types of data can be stored: value and hash.

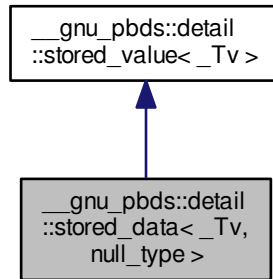
Definition at line 95 of file `types_traits.hpp`.

The documentation for this struct was generated from the following file:

- [types_traits.hpp](#)

4.297 `__gnu_pbds::detail::stored_data<_Tv, null_type>` Struct Template Reference

Inheritance diagram for `__gnu_pbds::detail::stored_data<_Tv, null_type>`:



Public Types

- typedef `_Tv` **value_type**

Public Attributes

- value_type **m_value**

4.297.1 Detailed Description

```
template<typename _Tv>
struct __gnu_pbds::detail::stored_data<_Tv, null_type>
```

Specialization for representation of stored data of just value type.

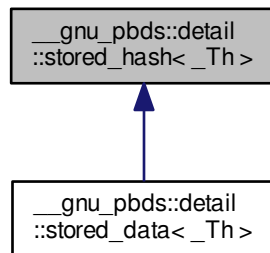
Definition at line 101 of file `types_traits.hpp`.

The documentation for this struct was generated from the following file:

- [types_traits.hpp](#)

4.298 `__gnu_pbds::detail::stored_hash<_Th>` Struct Template Reference

Inheritance diagram for `__gnu_pbds::detail::stored_hash<_Th>`:



Public Types

- typedef `_Th` **hash_type**

Public Attributes

- hash_type **m_hash**

4.298.1 Detailed Description

```
template<typename _Th>
struct __gnu_pbds::detail::stored_hash<_Th>
```

Stored hash.

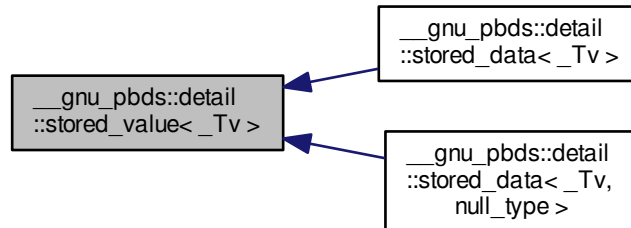
Definition at line 86 of file `types_traits.hpp`.

The documentation for this struct was generated from the following file:

- [types_traits.hpp](#)

4.299 `__gnu_pbds::detail::stored_value<_Tv>` Struct Template Reference

Inheritance diagram for `__gnu_pbds::detail::stored_value<_Tv>`:



Public Types

- typedef `_Tv` **value_type**

Public Attributes

- value_type **m_value**

4.299.1 Detailed Description

```
template<typename _Tv>
struct __gnu_pbds::detail::stored_value<_Tv>
```

Stored value.

Definition at line 78 of file `types_traits.hpp`.

The documentation for this struct was generated from the following file:

- [types_traits.hpp](#)

4.300 `__gnu_pbds::detail::synth_access_traits<Type_Traits, Set, _ATraits>` Struct Template Reference

Inherits `_ATraits`.

Public Types

- typedef `_ATraits` **base_type**
- typedef `base_type::const_iterator` **const_iterator**
- typedef `type_traits::const_reference` **const_reference**
- typedef `type_traits::key_const_reference` **key_const_reference**
- typedef `Type_Traits` **type_traits**

Public Member Functions

- **synth_access_traits** (`const base_type &`)
- `bool` **cmp_keys** (`key_const_reference, key_const_reference`) `const`
- `bool` **cmp_prefixes** (`const_iterator, const_iterator, const_iterator, const_iterator, bool compare_after=false`) `const`
- `bool` **equal_keys** (`key_const_reference, key_const_reference`) `const`
- `bool` **equal_prefixes** (`const_iterator, const_iterator, const_iterator, const_iterator, bool compare_after=true`) `const`

Static Public Member Functions

- `static key_const_reference` **extract_key** (`const_reference`)

4.300.1 Detailed Description

```
template<typename Type_Traits, bool Set, typename _ATraits>
struct __gnu_pbds::detail::synth_access_traits< Type_Traits, Set, _ATraits >
```

Synthetic element access traits.

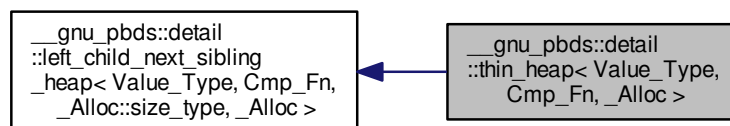
Definition at line 59 of file `synth_access_traits.hpp`.

The documentation for this struct was generated from the following file:

- [synth_access_traits.hpp](#)

4.301 `__gnu_pbds::detail::thin_heap< Value_Type, Cmp_Fn, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::thin_heap< Value_Type, Cmp_Fn, _Alloc >`:



Public Types

- typedef `_Alloc` **allocator_type**
- typedef `Cmp_Fn` **cmp_fn**
- typedef `base_type::const_iterator` **const_iterator**
- typedef `__rebind_a::const_pointer` **const_pointer**
- typedef `__rebind_a::const_reference` **const_reference**
- typedef `_Alloc::difference_type` **difference_type**
- typedef `base_type::iterator` **iterator**
- typedef `base_type::point_const_iterator` **point_const_iterator**
- typedef `base_type::point_iterator` **point_iterator**
- typedef `__rebind_a::pointer` **pointer**
- typedef `__rebind_a::reference` **reference**
- typedef `_Alloc::size_type` **size_type**
- typedef `Value_Type` **value_type**

Public Member Functions

- **iterator** **begin** ()
- **const_iterator** **begin** () const
- void **clear** ()
- bool **empty** () const
- **iterator** **end** ()
- **const_iterator** **end** () const
- void **erase** (**point_iterator**)
- template<typename Pred >
size_type **erase_if** (Pred)
- `Cmp_Fn` & **get_cmp_fn** ()
- const `Cmp_Fn` & **get_cmp_fn** () const
- void **join** (**thin_heap**< `Value_Type`, `Cmp_Fn`, `_Alloc` > &)
- size_type **max_size** () const
- void **modify** (**point_iterator**, const_reference)
- void **pop** ()
- **point_iterator** **push** (const_reference)
- size_type **size** () const
- template<typename Pred >
void **split** (Pred, **thin_heap**< `Value_Type`, `Cmp_Fn`, `_Alloc` > &)
- void **swap** (**left_child_next_sibling_heap**< `Value_Type`, `Cmp_Fn`, `_Alloc::size_type`, `_Alloc` > &)
- const_reference **top** () const

Protected Types

- typedef `base_type::node` **node**
- typedef `_Alloc::template rebind< left_child_next_sibling_heap_node_< Value_Type, _Alloc::size_type, _Alloc >::other` **node_allocator**
- typedef `base_type::node_const_pointer` **node_const_pointer**
- typedef `_Alloc::size_type` **node_metadata**
- typedef `base_type::node_pointer` **node_pointer**
- typedef `std::pair< node_pointer, node_pointer >` **node_pointer_pair**

Protected Member Functions

- **thin_heap** (const Cmp_Fn &)
- **thin_heap** (const [thin_heap](#)< Value_Type, Cmp_Fn, _Alloc > &)
- void **actual_erase_node** (node_pointer)
- void **bubble_to_top** (node_pointer)
- void **clear_imp** (node_pointer)
- template<typename It >
void **copy_from_range** (It, It)
- node_pointer **get_new_node_for_insert** (const_reference)
- node_pointer **prune** (Pred)
- void **swap** ([thin_heap](#)< Value_Type, Cmp_Fn, _Alloc > &)
- void **swap_with_parent** (node_pointer, node_pointer)
- void **to_linked_list** ()
- void **value_swap** ([left_child_next_sibling_heap](#) &)

Static Protected Member Functions

- static node_pointer **parent** (node_pointer)

Protected Attributes

- node_pointer **m_p_root**
- size_type **m_size**

4.301.1 Detailed Description

```
template<typename Value_Type, typename Cmp_Fn, typename _Alloc>
class __gnu_pbds::detail::thin_heap< Value_Type, Cmp_Fn, _Alloc >
```

Thin heap.

See Tarjan and Kaplan.

Definition at line 77 of file `thin_heap_.hpp`.

The documentation for this class was generated from the following file:

- [thin_heap_.hpp](#)

4.302 `__gnu_pbds::detail::tree_metadata_helper< Node_Update, _BTp >` Struct Template Reference

4.302.1 Detailed Description

```
template<typename Node_Update, bool _BTp>
struct __gnu_pbds::detail::tree_metadata_helper< Node_Update, _BTp >
```

Tree metadata helper.

Definition at line 58 of file `tree_policy/node_metadata_selector.hpp`.

The documentation for this struct was generated from the following file:

- [tree_policy/node_metadata_selector.hpp](#)

4.303 `__gnu_pbds::detail::tree_metadata_helper< Node_Update, false >` Struct Template Reference

Public Types

- typedef `Node_Update::metadata_type` **type**

4.303.1 Detailed Description

```
template<typename Node_Update>
struct __gnu_pbds::detail::tree_metadata_helper< Node_Update, false >
```

Specialization, false.

Definition at line 62 of file `tree_policy/node_metadata_selector.hpp`.

The documentation for this struct was generated from the following file:

- [tree_policy/node_metadata_selector.hpp](#)

4.304 `__gnu_pbds::detail::tree_metadata_helper< Node_Update, true >` Struct Template Reference

Public Types

- typedef `null_type` **type**

4.304.1 Detailed Description

```
template<typename Node_Update>
struct __gnu_pbds::detail::tree_metadata_helper< Node_Update, true >
```

Specialization, true.

Definition at line 69 of file tree_policy/node_metadata_selector.hpp.

The documentation for this struct was generated from the following file:

- [tree_policy/node_metadata_selector.hpp](#)

4.305 __gnu_pbds::detail::tree_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc > Struct Template Reference

Public Types

- typedef [tree_metadata_helper](#)< __node_u, null_update >::type **type**

4.305.1 Detailed Description

```
template<typename Key, typename Data, typename Cmp_Fn, template< typename Node_Cltr, typename Const_Iterator, typename
Cmp_Fn_, typename _Alloc > class Node_Update, typename _Alloc>
struct __gnu_pbds::detail::tree_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc >
```

Tree node metadata dispatch.

Definition at line 84 of file tree_policy/node_metadata_selector.hpp.

The documentation for this struct was generated from the following file:

- [tree_policy/node_metadata_selector.hpp](#)

4.306 __gnu_pbds::detail::tree_traits< Key, Data, Cmp_Fn, Node_Update, Tag, _Alloc > Struct Template Reference

4.306.1 Detailed Description

```
template<typename Key, typename Data, typename Cmp_Fn, template< typename Node_Cltr, typename Node_Itr, typename Cmp_Fn_,
typename _Alloc > class Node_Update, typename Tag, typename _Alloc>
struct __gnu_pbds::detail::tree_traits< Key, Data, Cmp_Fn, Node_Update, Tag, _Alloc >
```

Tree traits class, primary template.

Definition at line 70 of file branch_policy/traits.hpp.

The documentation for this struct was generated from the following file:

- [branch_policy/traits.hpp](#)

4.307 `__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc > Struct` Template Reference

Public Types

- typedef `tree_node_metadata_dispatch< Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type` **metadata_type**
- typedef `ov_tree_node_const_it_< value_type, metadata_type, _Alloc >` `node_const_iterator`
- typedef `ov_tree_node_it_< value_type, metadata_type, _Alloc >` **node_iterator**
- typedef `Node_Update< node_const_iterator, node_iterator, Cmp_Fn, _Alloc >` **node_update**
- typedef `__gnu_pbds::null_node_update< node_const_iterator, node_iterator, Cmp_Fn, _Alloc > * null_node` `←_update_pointer`

4.307.1 Detailed Description

```
template<typename Key, typename Mapped, class Cmp_Fn, template< typename Node_Cltr, class Node_Itr, class Cmp_Fn_, typename
_Alloc_ > class Node_Update, typename _Alloc>
struct __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >
```

Tree traits.

Definition at line 61 of file `ov_tree_map_/traits.hpp`.

4.307.2 Member Typedef Documentation

4.307.2.1 `template<typename Key , typename Mapped , class Cmp_Fn , template< typename Node_Cltr, class Node_Itr, class Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc > typedef ov_tree_node_const_it_< value_type, metadata_type, _Alloc> __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >::node_const_iterator`

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

Definition at line 95 of file `ov_tree_map_/traits.hpp`.

The documentation for this struct was generated from the following file:

- [ov_tree_map_/traits.hpp](#)

4.308 `__gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc > Struct` Template Reference

Public Types

- typedef `tree_node_metadata_dispatch< Key, null_type, Cmp_Fn, Node_Update, _Alloc >::type` **metadata_type**
- typedef `ov_tree_node_const_it_< value_type, metadata_type, _Alloc >` `node_const_iterator`
- typedef `node_const_iterator` **node_iterator**
- typedef `Node_Update< node_const_iterator, node_const_iterator, Cmp_Fn, _Alloc >` **node_update**
- typedef `__gnu_pbds::null_node_update< node_const_iterator, node_iterator, Cmp_Fn, _Alloc > * null_node` `←_update_pointer`

4.308.1 Detailed Description

```
template<typename Key, class Cmp_Fn, template< typename Node_Cltr, class Node_Itr, class Cmp_Fn_, typename _Alloc_ > class
Node_Update, typename _Alloc>
struct __gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >
```

Specialization.

Definition at line 132 of file `ov_tree_map_/traits.hpp`.

4.308.2 Member Typedef Documentation

```
4.308.2.1 template<typename Key , class Cmp_Fn , template< typename Node_Cltr, class Node_Itr, class Cmp_Fn_, typename
_Alloc_ > class Node_Update, typename _Alloc > typedef ov_tree_node_const_it_< value_type, metadata_type,
_Alloc> __gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc
>::node_const_iterator
```

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

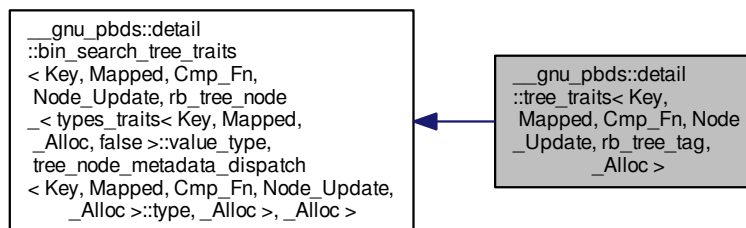
Definition at line 166 of file `ov_tree_map_/traits.hpp`.

The documentation for this struct was generated from the following file:

- [ov_tree_map_/traits.hpp](#)

4.309 __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc > Struct Template Reference

Inheritance diagram for `__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >`:



Public Types

- typedef `bin_search_tree_const_it_< typename _Alloc::template rebind< node >::other::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, false, _Alloc >` **const_reverse_iterator**
- typedef `rb_tree_node_< types_traits< Key, Mapped, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >` **node**
- typedef `bin_search_tree_const_node_it_< rb_tree_node_< types_traits< Key, Mapped, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >, point_const_iterator, point_iterator, _Alloc >` **node_const_iterator**
- typedef `bin_search_tree_node_it_< rb_tree_node_< types_traits< Key, Mapped, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >, point_const_iterator, point_iterator, _Alloc >` **node_iterator**
- typedef `Node_Update< node_const_iterator, node_iterator, Cmp_Fn, _Alloc >` **node_update**
- typedef `__gnu_pbds::null_node_update< node_const_iterator, node_iterator, Cmp_Fn, _Alloc > * null_node_update_pointer`
- typedef `bin_search_tree_const_it_< typename _Alloc::template rebind< node >::other::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, true, _Alloc >` **point_const_iterator**
- typedef `bin_search_tree_it_< typename _Alloc::template rebind< node >::other::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, true, _Alloc >` **point_iterator**
- typedef `bin_search_tree_it_< typename _Alloc::template rebind< node >::other::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, false, _Alloc >` **reverse_iterator**

4.309.1 Detailed Description

```
template<typename Key, typename Mapped, typename Cmp_Fn, template< typename Node_Cltr, typename Node_Itr, typename Cmp_Fn, typename _Alloc > class Node_Update, typename _Alloc>
```

```
struct __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >
```

Specialization.

Definition at line 61 of file `rb_tree_map_/traits.hpp`.

4.309.2 Member Typedef Documentation

- 4.309.2.1 `typedef bin_search_tree_const_node_it_< rb_tree_node_< types_traits< Key, Mapped, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >, point_const_iterator, point_iterator, _Alloc>` `__gnu_pbds::detail::bin_search_tree_traits< Key, Mapped, Cmp_Fn, Node_Update, rb_tree_node_< types_traits< Key, Mapped, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >, _Alloc >::node_const_iterator` `[inherited]`

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

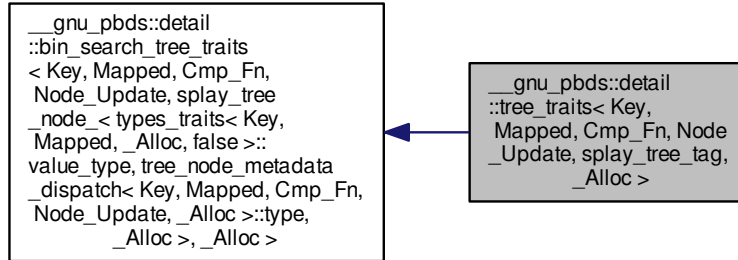
Definition at line 131 of file `bin_search_tree_/traits.hpp`.

The documentation for this struct was generated from the following file:

- [rb_tree_map_/traits.hpp](#)

4.310 `__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >` Struct Template Reference

Inheritance diagram for `__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >`:



Public Types

- typedef `bin_search_tree_const_it` < typename `_Alloc::template rebind< node >::other::pointer`, typename `type_traits::value_type`, typename `type_traits::pointer`, typename `type_traits::const_pointer`, typename `type_traits::reference`, typename `type_traits::const_reference`, false, `_Alloc` > **const_reverse_iterator**
- typedef `splay_tree_node` < `types_traits< Key, Mapped, _Alloc, false >::value_type`, `tree_node_metadata_dispatch< Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type`, `_Alloc` > **node**
- typedef `bin_search_tree_const_node_it` < `splay_tree_node< types_traits< Key, Mapped, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >`, `point_const_iterator`, `point_iterator`, `_Alloc` > **node_const_iterator**
- typedef `bin_search_tree_node_it` < `splay_tree_node< types_traits< Key, Mapped, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >`, `point_const_iterator`, `point_iterator`, `_Alloc` > **node_iterator**
- typedef `Node_Update` < `node_const_iterator`, `node_iterator`, `Cmp_Fn`, `_Alloc` > **node_update**
- typedef `__gnu_pbds::null_node_update` < `node_const_iterator`, `node_iterator`, `Cmp_Fn`, `_Alloc` > * **null_node_update_pointer**
- typedef `bin_search_tree_const_it` < typename `_Alloc::template rebind< node >::other::pointer`, typename `type_traits::value_type`, typename `type_traits::pointer`, typename `type_traits::const_pointer`, typename `type_traits::reference`, typename `type_traits::const_reference`, true, `_Alloc` > **point_const_iterator**
- typedef `bin_search_tree_it` < typename `_Alloc::template rebind< node >::other::pointer`, typename `type_traits::value_type`, typename `type_traits::pointer`, typename `type_traits::const_pointer`, typename `type_traits::reference`, typename `type_traits::const_reference`, true, `_Alloc` > **point_iterator**
- typedef `bin_search_tree_it` < typename `_Alloc::template rebind< node >::other::pointer`, typename `type_traits::value_type`, typename `type_traits::pointer`, typename `type_traits::const_pointer`, typename `type_traits::reference`, typename `type_traits::const_reference`, false, `_Alloc` > **reverse_iterator**

4.310.1 Detailed Description

```
template<typename Key, typename Mapped, typename Cmp_Fn, template< typename Node_Cltr, typename Node_Itr, typename
Cmp_Fn_, typename _Alloc > class Node_Update, typename _Alloc>
struct __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >
```

Specialization.

Definition at line 61 of file `splay_tree_/traits.hpp`.

4.310.2 Member Typedef Documentation

4.310.2.1 `typedef bin_search_tree_const_node_it_< splay_tree_node_< types_traits< Key, Mapped, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >, point_const_iterator, point_iterator, _Alloc> __gnu_pbds::detail::bin_search_tree_traits< Key, Mapped, Cmp_Fn, Node_Update, splay_tree_node_< types_traits< Key, Mapped, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >, _Alloc >::node_const_iterator` [inherited]

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

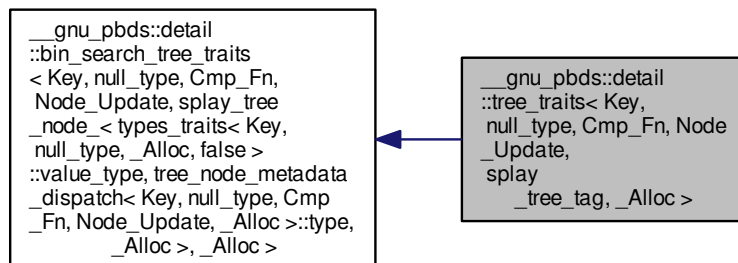
Definition at line 131 of file `bin_search_tree_/traits.hpp`.

The documentation for this struct was generated from the following file:

- [splay_tree_/traits.hpp](#)

4.311 `__gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >` Struct Template Reference

Inheritance diagram for `__gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >`:



Public Types

- typedef [bin_search_tree_const_it](#) < typename [_Alloc](#)::template rebind< [node](#) >::other::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, false, [_Alloc](#) > **const_reverse_iterator**
- typedef [splay_tree_node](#) < [types_traits](#)< Key, [null_type](#), [_Alloc](#), false >::value_type, [tree_node_metadata_dispatch](#) < Key, [null_type](#), [Cmp_Fn](#), [Node_Update](#), [_Alloc](#) >::type, [_Alloc](#) > **node**
- typedef [bin_search_tree_const_node_it](#) < [splay_tree_node](#) < [types_traits](#)< Key, [null_type](#), [_Alloc](#), false >::value_type, [tree_node_metadata_dispatch](#) < Key, [null_type](#), [Cmp_Fn](#), [Node_Update](#), [_Alloc](#) >::type, [_Alloc](#) >, [point_const_iterator](#), [point_iterator](#), [_Alloc](#) > **node_const_iterator**
- typedef [bin_search_tree_node_it](#) < [splay_tree_node](#) < [types_traits](#)< Key, [null_type](#), [_Alloc](#), false >::value_type, [tree_node_metadata_dispatch](#) < Key, [null_type](#), [Cmp_Fn](#), [Node_Update](#), [_Alloc](#) >::type, [_Alloc](#) >, [point_const_iterator](#), [point_iterator](#), [_Alloc](#) > **node_iterator**
- typedef [Node_Update](#) < [node_const_iterator](#), [node_iterator](#), [Cmp_Fn](#), [_Alloc](#) > **node_update**
- typedef [__gnu_pbds::null_node_update](#) < [node_const_iterator](#), [node_iterator](#), [Cmp_Fn](#), [_Alloc](#) > * **null_node_update_pointer**
- typedef [bin_search_tree_const_it](#) < typename [_Alloc](#)::template rebind< [node](#) >::other::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, true, [_Alloc](#) > **point_const_iterator**
- typedef [bin_search_tree_it](#) < typename [_Alloc](#)::template rebind< [node](#) >::other::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, true, [_Alloc](#) > **point_iterator**
- typedef [bin_search_tree_it](#) < typename [_Alloc](#)::template rebind< [node](#) >::other::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, false, [_Alloc](#) > **reverse_iterator**

4.311.1 Detailed Description

```
template<typename Key, class Cmp_Fn, template< typename Node_Cltr, class Node_Itr, class Cmp_Fn_, typename _Alloc > class Node_Update, typename _Alloc>
```

```
struct __gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >
```

Specialization.

Definition at line 81 of file [splay_tree_/traits.hpp](#).

4.311.2 Member Typedef Documentation

- 4.311.2.1 typedef [bin_search_tree_const_node_it](#) < [splay_tree_node](#) < [types_traits](#)< Key, [null_type](#), [_Alloc](#), false >::value_type, [tree_node_metadata_dispatch](#) < Key, [null_type](#), [Cmp_Fn](#), [Node_Update](#), [_Alloc](#) >::type, [_Alloc](#) >, [point_const_iterator](#), [point_iterator](#), [_Alloc](#) > [__gnu_pbds::detail::bin_search_tree_traits](#) < Key, [null_type](#), [Cmp_Fn](#), [Node_Update](#), [splay_tree_node](#) < [types_traits](#)< Key, [null_type](#), [_Alloc](#), false >::value_type, [tree_node_metadata_dispatch](#) < Key, [null_type](#), [Cmp_Fn](#), [Node_Update](#), [_Alloc](#) >::type, [_Alloc](#) >, [_Alloc](#) >::**node_const_iterator** [\[inherited\]](#)

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

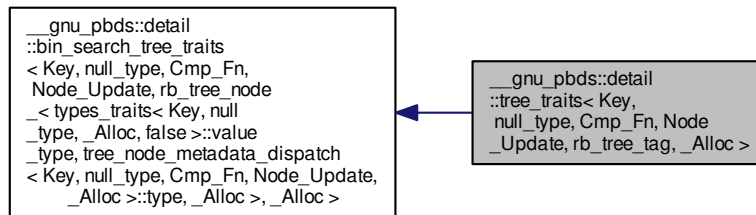
Definition at line 131 of file [bin_search_tree_/traits.hpp](#).

The documentation for this struct was generated from the following file:

- [splay_tree_/traits.hpp](#)

4.312 `__gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >` Struct Template Reference

Inheritance diagram for `__gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >`:



Public Types

- typedef `bin_search_tree_const_it_< typename _Alloc::template rebind< node >::other::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, false, _Alloc >` **const_reverse_iterator**
- typedef `rb_tree_node_< types_traits< Key, null_type, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, null_type, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >` **node**
- typedef `bin_search_tree_const_node_it_< rb_tree_node_< types_traits< Key, null_type, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, null_type, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >, point_const_iterator, point_iterator, _Alloc >` **node_const_iterator**
- typedef `bin_search_tree_node_it_< rb_tree_node_< types_traits< Key, null_type, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, null_type, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >, point_const_iterator, point_iterator, _Alloc >` **node_iterator**
- typedef `Node_Update< node_const_iterator, node_iterator, Cmp_Fn, _Alloc >` **node_update**
- typedef `__gnu_pbds::null_node_update< node_const_iterator, node_iterator, Cmp_Fn, _Alloc > * null_node_update_pointer`
- typedef `bin_search_tree_const_it_< typename _Alloc::template rebind< node >::other::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, true, _Alloc >` **point_const_iterator**
- typedef `bin_search_tree_it_< typename _Alloc::template rebind< node >::other::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, true, _Alloc >` **point_iterator**
- typedef `bin_search_tree_it_< typename _Alloc::template rebind< node >::other::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, false, _Alloc >` **reverse_iterator**

4.312.1 Detailed Description

```
template<typename Key, typename Cmp_Fn, template< typename Node_Cltr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc>
struct __gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >
```

Specialization.

Definition at line 85 of file `rb_tree_map_/traits.hpp`.

4.312.2 Member Typedef Documentation

```
4.312.2.1 typedef bin_search_tree_const_node_it_< rb_tree_node_< types_traits< Key, null_type, _Alloc, false
>::value_type, tree_node_metadata_dispatch< Key, null_type, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc
>, point_const_iterator, point_iterator, _Alloc> __gnu_pbds::detail::bin_search_tree_traits< Key,
null_type, Cmp_Fn, Node_Update, rb_tree_node_< types_traits< Key, null_type, _Alloc, false >::value_type,
tree_node_metadata_dispatch< Key, null_type, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >, _Alloc
>::node_const_iterator [inherited]
```

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

Definition at line 131 of file `bin_search_tree_/traits.hpp`.

The documentation for this struct was generated from the following file:

- [rb_tree_map_/traits.hpp](#)

4.313 __gnu_pbds::detail::trie_metadata_helper< Node_Update, _BTp > Struct Template Reference

4.313.1 Detailed Description

```
template<typename Node_Update, bool _BTp>
struct __gnu_pbds::detail::trie_metadata_helper< Node_Update, _BTp >
```

Trie metadata helper.

Definition at line 58 of file `trie_policy/node_metadata_selector.hpp`.

The documentation for this struct was generated from the following file:

- [trie_policy/node_metadata_selector.hpp](#)

4.314 __gnu_pbds::detail::trie_metadata_helper< Node_Update, false > Struct Template Reference

Public Types

- typedef Node_Update::metadata_type **type**

4.314.1 Detailed Description

```
template<typename Node_Update>
struct __gnu_pbds::detail::trie_metadata_helper< Node_Update, false >
```

Specialization, false.

Definition at line 62 of file `trie_policy/node_metadata_selector.hpp`.

The documentation for this struct was generated from the following file:

- [trie_policy/node_metadata_selector.hpp](#)

4.315 `__gnu_pbds::detail::trie_metadata_helper< Node_Update, true >` Struct Template Reference

Public Types

- typedef `null_type` **type**

4.315.1 Detailed Description

```
template<typename Node_Update>
struct __gnu_pbds::detail::trie_metadata_helper< Node_Update, true >
```

Specialization, true.

Definition at line 69 of file `trie_policy/node_metadata_selector.hpp`.

The documentation for this struct was generated from the following file:

- [trie_policy/node_metadata_selector.hpp](#)

4.316 `__gnu_pbds::detail::trie_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc >` Struct Template Reference

Public Types

- typedef `trie_metadata_helper< __node_u, null_update >::type` **type**

4.316.1 Detailed Description

```
template<typename Key, typename Data, typename Cmp_Fn, template< typename Node_Cltr, typename Const_Iterator, typename
Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc>
struct __gnu_pbds::detail::trie_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc >
```

Trie node metadata dispatch.

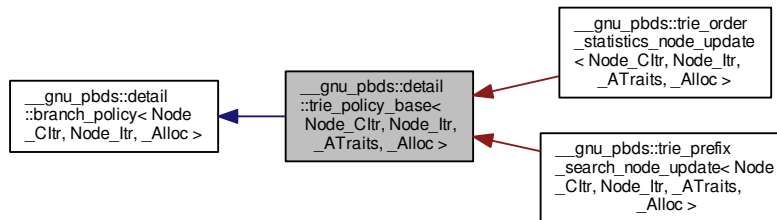
Definition at line 84 of file trie_policy/node_metadata_selector.hpp.

The documentation for this struct was generated from the following file:

- [trie_policy/node_metadata_selector.hpp](#)

4.317 __gnu_pbds::detail::trie_policy_base< Node_Cltr, Node_Itr, _ATraits, _Alloc > Class Template Reference

Inheritance diagram for __gnu_pbds::detail::trie_policy_base< Node_Cltr, Node_Itr, _ATraits, _Alloc >:



Public Types

- typedef `_ATraits` **access_traits**
- typedef `_Alloc` **allocator_type**
- typedef `node_const_iterator::value_type` **const_iterator**
- typedef `node_iterator::value_type` **iterator**
- typedef `base_type::key_const_reference` **key_const_reference**
- typedef `base_type::key_type` **key_type**
- typedef `null_type` **metadata_type**
- typedef `Node_Cltr` **node_const_iterator**
- typedef `Node_Itr` **node_iterator**
- typedef `allocator_type::size_type` **size_type**

Protected Types

- typedef `rebind_v::const_pointer` **const_pointer**
- typedef `rebind_v::const_reference` **const_reference**
- typedef `Node_Itr::value_type` **it_type**
- typedef `remove_const< key_type >::type` **rkkey_type**
- typedef `remove_const< value_type >::type` **rcvalue_type**
- typedef `_Alloc::template rebind< rkkey_type >::other` **rebind_k**
- typedef `_Alloc::template rebind< rcvalue_type >::other` **rebind_v**
- typedef `rebind_v::reference` **reference**
- typedef `std::iterator_traits< it_type >::value_type` **value_type**

Protected Member Functions

- virtual `const_iterator` **end** () const =0
- virtual `iterator` **end** ()=0
- `it_type` **end_iterator** () const
- virtual `const access_traits &` **get_access_traits** () const =0
- virtual `node_const_iterator` **node_begin** () const =0
- virtual `node_iterator` **node_begin** ()=0
- virtual `node_const_iterator` **node_end** () const =0
- virtual `node_iterator` **node_end** ()=0

Static Protected Member Functions

- static `size_type` **common_prefix_len** (`node_iterator`, `e_const_iterator`, `e_const_iterator`, `const access_traits &`)
- static `key_const_reference` **extract_key** (`const_reference r_val`)
- static `iterator` **leftmost_it** (`node_iterator`)
- static `bool` **less** (`e_const_iterator`, `e_const_iterator`, `e_const_iterator`, `e_const_iterator`, `const access_traits &`)
- static `iterator` **rightmost_it** (`node_iterator`)

4.317.1 Detailed Description

```
template<typename Node_Cltr, typename Node_Itr, typename _ATraits, typename _Alloc>
class __gnu_pbds::detail::trie_policy_base< Node_Cltr, Node_Itr, _ATraits, _Alloc >
```

Base class for trie policies.

Definition at line 53 of file `trie_policy_base.hpp`.

The documentation for this class was generated from the following file:

- [trie_policy_base.hpp](#)

4.318 `__gnu_pbds::detail::trie_traits< Key, Data, _ATraits, Node_Update, Tag, _Alloc >` Struct Template Reference

4.318.1 Detailed Description

```
template<typename Key, typename Data, typename _ATraits, template< typename Node_Cltr, typename Node_Itr, typename _A←
Traits_, typename _Alloc > class Node_Update, typename Tag, typename _Alloc>
struct __gnu_pbds::detail::trie_traits< Key, Data, _ATraits, Node_Update, Tag, _Alloc >
```

Trie traits class, primary template.

Definition at line 83 of file `branch_policy/traits.hpp`.

The documentation for this struct was generated from the following file:

- [branch_policy/traits.hpp](#)

4.319 `__gnu_pbds::detail::trie_traits< Key, Mapped, _ATraits, Node_Update, pat_trie_tag, _Alloc >` Struct Template Reference

Public Types

- typedef `_ATraits` **access_traits**
- typedef `base_type::Cltr< node, leaf, head, inode, true >` **const_iterator**
- typedef `base_type::Cltr< node, leaf, head, inode, false >` **const_reverse_iterator**
- typedef `base_type::Head< synth_access_traits, metadata >` **head**
- typedef `base_type::Inode< synth_access_traits, metadata >` **inode**
- typedef `base_type::Itr< node, leaf, head, inode, true >` **iterator**
- typedef `base_type::Leaf< synth_access_traits, metadata >` **leaf**
- typedef `base_type::Metadata< metadata_type, _Alloc >` **metadata**
- typedef `trie_node_metadata_dispatch< Key, Mapped, _ATraits, Node_Update, _Alloc >::type` **metadata_type**
- typedef `base_type::Node_base< synth_access_traits, metadata >` **node**
- typedef `base_type::Node_citer< node, leaf, head, inode, const_iterator, iterator, _Alloc >` **node_const_iterator**
- typedef `base_type::Node_iter< node, leaf, head, inode, const_iterator, iterator, _Alloc >` **node_iterator**
- typedef `Node_Update< node_const_iterator, node_iterator, _ATraits, _Alloc >` **node_update**
- typedef `null_node_update< node_const_iterator, node_iterator, _ATraits, _Alloc > * null_node_update_pointer`
- typedef `base_type::Itr< node, leaf, head, inode, false >` **reverse_iterator**
- typedef `__gnu_pbds::detail::synth_access_traits< type_traits, false, access_traits >` **synth_access_traits**

4.319.1 Detailed Description

```
template<typename Key, typename Mapped, typename _ATraits, template< typename Node_Cltr, typename Node_Itr, typename Cmp←
_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc>
struct __gnu_pbds::detail::trie_traits< Key, Mapped, _ATraits, Node_Update, pat_trie_tag, _Alloc >
```

Specialization.

Definition at line 62 of file `pat_trie_/traits.hpp`.

4.319.2 Member Typedef Documentation

4.319.2.1 `template<typename Key , typename Mapped , typename _ATraits , template< typename Node_Cltr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc > typedef base_type::_Node_citer<node, leaf, head, inode, const_iterator, iterator, _Alloc> __gnu_pbds::detail::trie_traits< Key, Mapped, _ATraits, Node_Update, pat_trie_tag, _Alloc >::node_const_iterator`

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

Definition at line 88 of file `pat_trie_/traits.hpp`.

4.319.2.2 `template<typename Key , typename Mapped , typename _ATraits , template< typename Node_Cltr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc > typedef Node_Update<node_const_iterator, node_iterator, _ATraits, _Alloc> __gnu_pbds::detail::trie_traits< Key, Mapped, _ATraits, Node_Update, pat_trie_tag, _Alloc >::node_update`

Type for node update.

Definition at line 93 of file `pat_trie_/traits.hpp`.

4.319.2.3 `template<typename Key , typename Mapped , typename _ATraits , template< typename Node_Cltr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc > typedef __gnu_pbds::detail::synth_access_traits<type_traits, false, access_traits> __gnu_pbds::detail::trie_traits< Key, Mapped, _ATraits, Node_Update, pat_trie_tag, _Alloc >::synth_access_traits`

Type for synthesized traits.

Definition at line 74 of file `pat_trie_/traits.hpp`.

The documentation for this struct was generated from the following file:

- [pat_trie_/traits.hpp](#)

4.320 `__gnu_pbds::detail::trie_traits< Key, null_type, _ATraits, Node_Update, pat_trie_tag, _Alloc >` Struct Template Reference

Public Types

- typedef `_ATraits` **access_traits**
- typedef `base_type::_Cltr< node, leaf, head, inode, true >` **const_iterator**
- typedef `base_type::_Cltr< node, leaf, head, inode, false >` **const_reverse_iterator**
- typedef `base_type::_Head< synth_access_traits, metadata >` **head**
- typedef `base_type::_Inode< synth_access_traits, metadata >` **inode**
- typedef `const_iterator` **iterator**
- typedef `base_type::_Leaf< synth_access_traits, metadata >` **leaf**
- typedef `base_type::_Metadata< metadata_type, _Alloc >` **metadata**
- typedef `trie_node_metadata_dispatch< Key, null_type, _ATraits, Node_Update, _Alloc >::type` **metadata_type**
- typedef `base_type::_Node_base< synth_access_traits, metadata >` **node**
- typedef `base_type::_Node_citer< node, leaf, head, inode, const_iterator, iterator, _Alloc >` **node_const_iterator**
- typedef `node_const_iterator` **node_iterator**
- typedef `Node_Update< node_const_iterator, node_iterator, _ATraits, _Alloc >` **node_update**
- typedef `null_node_update< node_const_iterator, node_const_iterator, _ATraits, _Alloc > * null_node_update` **←_pointer**
- typedef `const_reverse_iterator` **reverse_iterator**
- typedef `__gnu_pbds::detail::synth_access_traits< type_traits, true, access_traits >` **synth_access_traits**

4.320.1 Detailed Description

```
template<typename Key, typename _ATraits, template< typename Node_Cltr, typename Node_Itr, typename Cmp_Fn_, typename _↵
_Alloc_ > class Node_Update, typename _Alloc>
struct __gnu_pbds::detail::trie_traits< Key, null_type, _ATraits, Node_Update, pat_trie_tag, _Alloc >
```

Specialization.

Definition at line 109 of file pat_trie_/traits.hpp.

4.320.2 Member Typedef Documentation

```
4.320.2.1 template<typename Key , typename _ATraits , template< typename Node_Cltr, typename Node_Itr, typename Cmp_Fn_,
typename _Alloc_ > class Node_Update, typename _Alloc > typedef base_type::_Node_citer<node, leaf,
head, inode, const_iterator, iterator, _Alloc> __gnu_pbds::detail::trie_traits< Key, null_type, _ATraits,
Node_Update, pat_trie_tag, _Alloc >::node_const_iterator
```

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

Definition at line 135 of file pat_trie_/traits.hpp.

```
4.320.2.2 template<typename Key , typename _ATraits , template< typename Node_Cltr, typename Node_Itr, typename Cmp_Fn_,
typename _Alloc_ > class Node_Update, typename _Alloc > typedef Node_Update<node_const_iterator,
node_iterator, _ATraits, _Alloc> __gnu_pbds::detail::trie_traits< Key, null_type, _ATraits, Node_Update,
pat_trie_tag, _Alloc >::node_update
```

Type for node update.

Definition at line 140 of file pat_trie_/traits.hpp.

```
4.320.2.3 template<typename Key , typename _ATraits , template< typename Node_Cltr, typename Node_Itr, typename Cmp_Fn_,
typename _Alloc_ > class Node_Update, typename _Alloc > typedef __gnu_pbds::detail::synth_access↵
_traits<type_traits, true, access_traits> __gnu_pbds::detail::trie_traits< Key, null_type, _ATraits,
Node_Update, pat_trie_tag, _Alloc >::synth_access_traits
```

Type for synthesized traits.

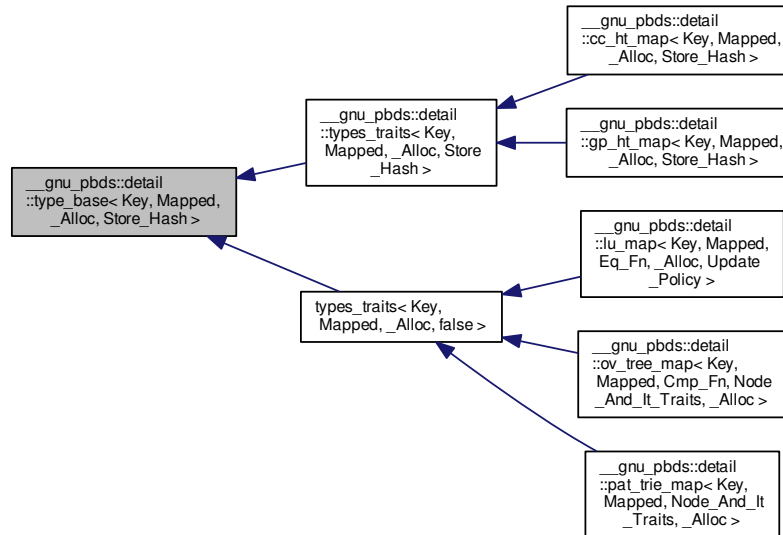
Definition at line 121 of file pat_trie_/traits.hpp.

The documentation for this struct was generated from the following file:

- [pat_trie_/traits.hpp](#)

4.321 `__gnu_pbds::detail::type_base< Key, Mapped, _Alloc, Store_Hash >` Struct Template Reference

Inheritance diagram for `__gnu_pbds::detail::type_base< Key, Mapped, _Alloc, Store_Hash >`:



4.321.1 Detailed Description

```
template<typename Key, typename Mapped, typename _Alloc, bool Store_Hash>
struct __gnu_pbds::detail::type_base< Key, Mapped, _Alloc, Store_Hash >
```

Primary template.

Definition at line 107 of file `types_traits.hpp`.

The documentation for this struct was generated from the following file:

- [types_traits.hpp](#)

4.322 `__gnu_pbds::detail::type_base< Key, Mapped, _Alloc, false >` Struct Template Reference

Public Types

- typedef `__rebind_va::const_pointer` **const_pointer**
- typedef `__rebind_va::const_reference` **const_reference**
- typedef `__rebind_ma::const_pointer` **mapped_const_pointer**
- typedef `__rebind_ma::const_reference` **mapped_const_reference**

- typedef __rebind_ma::pointer **mapped_pointer**
- typedef __rebind_ma::reference **mapped_reference**
- typedef __rebind_ma::value_type **mapped_type**
- typedef __rebind_va::pointer **pointer**
- typedef __rebind_va::reference **reference**
- typedef _Alloc::size_type **size_type**
- typedef [stored_data](#)< value_type, [null_type](#) > **stored_data_type**
- typedef __rebind_va::value_type **value_type**

4.322.1 Detailed Description

```
template<typename Key, typename Mapped, typename _Alloc>
struct __gnu_pbds::detail::type_base< Key, Mapped, _Alloc, false >
```

Specialization of type_base for the case where the hash value is not stored alongside each value.

Definition at line 114 of file types_traits.hpp.

The documentation for this struct was generated from the following file:

- [types_traits.hpp](#)

4.323 __gnu_pbds::detail::type_base< Key, Mapped, _Alloc, true > Struct Template Reference

Public Types

- typedef __rebind_va::const_pointer **const_pointer**
- typedef __rebind_va::const_reference **const_reference**
- typedef __rebind_ma::const_pointer **mapped_const_pointer**
- typedef __rebind_ma::const_reference **mapped_const_reference**
- typedef __rebind_ma::pointer **mapped_pointer**
- typedef __rebind_ma::reference **mapped_reference**
- typedef __rebind_ma::value_type **mapped_type**
- typedef __rebind_va::pointer **pointer**
- typedef __rebind_va::reference **reference**
- typedef _Alloc::size_type **size_type**
- typedef [stored_data](#)< value_type, size_type > **stored_data_type**
- typedef __rebind_va::value_type **value_type**

4.323.1 Detailed Description

```
template<typename Key, typename Mapped, typename _Alloc>
struct __gnu_pbds::detail::type_base< Key, Mapped, _Alloc, true >
```

Specialization of type_base for the case where the hash value is stored alongside each value.

Definition at line 147 of file types_traits.hpp.

The documentation for this struct was generated from the following file:

- [types_traits.hpp](#)

4.324 `__gnu_pbds::detail::type_base< Key, null_type, _Alloc, false >` Struct Template Reference

Public Types

- `typedef __rebind_va::const_pointer const_pointer`
- `typedef __rebind_va::const_reference const_reference`
- `typedef __rebind_ma::const_pointer mapped_const_pointer`
- `typedef __rebind_ma::const_reference mapped_const_reference`
- `typedef __rebind_ma::pointer mapped_pointer`
- `typedef __rebind_ma::reference mapped_reference`
- `typedef __rebind_ma::value_type mapped_type`
- `typedef __rebind_va::pointer pointer`
- `typedef __rebind_va::reference reference`
- `typedef _Alloc::size_type size_type`
- `typedef stored_data< value_type, null_type > stored_data_type`
- `typedef Key value_type`

Static Public Attributes

- static [null_type](#) **s_null_type**

4.324.1 Detailed Description

```
template<typename Key, typename _Alloc>
struct __gnu_pbds::detail::type_base< Key, null_type, _Alloc, false >
```

Specialization of `type_base` for the case where the hash value is not stored alongside each value.

Definition at line 181 of file `types_traits.hpp`.

The documentation for this struct was generated from the following file:

- [types_traits.hpp](#)

4.325 `__gnu_pbds::detail::type_base< Key, null_type, _Alloc, true >` Struct Template Reference

Public Types

- `typedef __rebind_va::const_pointer const_pointer`
- `typedef __rebind_va::const_reference const_reference`
- `typedef __rebind_ma::const_pointer mapped_const_pointer`
- `typedef __rebind_ma::const_reference mapped_const_reference`
- `typedef __rebind_ma::pointer mapped_pointer`
- `typedef __rebind_ma::reference mapped_reference`
- `typedef __rebind_ma::value_type mapped_type`
- `typedef __rebind_va::pointer pointer`
- `typedef __rebind_va::reference reference`
- `typedef _Alloc::size_type size_type`
- `typedef stored_data< value_type, size_type > stored_data_type`
- `typedef Key value_type`

Static Public Attributes

- static [null_type](#) **s_null_type**

4.325.1 Detailed Description

```
template<typename Key, typename _Alloc>
struct __gnu_pbds::detail::type_base< Key, null_type, _Alloc, true >
```

Specialization of type_base for the case where the hash value is stored alongside each value.

Definition at line 220 of file types_traits.hpp.

The documentation for this struct was generated from the following file:

- [types_traits.hpp](#)

4.326 __gnu_pbds::detail::type_dispatch< Key, Mapped, _Alloc, Store_Hash > Struct Template Reference

Public Types

- typedef [type_base](#)< Key, Mapped, _Alloc, Store_Hash > **type**

4.326.1 Detailed Description

```
template<typename Key, typename Mapped, typename _Alloc, bool Store_Hash>
struct __gnu_pbds::detail::type_dispatch< Key, Mapped, _Alloc, Store_Hash >
```

Type base dispatch.

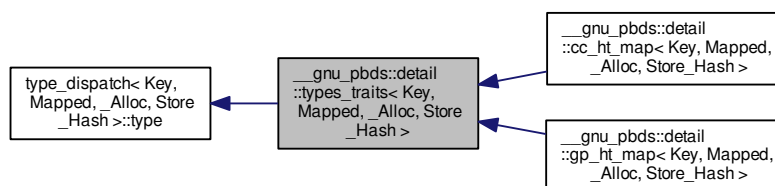
Definition at line 256 of file types_traits.hpp.

The documentation for this struct was generated from the following file:

- [types_traits.hpp](#)

4.327 __gnu_pbds::detail::types_traits< Key, Mapped, _Alloc, Store_Hash > Struct Template Reference

Inheritance diagram for __gnu_pbds::detail::types_traits< Key, Mapped, _Alloc, Store_Hash >:



Public Types

- typedef `std::pair< size_type, size_type >` **comp_hash**
- typedef `__rebind_a::const_pointer` **key_const_pointer**
- typedef `__rebind_a::const_reference` **key_const_reference**
- typedef `__rebind_a::pointer` **key_pointer**
- typedef `__rebind_a::reference` **key_reference**
- typedef `__rebind_a::value_type` **key_type**
- typedef `__nothrowcopy::indicator` **no_throw_indicator**
- typedef `_Alloc::size_type` **size_type**
- typedef `integral_constant< int, Store_Hash >` **store_extra**

Public Attributes

- no_throw_indicator **m_no_throw_copies_indicator**
- store_extra **m_store_extra_indicator**

4.327.1 Detailed Description

```
template<typename Key, typename Mapped, typename _Alloc, bool Store_Hash>
struct __gnu_pbds::detail::types_traits< Key, Mapped, _Alloc, Store_Hash >
```

Traits for abstract types.

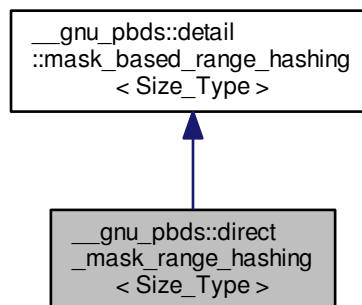
Definition at line 263 of file `types_traits.hpp`.

The documentation for this struct was generated from the following file:

- [types_traits.hpp](#)

4.328 `__gnu_pbds::direct_mask_range_hashing< Size_Type >` Class Template Reference

Inheritance diagram for `__gnu_pbds::direct_mask_range_hashing< Size_Type >`:



Public Types

- typedef Size_Type **size_type**

Public Member Functions

- void **swap** ([direct_mask_range_hashing](#)< Size_Type > &other)

Protected Member Functions

- void **notify_resized** (size_type size)
- size_type [operator\(\)](#) (size_type hash) const
- size_type **range_hash** (size_type hash) const
- void **swap** (mask_based_range_hashing &other)

4.328.1 Detailed Description

```
template<typename Size_Type = std::size_t>
class __gnu_pbds::direct_mask_range_hashing< Size_Type >
```

A mask range-hashing class (uses a bitmask).

Definition at line 109 of file hash_policy.hpp.

4.328.2 Member Function Documentation

4.328.2.1 `template<typename Size_Type > direct_mask_range_hashing< Size_Type >::size_type
__gnu_pbds::direct_mask_range_hashing< Size_Type >::operator() (size_type hash) const [inline],
[protected]`

Transforms the __hash value hash into a ranged-hash value (using a bit-mask).

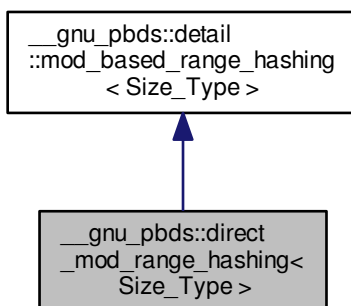
Definition at line 57 of file hash_policy.hpp.

The documentation for this class was generated from the following file:

- [hash_policy.hpp](#)

4.329 `__gnu_pbds::direct_mod_range_hashing< Size_Type >` Class Template Reference

Inheritance diagram for `__gnu_pbds::direct_mod_range_hashing< Size_Type >`:



Public Types

- typedef `Size_Type` **size_type**

Public Member Functions

- void **swap** ([direct_mod_range_hashing](#)< `Size_Type` > &other)

Protected Member Functions

- void **notify_resized** (size_type size)
- size_type [operator\(\)](#) (size_type hash) const
- size_type **range_hash** (size_type s) const
- void **swap** (mod_based_range_hashing &other)

4.329.1 Detailed Description

```
template<typename Size_Type = std::size_t>
class __gnu_pbds::direct_mod_range_hashing< Size_Type >
```

A mod range-hashing class (uses the modulo function).

Definition at line 141 of file `hash_policy.hpp`.

4.329.2 Member Function Documentation

4.329.2.1 `template<typename Size_Type > direct_mod_range_hashing< Size_Type >::size_type
__gnu_pbds::direct_mod_range_hashing< Size_Type >::operator() (size_type hash) const` `[inline],`
`[protected]`

Transforms the `__hash` value `hash` into a ranged-hash value (using a modulo operation).

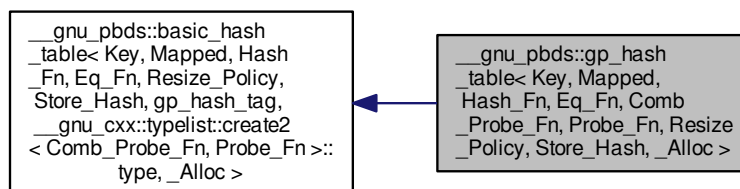
Definition at line 57 of file `hash_policy.hpp`.

The documentation for this class was generated from the following file:

- [hash_policy.hpp](#)

4.330 `__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >`:



Public Types

- typedef `Comb_Probe_Fn` **`comb_probe_fn`**
- typedef [gp_hash_tag](#) **`container_category`**
- typedef `Eq_Fn` **`eq_fn`**
- typedef `Hash_Fn` **`hash_fn`**
- typedef `Probe_Fn` **`probe_fn`**
- typedef `Resize_Policy` **`resize_policy`**

Public Member Functions

- [gp_hash_table](#) ()
- [gp_hash_table](#) (const hash_fn &h)
- [gp_hash_table](#) (const hash_fn &h, const eq_fn &e)
- [gp_hash_table](#) (const hash_fn &h, const eq_fn &e, const comb_probe_fn &cp)
- [gp_hash_table](#) (const hash_fn &h, const eq_fn &e, const comb_probe_fn &cp, const probe_fn &p)
- [gp_hash_table](#) (const hash_fn &h, const eq_fn &e, const comb_probe_fn &cp, const probe_fn &p, const resize_policy &rp)
- `template<typename It >`
[gp_hash_table](#) (It first, It last)
- `template<typename It >`
[gp_hash_table](#) (It first, It last, const hash_fn &h)
- `template<typename It >`
[gp_hash_table](#) (It first, It last, const hash_fn &h, const eq_fn &e)
- `template<typename It >`
[gp_hash_table](#) (It first, It last, const hash_fn &h, const eq_fn &e, const comb_probe_fn &cp)
- `template<typename It >`
[gp_hash_table](#) (It first, It last, const hash_fn &h, const eq_fn &e, const comb_probe_fn &cp, const probe_fn &p)
- `template<typename It >`
[gp_hash_table](#) (It first, It last, const hash_fn &h, const eq_fn &e, const comb_probe_fn &cp, const probe_fn &p, const resize_policy &rp)
- [gp_hash_table](#) (const [gp_hash_table](#) &other)
- [gp_hash_table](#) & **operator=** (const [gp_hash_table](#) &other)
- void **swap** ([gp_hash_table](#) &other)

4.330.1 Detailed Description

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn
= typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn = detail::default_comb_hash_fn::type, typename Probe_Fn
= typename detail::default_probe_fn<Comb_Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<
Comb_Probe_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> >
class __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >
```

A general-probing hash-based associative container.

Template Parameters

<i>Key</i>	Key type.
<i>Mapped</i>	Map type.
<i>Hash_Fn</i>	Hashing functor.
<i>Eq_Fn</i>	Equal functor.
<i>Comb_Probe_Fn</i>	Combining probe functor. If <i>Hash_Fn</i> is not null_type, then this is the ranged-probe functor; otherwise, this is the range-hashing functor. XXX See Design::Hash-Based Containers::Hash Policies.
<i>Probe_Fn</i>	Probe functor.
<i>Resize_Policy</i>	Resizes hash.
<i>Store_Hash</i>	Indicates whether the hash value will be stored along with each key. If <i>Hash_Fn</i> is null_type, then the container will not compile if this value is true
<i>_Alloc</i>	Allocator type.

Base tag choices are: `gp_hash_tag`.

Base is `basic_hash_table`.

Definition at line 368 of file `assoc_container.hpp`.

4.330.2 Constructor & Destructor Documentation

4.330.2.1 `template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn = detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >::gp_hash_table () [inline]`

Default constructor.

Definition at line 382 of file `assoc_container.hpp`.

4.330.2.2 `template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn = detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >::gp_hash_table (const hash_fn & h) [inline]`

Constructor taking some policy objects. `r_hash_fn` will be copied by the `hash_fn` object of the container object.

Definition at line 386 of file `assoc_container.hpp`.

4.330.2.3 `template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn = detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >::gp_hash_table (const hash_fn & h, const eq_fn & e) [inline]`

Constructor taking some policy objects. `r_hash_fn` will be copied by the `hash_fn` object of the container object, and `r_eq_fn` will be copied by the `eq_fn` object of the container object.

Definition at line 393 of file `assoc_container.hpp`.

4.330.2.4 `template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn = detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >::gp_hash_table (const hash_fn & h, const eq_fn & e, const comb_probe_fn & cp) [inline]`

Constructor taking some policy objects. `r_hash_fn` will be copied by the `hash_fn` object of the container object, `r_eq_fn` will be copied by the `eq_fn` object of the container object, and `r_comb_probe_fn` will be copied by the `comb_probe_fn` object of the container object.

Definition at line 401 of file `assoc_container.hpp`.

4.330.2.5 `template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn = detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >::gp_hash_table (const hash_fn & h, const eq_fn & e, const comb_probe_fn & cp, const probe_fn & p) [inline]`

Constructor taking some policy objects. `r_hash_fn` will be copied by the `hash_fn` object of the container object, `r_eq_fn` will be copied by the `eq_fn` object of the container object, `r_comb_probe_fn` will be copied by the `comb_probe_fn` object of the container object, and `r_probe_fn` will be copied by the `probe_fn` object of the container object.

Definition at line 410 of file `assoc_container.hpp`.

4.330.2.6 `template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn = detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >::gp_hash_table (const hash_fn & h, const eq_fn & e, const comb_probe_fn & cp, const probe_fn & p, const resize_policy & rp) [inline]`

Constructor taking some policy objects. `r_hash_fn` will be copied by the `hash_fn` object of the container object, `r_eq_fn` will be copied by the `eq_fn` object of the container object, `r_comb_probe_fn` will be copied by the `comb_probe_fn` object of the container object, `r_probe_fn` will be copied by the `probe_fn` object of the container object, and `r_resize_policy` will be copied by the `Resize_Policy` object of the container object.

Definition at line 422 of file `assoc_container.hpp`.

4.330.2.7 `template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn = detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> template<typename It > __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >::gp_hash_table (It first, It last) [inline]`

Constructor taking `__iterators` to a range of `value_types`. The `value_types` between `first_it` and `last_it` will be inserted into the container object.

Definition at line 430 of file `assoc_container.hpp`.

4.330.2.8 `template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn = detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> template<typename It > __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >::gp_hash_table (It first, It last, const hash_fn & h) [inline]`

Constructor taking __iterators to a range of value_types and some policy objects. The value_types between first_it and last_it will be inserted into the container object. r_hash_fn will be copied by the hash_fn object of the container object.

Definition at line 438 of file assoc_container.hpp.

4.330.2.9 `template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn = detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> template<typename It > __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >::gp_hash_table (It first, It last, const hash_fn & h, const eq_fn & e) [inline]`

Constructor taking __iterators to a range of value_types and some policy objects. The value_types between first_it and last_it will be inserted into the container object. r_hash_fn will be copied by the hash_fn object of the container object, and r_eq_fn will be copied by the eq_fn object of the container object.

Definition at line 449 of file assoc_container.hpp.

4.330.2.10 `template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn = detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> template<typename It > __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >::gp_hash_table (It first, It last, const hash_fn & h, const eq_fn & e, const comb_probe_fn & cp) [inline]`

Constructor taking __iterators to a range of value_types and some policy objects. The value_types between first_it and last_it will be inserted into the container object. r_hash_fn will be copied by the hash_fn object of the container object, r_eq_fn will be copied by the eq_fn object of the container object, and r_comb_probe_fn will be copied by the comb_probe_fn object of the container object.

Definition at line 461 of file assoc_container.hpp.

4.330.2.11 `template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn = detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> template<typename It > __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >::gp_hash_table (It first, It last, const hash_fn & h, const eq_fn & e, const comb_probe_fn & cp, const probe_fn & p) [inline]`

Constructor taking __iterators to a range of value_types and some policy objects. The value_types between first_it and last_it will be inserted into the container object. r_hash_fn will be copied by the hash_fn object of the container object, r_eq_fn will be copied by the eq_fn object of the container object, r_comb_probe_fn will be copied by the comb_probe_fn object of the container object, and r_probe_fn will be copied by the probe_fn object of the container object.

Definition at line 475 of file assoc_container.hpp.

```

4.330.2.12 template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type,
typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn =
detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_Probe_Fn>::type,
typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>::type, bool Store_Hash
= detail::default_store_hash, typename _Alloc = std::allocator<char>>> template<typename It >
__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy,
Store_Hash, _Alloc >::gp_hash_table ( It first, It last, const hash_fn & h, const eq_fn & e, const comb_probe_fn &
cp, const probe_fn & p, const resize_policy & rp ) [inline]

```

Constructor taking __iterators to a range of value_types and some policy objects. The value_types between first_it and last_it will be inserted into the container object. r_hash_fn will be copied by the hash_fn object of the container object, r_eq_fn will be copied by the eq_fn object of the container object, r_comb_probe_fn will be copied by the comb_↔_probe_fn object of the container object, r_probe_fn will be copied by the probe_fn object of the container object, and r_resize_policy will be copied by the resize_policy object of the container object.

Definition at line 491 of file assoc_container.hpp.

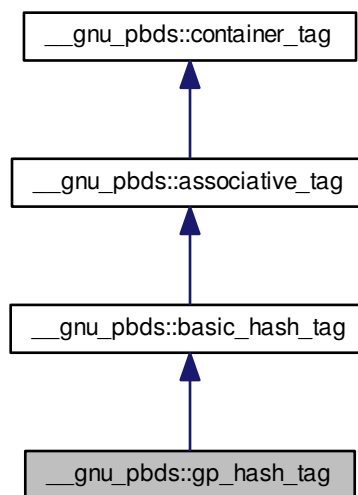
References `std::swap()`.

The documentation for this class was generated from the following file:

- [assoc_container.hpp](#)

4.331 __gnu_pbds::gp_hash_tag Struct Reference

Inheritance diagram for __gnu_pbds::gp_hash_tag:



4.331.1 Detailed Description

General-probing hash.

Definition at line 144 of file tag_and_trait.hpp.

The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

4.332 `__gnu_pbds::hash_exponential_size_policy< Size_Type >` Class Template Reference

Public Types

- typedef Size_Type **size_type**

Public Member Functions

- [hash_exponential_size_policy](#) (size_type start_size=8, size_type grow_factor=2)
- void **swap** ([hash_exponential_size_policy](#)< Size_Type > &other)

Protected Member Functions

- size_type **get_nearest_larger_size** (size_type size) const
- size_type **get_nearest_smaller_size** (size_type size) const

4.332.1 Detailed Description

```
template<typename Size_Type = std::size_t>
class __gnu_pbds::hash_exponential_size_policy< Size_Type >
```

A size policy whose sequence of sizes form an exponential sequence (typically powers of 2).

Definition at line 413 of file hash_policy.hpp.

4.332.2 Constructor & Destructor Documentation

4.332.2.1 `template<typename Size_Type > __gnu_pbds::hash_exponential_size_policy< Size_Type >::hash_exponential_size_policy (size_type start_size = 8, size_type grow_factor = 2)`

Default constructor, or onstructor taking a start_size, or constructor taking a start size and grow_factor. The policy will use the sequence of sizes start_size, start_size* grow_factor, start_size* grow_factor^2, ...

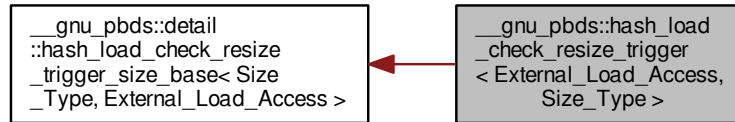
Definition at line 44 of file hash_policy.hpp.

The documentation for this class was generated from the following file:

- [hash_policy.hpp](#)

4.333 `__gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >` Class Template Reference

Inheritance diagram for `__gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >`:



Public Types

- enum { [external_load_access](#) }
- typedef `Size_Type` **size_type**

Public Member Functions

- [hash_load_check_resize_trigger](#) (float load_min=0.125, float load_max=0.5)
- `std::pair< float, float >` [get_loads](#) () const
- void [set_loads](#) (`std::pair< float, float >` load_pair)
- void **swap** ([hash_load_check_resize_trigger](#) &other)

Protected Member Functions

- bool **is_grow_needed** (size_type size, size_type num_entries) const
- bool **is_resize_needed** () const
- void [notify_cleared](#) ()
- void **notify_erase_search_collision** ()
- void **notify_erase_search_end** ()
- void **notify_erase_search_start** ()
- void **notify_erased** (size_type num_entries)
- void **notify_externally_resized** (size_type new_size)
- void **notify_find_search_collision** ()
- void **notify_find_search_end** ()
- void **notify_find_search_start** ()
- void **notify_insert_search_collision** ()
- void **notify_insert_search_end** ()
- void **notify_insert_search_start** ()
- void [notify_inserted](#) (size_type num_entries)
- void [notify_resized](#) (size_type new_size)

4.333.1 Detailed Description

```
template<bool External_Load_Access = false, typename Size_Type = std::size_t>
class __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >
```

A resize trigger policy based on a load check. It keeps the load factor between some load factors `load_min` and `load_max`.

Definition at line 175 of file `hash_policy.hpp`.

4.333.2 Member Enumeration Documentation

4.333.2.1 `template<bool External_Load_Access = false, typename Size_Type = std::size_t> anonymous enum`

Enumerator

external_load_access Specifies whether the load factor can be accessed externally. The two options have different trade-offs in terms of flexibility, genericity, and encapsulation.

Definition at line 180 of file `hash_policy.hpp`.

4.333.3 Constructor & Destructor Documentation

4.333.3.1 `template<bool External_Load_Access, typename Size_Type > __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >::hash_load_check_resize_trigger (float load_min = 0.125, float load_max = 0.5)`

Default constructor, or constructor taking `load_min` and `load_max` load factors between which this policy will keep the actual load.

Definition at line 47 of file `hash_policy.hpp`.

4.333.4 Member Function Documentation

4.333.4.1 `template<bool External_Load_Access, typename Size_Type > std::pair< float, float > __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >::get_loads () const [inline]`

Returns a pair of the minimal and maximal loads, respectively.

Definition at line 236 of file `hash_policy.hpp`.

4.333.4.2 `template<bool External_Load_Access, typename Size_Type > void __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >::notify_cleared () [protected]`

Notifies the table was cleared.

Definition at line 206 of file `hash_policy.hpp`.

4.333.4.3 `template<bool External_Load_Access, typename Size_Type > void __gnu_pbds::hash_load_check_↔
resize_trigger< External_Load_Access, Size_Type >::notify_inserted (size_type num_entries) [inline],
[protected]`

Notifies an element was inserted. The total number of entries in the table is `num_entries`.

Definition at line 109 of file `hash_policy.hpp`.

4.333.4.4 `template<bool External_Load_Access, typename Size_Type > void __gnu_pbds::hash_load_↔
check_resize_trigger< External_Load_Access, Size_Type >::notify_resized (size_type new_size)
[protected]`

Notifies the table was resized as a result of this object's signifying that a resize is needed.

Definition at line 151 of file `hash_policy.hpp`.

4.333.4.5 `template<bool External_Load_Access, typename Size_Type > void __gnu_pbds::hash_load_check_↔
_resize_trigger< External_Load_Access, Size_Type >::set_loads (std::pair< float, float > load_pair
)`

Sets the loads through a pair of the minimal and maximal loads, respectively.

Definition at line 245 of file `hash_policy.hpp`.

The documentation for this class was generated from the following file:

- [hash_policy.hpp](#)

4.334 `__gnu_pbds::hash_prime_size_policy` Class Reference

Public Types

- typedef `std::size_t` [size_type](#)

Public Member Functions

- [hash_prime_size_policy](#) ([size_type](#) start_size=8)
- void **swap** ([hash_prime_size_policy](#) &other)

Protected Member Functions

- [size_type](#) **get_nearest_larger_size** ([size_type](#) size) const
- [size_type](#) **get_nearest_smaller_size** ([size_type](#) size) const

4.334.1 Detailed Description

A size policy whose sequence of sizes form a nearly-exponential sequence of primes.

Definition at line 450 of file hash_policy.hpp.

4.334.2 Member Typedef Documentation

4.334.2.1 typedef std::size_t __gnu_pbds::hash_prime_size_policy::size_type

Size type.

Definition at line 454 of file hash_policy.hpp.

4.334.3 Constructor & Destructor Documentation

4.334.3.1 __gnu_pbds::hash_prime_size_policy::hash_prime_size_policy (size_type start_size = 8) [inline]

Default constructor, or onstructor taking a start_size The policy will use the sequence of sizes approximately start_size, start_size* 2, start_size* 2^2, ...

Definition at line 127 of file hash_policy.hpp.

The documentation for this class was generated from the following file:

- [hash_policy.hpp](#)

4.335 __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type > Class Template Reference

Inherits Size_Policy, and Trigger_Policy.

Public Types

- enum { **external_size_access** }
- typedef Size_Policy **size_policy**
- typedef Size_Type **size_type**
- typedef Trigger_Policy **trigger_policy**

Public Member Functions

- `hash_standard_resize_policy` ()
- `hash_standard_resize_policy` (const `Size_Policy` &`r_size_policy`)
- `hash_standard_resize_policy` (const `Size_Policy` &`r_size_policy`, const `Trigger_Policy` &`r_trigger_policy`)
- `size_type` `get_actual_size` () const
- `Size_Policy` & `get_size_policy` ()
- const `Size_Policy` & `get_size_policy` () const
- `Trigger_Policy` & `get_trigger_policy` ()
- const `Trigger_Policy` & `get_trigger_policy` () const
- void `resize` (`size_type` `suggested_new_size`)
- void `swap` (`hash_standard_resize_policy`< `Size_Policy`, `Trigger_Policy`, `External_Size_Access`, `Size_Type` > &`other`)

Protected Member Functions

- `size_type` `get_new_size` (`size_type` `size`, `size_type` `num_used_e`) const
- bool `is_resize_needed` () const
- void `notify_cleared` ()
- void `notify_erase_search_collision` ()
- void `notify_erase_search_end` ()
- void `notify_erase_search_start` ()
- void `notify_erased` (`size_type` `num_e`)
- void `notify_find_search_collision` ()
- void `notify_find_search_end` ()
- void `notify_find_search_start` ()
- void `notify_insert_search_collision` ()
- void `notify_insert_search_end` ()
- void `notify_insert_search_start` ()
- void `notify_inserted` (`size_type` `num_e`)
- void `notify_resized` (`size_type` `new_size`)

4.335.1 Detailed Description

```
template<typename Size_Policy = hash_exponential_size_policy<>, typename Trigger_Policy = hash_load_check_resize_↵
trigger<>, bool External_Size_Access = false, typename Size_Type = std::size_t>
class __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >
```

A resize policy which delegates operations to size and trigger policies.

Definition at line 489 of file `hash_policy.hpp`.

4.335.2 Constructor & Destructor Documentation

```
4.335.2.1 template<typename Size_Policy , typename Trigger_Policy , bool External_Size_Access, typename Size_Type >
__gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type
>::hash_standard_resize_policy ( )
```

Default constructor.

Definition at line 44 of file `hash_policy.hpp`.

```
4.335.2.2  template<typename Size_Policy, typename Trigger_Policy , bool External_Size_Access, typename Size_Type >
            __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type
            >::hash_standard_resize_policy ( const Size_Policy & r_size_policy )
```

constructor taking some policies `r_size_policy` will be copied by the `Size_Policy` object of this object.

Definition at line 50 of file `hash_policy.hpp`.

```
4.335.2.3  template<typename Size_Policy, typename Trigger_Policy, bool External_Size_Access, typename Size_Type >
            __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type
            >::hash_standard_resize_policy ( const Size_Policy & r_size_policy, const Trigger_Policy & r_trigger_policy )
```

constructor taking some policies. `r_size_policy` will be copied by the `Size_Policy` object of this object. `r_trigger_policy` will be copied by the `Trigger_Policy` object of this object.

Definition at line 56 of file `hash_policy.hpp`.

4.335.3 Member Function Documentation

```
4.335.3.1  template<typename Size_Policy , typename Trigger_Policy , bool External_Size_Access, typename Size_Type >
            hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >::size_type
            __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type
            >::get_actual_size ( ) const    [inline]
```

Returns the actual size of the container.

Definition at line 177 of file `hash_policy.hpp`.

```
4.335.3.2  template<typename Size_Policy , typename Trigger_Policy , bool External_Size_Access, typename Size_Type >
            hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >::size_type
            __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type
            >::get_new_size ( size_type size, size_type num_used_e ) const    [protected]
```

Queries what the new size should be, when the container is resized naturally. The current `__size` of the container is `size`, and the number of used entries within the container is `num_used_e`.

Definition at line 158 of file `hash_policy.hpp`.

```
4.335.3.3  template<typename Size_Policy , typename Trigger_Policy , bool External_Size_Access, typename Size_Type >
            Size_Policy & __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access,
            Size_Type >::get_size_policy ( )
```

Access to the `Size_Policy` object used.

Definition at line 242 of file `hash_policy.hpp`.

4.335.3.4 `template<typename Size_Policy , typename Trigger_Policy , bool External_Size_Access, typename Size_Type > const Size_Policy & __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >::get_size_policy () const`

Const access to the Size_Policy object used.

Definition at line 248 of file hash_policy.hpp.

4.335.3.5 `template<typename Size_Policy , typename Trigger_Policy , bool External_Size_Access, typename Size_Type > Trigger_Policy & __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >::get_trigger_policy ()`

Access to the Trigger_Policy object used.

Definition at line 230 of file hash_policy.hpp.

4.335.3.6 `template<typename Size_Policy , typename Trigger_Policy , bool External_Size_Access, typename Size_Type > const Trigger_Policy & __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >::get_trigger_policy () const`

Access to the Trigger_Policy object used.

Definition at line 236 of file hash_policy.hpp.

4.335.3.7 `template<typename Size_Policy , typename Trigger_Policy , bool External_Size_Access, typename Size_Type > void __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >::resize (size_type suggested_new_size)`

Resizes the container to suggested_new_size, a suggested size (the actual size will be determined by the Size_Policy object).

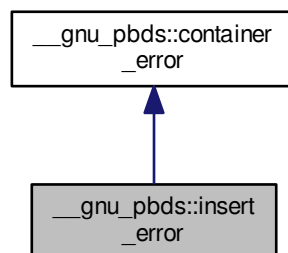
Definition at line 186 of file hash_policy.hpp.

The documentation for this class was generated from the following file:

- [hash_policy.hpp](#)

4.336 __gnu_pbds::insert_error Struct Reference

Inheritance diagram for __gnu_pbds::insert_error:



4.336.1 Detailed Description

An entry cannot be inserted into a container object for logical reasons (not, e.g., if memory is unavailable, in which case the `allocator_type`'s exception will be thrown).

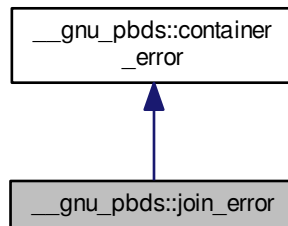
Definition at line 66 of file `exception.hpp`.

The documentation for this struct was generated from the following file:

- [exception.hpp](#)

4.337 `__gnu_pbds::join_error` Struct Reference

Inheritance diagram for `__gnu_pbds::join_error`:



4.337.1 Detailed Description

A join cannot be performed logical reasons (i.e., the ranges of the two container objects being joined overlaps).

Definition at line 70 of file `exception.hpp`.

The documentation for this struct was generated from the following file:

- [exception.hpp](#)

4.338 `__gnu_pbds::linear_probe_fn< Size_Type >` Class Template Reference

Public Types

- typedef `Size_Type` **size_type**

4.339 `__gnu_pbds::list_update< Key, Mapped, Eq_Fn, Update_Policy, _Alloc >` Class Template Reference 949

Public Member Functions

- void **swap** ([linear_probe_fn](#)< Size_Type > &other)

Protected Member Functions

- size_type [operator\(\)](#) (size_type i) const

4.338.1 Detailed Description

```
template<typename Size_Type = std::size_t>
class __gnu_pbds::linear_probe_fn< Size_Type >
```

A probe sequence policy using fixed increments.

Definition at line 61 of file hash_policy.hpp.

4.338.2 Member Function Documentation

4.338.2.1 `template<typename Size_Type > linear_probe_fn< Size_Type >::size_type __gnu_pbds::linear_probe_fn< Size_Type >::operator() (size_type i) const` `[inline], [protected]`

Returns the i-th offset from the hash value.

Definition at line 51 of file hash_policy.hpp.

The documentation for this class was generated from the following file:

- [hash_policy.hpp](#)

4.339 `__gnu_pbds::list_update< Key, Mapped, Eq_Fn, Update_Policy, _Alloc >` Class Template Reference

Inherits type< Key, Mapped, _Alloc, list_update_tag, __gnu_cxx::typelist::create2< Eq_Fn, Update_Policy >::type >.

Public Types

- typedef [list_update_tag](#) **container_category**
- typedef Eq_Fn **eq_fn**
- typedef Update_Policy **update_policy**

Public Member Functions

- template<typename It >
[list_update](#) (It first, It last)
- **list_update** (const [list_update](#) &other)
- [list_update](#) & **operator=** (const [list_update](#) &other)
- void **swap** ([list_update](#) &other)

4.339.1 Detailed Description

```
template<typename Key, typename Mapped, class Eq_Fn = typename detail::default_eq_fn<Key>::type, class Update_Policy =
detail::default_update_policy::type, class _Alloc = std::allocator<char>>
class __gnu_pbds::list_update< Key, Mapped, Eq_Fn, Update_Policy, _Alloc >
```

A list-update based associative container.

Template Parameters

<i>Key</i>	Key type.
<i>Mapped</i>	Map type.
<i>Eq_Fn</i>	Equal functor.
<i>Update_Policy</i>	Update policy, determines when an element will be moved to the front of the list.
<i>_Alloc</i>	Allocator type.

Base is detail::lu_map.

Definition at line 815 of file assoc_container.hpp.

4.339.2 Constructor & Destructor Documentation

4.339.2.1 `template<typename Key , typename Mapped , class Eq_Fn = typename detail::default_eq_fn<Key>::type, class Update_Policy = detail::default_update_policy::type, class _Alloc = std::allocator<char>> template<typename It > __gnu_pbds::list_update< Key, Mapped, Eq_Fn, Update_Policy, _Alloc >::list_update (It first, It last)`
`[inline]`

Constructor taking __iterators to a range of value_types. The value_types between first_it and last_it will be inserted into the container object.

Definition at line 831 of file assoc_container.hpp.

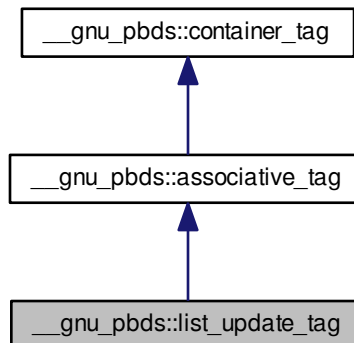
References std::swap().

The documentation for this class was generated from the following file:

- [assoc_container.hpp](#)

4.340 __gnu_pbds::list_update_tag Struct Reference

Inheritance diagram for __gnu_pbds::list_update_tag:



4.340.1 Detailed Description

List-update.

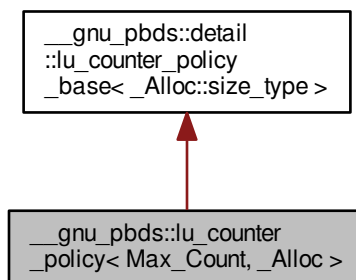
Definition at line 168 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

4.341 `__gnu_pbds::lu_counter_policy< Max_Count, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::lu_counter_policy< Max_Count, _Alloc >`:



Public Types

- enum { `max_count` }
- typedef `_Alloc` **allocator_type**
- typedef `__rebind_m::other::reference` `metadata_reference`
- typedef `detail::lu_counter_metadata< size_type >` `metadata_type`
- typedef `allocator_type::size_type` **size_type**

Public Member Functions

- `metadata_type operator() ()` const
- `bool operator() (metadata_reference r_data)` const

Private Member Functions

- `lu_counter_metadata< size_type > operator() (size_type max_size)` const
- `bool operator() (Metadata_Reference r_data, size_type m_max_count)` const

4.341.1 Detailed Description

```
template<std::size_t Max_Count = 5, typename _Alloc = std::allocator<char>>
class __gnu_pbds::lu_counter_policy< Max_Count, _Alloc >
```

A list-update policy that moves elements to the front of the list based on the counter algorithm.

Definition at line 92 of file list_update_policy.hpp.

4.341.2 Member Typedef Documentation

4.341.2.1 `template<std::size_t Max_Count = 5, typename _Alloc = std::allocator<char>> typedef __rebind_m::other::reference
__gnu_pbds::lu_counter_policy< Max_Count, _Alloc >::metadata_reference`

Reference to metadata on which this functor operates.

Definition at line 115 of file list_update_policy.hpp.

4.341.2.2 `template<std::size_t Max_Count = 5, typename _Alloc = std::allocator<char>> typedef
detail::lu_counter_metadata<size_type> __gnu_pbds::lu_counter_policy< Max_Count, _Alloc
>::metadata_type`

Metadata on which this functor operates.

Definition at line 107 of file list_update_policy.hpp.

4.341.3 Member Enumeration Documentation

4.341.3.1 `template<std::size_t Max_Count = 5, typename _Alloc = std::allocator<char>> anonymous enum`

Enumerator

max_count When some element is accessed this number of times, it will be moved to the front of the list.

Definition at line 99 of file list_update_policy.hpp.

4.341.4 Member Function Documentation

4.341.4.1 `template<std::size_t Max_Count = 5, typename _Alloc = std::allocator<char>> metadata_type
__gnu_pbds::lu_counter_policy< Max_Count, _Alloc >::operator() () const [inline]`

Creates a metadata object.

Definition at line 119 of file list_update_policy.hpp.


```
4.341.4.2 template<std::size_t Max_Count = 5, typename _Alloc = std::allocator<char>> bool __gnu_pbds::lu_counter_policy< Max_Count, _Alloc >::operator() ( metadata_reference r_data ) const
[inline]
```

Decides whether a metadata object should be moved to the front of the list.

Definition at line 125 of file `list_update_policy.hpp`.

The documentation for this class was generated from the following file:

- [list_update_policy.hpp](#)

4.342 `__gnu_pbds::lu_move_to_front_policy<_Alloc>` Class Template Reference

Public Types

- typedef `_Alloc` **allocator_type**
- typedef `__rebind_m::other::reference` [metadata_reference](#)
- typedef [null_type](#) `metadata_type`

Public Member Functions

- [metadata_type operator\(\)](#) () const
- bool [operator\(\)](#) ([metadata_reference](#) r_metadata) const

4.342.1 Detailed Description

```
template<typename _Alloc = std::allocator<char>>
class __gnu_pbds::lu_move_to_front_policy<_Alloc>
```

A list-update policy that unconditionally moves elements to the front of the list. A null type means that each link in a list-based container does not actually need metadata.

Definition at line 57 of file `list_update_policy.hpp`.

4.342.2 Member Typedef Documentation

```
4.342.2.1 template<typename _Alloc = std::allocator<char>> typedef __rebind_m::other::reference
__gnu_pbds::lu_move_to_front_policy<_Alloc>::metadata_reference
```

Reference to metadata on which this functor operates.

Definition at line 70 of file `list_update_policy.hpp`.

4.342.2.2 `template<typename _Alloc = std::allocator<char>> typedef null_type __gnu_pbds::lu_move_to_front_policy<_Alloc>::metadata_type`

Metadata on which this functor operates.

Definition at line 63 of file `list_update_policy.hpp`.

4.342.3 Member Function Documentation

4.342.3.1 `template<typename _Alloc = std::allocator<char>> metadata_type __gnu_pbds::lu_move_to_front_policy<_Alloc>::operator()() const [inline]`

Creates a metadata object.

Definition at line 74 of file `list_update_policy.hpp`.

4.342.3.2 `template<typename _Alloc = std::allocator<char>> bool __gnu_pbds::lu_move_to_front_policy<_Alloc>::operator()(metadata_reference r_metadata) const [inline]`

Decides whether a metadata object should be moved to the front of the list.

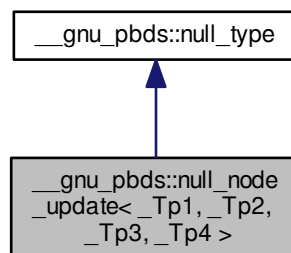
Definition at line 80 of file `list_update_policy.hpp`.

The documentation for this class was generated from the following file:

- [list_update_policy.hpp](#)

4.343 `__gnu_pbds::null_node_update<_Tp1, _Tp2, _Tp3, _Tp4>` Struct Template Reference

Inheritance diagram for `__gnu_pbds::null_node_update<_Tp1, _Tp2, _Tp3, _Tp4>`:



4.343.1 Detailed Description

```
template<typename _Tp1, typename _Tp2, typename _Tp3, typename _Tp4>  
struct __gnu_pbds::null_node_update< _Tp1, _Tp2, _Tp3, _Tp4 >
```

A null node updatator, indicating that no node updates are required.

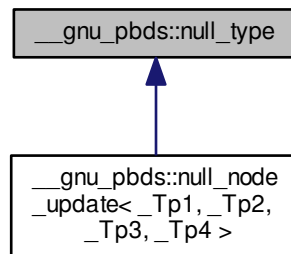
Definition at line 214 of file tag_and_trait.hpp.

The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

4.344 __gnu_pbds::null_type Struct Reference

Inheritance diagram for __gnu_pbds::null_type:



4.344.1 Detailed Description

Represents no type, or absence of type, for template tricks.

In a mapped-policy, indicates that an associative container is a set.

In a list-update policy, indicates that each link does not need metadata.

In a hash policy, indicates that the combining hash function is actually a ranged hash function.

In a probe policy, indicates that the combining probe function is actually a ranged probe function.

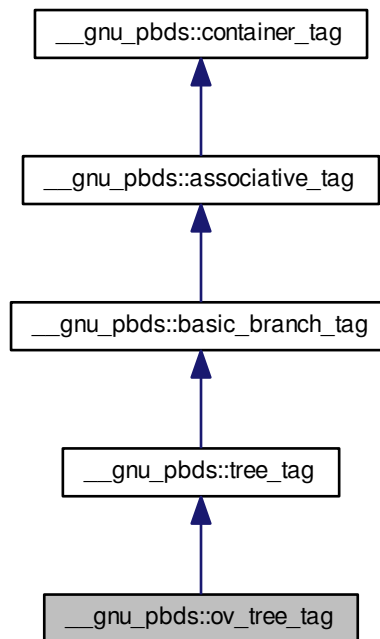
Definition at line 210 of file tag_and_trait.hpp.

The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

4.345 `__gnu_pbds::ov_tree_tag` Struct Reference

Inheritance diagram for `__gnu_pbds::ov_tree_tag`:



4.345.1 Detailed Description

Ordered-vector tree.

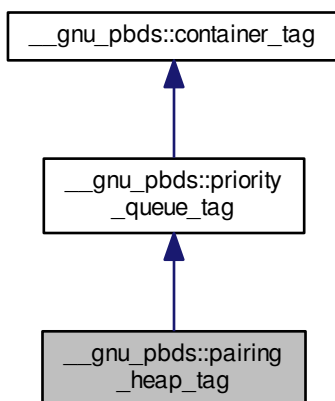
Definition at line 159 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

4.346 __gnu_pbds::pairing_heap_tag Struct Reference

Inheritance diagram for __gnu_pbds::pairing_heap_tag:



4.346.1 Detailed Description

Pairing-heap.

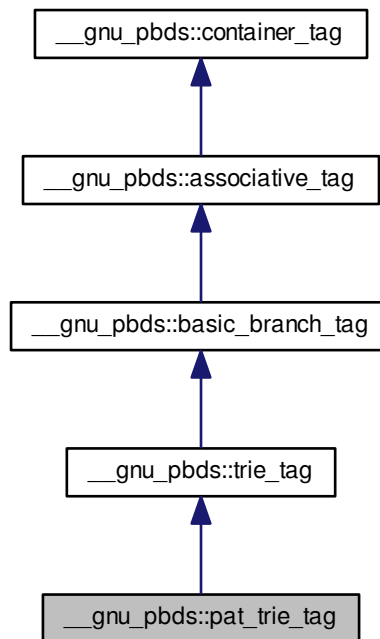
Definition at line 174 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

4.347 __gnu_pbds::pat_trie_tag Struct Reference

Inheritance diagram for __gnu_pbds::pat_trie_tag:



4.347.1 Detailed Description

PATRICIA trie.

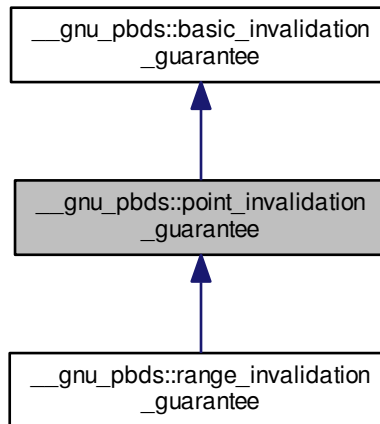
Definition at line 165 of file tag_and_trait.hpp.

The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

4.348 `__gnu_pbds::point_invalidation_guarantee` Struct Reference

Inheritance diagram for `__gnu_pbds::point_invalidation_guarantee`:



4.348.1 Detailed Description

Signifies an invalidation guarantee that includes all those of its base, and additionally, that any point-type iterator, pointer, or reference to a container object's mapped value type is valid as long as its corresponding entry has not be erased, regardless of modifications to the container object.

Definition at line 103 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

4.349 `__gnu_pbds::priority_queue<_Tv, Cmp_Fn, Tag, _Alloc>` Class Template Reference

Inherits `type<_Tv, Cmp_Fn, _Alloc, Tag>`.

Public Types

- typedef `_Alloc` **allocator_type**
- typedef `Cmp_Fn` **cmp_fn**
- typedef `base_type::const_iterator` **const_iterator**
- typedef `__rebind_va::const_pointer` **const_pointer**
- typedef `__rebind_va::const_reference` **const_reference**
- typedef `Tag` **container_category**
- typedef `allocator_type::difference_type` **difference_type**
- typedef `base_type::iterator` **iterator**
- typedef `base_type::point_const_iterator` **point_const_iterator**
- typedef `base_type::point_iterator` **point_iterator**
- typedef `__rebind_va::pointer` **pointer**
- typedef `__rebind_va::reference` **reference**
- typedef `allocator_type::size_type` **size_type**
- typedef `_Tv` **value_type**

Public Member Functions

- [priority_queue](#) (const `cmp_fn` &`r_cmp_fn`)
- `template<typename It >`
[priority_queue](#) (It `first_it`, It `last_it`)
- `template<typename It >`
[priority_queue](#) (It `first_it`, It `last_it`, const `cmp_fn` &`r_cmp_fn`)
- **priority_queue** (const [priority_queue](#) &`other`)
- [priority_queue](#) & **operator=** (const [priority_queue](#) &`other`)
- void **swap** ([priority_queue](#) &`other`)

4.349.1 Detailed Description

```
template<typename _Tv, typename Cmp_Fn = std::less<_Tv>, typename Tag = pairing_heap_tag, typename _Alloc = std::allocator<char>>
class __gnu_pbds::priority_queue<_Tv, Cmp_Fn, Tag, _Alloc>
```

A priority queue composed of one specific heap policy.

Template Parameters

<code>_Tv</code>	Value type.
<code>Cmp_Fn</code>	Comparison functor.
<code>Tag</code>	Instantiating data structure type, see <code>container_tag</code> .
<code>_Alloc</code>	Allocator type.

Base is dispatched at compile time via `Tag`, from the following choices: `binary_heap_tag`, `binomial_heap_tag`, `pairing_heap_tag`, `rc_binomial_heap_tag`, `thin_heap_tag`

Base choices are: `detail::binary_heap`, `detail::binomial_heap`, `detail::pairing_heap`, `detail::rc_binomial_heap`, `detail::thin_heap`.

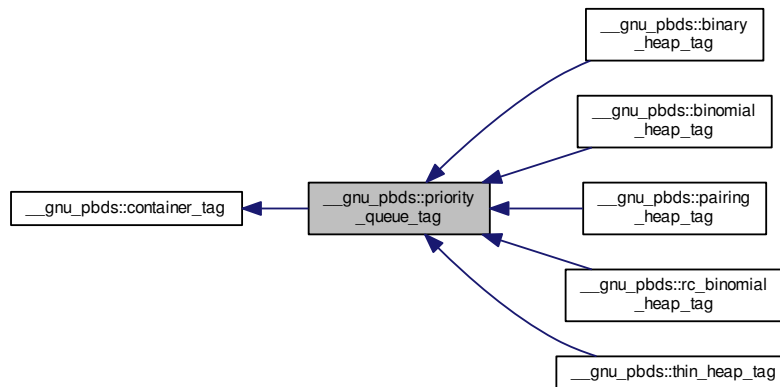
Definition at line 83 of file `priority_queue.hpp`.

The documentation for this class was generated from the following file:

- [priority_queue.hpp](#)

4.350 `__gnu_pbds::priority_queue_tag` Struct Reference

Inheritance diagram for `__gnu_pbds::priority_queue_tag`:



4.350.1 Detailed Description

Basic priority-queue.

Definition at line 171 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

4.351 `__gnu_pbds::quadratic_probe_fn< Size_Type >` Class Template Reference

Public Types

- typedef `Size_Type` **size_type**

Public Member Functions

- void **swap** ([quadratic_probe_fn](#)< `Size_Type` > &other)

Protected Member Functions

- `size_type operator() (size_type i) const`

4.351.1 Detailed Description

```
template<typename Size_Type = std::size_t>
class __gnu_pbds::quadratic_probe_fn< Size_Type >
```

A probe sequence policy using square increments.

Definition at line 85 of file `hash_policy.hpp`.

4.351.2 Member Function Documentation

4.351.2.1 `template<typename Size_Type > quadratic_probe_fn< Size_Type >::size_type __gnu_pbds::quadratic_probe_fn< Size_Type >::operator() (size_type i) const [inline], [protected]`

Returns the i-th offset from the hash value.

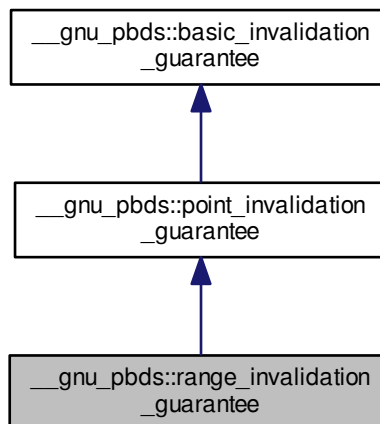
Definition at line 51 of file `hash_policy.hpp`.

The documentation for this class was generated from the following file:

- [hash_policy.hpp](#)

4.352 __gnu_pbds::range_invalidation_guarantee Struct Reference

Inheritance diagram for `__gnu_pbds::range_invalidation_guarantee`:



4.352.1 Detailed Description

Signifies an invalidation guarantee that includes all those of its base, and additionally, that any range-type iterator (including the returns of `begin()` and `end()`) is in the correct relative positions to other range-type iterators as long as its corresponding entry has not be erased, regardless of modifications to the container object.

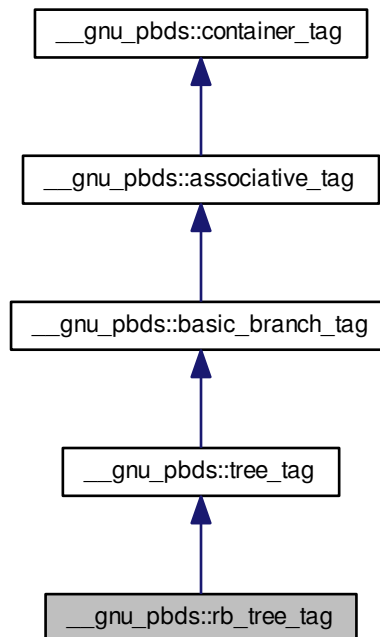
Definition at line 114 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

4.353 __gnu_pbds::rb_tree_tag Struct Reference

Inheritance diagram for `__gnu_pbds::rb_tree_tag`:



4.353.1 Detailed Description

Red-black tree.

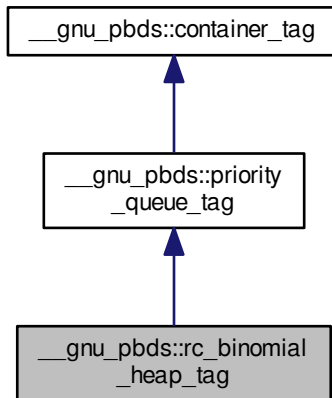
Definition at line 153 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

4.354 `__gnu_pbds::rc_binomial_heap_tag` Struct Reference

Inheritance diagram for `__gnu_pbds::rc_binomial_heap_tag`:



4.354.1 Detailed Description

Redundant-counter binomial-heap.

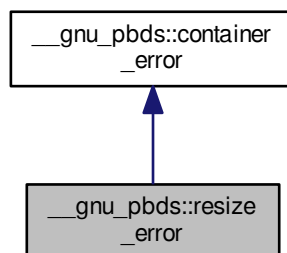
Definition at line 180 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

4.355 `__gnu_pbds::resize_error` Struct Reference

Inheritance diagram for `__gnu_pbds::resize_error`:



4.355.1 Detailed Description

A container cannot be resized.

Definition at line 73 of file `exception.hpp`.

The documentation for this struct was generated from the following file:

- [exception.hpp](#)

4.356 `__gnu_pbds::sample_probe_fn` Class Reference

Public Types

- typedef std::size_t **size_type**

Public Member Functions

- [sample_probe_fn](#) ()
- [sample_probe_fn](#) (const [sample_probe_fn](#) &)
- void [swap](#) ([sample_probe_fn](#) &)

Protected Member Functions

- size_type [operator\(\)](#) (key_const_reference r_key, size_type i) const

4.356.1 Detailed Description

A sample probe policy.

Definition at line 47 of file `sample_probe_fn.hpp`.

4.356.2 Constructor & Destructor Documentation

4.356.2.1 `__gnu_pbds::sample_probe_fn::sample_probe_fn ()`

Default constructor.

4.356.2.2 `__gnu_pbds::sample_probe_fn::sample_probe_fn (const sample_probe_fn &)`

Copy constructor.

4.356.3 Member Function Documentation

4.356.3.1 `size_type __gnu_pbds::sample_probe_fn::operator() (key_const_reference r_key, size_type i) const` `[inline]`,
`[protected]`

Returns the i-th offset from the hash value of some key r_key.

4.356.3.2 `void __gnu_pbds::sample_probe_fn::swap (sample_probe_fn &)` `[inline]`

Swaps content.

The documentation for this class was generated from the following file:

- [sample_probe_fn.hpp](#)

4.357 __gnu_pbds::sample_range_hashing Class Reference

Public Types

- `typedef std::size_t` [size_type](#)

Public Member Functions

- [sample_range_hashing](#) ()
- [sample_range_hashing](#) (const [sample_range_hashing](#) &other)
- `void` [swap](#) ([sample_range_hashing](#) &other)

Protected Member Functions

- `void` [notify_resized](#) ([size_type](#))
- [size_type](#) [operator\(\)](#) ([size_type](#)) const

4.357.1 Detailed Description

A sample range-hashing functor.

Definition at line 47 of file `sample_range_hashing.hpp`.

4.357.2 Member Typedef Documentation

4.357.2.1 `typedef std::size_t __gnu_pbds::sample_range_hashing::size_type`

Size type.

Definition at line 51 of file `sample_range_hashing.hpp`.

4.357.3 Constructor & Destructor Documentation

4.357.3.1 `__gnu_pbds::sample_range_hashing::sample_range_hashing ()`

Default constructor.

4.357.3.2 `__gnu_pbds::sample_range_hashing::sample_range_hashing (const sample_range_hashing & other)`

Copy constructor.

4.357.4 Member Function Documentation

4.357.4.1 `void __gnu_pbds::sample_range_hashing::notify_resized (size_type)` `[protected]`

Notifies the policy object that the container's size has changed to argument's size.

4.357.4.2 `size_type __gnu_pbds::sample_range_hashing::operator() (size_type) const` `[inline]`, `[protected]`

Transforms the `__hash` value `hash` into a ranged-hash value.

4.357.4.3 `void __gnu_pbds::sample_range_hashing::swap (sample_range_hashing & other)` `[inline]`

Swaps content.

The documentation for this class was generated from the following file:

- [sample_range_hashing.hpp](#)

4.358 `__gnu_pbds::sample_ranged_hash_fn` Class Reference

Public Types

- typedef `std::size_t` **`size_type`**

Public Member Functions

- [sample_ranged_hash_fn](#) ()
- [sample_ranged_hash_fn](#) (const [sample_ranged_hash_fn](#) &)
- void [swap](#) ([sample_ranged_hash_fn](#) &)

Protected Member Functions

- void [notify_resized](#) (*size_type*)
- *size_type* [operator\(\)](#) (*key_const_reference*) const

4.358.1 Detailed Description

A sample ranged-hash functor.

Definition at line 47 of file `sample_ranged_hash_fn.hpp`.

4.358.2 Constructor & Destructor Documentation

4.358.2.1 `__gnu_pbds::sample_ranged_hash_fn::sample_ranged_hash_fn ()`

Default constructor.

4.358.2.2 `__gnu_pbds::sample_ranged_hash_fn::sample_ranged_hash_fn (const sample_ranged_hash_fn &)`

Copy constructor.

4.358.3 Member Function Documentation

4.358.3.1 `void __gnu_pbds::sample_ranged_hash_fn::notify_resized (size_type)` `[protected]`

Notifies the policy object that the container's `__size` has changed to `size`.

4.358.3.2 `size_type __gnu_pbds::sample_ranged_hash_fn::operator() (key_const_reference) const` `[inline]`, `[protected]`

Transforms `key_const_reference` into a position within the table.

4.358.3.3 `void __gnu_pbds::sample_ranged_hash_fn::swap (sample_ranged_hash_fn &)` `[inline]`

Swaps content.

The documentation for this class was generated from the following file:

- [sample_ranged_hash_fn.hpp](#)

4.359 `__gnu_pbds::sample_ranged_probe_fn` Class Reference

Public Types

- typedef `std::size_t` **size_type**

Public Member Functions

- **sample_ranged_probe_fn** (const [sample_ranged_probe_fn](#) &)
- void **swap** ([sample_ranged_probe_fn](#) &)

Protected Member Functions

- void **notify_resized** (size_type)
- size_type **operator()** (key_const_reference, std::size_t, size_type) const

4.359.1 Detailed Description

A sample ranged-probe functor.

Definition at line 47 of file `sample_ranged_probe_fn.hpp`.

The documentation for this class was generated from the following file:

- [sample_ranged_probe_fn.hpp](#)

4.360 `__gnu_pbds::sample_resize_policy` Class Reference

Public Types

- typedef std::size_t [size_type](#)

Public Member Functions

- [sample_resize_policy](#) ()
- [sample_range_hashing](#) (const [sample_resize_policy](#) &other)
- void [swap](#) ([sample_resize_policy](#) &other)

Protected Member Functions

- [size_type](#) [get_new_size](#) ([size_type](#) size, [size_type](#) num_used_e) const
- bool [is_resize_needed](#) () const
- void [notify_cleared](#) ()
- void [notify_erase_search_collision](#) ()
- void [notify_erase_search_end](#) ()
- void [notify_erase_search_start](#) ()
- void [notify_erased](#) ([size_type](#) num_e)
- void [notify_find_search_collision](#) ()
- void [notify_find_search_end](#) ()
- void [notify_find_search_start](#) ()
- void [notify_insert_search_collision](#) ()
- void [notify_insert_search_end](#) ()
- void [notify_insert_search_start](#) ()
- void [notify_inserted](#) ([size_type](#) num_e)
- void [notify_resized](#) ([size_type](#) new_size)

4.360.1 Detailed Description

A sample resize policy.

Definition at line 47 of file sample_resize_policy.hpp.

4.360.2 Member Typedef Documentation

4.360.2.1 `typedef std::size_t __gnu_pbds::sample_resize_policy::size_type`

Size type.

Definition at line 51 of file sample_resize_policy.hpp.

4.360.3 Constructor & Destructor Documentation

4.360.3.1 `__gnu_pbds::sample_resize_policy::sample_resize_policy ()`

Default constructor.

4.360.4 Member Function Documentation

4.360.4.1 `size_type __gnu_pbds::sample_resize_policy::get_new_size (size_type size, size_type num_used_e) const` [protected]

Queries what the new size should be.

4.360.4.2 `bool __gnu_pbds::sample_resize_policy::is_resize_needed () const` [inline], [protected]

Queries whether a resize is needed.

4.360.4.3 `void __gnu_pbds::sample_resize_policy::notify_cleared ()` [protected]

Notifies the table was cleared.

4.360.4.4 `void __gnu_pbds::sample_resize_policy::notify_erase_search_collision ()` [inline], [protected]

Notifies a search encountered a collision.

4.360.4.5 `void __gnu_pbds::sample_resize_policy::notify_erase_search_end ()` [inline], [protected]

Notifies a search ended.

4.360.4.6 void __gnu_pbds::sample_resize_policy::notify_erase_search_start () [inline],[protected]

Notifies a search started.

4.360.4.7 void __gnu_pbds::sample_resize_policy::notify_erased (size_type *num_e*) [inline],[protected]

Notifies an element was erased.

4.360.4.8 void __gnu_pbds::sample_resize_policy::notify_find_search_collision () [inline],[protected]

Notifies a search encountered a collision.

4.360.4.9 void __gnu_pbds::sample_resize_policy::notify_find_search_end () [inline],[protected]

Notifies a search ended.

4.360.4.10 void __gnu_pbds::sample_resize_policy::notify_find_search_start () [inline],[protected]

Notifies a search started.

4.360.4.11 void __gnu_pbds::sample_resize_policy::notify_insert_search_collision () [inline],[protected]

Notifies a search encountered a collision.

4.360.4.12 void __gnu_pbds::sample_resize_policy::notify_insert_search_end () [inline],[protected]

Notifies a search ended.

4.360.4.13 void __gnu_pbds::sample_resize_policy::notify_insert_search_start () [inline],[protected]

Notifies a search started.

4.360.4.14 void __gnu_pbds::sample_resize_policy::notify_inserted (size_type *num_e*) [inline],[protected]

Notifies an element was inserted.

4.360.4.15 void __gnu_pbds::sample_resize_policy::notify_resized (size_type *new_size*) [protected]

Notifies the table was resized to new_size.

4.360.4.16 __gnu_pbds::sample_resize_policy::sample_range_hashing (const sample_resize_policy & *other*)

Copy constructor.

4.360.4.17 void `__gnu_pbds::sample_resize_policy::swap (sample_resize_policy & other)` [`inline`]

Swaps content.

The documentation for this class was generated from the following file:

- [sample_resize_policy.hpp](#)

4.361 `__gnu_pbds::sample_resize_trigger` Class Reference

Public Types

- typedef std::size_t [size_type](#)

Public Member Functions

- [sample_resize_trigger](#) ()
- [sample_range_hashing](#) (const [sample_resize_trigger](#) &)
- void [swap](#) ([sample_resize_trigger](#) &)

Protected Member Functions

- bool [is_grow_needed](#) ([size_type](#) size, [size_type](#) num_entries) const
- bool [is_resize_needed](#) () const
- void [notify_cleared](#) ()
- void [notify_erase_search_collision](#) ()
- void [notify_erase_search_end](#) ()
- void [notify_erase_search_start](#) ()
- void [notify_erased](#) ([size_type](#) num_entries)
- void [notify_externally_resized](#) ([size_type](#) new_size)
- void [notify_find_search_collision](#) ()
- void [notify_find_search_end](#) ()
- void [notify_find_search_start](#) ()
- void [notify_insert_search_collision](#) ()
- void [notify_insert_search_end](#) ()
- void [notify_insert_search_start](#) ()
- void [notify_inserted](#) ([size_type](#) num_entries)
- void [notify_resized](#) ([size_type](#) new_size)

4.361.1 Detailed Description

A sample resize trigger policy.

Definition at line 47 of file `sample_resize_trigger.hpp`.

4.361.2 Member Typedef Documentation

4.361.2.1 `typedef std::size_t __gnu_pbds::sample_resize_trigger::size_type`

Size type.

Definition at line 51 of file `sample_resize_trigger.hpp`.

4.361.3 Constructor & Destructor Documentation

4.361.3.1 `__gnu_pbds::sample_resize_trigger::sample_resize_trigger ()`

Default constructor.

4.361.4 Member Function Documentation

4.361.4.1 `bool __gnu_pbds::sample_resize_trigger::is_grow_needed (size_type size, size_type num_entries) const` `[inline]`, `[protected]`

Queries whether a grow is needed.

4.361.4.2 `bool __gnu_pbds::sample_resize_trigger::is_resize_needed () const` `[inline]`, `[protected]`

Queries whether a resize is needed.

4.361.4.3 `void __gnu_pbds::sample_resize_trigger::notify_cleared ()` `[protected]`

Notifies the table was cleared.

4.361.4.4 `void __gnu_pbds::sample_resize_trigger::notify_erase_search_collision ()` `[inline]`, `[protected]`

Notifies a search encountered a collision.

4.361.4.5 `void __gnu_pbds::sample_resize_trigger::notify_erase_search_end ()` `[inline]`, `[protected]`

Notifies a search ended.

4.361.4.6 `void __gnu_pbds::sample_resize_trigger::notify_erase_search_start ()` `[inline]`, `[protected]`

Notifies a search started.

4.361.4.7 `void __gnu_pbds::sample_resize_trigger::notify_erased (size_type num_entries)` `[inline]`, `[protected]`

Notifies an element was erased.

4.361.4.8 void __gnu_pbds::sample_resize_trigger::notify_externally_resized (size_type *new_size*) [protected]

Notifies the table was resized externally.

4.361.4.9 void __gnu_pbds::sample_resize_trigger::notify_find_search_collision () [inline],[protected]

Notifies a search encountered a collision.

4.361.4.10 void __gnu_pbds::sample_resize_trigger::notify_find_search_end () [inline],[protected]

Notifies a search ended.

4.361.4.11 void __gnu_pbds::sample_resize_trigger::notify_find_search_start () [inline],[protected]

Notifies a search started.

4.361.4.12 void __gnu_pbds::sample_resize_trigger::notify_insert_search_collision () [inline],[protected]

Notifies a search encountered a collision.

4.361.4.13 void __gnu_pbds::sample_resize_trigger::notify_insert_search_end () [inline],[protected]

Notifies a search ended.

4.361.4.14 void __gnu_pbds::sample_resize_trigger::notify_insert_search_start () [inline],[protected]

Notifies a search started.

4.361.4.15 void __gnu_pbds::sample_resize_trigger::notify_inserted (size_type *num_entries*) [inline],[protected]

Notifies an element was inserted. the total number of entries in the table is num_entries.

4.361.4.16 void __gnu_pbds::sample_resize_trigger::notify_resized (size_type *new_size*) [protected]

Notifies the table was resized as a result of this object's signifying that a resize is needed.

4.361.4.17 __gnu_pbds::sample_resize_trigger::sample_range_hashing (const sample_resize_trigger &)

Copy constructor.

4.361.4.18 void __gnu_pbds::sample_resize_trigger::swap (sample_resize_trigger &) [inline]

Swaps content.

The documentation for this class was generated from the following file:

- [sample_resize_trigger.hpp](#)

4.362 `__gnu_pbds::sample_size_policy` Class Reference

Public Types

- typedef std::size_t [size_type](#)

Public Member Functions

- [sample_size_policy](#) ()
- [sample_range_hashing](#) (const [sample_size_policy](#) &)
- void [swap](#) ([sample_size_policy](#) &other)

Protected Member Functions

- [size_type get_nearest_larger_size](#) ([size_type](#) size) const
- [size_type get_nearest_smaller_size](#) ([size_type](#) size) const

4.362.1 Detailed Description

A sample size policy.

Definition at line 47 of file `sample_size_policy.hpp`.

4.362.2 Member Typedef Documentation

4.362.2.1 typedef std::size_t `__gnu_pbds::sample_size_policy::size_type`

Size type.

Definition at line 51 of file `sample_size_policy.hpp`.

4.362.3 Constructor & Destructor Documentation

4.362.3.1 `__gnu_pbds::sample_size_policy::sample_size_policy ()`

Default constructor.

4.362.4 Member Function Documentation

4.362.4.1 `size_type __gnu_pbds::sample_size_policy::get_nearest_larger_size (size_type size) const` `[inline]`, `[protected]`

Given a `__size` size, returns a `__size` that is larger.

4.362.4.2 **size_type __gnu_pbds::sample_size_policy::get_nearest_smaller_size (size_type size) const** [inline],
[protected]

Given a __size size, returns a __size that is smaller.

4.362.4.3 **__gnu_pbds::sample_size_policy::sample_range_hashing (const sample_size_policy &)**

Copy constructor.

4.362.4.4 **void __gnu_pbds::sample_size_policy::swap (sample_size_policy & other)** [inline]

Swaps content.

The documentation for this class was generated from the following file:

- [sample_size_policy.hpp](#)

4.363 **__gnu_pbds::sample_tree_node_update< Const_Node_Iter, Node_Iter, Cmp_Fn, _Alloc > Class Template Reference**

4.363.1 Detailed Description

```
template<typename Const_Node_Iter, typename Node_Iter, typename Cmp_Fn, typename _Alloc>
class __gnu_pbds::sample_tree_node_update< Const_Node_Iter, Node_Iter, Cmp_Fn, _Alloc >
```

A sample node updator.

Definition at line 49 of file sample_tree_node_update.hpp.

The documentation for this class was generated from the following file:

- [sample_tree_node_update.hpp](#)

4.364 **__gnu_pbds::sample_trie_access_traits Struct Reference**

Public Types

- enum { **max_size** }
- typedef __Alloc::template rebind< [key_type](#) > **__rebind_k**
- typedef std::string::const_iterator **const_iterator**
- typedef char [e_type](#)
- typedef __rebind_k::other::const_reference **key_const_reference**
- typedef [std::string](#) **key_type**
- typedef std::size_t **size_type**

Static Public Member Functions

- static const_iterator [begin](#) (key_const_reference)
- static size_type [e_pos](#) (e_type)
- static const_iterator [end](#) (key_const_reference)

4.364.1 Detailed Description

A sample trie element access traits.

Definition at line 47 of file `sample_trie_access_traits.hpp`.

4.364.2 Member Typedef Documentation

4.364.2.1 `typedef char __gnu_pbds::sample_trie_access_traits::e_type`

Element type.

Definition at line 57 of file `sample_trie_access_traits.hpp`.

4.364.3 Member Function Documentation

4.364.3.1 `static const_iterator __gnu_pbds::sample_trie_access_traits::begin (key_const_reference) [inline], [static]`

Returns a const_iterator to the first element of r_key.

4.364.3.2 `static size_type __gnu_pbds::sample_trie_access_traits::e_pos (e_type) [inline], [static]`

Maps an element to a position.

4.364.3.3 `static const_iterator __gnu_pbds::sample_trie_access_traits::end (key_const_reference) [inline], [static]`

Returns a const_iterator to the after-last element of r_key.

The documentation for this struct was generated from the following file:

- [sample_trie_access_traits.hpp](#)

4.365 `__gnu_pbds::sample_trie_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >` Class Template Reference

Public Types

- `typedef std::size_t metadata_type`

Protected Member Functions

- [sample_trie_node_update](#) ()
- void [operator](#)() (node_iterator, node_const_iterator) const

4.365.1 Detailed Description

```
template<typename Node_Cltr, typename Node_Itr, typename _ATraits, typename _Alloc>
class __gnu_pbds::sample_trie_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >
```

A sample node updator.

Definition at line 49 of file sample_trie_node_update.hpp.

4.365.2 Constructor & Destructor Documentation

```
4.365.2.1 template<typename Node_Cltr , typename Node_Itr , typename _ATraits , typename _Alloc >
        __gnu_pbds::sample_trie_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::sample_trie_node_update
        ( ) [protected]
```

Default constructor.

4.365.3 Member Function Documentation

```
4.365.3.1 template<typename Node_Cltr , typename Node_Itr , typename _ATraits , typename _Alloc > void
        __gnu_pbds::sample_trie_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::operator() ( node_iterator ,
        node_const_iterator ) const [inline], [protected]
```

Updates the rank of a node through a node_iterator node_it; end_nd_it is the end node iterator.

The documentation for this class was generated from the following file:

- [sample_trie_node_update.hpp](#)

4.366 __gnu_pbds::sample_update_policy Struct Reference

Public Member Functions

- [sample_update_policy](#) ()
- [sample_update_policy](#) (const [sample_update_policy](#) &)
- void [swap](#) ([sample_update_policy](#) &other)

Protected Types

- typedef some_metadata_type [metadata_type](#)

Protected Member Functions

- `metadata_type operator() () const`
- `bool operator() (metadata_reference) const`

4.366.1 Detailed Description

A sample list-update policy.

Definition at line 47 of file `sample_update_policy.hpp`.

4.366.2 Member Typedef Documentation

4.366.2.1 `typedef some_metadata_type __gnu_pbds::sample_update_policy::metadata_type` `[protected]`

Metadata on which this functor operates.

Definition at line 61 of file `sample_update_policy.hpp`.

4.366.3 Constructor & Destructor Documentation

4.366.3.1 `__gnu_pbds::sample_update_policy::sample_update_policy ()`

Default constructor.

4.366.3.2 `__gnu_pbds::sample_update_policy::sample_update_policy (const sample_update_policy &)`

Copy constructor.

4.366.4 Member Function Documentation

4.366.4.1 `metadata_type __gnu_pbds::sample_update_policy::operator() () const` `[protected]`

Creates a metadata object.

4.366.4.2 `bool __gnu_pbds::sample_update_policy::operator() (metadata_reference) const` `[protected]`

Decides whether a metadata object should be moved to the front of the list. A list-update based containers object will call this method to decide whether to move a node to the front of the list. The method should return true if the node should be moved to the front of the list.

4.366.4.3 void __gnu_pbds::sample_update_policy::swap (sample_update_policy & *other*) [inline]

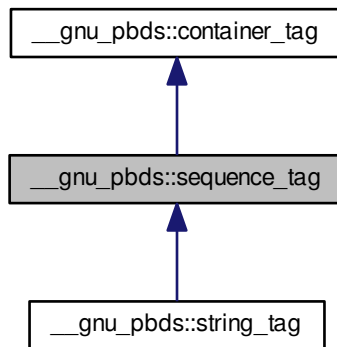
Swaps content.

The documentation for this struct was generated from the following file:

- [sample_update_policy.hpp](#)

4.367 __gnu_pbds::sequence_tag Struct Reference

Inheritance diagram for __gnu_pbds::sequence_tag:



4.367.1 Detailed Description

Basic sequence.

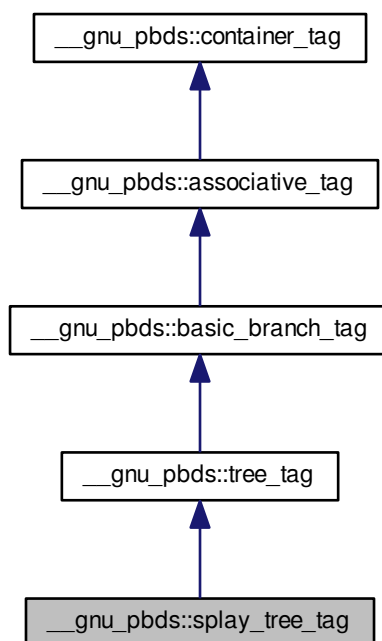
Definition at line 129 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

4.368 __gnu_pbds::splay_tree_tag Struct Reference

Inheritance diagram for __gnu_pbds::splay_tree_tag:



4.368.1 Detailed Description

Splay tree.

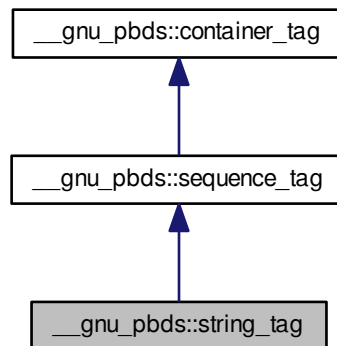
Definition at line 156 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

4.369 `__gnu_pbds::string_tag` Struct Reference

Inheritance diagram for `__gnu_pbds::string_tag`:



4.369.1 Detailed Description

Basic string container, inclusive of strings, ropes, etc.

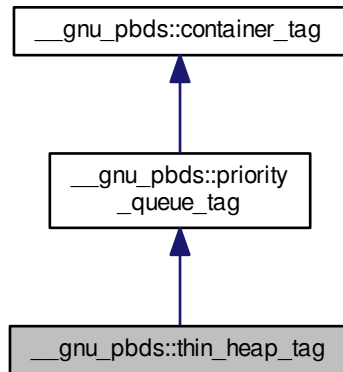
Definition at line 132 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

4.370 `__gnu_pbds::thin_heap_tag` Struct Reference

Inheritance diagram for `__gnu_pbds::thin_heap_tag`:



4.370.1 Detailed Description

Thin heap.

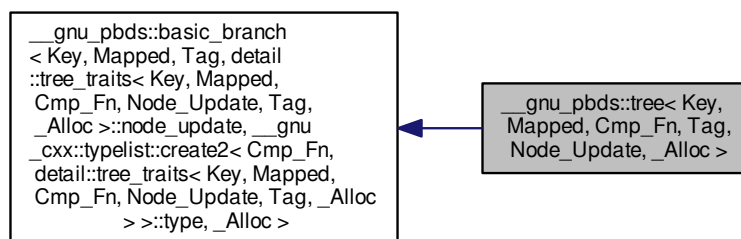
Definition at line 186 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

4.371 `__gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc >`:



Public Types

- typedef Cmp_Fn [cmp_fn](#)
- typedef [detail::tree_traits](#)< Key, Mapped, Cmp_Fn, Node_Update, Tag, _Alloc >::node_update **node_update**

Public Member Functions

- [tree](#) (const [cmp_fn](#) &c)
- template<typename It >
[tree](#) (It first, It last)
- template<typename It >
[tree](#) (It first, It last, const [cmp_fn](#) &c)
- **tree** (const [tree](#) &other)
- [tree](#) & **operator=** (const [tree](#) &other)
- void **swap** ([tree](#) &other)

4.371.1 Detailed Description

```
template<typename Key, typename Mapped, typename Cmp_Fn = std::less<Key>, typename Tag = rb_tree_tag, template< typename
Node_Cltr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update = null_node_update, typename _Alloc =
std::allocator<char>>>
class __gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc >
```

A tree-based container.

Template Parameters

<i>Key</i>	Key type.
<i>Mapped</i>	Map type.
<i>Cmp_Fn</i>	Comparison functor.
<i>Tag</i>	Instantiating data structure type, see container_tag.
<i>Node_Update</i>	Updates tree internal-nodes, restores invariants when invalidated. XXX See design::tree-based-containersnode invariants.
<i>_Alloc</i>	Allocator type.

Base tag choices are: [ov_tree_tag](#), [rb_tree_tag](#), [splay_tree_tag](#).

Base is [basic_branch](#).

Definition at line 635 of file [assoc_container.hpp](#).

4.371.2 Member Typedef Documentation

4.371.2.1 `template<typename Key , typename Mapped , typename Cmp_Fn = std::less<Key>, typename Tag = rb_tree_tag, template< typename Node_Cltr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update = null_node_update, typename _Alloc = std::allocator<char>> > typedef Cmp_Fn __gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc >::cmp_fn`

Comparison functor type.

Definition at line 642 of file `assoc_container.hpp`.

4.371.3 Constructor & Destructor Documentation

4.371.3.1 `template<typename Key , typename Mapped , typename Cmp_Fn = std::less<Key>, typename Tag = rb_tree_tag, template< typename Node_Cltr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update = null_node_update, typename _Alloc = std::allocator<char>> > __gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc >::tree (const cmp_fn & c) [inline]`

Constructor taking some policy objects. `r_cmp_fn` will be copied by the `Cmp_Fn` object of the container object.

Definition at line 648 of file `assoc_container.hpp`.

4.371.3.2 `template<typename Key , typename Mapped , typename Cmp_Fn = std::less<Key>, typename Tag = rb_tree_tag, template< typename Node_Cltr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update = null_node_update, typename _Alloc = std::allocator<char>> > template<typename It > __gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc >::tree (It first, It last) [inline]`

Constructor taking `__iterators` to a range of `value_types`. The `value_types` between `first_it` and `last_it` will be inserted into the container object.

Definition at line 655 of file `assoc_container.hpp`.

4.371.3.3 `template<typename Key , typename Mapped , typename Cmp_Fn = std::less<Key>, typename Tag = rb_tree_tag, template< typename Node_Cltr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update = null_node_update, typename _Alloc = std::allocator<char>> > template<typename It > __gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc >::tree (It first, It last, const cmp_fn & c) [inline]`

Constructor taking `__iterators` to a range of `value_types` and some policy objects The `value_types` between `first_it` and `last_it` will be inserted into the container object. `r_cmp_fn` will be copied by the `cmp_fn` object of the container object.

Definition at line 663 of file `assoc_container.hpp`.

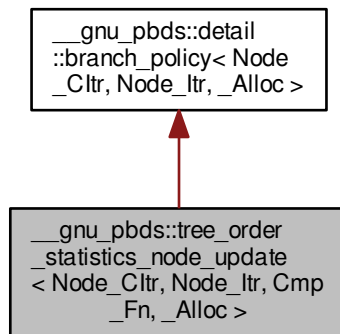
References `std::swap()`.

The documentation for this class was generated from the following file:

- [assoc_container.hpp](#)

4.372 `__gnu_pbds::tree_order_statistics_node_update< Node_Cltr, Node_Itr, Cmp_Fn, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::tree_order_statistics_node_update< Node_Cltr, Node_Itr, Cmp_Fn, _Alloc >`:



Public Types

- typedef `_Alloc` **allocator_type**
- typedef `Cmp_Fn` **cmp_fn**
- typedef `node_const_iterator::value_type` **const_iterator**
- typedef `node_iterator::value_type` **iterator**
- typedef `base_type::key_const_reference` **key_const_reference**
- typedef `base_type::key_type` **key_type**
- typedef `size_type` **metadata_type**
- typedef `Node_Cltr` **node_const_iterator**
- typedef `Node_Itr` **node_iterator**
- typedef `allocator_type::size_type` **size_type**

Public Member Functions

- `const_iterator` [find_by_order](#) (`size_type`) const
- `iterator` [find_by_order](#) (`size_type`)
- `size_type` [order_of_key](#) (`key_const_reference`) const

Protected Member Functions

- void [operator\(\)](#) (`node_iterator`, `node_const_iterator`) const

Private Types

- typedef Node_Itr::value_type **it_type**
- typedef remove_const< key_type >::type **rckey_type**
- typedef remove_const< value_type >::type **rcvalue_type**
- typedef _Alloc::template rebind< rckey_type >::other **rebind_k**
- typedef _Alloc::template rebind< rcvalue_type >::other **rebind_v**
- typedef rebind_v::reference **reference**
- typedef std::iterator_traits< it_type >::value_type **value_type**

Private Member Functions

- virtual it_type **end** ()=0
- it_type **end_iterator** () const

Static Private Member Functions

- static key_const_reference **extract_key** (const_reference r_val)

4.372.1 Detailed Description

```
template<typename Node_Cltr, typename Node_Itr, typename Cmp_Fn, typename _Alloc>
class __gnu_pbds::tree_order_statistics_node_update< Node_Cltr, Node_Itr, Cmp_Fn, _Alloc >
```

Functor updating ranks of entrees.

Definition at line 64 of file tree_policy.hpp.

4.372.2 Member Function Documentation

```
4.372.2.1 template<typename Node_Cltr , typename Node_Itr , typename Cmp_Fn , typename _Alloc >
tree_order_statistics_node_update< Node_Cltr, Node_Itr, Cmp_Fn, _Alloc >::const_iterator
__gnu_pbds::tree_order_statistics_node_update< Node_Cltr, Node_Itr, Cmp_Fn, _Alloc >::find_by_order (
size_type order ) const [inline]
```

Finds an entry by `__order`. Returns a `const_iterator` to the entry with the `__order` order, or a `const_iterator` to the container object's end if order is at least the size of the container object.

Definition at line 72 of file tree_policy.hpp.

```
4.372.2.2 template<typename Node_Cltr , typename Node_Itr , typename Cmp_Fn , typename _Alloc >
tree_order_statistics_node_update< Node_Cltr, Node_Itr, Cmp_Fn, _Alloc >::iterator
__gnu_pbds::tree_order_statistics_node_update< Node_Cltr, Node_Itr, Cmp_Fn, _Alloc >::find_by_order (
size_type order ) [inline]
```

Finds an entry by `__order`. Returns an `iterator` to the entry with the `__order` order, or an `iterator` to the container object's end if order is at least the size of the container object.

Definition at line 45 of file tree_policy.hpp.

```
4.372.2.3  template<typename Node_Cltr , typename Node_Itr , typename Cmp_Fn , typename _Alloc > void
            __gnu_pbds::tree_order_statistics_node_update< Node_Cltr, Node_Itr, Cmp_Fn, _Alloc >::operator() (
            node_iterator node_it, node_const_iterator end_nd_it ) const    [inline], [protected]
```

Updates the rank of a node through a node_iterator node_it; end_nd_it is the end node iterator.

Definition at line 108 of file tree_policy.hpp.

```
4.372.2.4  template<typename Node_Cltr , typename Node_Itr , typename Cmp_Fn , typename _Alloc >
            tree_order_statistics_node_update< Node_Cltr, Node_Itr, Cmp_Fn, _Alloc >::size_type
            __gnu_pbds::tree_order_statistics_node_update< Node_Cltr, Node_Itr, Cmp_Fn, _Alloc >::order_of_key (
            key_const_reference r_key ) const    [inline]
```

Returns the order of a key within a sequence. For exapmle, if r_key is the smallest key, this method will return 0; if r_key is a key between the smallest and next key, this method will return 1; if r_key is a key larger than the largest key, this method will return the size of r_c.

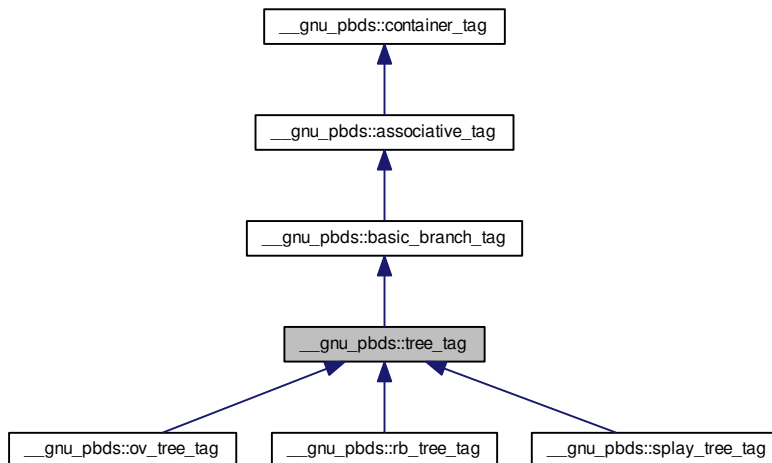
Definition at line 78 of file tree_policy.hpp.

The documentation for this class was generated from the following file:

- [tree_policy.hpp](#)

4.373 __gnu_pbds::tree_tag Struct Reference

Inheritance diagram for __gnu_pbds::tree_tag:



4.373.1 Detailed Description

Basic tree structure.

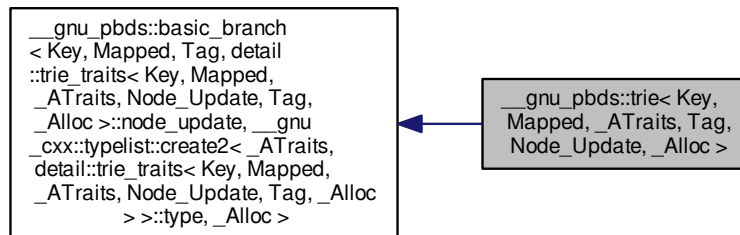
Definition at line 150 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

4.374 `__gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc >`:



Public Types

- typedef `_ATraits` [access_traits](#)
- typedef [detail::trie_traits< Key, Mapped, _ATraits, Node_Update, Tag, _Alloc >::node_update](#) **node_update**

Public Member Functions

- [trie](#) (const [access_traits](#) &t)
- template<typename It >
[trie](#) (It first, It last)
- template<typename It >
[trie](#) (It first, It last, const [access_traits](#) &t)
- **trie** (const [trie](#) &other)
- [trie](#) & **operator=** (const [trie](#) &other)
- void **swap** ([trie](#) &other)

4.374.1 Detailed Description

```

template<typename Key, typename Mapped, typename _ATraits = typename detail::default_trie_access_traits<Key>::type, typename
Tag = pat_trie_tag, template< typename Node_Citr, typename Node_Itr, typename _ATraits_, typename _Alloc_ > class Node_Update
= null_node_update, typename _Alloc = std::allocator<char>>>
class __gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc >
  
```

A trie-based container.

Template Parameters

<i>Key</i>	Key type.
<i>Mapped</i>	Map type.
<i>_ATraits</i>	Element access traits.
<i>Tag</i>	Instantiating data structure type, see container_tag.
<i>Node_Update</i>	Updates trie internal-nodes, restores invariants when invalidated. XXX See design::tree-based-containersnode invariants.
<i>_Alloc</i>	Allocator type.

Base tag choice is pat_trie_tag.

Base is basic_branch.

Definition at line 731 of file assoc_container.hpp.

4.374.2 Member Typedef Documentation

4.374.2.1 `template<typename Key , typename Mapped , typename _ATraits = typename detail::default_trie_access_traits<↵
Key>::type, typename Tag = pat_trie_tag, template< typename Node_Cltr, typename Node_Itr, typename _ATraits_,
typename _Alloc_ > class Node_Update = null_node_update, typename _Alloc = std::allocator<char>>> typedef _ATraits
__gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc >::access_traits`

Element access traits type.

Definition at line 738 of file assoc_container.hpp.

4.374.3 Constructor & Destructor Documentation

4.374.3.1 `template<typename Key , typename Mapped , typename _ATraits = typename detail::default_trie_access_traits<Key>↵
::type, typename Tag = pat_trie_tag, template< typename Node_Cltr, typename Node_Itr, typename _ATraits_, typename
Alloc > class Node_Update = null_node_update, typename _Alloc = std::allocator<char>>> __gnu_pbds::trie<
Key, Mapped, _ATraits, Tag, Node_Update, _Alloc >::trie (const access_traits & t) [inline]`

Constructor taking some policy objects. r_access_traits will be copied by the _ATraits object of the container object.

Definition at line 744 of file assoc_container.hpp.

4.374.3.2 `template<typename Key , typename Mapped , typename _ATraits = typename detail::default_trie_access_traits<Key>↵
::type, typename Tag = pat_trie_tag, template< typename Node_Cltr, typename Node_Itr, typename _ATraits_, typename
Alloc > class Node_Update = null_node_update, typename _Alloc = std::allocator<char>>> template<typename It >
__gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc >::trie (It first, It last) [inline]`

Constructor taking __iterators to a range of value_types. The value_types between first_it and last_it will be inserted into the container object.

Definition at line 751 of file assoc_container.hpp.

4.374.3.3 `template<typename Key , typename Mapped , typename _ATraits = typename detail::default_trie_access_traits<Key>::type, typename Tag = pat_trie_tag, template< typename Node_Cltr, typename Node_Itr, typename _ATraits_, typename _Alloc_ > class Node_Update = null_node_update, typename _Alloc = std::allocator<char>>> template<typename It > __gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc >::trie (It first, It last, const access_traits & t) [inline]`

Constructor taking `__iterators` to a range of `value_types` and some policy objects. The `value_types` between `first_it` and `last_it` will be inserted into the container object.

Definition at line 758 of file `assoc_container.hpp`.

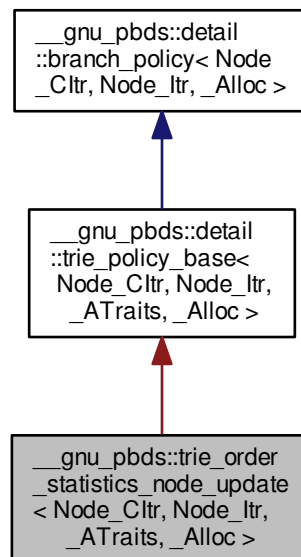
References `std::swap()`.

The documentation for this class was generated from the following file:

- [assoc_container.hpp](#)

4.375 `__gnu_pbds::trie_order_statistics_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc > Class` Template Reference

Inheritance diagram for `__gnu_pbds::trie_order_statistics_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >`:



Public Types

- typedef access_traits::const_iterator **a_const_iterator**
- typedef _ATraits **access_traits**
- typedef _Alloc **allocator_type**
- typedef node_const_iterator::value_type **const_iterator**
- typedef node_iterator::value_type **iterator**
- typedef base_type::key_const_reference **key_const_reference**
- typedef base_type::key_type **key_type**
- typedef size_type **metadata_type**
- typedef Node_Cltr **node_const_iterator**
- typedef Node_Itr **node_iterator**
- typedef allocator_type::size_type **size_type**

Public Member Functions

- const_iterator [find_by_order](#) (size_type) const
- iterator [find_by_order](#) (size_type)
- size_type [order_of_key](#) (key_const_reference) const
- size_type [order_of_prefix](#) (a_const_iterator, a_const_iterator) const

Protected Member Functions

- void [operator\(\)](#) (node_iterator, node_const_iterator) const

Private Types

- typedef Node_Itr::value_type **it_type**
- typedef remove_const< key_type >::type **rkey_type**
- typedef remove_const< value_type >::type **rcvalue_type**
- typedef _Alloc::template rebind< rkey_type >::other **rebind_k**
- typedef _Alloc::template rebind< rcvalue_type >::other **rebind_v**
- typedef rebind_v::reference **reference**
- typedef std::iterator_traits< it_type >::value_type **value_type**

Private Member Functions

- virtual const_iterator **end** () const =0
- it_type **end_iterator** () const
- virtual const access_traits & **get_access_traits** () const =0

Static Private Member Functions

- static size_type **common_prefix_len** (node_iterator, e_const_iterator, e_const_iterator, const access_traits &)
- static key_const_reference **extract_key** (const_reference r_val)
- static iterator **leftmost_it** (node_iterator)
- static bool **less** (e_const_iterator, e_const_iterator, e_const_iterator, e_const_iterator, const access_traits &)
- static iterator **rightmost_it** (node_iterator)

4.375.1 Detailed Description

```
template<typename Node_Cltr, typename Node_Itr, typename _ATraits, typename _Alloc>
class __gnu_pbds::trie_order_statistics_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >
```

Functor updating ranks of entrees.

Definition at line 253 of file `trie_policy.hpp`.

4.375.2 Member Function Documentation

```
4.375.2.1 template<typename Node_Cltr , typename Node_Itr , typename _ATraits , typename _Alloc >
    trie_order_statistics_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::const_iterator
    __gnu_pbds::trie_order_statistics_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::find_by_order (
        size_type order ) const    [inline]
```

Finds an entry by `__order`. Returns a `const_iterator` to the entry with the `__order` order, or a `const_iterator` to the container object's end if order is at least the size of the container object.

Definition at line 79 of file `trie_policy.hpp`.

```
4.375.2.2 template<typename Node_Cltr , typename Node_Itr , typename _ATraits , typename _Alloc
    > trie_order_statistics_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::iterator
    __gnu_pbds::trie_order_statistics_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::find_by_order (
        size_type order )    [inline]
```

Finds an entry by `__order`. Returns an iterator to the entry with the `__order` order, or an iterator to the container object's end if order is at least the size of the container object.

Definition at line 45 of file `trie_policy.hpp`.

```
4.375.2.3 template<typename Node_Cltr , typename Node_Itr , typename _ATraits , typename _Alloc > void
    __gnu_pbds::trie_order_statistics_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::operator() (
        node_iterator nd_it, node_const_iterator ) const    [inline], [protected]
```

Updates the rank of a node through a `node_iterator` `node_it`; `end_nd_it` is the end node iterator.

Definition at line 152 of file `trie_policy.hpp`.

```
4.375.2.4 template<typename Node_Cltr , typename Node_Itr , typename _ATraits , typename _Alloc >
    trie_order_statistics_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::size_type
    __gnu_pbds::trie_order_statistics_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::order_of_key (
        key_const_reference r_key ) const    [inline]
```

Returns the order of a key within a sequence. For exapmle, if `r_key` is the smallest key, this method will return 0; if `r_key` is a key between the smallest and next key, this method will return 1; if `r_key` is a key larger than the largest key, this method will return the size of `r_c`.

Definition at line 85 of file `trie_policy.hpp`.

```

4.375.2.5 template<typename Node_Cltr , typename Node_Itr , typename _ATraits , typename _Alloc >
    trie_order_statistics_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::size_type
    __gnu_pbds::trie_order_statistics_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::order_of_prefix (
        a_const_iterator b, a_const_iterator e ) const    [inline]

```

Returns the order of a prefix within a sequence. For example, if [b, e] is the smallest prefix, this method will return 0; if r_key is a key between the smallest and next key, this method will return 1; if r_key is a key larger than the largest key, this method will return the size of r_c.

Definition at line 96 of file trie_policy.hpp.

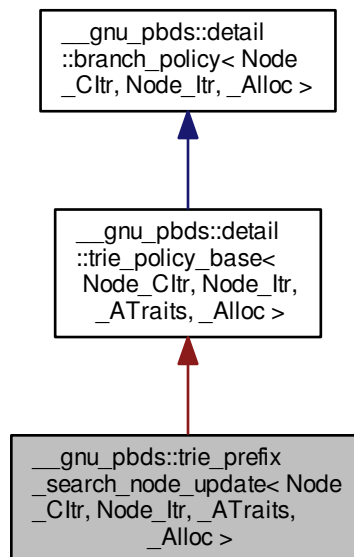
References `__gnu_pbds::trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >::begin()`, `__gnu_pbds::trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >::e_pos()`, and `__gnu_pbds::trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >::end()`.

The documentation for this class was generated from the following file:

- [trie_policy.hpp](#)

4.376 `__gnu_pbds::trie_prefix_search_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::trie_prefix_search_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >`:



Public Types

- typedef `access_traits::const_iterator` [a_const_iterator](#)
- typedef `_ATraits` [access_traits](#)
- typedef `_Alloc` [allocator_type](#)
- typedef `node_const_iterator::value_type` **const_iterator**
- typedef `node_iterator::value_type` **iterator**
- typedef `base_type::key_const_reference` **key_const_reference**
- typedef `base_type::key_type` **key_type**
- typedef [null_type](#) **metadata_type**
- typedef `Node_Cltr` **node_const_iterator**
- typedef `Node_Itr` **node_iterator**
- typedef `allocator_type::size_type` [size_type](#)

Public Member Functions

- `std::pair< const_iterator, const_iterator >` [prefix_range](#) (`key_const_reference`) const
- `std::pair< iterator, iterator >` [prefix_range](#) (`key_const_reference`)
- `std::pair< const_iterator, const_iterator >` [prefix_range](#) ([a_const_iterator](#), [a_const_iterator](#)) const
- `std::pair< iterator, iterator >` [prefix_range](#) ([a_const_iterator](#), [a_const_iterator](#))

Protected Member Functions

- void [operator\(\)](#) (`node_iterator node_it`, `node_const_iterator end_nd_it`) const

Private Types

- typedef `rebind_v::const_pointer` **const_pointer**
- typedef `rebind_v::const_reference` **const_reference**
- typedef `Node_Itr::value_type` **it_type**
- typedef `remove_const< key_type >::type` **rckey_type**
- typedef `remove_const< value_type >::type` **rcvalue_type**
- typedef `_Alloc::template rebind< rckey_type >::other` **rebind_k**
- typedef `_Alloc::template rebind< rcvalue_type >::other` **rebind_v**
- typedef `rebind_v::reference` **reference**
- typedef `std::iterator_traits< it_type >::value_type` **value_type**

Private Member Functions

- `it_type` **end_iterator** () const

Static Private Member Functions

- static [size_type](#) **common_prefix_len** (`node_iterator`, `e_const_iterator`, `e_const_iterator`, const [access_traits](#) &)
- static `key_const_reference` **extract_key** (`const_reference r_val`)
- static `iterator` **leftmost_it** (`node_iterator`)
- static bool **less** (`e_const_iterator`, `e_const_iterator`, `e_const_iterator`, `e_const_iterator`, const [access_traits](#) &)
- static `iterator` **rightmost_it** (`node_iterator`)

4.376.1 Detailed Description

```
template<typename Node_Cltr, typename Node_Itr, typename _ATraits, typename _Alloc>
class __gnu_pbds::trie_prefix_search_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >
```

A node updatator that allows tries to be searched for the range of values that match a certain prefix.

Definition at line 155 of file trie_policy.hpp.

4.376.2 Member Typedef Documentation

```
4.376.2.1 template<typename Node_Cltr, typename Node_Itr, typename _ATraits, typename _Alloc> typedef
access_traits::const_iterator __gnu_pbds::trie_prefix_search_node_update< Node_Cltr, Node_Itr, _ATraits,
_Alloc >::a_const_iterator
```

Const element iterator.

Definition at line 168 of file trie_policy.hpp.

```
4.376.2.2 template<typename Node_Cltr, typename Node_Itr, typename _ATraits, typename _Alloc> typedef _ATraits
__gnu_pbds::trie_prefix_search_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::access_traits
```

Element access traits.

Definition at line 165 of file trie_policy.hpp.

```
4.376.2.3 template<typename Node_Cltr, typename Node_Itr, typename _ATraits, typename _Alloc> typedef _Alloc
__gnu_pbds::trie_prefix_search_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::allocator_type
```

_Alloc type.

Definition at line 171 of file trie_policy.hpp.

```
4.376.2.4 template<typename Node_Cltr, typename Node_Itr, typename _ATraits, typename _Alloc> typedef
allocator_type::size_type __gnu_pbds::trie_prefix_search_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc
>::size_type
```

Size type.

Definition at line 174 of file trie_policy.hpp.

4.376.3 Member Function Documentation

```
4.376.3.1 template<typename Node_Cltr , typename Node_Itr , typename _ATraits , typename _Alloc > void
__gnu_pbds::trie_prefix_search_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::operator() (
node_iterator node_it, node_const_iterator end_nd_it ) const [inline], [protected]
```

Called to update a node's metadata.

Definition at line 139 of file trie_policy.hpp.

4.376.3.2 `template<typename Node_Cltr , typename Node_Itr , typename _ATraits , typename _Alloc > std::pair<typename trie_prefix_search_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::const_iterator,typename trie_prefix_search_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::const_iterator >__gnu_pbds::trie_prefix_search_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::prefix_range (key_const_reference r_key) const`

Finds the const iterator range corresponding to all values whose prefixes match `r_key`.

Definition at line 47 of file `trie_policy.hpp`.

4.376.3.3 `template<typename Node_Cltr , typename Node_Itr , typename _ATraits , typename _Alloc > std::pair<typename trie_prefix_search_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::iterator,typename trie_prefix_search_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::iterator >__gnu_pbds::trie_prefix_search_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::prefix_range (key_const_reference r_key)`

Finds the iterator range corresponding to all values whose prefixes match `r_key`.

Definition at line 58 of file `trie_policy.hpp`.

4.376.3.4 `template<typename Node_Cltr , typename Node_Itr , typename _ATraits , typename _Alloc > std::pair<typename trie_prefix_search_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::const_iterator,typename trie_prefix_search_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::const_iterator >__gnu_pbds::trie_prefix_search_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::prefix_range (a_const_iterator b, a_const_iterator e) const`

Finds the const iterator range corresponding to all values whose prefixes match `[b, e)`.

Definition at line 69 of file `trie_policy.hpp`.

4.376.3.5 `template<typename Node_Cltr , typename Node_Itr , typename _ATraits , typename _Alloc > std::pair<typename trie_prefix_search_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::iterator,typename trie_prefix_search_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::iterator >__gnu_pbds::trie_prefix_search_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::prefix_range (a_const_iterator b, a_const_iterator e)`

Finds the iterator range corresponding to all values whose prefixes match `[b, e)`.

Definition at line 84 of file `trie_policy.hpp`.

The documentation for this class was generated from the following file:

- [trie_policy.hpp](#)

4.377 `__gnu_pbds::trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >` Struct Template Reference

Public Types

- enum { **reverse** }
- enum { **min_e_val**, **max_e_val**, **max_size** }
- typedef `_Alloc::template rebind< key_type > __rebind_k`
- typedef `detail::__conditional_type< Reverse, typename String::const_reverse_iterator, typename String::const_iterator >::__type const_iterator`
- typedef `std::iterator_traits< const_iterator >::value_type e_type`
- typedef `__rebind_k::other::const_reference key_const_reference`
- typedef `String key_type`
- typedef `_Alloc::size_type size_type`

Static Public Member Functions

- static `const_iterator begin` (key_const_reference)
- static `size_type e_pos` (e_type e)
- static `const_iterator end` (key_const_reference)

4.377.1 Detailed Description

```
template<typename String = std::string, typename String::value_type Min_E_Val = detail::__numeric_traits<typename String::value_type>::__min,
typename String::value_type Max_E_Val = detail::__numeric_traits<typename String::value_type>::__max, bool Reverse = false,
typename _Alloc = std::allocator<char>>>
struct __gnu_pbds::trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >
```

Element access traits for string types.

Template Parameters

<i>String</i>	String type.
<i>Min_E_Val</i>	Minimal element value.
<i>Max_E_Val</i>	Maximum element value.
<i>Reverse</i>	Reverse iteration should be used. Default: false.
<i>_Alloc</i>	Allocator type.

Definition at line 74 of file `trie_policy.hpp`.

4.377.2 Member Typedef Documentation

4.377.2.1 `template<typename String = std::string, typename String::value_type Min_E_Val = detail::__numeric_traits<typename String::value_type>::__min, typename String::value_type Max_E_Val = detail::__numeric_traits<typename String::value_type>::__max, bool Reverse = false, typename _Alloc = std::allocator<char>> typedef detail::__conditional_type<Reverse, typename String::const_reverse_iterator, typename String::const_iterator>::__type __gnu_pbds::trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >::const_iterator`

Element const iterator type.

Definition at line 90 of file `trie_policy.hpp`.

4.377.2.2 `template<typename String = std::string, typename String::value_type Min_E_Val = detail::__numeric_traits<typename String::value_type>::__min, typename String::value_type Max_E_Val = detail::__numeric_traits<typename String::value_type>::__max, bool Reverse = false, typename _Alloc = std::allocator<char>> typedef std::iterator_traits<const_iterator>::value_type __gnu_pbds::trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >::e_type`

Element type.

Definition at line 93 of file `trie_policy.hpp`.

4.377.3 Member Function Documentation

4.377.3.1 `template<typename String , typename String::value_type Min_E_Val, typename String::value_type Max_E_Val, bool Reverse, typename _Alloc > trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >::const_iterator __gnu_pbds::trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >::begin (key_const_reference r_key) [inline], [static]`

Returns a `const_iterator` to the first element of `key_const_reference` agumnet.

Definition at line 57 of file `trie_policy.hpp`.

Referenced by `__gnu_pbds::trie_order_statistics_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::order_of_← prefix()`.

4.377.3.2 `template<typename String , typename String::value_type Min_E_Val, typename String::value_type Max_E_Val, bool Reverse, typename _Alloc > trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >::size_type __gnu_pbds::trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >::e_pos (e_type e) [inline], [static]`

Maps an element to a position.

Definition at line 49 of file `trie_policy.hpp`.

Referenced by `__gnu_pbds::trie_order_statistics_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::order_of_← prefix()`.

```

4.377.3.3  template<typename String , typename String::value_type Min_E_Val, typename String::value_type Max_E_Val,
              bool Reverse, typename _Alloc > trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc
              >::const_iterator __gnu_pbds::trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc
              >::end ( key_const_reference r_key )  [inline],[static]

```

Returns a const_iterator to the after-last element of key_const_reference argument.

Definition at line 65 of file trie_policy.hpp.

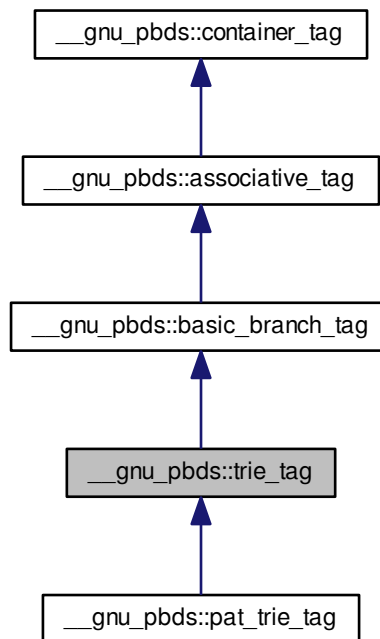
Referenced by __gnu_pbds::trie_order_statistics_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::order_of_↵
prefix().

The documentation for this struct was generated from the following file:

- [trie_policy.hpp](#)

4.378 __gnu_pbds::trie_tag Struct Reference

Inheritance diagram for __gnu_pbds::trie_tag:



4.378.1 Detailed Description

Basic trie structure.

Definition at line 162 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

4.379 `__gnu_pbds::trivial_iterator_tag` Struct Reference

4.379.1 Detailed Description

A trivial iterator tag. Signifies that the iterators has none of `std::iterators`'s movement abilities.

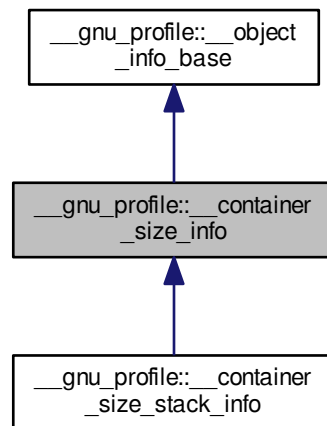
Definition at line 75 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag_and_trait.hpp](#)

4.380 `__gnu_profile::__container_size_info` Class Reference

Inheritance diagram for `__gnu_profile::__container_size_info`:



Public Member Functions

- **__container_size_info** (const [__container_size_info](#) &__o)
- **__container_size_info** (__stack_t __stack, std::size_t __num)
- [std::string __advice](#) () const
- void **__destruct** (std::size_t __num, std::size_t __inum)
- bool **__is_valid** () const
- float **__magnitude** () const
- void **__merge** (const [__container_size_info](#) &__o)
- void **__resize** (std::size_t __from, std::size_t __to)
- float **__resize_cost** (std::size_t __from, std::size_t)
- [__stack_t __stack](#) () const
- void **__write** (FILE *__f) const

Protected Attributes

- [__stack_t __M_stack](#)
- bool **__M_valid**

4.380.1 Detailed Description

A container size instrumentation line in the object table.

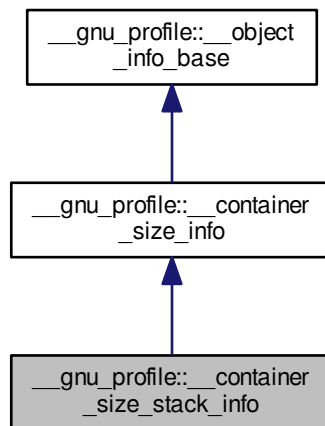
Definition at line 42 of file profiler_container_size.h.

The documentation for this class was generated from the following file:

- [profiler_container_size.h](#)

4.381 [__gnu_profile::__container_size_stack_info](#) Class Reference

Inheritance diagram for [__gnu_profile::__container_size_stack_info](#):



Public Member Functions

- `__container_size_stack_info` (const `__container_size_info` &__o)
- `std::string __advice` () const
- void `__destruct` (std::size_t __num, std::size_t __inum)
- bool `__is_valid` () const
- float `__magnitude` () const
- void `__merge` (const `__container_size_info` &__o)
- void `__resize` (std::size_t __from, std::size_t __to)
- float `__resize_cost` (std::size_t __from, std::size_t)
- `__stack_t __stack` () const
- void `__write` (FILE *__f) const

Protected Attributes

- `__stack_t __M_stack`
- bool `__M_valid`

4.381.1 Detailed Description

A container size instrumentation line in the stack table.

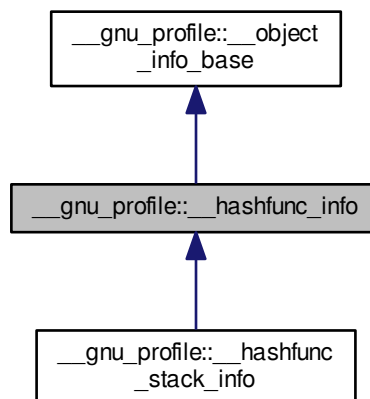
Definition at line 154 of file `profiler_container_size.h`.

The documentation for this class was generated from the following file:

- [profiler_container_size.h](#)

4.382 `__gnu_profile::__hashfunc_info` Class Reference

Inheritance diagram for `__gnu_profile::__hashfunc_info`:



Public Member Functions

- **__hashfunc_info** (const [__hashfunc_info](#) &__o)
- **__hashfunc_info** (__stack_t __stack)
- [std::string](#) **__advice** () const
- void **__destruct** (std::size_t __chain, std::size_t __accesses, std::size_t __hops)
- bool **__is_valid** () const
- float **__magnitude** () const
- void **__merge** (const [__hashfunc_info](#) &__o)
- [__stack_t](#) **__stack** () const
- void **__write** (FILE *__f) const

Protected Attributes

- [__stack_t](#) **_M_stack**
- bool **_M_valid**

4.382.1 Detailed Description

A hash performance instrumentation line in the object table.

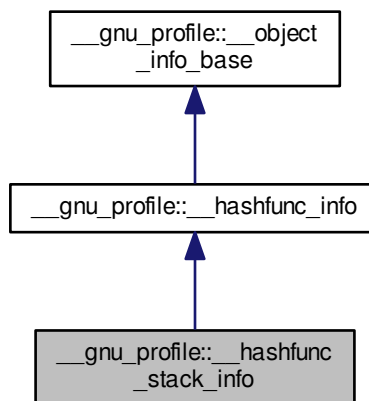
Definition at line 40 of file profiler_hash_func.h.

The documentation for this class was generated from the following file:

- [profiler_hash_func.h](#)

4.383 [__gnu_profile::__hashfunc_stack_info](#) Class Reference

Inheritance diagram for [__gnu_profile::__hashfunc_stack_info](#):



Public Member Functions

- `__hashfunc_stack_info` (const `__hashfunc_info` &__o)
- `std::string __advice` () const
- `void __destruct` (std::size_t __chain, std::size_t __accesses, std::size_t __hops)
- `bool __is_valid` () const
- `float __magnitude` () const
- `void __merge` (const `__hashfunc_info` &__o)
- `__stack_t __stack` () const
- `void __write` (FILE *__f) const

Protected Attributes

- `__stack_t __M_stack`
- `bool __M_valid`

4.383.1 Detailed Description

A hash performance instrumentation line in the stack table.

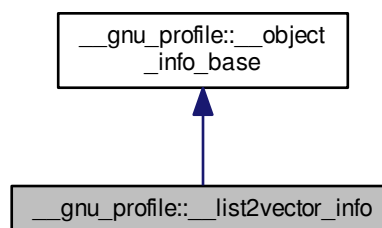
Definition at line 95 of file `profiler_hash_func.h`.

The documentation for this class was generated from the following file:

- [profiler_hash_func.h](#)

4.384 `__gnu_profile::__list2vector_info` Class Reference

Inheritance diagram for `__gnu_profile::__list2vector_info`:



Public Member Functions

- **__list2vector_info** (__stack_t __stack)
- **__list2vector_info** (const [__list2vector_info](#) &__o)
- [std::string](#) **__advice** () const
- bool **__is_valid** ()
- bool **__is_valid** () const
- std::size_t **__iterate** ()
- float **__list_cost** ()
- float **__magnitude** () const
- void **__merge** (const [__list2vector_info](#) &__o)
- void **__opr_insert** (std::size_t __shift, std::size_t __size)
- void **__opr_iterate** (std::size_t __num)
- std::size_t **__resize** ()
- void **__resize** (std::size_t __from, std::size_t)
- void **__set_invalid** ()
- void **__set_list_cost** (float __lc)
- void **__set_vector_cost** (float __vc)
- std::size_t **__shift_count** ()
- __stack_t **__stack** () const
- void **__write** (FILE * __f) const

Protected Attributes

- __stack_t **_M_stack**

4.384.1 Detailed Description

A list-to-vector instrumentation line in the object table.

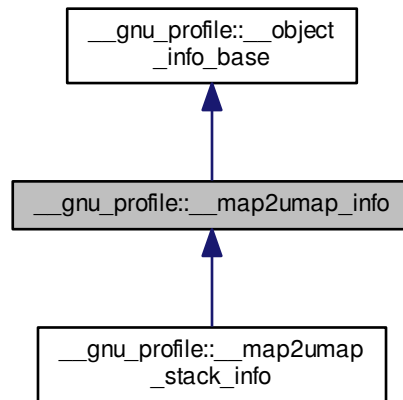
Definition at line 42 of file profiler_list_to_vector.h.

The documentation for this class was generated from the following file:

- [profiler_list_to_vector.h](#)

4.385 __gnu_profile::__map2umap_info Class Reference

Inheritance diagram for __gnu_profile::__map2umap_info:



Public Member Functions

- **__map2umap_info** (__stack_t __stack)
- **__map2umap_info** (const [__map2umap_info](#) &__o)
- [std::string](#) **__advice** () const
- bool **__is_valid** () const
- float **__magnitude** () const
- void **__merge** (const [__map2umap_info](#) &__o)
- void **__record_erase** (std::size_t __size, std::size_t __count)
- void **__record_find** (std::size_t __size)
- void **__record_insert** (std::size_t __size, std::size_t __count)
- void **__record_invalidate** ()
- void **__record_iterate** (std::size_t __count)
- __stack_t **__stack** () const
- void **__write** (FILE *__f) const

Protected Attributes

- __stack_t **_M_stack**

4.385.1 Detailed Description

A map-to-unordered_map instrumentation line in the object table.

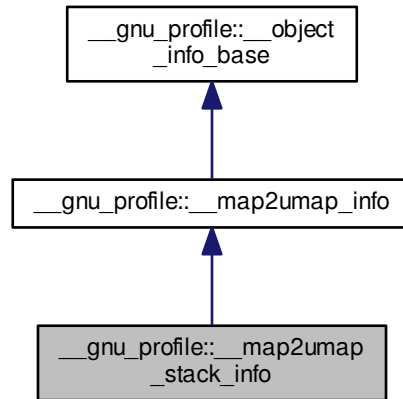
Definition at line 66 of file `profiler_map_to_unordered_map.h`.

The documentation for this class was generated from the following file:

- [profiler_map_to_unordered_map.h](#)

4.386 `__gnu_profile::__map2umap_stack_info` Class Reference

Inheritance diagram for `__gnu_profile::__map2umap_stack_info`:



Public Member Functions

- `__map2umap_stack_info` (const `__map2umap_info` &`__o`)
- `std::string __advice` () const
- `bool __is_valid` () const
- `float __magnitude` () const
- `void __merge` (const `__map2umap_info` &`__o`)
- `void __record_erase` (std::size_t `__size`, std::size_t `__count`)
- `void __record_find` (std::size_t `__size`)
- `void __record_insert` (std::size_t `__size`, std::size_t `__count`)
- `void __record_invalidate` ()
- `void __record_iterate` (std::size_t `__count`)
- `__stack_t __stack` () const
- `void __write` (FILE *`__f`) const

Protected Attributes

- `__stack_t __M_stack`

4.386.1 Detailed Description

A map-to-unordered_map instrumentation line in the stack table.

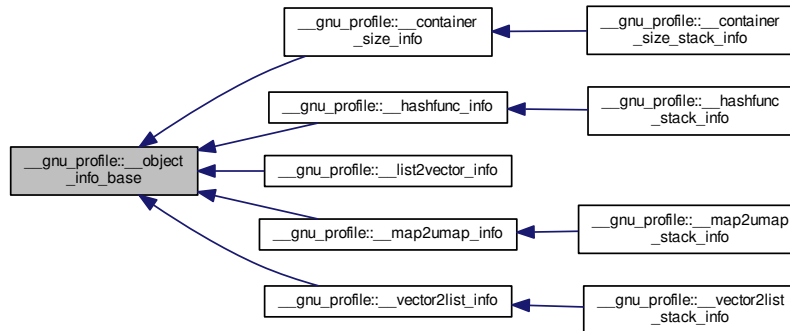
Definition at line 170 of file `profiler_map_to_unordered_map.h`.

The documentation for this class was generated from the following file:

- `profiler_map_to_unordered_map.h`

4.387 __gnu_profile::__object_info_base Class Reference

Inheritance diagram for __gnu_profile::__object_info_base:



Public Member Functions

- `__object_info_base` (`__stack_t __stack`)
- `__object_info_base` (const `__object_info_base` &__o)
- `bool __is_valid` () const
- `__stack_t __stack` () const
- virtual void `__write` (FILE *__f) const =0

Protected Attributes

- `__stack_t __M_stack`
- `bool __M_valid`

4.387.1 Detailed Description

Base class for a line in the object table.

Definition at line 123 of file `profiler_node.h`.

The documentation for this class was generated from the following file:

- [profiler_node.h](#)

4.388 __gnu_profile::__reentrance_guard Struct Reference

Static Public Member Functions

- static bool `__get_in` ()
- static bool & `__inside` ()

4.388.1 Detailed Description

Reentrance guard.

Mechanism to protect all `__gnu_profile` operations against recursion, multithreaded and exception reentrance.

Definition at line 58 of file `profiler.h`.

The documentation for this struct was generated from the following file:

- [profiler.h](#)

4.389 `__gnu_profile::__stack_hash` Class Reference

Public Member Functions

- `std::size_t operator() (__stack_t __s) const`
- `bool operator() (__stack_t __stack1, __stack_t __stack2) const`

4.389.1 Detailed Description

Hash function for summary trace using call stack as index.

Definition at line 89 of file `profiler_node.h`.

The documentation for this class was generated from the following file:

- [profiler_node.h](#)

4.390 `__gnu_profile::__stack_info_base< __object_info >` Class Template Reference

Public Member Functions

- `__stack_info_base (const __object_info &__info)=0`
- `virtual const char * __get_id () const =0`
- `virtual float __magnitude () const =0`
- `void __merge (const __object_info &__info)=0`

4.390.1 Detailed Description

```
template<typename __object_info>
class __gnu_profile::__stack_info_base< __object_info >
```

Base class for a line in the stack table.

Definition at line 154 of file `profiler_node.h`.

The documentation for this class was generated from the following file:

- [profiler_node.h](#)

4.391 `__gnu_profile::__trace_base< __object_info, __stack_info >` Class Template Reference

Public Member Functions

- void **__add_object** (`__object_t` object, `__object_info` __info)
- void **__collect_warnings** (`__warning_vector_t` &__warnings)
- `__object_info` * **__get_object_info** (`__object_t` __object)
- void **__retire_object** (`__object_t` __object)
- void **__write** (`FILE` *__f)

Protected Attributes

- const char * **__id**

4.391.1 Detailed Description

```
template<typename __object_info, typename __stack_info>
class __gnu_profile::__trace_base< __object_info, __stack_info >
```

Base class for all trace producers.

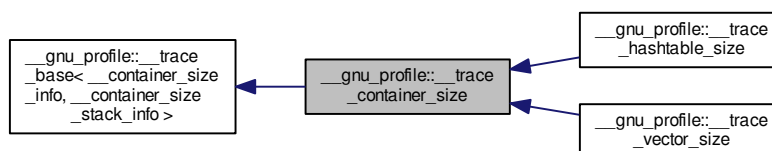
Definition at line 183 of file `profiler_trace.h`.

The documentation for this class was generated from the following file:

- [profiler_trace.h](#)

4.392 `__gnu_profile::__trace_container_size` Class Reference

Inheritance diagram for `__gnu_profile::__trace_container_size`:



Public Member Functions

- void **__add_object** (__object_t object, [__container_size_info](#) __info)
- void **__collect_warnings** (__warning_vector_t &__warnings)
- void **__construct** (const void *__obj, std::size_t __inum)
- void **__destruct** (const void *__obj, std::size_t __num, std::size_t __inum)
- [__container_size_info](#) * **__get_object_info** (__object_t __object)
- void **__insert** (const __object_t __obj, __stack_t __stack, std::size_t __num)
- void **__resize** (const void *__obj, int __from, int __to)
- void **__retire_object** (__object_t __object)
- void **__write** (FILE *__f)

Protected Attributes

- const char * **__id**

4.392.1 Detailed Description

Container size instrumentation trace producer.

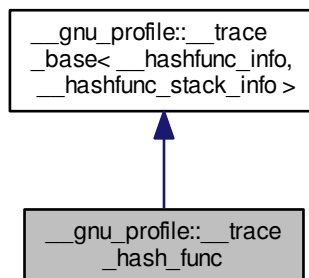
Definition at line 164 of file profiler_container_size.h.

The documentation for this class was generated from the following file:

- [profiler_container_size.h](#)

4.393 `__gnu_profile::__trace_hash_func` Class Reference

Inheritance diagram for `__gnu_profile::__trace_hash_func`:



Public Member Functions

- void `__add_object` (`__object_t` object, `__hashfunc_info` __info)
- void `__collect_warnings` (`__warning_vector_t` &__warnings)
- void `__destruct` (const void *__obj, std::size_t __chain, std::size_t __accesses, std::size_t __hops)
- `__hashfunc_info` * `__get_object_info` (`__object_t` __object)
- void `__insert` (`__object_t` __obj, `__stack_t` __stack)
- void `__retire_object` (`__object_t` __object)
- void `__write` (FILE *__f)

Protected Attributes

- const char * `__id`

4.393.1 Detailed Description

Hash performance instrumentation producer.

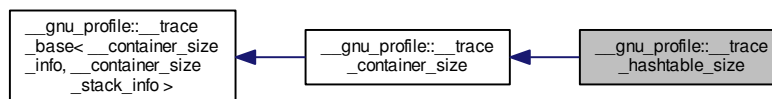
Definition at line 105 of file `profiler_hash_func.h`.

The documentation for this class was generated from the following file:

- [profiler_hash_func.h](#)

4.394 `__gnu_profile::__trace_hashtable_size` Class Reference

Inheritance diagram for `__gnu_profile::__trace_hashtable_size`:



Public Member Functions

- void `__add_object` (`__object_t` object, `__container_size_info` __info)
- void `__collect_warnings` (`__warning_vector_t` &__warnings)
- void `__construct` (const void *__obj, std::size_t __inum)
- void `__destruct` (const void *__obj, std::size_t __num, std::size_t __inum)
- `__container_size_info` * `__get_object_info` (`__object_t` __object)
- void `__insert` (const `__object_t` __obj, `__stack_t` __stack, std::size_t __num)
- void `__resize` (const void *__obj, int __from, int __to)
- void `__retire_object` (`__object_t` __object)
- void `__write` (FILE *__f)

Protected Attributes

- `const char * __id`

4.394.1 Detailed Description

Hashtable size instrumentation trace producer.

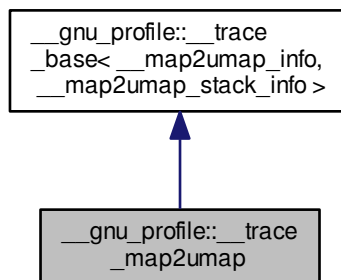
Definition at line 42 of file `profiler_hashtable_size.h`.

The documentation for this class was generated from the following file:

- [profiler_hashtable_size.h](#)

4.395 `__gnu_profile::__trace_map2umap` Class Reference

Inheritance diagram for `__gnu_profile::__trace_map2umap`:



Public Member Functions

- `void __add_object (__object_t object, __map2umap_info __info)`
- `void __collect_warnings (__warning_vector_t &__warnings)`
- `__map2umap_info * __get_object_info (__object_t __object)`
- `void __retire_object (__object_t __object)`
- `void __write (FILE *__f)`

Protected Attributes

- `const char * __id`

4.395.1 Detailed Description

Map-to-unordered_map instrumentation producer.

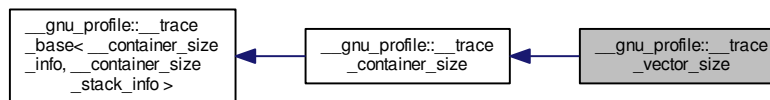
Definition at line 179 of file `profiler_map_to_unordered_map.h`.

The documentation for this class was generated from the following file:

- [profiler_map_to_unordered_map.h](#)

4.396 `__gnu_profile::__trace_vector_size` Class Reference

Inheritance diagram for `__gnu_profile::__trace_vector_size`:



Public Member Functions

- void **__add_object** (`__object_t` object, [__container_size_info](#) __info)
- void **__collect_warnings** (`__warning_vector_t` &__warnings)
- void **__construct** (const void *__obj, std::size_t __inum)
- void **__destruct** (const void *__obj, std::size_t __num, std::size_t __inum)
- [__container_size_info](#) * **__get_object_info** (`__object_t` __object)
- void **__insert** (const `__object_t` __obj, `__stack_t` __stack, std::size_t __num)
- void **__resize** (const void *__obj, int __from, int __to)
- void **__retire_object** (`__object_t` __object)
- void **__write** (FILE *__f)

Protected Attributes

- const char * **__id**

4.396.1 Detailed Description

Hashtable size instrumentation trace producer.

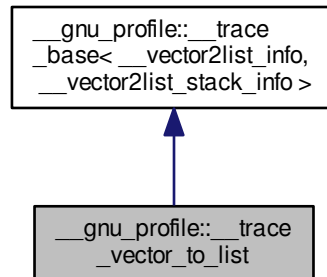
Definition at line 42 of file `profiler_vector_size.h`.

The documentation for this class was generated from the following file:

- [profiler_vector_size.h](#)

4.397 __gnu_profile::__trace_vector_to_list Class Reference

Inheritance diagram for __gnu_profile::__trace_vector_to_list:



Public Member Functions

- void **__add_object** (__object_t object, [__vector2list_info](#) __info)
- void **__collect_warnings** (__warning_vector_t &__warnings)
- void **__destruct** (const void *__obj)
- [__vector2list_info](#) * **__find** (const void *__obj)
- [__vector2list_info](#) * **__get_object_info** (__object_t __object)
- void **__insert** (__object_t __obj, __stack_t __stack)
- void **__invalid_operator** (const void *__obj)
- float **__list_cost** (std::size_t __shift, std::size_t __iterate, std::size_t __resize)
- void **__opr_find** (const void *__obj, std::size_t __size)
- void **__opr_insert** (const void *__obj, std::size_t __pos, std::size_t __num)
- void **__opr_iterate** (const void *__obj, std::size_t __num)
- void **__resize** (const void *__obj, std::size_t __from, std::size_t __to)
- void **__retire_object** (__object_t __object)
- float **__vector_cost** (std::size_t __shift, std::size_t __iterate, std::size_t __resize)
- void **__write** (FILE *__f)

Protected Attributes

- const char * **__id**

4.397.1 Detailed Description

Vector-to-list instrumentation producer.

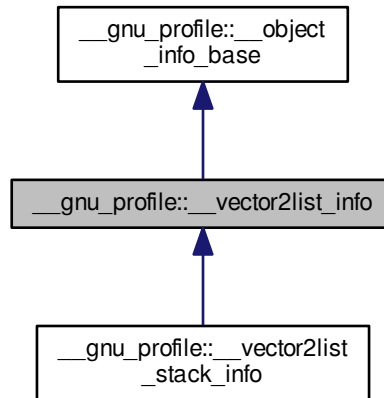
Definition at line 158 of file profiler_vector_to_list.h.

The documentation for this class was generated from the following file:

- [profiler_vector_to_list.h](#)

4.398 `__gnu_profile::__vector2list_info` Class Reference

Inheritance diagram for `__gnu_profile::__vector2list_info`:



Public Member Functions

- `__vector2list_info` (`__stack_t __stack`)
- `__vector2list_info` (`const __vector2list_info &__o`)
- `std::string __advice` () `const`
- `bool __is_valid` ()
- `bool __is_valid` () `const`
- `std::size_t __iterate` ()
- `float __list_cost` ()
- `float __magnitude` () `const`
- `void __merge` (`const __vector2list_info &__o`)
- `void __opr_find` (`std::size_t __size`)
- `void __opr_insert` (`std::size_t __pos`, `std::size_t __num`)
- `void __opr_iterate` (`std::size_t __num`)
- `std::size_t __resize` ()
- `void __resize` (`std::size_t __from`, `std::size_t`)
- `void __set_invalid` ()
- `void __set_list_cost` (`float __lc`)
- `void __set_vector_cost` (`float __vc`)
- `std::size_t __shift_count` ()
- `__stack_t __stack` () `const`
- `void __write` (`FILE *__f`) `const`

Protected Attributes

- `__stack_t __M_stack`

4.398.1 Detailed Description

A vector-to-list instrumentation line in the object table.

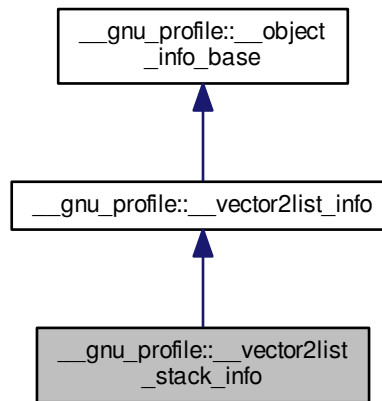
Definition at line 40 of file profiler_vector_to_list.h.

The documentation for this class was generated from the following file:

- [profiler_vector_to_list.h](#)

4.399 __gnu_profile::__vector2list_stack_info Class Reference

Inheritance diagram for __gnu_profile::__vector2list_stack_info:



Public Member Functions

- **__vector2list_stack_info** (const [__vector2list_info](#) &__o)
- [std::string](#) **__advice** () const
- bool **__is_valid** ()
- bool **__is_valid** () const
- [std::size_t](#) **__iterate** ()
- float **__list_cost** ()
- float **__magnitude** () const
- void **__merge** (const [__vector2list_info](#) &__o)
- void **__opr_find** ([std::size_t](#) __size)
- void **__opr_insert** ([std::size_t](#) __pos, [std::size_t](#) __num)
- void **__opr_iterate** ([std::size_t](#) __num)
- [std::size_t](#) **__resize** ()

- void `__resize` (std::size_t __from, std::size_t)
- void `__set_invalid` ()
- void `__set_list_cost` (float __lc)
- void `__set_vector_cost` (float __vc)
- std::size_t `__shift_count` ()
- `__stack_t __stack` () const
- void `__write` (FILE *__f) const

Protected Attributes

- `__stack_t __M_stack`

4.399.1 Detailed Description

A vector-to-list instrumentation line in the stack table.

Definition at line 148 of file `profiler_vector_to_list.h`.

The documentation for this class was generated from the following file:

- [profiler_vector_to_list.h](#)

4.400 `__gnu_profile::__warning_data` Struct Reference

Public Member Functions

- `__warning_data` (float __m, __stack_t __c, const char *__id, const [std::string](#) &__msg)
- bool `operator<` (const [__warning_data](#) &__other) const

Public Attributes

- `__stack_t __context`
- float `__magnitude`
- const char * `__warning_id`
- [std::string](#) `__warning_message`

4.400.1 Detailed Description

Representation of a warning.

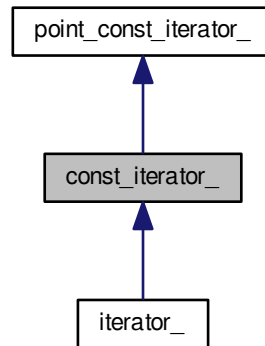
Definition at line 73 of file `profiler_trace.h`.

The documentation for this struct was generated from the following file:

- [profiler_trace.h](#)

4.401 `const_iterator_` Class Reference

Inheritance diagram for `const_iterator_`:



Public Types

- typedef `const_pointer_` [const_pointer](#)
- typedef `const_reference_` [const_reference](#)
- typedef `_Alloc::difference_type` [difference_type](#)
- typedef `std::forward_iterator_tag` [iterator_category](#)
- typedef `pointer_` [pointer](#)
- typedef `reference_` [reference](#)
- typedef `value_type_` [value_type](#)

Public Member Functions

- [const_iterator_\(\)](#)
- `bool operator!=` (`const` [point_iterator_](#) &`other`) `const`
- `bool operator!=` (`const` [point_const_iterator_](#) &`other`) `const`
- `const_reference operator*` () `const`
- `const_iterator_ & operator++` ()
- `const_iterator_ operator++` (`int`)
- `const_pointer operator->` () `const`
- `bool operator==` (`const` [point_iterator_](#) &`other`) `const`
- `bool operator==` (`const` [point_const_iterator_](#) &`other`) `const`

Protected Types

- typedef [point_const_iterator_](#) `base_type`

Protected Member Functions

- `const_iterator_` (`const_pointer_ p_value`, `PB_DS_GEN_POS pos`, `const PB_DS_CLASS_C_DEC *p_tbl`)

Protected Attributes

- `const PB_DS_CLASS_C_DEC * m_p_tbl`
- `const_pointer m_p_value`
- `PB_DS_GEN_POS m_pos`

Friends

- class `PB_DS_CLASS_C_DEC`

4.401.1 Detailed Description

Const range-type iterator.

Definition at line 43 of file `unordered_iterator/const_iterator.hpp`.

4.401.2 Member Typedef Documentation

4.401.2.1 `typedef const_pointer_ const_iterator_::const_pointer`

Iterator's const pointer type.

Definition at line 60 of file `unordered_iterator/const_iterator.hpp`.

4.401.2.2 `typedef const_reference_ const_iterator_::const_reference`

Iterator's const reference type.

Definition at line 66 of file `unordered_iterator/const_iterator.hpp`.

4.401.2.3 `typedef _Alloc::difference_type const_iterator_::difference_type`

Difference type.

Definition at line 51 of file `unordered_iterator/const_iterator.hpp`.

4.401.2.4 `typedef std::forward_iterator_tag const_iterator_::iterator_category`

Category.

Definition at line 48 of file `unordered_iterator/const_iterator.hpp`.

4.401.2.5 `typedef pointer_const_iterator::pointer`

Iterator's pointer type.

Definition at line 57 of file `unordered_iterator/const_iterator.hpp`.

4.401.2.6 `typedef reference_const_iterator::reference`

Iterator's reference type.

Definition at line 63 of file `unordered_iterator/const_iterator.hpp`.

4.401.2.7 `typedef value_type_const_iterator::value_type`

Iterator's value type.

Definition at line 54 of file `unordered_iterator/const_iterator.hpp`.

4.401.3 Constructor & Destructor Documentation

4.401.3.1 `const_iterator::const_iterator() [inline]`

Default constructor.

Definition at line 69 of file `unordered_iterator/const_iterator.hpp`.

4.401.3.2 `const_iterator::const_iterator(const_pointer_p_value, PB_DS_GEN_POS pos, const PB_DS_CLASS_C_DEC * p_tbl) [inline], [protected]`

Constructor used by the table to initiate the generalized pointer and position (e.g., this is called from within a `find()` of a table).

Definition at line 97 of file `unordered_iterator/const_iterator.hpp`.

4.401.4 Member Function Documentation

4.401.4.1 `bool point_const_iterator::operator!=(const point_iterator_ & other) const [inline], [inherited]`

Compares content (negatively) to a different iterator object.

Definition at line 118 of file `unordered_iterator/point_const_iterator.hpp`.

4.401.4.2 `bool point_const_iterator::operator!=(const point_const_iterator_ & other) const [inline], [inherited]`

Compares content (negatively) to a different iterator object.

Definition at line 123 of file `unordered_iterator/point_const_iterator.hpp`.

4.401.4.3 const_reference point_const_iterator::operator*() const [inline],[inherited]

Access.

Definition at line 100 of file unordered_iterator/point_const_iterator.hpp.

4.401.4.4 const_iterator_ & const_iterator::operator++() [inline]

Increments.

Definition at line 74 of file unordered_iterator/const_iterator.hpp.

References `m_p_tbl`.

4.401.4.5 const_iterator_ const_iterator::operator++(int) [inline]

Increments.

Definition at line 82 of file unordered_iterator/const_iterator.hpp.

References `m_p_tbl`.

4.401.4.6 const_pointer point_const_iterator::operator->() const [inline],[inherited]

Access.

Definition at line 92 of file unordered_iterator/point_const_iterator.hpp.

4.401.4.7 bool point_const_iterator::operator==(const point_iterator_ & other) const [inline],[inherited]

Compares content to a different iterator object.

Definition at line 108 of file unordered_iterator/point_const_iterator.hpp.

4.401.4.8 bool point_const_iterator::operator==(const point_const_iterator_ & other) const [inline],[inherited]

Compares content to a different iterator object.

Definition at line 113 of file unordered_iterator/point_const_iterator.hpp.

4.401.5 Member Data Documentation

4.401.5.1 const PB_DS_CLASS_C_DEC* const_iterator::m_p_tbl [protected]

Pointer to the table object which created the iterator (used for incrementing its position).

Definition at line 106 of file unordered_iterator/const_iterator.hpp.

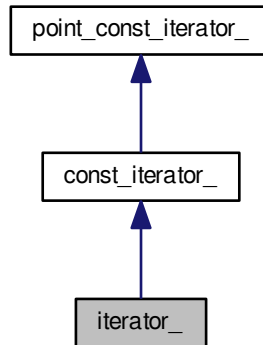
Referenced by `operator++()`, and `iterator::operator++()`.

The documentation for this class was generated from the following file:

- [unordered_iterator/const_iterator.hpp](#)

4.402 iterator_ Class Reference

Inheritance diagram for iterator_:



Public Types

- typedef const_pointer_ [const_pointer](#)
- typedef const_reference_ [const_reference](#)
- typedef _Alloc::difference_type [difference_type](#)
- typedef [std::forward_iterator_tag](#) [iterator_category](#)
- typedef pointer_ [pointer](#)
- typedef reference_ [reference](#)
- typedef value_type_ [value_type](#)

Public Member Functions

- [iterator_\(\)](#)
- [operator const point_iterator_\(\)](#) const
- [operator point_iterator_\(\)](#)
- bool [operator!=](#) (const [point_iterator_](#) &other) const
- bool [operator!=](#) (const [point_const_iterator_](#) &other) const
- [reference operator*](#) () const
- [iterator_ & operator++](#) ()
- [iterator_ operator++](#) (int)
- [pointer operator->](#) () const
- bool [operator==](#) (const [point_iterator_](#) &other) const
- bool [operator==](#) (const [point_const_iterator_](#) &other) const

Protected Types

- typedef [const_iterator_](#) [base_type](#)

Protected Member Functions

- `iterator_ (pointer p_value, PB_DS_GEN_POS pos, PB_DS_CLASS_C_DEC *p_tbl)`

Protected Attributes

- `const PB_DS_CLASS_C_DEC * m_p_tbl`
- `const_pointer m_p_value`
- `PB_DS_GEN_POS m_pos`

Friends

- class `PB_DS_CLASS_C_DEC`

4.402.1 Detailed Description

Range-type iterator.

Definition at line 43 of file `iterator.hpp`.

4.402.2 Member Typedef Documentation

4.402.2.1 `typedef const_pointer_iterator_::const_pointer`

Iterator's const pointer type.

Definition at line 60 of file `iterator.hpp`.

4.402.2.2 `typedef const_reference_iterator_::const_reference`

Iterator's const reference type.

Definition at line 66 of file `iterator.hpp`.

4.402.2.3 `typedef _Alloc::difference_type iterator_::difference_type`

Difference type.

Definition at line 51 of file `iterator.hpp`.

4.402.2.4 `typedef std::forward_iterator_tag iterator_::iterator_category`

Category.

Definition at line 48 of file `iterator.hpp`.

4.402.2.5 `typedef pointer_iterator::pointer`

Iterator's pointer type.

Definition at line 57 of file iterator.hpp.

4.402.2.6 `typedef reference_iterator::reference`

Iterator's reference type.

Definition at line 63 of file iterator.hpp.

4.402.2.7 `typedef value_type_iterator::value_type`

Iterator's value type.

Definition at line 54 of file iterator.hpp.

4.402.3 Constructor & Destructor Documentation

4.402.3.1 `iterator::iterator() [inline]`

Default constructor.

Definition at line 70 of file iterator.hpp.

4.402.3.2 `iterator::iterator(pointer p_value, PB_DS_GEN_POS pos, PB_DS_CLASS_C_DEC * p_tbl) [inline], [protected]`

Constructor used by the table to initiate the generalized pointer and position (e.g., this is called from within a `find()` of a table).

Definition at line 125 of file iterator.hpp.

4.402.4 Member Function Documentation

4.402.4.1 `iterator::operator const point_iterator() const [inline]`

Conversion to a point-type iterator.

Definition at line 80 of file iterator.hpp.

4.402.4.2 `iterator::operator point_iterator() [inline]`

Conversion to a point-type iterator.

Definition at line 75 of file iterator.hpp.

4.402.4.3 `bool point_const_iterator::operator!=(const point_iterator_ & other) const` `[inline]`, `[inherited]`

Compares content (negatively) to a different iterator object.

Definition at line 118 of file `unordered_iterator/point_const_iterator.hpp`.

4.402.4.4 `bool point_const_iterator::operator!=(const point_const_iterator_ & other) const` `[inline]`, `[inherited]`

Compares content (negatively) to a different iterator object.

Definition at line 123 of file `unordered_iterator/point_const_iterator.hpp`.

4.402.4.5 `reference iterator::operator*() const` `[inline]`

Access.

Definition at line 93 of file `iterator.hpp`.

4.402.4.6 `iterator_ & iterator::operator++()` `[inline]`

Increments.

Definition at line 101 of file `iterator.hpp`.

References `const_iterator::m_p_tbl`.

4.402.4.7 `iterator_iterator::operator++(int)` `[inline]`

Increments.

Definition at line 109 of file `iterator.hpp`.

References `const_iterator::m_p_tbl`.

4.402.4.8 `pointer iterator::operator->() const` `[inline]`

Access.

Definition at line 85 of file `iterator.hpp`.

4.402.4.9 `bool point_const_iterator::operator==(const point_iterator_ & other) const` `[inline]`, `[inherited]`

Compares content to a different iterator object.

Definition at line 108 of file `unordered_iterator/point_const_iterator.hpp`.

4.402.4.10 `bool point_const_iterator::operator== (const point_const_iterator_ & other) const` `[inline]`,
`[inherited]`

Compares content to a different iterator object.

Definition at line 113 of file `unordered_iterator/point_const_iterator.hpp`.

4.402.5 Member Data Documentation

4.402.5.1 `const PB_DS_CLASS_C_DEC* const_iterator::m_p_tbl` `[protected]`, `[inherited]`

Pointer to the table object which created the iterator (used for incrementing its position).

Definition at line 106 of file `unordered_iterator/const_iterator.hpp`.

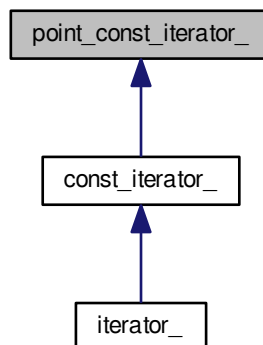
Referenced by `const_iterator::operator++()`, and `operator++()`.

The documentation for this class was generated from the following file:

- [iterator.hpp](#)

4.403 point_const_iterator_ Class Reference

Inheritance diagram for `point_const_iterator_`:



Public Types

- typedef const_pointer_ [const_pointer](#)
- typedef const_reference_ [const_reference](#)
- typedef trivial_iterator_difference_type [difference_type](#)
- typedef trivial_iterator_tag [iterator_category](#)
- typedef pointer_ [pointer](#)
- typedef reference_ [reference](#)
- typedef value_type_ [value_type](#)

Public Member Functions

- **point_const_iterator_** ([const_pointer](#) p_value)
- [point_const_iterator_](#) ()
- [point_const_iterator_](#) (const [point_const_iterator_](#) &other)
- [point_const_iterator_](#) (const [point_iterator_](#) &other)
- bool [operator!=](#) (const [point_iterator_](#) &other) const
- bool [operator!=](#) (const [point_const_iterator_](#) &other) const
- [const_reference](#) [operator*](#) () const
- [const_pointer](#) [operator->](#) () const
- bool [operator==](#) (const [point_iterator_](#) &other) const
- bool [operator==](#) (const [point_const_iterator_](#) &other) const

Protected Attributes

- [const_pointer](#) **m_p_value**

Friends

- class **PB_DS_CLASS_C_DEC**
- class **point_iterator_**

4.403.1 Detailed Description

Const point-type iterator.

Definition at line 45 of file unordered_iterator/point_const_iterator.hpp.

4.403.2 Member Typedef Documentation

4.403.2.1 typedef const_pointer_ point_const_iterator_::const_pointer

Iterator's const pointer type.

Definition at line 61 of file unordered_iterator/point_const_iterator.hpp.

4.403.2.2 `typedef const_reference_point_const_iterator::const_reference`

Iterator's const reference type.

Definition at line 67 of file `unordered_iterator/point_const_iterator.hpp`.

4.403.2.3 `typedef trivial_iterator_difference_type_point_const_iterator::difference_type`

Difference type.

Definition at line 52 of file `unordered_iterator/point_const_iterator.hpp`.

4.403.2.4 `typedef trivial_iterator_tag_point_const_iterator::iterator_category`

Category.

Definition at line 49 of file `unordered_iterator/point_const_iterator.hpp`.

4.403.2.5 `typedef pointer_point_const_iterator::pointer`

Iterator's pointer type.

Definition at line 58 of file `unordered_iterator/point_const_iterator.hpp`.

4.403.2.6 `typedef reference_point_const_iterator::reference`

Iterator's reference type.

Definition at line 64 of file `unordered_iterator/point_const_iterator.hpp`.

4.403.2.7 `typedef value_type_point_const_iterator::value_type`

Iterator's value type.

Definition at line 55 of file `unordered_iterator/point_const_iterator.hpp`.

4.403.3 Constructor & Destructor Documentation

4.403.3.1 `point_const_iterator::point_const_iterator() [inline]`

Default constructor.

Definition at line 75 of file `unordered_iterator/point_const_iterator.hpp`.

4.403.3.2 `point_const_iterator::point_const_iterator(const point_const_iterator_ & other) [inline]`

Copy constructor.

Definition at line 80 of file `unordered_iterator/point_const_iterator.hpp`.

4.403.3.3 point_const_iterator_::point_const_iterator_ (const point_iterator_ & other) [inline]

Copy constructor.

Definition at line 86 of file unordered_iterator/point_const_iterator.hpp.

4.403.4 Member Function Documentation

4.403.4.1 bool point_const_iterator_::operator!=(const point_iterator_ & other) const [inline]

Compares content (negatively) to a different iterator object.

Definition at line 118 of file unordered_iterator/point_const_iterator.hpp.

4.403.4.2 bool point_const_iterator_::operator!=(const point_const_iterator_ & other) const [inline]

Compares content (negatively) to a different iterator object.

Definition at line 123 of file unordered_iterator/point_const_iterator.hpp.

4.403.4.3 const_reference point_const_iterator_::operator*() const [inline]

Access.

Definition at line 100 of file unordered_iterator/point_const_iterator.hpp.

4.403.4.4 const_pointer point_const_iterator_::operator->() const [inline]

Access.

Definition at line 92 of file unordered_iterator/point_const_iterator.hpp.

4.403.4.5 bool point_const_iterator_::operator==(const point_iterator_ & other) const [inline]

Compares content to a different iterator object.

Definition at line 108 of file unordered_iterator/point_const_iterator.hpp.

4.403.4.6 bool point_const_iterator_::operator==(const point_const_iterator_ & other) const [inline]

Compares content to a different iterator object.

Definition at line 113 of file unordered_iterator/point_const_iterator.hpp.

The documentation for this class was generated from the following file:

- [unordered_iterator/point_const_iterator.hpp](#)

4.404 point_iterator_ Class Reference

Public Types

- typedef const_pointer_ [const_pointer](#)
- typedef const_reference_ [const_reference](#)
- typedef trivial_iterator_difference_type [difference_type](#)
- typedef trivial_iterator_tag [iterator_category](#)
- typedef pointer_ [pointer](#)
- typedef reference_ [reference](#)
- typedef value_type_ [value_type](#)

Public Member Functions

- [point_iterator_\(\)](#)
- [point_iterator_\(const point_iterator_ &other\)](#)
- **point_iterator_()** ([pointer](#) p_value)
- bool [operator!=](#) (const [point_iterator_](#) &other) const
- bool [operator!=](#) (const [point_const_iterator_](#) &other) const
- [reference operator*](#) () const
- [pointer operator->](#) () const
- bool [operator==](#) (const [point_iterator_](#) &other) const
- bool [operator==](#) (const [point_const_iterator_](#) &other) const

Protected Attributes

- [pointer m_p_value](#)

Friends

- class **PB_DS_CLASS_C_DEC**
- class **point_const_iterator_**

4.404.1 Detailed Description

Find type iterator.

Definition at line 43 of file point_iterator.hpp.

4.404.2 Member Typedef Documentation

4.404.2.1 typedef const_pointer_ point_iterator_::const_pointer

Iterator's const pointer type.

Definition at line 59 of file point_iterator.hpp.

4.404.2.2 typedef const_reference_ point_iterator_::const_reference

Iterator's const reference type.

Definition at line 65 of file point_iterator.hpp.

4.404.2.3 typedef trivial_iterator_difference_type point_iterator_::difference_type

Difference type.

Definition at line 50 of file point_iterator.hpp.

4.404.2.4 typedef trivial_iterator_tag point_iterator_::iterator_category

Category.

Definition at line 47 of file point_iterator.hpp.

4.404.2.5 typedef pointer_ point_iterator_::pointer

Iterator's pointer type.

Definition at line 56 of file point_iterator.hpp.

4.404.2.6 typedef reference_ point_iterator_::reference

Iterator's reference type.

Definition at line 62 of file point_iterator.hpp.

4.404.2.7 typedef value_type_ point_iterator_::value_type

Iterator's value type.

Definition at line 53 of file point_iterator.hpp.

4.404.3 Constructor & Destructor Documentation

4.404.3.1 point_iterator_::point_iterator_() [inline]

Default constructor.

Definition at line 69 of file point_iterator.hpp.

Referenced by operator!==().

4.404.3.2 `point_iterator::point_iterator_ (const point_iterator_ & other)` `[inline]`

Copy constructor.

Definition at line 75 of file `point_iterator.hpp`.

4.404.4 Member Function Documentation

4.404.4.1 `bool point_iterator::operator!= (const point_iterator_ & other) const` `[inline]`

Compares content to a different iterator object.

Definition at line 107 of file `point_iterator.hpp`.

4.404.4.2 `bool point_iterator::operator!= (const point_const_iterator_ & other) const` `[inline]`

Compares content (negatively) to a different iterator object.

Definition at line 112 of file `point_iterator.hpp`.

References `point_iterator_()`.

4.404.4.3 `reference point_iterator::operator* () const` `[inline]`

Access.

Definition at line 89 of file `point_iterator.hpp`.

4.404.4.4 `pointer point_iterator::operator-> () const` `[inline]`

Access.

Definition at line 81 of file `point_iterator.hpp`.

4.404.4.5 `bool point_iterator::operator== (const point_iterator_ & other) const` `[inline]`

Compares content to a different iterator object.

Definition at line 97 of file `point_iterator.hpp`.

4.404.4.6 `bool point_iterator::operator== (const point_const_iterator_ & other) const` `[inline]`

Compares content to a different iterator object.

Definition at line 102 of file `point_iterator.hpp`.

The documentation for this class was generated from the following file:

- [point_iterator.hpp](#)

4.405 std::__atomic_base<_ITp> Struct Template Reference

Public Member Functions

- **__atomic_base** (const [__atomic_base](#) &)=delete
- constexpr **__atomic_base** (__int_type __i) noexcept
- **__attribute__** ((always_inline)) void store(__int_type __i
- bool **is_lock_free** () const noexcept
- bool **is_lock_free** () const volatile noexcept
- **operator __int_type** () const noexcept
- **operator __int_type** () const volatile noexcept
- __int_type **operator&=** (__int_type __i) noexcept
- __int_type **operator&=** (__int_type __i) volatile noexcept
- __int_type **operator++** (int) noexcept
- __int_type **operator++** (int) volatile noexcept
- __int_type **operator++** () noexcept
- __int_type **operator++** () volatile noexcept
- __int_type **operator+=** (__int_type __i) noexcept
- __int_type **operator+=** (__int_type __i) volatile noexcept
- __int_type **operator--** (int) noexcept
- __int_type **operator--** (int) volatile noexcept
- __int_type **operator--** () noexcept
- __int_type **operator--** () volatile noexcept
- __int_type **operator-=** (__int_type __i) noexcept
- __int_type **operator-=** (__int_type __i) volatile noexcept
- [__atomic_base](#) & **operator=** (const [__atomic_base](#) &)=delete
- [__atomic_base](#) & **operator=** (const [__atomic_base](#) &) volatile=delete
- __int_type **operator=** (__int_type __i) noexcept
- __int_type **operator=** (__int_type __i) volatile noexcept
- __int_type **operator^=** (__int_type __i) noexcept
- __int_type **operator^=** (__int_type __i) volatile noexcept
- __int_type **operator|=** (__int_type __i) noexcept
- __int_type **operator|=** (__int_type __i) volatile noexcept

4.405.1 Detailed Description

```
template<typename _ITp>
struct std::__atomic_base<_ITp>
```

Base class for atomic integrals.

Definition at line 121 of file atomic_base.h.

The documentation for this struct was generated from the following file:

- [atomic_base.h](#)

4.406 `std::__atomic_base<_PTp * >` Struct Template Reference

Public Member Functions

- `__atomic_base` (const `__atomic_base` &)=delete
- `constexpr __atomic_base` (`__pointer_type` __p) noexcept
- `__attribute__` ((always_inline)) void store(`__pointer_type` __p
- `bool is_lock_free` () const noexcept
- `bool is_lock_free` () const volatile noexcept
- `operator __pointer_type` () const noexcept
- `operator __pointer_type` () const volatile noexcept
- `__pointer_type operator++` (int) noexcept
- `__pointer_type operator++` (int) volatile noexcept
- `__pointer_type operator++` () noexcept
- `__pointer_type operator++` () volatile noexcept
- `__pointer_type operator+=` (ptrdiff_t __d) noexcept
- `__pointer_type operator+=` (ptrdiff_t __d) volatile noexcept
- `__pointer_type operator--` (int) noexcept
- `__pointer_type operator--` (int) volatile noexcept
- `__pointer_type operator--` () noexcept
- `__pointer_type operator--` () volatile noexcept
- `__pointer_type operator-=` (ptrdiff_t __d) noexcept
- `__pointer_type operator-=` (ptrdiff_t __d) volatile noexcept
- `__atomic_base & operator=` (const `__atomic_base` &)=delete
- `__atomic_base & operator=` (const `__atomic_base` &) volatile=delete
- `__pointer_type operator=` (`__pointer_type` __p) noexcept
- `__pointer_type operator=` (`__pointer_type` __p) volatile noexcept

4.406.1 Detailed Description

```
template<typename _PTp>
struct std::__atomic_base<_PTp * >
```

Partial specialization for pointer types.

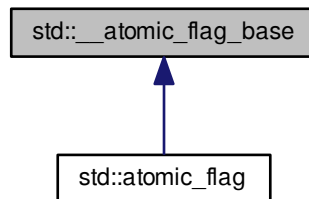
Definition at line 669 of file `atomic_base.h`.

The documentation for this struct was generated from the following file:

- [atomic_base.h](#)

4.407 std::__atomic_flag_base Struct Reference

Inheritance diagram for std::__atomic_flag_base:



Public Attributes

- __atomic_flag_data_type **M_i**

4.407.1 Detailed Description

Base type for atomic_flag.

Base type is POD with data, allowing atomic_flag to derive from it and meet the standard layout type requirement. In addition to compatibility with a C interface, this allows different implementations of atomic_flag to use the same atomic operation functions, via a standard conversion to the __atomic_flag_base argument.

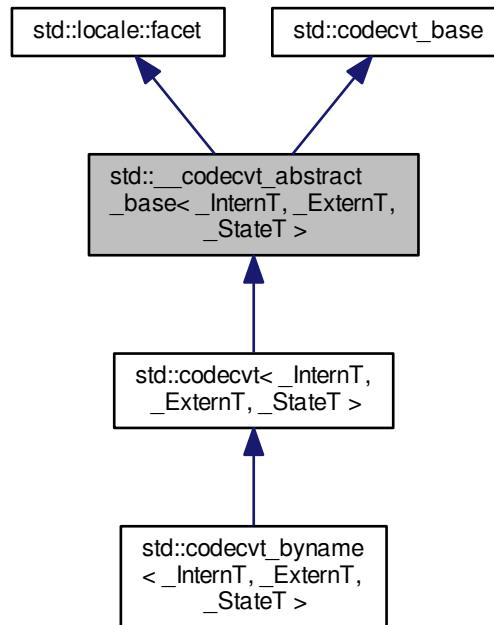
Definition at line 265 of file atomic_base.h.

The documentation for this struct was generated from the following file:

- [atomic_base.h](#)

4.408 `std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >` Class Template Reference

Inheritance diagram for `std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >`:



Public Types

- typedef `_ExternT` **extern_type**
- typedef `_InternT` **intern_type**
- typedef `codecvt_base::result` **result**
- typedef `_StateT` **state_type**

Public Member Functions

- bool **always_noconv** () const throw ()
- int **encoding** () const throw ()
- result **in** (state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *__&__from_next, intern_type *__to, intern_type *__to_end, intern_type *__&__to_next) const
- int **length** (state_type &__state, const extern_type *__from, const extern_type *__end, size_t __max) const
- int **max_length** () const throw ()
- result **out** (state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *__&__from_next, extern_type *__to, extern_type *__to_end, extern_type *__&__to_next) const
- result **unshift** (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *__&__to_next) const

Protected Member Functions

- **__codecvt_abstract_base** (size_t __refs=0)
- virtual bool **do_always_noconv** () const =0 throw ()
- virtual int **do_encoding** () const =0 throw ()
- virtual result **do_in** (state_type &__state, const extern_type * __from, const extern_type * __from_end, const extern_type * __from_next, intern_type * __to, intern_type * __to_end, intern_type * __to_next) const =0
- virtual int **do_length** (state_type &, const extern_type * __from, const extern_type * __end, size_t __max) const =0
- virtual int **do_max_length** () const =0 throw ()
- virtual result **do_out** (state_type &__state, const intern_type * __from, const intern_type * __from_end, const intern_type * __from_next, extern_type * __to, extern_type * __to_end, extern_type * __to_next) const =0
- virtual result **do_unshift** (state_type &__state, extern_type * __to, extern_type * __to_end, extern_type * __to_next) const =0

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char * __s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_type_c_locale** (__c_locale __cloc, const char * __s)

4.408.1 Detailed Description

```
template<typename _InternT, typename _ExternT, typename _StateT>
class std::__codecvt_abstract_base<_InternT, _ExternT, _StateT >
```

Common base for codecvt functions.

This template class provides implementations of the public functions that forward to the protected virtual functions.

This template also provides abstract stubs for the protected virtual functions.

Definition at line 68 of file codecvt.h.

4.408.2 Member Function Documentation

4.408.2.1 `template<typename _InternT, typename _ExternT, typename _StateT> virtual result std::__codecvt_abstract_base<_InternT, _ExternT, _StateT >::do_out (state_type & __state, const intern_type * __from, const intern_type * __from_end, const intern_type * __from_next, extern_type * __to, extern_type * __to_end, extern_type * __to_next) const [protected], [pure virtual]`

Convert from internal to external character set.

Converts input string of intern_type to output string of extern_type. This function is a hook for derived classes to change the value returned.

See also

out for more information.

Implemented in `std::codecvt< wchar_t, char, mbstate_t >`, `std::codecvt< char, char, mbstate_t >`, `std::codecvt< _InternT, _ExternT, _StateT >`, and `std::codecvt< _InternT, _ExternT, encoding_state >`.

```
4.408.2.2  template<typename _InternT, typename _ExternT, typename _StateT> result std::__codecvt_abstract_base<
    _InternT, _ExternT, _StateT >::in ( state_type & __state, const extern_type * __from, const extern_type * __from_end,
    const extern_type * & __from_next, intern_type * __to, intern_type * __to_end, intern_type * & __to_next ) const
    [inline]
```

Convert from external to internal character set.

Converts input string of `extern_type` to output string of `intern_type`. This is analogous to `mbsrtowcs`. It does this by calling `codecvt::do_in`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The `state` argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how `state` is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

Parameters

<code>__state</code>	Persistent conversion state data.
<code>__from</code>	Start of input.
<code>__from_end</code>	End of input.
<code>__from_next</code>	Returns start of unconverted data.
<code>__to</code>	Start of output buffer.
<code>__to_end</code>	End of output buffer.
<code>__to_next</code>	Returns start of unused output area.

Returns

`codecvt_base::result`.

Definition at line 196 of file `codecvt.h`.

```
4.408.2.3  template<typename _InternT, typename _ExternT, typename _StateT> result std::__codecvt_abstract_base<
    _InternT, _ExternT, _StateT >::out ( state_type & __state, const intern_type * __from, const intern_type * __from_end,
    const intern_type * & __from_next, extern_type * __to, extern_type * __to_end, extern_type * & __to_next ) const
    [inline]
```

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This is analogous to `wcsrtombs`. It does this by calling `codecvt::do_out`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

Parameters

<code>__state</code>	Persistent conversion state data.
<code>__from</code>	Start of input.
<code>__from_end</code>	End of input.
<code>__from_next</code>	Returns start of unconverted data.
<code>__to</code>	Start of output buffer.
<code>__to_end</code>	End of output buffer.
<code>__to_next</code>	Returns start of unused output area.

Returns

`codecvt_base::result`.

Definition at line 116 of file `codecvt.h`.

4.408.2.4 `template<typename _InternT, typename _ExternT, typename _StateT> result std::__codecvt_abstract_base<_InternT, _ExternT, _StateT>::unshift (state_type & __state, extern_type * __to, extern_type * __to_end, extern_type *& __to_next) const [inline]`

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling `codecvt::do_unshift()`.

For example, if 4 external characters always converted to 1 internal character, and input to `in()` had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of `codecvt_base::result`. If the state could be reset and data written, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the output has insufficient space, returns `codecvt_base::partial`. Otherwise the reset failed and `codecvt_base::error` is returned.

Parameters

<code>__state</code>	Persistent conversion state data.
<code>__to</code>	Start of output buffer.
<code>__to_end</code>	End of output buffer.
<code>__to_next</code>	Returns start of unused output area.

Returns

codecvt_base::result.

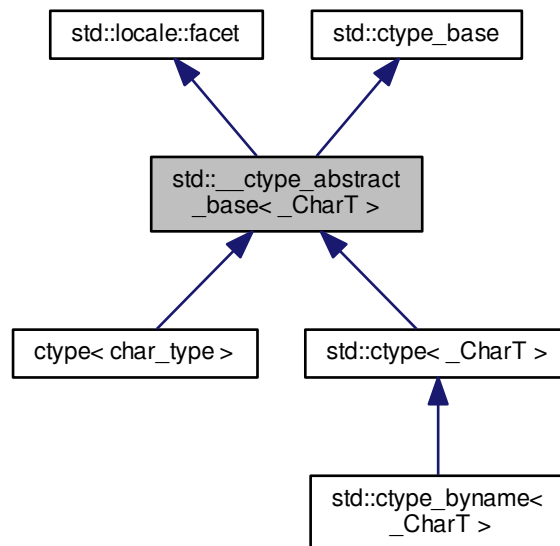
Definition at line 155 of file codecvt.h.

The documentation for this class was generated from the following file:

- [codecvt.h](#)

4.409 std::__ctype_abstract_base< _CharT > Class Template Reference

Inheritance diagram for std::__ctype_abstract_base< _CharT >:

**Public Types**

- typedef const int * **__to_type**
- typedef _CharT [char_type](#)
- typedef unsigned short **mask**

Public Member Functions

- bool **is** (mask __m, char_type __c) const
- const char_type * **is** (const char_type * __lo, const char_type * __hi, mask * __vec) const
- char **narrow** (char_type __c, char __dfault) const
- const char_type * **narrow** (const char_type * __lo, const char_type * __hi, char __dfault, char * __to) const
- const char_type * **scan_is** (mask __m, const char_type * __lo, const char_type * __hi) const
- const char_type * **scan_not** (mask __m, const char_type * __lo, const char_type * __hi) const
- char_type **tolower** (char_type __c) const
- const char_type * **tolower** (char_type * __lo, const char_type * __hi) const
- char_type **toupper** (char_type __c) const
- const char_type * **toupper** (char_type * __lo, const char_type * __hi) const
- char_type **widen** (char __c) const
- const char * **widen** (const char * __lo, const char * __hi, char_type * __to) const

Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const mask **upper**
- static const mask **xdigit**

Protected Member Functions

- **__ctype_abstract_base** (size_t __refs=0)
- virtual bool **do_is** (mask __m, char_type __c) const =0
- virtual const char_type * **do_is** (const char_type * __lo, const char_type * __hi, mask * __vec) const =0
- virtual char **do_narrow** (char_type __c, char __dfault) const =0
- virtual const char_type * **do_narrow** (const char_type * __lo, const char_type * __hi, char __dfault, char * __to) const =0
- virtual const char_type * **do_scan_is** (mask __m, const char_type * __lo, const char_type * __hi) const =0
- virtual const char_type * **do_scan_not** (mask __m, const char_type * __lo, const char_type * __hi) const =0
- virtual char_type **do_tolower** (char_type __c) const =0
- virtual const char_type * **do_tolower** (char_type * __lo, const char_type * __hi) const =0
- virtual char_type **do_toupper** (char_type __c) const =0
- virtual const char_type * **do_toupper** (char_type * __lo, const char_type * __hi) const =0
- virtual char_type **do_widen** (char __c) const =0
- virtual const char * **do_widen** (const char * __lo, const char * __hi, char_type * __to) const =0

Static Protected Member Functions

- static `__c_locale _S_clone_c_locale (__c_locale &__cloc) throw ()`
- static void `_S_create_c_locale (__c_locale &__cloc, const char *__s, __c_locale __old=0)`
- static void `_S_destroy_c_locale (__c_locale &__cloc)`
- static `__c_locale _S_get_c_locale ()`
- static const char * `_S_get_c_name () throw ()`
- static `__c_locale _S_lc_ctype_c_locale (__c_locale __cloc, const char *__s)`

4.409.1 Detailed Description

```
template<typename _CharT>
class std::__ctype_abstract_base< _CharT >
```

Common base for ctype facet.

This template class provides implementations of the public functions that forward to the protected virtual functions.

This template also provides abstract stubs for the protected virtual functions.

Definition at line 143 of file `locale_facets.h`.

4.409.2 Member Typedef Documentation

4.409.2.1 `template<typename _CharT> typedef _CharT std::__ctype_abstract_base< _CharT >::char_type`

Typedef for the template parameter.

Definition at line 148 of file `locale_facets.h`.

4.409.3 Member Function Documentation

4.409.3.1 `template<typename _CharT> virtual bool std::__ctype_abstract_base< _CharT >::do_is (mask __m, char_type __c) const [protected], [pure virtual]`

Test `char_type` classification.

This function finds a mask `M` for `c` and compares it to mask `m`.

`do_is()` is a hook for a derived facet to change the behavior of classifying. `do_is()` must always return the same result for the same input.

Parameters

<code>__c</code>	The <code>char_type</code> to find the mask of.
<code>__m</code>	The mask to compare against.

Returns

(M & __m) != 0.

Implemented in [std::ctype<wchar_t>](#), [std::ctype<_CharT>](#), and [std::ctype<char_type>](#).

Referenced by [std::__ctype_abstract_base<wchar_t>::is\(\)](#), and [std::__ctype_abstract_base<wchar_t>::narrow\(\)](#).

4.409.3.2 `template<typename _CharT> virtual const char_type* std::__ctype_abstract_base<_CharT>::do_is (const char_type * __lo, const char_type * __hi, mask * __vec) const` [protected], [pure virtual]

Return a mask array.

This function finds the mask for each char_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the input.

do_is() is a hook for a derived facet to change the behavior of classifying. do_is() must always return the same result for the same input.

Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.
<code>__vec</code>	Pointer to an array of mask storage.

Returns

`__hi`.

Implemented in [std::ctype<wchar_t>](#), [std::ctype<_CharT>](#), and [std::ctype<char_type>](#).

4.409.3.3 `template<typename _CharT> virtual char std::__ctype_abstract_base<_CharT>::do_narrow (char_type __c, char __dfault) const` [protected], [pure virtual]

Narrow char_type to char.

This virtual function converts the argument to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead.

do_narrow() is a hook for a derived facet to change the behavior of narrowing. do_narrow() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See [codecvt](#) for that.

Parameters

<code>__c</code>	The char_type to convert.
<code>__dfault</code>	Char to return if conversion fails.

Returns

The converted char.

Implemented in [std::ctype< wchar_t >](#), [std::ctype< _CharT >](#), and [std::ctype< char_type >](#).

Referenced by `std::__ctype_abstract_base< wchar_t >::narrow()`, and `std::ctype< char >::narrow()`.

4.409.3.4 `template<typename _CharT> virtual const char_type* std::__ctype_abstract_base< _CharT >::do_narrow (const char_type * __lo, const char_type * __hi, char __default, char * __to) const` [protected], [pure virtual]

Narrow `char_type` array to `char`.

This virtual function converts each `char_type` in the range `[__lo, __hi)` to `char` using the simplest reasonable transformation and writes the results to the destination array. For any element in the input that cannot be converted, `__default` is used instead.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.
<code>__default</code>	Char to use if conversion fails.
<code>__to</code>	Pointer to the destination array.

Returns

`__hi`.

Implemented in [std::ctype< wchar_t >](#), [std::ctype< _CharT >](#), and [std::ctype< char_type >](#).

4.409.3.5 `template<typename _CharT> virtual const char_type* std::__ctype_abstract_base< _CharT >::do_scan_is (mask __m, const char_type * __lo, const char_type * __hi) const` [protected], [pure virtual]

Find `char_type` matching mask.

This function searches for and returns the first `char_type` `c` in `[__lo, __hi)` for which `is(__m, c)` is true.

`do_scan_is()` is a hook for a derived facet to change the behavior of match searching. `do_is()` must always return the same result for the same input.

Parameters

<code>__m</code>	The mask to compare against.
<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.

Returns

Pointer to a matching char_type if found, else __hi.

Implemented in [std::ctype<wchar_t>](#), [std::ctype<_CharT>](#), and [std::ctype<char_type>](#).

Referenced by [std::__ctype_abstract_base<wchar_t>::narrow\(\)](#), and [std::__ctype_abstract_base<wchar_t>::scan_is\(\)](#).

4.409.3.6 `template<typename _CharT> virtual const char_type* std::__ctype_abstract_base<_CharT>::do_scan_not (mask __m, const char_type* __lo, const char_type* __hi) const` [protected], [pure virtual]

Find char_type not matching mask.

This function searches for and returns a pointer to the first char_type c of [lo,hi) for which is(m,c) is false.

do_scan_is() is a hook for a derived facet to change the behavior of match searching. do_is() must always return the same result for the same input.

Parameters

<code>__m</code>	The mask to compare against.
<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.

Returns

Pointer to a non-matching char_type if found, else __hi.

Implemented in [std::ctype<wchar_t>](#), [std::ctype<_CharT>](#), and [std::ctype<char_type>](#).

Referenced by [std::__ctype_abstract_base<wchar_t>::narrow\(\)](#), and [std::__ctype_abstract_base<wchar_t>::scan_not\(\)](#).

4.409.3.7 `template<typename _CharT> virtual char_type std::__ctype_abstract_base<_CharT>::do_tolower (char_type __c) const` [protected], [pure virtual]

Convert to lowercase.

This virtual function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

do_tolower() is a hook for a derived facet to change the behavior of lowercasing. do_tolower() must always return the same result for the same input.

Parameters

\leftrightarrow _c	The char_type to convert.
-------------------------	---------------------------

Returns

The lowercase char_type if convertible, else __c.

Implemented in [std::ctype< wchar_t >](#), [std::ctype< _CharT >](#), and [std::ctype< char_type >](#).

Referenced by [std::__ctype_abstract_base< wchar_t >::narrow\(\)](#), [std::ctype< char >::table\(\)](#), [std::__ctype_abstract_base< wchar_t >::tolower\(\)](#), and [std::ctype< char >::tolower\(\)](#).

4.409.3.8 `template<typename _CharT> virtual const char_type* std::__ctype_abstract_base< _CharT >::do_tolower (char_type * __lo, const char_type * __hi) const` [protected], [pure virtual]

Convert array to lowercase.

This virtual function converts each char_type in the range [[__lo](#),[__hi](#)) to lowercase if possible. Other elements remain untouched.

[do_tolower\(\)](#) is a hook for a derived facet to change the behavior of lowercasing. [do_tolower\(\)](#) must always return the same result for the same input.

Parameters

\leftrightarrow _lo	Pointer to start of range.
\leftrightarrow _hi	Pointer to end of range.

Returns

[__hi](#).

Implemented in [std::ctype< wchar_t >](#), [std::ctype< _CharT >](#), and [std::ctype< char_type >](#).

4.409.3.9 `template<typename _CharT> virtual char_type std::__ctype_abstract_base< _CharT >::do_toupper (char_type __c) const` [protected], [pure virtual]

Convert to uppercase.

This virtual function converts the char_type argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

[do_toupper\(\)](#) is a hook for a derived facet to change the behavior of uppercasing. [do_toupper\(\)](#) must always return the same result for the same input.

Parameters

\leftrightarrow _c	The char_type to convert.
-------------------------	---------------------------

Returns

The uppercase char_type if convertible, else __c.

Implemented in [std::ctype<wchar_t>](#), [std::ctype<_CharT>](#), and [std::ctype<char_type>](#).

Referenced by [std::__ctype_abstract_base<wchar_t>::narrow\(\)](#), [std::ctype<char>::table\(\)](#), [std::__ctype_abstract_base<wchar_t>::toupper\(\)](#), and [std::ctype<char>::toupper\(\)](#).

4.409.3.10 `template<typename _CharT> virtual const char_type* std::__ctype_abstract_base<_CharT>::do_toupper (char_type * __lo, const char_type * __hi) const` [protected], [pure virtual]

Convert array to uppercase.

This virtual function converts each char_type in the range [`__lo`,`__hi`) to uppercase if possible. Other elements remain untouched.

`do_toupper()` is a hook for a derived facet to change the behavior of uppercasing. `do_toupper()` must always return the same result for the same input.

Parameters

\leftrightarrow __lo	Pointer to start of range.
\leftrightarrow __hi	Pointer to end of range.

Returns

`__hi`.

Implemented in [std::ctype<wchar_t>](#), [std::ctype<_CharT>](#), and [std::ctype<char_type>](#).

4.409.3.11 `template<typename _CharT> virtual char_type std::__ctype_abstract_base<_CharT>::do_widen (char __c) const` [protected], [pure virtual]

Widen char.

This virtual function converts the char to char_type using the simplest reasonable transformation.

`do_widen()` is a hook for a derived facet to change the behavior of widening. `do_widen()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

<code>_↵</code>	The char to convert.
<code>_c</code>	

Returns

The converted `char_type`

Implemented in [std::ctype< wchar_t >](#), [std::ctype< _CharT >](#), and [std::ctype< char_type >](#).

Referenced by `std::__ctype_abstract_base< wchar_t >::narrow()`, `std::__ctype_abstract_base< wchar_t >::widen()`, and `std::ctype< char >::widen()`.

```
4.409.3.12 template<typename _CharT> virtual const char* std::__ctype_abstract_base< _CharT >::do_widen ( const char
* __lo, const char * __hi, char_type * __to ) const [protected], [pure virtual]
```

Widen char array.

This function converts each char in the input to `char_type` using the simplest reasonable transformation.

`do_widen()` is a hook for a derived facet to change the behavior of widening. `do_widen()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

<code>_↵</code> <code>_lo</code>	Pointer to start range.
<code>_↵</code> <code>_hi</code>	Pointer to end of range.
<code>_↵</code> <code>_to</code>	Pointer to the destination array.

Returns

`__hi`.

Implemented in [std::ctype< wchar_t >](#), [std::ctype< _CharT >](#), and [std::ctype< char_type >](#).

```
4.409.3.13 template<typename _CharT> bool std::__ctype_abstract_base< _CharT >::is ( mask __m, char_type __c )
const [inline]
```

Test `char_type` classification.

This function finds a mask `M` for `__c` and compares it to mask `__m`. It does so by returning the value of `ctype<char_↵ type>::do_is()`.

Parameters

<code>__c</code>	The char_type to compare the mask of.
<code>__m</code>	The mask to compare against.

Returns

`(M & __m) != 0.`

Definition at line 162 of file locale_facets.h.

Referenced by `std::ctype<char>::is()`, `std::isalnum()`, `std::isalpha()`, `std::iscntrl()`, `std::isdigit()`, `std::isgraph()`, `std::islower()`, `std::isprint()`, `std::ispunct()`, `std::isspace()`, `std::isupper()`, and `std::isxdigit()`.

4.409.3.14 `template<typename _CharT> const char_type* std::__ctype_abstract_base<_CharT>::is (const char_type * __lo, const char_type * __hi, mask * __vec) const [inline]`

Return a mask array.

This function finds the mask for each char_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the char array. It does so by returning the value of `ctype<char_type>::do_is()`.

Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.
<code>__vec</code>	Pointer to an array of mask storage.

Returns

`__hi.`

Definition at line 179 of file locale_facets.h.

4.409.3.15 `template<typename _CharT> char std::__ctype_abstract_base<_CharT>::narrow (char_type __c, char __dfault) const [inline]`

Narrow char_type to char.

This function converts the char_type to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead. It does so by returning `ctype<char_type>::do_narrow(__c)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

<code>__c</code>	The <code>char_type</code> to convert.
<code>__dfault</code>	Char to return if conversion fails.

Returns

The converted char.

Definition at line 324 of file `locale_facets.h`.

```
4.409.3.16 template<typename _CharT> const char_type* std::__ctype_abstract_base<_CharT>::narrow ( const
char_type * __lo, const char_type * __hi, char __dfault, char * __to ) const [inline]
```

Narrow array to char array.

This function converts each `char_type` in the input to `char` using the simplest reasonable transformation and writes the results to the destination array. For any `char_type` in the input that cannot be converted, `dfault` is used instead. It does so by returning `ctype<char_type>::do_narrow(__lo, __hi, __dfault, __to)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.
<code>__dfault</code>	Char to use if conversion fails.
<code>__to</code>	Pointer to the destination array.

Returns

`__hi`.

Definition at line 346 of file `locale_facets.h`.

```
4.409.3.17 template<typename _CharT> const char_type* std::__ctype_abstract_base<_CharT>::scan_is ( mask __m,
const char_type * __lo, const char_type * __hi ) const [inline]
```

Find `char_type` matching a mask.

This function searches for and returns the first `char_type` `c` in `[lo,hi)` for which `is(m,c)` is true. It does so by returning `ctype<char_type>::do_scan_is()`.

Parameters

<code>__m</code>	The mask to compare against.
<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.

Returns

Pointer to matching char_type if found, else __hi.

Definition at line 195 of file locale_facets.h.

Referenced by std::ctype<char>::is().

4.409.3.18 `template<typename _CharT> const char_type* std::__ctype_abstract_base<_CharT>::scan_not (mask __m,
const char_type * __lo, const char_type * __hi) const [inline]`

Find char_type not matching a mask.

This function searches for and returns the first char_type c in [lo,hi) for which is(m,c) is false. It does so by returning ctype<char_type>::do_scan_not().

Parameters

<code>__m</code>	The mask to compare against.
<code>__lo</code>	Pointer to first char in range.
<code>__hi</code>	Pointer to end of range.

Returns

Pointer to non-matching char if found, else __hi.

Definition at line 211 of file locale_facets.h.

Referenced by std::ctype<char>::scan_is().

4.409.3.19 `template<typename _CharT> char_type std::__ctype_abstract_base<_CharT>::tolower (char_type __c)
const [inline]`

Convert to lowercase.

This function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning ctype<char_type>::do_tolower(c).

Parameters

<code>__c</code>	The char_type to convert.
------------------	---------------------------

Returns

The lowercase `char_type` if convertible, else `__c`.

Definition at line 254 of file `locale_facets.h`.

Referenced by `std::tolower()`.

4.409.3.20 `template<typename _CharT> const char_type* std::__ctype_abstract_base<_CharT>::tolower (char_type
*__lo, const char_type * __hi) const [inline]`

Convert array to lowercase.

This function converts each `char_type` in the range `[__lo,__hi)` to lowercase if possible. Other elements remain untouched. It does so by returning `ctype<char_type>::do_tolower(__lo, __hi)`.

Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.

Returns

`__hi`.

Definition at line 269 of file `locale_facets.h`.

4.409.3.21 `template<typename _CharT> char_type std::__ctype_abstract_base<_CharT>::toupper (char_type __c)
const [inline]`

Convert to uppercase.

This function converts the argument to uppercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning `ctype<char_type>::do_toupper()`.

Parameters

<code>__c</code>	The <code>char_type</code> to convert.
------------------	--

Returns

The uppercase `char_type` if convertible, else `__c`.

Definition at line 225 of file `locale_facets.h`.

Referenced by `std::toupper()`.

4.409.3.22 `template<typename _CharT> const char_type* std::__ctype_abstract_base<_CharT>::toupper (char_type
* __lo, const char_type * __hi) const [inline]`

Convert array to uppercase.

This function converts each char_type in the range [lo,hi) to uppercase if possible. Other elements remain untouched. It does so by returning ctype<char_type>::do_toupper(lo, hi).

Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.

Returns

`__hi`.

Definition at line 240 of file locale_facets.h.

4.409.3.23 `template<typename _CharT> char_type std::__ctype_abstract_base<_CharT>::widen (char __c) const
[inline]`

Widen char to char_type.

This function converts the char argument to char_type using the simplest reasonable transformation. It does so by returning ctype<char_type>::do_widen(c).

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

<code>__c</code>	The char to convert.
------------------	----------------------

Returns

The converted char_type.

Definition at line 286 of file locale_facets.h.

4.409.3.24 `template<typename _CharT> const char* std::__ctype_abstract_base<_CharT>::widen (const char * __lo,
const char * __hi, char_type * __to) const [inline]`

Widen array to char_type.

This function converts each char in the input to char_type using the simplest reasonable transformation. It does so by returning ctype<char_type>::do_widen(c).

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

<code>_↔ _lo</code>	Pointer to start of range.
<code>_↔ _hi</code>	Pointer to end of range.
<code>_↔ _to</code>	Pointer to the destination array.

Returns

`__hi`.

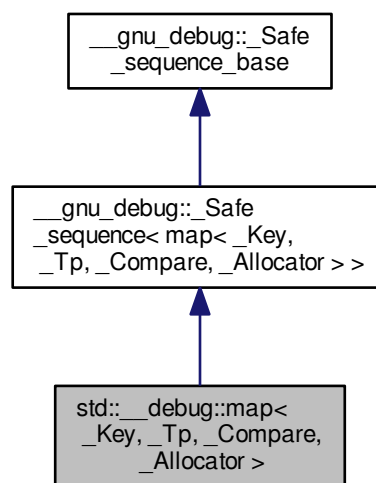
Definition at line 305 of file `locale_facets.h`.

The documentation for this class was generated from the following file:

- [locale_facets.h](#)

4.410 `std::__debug::map< _Key, _Tp, _Compare, _Allocator >` Class Template Reference

Inheritance diagram for `std::__debug::map< _Key, _Tp, _Compare, _Allocator >`:



Public Types

- typedef `_Allocator` **allocator_type**
- typedef `__gnu_debug::Safe_iterator<_Base_const_iterator, map>` **const_iterator**
- typedef `_Base::const_pointer` **const_pointer**
- typedef `_Base::const_reference` **const_reference**
- typedef `std::reverse_iterator<const_iterator>` **const_reverse_iterator**
- typedef `_Base::difference_type` **difference_type**
- typedef `__gnu_debug::Safe_iterator<_Base_iterator, map>` **iterator**
- typedef `_Compare` **key_compare**
- typedef `_Key` **key_type**
- typedef `_Tp` **mapped_type**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `std::reverse_iterator<iterator>` **reverse_iterator**
- typedef `_Base::size_type` **size_type**
- typedef `std::pair<const_Key, _Tp>` **value_type**

Public Member Functions

- **map** (const `_Compare` &__comp, const `_Allocator` &__a=_Allocator())
- template<typename `_InputIterator` >
map (`_InputIterator` __first, `_InputIterator` __last, const `_Compare` &__comp=_Compare(), const `_Allocator` &__a=_Allocator())
- **map** (const `map` &__x)
- **map** (const `_Base` &__x)
- **map** (`map` &&__x) noexcept(is_nothrow_copy_constructible<`_Compare`>::value)
- **map** (initializer_list< `value_type` > __l, const `_Compare` &__c=_Compare(), const `allocator_type` &__a=allocator_type())
- **map** (const `allocator_type` &__a)
- **map** (const `map` &__m, const `allocator_type` &__a)
- **map** (`map` &&__m, const `allocator_type` &__a)
- **map** (initializer_list< `value_type` > __l, const `allocator_type` &__a)
- template<typename `_InputIterator` >
map (`_InputIterator` __first, `_InputIterator` __last, const `allocator_type` &__a)
- void **_M_attach** (`_Safe_iterator_base` *__it, bool __constant)
- void **_M_attach_single** (`_Safe_iterator_base` *__it, bool __constant) throw ()
- `_Base` & **_M_base** () noexcept
- const `_Base` & **_M_base** () const noexcept
- void **_M_detach** (`_Safe_iterator_base` *__it)
- void **_M_detach_single** (`_Safe_iterator_base` *__it) throw ()
- void **_M_invalidate_all** () const
- void **_M_invalidate_if** (`_Predicate` __pred)
- void **_M_transfer_from_if** (`_Safe_sequence` &__from, `_Predicate` __pred)
- `iterator` **begin** () noexcept
- `const_iterator` **begin** () const noexcept
- `const_iterator` **cbegin** () const noexcept
- `const_iterator` **cend** () const noexcept
- void **clear** () noexcept
- `const_reverse_iterator` **crbegin** () const noexcept

- `const_reverse_iterator` **crend** () const noexcept
- `template<typename... _Args>`
`std::pair< iterator, bool > emplace (_Args &&...__args)`
- `template<typename... _Args>`
`iterator emplace_hint (const_iterator __pos, _Args &&...__args)`
- `iterator` **end** () noexcept
- `const_iterator` **end** () const noexcept
- `std::pair< iterator, iterator > equal_range (const key_type &__x)`
- `std::pair< const_iterator, const_iterator > equal_range (const key_type &__x) const`
- `iterator` **erase** (const_iterator __position)
- `iterator` **erase** (iterator __position)
- `size_type` **erase** (const key_type &__x)
- `iterator` **erase** (const_iterator __first, const_iterator __last)
- `iterator` **find** (const key_type &__x)
- `const_iterator` **find** (const key_type &__x) const
- `std::pair< iterator, bool > insert (const value_type &__x)`
- `template<typename _Pair, typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type>`
`std::pair< iterator, bool > insert (_Pair &&__x)`
- `void insert (std::initializer_list< value_type > __list)`
- `iterator` **insert** (const_iterator __position, const value_type &__x)
- `template<typename _Pair, typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type>`
`iterator` **insert** (const_iterator __position, _Pair &&__x)
- `template<typename _InputIterator >`
`void insert (_InputIterator __first, _InputIterator __last)`
- `iterator` **lower_bound** (const key_type &__x)
- `const_iterator` **lower_bound** (const key_type &__x) const
- `map & operator= (const map &__x)`
- `map & operator= (map &&__x) noexcept(_Alloc_traits::_S_nothrow_move())`
- `map & operator= (initializer_list< value_type > __l)`
- `reverse_iterator` **rbegin** () noexcept
- `const_reverse_iterator` **rbegin** () const noexcept
- `reverse_iterator` **rend** () noexcept
- `const_reverse_iterator` **rend** () const noexcept
- `void swap (map &__x) noexcept(_Alloc_traits::_S_nothrow_swap())`
- `iterator` **upper_bound** (const key_type &__x)
- `const_iterator` **upper_bound** (const key_type &__x) const

Public Attributes

- `_Safe_iterator_base * _M_const_iterators`
- `_Safe_iterator_base * _M_iterators`
- `unsigned int _M_version`

Protected Member Functions

- `void _M_detach_all ()`
- `void _M_detach_singular ()`
- `__gnu_cxx::__mutex & _M_get_mutex () throw ()`
- `void _M_revalidate_singular ()`
- `void _M_swap (_Safe_sequence_base &__x)`

4.410.1 Detailed Description

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Allocator = std::allocator<std::pair<const _Key, _Tp>>>
class std::__debug::map<_Key, _Tp, _Compare, _Allocator >
```

Class std::map with safety/checking/debug instrumentation.

Definition at line 43 of file debug/map.h.

4.410.2 Member Function Documentation

4.410.2.1 void __gnu_debug::Safe_sequence_base::M_attach (_Safe_iterator_base * __it, bool __constant)
[inherited]

Attach an iterator to this sequence.

4.410.2.2 void __gnu_debug::Safe_sequence_base::M_attach_single (_Safe_iterator_base * __it, bool __constant) throw)
[inherited]

Likewise but not thread safe.

4.410.2.3 void __gnu_debug::Safe_sequence_base::M_detach (_Safe_iterator_base * __it) [inherited]

Detach an iterator from this sequence

4.410.2.4 void __gnu_debug::Safe_sequence_base::M_detach_all () [protected], [inherited]

Detach all iterators, leaving them singular.

4.410.2.5 void __gnu_debug::Safe_sequence_base::M_detach_single (_Safe_iterator_base * __it) throw)
[inherited]

Likewise but not thread safe.

4.410.2.6 void __gnu_debug::Safe_sequence_base::M_detach_singular () [protected], [inherited]

Detach all singular iterators.

Postcondition

for all iterators i attached to this sequence, i->_M_version == _M_version.

4.410.2.7 __gnu_cxx::mutex& __gnu_debug::Safe_sequence_base::M_get_mutex () throw) [protected],
[inherited]

For use in _Safe_sequence.

4.410.2.8 void __gnu_debug::Safe_sequence_base::M_invalidate_all() const [inline],[inherited]

Invalidates all iterators.

Definition at line 242 of file safe_base.h.

References __gnu_debug::Safe_iterator_base::M_attach(), __gnu_debug::Safe_iterator_base::M_attach_single(), __gnu_debug::Safe_iterator_base::M_detach(), and __gnu_debug::Safe_iterator_base::M_detach_single().

4.410.2.9 void __gnu_debug::Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::M_invalidate_if(_Predicate __pred) [inherited]

Invalidates all iterators *x* that reference this sequence, are not singular, and for which __pred(*x*) returns true. __pred will be invoked with the normal iterators nested in the safe ones.

4.410.2.10 void __gnu_debug::Safe_sequence_base::M_revalidate_singular() [protected],[inherited]

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

4.410.2.11 void __gnu_debug::Safe_sequence_base::M_swap(_Safe_sequence_base & __x) [protected],[inherited]

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

4.410.2.12 void __gnu_debug::Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::M_transfer_from_if(_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > > & __from, _Predicate __pred) [inherited]

Transfers all iterators *x* that reference *from* sequence, are not singular, and for which __pred(*x*) returns true. __pred will be invoked with the normal iterators nested in the safe ones.

4.410.3 Member Data Documentation

4.410.3.1 _Safe_iterator_base* __gnu_debug::Safe_sequence_base::M_const_iterators [inherited]

The list of constant iterators that reference this container.

Definition at line 184 of file safe_base.h.

4.410.3.2 _Safe_iterator_base* __gnu_debug::Safe_sequence_base::M_iterators [inherited]

The list of mutable iterators that reference this container.

Definition at line 181 of file safe_base.h.

4.410.3.3 `unsigned int __gnu_debug::Safe_sequence_base::M_version` [mutable],[inherited]

The container version number. This number may never be 0.

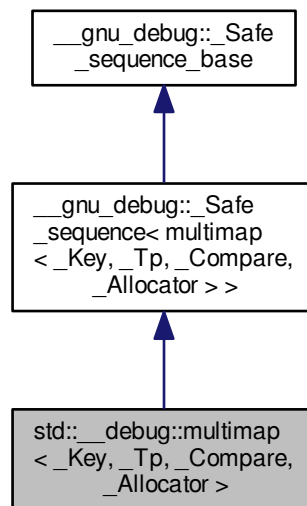
Definition at line 187 of file `safe_base.h`.

The documentation for this class was generated from the following file:

- [debug/map.h](#)

4.411 `std::__debug::multimap<_Key,_Tp,_Compare,_Allocator>` Class Template Reference

Inheritance diagram for `std::__debug::multimap<_Key,_Tp,_Compare,_Allocator>`:



Public Types

- typedef `_Allocator` **allocator_type**
- typedef `__gnu_debug::Safe_iterator<_Base_const_iterator, multimap>` **const_iterator**
- typedef `_Base::const_pointer` **const_pointer**
- typedef `_Base::const_reference` **const_reference**
- typedef `std::reverse_iterator<const_iterator>` **const_reverse_iterator**
- typedef `_Base::difference_type` **difference_type**
- typedef `__gnu_debug::Safe_iterator<_Base_iterator, multimap>` **iterator**
- typedef `_Compare` **key_compare**
- typedef `_Key` **key_type**
- typedef `_Tp` **mapped_type**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `std::reverse_iterator<iterator>` **reverse_iterator**
- typedef `_Base::size_type` **size_type**
- typedef `std::pair<const_Key,_Tp>` **value_type**

Public Member Functions

- **multimap** (const `_Compare` &__comp, const `_Allocator` &__a=_Allocator())
- template<typename `_InputIterator` >
multimap (`_InputIterator` __first, `_InputIterator` __last, const `_Compare` &__comp=_Compare(), const `_Allocator` &__a=_Allocator())
- **multimap** (const `multimap` &__x)
- **multimap** (const `_Base` &__x)
- **multimap** (`multimap` &&__x) noexcept(is_nothrow_copy_constructible< `_Compare` >::value)
- **multimap** (initializer_list< `value_type` > __l, const `_Compare` &__c=_Compare(), const `allocator_type` &__a=allocator_type())
- **multimap** (const `allocator_type` &__a)
- **multimap** (const `multimap` &__m, const `allocator_type` &__a)
- **multimap** (`multimap` &&__m, const `allocator_type` &__a)
- **multimap** (initializer_list< `value_type` > __l, const `allocator_type` &__a)
- template<typename `_InputIterator` >
multimap (`_InputIterator` __first, `_InputIterator` __last, const `allocator_type` &__a)
- void **_M_attach** (`_Safe_iterator_base` *__it, bool __constant)
- void **_M_attach_single** (`_Safe_iterator_base` *__it, bool __constant) throw ()
- **_Base** & **_M_base** () noexcept
- const **_Base** & **_M_base** () const noexcept
- void **_M_detach** (`_Safe_iterator_base` *__it)
- void **_M_detach_single** (`_Safe_iterator_base` *__it) throw ()
- void **_M_invalidate_all** () const
- void **_M_invalidate_if** (`_Predicate` __pred)
- void **_M_transfer_from_if** (`_Safe_sequence` &__from, `_Predicate` __pred)
- **iterator begin** () noexcept
- const **iterator begin** () const noexcept
- const **iterator cbegin** () const noexcept
- const **iterator cend** () const noexcept
- void **clear** () noexcept
- const **reverse_iterator crbegin** () const noexcept
- const **reverse_iterator crend** () const noexcept
- template<typename... `_Args` >
iterator emplace (`_Args` &&... __args)
- template<typename... `_Args` >
iterator emplace_hint (const **iterator** __pos, `_Args` &&... __args)
- **iterator end** () noexcept
- const **iterator end** () const noexcept
- std::pair< **iterator**, **iterator** > **equal_range** (const `key_type` &__x)
- std::pair< const **iterator**, const **iterator** > **equal_range** (const `key_type` &__x) const
- **iterator erase** (const **iterator** __position)
- **iterator erase** (**iterator** __position)
- `size_type` **erase** (const `key_type` &__x)
- **iterator erase** (const **iterator** __first, const **iterator** __last)
- **iterator find** (const `key_type` &__x)
- const **iterator find** (const `key_type` &__x) const
- **iterator insert** (const `value_type` &__x)
- template<typename `_Pair` , typename = typename std::enable_if<std::is_constructible< `value_type`, `_Pair`&& >::value>::type>
iterator insert (`_Pair` &&__x)
- void **insert** (std::initializer_list< `value_type` > __list)

- [iterator insert](#) ([const_iterator](#) __position, [const value_type](#) &__x)
- [template<typename _Pair, typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type>](#)
[iterator insert](#) ([const_iterator](#) __position, [_Pair](#) &&__x)
- [template<typename _InputIterator >](#)
[void insert](#) ([_InputIterator](#) __first, [_InputIterator](#) __last)
- [iterator lower_bound](#) ([const key_type](#) &__x)
- [const_iterator lower_bound](#) ([const key_type](#) &__x) [const](#)
- [multimap & operator=](#) ([const multimap](#) &__x)
- [multimap & operator=](#) ([multimap](#) &&__x) [noexcept\(_Alloc_traits::S_nothrow_move\(\)\)](#)
- [multimap & operator=](#) ([initializer_list](#)< [value_type](#) > __l)
- [reverse_iterator rbegin](#) () [noexcept](#)
- [const_reverse_iterator rbegin](#) () [const](#) [noexcept](#)
- [reverse_iterator rend](#) () [noexcept](#)
- [const_reverse_iterator rend](#) () [const](#) [noexcept](#)
- [void swap](#) ([multimap](#) &__x) [noexcept\(_Alloc_traits::S_nothrow_swap\(\)\)](#)
- [iterator upper_bound](#) ([const key_type](#) &__x)
- [const_iterator upper_bound](#) ([const key_type](#) &__x) [const](#)

Public Attributes

- [_Safe_iterator_base](#) * [_M_const_iterators](#)
- [_Safe_iterator_base](#) * [_M_iterators](#)
- [unsigned int](#) [_M_version](#)

Protected Member Functions

- [void](#) [_M_detach_all](#) ()
- [void](#) [_M_detach_singular](#) ()
- [__gnu_cxx::__mutex & _M_get_mutex](#) () [throw \(\)](#)
- [void](#) [_M_revalidate_singular](#) ()
- [void](#) [_M_swap](#) ([_Safe_sequence_base](#) &__x)

4.411.1 Detailed Description

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Allocator = std::allocator<std::pair<const _Key, _Tp>>>
class std::__debug::multimap< _Key, _Tp, _Compare, _Allocator >
```

Class std::multimap with safety/checking/debug instrumentation.

Definition at line 43 of file debug/multimap.h.

4.411.2 Member Function Documentation

4.411.2.1 [void](#) [__gnu_debug::Safe_sequence_base::M_attach](#) ([_Safe_iterator_base](#) * __it, [bool](#) __constant)
[[inherited](#)]

Attach an iterator to this sequence.

4.411.2.2 void __gnu_debug::Safe_sequence_base::M_attach_single (_Safe_iterator_base * __it, bool __constant) throw)
[inherited]

Likewise but not thread safe.

4.411.2.3 void __gnu_debug::Safe_sequence_base::M_detach (_Safe_iterator_base * __it) [inherited]

Detach an iterator from this sequence

4.411.2.4 void __gnu_debug::Safe_sequence_base::M_detach_all () [protected],[inherited]

Detach all iterators, leaving them singular.

4.411.2.5 void __gnu_debug::Safe_sequence_base::M_detach_single (_Safe_iterator_base * __it) throw)
[inherited]

Likewise but not thread safe.

4.411.2.6 void __gnu_debug::Safe_sequence_base::M_detach_singular () [protected],[inherited]

Detach all singular iterators.

Postcondition

for all iterators i attached to this sequence, i->_M_version == _M_version.

4.411.2.7 __gnu_cxx::mutex& __gnu_debug::Safe_sequence_base::M_get_mutex () throw) [protected],
[inherited]

For use in _Safe_sequence.

4.411.2.8 void __gnu_debug::Safe_sequence_base::M_invalidate_all () const [inline],[inherited]

Invalidates all iterators.

Definition at line 242 of file safe_base.h.

References __gnu_debug::Safe_iterator_base::M_attach(), __gnu_debug::Safe_iterator_base::M_attach_single(),
__gnu_debug::Safe_iterator_base::M_detach(), and __gnu_debug::Safe_iterator_base::M_detach_single().

4.411.2.9 void __gnu_debug::Safe_sequence< multimap< _Key, _Tp, _Compare, _Allocator > >::M_invalidate_if (
_Predicate __pred) [inherited]

Invalidates all iterators x that reference this sequence, are not singular, and for which __pred(x) returns true.
__pred will be invoked with the normal iterators nested in the safe ones.

4.411.2.10 void __gnu_debug::Safe_sequence_base::M_revalidate_singular () [protected],[inherited]

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

4.411.2.11 void __gnu_debug::Safe_sequence_base::M_swap (_Safe_sequence_base & __x) [protected],[inherited]

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

4.411.2.12 void __gnu_debug::Safe_sequence<multimap<_Key,_Tp,_Compare,_Allocator>>::M_transfer_from_if (_Safe_sequence<multimap<_Key,_Tp,_Compare,_Allocator>> & __from, _Predicate __pred) [inherited]

Transfers all iterators *x* that reference *from* sequence, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

4.411.3 Member Data Documentation

4.411.3.1 _Safe_iterator_base* __gnu_debug::Safe_sequence_base::M_const_iterators [inherited]

The list of constant iterators that reference this container.

Definition at line 184 of file safe_base.h.

4.411.3.2 _Safe_iterator_base* __gnu_debug::Safe_sequence_base::M_iterators [inherited]

The list of mutable iterators that reference this container.

Definition at line 181 of file safe_base.h.

4.411.3.3 unsigned int __gnu_debug::Safe_sequence_base::M_version [mutable],[inherited]

The container version number. This number may never be 0.

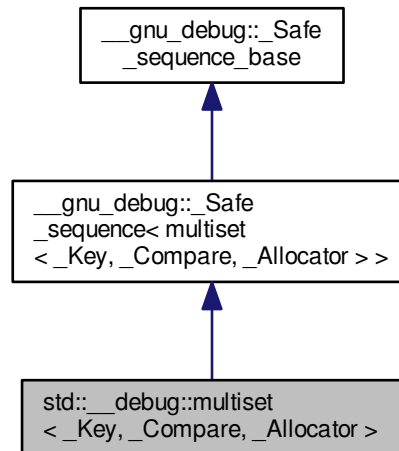
Definition at line 187 of file safe_base.h.

The documentation for this class was generated from the following file:

- [debug/multimap.h](#)

4.412 `std::__debug::multiset< _Key, _Compare, _Allocator >` Class Template Reference

Inheritance diagram for `std::__debug::multiset< _Key, _Compare, _Allocator >`:



Public Types

- typedef `_Allocator` **allocator_type**
- typedef `__gnu_debug::__Safe_iterator< _Base_const_iterator, multiset >` **const_iterator**
- typedef `_Base::const_pointer` **const_pointer**
- typedef `_Base::const_reference` **const_reference**
- typedef `std::reverse_iterator< const_iterator >` **const_reverse_iterator**
- typedef `_Base::difference_type` **difference_type**
- typedef `__gnu_debug::__Safe_iterator< _Base_iterator, multiset >` **iterator**
- typedef `_Compare` **key_compare**
- typedef `_Key` **key_type**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse_iterator**
- typedef `_Base::size_type` **size_type**
- typedef `_Compare` **value_compare**
- typedef `_Key` **value_type**

Public Member Functions

- **multiset** (`const _Compare &__comp, const _Allocator &__a=_Allocator()`)
- template<typename `_InputIterator` >
multiset (`_InputIterator __first, _InputIterator __last, const _Compare &__comp=_Compare(), const _Allocator &__a=_Allocator()`)

- **multiset** (const [multiset](#) &__x)
- **multiset** (const [_Base](#) &__x)
- **multiset** ([multiset](#) &&__x) noexcept(is_nothrow_copy_constructible<_Compare>::value)
- **multiset** (initializer_list<value_type> __l, const _Compare &__comp=_Compare(), const allocator_type &__a←a=allocator_type())
- **multiset** (const allocator_type &__a)
- **multiset** (const [multiset](#) &__m, const allocator_type &__a)
- **multiset** ([multiset](#) &&__m, const allocator_type &__a)
- **multiset** (initializer_list<value_type> __l, const allocator_type &__a)
- template<typename _InputIterator>
 multiset (_InputIterator __first, _InputIterator __last, const allocator_type &__a)
- void [_M_attach](#) (_Safe_iterator_base *__it, bool __constant)
- void [_M_attach_single](#) (_Safe_iterator_base *__it, bool __constant) throw ()
- [_Base](#) & [_M_base](#) () noexcept
- const [_Base](#) & [_M_base](#) () const noexcept
- void [_M_detach](#) (_Safe_iterator_base *__it)
- void [_M_detach_single](#) (_Safe_iterator_base *__it) throw ()
- void [_M_invalidate_all](#) () const
- void [_M_invalidate_if](#) (_Predicate __pred)
- void [_M_transfer_from_if](#) (_Safe_sequence &__from, _Predicate __pred)
- [iterator](#) **begin** () noexcept
- [const_iterator](#) **begin** () const noexcept
- [const_iterator](#) **cbegin** () const noexcept
- [const_iterator](#) **cend** () const noexcept
- void **clear** () noexcept
- [const_reverse_iterator](#) **crbegin** () const noexcept
- [const_reverse_iterator](#) **crend** () const noexcept
- template<typename... _Args>
 [iterator](#) **emplace** (_Args &&... __args)
- template<typename... _Args>
 [iterator](#) **emplace_hint** ([const_iterator](#) __pos, _Args &&... __args)
- [iterator](#) **end** () noexcept
- [const_iterator](#) **end** () const noexcept
- [std::pair](#)<[iterator](#), [iterator](#)> **equal_range** (const key_type &__x)
- [std::pair](#)<[const_iterator](#), [const_iterator](#)> **equal_range** (const key_type &__x) const
- [iterator](#) **erase** ([const_iterator](#) __position)
- size_type **erase** (const key_type &__x)
- [iterator](#) **erase** ([const_iterator](#) __first, [const_iterator](#) __last)
- [iterator](#) **find** (const key_type &__x)
- [const_iterator](#) **find** (const key_type &__x) const
- [iterator](#) **insert** (const value_type &__x)
- [iterator](#) **insert** (value_type &&__x)
- [iterator](#) **insert** ([const_iterator](#) __position, const value_type &__x)
- [iterator](#) **insert** ([const_iterator](#) __position, value_type &&__x)
- template<typename _InputIterator>
 void **insert** (_InputIterator __first, _InputIterator __last)
- void **insert** (initializer_list<value_type> __l)
- [iterator](#) **lower_bound** (const key_type &__x)
- [const_iterator](#) **lower_bound** (const key_type &__x) const
- [multiset](#) & **operator=** (const [multiset](#) &__x)
- [multiset](#) & **operator=** ([multiset](#) &&__x) noexcept(_Alloc_traits::_S_nothrow_move())

- `multiset` & `operator=` (initializer_list< value_type > __l)
- `reverse_iterator` `rbegin` () noexcept
- `const_reverse_iterator` `rbegin` () const noexcept
- `reverse_iterator` `rend` () noexcept
- `const_reverse_iterator` `rend` () const noexcept
- void `swap` (`multiset` &__x) noexcept(_Alloc_traits::S_nothrow_swap())
- `iterator` `upper_bound` (const key_type &__x)
- `const_iterator` `upper_bound` (const key_type &__x) const

Public Attributes

- `_Safe_iterator_base` * `_M_const_iterators`
- `_Safe_iterator_base` * `_M_iterators`
- unsigned int `_M_version`

Protected Member Functions

- void `_M_detach_all` ()
- void `_M_detach_singular` ()
- `__gnu_cxx::__mutex` & `_M_get_mutex` () throw ()
- void `_M_revalidate_singular` ()
- void `_M_swap` (_Safe_sequence_base &__x)

4.412.1 Detailed Description

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Allocator = std::allocator<_Key>>
class std::__debug::multiset< _Key, _Compare, _Allocator >
```

Class std::multiset with safety/checking/debug instrumentation.

Definition at line 43 of file debug/multiset.h.

4.412.2 Member Function Documentation

4.412.2.1 void `__gnu_debug::Safe_sequence_base::M_attach` (`_Safe_iterator_base` * __it, bool __constant)
[inherited]

Attach an iterator to this sequence.

4.412.2.2 void `__gnu_debug::Safe_sequence_base::M_attach_single` (`_Safe_iterator_base` * __it, bool __constant) throw()
[inherited]

Likewise but not thread safe.

4.412.2.3 void __gnu_debug::Safe_sequence_base::M_detach (_Safe_iterator_base * __it) [inherited]

Detach an iterator from this sequence

4.412.2.4 void __gnu_debug::Safe_sequence_base::M_detach_all () [protected],[inherited]

Detach all iterators, leaving them singular.

4.412.2.5 void __gnu_debug::Safe_sequence_base::M_detach_single (_Safe_iterator_base * __it) throw ()
[inherited]

Likewise but not thread safe.

4.412.2.6 void __gnu_debug::Safe_sequence_base::M_detach_singular () [protected],[inherited]

Detach all singular iterators.

Postcondition

for all iterators i attached to this sequence, i->_M_version == _M_version.

4.412.2.7 __gnu_cxx::mutex& __gnu_debug::Safe_sequence_base::M_get_mutex () throw () [protected],
[inherited]

For use in _Safe_sequence.

4.412.2.8 void __gnu_debug::Safe_sequence_base::M_invalidate_all () const [inline],[inherited]

Invalidates all iterators.

Definition at line 242 of file safe_base.h.

References __gnu_debug::Safe_iterator_base::M_attach(), __gnu_debug::Safe_iterator_base::M_attach_single(), __gnu_debug::Safe_iterator_base::M_detach(), and __gnu_debug::Safe_iterator_base::M_detach_single().

4.412.2.9 void __gnu_debug::Safe_sequence< multiset<_Key,_Compare,_Allocator> >::M_invalidate_if (_Predicate __pred) [inherited]

Invalidates all iterators x that reference this sequence, are not singular, and for which __pred(x) returns true. __pred will be invoked with the normal iterators nested in the safe ones.

4.412.2.10 void __gnu_debug::Safe_sequence_base::M_revalidate_singular () [protected],[inherited]

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

4.412.2.11 `void __gnu_debug::Safe_sequence_base::M_swap (_Safe_sequence_base & __x)` [protected],
[inherited]

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

4.412.2.12 `void __gnu_debug::Safe_sequence< multiset< _Key, _Compare, _Allocator > >::M_transfer_from_if (`
`_Safe_sequence< multiset< _Key, _Compare, _Allocator > > & __from, _Predicate __pred)` [inherited]

Transfers all iterators `x` that reference `from` sequence, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

4.412.3 Member Data Documentation

4.412.3.1 `_Safe_iterator_base* __gnu_debug::Safe_sequence_base::M_const_iterators` [inherited]

The list of constant iterators that reference this container.

Definition at line 184 of file `safe_base.h`.

4.412.3.2 `_Safe_iterator_base* __gnu_debug::Safe_sequence_base::M_iterators` [inherited]

The list of mutable iterators that reference this container.

Definition at line 181 of file `safe_base.h`.

4.412.3.3 `unsigned int __gnu_debug::Safe_sequence_base::M_version` [mutable], [inherited]

The container version number. This number may never be 0.

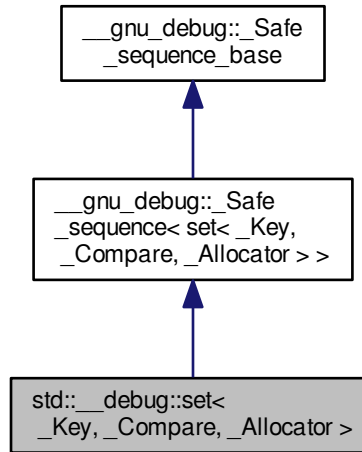
Definition at line 187 of file `safe_base.h`.

The documentation for this class was generated from the following file:

- [debug/multiset.h](#)

4.413 std::__debug::set< _Key, _Compare, _Allocator > Class Template Reference

Inheritance diagram for std::__debug::set< _Key, _Compare, _Allocator >:



Public Types

- typedef _Allocator **allocator_type**
- typedef __gnu_debug::__Safe_iterator< _Base_const_iterator, set > **const_iterator**
- typedef _Base::const_pointer **const_pointer**
- typedef _Base::const_reference **const_reference**
- typedef std::reverse_iterator< const_iterator > **const_reverse_iterator**
- typedef _Base::difference_type **difference_type**
- typedef __gnu_debug::__Safe_iterator< _Base_iterator, set > **iterator**
- typedef _Compare **key_compare**
- typedef _Key **key_type**
- typedef _Base::pointer **pointer**
- typedef _Base::reference **reference**
- typedef std::reverse_iterator< iterator > **reverse_iterator**
- typedef _Base::size_type **size_type**
- typedef _Compare **value_compare**
- typedef _Key **value_type**

Public Member Functions

- **set** (const _Compare &__comp, const _Allocator &__a=_Allocator())
- template<typename _InputIterator >
set (_InputIterator __first, _InputIterator __last, const _Compare &__comp=_Compare(), const _Allocator &__a=_Allocator())

- **set** (const [set](#) & __x)
- **set** (const [_Base](#) & __x)
- **set** ([set](#) && __x) noexcept(is_nothrow_copy_constructible< [_Compare](#) >::value)
- **set** (initializer_list< value_type > __l, const [_Compare](#) & __comp=[_Compare](#)(), const allocator_type & __a ← a=allocator_type())
- **set** (const allocator_type & __a)
- **set** (const [set](#) & __x, const allocator_type & __a)
- **set** ([set](#) && __x, const allocator_type & __a)
- **set** (initializer_list< value_type > __l, const allocator_type & __a)
- template<typename [_InputIterator](#) >
 set ([_InputIterator](#) __first, [_InputIterator](#) __last, const allocator_type & __a)
- void [_M_attach](#) ([_Safe_iterator_base](#) * __it, bool __constant)
- void [_M_attach_single](#) ([_Safe_iterator_base](#) * __it, bool __constant) throw ()
- [_Base](#) & [_M_base](#) () noexcept
- const [_Base](#) & [_M_base](#) () const noexcept
- void [_M_detach](#) ([_Safe_iterator_base](#) * __it)
- void [_M_detach_single](#) ([_Safe_iterator_base](#) * __it) throw ()
- void [_M_invalidate_all](#) () const
- void [_M_invalidate_if](#) ([_Predicate](#) __pred)
- void [_M_transfer_from_if](#) ([_Safe_sequence](#) & __from, [_Predicate](#) __pred)
- [iterator](#) **begin** () noexcept
- const_iterator **begin** () const noexcept
- const_iterator **cbegin** () const noexcept
- const_iterator **cend** () const noexcept
- void **clear** () noexcept
- const_reverse_iterator **crbegin** () const noexcept
- const_reverse_iterator **crend** () const noexcept
- template<typename... [_Args](#)>
 std::pair< [iterator](#), bool > **emplace** ([_Args](#) &&... __args)
- template<typename... [_Args](#)>
 [iterator](#) **emplace_hint** (const_iterator __pos, [_Args](#) &&... __args)
- [iterator](#) **end** () noexcept
- const_iterator **end** () const noexcept
- std::pair< [iterator](#), [iterator](#) > **equal_range** (const key_type & __x)
- std::pair< const_iterator, const_iterator > **equal_range** (const key_type & __x) const
- [iterator](#) **erase** (const_iterator __position)
- size_type **erase** (const key_type & __x)
- [iterator](#) **erase** (const_iterator __first, const_iterator __last)
- [iterator](#) **find** (const key_type & __x)
- const_iterator **find** (const key_type & __x) const
- std::pair< [iterator](#), bool > **insert** (const value_type & __x)
- std::pair< [iterator](#), bool > **insert** (value_type && __x)
- [iterator](#) **insert** (const_iterator __position, const value_type & __x)
- [iterator](#) **insert** (const_iterator __position, value_type && __x)
- template<typename [_InputIterator](#) >
 void **insert** ([_InputIterator](#) __first, [_InputIterator](#) __last)
- void **insert** (initializer_list< value_type > __l)
- [iterator](#) **lower_bound** (const key_type & __x)
- const_iterator **lower_bound** (const key_type & __x) const
- [set](#) & operator= (const [set](#) & __x)
- [set](#) & operator= ([set](#) && __x) noexcept([_Alloc_traits](#)::_S_nothrow_move())

- [set](#) & **operator=** (initializer_list< value_type > __l)
- [reverse_iterator](#) **rbegin** () noexcept
- [const_reverse_iterator](#) **rbegin** () const noexcept
- [reverse_iterator](#) **rend** () noexcept
- [const_reverse_iterator](#) **rend** () const noexcept
- void **swap** ([set](#) &__x) noexcept(_Alloc_traits::_S_nothrow_swap())
- [iterator](#) **upper_bound** (const key_type &__x)
- [const_iterator](#) **upper_bound** (const key_type &__x) const

Public Attributes

- [_Safe_iterator_base](#) * [_M_const_iterators](#)
- [_Safe_iterator_base](#) * [_M_iterators](#)
- unsigned int [_M_version](#)

Protected Member Functions

- void [_M_detach_all](#) ()
- void [_M_detach_singular](#) ()
- [__gnu_cxx::__mutex](#) & [_M_get_mutex](#) () throw ()
- void [_M_revalidate_singular](#) ()
- void [_M_swap](#) ([_Safe_sequence_base](#) &__x)

4.413.1 Detailed Description

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Allocator = std::allocator<_Key>>
class std::__debug::set< _Key, _Compare, _Allocator >
```

Class std::set with safety/checking/debug instrumentation.

Definition at line 43 of file debug/set.h.

4.413.2 Member Function Documentation

4.413.2.1 void [__gnu_debug::Safe_sequence_base::M_attach](#) ([_Safe_iterator_base](#) * __it, bool __constant)
[inherited]

Attach an iterator to this sequence.

4.413.2.2 void [__gnu_debug::Safe_sequence_base::M_attach_single](#) ([_Safe_iterator_base](#) * __it, bool __constant) throw()
[inherited]

Likewise but not thread safe.

4.413.2.3 void __gnu_debug::Safe_sequence_base::M_detach (_Safe_iterator_base * __it) [inherited]

Detach an iterator from this sequence

4.413.2.4 void __gnu_debug::Safe_sequence_base::M_detach_all () [protected],[inherited]

Detach all iterators, leaving them singular.

4.413.2.5 void __gnu_debug::Safe_sequence_base::M_detach_single (_Safe_iterator_base * __it) throw)
[inherited]

Likewise but not thread safe.

4.413.2.6 void __gnu_debug::Safe_sequence_base::M_detach_singular () [protected],[inherited]

Detach all singular iterators.

Postcondition

for all iterators i attached to this sequence, i->_M_version == _M_version.

4.413.2.7 __gnu_cxx::mutex& __gnu_debug::Safe_sequence_base::M_get_mutex () throw) [protected],
[inherited]

For use in _Safe_sequence.

4.413.2.8 void __gnu_debug::Safe_sequence_base::M_invalidate_all () const [inline],[inherited]

Invalidates all iterators.

Definition at line 242 of file safe_base.h.

References __gnu_debug::Safe_iterator_base::M_attach(), __gnu_debug::Safe_iterator_base::M_attach_single(), __gnu_debug::Safe_iterator_base::M_detach(), and __gnu_debug::Safe_iterator_base::M_detach_single().

4.413.2.9 void __gnu_debug::Safe_sequence< set< _Key, _Compare, _Allocator > >::M_invalidate_if (_Predicate
__pred) [inherited]

Invalidates all iterators x that reference this sequence, are not singular, and for which __pred(x) returns true. __pred will be invoked with the normal iterators nested in the safe ones.

4.413.2.10 void __gnu_debug::Safe_sequence_base::M_revalidate_singular () [protected],[inherited]

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

4.413.2.11 void __gnu_debug::_Safe_sequence_base::_M_swap (_Safe_sequence_base & __x) [protected],
[inherited]

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

4.413.2.12 void __gnu_debug::_Safe_sequence<set<_Key, _Compare, _Allocator >>::_M_transfer_from_if (
_Safe_sequence<set<_Key, _Compare, _Allocator >> & __from, _Predicate __pred) [inherited]

Transfers all iterators *x* that reference *from* sequence, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

4.413.3 Member Data Documentation

4.413.3.1 _Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators [inherited]

The list of constant iterators that reference this container.

Definition at line 184 of file `safe_base.h`.

4.413.3.2 _Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators [inherited]

The list of mutable iterators that reference this container.

Definition at line 181 of file `safe_base.h`.

4.413.3.3 unsigned int __gnu_debug::_Safe_sequence_base::_M_version [mutable],[inherited]

The container version number. This number may never be 0.

Definition at line 187 of file `safe_base.h`.

The documentation for this class was generated from the following file:

- [debug/set.h](#)

4.414 std::__detail::_BracketMatcher<_TraitsT, __icase, __collate > Struct Template Reference

Public Types

- typedef _TraitsT::char_class_type **_CharClassT**
- typedef _TransT::_CharT **_CharT**
- typedef _TraitsT::string_type **_StringT**
- typedef _TransT::_StrTransT **_StrTransT**
- typedef _RegexTranslator<_TraitsT, __icase, __collate > **_TransT**

Public Member Functions

- **_BracketMatcher** (bool __is_non_matching, const _TraitsT &__traits)
- void **_M_add_char** (_CharT __c)
- void **_M_add_character_class** (const _StringT &__s, bool __neg)
- _StringT **_M_add_collate_element** (const _StringT &__s)
- void **_M_add_equivalence_class** (const _StringT &__s)
- void **_M_make_range** (_CharT __l, _CharT __r)
- void **_M_ready** ()
- bool **operator()** (_CharT __ch) const

4.414.1 Detailed Description

```
template<typename _TraitsT, bool __icase, bool __collate>
struct std::__detail::_BracketMatcher< _TraitsT, __icase, __collate >
```

Matches a character range (bracket expression)

Definition at line 43 of file regex_compiler.h.

The documentation for this struct was generated from the following file:

- [regex_compiler.h](#)

4.415 std::__detail::_Compiler< _TraitsT > Class Template Reference

Public Types

- typedef _TraitsT::char_type **_CharT**
- typedef [regex_constants::syntax_option_type](#) **_FlagT**
- typedef const _CharT * **_IterT**
- typedef _NFA< _TraitsT > **_RegexT**

Public Member Functions

- **_Compiler** (_IterT __b, _IterT __e, const _TraitsT &__traits, [_FlagT](#) __flags)
- [std::shared_ptr](#)< _RegexT > **_M_get_nfa** ()

4.415.1 Detailed Description

```
template<typename _TraitsT>
class std::__detail::_Compiler< _TraitsT >
```

Builds an NFA from an input iterator interval.

The _TraitsT type should fulfill requirements [28.3].

Definition at line 51 of file regex_compiler.h.

The documentation for this class was generated from the following file:

- [regex_compiler.h](#)

4.416 std::__detail::_Default_ranged_hash Struct Reference

4.416.1 Detailed Description

Default ranged hash function H. In principle it should be a function object composed from objects of type H1 and H2 such that $h(k, N) = h2(h1(k), N)$, but that would mean making extra copies of h1 and h2. So instead we'll just use a tag to tell class template hashtable to do that composition.

Definition at line 457 of file hashtable_policy.h.

The documentation for this struct was generated from the following file:

- [hashtable_policy.h](#)

4.417 std::__detail::_Equal_helper< _Key, _Value, _ExtractKey, _Equal, _HashCodeType, __cache_hash_code > Struct Template Reference

4.417.1 Detailed Description

```
template<typename _Key, typename _Value, typename _ExtractKey, typename _Equal, typename _HashCodeType, bool __cache_hash_code>
struct std::__detail::_Equal_helper< _Key, _Value, _ExtractKey, _Equal, _HashCodeType, __cache_hash_code >
```

Primary class template _Equal_helper.

Definition at line 1316 of file hashtable_policy.h.

The documentation for this struct was generated from the following file:

- [hashtable_policy.h](#)

4.418 std::__detail::_Equal_helper< _Key, _Value, _ExtractKey, _Equal, _HashCodeType, false > Struct Template Reference

Static Public Member Functions

- static bool **_S_equals** (const _Equal &__eq, const _ExtractKey &__extract, const _Key &__k, _HashCodeType, [_Hash_node](#)< _Value, false > *__n)

4.418.1 Detailed Description

```
template<typename _Key, typename _Value, typename _ExtractKey, typename _Equal, typename _HashCodeType>
struct std::__detail::_Equal_helper< _Key, _Value, _ExtractKey, _Equal, _HashCodeType, false >
```

Specialization.

Definition at line 1332 of file hashtable_policy.h.

The documentation for this struct was generated from the following file:

- [hashtable_policy.h](#)

4.419 `std::__detail::_Equal_helper< _Key, _Value, _ExtractKey, _Equal, _HashCodeType, true >` Struct Template Reference

Static Public Member Functions

- static bool **_S_equals** (const _Equal &__eq, const _ExtractKey &__extract, const _Key &__k, _HashCodeType __c, [_Hash_node](#)< _Value, true > *__n)

4.419.1 Detailed Description

```
template<typename _Key, typename _Value, typename _ExtractKey, typename _Equal, typename _HashCodeType>
struct std::__detail::_Equal_helper< _Key, _Value, _ExtractKey, _Equal, _HashCodeType, true >
```

Specialization.

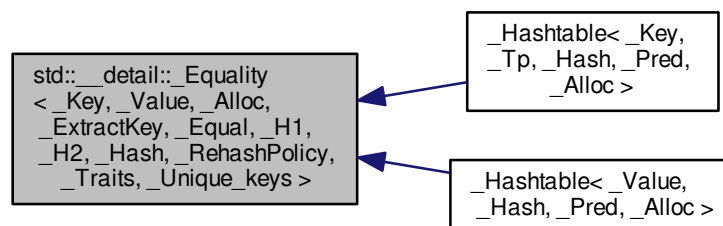
Definition at line 1321 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable_policy.h](#)

4.420 `std::__detail::_Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, _Unique_keys >` Struct Template Reference

Inheritance diagram for `std::__detail::_Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, _Unique_keys >`:



4.420.1 Detailed Description

```
template<typename _Key, typename _Value, typename _Alloc, typename _ExtractKey, typename _Equal, typename _H1, typename _H2,
typename _Hash, typename _RehashPolicy, typename _Traits, bool _Unique_keys = _Traits::__unique_keys::value>
struct std::__detail::_Equality<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, _Unique_keys >
```

Primary class template `_Equality`.

This is for implementing equality comparison for unordered containers, per N3068, by John Lakos and Pablo Halpern. Algorithmically, we follow closely the reference implementations therein.

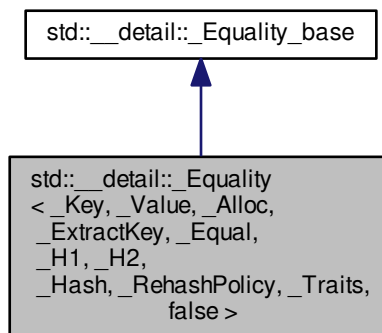
Definition at line 1796 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable_policy.h](#)

4.421 `std::__detail::_Equality<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false >` Struct Template Reference

Inheritance diagram for `std::__detail::_Equality<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false >`:



Public Types

- using `__hashtable` = `_Hashtable<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >`

Public Member Functions

- `bool _M_equal (const __hashtable &) const`

Static Protected Member Functions

- `template<typename _Uiterator >
static bool _S_is_permutation (_Uiterator, _Uiterator, _Uiterator)`

4.421.1 Detailed Description

```
template<typename _Key, typename _Value, typename _Alloc, typename _ExtractKey, typename _Equal, typename _H1, typename _H2,
typename _Hash, typename _RehashPolicy, typename _Traits>
struct std::__detail::Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false >
```

Specialization.

Definition at line 1841 of file hashtable_policy.h.

The documentation for this struct was generated from the following file:

- [hashtable_policy.h](#)

4.422 `std::__detail::Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true >` Struct Template Reference

Public Types

- `using __hashtable = _Hashtable< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >`

Public Member Functions

- `bool _M_equal (const __hashtable &) const`

4.422.1 Detailed Description

```
template<typename _Key, typename _Value, typename _Alloc, typename _ExtractKey, typename _Equal, typename _H1, typename _H2,
typename _Hash, typename _RehashPolicy, typename _Traits>
struct std::__detail::Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true >
```

Specialization.

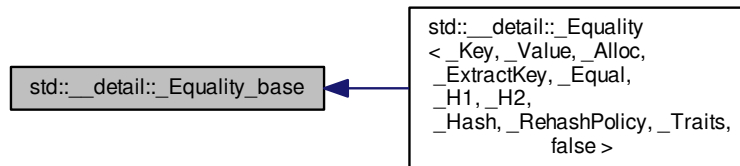
Definition at line 1803 of file hashtable_policy.h.

The documentation for this struct was generated from the following file:

- [hashtable_policy.h](#)

4.423 std::__detail::_Equality_base Struct Reference

Inheritance diagram for std::__detail::_Equality_base:



Static Protected Member Functions

- `template<typename _Uiterator >`
`static bool _S_is_permutation (_Uiterator, _Uiterator, _Uiterator)`

4.423.1 Detailed Description

struct `_Equality_base`.

Common types and functions for class `_Equality`.

Definition at line 1729 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable_policy.h](#)

4.424 std::__detail::_Executor<_Bilter, _Alloc, _TraitsT, __dfs_mode > Class Template Reference

Public Types

- `typedef iterator_traits<_Bilter>::value_type _CharT`
- `typedef _TraitsT::char_class_type _ClassT`
- `typedef regex_constants::match_flag_type _FlagT`
- `typedef _NFA<_TraitsT> _NFAT`
- `typedef basic_regex<_CharT, _TraitsT> _RegexT`
- `typedef std::vector<sub_match<_Bilter>, _Alloc> _ResultsVec`

Public Member Functions

- **_Executor** (_Bilter __begin, _Bilter __end, [_ResultsVec](#) &__results, const [_RegexT](#) &__re, [_FlagT](#) __flags)
- bool **_M_match** ()
- bool **_M_search** ()
- bool **_M_search_from_first** ()

Public Attributes

- const _Bilter **_M_begin**
- [_ResultsVec](#) **_M_cur_results**
- _Bilter **_M_current**
- const _Bilter **_M_end**
- [_FlagT](#) **_M_flags**
- bool **_M_has_sol**
- [std::unique_ptr](#)< [vector](#)< [pair](#)< _StateIdT, [_ResultsVec](#) > > > **_M_match_queue**
- const _NFAT & **_M_nfa**
- const [_RegexT](#) & **_M_re**
- [_ResultsVec](#) & **_M_results**
- _StateIdT **_M_start_state**
- [std::unique_ptr](#)< [vector](#)< bool > > **_M_visited**

4.424.1 Detailed Description

```
template<typename _Bilter, typename _Alloc, typename _TraitsT, bool __dfs_mode>
class std::__detail::_Executor< _Bilter, _Alloc, _TraitsT, __dfs_mode >
```

Takes a regex and an input string in and do the matching.

The `_Executor` class has two modes: DFS mode and BFS mode, controlled by the template parameter `__dfs_mode`.

Definition at line 61 of file `regex.h`.

The documentation for this class was generated from the following files:

- [regex.h](#)
- [regex_executor.h](#)

4.425 `std::__detail::_Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache_hash_code >` Struct Template Reference

4.425.1 Detailed Description

```
template<typename _Key, typename _Value, typename _ExtractKey, typename _H1, typename _H2, typename _Hash, bool __cache_hash_code↵
hash_code>
struct std::__detail::_Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache_hash_code >
```

Primary class template `_Hash_code_base`.

Encapsulates two policy issues that aren't quite orthogonal. (1) the difference between using a ranged hash function and using the combination of a hash function and a range-hashing function. In the former case we don't have such things as hash codes, so we have a dummy type as placeholder. (2) Whether or not we cache hash codes. Caching hash codes is meaningless if we have a ranged hash function.

We also put the key extraction objects here, for convenience. Each specialization derives from one or more of the template parameters to benefit from Ebo. This is important as this type is inherited in some cases by the `_Local_iterator_base` type used to implement `local_iterator` and `const_local_iterator`. As with any iterator type we prefer to make it as small as possible.

Primary template is unused except as a hook for specializations.

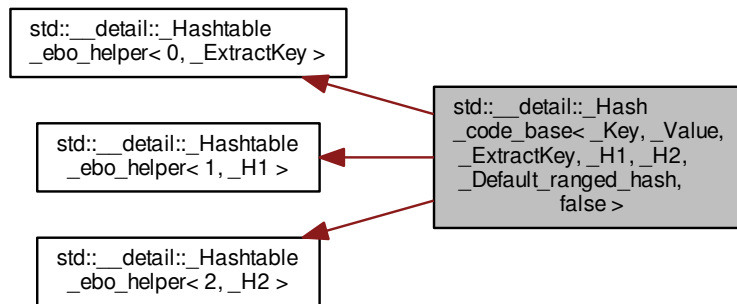
Definition at line 1057 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable_policy.h](#)

4.426 `std::__detail::_Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Default_ranged_hash, false >` Struct Template Reference

Inheritance diagram for `std::__detail::_Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Default_ranged_hash, false >`
`hash, false >`:



Public Types

- typedef `_H1 hasher`

Public Member Functions

- hasher `hash_function ()` const

Protected Types

- typedef `std::size_t __hash_code`
- typedef `_Hash_node<_Value, false> __node_type`

Protected Member Functions

- `_Hash_code_base` (`const _ExtractKey &__ex`, `const _H1 &__h1`, `const _H2 &__h2`, `const _Default_ranged_hash &`)
- `std::size_t _M_bucket_index` (`const _Key &`, `__hash_code __c`, `std::size_t __n`) const
- `std::size_t _M_bucket_index` (`const __node_type *__p`, `std::size_t __n`) const noexcept(`noexcept(declval< const _H1 &>()(declval< const _Key &>())&&noexcept(declval< const _H2 &>()(__hash_code) 0, (std::size_t) 0)))`)
- `void _M_copy_code` (`__node_type *`, `const __node_type *`) const
- `const _ExtractKey &_M_extract ()` const
- `_ExtractKey &_M_extract ()`
- `const _H1 &_M_h1 ()` const
- `_H1 &_M_h1 ()`
- `const _H2 &_M_h2 ()` const
- `_H2 &_M_h2 ()`
- `__hash_code _M_hash_code` (`const _Key &__k`) const
- `void _M_store_code` (`__node_type *`, `__hash_code`) const
- `void _M_swap` (`_Hash_code_base &__x`)

Friends

- struct `_Local_iterator_base<_Key, _Value, _ExtractKey, _H1, _H2, _Default_ranged_hash, false>`

4.426.1 Detailed Description

```
template<typename _Key, typename _Value, typename _ExtractKey, typename _H1, typename _H2>
struct std::__detail::_Hash_code_base<_Key, _Value, _ExtractKey, _H1, _H2, _Default_ranged_hash, false>
```

Specialization: hash function and range-hashing function, no caching of hash codes. Provides typedef and accessor required by C++ 11.

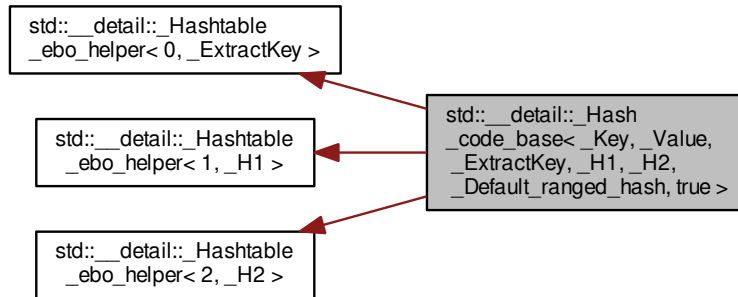
Definition at line 1139 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable_policy.h](#)

4.427 `std::__detail::__Hash_code_base<_Key,_Value,_ExtractKey,_H1,_H2,_Default_ranged_hash,true>` **Struct Template Reference**

Inheritance diagram for `std::__detail::__Hash_code_base<_Key,_Value,_ExtractKey,_H1,_H2,_Default_ranged_hash,true>`:



Public Types

- typedef `_H1 hasher`

Public Member Functions

- hasher **hash_function** () const

Protected Types

- typedef `std::size_t __hash_code`
- typedef `__Hash_node<_Value,true> __node_type`

Protected Member Functions

- **__Hash_code_base** (const `_ExtractKey` &__ex, const `_H1` &__h1, const `_H2` &__h2, const `__Default_ranged_hash` &)
- `std::size_t __M_bucket_index` (const `_Key` &, `__hash_code` __c, `std::size_t` __n) const
- `std::size_t __M_bucket_index` (const `__node_type` *__p, `std::size_t` __n) const noexcept(noexcept(declval<const `_H2` &>()((__hash_code) 0,(std::size_t) 0)))
- void **__M_copy_code** (`__node_type` *__to, const `__node_type` *__from) const
- const `_ExtractKey` & **__M_extract** () const
- `_ExtractKey` & **__M_extract** ()
- const `_H1` & **__M_h1** () const
- `_H1` & **__M_h1** ()
- const `_H2` & **__M_h2** () const
- `_H2` & **__M_h2** ()
- `__hash_code` **__M_hash_code** (const `_Key` &__k) const
- void **__M_store_code** (`__node_type` *__n, `__hash_code` __c) const
- void **__M_swap** (`__Hash_code_base` &__x)

Friends

- struct `_Local_iterator_base`< `_Key`, `_Value`, `_ExtractKey`, `_H1`, `_H2`, `_Default_ranged_hash`, `true` >

4.427.1 Detailed Description

```
template<typename _Key, typename _Value, typename _ExtractKey, typename _H1, typename _H2>
struct std::__detail::_Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Default_ranged_hash, true >
```

Specialization: hash function and range-hashing function, caching hash codes. H is provided but ignored. Provides typedef and accessor required by C++ 11.

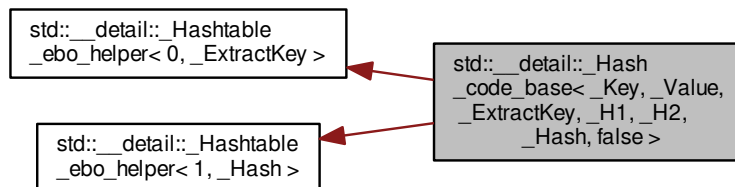
Definition at line 1228 of file hashtable_policy.h.

The documentation for this struct was generated from the following file:

- [hashtable_policy.h](#)

4.428 std::__detail::_Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, false > Struct Template Reference

Inheritance diagram for `std::__detail::_Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, false >`:



Protected Types

- typedef void * `__hash_code`
- typedef [_Hash_node](#)< `_Value`, `false` > `__node_type`

Protected Member Functions

- **_Hash_code_base** (const _ExtractKey &__ex, const _H1 &, const _H2 &, const _Hash &__h)
- **std::size_t _M_bucket_index** (const _Key &__k, __hash_code, std::size_t __n) const
- **std::size_t _M_bucket_index** (const __node_type *__p, std::size_t __n) const noexcept(noexcept(declval< const _Hash &>()(declval< const _Key &>()), (std::size_t) 0)))
- **void _M_copy_code** (__node_type *, const __node_type *) const
- **const _ExtractKey & _M_extract** () const
- **_ExtractKey & _M_extract** ()
- **__hash_code _M_hash_code** (const _Key &__key) const
- **const _Hash & _M_ranged_hash** () const
- **_Hash & _M_ranged_hash** ()
- **void _M_store_code** (__node_type *, __hash_code) const
- **void _M_swap** (_Hash_code_base &__x)

4.428.1 Detailed Description

```
template<typename _Key, typename _Value, typename _ExtractKey, typename _H1, typename _H2, typename _Hash>
struct std::__detail::_Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, false >
```

Specialization: ranged hash function, no caching hash codes. H1 and H2 are provided but ignored. We define a dummy hash code type.

Definition at line 1063 of file hashtable_policy.h.

The documentation for this struct was generated from the following file:

- [hashtable_policy.h](#)

4.429 std::__detail::_Hash_node< _Value, _Cache_hash_code > Struct Template Reference

4.429.1 Detailed Description

```
template<typename _Value, bool _Cache_hash_code>
struct std::__detail::_Hash_node< _Value, _Cache_hash_code >
```

Primary template struct _Hash_node.

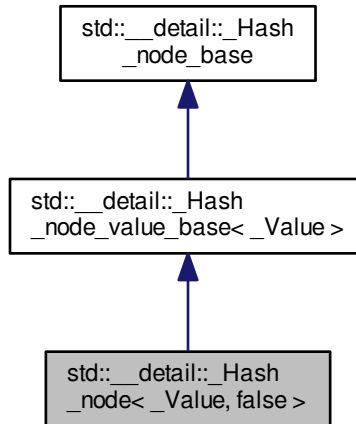
Definition at line 272 of file hashtable_policy.h.

The documentation for this struct was generated from the following file:

- [hashtable_policy.h](#)

4.430 `std::__detail::_Hash_node<_Value, false >` Struct Template Reference

Inheritance diagram for `std::__detail::_Hash_node<_Value, false >`:



Public Types

- typedef `_Value` **value_type**

Public Member Functions

- `_Hash_node * _M_next ()` const noexcept
- `_Value & _M_v ()` noexcept
- `const _Value & _M_v ()` const noexcept
- `_Value * _M_valptr ()` noexcept
- `const _Value * _M_valptr ()` const noexcept

Public Attributes

- `_Hash_node_base * _M_nxt`
- `__gnu_cxx::__aligned_buffer<_Value> _M_storage`

4.430.1 Detailed Description

```
template<typename _Value>
struct std::__detail::_Hash_node< _Value, false >
```

Specialization for nodes without caches, struct _Hash_node.

Base class is __detail::_Hash_node_value_base.

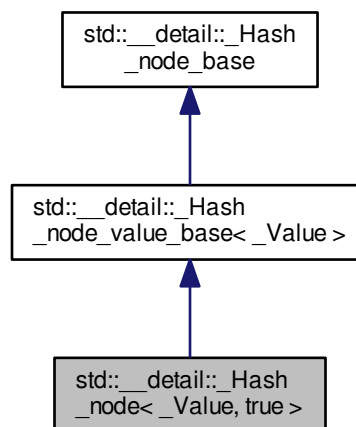
Definition at line 295 of file hashtable_policy.h.

The documentation for this struct was generated from the following file:

- [hashtable_policy.h](#)

4.431 std::__detail::_Hash_node< _Value, true > Struct Template Reference

Inheritance diagram for std::__detail::_Hash_node< _Value, true >:



Public Types

- typedef _Value **value_type**

Public Member Functions

- [_Hash_node](#) * **M_next** () const noexcept
- _Value & **M_v** () noexcept
- const _Value & **M_v** () const noexcept
- _Value * **M_valptr** () noexcept
- const _Value * **M_valptr** () const noexcept

Public Attributes

- `std::size_t _M_hash_code`
- `_Hash_node_base * _M_nxt`
- `__gnu_cxx::__aligned_buffer< _Value > _M_storage`

4.431.1 Detailed Description

```
template<typename _Value>
struct std::__detail::_Hash_node< _Value, true >
```

Specialization for nodes with caches, struct `_Hash_node`.

Base class is `__detail::_Hash_node_value_base`.

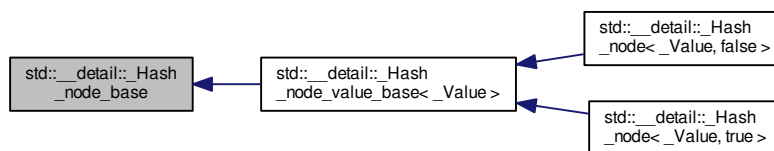
Definition at line 280 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable_policy.h](#)

4.432 `std::__detail::_Hash_node_base` Struct Reference

Inheritance diagram for `std::__detail::_Hash_node_base`:



Public Member Functions

- `_Hash_node_base (_Hash_node_base * __next) noexcept`

Public Attributes

- `_Hash_node_base * _M_nxt`

4.432.1 Detailed Description

struct _Hash_node_base

Nodes, used to wrap elements stored in the hash table. A policy template parameter of class template _Hashtable controls whether nodes also store a hash code. In some cases (e.g. strings) this may be a performance win.

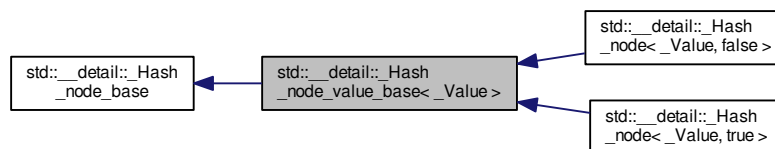
Definition at line 230 of file hashtable_policy.h.

The documentation for this struct was generated from the following file:

- [hashtable_policy.h](#)

4.433 std::__detail::__Hash_node_value_base< _Value > Struct Template Reference

Inheritance diagram for std::__detail::__Hash_node_value_base< _Value >:



Public Types

- typedef _Value **value_type**

Public Member Functions

- _Value & **_M_v** () noexcept
- const _Value & **_M_v** () const noexcept
- _Value * **_M_valptr** () noexcept
- const _Value * **_M_valptr** () const noexcept

Public Attributes

- [_Hash_node_base](#) * **_M_nxt**
- __gnu_cxx::__aligned_buffer< _Value > **_M_storage**

4.433.1 Detailed Description

```
template<typename _Value>
struct std::__detail::_Hash_node_value_base< _Value >
```

```
struct _Hash_node_value_base
```

Node type with the value to store.

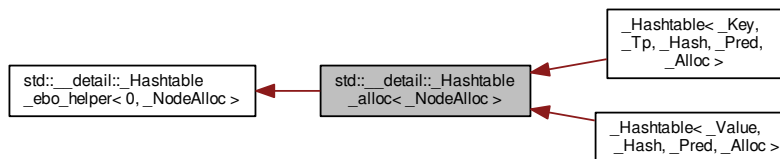
Definition at line 245 of file hashtable_policy.h.

The documentation for this struct was generated from the following file:

- [hashtable_policy.h](#)

4.434 std::__detail::_Hashtable_alloc< _NodeAlloc > Struct Template Reference

Inheritance diagram for std::__detail::_Hashtable_alloc< _NodeAlloc >:



Public Types

- using `__bucket_alloc_traits` = `std::allocator_traits< __bucket_alloc_type >`
- using `__bucket_alloc_type` = `typename __alloc_traits::rebind< __node_alloc_type, __bucket_type >::__type`
- using `__bucket_type` = `__node_base *`
- using `__node_alloc_traits` = `__gnu_cxx::__alloc_traits< __node_alloc_type >`
- using `__node_alloc_type` = `_NodeAlloc`
- using `__node_base` = `__detail::_Hash_node_base`
- using `__node_type` = `typename _NodeAlloc::value_type`
- using `__value_alloc_traits` = `std::allocator_traits< __value_alloc_type >`
- using `__value_alloc_type` = `typename __alloc_traits::rebind< __node_alloc_type, __value_type >::__type`
- using `__value_type` = `typename __node_type::value_type`

Public Member Functions

- `_Hashtable_alloc` (const `_Hashtable_alloc` &)=default
- `_Hashtable_alloc` (`_Hashtable_alloc` &&)=default
- `template<typename _Alloc > _Hashtable_alloc` (`_Alloc` &&__a)
- `__bucket_type * _M_allocate_buckets` (std::size_t __n)
- `template<typename... _Args> __node_type * _M_allocate_node` (`_Args` &&... __args)
- `void _M_deallocate_buckets` (`__bucket_type *`, std::size_t __n)
- `void _M_deallocate_node` (`__node_type *` __n)
- `void _M_deallocate_nodes` (`__node_type *` __n)
- `__node_alloc_type & _M_node_allocator` ()
- `const __node_alloc_type & _M_node_allocator` () const

4.434.1 Detailed Description

```
template<typename _NodeAlloc>
struct std::__detail::__Hashtable_alloc< _NodeAlloc >
```

This type deals with all allocation and keeps an allocator instance through inheritance to benefit from EBO when possible.

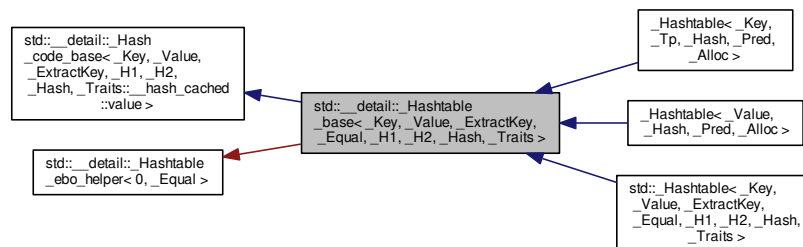
Definition at line 106 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable_policy.h](#)

4.435 `std::__detail::__Hashtable_base< _Key, _Value, _ExtractKey, _Equal, _H1, _H2, _Hash, _Traits >` Struct Template Reference

Inheritance diagram for `std::__detail::__Hashtable_base< _Key, _Value, _ExtractKey, _Equal, _H1, _H2, _Hash, _Traits >`:



Public Types

- using **__constant_iterators** = typename __traits_type::__constant_iterators
- using **__hash_cached** = typename __traits_type::__hash_cached
- using **__hash_code** = typename __hash_code_base::__hash_code
- using **__hash_code_base** = [_Hash_code_base](#)< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __hash_↵ cached::value >
- using **__ireturn_type** = typename std::conditional< __unique_keys::value, [std::pair](#)< iterator, bool >, iterator >::type
- using **__node_type** = typename __hash_code_base::__node_type
- using **__traits_type** = _Traits
- using **__unique_keys** = typename __traits_type::__unique_keys
- using **const_iterator** = [__detail::__Node_const_iterator](#)< value_type, __constant_iterators::value, __hash_↵ cached::value >
- using **const_local_iterator** = [__detail::__Local_const_iterator](#)< key_type, value_type, _ExtractKey, _H1, _H2, ↵ _Hash, __constant_iterators::value, __hash_cached::value >
- typedef std::ptrdiff_t **difference_type**
- using **iterator** = [__detail::__Node_iterator](#)< value_type, __constant_iterators::value, __hash_cached::value >
- typedef _Equal **key_equal**
- typedef _Key **key_type**
- using **local_iterator** = [__detail::__Local_iterator](#)< key_type, value_type, _ExtractKey, _H1, _H2, _Hash, __↵ constant_iterators::value, __hash_cached::value >
- typedef std::size_t **size_type**
- typedef _Value **value_type**

Protected Member Functions

- **_Hashtable_base** (const _ExtractKey &__ex, const _H1 &__h1, const _H2 &__h2, const _Hash &__hash, const _Equal &__eq)
- const _Equal & **_M_eq** () const
- _Equal & **_M_eq** ()
- bool **_M_equals** (const _Key &__k, __hash_code __c, __node_type *__n) const
- void **_M_swap** ([_Hashtable_base](#) &__x)

4.435.1 Detailed Description

template<typename _Key, typename _Value, typename _ExtractKey, typename _Equal, typename _H1, typename _H2, typename _Hash, typename _Traits>

struct std::__detail::__Hashtable_base< _Key, _Value, _ExtractKey, _Equal, _H1, _H2, _Hash, _Traits >

Primary class template `_Hashtable_base`.

Helper class adding management of `_Equal` functor to `_Hash_code_base` type.

Base class templates are:

- `__detail::__Hash_code_base`
- `__detail::__Hashtable_ebo_helper`

Definition at line 58 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable_policy.h](#)

4.436 std::__detail::_Hashtable_ebo_helper< _Nm, _Tp, __use_ebo > Struct Template Reference

4.436.1 Detailed Description

```
template<int _Nm, typename _Tp, bool __use_ebo = !_is_final(_Tp) && __is_empty(_Tp)>
struct std::__detail::_Hashtable_ebo_helper< _Nm, _Tp, __use_ebo >
```

Primary class template _Hashtable_ebo_helper.

Helper class using EBO when it is not forbidden (the type is not final) and when it is worth it (the type is empty.)

Definition at line 977 of file hashtable_policy.h.

The documentation for this struct was generated from the following file:

- [hashtable_policy.h](#)

4.437 std::__detail::_Hashtable_ebo_helper< _Nm, _Tp, false > Struct Template Reference

Public Member Functions

- template<typename _OtherTp >
 _Hashtable_ebo_helper (_OtherTp &&__tp)

Static Public Member Functions

- static const _Tp & **_S_cget** (const [_Hashtable_ebo_helper](#) &__eboh)
- static _Tp & **_S_get** ([_Hashtable_ebo_helper](#) &__eboh)

4.437.1 Detailed Description

```
template<int _Nm, typename _Tp>
struct std::__detail::_Hashtable_ebo_helper< _Nm, _Tp, false >
```

Specialization not using EBO.

Definition at line 1002 of file hashtable_policy.h.

The documentation for this struct was generated from the following file:

- [hashtable_policy.h](#)

4.438 std::__detail::_Hashtable_ebo_helper< _Nm, _Tp, true > Struct Template Reference

Inherits _Tp.

Public Member Functions

- `template<typename _OtherTp >
_Hashtable_ebo_helper (_OtherTp && __tp)`

Static Public Member Functions

- `static const _Tp & _S_cget (const _Hashtable_ebo_helper & __eboh)`
- `static _Tp & _S_get (_Hashtable_ebo_helper & __eboh)`

4.438.1 Detailed Description

```
template<int _Nm, typename _Tp>
struct std::__detail::_Hashtable_ebo_helper< _Nm, _Tp, true >
```

Specialization using EBO.

Definition at line 981 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable_policy.h](#)

4.439 `std::__detail::_Hashtable_traits< _Cache_hash_code, _Constant_iterators, _Unique_keys >` Struct Template Reference

Public Types

- `template<bool _Cond>
using __bool_constant = integral_constant< bool, _Cond >`
- `using __constant_iterators = __bool_constant< _Constant_iterators >`
- `using __hash_cached = __bool_constant< _Cache_hash_code >`
- `using __unique_keys = __bool_constant< _Unique_keys >`

4.439.1 Detailed Description

```
template<bool _Cache_hash_code, bool _Constant_iterators, bool _Unique_keys>
struct std::__detail::_Hashtable_traits< _Cache_hash_code, _Constant_iterators, _Unique_keys >
```

`struct _Hashtable_traits`

Important traits for hash tables.

Template Parameters

<code>_Cache_hash_code</code>	Boolean value. True if the value of the hash function is stored along with the value. This is a time-space tradeoff. Storing it may improve lookup speed by reducing the number of times we need to call the <code>_Equal</code> function.
<code>_Constant_iterators</code>	Boolean value. True if iterator and <code>const_iterator</code> are both constant iterator types. This is true for <code>unordered_set</code> and <code>unordered_multiset</code> , false for <code>unordered_map</code> and <code>unordered_multimap</code> .
<code>_Unique_keys</code>	Boolean value. True if the return value of <code>_Hashtable::count(k)</code> is always at most one, false if it may be an arbitrary number. This is true for <code>unordered_set</code> and <code>unordered_map</code> , false for <code>unordered_multiset</code> and <code>unordered_multimap</code> .

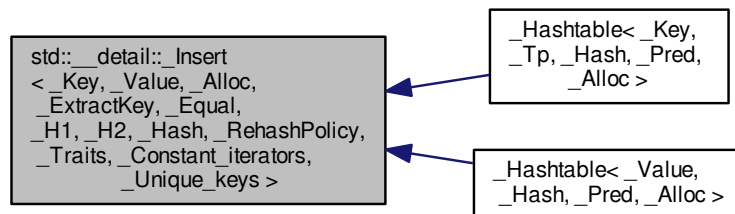
Definition at line 212 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable_policy.h](#)

4.440 `std::__detail::Insert<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, _Constant_iterators, _Unique_keys>` Struct Template Reference

Inheritance diagram for `std::__detail::Insert<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, _Constant_iterators, _Unique_keys>`:



4.440.1 Detailed Description

```

template<typename _Key, typename _Value, typename _Alloc, typename _ExtractKey, typename _Equal, typename _H1, typename _H2, typename _Hash, typename _RehashPolicy, typename _Traits, bool _Constant_iterators = _Traits::__constant_iterators::value, bool _Unique_keys = _Traits::__unique_keys::value>
struct std::__detail::Insert<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, _Constant_iterators, _Unique_keys>

```

Primary class template `_Insert`.

Select insert member functions appropriate to `_Hashtable` policy choices.

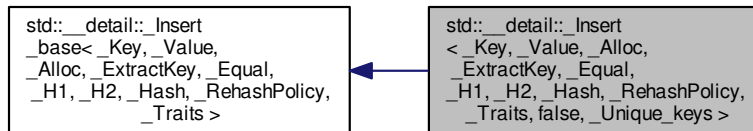
Definition at line 790 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable_policy.h](#)

4.441 `std::__detail::Insert<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false, _Unique_keys >` Struct Template Reference

Inheritance diagram for `std::__detail::Insert<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false, _Unique_keys >`:



Public Types

- using **__base_type** = `_Insert_base<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >`
- using **__hashtable** = `typename __base_type::__hashtable`
- using **__ireturn_type** = `typename __base_type::__ireturn_type`
- `template<typename _Pair >`
using **__is_cons** = `std::is_constructible<value_type, _Pair && >`
- using **__unique_keys** = `typename __base_type::__unique_keys`
- `template<typename _Pair >`
using **__IFcons** = `std::enable_if<__is_cons<_Pair >::value >`
- `template<typename _Pair >`
using **__IFcons_p** = `typename __IFcons<_Pair >::type`
- using **const_iterator** = `typename __base_type::const_iterator`
- using **iterator** = `typename __base_type::iterator`
- using **value_type** = `typename __base_type::value_type`

Public Member Functions

- `__ireturn_type insert (const value_type &__v)`
- `iterator insert (const_iterator __hint, const value_type &__v)`
- `void insert (initializer_list<value_type > __l)`
- `template<typename _InputIterator >`
`void insert (_InputIterator __first, _InputIterator __last)`
- `template<typename _Pair, typename = __IFcons_p<_Pair >>`
`__ireturn_type insert (_Pair &&__v)`
- `template<typename _Pair, typename = __IFcons_p<_Pair >>`
`iterator insert (const_iterator __hint, _Pair &&__v)`

Protected Types

- using `__hashtable_base` = `__Hashtable_base`<_Key, _Value, _ExtractKey, _Equal, _H1, _H2, _Hash, _Traits >
- using `__node_alloc_type` = `typename __alloctr_rebind<_Alloc, __node_type >::__type`
- using `__node_gen_type` = `_AllocNode<__node_alloc_type >`
- using `__node_type` = `__Hash_node`<_Value, _Traits::__hash_cached::value >
- using `size_type` = `typename __hashtable_base::size_type`

Protected Member Functions

- `__hashtable` & `_M_conjure_hashtable` ()
- `template<typename _InputIterator, typename _NodeGetter >`
`void _M_insert_range` (_InputIterator __first, _InputIterator __last, const _NodeGetter &)

4.441.1 Detailed Description

```
template<typename _Key, typename _Value, typename _Alloc, typename _ExtractKey, typename _Equal, typename _H1, typename _H2,
typename _Hash, typename _RehashPolicy, typename _Traits, bool _Unique_keys>
struct std::__detail::Insert<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false, _Unique_keys
>
```

Specialization.

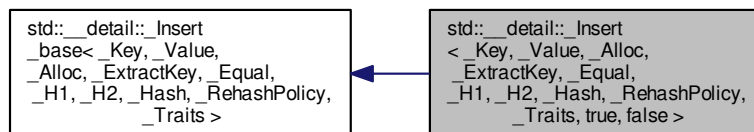
Definition at line 879 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable_policy.h](#)

4.442 std::__detail::Insert<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true, false > Struct Template Reference

Inheritance diagram for `std::__detail::Insert<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true, false >`:



Public Types

- using **__base_type** = [_Insert_base](#)< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >
- using **__hashtable** = typename [__base_type::__hashtable](#)
- using **__node_gen_type** = typename [__base_type::__node_gen_type](#)
- using **__unique_keys** = typename [__base_type::__unique_keys](#)
- using **const_iterator** = typename [__base_type::const_iterator](#)
- using **iterator** = typename [__base_type::iterator](#)
- using **value_type** = typename [__base_type::value_type](#)

Public Member Functions

- [__ireturn_type insert](#) (const value_type &__v)
- iterator [insert](#) (const_iterator __hint, const value_type &__v)
- void [insert](#) (initializer_list< value_type > __l)
- template<typename _InputIterator >
void [insert](#) (_InputIterator __first, _InputIterator __last)
- iterator [insert](#) (value_type &&__v)
- iterator [insert](#) (const_iterator __hint, value_type &&__v)

Protected Types

- using **__hashtable_base** = [_Hashtable_base](#)< _Key, _Value, _ExtractKey, _Equal, _H1, _H2, _Hash, _Traits >
- using **__ireturn_type** = typename [__hashtable_base::__ireturn_type](#)
- using **__node_alloc_type** = typename [_allocr_rebind](#)< _Alloc, [__node_type](#) >::__type
- using **__node_type** = [_Hash_node](#)< _Value, _Traits::__hash_cached::value >
- using **size_type** = typename [__hashtable_base::size_type](#)

Protected Member Functions

- [__hashtable & _M_conjure_hashtable](#) ()
- template<typename _InputIterator, typename _NodeGetter >
void [_M_insert_range](#) (_InputIterator __first, _InputIterator __last, const _NodeGetter &)

4.442.1 Detailed Description

```
template<typename _Key, typename _Value, typename _Alloc, typename _ExtractKey, typename _Equal, typename _H1, typename _H2,
typename _Hash, typename _RehashPolicy, typename _Traits>
struct std::__detail::__Insert< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true, false >
```

Specialization.

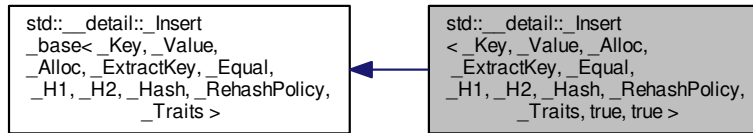
Definition at line 838 of file hashtable_policy.h.

The documentation for this struct was generated from the following file:

- [hashtable_policy.h](#)

4.443 `std::__detail::Insert<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true, true >` Struct Template Reference

Inheritance diagram for `std::__detail::Insert<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true, true >`:



Public Types

- using `__base_type` = `Insert_base<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >`
- using `__hashtable` = `typename __base_type::__hashtable`
- using `__node_gen_type` = `typename __base_type::__node_gen_type`
- using `__unique_keys` = `typename __base_type::__unique_keys`
- using `const_iterator` = `typename __base_type::const_iterator`
- using `iterator` = `typename __base_type::iterator`
- using `value_type` = `typename __base_type::value_type`

Public Member Functions

- `__ireturn_type insert` (`const value_type &__v`)
- `iterator insert` (`const_iterator __hint, const value_type &__v`)
- `void insert` (`initializer_list<value_type> __l`)
- `template<typename _InputIterator>`
`void insert` (`_InputIterator __first, _InputIterator __last`)
- `std::pair<iterator, bool> insert` (`value_type &&__v`)
- `iterator insert` (`const_iterator __hint, value_type &&__v`)

Protected Types

- using `__hashtable_base` = `Hashtable_base<_Key, _Value, _ExtractKey, _Equal, _H1, _H2, _Hash, _Traits >`
- using `__ireturn_type` = `typename __hashtable_base::__ireturn_type`
- using `__node_alloc_type` = `typename __alloctr_rebind<_Alloc, __node_type>::__type`
- using `__node_type` = `Hash_node<_Value, _Traits::__hash_cached::value >`
- using `size_type` = `typename __hashtable_base::size_type`

Protected Member Functions

- [__hashtable](#) & [_M_conjure_hashtable](#) ()
- `template<typename _InputIterator, typename _NodeGetter >`
`void _M_insert_range (_InputIterator __first, _InputIterator __last, const _NodeGetter &)`

4.443.1 Detailed Description

```
template<typename _Key, typename _Value, typename _Alloc, typename _ExtractKey, typename _Equal, typename _H1, typename _H2,
typename _Hash, typename _RehashPolicy, typename _Traits>
struct std::__detail::__Insert< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true, true >
```

Specialization.

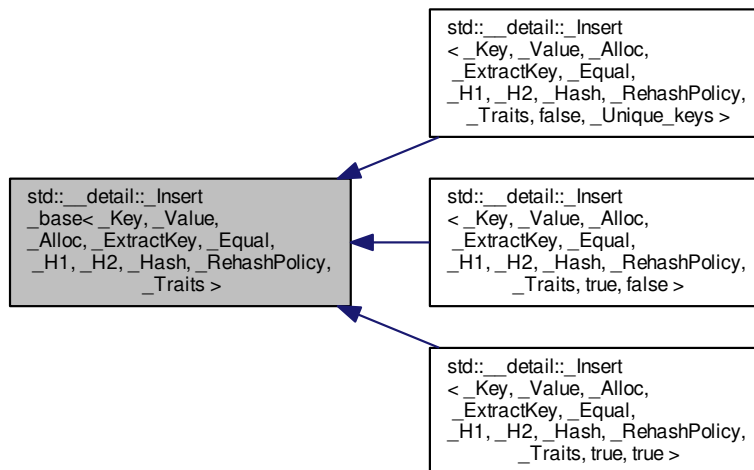
Definition at line 797 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable_policy.h](#)

4.444 `std::__detail::__Insert_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >` Struct Template Reference

Inheritance diagram for `std::__detail::__Insert_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >`:



Public Member Functions

- `__ireturn_type insert` (const value_type &__v)
- iterator `insert` (const_iterator __hint, const value_type &__v)
- void `insert` (initializer_list< value_type > __l)
- template<typename _InputIterator >
void `insert` (_InputIterator __first, _InputIterator __last)

Protected Types

- using `__hashtable` = [_Hashtable](#)< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >
- using `__hashtable_base` = [_Hashtable_base](#)< _Key, _Value, _ExtractKey, _Equal, _H1, _H2, _Hash, _Traits >
- using `__ireturn_type` = typename `__hashtable_base`::__ireturn_type
- using `__node_alloc_type` = typename __alloctr_rebind< _Alloc, [__node_type](#) >::__type
- using `__node_gen_type` = _AllocNode< __node_alloc_type >
- using `__node_type` = [_Hash_node](#)< _Value, _Traits::__hash_cached::value >
- using `__unique_keys` = typename `__hashtable_base`::__unique_keys
- using `const_iterator` = typename [__hashtable_base](#)::const_iterator
- using `iterator` = typename [__hashtable_base](#)::iterator
- using `size_type` = typename `__hashtable_base`::size_type
- using `value_type` = typename `__hashtable_base`::value_type

Protected Member Functions

- [__hashtable](#) & `_M_conjure_hashtable` ()
- template<typename _InputIterator, typename _NodeGetter >
void `_M_insert_range` (_InputIterator __first, _InputIterator __last, const _NodeGetter &)

4.444.1 Detailed Description

```
template<typename _Key, typename _Value, typename _Alloc, typename _ExtractKey, typename _Equal, typename _H1, typename _H2,
typename _Hash, typename _RehashPolicy, typename _Traits>
struct std::__detail::__Insert_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >
```

Primary class template `_Insert_base`.

`insert` member functions appropriate to all `_Hashtables`.

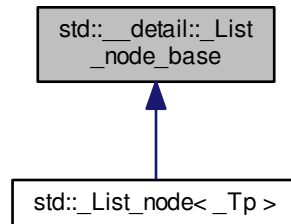
Definition at line 685 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable_policy.h](#)

4.445 `std::__detail::_List_node_base` Struct Reference

Inheritance diagram for `std::__detail::_List_node_base`:



Public Member Functions

- `void _M_hook (_List_node_base *const __position) noexcept`
- `void _M_reverse () noexcept`
- `void _M_transfer (_List_node_base *const __first, _List_node_base *const __last) noexcept`
- `void _M_unhook () noexcept`

Static Public Member Functions

- `static void swap (_List_node_base &__x, _List_node_base &__y) noexcept`

Public Attributes

- `_List_node_base * _M_next`
- `_List_node_base * _M_prev`

4.445.1 Detailed Description

Common part of a node in the list.

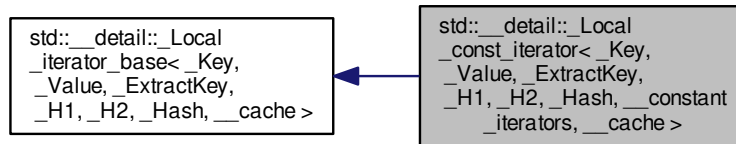
Definition at line 77 of file `stl_list.h`.

The documentation for this struct was generated from the following file:

- [stl_list.h](#)

4.446 `std::__detail::__Local_const_iterator< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __constant_iterators, __cache >` Struct Template Reference

Inheritance diagram for `std::__detail::__Local_const_iterator< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __constant_iterators, __cache >`:



Public Types

- typedef `std::ptrdiff_t` **difference_type**
- typedef `std::forward_iterator_tag` **iterator_category**
- typedef `const _Value *` **pointer**
- typedef `const _Value &` **reference**
- typedef `_Value` **value_type**

Public Member Functions

- **_Local_const_iterator** (`const __hash_code_base &__base, _Hash_node< _Value, __cache > *__p, std::size_t __bkt, std::size_t __bkt_count`)
- **_Local_const_iterator** (`const _Local_iterator< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __constant_iterators, __cache > &__x`)
- reference **operator*** () const
- [_Local_const_iterator](#) & **operator++** ()
- [_Local_const_iterator](#) **operator++** (int)
- pointer **operator->** () const

4.446.1 Detailed Description

```
template<typename _Key, typename _Value, typename _ExtractKey, typename _H1, typename _H2, typename _Hash, bool __constant_iterators, bool __cache>
```

```
struct std::__detail::__Local_const_iterator< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __constant_iterators, __cache >
```

local const_iterators

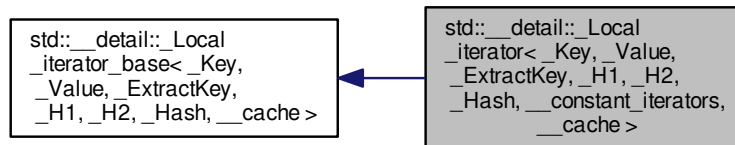
Definition at line 1578 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable_policy.h](#)

4.447 `std::__detail::_Local_iterator< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __constant_iterators, __cache >` Struct Template Reference

Inheritance diagram for `std::__detail::_Local_iterator< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __constant_iterators, __cache >`:



Public Types

- typedef `std::ptrdiff_t` **difference_type**
- typedef [std::forward_iterator_tag](#) **iterator_category**
- typedef `std::conditional< __constant_iterators, const _Value *, _Value * >::type` **pointer**
- typedef `std::conditional< __constant_iterators, const _Value &, _Value & >::type` **reference**
- typedef `_Value` **value_type**

Public Member Functions

- **_Local_iterator** (`const __hash_code_base &__base`, [_Hash_node](#)`< _Value, __cache > *__p`, `std::size_t __bkt`, `std::size_t __bkt_count`)
- reference **operator*** () const
- [_Local_iterator](#) & **operator++** ()
- [_Local_iterator](#) **operator++** (int)
- pointer **operator->** () const

4.447.1 Detailed Description

```

template<typename _Key, typename _Value, typename _ExtractKey, typename _H1, typename _H2, typename _Hash, bool __constant_iterators, bool __cache>
struct std::__detail::_Local_iterator< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __constant_iterators, __cache >

```

local iterators

Definition at line 1523 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable_policy.h](#)

4.448 `std::__detail::_Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache_hash_code`
> Struct Template Reference

4.448.1 Detailed Description

```
template<typename _Key, typename _Value, typename _ExtractKey, typename _H1, typename _H2, typename _Hash, bool __cache_hash_code←
hash_code>
struct std::__detail::_Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache_hash_code >
```

Primary class template `_Local_iterator_base`.

Base class for local iterators, used to iterate within a bucket but not between buckets.

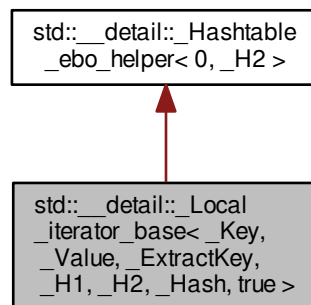
Definition at line 1032 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable_policy.h](#)

4.449 `std::__detail::_Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, true >` **Struct Template Reference**

Inheritance diagram for `std::__detail::_Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, true >`:



Public Member Functions

- `const void * _M_curr () const`
- `std::size_t _M_get_bucket () const`

Protected Types

- using `__base_type` = `_Hashtable_ebo_helper`< 0, `_H2` >
- using `__hash_code_base` = `_Hash_code_base`< `_Key`, `_Value`, `_ExtractKey`, `_H1`, `_H2`, `_Hash`, `true` >

Protected Member Functions

- `_Local_iterator_base` (const `__hash_code_base` & `__base`, `_Hash_node`< `_Value`, `true` > * `__p`, `std::size_t` `__bkt`, `std::size_t` `__bkt_count`)
- void `_M_incr` ()

Protected Attributes

- `std::size_t` `_M_bucket`
- `std::size_t` `_M_bucket_count`
- `_Hash_node`< `_Value`, `true` > * `_M_cur`

4.449.1 Detailed Description

```
template<typename _Key, typename _Value, typename _ExtractKey, typename _H1, typename _H2, typename _Hash>
struct std::__detail::__Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, true >
```

Partial specialization used when nodes contain a cached hash code.

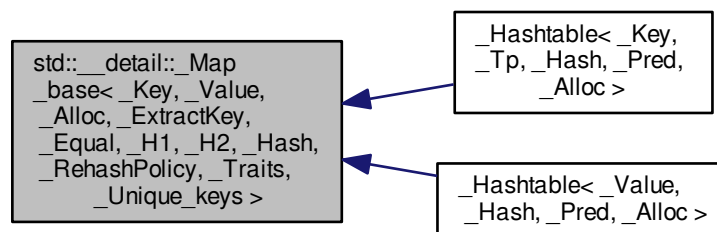
Definition at line 1344 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable_policy.h](#)

4.450 `std::__detail::__Map_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, _Unique_keys >` Struct Template Reference

Inheritance diagram for `std::__detail::__Map_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, _Unique_keys >`:



4.450.1 Detailed Description

```
template<typename _Key, typename _Value, typename _Alloc, typename _ExtractKey, typename _Equal, typename _H1, typename _H2,
typename _Hash, typename _RehashPolicy, typename _Traits, bool _Unique_keys = _Traits::__unique_keys::value>
struct std::__detail::_Map_base<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, _Unique_keys >
```

Primary class template `_Map_base`.

If the hashtable has a value type of the form `pair<T1, T2>` and a key extraction policy (`_ExtractKey`) that returns the first part of the pair, the hashtable gets a `mapped_type` typedef. If it satisfies those criteria and also has unique keys, then it also gets an operator[].

Definition at line 532 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable_policy.h](#)

4.451 `std::__detail::_Map_base<_Key, _Pair, _Alloc, _Select1st, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false >` Struct Template Reference

Public Types

- using **mapped_type** = typename `std::tuple_element< 1, _Pair >::type`

4.451.1 Detailed Description

```
template<typename _Key, typename _Pair, typename _Alloc, typename _Equal, typename _H1, typename _H2, typename _Hash, type-
name _RehashPolicy, typename _Traits>
struct std::__detail::_Map_base<_Key, _Pair, _Alloc, _Select1st, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false >
```

Partial specialization, `__unique_keys` set to false.

Definition at line 538 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable_policy.h](#)

4.452 `std::__detail::_Map_base<_Key, _Pair, _Alloc, _Select1st, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true >` Struct Template Reference

Public Types

- using **iterator** = typename `__hashtable_base::iterator`
- using **key_type** = typename `__hashtable_base::key_type`
- using **mapped_type** = typename `std::tuple_element< 1, _Pair >::type`

Public Member Functions

- mapped_type & **at** (const key_type &__k)
- const mapped_type & **at** (const key_type &__k) const
- mapped_type & **operator[]** (const key_type &__k)
- mapped_type & **operator[]** (key_type &&__k)

4.452.1 Detailed Description

```
template<typename _Key, typename _Pair, typename _Alloc, typename _Equal, typename _H1, typename _H2, typename _Hash, type-
name _RehashPolicy, typename _Traits>
struct std::__detail::_Map_base< _Key, _Pair, _Alloc, _Select1st, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true >
```

Partial specialization, __unique_keys set to true.

Definition at line 548 of file hashtable_policy.h.

The documentation for this struct was generated from the following file:

- [hashtable_policy.h](#)

4.453 std::__detail::_Mod_range_hashing Struct Reference

Public Types

- typedef std::size_t **first_argument_type**
- typedef std::size_t **result_type**
- typedef std::size_t **second_argument_type**

Public Member Functions

- result_type **operator()** (first_argument_type __num, second_argument_type __den) const noexcept

4.453.1 Detailed Description

Default range hashing function: use division to fold a large number into the range [0, N).

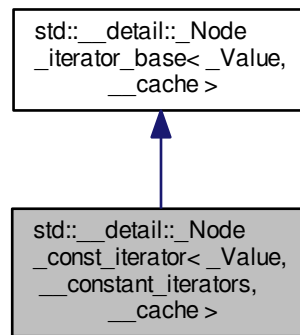
Definition at line 440 of file hashtable_policy.h.

The documentation for this struct was generated from the following file:

- [hashtable_policy.h](#)

4.454 `std::__detail::_Node_const_iterator< _Value, __constant_iterators, __cache >` Struct Template Reference

Inheritance diagram for `std::__detail::_Node_const_iterator< _Value, __constant_iterators, __cache >`:



Public Types

- typedef `std::ptrdiff_t` **difference_type**
- typedef [std::forward_iterator_tag](#) **iterator_category**
- typedef `const _Value *` **pointer**
- typedef `const _Value &` **reference**
- typedef `_Value` **value_type**

Public Member Functions

- **_Node_const_iterator** (`__node_type * __p`) noexcept
- **_Node_const_iterator** (`const _Node_iterator< _Value, __constant_iterators, __cache > & __x`) noexcept
- `void` **_M_incr** () noexcept
- `reference` **operator*** () const noexcept
- **_Node_const_iterator** & **operator++** () noexcept
- **_Node_const_iterator** **operator++** (int) noexcept
- `pointer` **operator->** () const noexcept

Public Attributes

- `__node_type *` **_M_cur**

4.454.1 Detailed Description

```
template<typename _Value, bool __constant_iterators, bool __cache>
struct std::__detail::_Node_const_iterator< _Value, __constant_iterators, __cache >
```

Node const_iterators, used to iterate through all the hashtable.

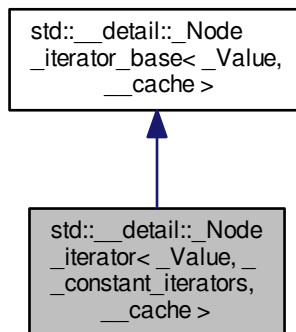
Definition at line 385 of file hashtable_policy.h.

The documentation for this struct was generated from the following file:

- [hashtable_policy.h](#)

4.455 std::__detail::_Node_iterator< _Value, __constant_iterators, __cache > Struct Template Reference

Inheritance diagram for std::__detail::_Node_iterator< _Value, __constant_iterators, __cache >:



Public Types

- typedef std::ptrdiff_t **difference_type**
- typedef [std::forward_iterator_tag](#) **iterator_category**
- using **pointer** = typename std::conditional< __constant_iterators, const _Value *, _Value * >::type
- using **reference** = typename std::conditional< __constant_iterators, const _Value &, _Value & >::type
- typedef _Value **value_type**

Public Member Functions

- **_Node_iterator** (__node_type * __p) noexcept
- void **_M_incr** () noexcept
- reference **operator*** () const noexcept
- [_Node_iterator](#) & **operator++** () noexcept
- [_Node_iterator](#) **operator++** (int) noexcept
- pointer **operator->** () const noexcept

Public Attributes

- `__node_type * _M_cur`

4.455.1 Detailed Description

```
template<typename _Value, bool __constant_iterators, bool __cache>
struct std::__detail::_Node_iterator< _Value, __constant_iterators, __cache >
```

Node iterators, used to iterate through all the hashtable.

Definition at line 334 of file hashtable_policy.h.

The documentation for this struct was generated from the following file:

- [hashtable_policy.h](#)

4.456 std::__detail::_Node_iterator_base< _Value, _Cache_hash_code > Struct Template Reference

Public Types

- using `__node_type = _Hash_node< _Value, _Cache_hash_code >`

Public Member Functions

- `_Node_iterator_base` (`__node_type * __p`) noexcept
- `void _M_incr ()` noexcept

Public Attributes

- `__node_type * _M_cur`

4.456.1 Detailed Description

```
template<typename _Value, bool _Cache_hash_code>
struct std::__detail::_Node_iterator_base< _Value, _Cache_hash_code >
```

Base class for node iterators.

Definition at line 304 of file hashtable_policy.h.

The documentation for this struct was generated from the following file:

- [hashtable_policy.h](#)

4.457 std::__detail::_Prime_rehash_policy Struct Reference

Public Types

- enum { **_S_n_primes** }
- typedef std::size_t **_State**

Public Member Functions

- **_Prime_rehash_policy** (float __z=1.0)
- std::size_t **_M_bkt_for_elements** (std::size_t __n) const
- [std::pair](#)< bool, std::size_t > **_M_need_rehash** (std::size_t __n_bkt, std::size_t __n_elt, std::size_t __n_ins) const
- std::size_t **_M_next_bkt** (std::size_t __n) const
- void **_M_reset** () noexcept
- void **_M_reset** (_State __state)
- _State **_M_state** () const
- float **max_load_factor** () const noexcept

Public Attributes

- float **_M_max_load_factor**
- std::size_t **_M_next_resize**

Static Public Attributes

- static const std::size_t **_S_growth_factor**

4.457.1 Detailed Description

Default value for rehash policy. Bucket size is (usually) the smallest prime that keeps the load factor small enough.

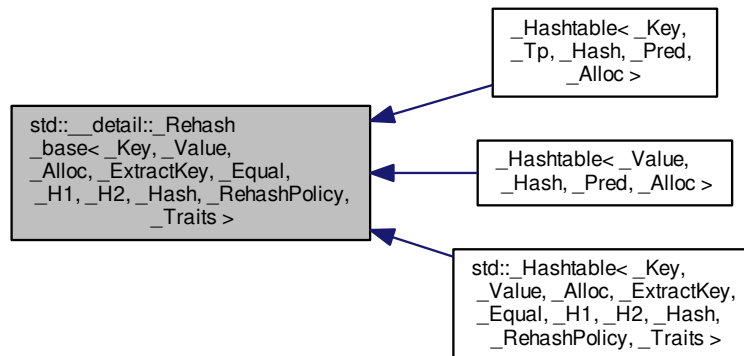
Definition at line 461 of file hashtable_policy.h.

The documentation for this struct was generated from the following file:

- [hashtable_policy.h](#)

4.458 `std::__detail::_Rehash_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >` Struct Template Reference

Inheritance diagram for `std::__detail::_Rehash_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >`:



4.458.1 Detailed Description

```

template<typename _Key, typename _Value, typename _Alloc, typename _ExtractKey, typename _Equal, typename _H1, typename _H2,
typename _Hash, typename _RehashPolicy, typename _Traits>
struct std::__detail::_Rehash_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >

```

Primary class template `_Rehash_base`.

Give hashtable the `max_load_factor` functions and `reserve` iff the rehash policy is `_Prime_rehash_policy`.

Definition at line 934 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable_policy.h](#)

4.459 `std::__detail::_Rehash_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _Prime_rehash_policy, _Traits >` Struct Template Reference

Public Types

- using `__hashtable` = `_Hashtable< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _Prime_rehash_policy, _Traits >`

Public Member Functions

- float **max_load_factor** () const noexcept
- void **max_load_factor** (float __z)
- void **reserve** (std::size_t __n)

4.459.1 Detailed Description

```
template<typename _Key, typename _Value, typename _Alloc, typename _ExtractKey, typename _Equal, typename _H1, typename _H2,
typename _Hash, typename _Traits>
struct std::__detail::__Rehash_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _Prime_rehash_policy, _Traits >
```

Specialization.

Definition at line 940 of file hashtable_policy.h.

The documentation for this struct was generated from the following file:

- [hashtable_policy.h](#)

4.460 std::__detail::__Scanner< _CharT > Class Template Reference

Inherits std::__detail::__ScannerBase.

Public Types

- typedef const [std::ctype](#)< _CharT > **_CtypeT**
- typedef [regex_constants::syntax_option_type](#) **_FlagT**
- typedef const _CharT * **_IterT**
- typedef [std::basic_string](#)< _CharT > **_StringT**
- enum **_TokenT** {
 _S_token_anychar, **_S_token_ord_char**, **_S_token_oct_num**, **_S_token_hex_num**,
 _S_token_backref, **_S_token_subexpr_begin**, **_S_token_subexpr_no_group_begin**, **_S_token_subexpr**↵
 _lookahead_begin,
 _S_token_subexpr_end, **_S_token_bracket_begin**, **_S_token_bracket_neg_begin**, **_S_token_bracket**↵
 end,
 _S_token_interval_begin, **_S_token_interval_end**, **_S_token_quoted_class**, **_S_token_char_class_name**,
 _S_token_collsymbol, **_S_token_equiv_class_name**, **_S_token_opt**, **_S_token_or**,
 _S_token_closure0, **_S_token_closure1**, **_S_token_ungreedy**, **_S_token_line_begin**,
 _S_token_line_end, **_S_token_word_bound**, **_S_token_comma**, **_S_token_dup_count**,
 _S_token_eof, **_S_token_unknown** }

Public Member Functions

- **_Scanner** (_IterT __begin, _IterT __end, [_FlagT](#) __flags, [std::locale](#) __loc)
- void **_M_advance** ()
- [_TokenT](#) **_M_get_token** () const
- const [_StringT](#) & **_M_get_value** () const

Protected Types

- enum **_StateT** { **_S_state_normal**, **_S_state_in_brace**, **_S_state_in_bracket** }

Protected Member Functions

- const char * **_M_find_escape** (char __c)
- bool **_M_is_awk** () const
- bool **_M_is_basic** () const
- bool **_M_is_ecma** () const
- bool **_M_is_extended** () const
- bool **_M_is_grep** () const

Protected Attributes

- bool **_M_at_bracket_start**
- const [std::pair](#)< char, char > **_M_awk_escape_tbl** [11]
- const char * **_M_basic_spec_char**
- const [std::pair](#)< char, char > **_M_ecma_escape_tbl** [8]
- const char * **_M_ecma_spec_char**
- const [std::pair](#)< char, char > * **_M_escape_tbl**
- const char * **_M_extended_spec_char**
- [_FlagT](#) **_M_flags**
- const char * **_M_spec_char**
- **_StateT** **_M_state**
- **_TokenT** **_M_token**
- const [std::pair](#)< char, **_TokenT** > **_M_token_tbl** [9]

4.460.1 Detailed Description

```
template<typename _CharT>
class std::__detail::_Scanner<_CharT>
```

Scans an input range for regex tokens.

The `_Scanner` class interprets the regular expression pattern in the input range passed to its constructor as a sequence of parse tokens passed to the regular expression compiler. The sequence of tokens provided depends on the flag settings passed to the constructor: different regular expression grammars will interpret the same input pattern in syntactically different ways.

Definition at line 201 of file `regex_scanner.h`.

4.460.2 Member Enumeration Documentation

4.460.2.1 enum std::__detail::_ScannerBase::_TokenT [inherited]

Token types returned from the scanner.

Definition at line 46 of file regex_scanner.h.

The documentation for this class was generated from the following file:

- [regex_scanner.h](#)

4.461 std::__detail::_StateSeq<_TraitsT> Class Template Reference

Public Types

- typedef _NFA<_TraitsT> **_RegexT**

Public Member Functions

- **_StateSeq** (_RegexT &__nfa, _StateIdT __s)
- **_StateSeq** (_RegexT &__nfa, _StateIdT __s, _StateIdT __end)
- void **_M_append** (_StateIdT __id)
- void **_M_append** (const [_StateSeq](#) &__s)
- [_StateSeq](#) **_M_clone** ()

Public Attributes

- _StateIdT **_M_end**
- _RegexT & **_M_nfa**
- _StateIdT **_M_start**

4.461.1 Detailed Description

```
template<typename _TraitsT>
class std::__detail::_StateSeq<_TraitsT>
```

Describes a sequence of one or more `_State`, its current start and end(s). This structure contains fragments of an NFA during construction.

Definition at line 265 of file regex_automaton.h.

The documentation for this class was generated from the following file:

- [regex_automaton.h](#)

4.462 std::__exception_ptr::exception_ptr Class Reference

Public Member Functions

- **exception_ptr** (const [exception_ptr](#) &) noexcept
- **exception_ptr** (nullptr_t) noexcept
- **exception_ptr** ([exception_ptr](#) &&__o) noexcept
- const class std::type_info * **__cxa_exception_type** () const noexcept __attribute__((__pure__))
- **operator bool** () const
- [exception_ptr](#) & **operator=** (const [exception_ptr](#) &) noexcept
- [exception_ptr](#) & **operator=** ([exception_ptr](#) &&__o) noexcept
- void **swap** ([exception_ptr](#) &) noexcept

Friends

- bool **operator==** (const [exception_ptr](#) &, const [exception_ptr](#) &) noexcept __attribute__((__pure__))
- [exception_ptr](#) **std::current_exception** () noexcept
- void **std::rethrow_exception** ([exception_ptr](#))

4.462.1 Detailed Description

An opaque pointer to an arbitrary exception.

Definition at line 75 of file [exception_ptr.h](#).

The documentation for this class was generated from the following file:

- [exception_ptr.h](#)

4.463 std::__has_iterator_category_helper<_Tp> Class Template Reference

Public Attributes

- decltype(__test<_Tp>(0)) typedef **type**

4.463.1 Detailed Description

```
template<typename _Tp>
class std::__has_iterator_category_helper<_Tp>
```

Traits class for iterators.

This class does nothing but define nested typedefs. The general version simply *forwards* the nested typedefs from the Iterator argument. Specialized versions for pointers and pointers-to-const provide tighter, more correct semantics.

Definition at line 142 of file [stl_iterator_base_types.h](#).

The documentation for this class was generated from the following file:

- [stl_iterator_base_types.h](#)

4.464 `std::__parallel::_CRandNumber<_MustBeInt >` Struct Template Reference

Public Member Functions

- `int operator() (int __limit)`

4.464.1 Detailed Description

```
template<typename _MustBeInt = int>
struct std::__parallel::_CRandNumber<_MustBeInt >
```

Functor wrapper for `std::rand()`.

Definition at line 1649 of file `algo.h`.

The documentation for this struct was generated from the following file:

- [algo.h](#)

4.465 `std::__profile::map<_Key, _Tp, _Compare, _Allocator >` Class Template Reference

Inherits `map<_Key, _Tp, _Compare, _Allocator >`.

Public Types

- typedef `_Allocator` **allocator_type**
- typedef `_Base::const_iterator` **const_iterator**
- typedef `_Base::const_pointer` **const_pointer**
- typedef `_Base::const_reference` **const_reference**
- typedef [std::reverse_iterator](#)< `const_iterator` > **const_reverse_iterator**
- typedef `_Base::difference_type` **difference_type**
- typedef `_Base::iterator` **iterator**
- typedef `_Compare` **key_compare**
- typedef `_Key` **key_type**
- typedef `_Tp` **mapped_type**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef [std::reverse_iterator](#)< `iterator` > **reverse_iterator**
- typedef `_Base::size_type` **size_type**
- typedef [std::pair](#)< `const _Key`, `_Tp` > **value_type**

Public Member Functions

- **map** (const `_Compare` &__comp, const `_Allocator` &__a= `_Allocator`())
- `template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>>`
map (`_InputIterator` __first, `_InputIterator` __last, const `_Compare` &__comp= `_Compare`(), const `_Allocator` &__a= `_Allocator`())
- **map** (const `map` &__x)
- **map** (const `_Base` &__x)
- **map** (`map` &&__x) noexcept(is_nothrow_copy_constructible< `_Compare` >::value)
- **map** (initializer_list< `value_type` > __l, const `_Compare` &__c= `_Compare`(), const `allocator_type` &__a= `allocator_type`())
- **map** (const `allocator_type` &__a)
- **map** (const `map` &__x, const `allocator_type` &__a)
- **map** (`map` &&__x, const `allocator_type` &__a) noexcept(is_nothrow_copy_constructible< `_Compare` >::value && `_Alloc_traits::S_always_equal`())
- **map** (initializer_list< `value_type` > __l, const `allocator_type` &__a)
- `template<typename _InputIterator>`
map (`_InputIterator` __first, `_InputIterator` __last, const `allocator_type` &__a)
- `_Base` & **_M_base** () noexcept
- const `_Base` & **_M_base** () const noexcept
- mapped_type & **at** (const key_type &__k)
- const mapped_type & **at** (const key_type &__k) const
- iterator **begin** () noexcept
- const_iterator **begin** () const noexcept
- const_iterator **cbegin** () const noexcept
- const_iterator **cend** () const noexcept
- void **clear** () noexcept
- size_type **count** (const key_type &__x) const
- const_reverse_iterator **crbegin** () const noexcept
- const_reverse_iterator **crend** () const noexcept
- `template<typename... _Args>`
`std::pair`< iterator, bool > **emplace** (_Args &&...__args)
- `template<typename... _Args>`
iterator **emplace_hint** (const_iterator __pos, _Args &&...__args)
- iterator **end** () noexcept
- const_iterator **end** () const noexcept
- `std::pair`< iterator, iterator > **equal_range** (const key_type &__x)
- `std::pair`< const_iterator, const_iterator > **equal_range** (const key_type &__x) const
- iterator **erase** (const_iterator __position)
- iterator **erase** (iterator __position)
- size_type **erase** (const key_type &__x)
- iterator **erase** (const_iterator __first, const_iterator __last)
- iterator **find** (const key_type &__x)
- const_iterator **find** (const key_type &__x) const
- `std::pair`< iterator, bool > **insert** (const `value_type` &__x)
- `template<typename _Pair, typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type>`
`std::pair`< iterator, bool > **insert** (_Pair &&__x)
- void **insert** (std::initializer_list< `value_type` > __list)
- iterator **insert** (const_iterator __position, const `value_type` &__x)
- `template<typename _Pair, typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type>`
iterator **insert** (const_iterator __position, _Pair &&__x)

- `template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>>`
`void insert (_InputIterator __first, _InputIterator __last)`
- iterator **lower_bound** (const key_type &__x)
- const_iterator **lower_bound** (const key_type &__x) const
- `map` & **operator=** (const `map` &)=default
- `map` & **operator=** (`map` &&)=default
- `map` & **operator=** (initializer_list< value_type > __l)
- mapped_type & **operator[]** (const key_type &__k)
- mapped_type & **operator[]** (key_type &&__k)
- `reverse_iterator` **rbegin** () noexcept
- `const_reverse_iterator` **rbegin** () const noexcept
- `reverse_iterator` **rend** () noexcept
- `const_reverse_iterator` **rend** () const noexcept
- void **swap** (`map` &__x) noexcept(_Alloc_traits::_S_nothrow_swap())
- iterator **upper_bound** (const key_type &__x)
- const_iterator **upper_bound** (const key_type &__x) const

4.465.1 Detailed Description

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Allocator = std::allocator<std::pair<const _Key, _Tp>>>>
class std::__profile::map< _Key, _Tp, _Compare, _Allocator >
```

Class `std::map` wrapper with performance instrumentation.

Definition at line 41 of file `profile/map.h`.

The documentation for this class was generated from the following file:

- [profile/map.h](#)

4.466 `std::__profile::multimap< _Key, _Tp, _Compare, _Allocator >` Class Template Reference

Inherits `multimap< _Key, _Tp, _Compare, _Allocator >`.

Public Types

- typedef `_Allocator` **allocator_type**
- typedef `_Base::const_iterator` **const_iterator**
- typedef `_Base::const_pointer` **const_pointer**
- typedef `_Base::const_reference` **const_reference**
- typedef `_Base::const_reverse_iterator` **const_reverse_iterator**
- typedef `_Base::difference_type` **difference_type**
- typedef `_Base::iterator` **iterator**
- typedef `_Compare` **key_compare**
- typedef `_Key` **key_type**
- typedef `_Tp` **mapped_type**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `_Base::reverse_iterator` **reverse_iterator**
- typedef `_Base::size_type` **size_type**
- typedef `std::pair< const _Key, _Tp >` **value_type**

Public Member Functions

- **multimap** (const _Compare &__comp, const _Allocator &__a= _Allocator())
- template<typename _InputIterator, typename = std::RequireInputIter<_InputIterator>>>
multimap (_InputIterator __first, _InputIterator __last, const _Compare &__comp= _Compare(), const _Allocator &__a= _Allocator())
- **multimap** (const **multimap** &)=default
- **multimap** (**multimap** &&)=default
- **multimap** (initializer_list< **value_type** > __l, const _Compare &__c= _Compare(), const allocator_type &__a=allocator_type())
- **multimap** (const allocator_type &__a)
- **multimap** (const **multimap** &__x, const allocator_type &__a)
- **multimap** (**multimap** && __x, const allocator_type &__a) noexcept(is_nothrow_copy_constructible< _Compare >::value && Alloc_traits::S_always_equal())
- **multimap** (initializer_list< **value_type** > __l, const allocator_type &__a)
- template<typename _InputIterator >
multimap (_InputIterator __first, _InputIterator __last, const allocator_type &__a)
- **multimap** (const _Base &__x)
- _Base & _M_base () noexcept
- const _Base & _M_base () const noexcept
- iterator **begin** () noexcept
- const_iterator **begin** () const noexcept
- const_iterator **cbegin** () const noexcept
- const_iterator **end** () const noexcept
- void **clear** () noexcept
- const_reverse_iterator **crbegin** () const noexcept
- const_reverse_iterator **crend** () const noexcept
- template<typename... _Args>
iterator **emplace** (_Args &&... __args)
- template<typename... _Args>
iterator **emplace_hint** (const_iterator __pos, _Args &&... __args)
- iterator **end** () noexcept
- const_iterator **end** () const noexcept
- std::pair< iterator, iterator > **equal_range** (const key_type &__x)
- std::pair< const_iterator, const_iterator > **equal_range** (const key_type &__x) const
- iterator **erase** (const_iterator __position)
- iterator **erase** (iterator __position)
- size_type **erase** (const key_type &__x)
- iterator **erase** (const_iterator __first, const_iterator __last)
- iterator **find** (const key_type &__x)
- const_iterator **find** (const key_type &__x) const
- iterator **insert** (const **value_type** &__x)
- template<typename _Pair, typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type>
iterator **insert** (_Pair &&__x)
- void **insert** (std::initializer_list< **value_type** > __list)
- iterator **insert** (const_iterator __position, const **value_type** &__x)
- template<typename _Pair, typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type>
iterator **insert** (const_iterator __position, _Pair &&__x)
- template<typename _InputIterator, typename = std::RequireInputIter<_InputIterator>>>
void **insert** (_InputIterator __first, _InputIterator __last)
- iterator **lower_bound** (const key_type &__x)

- const_iterator **lower_bound** (const key_type &__x) const
- [multimap](#) & **operator=** (const [multimap](#) &)=default
- [multimap](#) & **operator=** ([multimap](#) &&)=default
- [multimap](#) & **operator=** (initializer_list< [value_type](#) > __l)
- reverse_iterator **rbegin** () noexcept
- const_reverse_iterator **rbegin** () const noexcept
- reverse_iterator **rend** () noexcept
- const_reverse_iterator **rend** () const noexcept
- void **swap** ([multimap](#) &__x) noexcept(_Alloc_traits::_S_nothrow_swap())
- iterator **upper_bound** (const key_type &__x)
- const_iterator **upper_bound** (const key_type &__x) const

4.466.1 Detailed Description

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Allocator = std::allocator<std::pair<const _Key, _Tp> >>
class std::__profile::multimap<_Key, _Tp, _Compare, _Allocator >
```

Class std::multimap wrapper with performance instrumentation.

Definition at line 41 of file profile/multimap.h.

The documentation for this class was generated from the following file:

- [profile/multimap.h](#)

4.467 std::__profile::multiset<_Key, _Compare, _Allocator > Class Template Reference

Inherits multiset<_Key, _Compare, _Allocator >.

Public Types

- typedef _Allocator **allocator_type**
- typedef _Base::const_iterator **const_iterator**
- typedef _Base::const_pointer **const_pointer**
- typedef _Base::const_reference **const_reference**
- typedef _Base::const_reverse_iterator **const_reverse_iterator**
- typedef _Base::difference_type **difference_type**
- typedef _Base::iterator **iterator**
- typedef _Compare **key_compare**
- typedef _Key **key_type**
- typedef _Base::pointer **pointer**
- typedef _Base::reference **reference**
- typedef _Base::reverse_iterator **reverse_iterator**
- typedef _Base::size_type **size_type**
- typedef _Compare **value_compare**
- typedef _Key **value_type**

Public Member Functions

- **multiset** (const _Compare &__comp, const _Allocator &__a=_Allocator())
- template<typename _InputIterator, typename = std::RequireInputIter<_InputIterator>>>
 multiset (_InputIterator __first, _InputIterator __last, const _Compare &__comp=_Compare(), const _Allocator &__a=_Allocator())
- **multiset** (const **multiset** &)=default
- **multiset** (**multiset** &&)=default
- **multiset** (initializer_list< value_type > __l, const _Compare &__comp=_Compare(), const allocator_type &__a=allocator_type())
- **multiset** (const allocator_type &__a)
- **multiset** (const **multiset** &__x, const allocator_type &__a)
- **multiset** (**multiset** && __x, const allocator_type &__a) noexcept(is_nothrow_copy_constructible< _Compare > &::value && Alloc_traits::S_always_equal())
- **multiset** (initializer_list< value_type > __l, const allocator_type &__a)
- template<typename _InputIterator >
 multiset (_InputIterator __first, _InputIterator __last, const allocator_type &__a)
- **multiset** (const **_Base** &__x)
- **_Base** & **_M_base** () noexcept
- const **_Base** & **_M_base** () const noexcept
- iterator **begin** () noexcept
- const_iterator **begin** () const noexcept
- const_iterator **cbegin** () const noexcept
- const_iterator **cend** () const noexcept
- void **clear** () noexcept
- const_reverse_iterator **crbegin** () const noexcept
- const_reverse_iterator **crend** () const noexcept
- template<typename... _Args>
 iterator **emplace** (_Args &&... __args)
- template<typename... _Args>
 iterator **emplace_hint** (const_iterator __pos, _Args &&... __args)
- iterator **end** () noexcept
- const_iterator **end** () const noexcept
- **std::pair**< iterator, iterator > **equal_range** (const key_type &__x)
- **std::pair**< const_iterator, const_iterator > **equal_range** (const key_type &__x) const
- iterator **erase** (const_iterator __position)
- size_type **erase** (const key_type &__x)
- iterator **erase** (const_iterator __first, const_iterator __last)
- iterator **find** (const key_type &__x)
- const_iterator **find** (const key_type &__x) const
- iterator **insert** (const value_type &__x)
- iterator **insert** (value_type && __x)
- iterator **insert** (const_iterator __position, const value_type &__x)
- iterator **insert** (const_iterator __position, value_type && __x)
- template<typename _InputIterator, typename = std::RequireInputIter<_InputIterator>>>
 void **insert** (_InputIterator __first, _InputIterator __last)
- void **insert** (initializer_list< value_type > __l)
- iterator **lower_bound** (const key_type &__x)
- const_iterator **lower_bound** (const key_type &__x) const
- **multiset** & **operator=** (const **multiset** &)=default
- **multiset** & **operator=** (**multiset** &&)=default

- [multiset](#) & **operator=** (initializer_list< value_type > __l)
- reverse_iterator **rbegin** () noexcept
- const_reverse_iterator **rbegin** () const noexcept
- reverse_iterator **rend** () noexcept
- const_reverse_iterator **rend** () const noexcept
- void **swap** ([multiset](#) &__x) noexcept(_Alloc_traits::S_nothrow_swap())
- iterator **upper_bound** (const key_type &__x)
- const_iterator **upper_bound** (const key_type &__x) const

4.467.1 Detailed Description

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Allocator = std::allocator<_Key>>
class std::__profile::multiset< _Key, _Compare, _Allocator >
```

Class std::multiset wrapper with performance instrumentation.

Definition at line 41 of file profile/multiset.h.

The documentation for this class was generated from the following file:

- [profile/multiset.h](#)

4.468 std::__profile::set< _Key, _Compare, _Allocator > Class Template Reference

Inherits set< _Key, _Compare, _Allocator >.

Public Types

- typedef _Allocator **allocator_type**
- typedef _Base::const_iterator **const_iterator**
- typedef _Base::const_pointer **const_pointer**
- typedef _Base::const_reference **const_reference**
- typedef _Base::const_reverse_iterator **const_reverse_iterator**
- typedef _Base::difference_type **difference_type**
- typedef _Base::iterator **iterator**
- typedef _Compare **key_compare**
- typedef _Key **key_type**
- typedef _Base::pointer **pointer**
- typedef _Base::reference **reference**
- typedef _Base::reverse_iterator **reverse_iterator**
- typedef _Base::size_type **size_type**
- typedef _Compare **value_compare**
- typedef _Key **value_type**

Public Member Functions

- **set** (const `_Compare` &__comp, const `_Allocator` &__a= `_Allocator`())
- template<typename `_InputIterator` , typename = `std::RequireInputIter<_InputIterator>>`
set (`_InputIterator` __first, `_InputIterator` __last, const `_Compare` &__comp= `_Compare`(), const `_Allocator` &__a= `_Allocator`())
- **set** (const `set` &)=default
- **set** (`set` &&)=default
- **set** (initializer_list< `value_type` > __l, const `_Compare` &__comp= `_Compare`(), const `allocator_type` &__a= `allocator_type`())
- **set** (const `allocator_type` &__a)
- **set** (const `set` &__x, const `allocator_type` &__a)
- **set** (`set` &&__x, const `allocator_type` &__a) noexcept(is_nothrow_copy_constructible< `_Compare` >::value &&__a) noexcept(Alloc_traits::S_always_equal())
- **set** (initializer_list< `value_type` > __l, const `allocator_type` &__a)
- template<typename `_InputIterator` >
set (`_InputIterator` __first, `_InputIterator` __last, const `allocator_type` &__a)
- **set** (const `_Base` &__x)
- `_Base` & `_M_base` () noexcept
- const `_Base` & `_M_base` () const noexcept
- iterator **begin** () noexcept
- const_iterator **begin** () const noexcept
- const_iterator **cbegin** () const noexcept
- const_iterator **end** () const noexcept
- void **clear** () noexcept
- const_reverse_iterator **crbegin** () const noexcept
- const_reverse_iterator **crend** () const noexcept
- template<typename... `_Args`>
`std::pair`< iterator, bool > **emplace** (`_Args` &&... __args)
- template<typename... `_Args`>
iterator **emplace_hint** (const_iterator __pos, `_Args` &&... __args)
- iterator **end** () noexcept
- const_iterator **end** () const noexcept
- `std::pair`< iterator, iterator > **equal_range** (const `key_type` &__x)
- `std::pair`< const_iterator, const_iterator > **equal_range** (const `key_type` &__x) const
- iterator **erase** (const_iterator __position)
- size_type **erase** (const `key_type` &__x)
- iterator **erase** (const_iterator __first, const_iterator __last)
- iterator **find** (const `key_type` &__x)
- const_iterator **find** (const `key_type` &__x) const
- `std::pair`< iterator, bool > **insert** (const `value_type` &__x)
- `std::pair`< iterator, bool > **insert** (`value_type` &&__x)
- iterator **insert** (const_iterator __position, const `value_type` &__x)
- iterator **insert** (const_iterator __position, `value_type` &&__x)
- template<typename `_InputIterator` , typename = `std::RequireInputIter<_InputIterator>>`
void **insert** (`_InputIterator` __first, `_InputIterator` __last)
- void **insert** (initializer_list< `value_type` > __l)
- iterator **lower_bound** (const `key_type` &__x)
- const_iterator **lower_bound** (const `key_type` &__x) const
- `set` & **operator=** (const `set` &)=default
- `set` & **operator=** (`set` &&)=default

- [set](#) & **operator=** (initializer_list< value_type > __l)
- reverse_iterator **rbegin** () noexcept
- const_reverse_iterator **rbegin** () const noexcept
- reverse_iterator **rend** () noexcept
- const_reverse_iterator **rend** () const noexcept
- void **swap** ([set](#) &__x) noexcept(_Alloc_traits::_S_nothrow_swap())
- iterator **upper_bound** (const key_type &__x)
- const_iterator **upper_bound** (const key_type &__x) const

4.468.1 Detailed Description

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Allocator = std::allocator<_Key>>
class std::__profile::set< _Key, _Compare, _Allocator >
```

Class std::set wrapper with performance instrumentation.

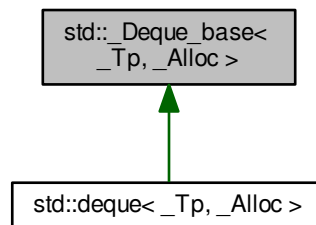
Definition at line 41 of file profile/set.h.

The documentation for this class was generated from the following file:

- [profile/set.h](#)

4.469 std::_Deque_base< _Tp, _Alloc > Class Template Reference

Inheritance diagram for std::_Deque_base< _Tp, _Alloc >:



Public Types

- typedef _Alloc **allocator_type**
- typedef [_Deque_iterator](#)< _Tp, const _Tp &, const _Tp * > **const_iterator**
- typedef [_Deque_iterator](#)< _Tp, _Tp &, _Tp * > **iterator**

Public Member Functions

- **_Deque_base** (size_t __num_elements)
- **_Deque_base** (const allocator_type &__a, size_t __num_elements)
- **_Deque_base** (const allocator_type &__a)
- **_Deque_base** ([_Deque_base](#) &&__x)
- allocator_type **get_allocator** () const noexcept

Protected Types

- enum { **_S_initial_map_size** }
- typedef _Alloc::template rebind<_Tp*>::other **_Map_alloc_type**
- typedef _Alloc::template rebind<_Tp*>::other **_Tp_alloc_type**

Protected Member Functions

- _Tp** **_M_allocate_map** (size_t __n)
- _Tp* **_M_allocate_node** ()
- void **_M_create_nodes** (_Tp** __nstart, _Tp** __nfinish)
- void **_M_deallocate_map** (_Tp** __p, size_t __n) noexcept
- void **_M_deallocate_node** (_Tp* __p) noexcept
- void **_M_destroy_nodes** (_Tp** __nstart, _Tp** __nfinish) noexcept
- _Map_alloc_type **_M_get_map_allocator** () const noexcept
- _Tp_alloc_type & **_M_get_Tp_allocator** () noexcept
- const _Tp_alloc_type & **_M_get_Tp_allocator** () const noexcept
- void **_M_initialize_map** (size_t)

Protected Attributes

- **_Deque_impl** **_M_impl**

4.469.1 Detailed Description

```
template<typename _Tp, typename _Alloc>
class std::_Deque_base<_Tp, _Alloc>
```

Deque base class. This class provides the unified face for deque's allocation. This class's constructor and destructor allocate and deallocate (but do not initialize) storage. This makes exception safety easier.

Nothing in this class ever constructs or destroys an actual Tp element. (Deque handles that itself.) Only/All memory management is performed here.

Definition at line 444 of file stl_deque.h.

4.469.2 Member Function Documentation

4.469.2.1 `template<typename _Tp, typename _Alloc> void std::_Deque_base<_Tp, _Alloc>::_M_initialize_map (size_t __num_elements)` [protected]

Layout storage.

Parameters

<code>__num_elements</code>	The count of T's for which to allocate space at first.
-----------------------------	--

Returns

Nothing.

The initial underlying memory layout is a bit complicated...

Definition at line 587 of file `stl_deque.h`.

References `std::max()`.

The documentation for this class was generated from the following file:

- [stl_deque.h](#)

4.470 `std::_Deque_iterator< _Tp, _Ref, _Ptr >` Struct Template Reference

Public Types

- `typedef _Tp** _Map_pointer`
- `typedef _Deque_iterator _Self`
- `typedef _Deque_iterator< _Tp, const _Tp &, const _Tp * > const_iterator`
- `typedef ptrdiff_t difference_type`
- `typedef _Deque_iterator< _Tp, _Tp &, _Tp * > iterator`
- `typedef std::random_access_iterator_tag iterator_category`
- `typedef _Ptr pointer`
- `typedef _Ref reference`
- `typedef size_t size_type`
- `typedef _Tp value_type`

Public Member Functions

- `_Deque_iterator` (`_Tp *__x`, `_Map_pointer __y`) `noexcept`
- `_Deque_iterator` (`const iterator &__x`) `noexcept`
- `iterator _M_const_cast` () `const noexcept`
- `void _M_set_node` (`_Map_pointer __new_node`) `noexcept`
- `reference operator*` () `const noexcept`
- `_Self operator+` (`difference_type __n`) `const noexcept`
- `_Self & operator++` () `noexcept`
- `_Self operator++` (`int`) `noexcept`
- `_Self & operator+=` (`difference_type __n`) `noexcept`
- `_Self operator-` (`difference_type __n`) `const noexcept`
- `_Self & operator--` () `noexcept`
- `_Self operator--` (`int`) `noexcept`
- `_Self & operator-=` (`difference_type __n`) `noexcept`
- `pointer operator->` () `const noexcept`
- `reference operator[]` (`difference_type __n`) `const noexcept`

Static Public Member Functions

- static size_t **S_buffer_size** () noexcept

Public Attributes

- _Tp * **_M_cur**
- _Tp * **_M_first**
- _Tp * **_M_last**
- _Map_pointer **_M_node**

4.470.1 Detailed Description

```
template<typename _Tp, typename _Ref, typename _Ptr>
struct std::_Deque_iterator< _Tp, _Ref, _Ptr >
```

A deque::iterator.

Quite a bit of intelligence here. Much of the functionality of deque is actually passed off to this class. A deque holds two of these internally, marking its valid range. Access to elements is done as offsets of either of those two, relying on operator overloading in this class.

All the functions are op overloads except for _M_set_node.

Definition at line 106 of file stl_deque.h.

4.470.2 Member Function Documentation

4.470.2.1 `template<typename _Tp, typename _Ref, typename _Ptr> void std::_Deque_iterator< _Tp, _Ref, _Ptr >::_M_set_node (_Map_pointer __new_node) [inline], [noexcept]`

Prepares to traverse new_node. Sets everything except _M_cur, which should therefore be set by the caller immediately afterwards, based on _M_first and _M_last.

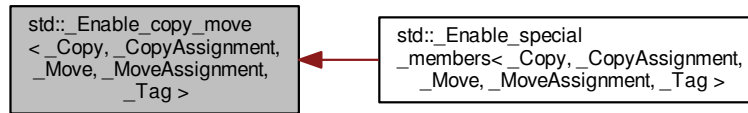
Definition at line 238 of file stl_deque.h.

The documentation for this struct was generated from the following file:

- [stl_deque.h](#)

4.471 `std::_Enable_copy_move< _Copy, _CopyAssignment, _Move, _MoveAssignment, _Tag >` Struct Template Reference

Inheritance diagram for `std::_Enable_copy_move< _Copy, _CopyAssignment, _Move, _MoveAssignment, _Tag >`:



4.471.1 Detailed Description

```
template<bool _Copy, bool _CopyAssignment, bool _Move, bool _MoveAssignment, typename _Tag = void>
struct std::_Enable_copy_move< _Copy, _CopyAssignment, _Move, _MoveAssignment, _Tag >
```

A mixin helper to conditionally enable or disable the copy/move special members.

See also

`_Enable_special_members`

Definition at line 64 of file `enable_special_members.h`.

The documentation for this struct was generated from the following file:

- [enable_special_members.h](#)

4.472 `std::_Enable_default_constructor< _Switch, _Tag >` Struct Template Reference

4.472.1 Detailed Description

```
template<bool _Switch, typename _Tag = void>
struct std::_Enable_default_constructor< _Switch, _Tag >
```

A mixin helper to conditionally enable or disable the default constructor.

See also

`_Enable_special_members`

Definition at line 45 of file `enable_special_members.h`.

The documentation for this struct was generated from the following file:

- [enable_special_members.h](#)

4.473 std::_Enable_destructor< _Switch, _Tag > Struct Template Reference

4.473.1 Detailed Description

```
template<bool _Switch, typename _Tag = void>
struct std::_Enable_destructor< _Switch, _Tag >
```

A mixin helper to conditionally enable or disable the default destructor.

See also

[_Enable_special_members](#)

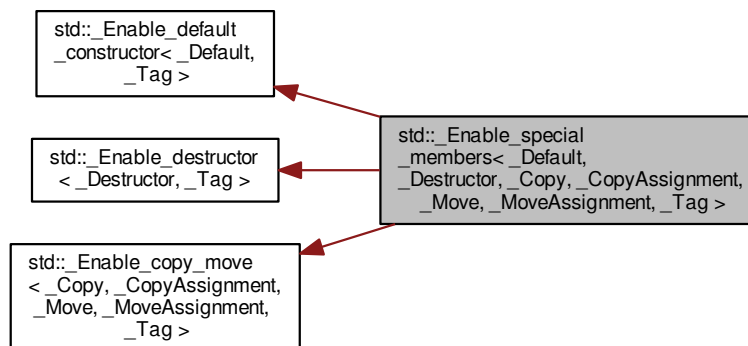
Definition at line 54 of file `enable_special_members.h`.

The documentation for this struct was generated from the following file:

- [enable_special_members.h](#)

4.474 std::_Enable_special_members< _Default, _Destructor, _Copy, _CopyAssignment, _Move, _MoveAssignment, _Tag > Struct Template Reference

Inheritance diagram for `std::_Enable_special_members< _Default, _Destructor, _Copy, _CopyAssignment, _Move, _MoveAssignment, _Tag >`:



4.474.1 Detailed Description

```
template<bool _Default, bool _Destructor, bool _Copy, bool _CopyAssignment, bool _Move, bool _MoveAssignment, typename _Tag =
void>
struct std::_Enable_special_members< _Default, _Destructor, _Copy, _CopyAssignment, _Move, _MoveAssignment, _Tag >
```

A mixin helper to conditionally enable or disable the special members.

The `_Tag` type parameter is to make mixin bases unique and thus avoid ambiguities.

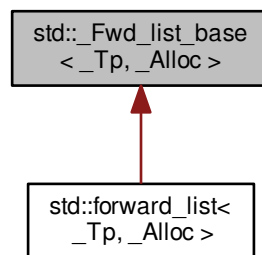
Definition at line 77 of file `enable_special_members.h`.

The documentation for this struct was generated from the following file:

- [enable_special_members.h](#)

4.475 `std::_Fwd_list_base< _Tp, _Alloc >` Struct Template Reference

Inheritance diagram for `std::_Fwd_list_base< _Tp, _Alloc >`:



Public Types

- typedef `_Fwd_list_node< _Tp > _Node`
- typedef `_Fwd_list_const_iterator< _Tp > const_iterator`
- typedef `_Fwd_list_iterator< _Tp > iterator`

Public Member Functions

- `_Fwd_list_base` (`const _Node_alloc_type &__a`)
- `_Fwd_list_base` (`_Fwd_list_base && __lst, const _Node_alloc_type &__a`)
- `_Fwd_list_base` (`_Fwd_list_base && __lst`)
- `_Node_alloc_type & _M_get_Node_allocator` () noexcept
- `const _Node_alloc_type & _M_get_Node_allocator` () const noexcept

Protected Types

- typedef [__gnu_cxx::__alloc_traits](#)< _Alloc > **_Alloc_traits**
- typedef [__gnu_cxx::__alloc_traits](#)< _Node_alloc_type > **_Node_alloc_traits**
- typedef _Alloc_traits::template rebind< [_Fwd_list_node](#)< _Tp > >::other **_Node_alloc_type**
- typedef _Alloc_traits::template rebind< _Tp >::other **_Tp_alloc_type**

Protected Member Functions

- template<typename... _Args>
[_Node](#) * **_M_create_node** (_Args &&... __args)
- [_Fwd_list_node_base](#) * **_M_erase_after** ([_Fwd_list_node_base](#) * __pos)
- [_Fwd_list_node_base](#) * **_M_erase_after** ([_Fwd_list_node_base](#) * __pos, [_Fwd_list_node_base](#) * __last)
- [_Node](#) * **_M_get_node** ()
- template<typename... _Args>
[_Fwd_list_node_base](#) * **_M_insert_after** (const_iterator __pos, _Args &&... __args)
- void **_M_put_node** ([_Node](#) * __p)

Protected Attributes

- [_Fwd_list_impl](#) **_M_impl**

4.475.1 Detailed Description

```
template<typename _Tp, typename _Alloc>
struct std::_Fwd_list_base< _Tp, _Alloc >
```

Base class for forward_list.

Definition at line 274 of file forward_list.h.

The documentation for this struct was generated from the following file:

- [forward_list.h](#)

4.476 std::_Fwd_list_const_iterator< _Tp > Struct Template Reference

Public Types

- typedef const [_Fwd_list_node](#)< _Tp > **_Node**
- typedef [_Fwd_list_const_iterator](#)< _Tp > **_Self**
- typedef ptrdiff_t **difference_type**
- typedef [_Fwd_list_iterator](#)< _Tp > **iterator**
- typedef [std::forward_iterator_tag](#) **iterator_category**
- typedef const _Tp * **pointer**
- typedef const _Tp & **reference**
- typedef _Tp **value_type**

Public Member Functions

- [_Fwd_list_const_iterator](#) (const [_Fwd_list_node_base](#) * __n) noexcept
- [_Fwd_list_const_iterator](#) (const [iterator](#) & __iter) noexcept
- [_Self](#) [_M_next](#) () const noexcept
- bool **operator!=** (const [_Self](#) & __x) const noexcept
- reference **operator*** () const noexcept
- [_Self](#) & **operator++** () noexcept
- [_Self](#) **operator++** (int) noexcept
- pointer **operator->** () const noexcept
- bool **operator==** (const [_Self](#) & __x) const noexcept

Public Attributes

- const [_Fwd_list_node_base](#) * [_M_node](#)

4.476.1 Detailed Description

```
template<typename _Tp>
struct std::_Fwd_list_const_iterator< _Tp >
```

A forward_list::const_iterator.

All the functions are op overloads.

Definition at line 187 of file forward_list.h.

The documentation for this struct was generated from the following file:

- [forward_list.h](#)

4.477 std::_Fwd_list_iterator< _Tp > Struct Template Reference

Public Types

- typedef [_Fwd_list_node](#)< _Tp > **_Node**
- typedef [_Fwd_list_iterator](#)< _Tp > **_Self**
- typedef ptrdiff_t **difference_type**
- typedef [std::forward_iterator_tag](#) **iterator_category**
- typedef _Tp * **pointer**
- typedef _Tp & **reference**
- typedef _Tp **value_type**

Public Member Functions

- [_Fwd_list_iterator](#) ([_Fwd_list_node_base](#) *__n) noexcept
- [_Self _M_next](#) () const noexcept
- bool **operator!=** (const [_Self](#) &__x) const noexcept
- reference **operator*** () const noexcept
- [_Self](#) & **operator++** () noexcept
- [_Self](#) **operator++** (int) noexcept
- pointer **operator->** () const noexcept
- bool **operator==** (const [_Self](#) &__x) const noexcept

Public Attributes

- [_Fwd_list_node_base](#) * **_M_node**

4.477.1 Detailed Description

```
template<typename _Tp>
struct std::_Fwd_list_iterator<_Tp>
```

A forward_list::iterator.

All the functions are op overloads.

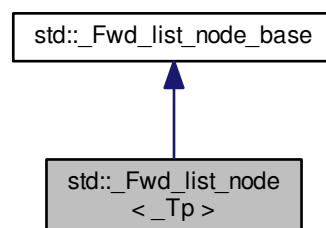
Definition at line 120 of file forward_list.h.

The documentation for this struct was generated from the following file:

- [forward_list.h](#)

4.478 std::_Fwd_list_node<_Tp> Struct Template Reference

Inheritance diagram for std::_Fwd_list_node<_Tp>:



Public Member Functions

- `void _M_reverse_after () noexcept`
- `_Fwd_list_node_base * _M_transfer_after (_Fwd_list_node_base * __begin, _Fwd_list_node_base * __end) noexcept`
- `_Tp * _M_valptr () noexcept`
- `const _Tp * _M_valptr () const noexcept`

Public Attributes

- `_Fwd_list_node_base * _M_next`
- `__gnu_cxx::__aligned_buffer<_Tp> _M_storage`

4.478.1 Detailed Description

```
template<typename _Tp>
struct std::_Fwd_list_node< _Tp >
```

A helper node class for `forward_list`. This is just a linked list with uninitialized storage for a data value in each node. There is a sorting utility method.

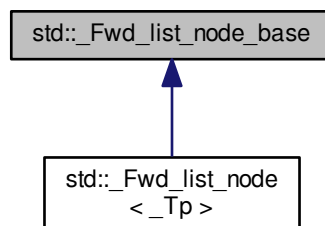
Definition at line 98 of file `forward_list.h`.

The documentation for this struct was generated from the following file:

- [forward_list.h](#)

4.479 `std::_Fwd_list_node_base` Struct Reference

Inheritance diagram for `std::_Fwd_list_node_base`:



Public Member Functions

- `void _M_reverse_after ()` noexcept
- `_Fwd_list_node_base * _M_transfer_after (_Fwd_list_node_base * __begin, _Fwd_list_node_base * __end)` noexcept

Public Attributes

- `_Fwd_list_node_base * _M_next`

4.479.1 Detailed Description

A helper basic node class for `forward_list`. This is just a linked list with nothing inside it. There are purely list shuffling utility methods here.

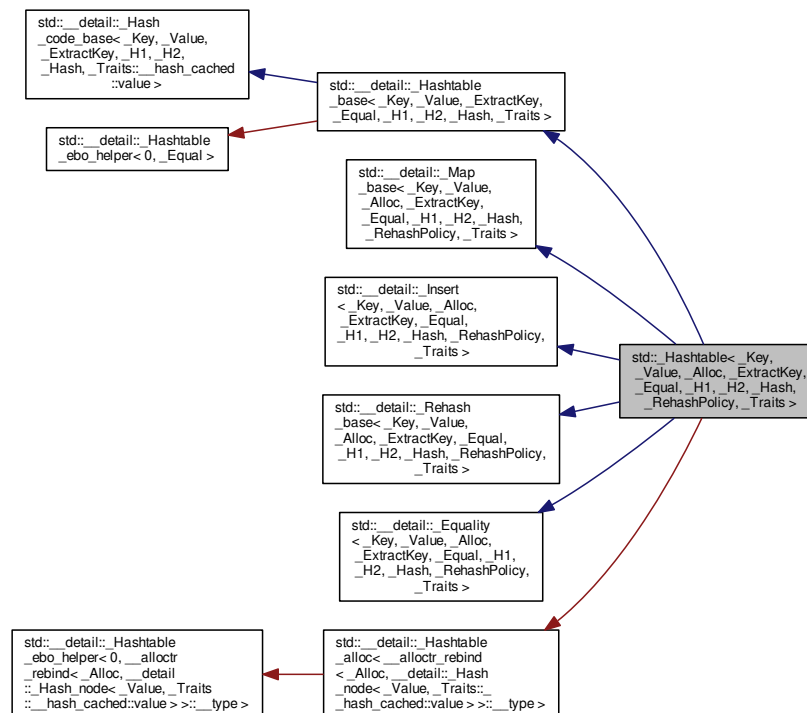
Definition at line 53 of file `forward_list.h`.

The documentation for this struct was generated from the following file:

- [forward_list.h](#)

4.480 `std::_Hashtable<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >` Class Template Reference

Inheritance diagram for `std::_Hashtable<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >`:



Public Types

- typedef `_Alloc` **allocator_type**
- using **const_iterator** = typename `__hashtable_base::const_iterator`
- using **const_local_iterator** = typename `__hashtable_base::const_local_iterator`
- typedef `__value_alloc_traits::const_pointer` **const_pointer**
- typedef `const value_type &` **const_reference**
- using **difference_type** = typename `__hashtable_base::difference_type`
- using **iterator** = typename `__hashtable_base::iterator`
- typedef `__Equal` **key_equal**
- typedef `__Key` **key_type**
- using **local_iterator** = typename `__hashtable_base::local_iterator`
- typedef `__value_alloc_traits::pointer` **pointer**
- typedef `value_type &` **reference**
- using **size_type** = typename `__hashtable_base::size_type`
- typedef `__Value` **value_type**

Public Member Functions

- **_Hashtable** (size_type __bucket_hint, const _H1 &, const _H2 &, const _Hash &, const _Equal &, const _ExtractKey &, const allocator_type &)
- template<typename _InputIterator >
 _Hashtable (_InputIterator __first, _InputIterator __last, size_type __bucket_hint, const _H1 &, const _H2 &, const _Hash &, const _Equal &, const _ExtractKey &, const allocator_type &)
- **_Hashtable** (const [_Hashtable](#) &)
- **_Hashtable** ([_Hashtable](#) &&) noexcept
- **_Hashtable** (const [_Hashtable](#) &, const allocator_type &)
- **_Hashtable** ([_Hashtable](#) &&, const allocator_type &)
- **_Hashtable** (const allocator_type & __a)
- **_Hashtable** (size_type __n=10, const _H1 & __hf=_H1(), const key_equal & __eq=key_equal(), const allocator_type & __a=allocator_type())
- template<typename _InputIterator >
 _Hashtable (_InputIterator __f, _InputIterator __l, size_type __n=0, const _H1 & __hf=_H1(), const key_equal & __eq=key_equal(), const allocator_type & __a=allocator_type())
- **_Hashtable** (initializer_list< value_type > __l, size_type __n=0, const _H1 & __hf=_H1(), const key_equal & __eq=key_equal(), const allocator_type & __a=allocator_type())
- const `_RehashPolicy` & **__rehash_policy** () const
- void **__rehash_policy** (const `_RehashPolicy` &)
- template<typename... _Args>
 std::pair< typename [_Hashtable](#)< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >::iterator, bool > **_M_emplace** (std::true_type, _Args &&... __args)
- template<typename _Arg, typename _NodeGenerator >
 std::pair< typename [_Hashtable](#)< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >::iterator, bool > **_M_insert** (_Arg && __v, const _NodeGenerator & __node_gen, std::true_type)
- iterator **begin** () noexcept
- const_iterator **begin** () const noexcept
- local_iterator **begin** (size_type __n)
- const_local_iterator **begin** (size_type __n) const
- size_type **bucket** (const key_type & __k) const
- size_type **bucket_count** () const noexcept
- size_type **bucket_size** (size_type __n) const

- `const_iterator` **cbegin** () const noexcept
- `const_local_iterator` **cbegin** (size_type __n) const
- `const_iterator` **cend** () const noexcept
- `const_local_iterator` **cend** (size_type __n) const
- `void` **clear** () noexcept
- size_type **count** (const key_type &__k) const
- template<typename... _Args>
 __ireturn_type **emplace** (_Args &&... __args)
- template<typename... _Args>
 iterator **emplace_hint** (const_iterator __hint, _Args &&... __args)
- `bool` **empty** () const noexcept
- `iterator` **end** () noexcept
- `const_iterator` **end** () const noexcept
- `local_iterator` **end** (size_type __n)
- `const_local_iterator` **end** (size_type __n) const
- `std::pair`< iterator, iterator > **equal_range** (const key_type &__k)
- `std::pair`< const_iterator, const_iterator > **equal_range** (const key_type &__k) const
- `iterator` **erase** (const_iterator)
- `iterator` **erase** (iterator __it)
- size_type **erase** (const key_type &__k)
- `iterator` **erase** (const_iterator, const_iterator)
- `iterator` **find** (const key_type &__k)
- `const_iterator` **find** (const key_type &__k) const
- `allocator_type` **get_allocator** () const noexcept
- `key_equal` **key_eq** () const
- `float` **load_factor** () const noexcept
- size_type **max_bucket_count** () const noexcept
- size_type **max_size** () const noexcept
- `_Hashtable` & **operator=** (const `_Hashtable` &__ht)
- `_Hashtable` & **operator=** (`_Hashtable` &&__ht) noexcept(__node_alloc_traits::S_nothrow_move())
- `_Hashtable` & **operator=** (initializer_list< value_type > __l)
- `void` **rehash** (size_type __n)
- size_type **size** () const noexcept
- `void` **swap** (`_Hashtable` &) noexcept(__node_alloc_traits::S_nothrow_swap())

Protected Member Functions

- size_type **_M_bucket_index** (__node_type * __n) const noexcept
- size_type **_M_bucket_index** (const key_type &__k, __hash_code __c) const
- template<typename... _Args>
 std::pair< iterator, bool > **_M_emplace** (std::true_type, _Args &&... __args)
- template<typename... _Args>
 iterator **_M_emplace** (std::false_type __uk, _Args &&... __args)
- template<typename... _Args>
 iterator **_M_emplace** (const_iterator, std::true_type __uk, _Args &&... __args)
- template<typename... _Args>
 iterator **_M_emplace** (const_iterator, std::false_type, _Args &&... __args)
- `const_Equal` & **_M_eq** () const
- `_Equal` & **_M_eq** ()
- `bool` **_M_equals** (const _Key &__k, __hash_code __c, __node_type * __n) const

- `size_type _M_erase (std::true_type, const key_type &)`
- `size_type _M_erase (std::false_type, const key_type &)`
- `iterator _M_erase (size_type __bkt, __node_base * __prev_n, __node_type * __n)`
- `__node_base * _M_find_before_node (size_type, const key_type &, __hash_code) const`
- `__node_type * _M_find_node (size_type __bkt, const key_type & __key, __hash_code __c) const`
- `__node_base * _M_get_previous_node (size_type __bkt, __node_base * __n)`
- `template<typename _Arg, typename _NodeGenerator >`
`std::pair< iterator, bool > _M_insert (_Arg &&, const _NodeGenerator &, std::true_type)`
- `template<typename _Arg, typename _NodeGenerator >`
`iterator _M_insert (_Arg && __arg, const _NodeGenerator & __node_gen, std::false_type __uk)`
- `template<typename _Arg, typename _NodeGenerator >`
`iterator _M_insert (const_iterator, _Arg && __arg, const _NodeGenerator & __node_gen, std::true_type __uk)`
- `template<typename _Arg, typename _NodeGenerator >`
`iterator _M_insert (const_iterator, _Arg &&, const _NodeGenerator &, std::false_type)`
- `void _M_insert_bucket_begin (size_type, __node_type *)`
- `iterator _M_insert_multi_node (__node_type * __hint, __hash_code __code, __node_type * __n)`
- `iterator _M_insert_unique_node (size_type __bkt, __hash_code __code, __node_type * __n)`
- `void _M_remove_bucket_begin (size_type __bkt, __node_type * __next_n, size_type __next_bkt)`
- `void _M_swap (_Hashtable_base & __x)`

Private Types

- `using __bucket_alloc_traits = std::allocator_traits< __bucket_alloc_type >`
- `using __bucket_alloc_type = typename __alloctr_rebind< __node_alloc_type, __bucket_type >::__type`
- `using __value_alloc_type = typename __alloctr_rebind< __node_alloc_type, __value_type >::__type`
- `using __value_type = typename __node_type::value_type`

Private Member Functions

- `__bucket_type * _M_allocate_buckets (std::size_t __n)`
- `__node_type * _M_allocate_node (_Args &&... __args)`
- `void _M_deallocate_buckets (__bucket_type *, std::size_t __n)`
- `void _M_deallocate_node (__node_type * __n)`
- `void _M_deallocate_nodes (__node_type * __n)`
- `__node_alloc_type & _M_node_allocator ()`
- `const __node_alloc_type & _M_node_allocator () const`

Friends

- `template<typename _Keya, typename _Valuea, typename _Alloca, typename _ExtractKeya, typename _Equala, typename _H1a, typename _H2a, typename _Hasha, typename _RehashPolicya, typename _Traitsa, bool _Constant_iteratorsa, bool _Unique_keysa>`
`struct __detail::Insert`
- `template<typename _Keya, typename _Valuea, typename _Alloca, typename _ExtractKeya, typename _Equala, typename _H1a, typename _H2a, typename _Hasha, typename _RehashPolicya, typename _Traitsa >`
`struct __detail::Insert_base`
- `template<typename _Keya, typename _Valuea, typename _Alloca, typename _ExtractKeya, typename _Equala, typename _H1a, typename _H2a, typename _Hasha, typename _RehashPolicya, typename _Traitsa, bool _Unique_keysa>`
`struct __detail::Map_base`

4.480.1 Detailed Description

```
template<typename _Key, typename _Value, typename _Alloc, typename _ExtractKey, typename _Equal, typename _H1, typename _H2,
typename _Hash, typename _RehashPolicy, typename _Traits>
class std::_Hashtable<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >
```

Primary class template `_Hashtable`.

Template Parameters

<code>_Value</code>	CopyConstructible type.
<code>_Key</code>	CopyConstructible type.
<code>_Alloc</code>	An allocator type ([lib.allocator.requirements]) whose <code>_Alloc::value_type</code> is <code>_Value</code> . As a conforming extension, we allow for <code>_Alloc::value_type != _Value</code> .
<code>_ExtractKey</code>	Function object that takes an object of type <code>_Value</code> and returns a value of type <code>_Key</code> .
<code>_Equal</code>	Function object that takes two objects of type <code>k</code> and returns a bool-like value that is true if the two objects are considered equal.
<code>_H1</code>	The hash function. A unary function object with argument type <code>_Key</code> and result type <code>size_t</code> . Return values should be distributed over the entire range <code>[0, numeric_limits<size_t>:max()]</code> .
<code>_H2</code>	The range-hashing function (in the terminology of Tavori and Dreizin). A binary function object whose argument types and result type are all <code>size_t</code> . Given arguments <code>r</code> and <code>N</code> , the return value is in the range <code>[0, N)</code> .
<code>_Hash</code>	The ranged hash function (Tavori and Dreizin). A binary function whose argument types are <code>_Key</code> and <code>size_t</code> and whose result type is <code>size_t</code> . Given arguments <code>k</code> and <code>N</code> , the return value is in the range <code>[0, N)</code> . Default: <code>hash(k, N) = h2(h1(k), N)</code> . If <code>_Hash</code> is anything other than the default, <code>_H1</code> and <code>_H2</code> are ignored.
<code>_RehashPolicy</code>	Policy class with three members, all of which govern the bucket count. <code>_M_next_bkt(n)</code> returns a bucket count no smaller than <code>n</code> . <code>_M_bkt_for_elements(n)</code> returns a bucket count appropriate for an element count of <code>n</code> . <code>_M_need_rehash(n_bkt, n_elt, n_ins)</code> determines whether, if the current bucket count is <code>n_bkt</code> and the current element count is <code>n_elt</code> , we need to increase the bucket count. If so, returns <code>make_pair(true, n)</code> , where <code>n</code> is the new bucket count. If not, returns <code>make_pair(false, <anything>)</code>
<code>_Traits</code>	Compile-time class with three boolean <code>std::integral_constant</code> members: <code>__cache_hash_code</code> , <code>__constant_iterators</code> , <code>__unique_keys</code> .

Each `_Hashtable` data structure has:

- `_Bucket[] _M_buckets`
- `_Hash_node_base _M_before_begin`
- `size_type _M_bucket_count`
- `size_type _M_element_count`

with `_Bucket` being `_Hash_node*` and `_Hash_node` containing:

- `_Hash_node* _M_next`

- `Tp _M_value`
- `size_t _M_hash_code` if `cache_hash_code` is true

In terms of Standard containers the hashtable is like the aggregation of:

- `std::forward_list<_Node>` containing the elements
- `std::vector<std::forward_list<_Node>::iterator>` representing the buckets

The non-empty buckets contain the node before the first node in the bucket. This design makes it possible to implement something like `std::forward_list::insert_after` on container insertion and `std::forward_list::erase_after` on container erase calls. `_M_before_begin` is equivalent to `std::forward_list::before_begin`. Empty buckets contain `nullptr`. Note that one of the non-empty buckets contains `&_M_before_begin` which is not a dereferenceable node so the node pointer in a bucket shall never be dereferenced, only its next node can be.

Walking through a bucket's nodes requires a check on the hash code to see if each node is still in the bucket. Such a design assumes a quite efficient hash functor and is one of the reasons it is highly advisable to set `__cache_hash_code` to true.

The container iterators are simply built from nodes. This way incrementing the iterator is perfectly efficient independent of how many empty buckets there are in the container.

On insert we compute the element's hash code and use it to find the bucket index. If the element must be inserted in an empty bucket we add it at the beginning of the singly linked list and make the bucket point to `_M_before_begin`. The bucket that used to point to `_M_before_begin`, if any, is updated to point to its new before begin node.

On erase, the simple iterator design requires using the hash functor to get the index of the bucket to update. For this reason, when `__cache_hash_code` is set to false the hash functor must not throw and this is enforced by a static assertion.

Functionality is implemented by decomposition into base classes, where the derived `_Hashtable` class is used in `_Map_base`, `_Insert`, `_Rehash_base`, and `_Equality` base classes to access the "this" pointer. `_Hashtable_base` is used in the base classes as a non-recursive, fully-completed-type so that detailed nested type information, such as iterator type and node type, can be used. This is similar to the "Curiously Recurring Template Pattern" (CRTP) technique, but uses a reconstructed, not explicitly passed, template pattern.

Base class templates are:

- `__detail::_Hashtable_base`
- `__detail::_Map_base`
- `__detail::_Insert`
- `__detail::_Rehash_base`
- `__detail::_Equality`

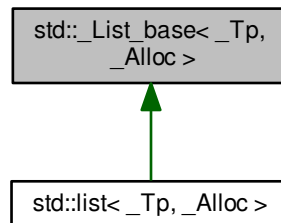
Definition at line 170 of file `bits/hashtable.h`.

The documentation for this class was generated from the following file:

- [bits/hashtable.h](#)

4.481 std::_List_base< _Tp, _Alloc > Class Template Reference

Inheritance diagram for std::_List_base< _Tp, _Alloc >:



Public Types

- typedef `_Alloc` **allocator_type**

Public Member Functions

- **_List_base** (const `_Node_alloc_type` &__a) noexcept
- **_List_base** (`_List_base` &&__x) noexcept
- void **_M_clear** () noexcept
- `_Node_alloc_type` & **_M_get_Node_allocator** () noexcept
- const `_Node_alloc_type` & **_M_get_Node_allocator** () const noexcept
- `_Tp_alloc_type` **_M_get_Tp_allocator** () const noexcept
- void **_M_init** () noexcept
- `allocator_type` **get_allocator** () const noexcept

Protected Types

- typedef `_Alloc::template rebind< _List_node< _Tp > >::other` **_Node_alloc_type**
- typedef `_Alloc::template rebind< _Tp >::other` **_Tp_alloc_type**

Protected Member Functions

- `_List_node`< _Tp > * **_M_get_node** ()
- void **_M_put_node** (`_List_node`< _Tp > *__p) noexcept

Protected Attributes

- `_List_impl` **_M_impl**

4.481.1 Detailed Description

```
template<typename _Tp, typename _Alloc>
class std::_List_base< _Tp, _Alloc >
```

See bits/stl_deque.h's `_Deque_base` for an explanation.

Definition at line 298 of file `stl_list.h`.

The documentation for this class was generated from the following file:

- [stl_list.h](#)

4.482 `std::_List_const_iterator< _Tp >` Struct Template Reference

Public Types

- typedef const `_List_node`< _Tp > `_Node`
- typedef `_List_const_iterator`< _Tp > `_Self`
- typedef ptrdiff_t `difference_type`
- typedef `_List_iterator`< _Tp > `iterator`
- typedef `std::bidirectional_iterator_tag` `iterator_category`
- typedef const _Tp * `pointer`
- typedef const _Tp & `reference`
- typedef _Tp `value_type`

Public Member Functions

- `_List_const_iterator` (const `__detail::_List_node_base` * __x) noexcept
- `_List_const_iterator` (const `iterator` & __x) noexcept
- `iterator_M_const_cast` () const noexcept
- bool `operator!=` (const `_Self` & __x) const noexcept
- reference `operator*` () const noexcept
- `_Self` & `operator++` () noexcept
- `_Self` `operator++` (int) noexcept
- `_Self` & `operator--` () noexcept
- `_Self` `operator--` (int) noexcept
- pointer `operator->` () const noexcept
- bool `operator==` (const `_Self` & __x) const noexcept

Public Attributes

- const `__detail::_List_node_base` * `_M_node`

4.482.1 Detailed Description

```
template<typename _Tp>
struct std::_List_const_iterator< _Tp >
```

A `list::const_iterator`.

All the functions are op overloads.

Definition at line 204 of file `stl_list.h`.

The documentation for this struct was generated from the following file:

- [stl_list.h](#)

4.483 `std::_List_iterator< _Tp >` Struct Template Reference

Public Types

- typedef [_List_node](#)< _Tp > **_Node**
- typedef [_List_iterator](#)< _Tp > **_Self**
- typedef ptrdiff_t **difference_type**
- typedef [std::bidirectional_iterator_tag](#) **iterator_category**
- typedef _Tp * **pointer**
- typedef _Tp & **reference**
- typedef _Tp **value_type**

Public Member Functions

- **_List_iterator** ([__detail::_List_node_base](#) *__x) noexcept
- **_Self_M_const_cast** () const noexcept
- bool **operator!=** (const **_Self** &__x) const noexcept
- reference **operator*** () const noexcept
- **_Self** & **operator++** () noexcept
- **_Self** **operator++** (int) noexcept
- **_Self** & **operator--** () noexcept
- **_Self** **operator--** (int) noexcept
- pointer **operator->** () const noexcept
- bool **operator==** (const **_Self** &__x) const noexcept

Public Attributes

- [__detail::_List_node_base](#) * **_M_node**

4.483.1 Detailed Description

```
template<typename _Tp>
struct std::_List_iterator< _Tp >
```

A list::iterator.

All the functions are op overloads.

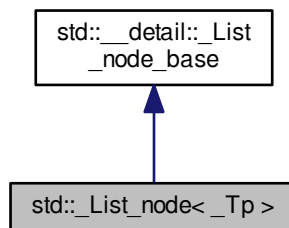
Definition at line 125 of file stl_list.h.

The documentation for this struct was generated from the following file:

- [stl_list.h](#)

4.484 std::_List_node< _Tp > Struct Template Reference

Inheritance diagram for std::_List_node< _Tp >:



Public Member Functions

- template<typename... _Args>
 _List_node (_Args &&... __args)
- void **_M_hook** (_List_node_base *const __position) noexcept
- void **_M_reverse** () noexcept
- void **_M_transfer** (_List_node_base *const __first, _List_node_base *const __last) noexcept
- void **_M_unhook** () noexcept

Static Public Member Functions

- static void **swap** (_List_node_base &__x, _List_node_base &__y) noexcept

Public Attributes

- `_Tp _M_data`
- `_List_node_base * _M_next`
- `_List_node_base * _M_prev`

4.484.1 Detailed Description

```
template<typename _Tp>  
struct std::_List_node<_Tp>
```

An actual node in the list.

Definition at line 106 of file `stl_list.h`.

4.484.2 Member Data Documentation

4.484.2.1 `template<typename _Tp> _Tp std::_List_node<_Tp>::_M_data`

< User's data.

Definition at line 109 of file `stl_list.h`.

Referenced by `std::list<__inp, __rebind_inp>::reverse()`.

The documentation for this struct was generated from the following file:

- [stl_list.h](#)

4.485 `std::_Sp_ebo_helper<_Nm,_Tp,false>` Struct Template Reference

Public Member Functions

- `_Sp_ebo_helper` (`const _Tp &__tp`)

Static Public Member Functions

- `static _Tp & _S_get` (`_Sp_ebo_helper &__eboh`)

4.485.1 Detailed Description

```
template<int _Nm, typename _Tp>
struct std::_Sp_ebo_helper< _Nm, _Tp, false >
```

Specialization not using EBO.

Definition at line 418 of file shared_ptr_base.h.

The documentation for this struct was generated from the following file:

- [shared_ptr_base.h](#)

4.486 std::_Sp_ebo_helper< _Nm, _Tp, true > Struct Template Reference

Inherits `_Tp`.

Public Member Functions

- `_Sp_ebo_helper` (const `_Tp` &__tp)

Static Public Member Functions

- static `_Tp` & `_S_get` (`_Sp_ebo_helper` &__eboh)

4.486.1 Detailed Description

```
template<int _Nm, typename _Tp>
struct std::_Sp_ebo_helper< _Nm, _Tp, true >
```

Specialization using EBO.

Definition at line 408 of file shared_ptr_base.h.

The documentation for this struct was generated from the following file:

- [shared_ptr_base.h](#)

4.487 std::_Temporary_buffer< _ForwardIterator, _Tp > Class Template Reference

Public Types

- typedef pointer **iterator**
- typedef value_type * **pointer**
- typedef ptrdiff_t **size_type**
- typedef `_Tp` **value_type**

Public Member Functions

- [_Temporary_buffer](#) ([_ForwardIterator](#) __first, [_ForwardIterator](#) __last)
- iterator [begin](#) ()
- iterator [end](#) ()
- size_type [requested_size](#) () const
- size_type [size](#) () const

Protected Attributes

- pointer [_M_buffer](#)
- size_type [_M_len](#)
- size_type [_M_original_len](#)

4.487.1 Detailed Description

```
template<typename _ForwardIterator, typename _Tp>
class std::_Temporary_buffer<_ForwardIterator, _Tp>
```

This class is used in two places: `stl_algo.h` and `ext/memory`, where it is wrapped as the `temporary_buffer` class. See `temporary_buffer` docs for more notes.

Definition at line 122 of file `stl_tempbuf.h`.

4.487.2 Constructor & Destructor Documentation

4.487.2.1 `template<typename _ForwardIterator, typename _Tp> std::_Temporary_buffer<_ForwardIterator, _Tp>::_Temporary_buffer (_ForwardIterator __first, _ForwardIterator __last)`

Constructs a temporary buffer of a size somewhere between zero and the size of the given range.

Definition at line 244 of file `stl_tempbuf.h`.

References `std::pair<_T1, _T2>::first`, `std::get_temporary_buffer()`, `std::return_temporary_buffer()`, and `std::pair<_T1, _T2>::second`.

Referenced by `std::_Temporary_buffer<_ForwardIterator, _Tp>::end()`.

4.487.3 Member Function Documentation

4.487.3.1 `template<typename _ForwardIterator, typename _Tp> iterator std::_Temporary_buffer<_ForwardIterator, _Tp>::begin () [inline]`

As per Table mumble.

Definition at line 151 of file `stl_tempbuf.h`.

Referenced by `std::__stable_partition_adaptive()`.

4.487.3.2 `template<typename _ForwardIterator, typename _Tp> iterator std::_Temporary_buffer< _ForwardIterator, _Tp >::end () [inline]`

As per Table mumble.

Definition at line 156 of file `stl_tempbuf.h`.

References `std::_addressof()`, `std::_Construct()`, `std::_Destroy()`, `std::_Temporary_buffer< _ForwardIterator, _Tp >::_Temporary_buffer()`, and `std::return_temporary_buffer()`.

4.487.3.3 `template<typename _ForwardIterator, typename _Tp> size_type std::_Temporary_buffer< _ForwardIterator, _Tp >::requested_size () const [inline]`

Returns the size requested by the constructor; may be `>size()`.

Definition at line 146 of file `stl_tempbuf.h`.

Referenced by `std::__stable_partition_adaptive()`.

4.487.3.4 `template<typename _ForwardIterator, typename _Tp> size_type std::_Temporary_buffer< _ForwardIterator, _Tp >::size () const [inline]`

As per Table mumble.

Definition at line 141 of file `stl_tempbuf.h`.

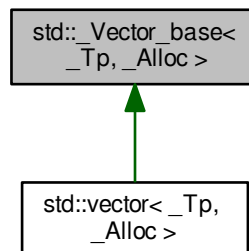
Referenced by `std::__stable_partition_adaptive()`.

The documentation for this class was generated from the following file:

- [stl_tempbuf.h](#)

4.488 `std::_Vector_base< _Tp, _Alloc >` Struct Template Reference

Inheritance diagram for `std::_Vector_base< _Tp, _Alloc >`:



Public Types

- typedef [__gnu_cxx::__alloc_traits](#)< _Alloc >::template rebind< _Tp >::other **_Tp_alloc_type**
- typedef _Alloc **allocator_type**
- typedef [__gnu_cxx::__alloc_traits](#)< _Tp_alloc_type >::pointer **pointer**

Public Member Functions

- **_Vector_base** (const allocator_type &__a) noexcept
- **_Vector_base** (size_t __n)
- **_Vector_base** (size_t __n, const allocator_type &__a)
- **_Vector_base** (_Tp_alloc_type &&__a) noexcept
- **_Vector_base** ([_Vector_base](#) &&__x) noexcept
- **_Vector_base** ([_Vector_base](#) &&__x, const allocator_type &__a)
- pointer **_M_allocate** (size_t __n)
- void **_M_deallocate** (pointer __p, size_t __n)
- _Tp_alloc_type & **_M_get_Tp_allocator** () noexcept
- const _Tp_alloc_type & **_M_get_Tp_allocator** () const noexcept
- allocator_type **get_allocator** () const noexcept

Public Attributes

- [_Vector_impl](#) **_M_impl**

4.488.1 Detailed Description

```
template<typename _Tp, typename _Alloc>
struct std::_Vector_base< _Tp, _Alloc >
```

See bits/stl_deque.h's _Deque_base for an explanation.

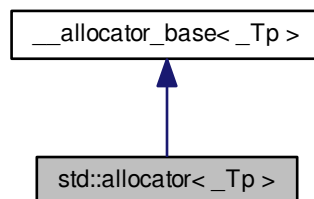
Definition at line 72 of file stl_vector.h.

The documentation for this struct was generated from the following file:

- [stl_vector.h](#)

4.489 std::allocator< _Tp > Class Template Reference

Inheritance diagram for std::allocator< _Tp >:



Public Types

- typedef const _Tp * **const_pointer**
- typedef const _Tp & **const_reference**
- typedef ptrdiff_t **difference_type**
- typedef _Tp * **pointer**
- typedef true_type **propagate_on_container_move_assignment**
- typedef _Tp & **reference**
- typedef size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- **allocator** (const [allocator](#) &__a) throw ()
- template<typename _Tp1 >
 allocator (const [allocator](#)<_Tp1> &) throw ()
- pointer **address** (reference __x) const noexcept
- const_pointer **address** (const_reference __x) const noexcept
- pointer **allocate** (size_type __n, const void * __p)
- template<typename _Up, typename... _Args>
 void **construct** (_Up * __p, _Args &&... __args)
- void **deallocate** (pointer __p, size_type __n)
- template<typename _Up>
 void **destroy** (_Up * __p)
- size_type **max_size** () const noexcept

4.489.1 Detailed Description

```
template<typename _Tp>
class std::allocator<_Tp>
```

The *standard* allocator, as per [20.4].

See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt04ch11.html> for further details.

Template Parameters

<code>_Tp</code>	Type of allocated object.
------------------	---------------------------

Definition at line 92 of file `allocator.h`.

The documentation for this class was generated from the following file:

- [allocator.h](#)

4.490 `std::allocator< void >` Class Template Reference

Public Types

- typedef const void * **const_pointer**
- typedef ptrdiff_t **difference_type**
- typedef void * **pointer**
- typedef true_type **propagate_on_container_move_assignment**
- typedef size_t **size_type**
- typedef void **value_type**

4.490.1 Detailed Description

```
template<>  
class std::allocator< void >
```

`allocator<void>` specialization.

Definition at line 63 of file `allocator.h`.

The documentation for this class was generated from the following file:

- [allocator.h](#)

4.491 `std::allocator_arg_t` Struct Reference

4.491.1 Detailed Description

[`allocator.tag`]

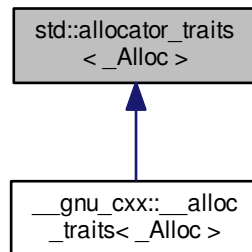
Definition at line 39 of file `uses_allocator.h`.

The documentation for this struct was generated from the following file:

- `uses_allocator.h`

4.492 `std::allocator_traits<_Alloc>` Struct Template Reference

Inheritance diagram for `std::allocator_traits<_Alloc>`:



Public Types

- typedef `_Alloc` [allocator_type](#)
- typedef `__const_pointer` [const_pointer](#)
- typedef `__const_void_pointer` [const_void_pointer](#)
- typedef `__difference_type` [difference_type](#)
- typedef `__pointer` [pointer](#)
- typedef `__propagate_on_container_copy_assignment` [propagate_on_container_copy_assignment](#)
- typedef `__propagate_on_container_move_assignment` [propagate_on_container_move_assignment](#)
- typedef `__propagate_on_container_swap` [propagate_on_container_swap](#)
- template<typename `_Tp`>
using **rebind_alloc** = typename `__alloc_traits::rebind_alloc<_Alloc, _Tp>`
- template<typename `_Tp`>
using **rebind_traits** = `allocator_traits<rebind_alloc<_Tp>>`
- typedef `__size_type` [size_type](#)
- typedef `_Alloc::value_type` [value_type](#)
- typedef `__void_pointer` [void_pointer](#)

Static Public Member Functions

- static [pointer allocate](#) (`_Alloc &__a, size_type __n`)
- static [pointer allocate](#) (`_Alloc &__a, size_type __n, const_void_pointer __hint`)
- template<typename `_Tp`, typename... `_Args`>
static auto [construct](#) (`_Alloc &__a, _Tp *__p, _Args &&...__args`) -> `decltype(_S_construct(__a, __p, std::forward<_Args>(__args)...))`
- static void [deallocate](#) (`_Alloc &__a, pointer __p, size_type __n`)
- template<typename `_Tp`>
static void [destroy](#) (`_Alloc &__a, _Tp *__p`)
- static [size_type max_size](#) (`const _Alloc &__a`) noexcept
- static `_Alloc` [select_on_container_copy_construction](#) (`const _Alloc &__rhs`)

4.492.1 Detailed Description

```
template<typename _Alloc>
struct std::allocator_traits<_Alloc>
```

Uniform interface to all allocator types.

Definition at line 80 of file bits/alloc_traits.h.

4.492.2 Member Typedef Documentation

4.492.2.1 template<typename _Alloc> typedef _Alloc std::allocator_traits<_Alloc>::allocator_type

The allocator type.

Definition at line 83 of file bits/alloc_traits.h.

4.492.2.2 template<typename _Alloc> typedef __const_pointer std::allocator_traits<_Alloc>::const_pointer

The allocator's const pointer type.

`Alloc::const_pointer` if that type exists, otherwise `pointer_traits<pointer>::rebind<const value_type>`

Definition at line 113 of file bits/alloc_traits.h.

4.492.2.3 template<typename _Alloc> typedef __const_void_pointer std::allocator_traits<_Alloc>::const_void_pointer

The allocator's const void pointer type.

`Alloc::const_void_pointer` if that type exists, otherwise `pointer_traits<pointer>::rebind<const void>`

Definition at line 135 of file bits/alloc_traits.h.

4.492.2.4 template<typename _Alloc> typedef __difference_type std::allocator_traits<_Alloc>::difference_type

The allocator's difference type.

`Alloc::difference_type` if that type exists, otherwise `pointer_traits<pointer>::difference_type`

Definition at line 146 of file bits/alloc_traits.h.

4.492.2.5 template<typename _Alloc> typedef __pointer std::allocator_traits<_Alloc>::pointer

The allocator's pointer type.

`Alloc::pointer` if that type exists, otherwise `value_type*`

Definition at line 102 of file bits/alloc_traits.h.

4.492.2.6 `template<typename _Alloc> typedef __propagate_on_container_copy_assignment std::allocator_traits< _Alloc >::propagate_on_container_copy_assignment`

How the allocator is propagated on copy assignment.

`Alloc::propagate_on_container_copy_assignment` if that type exists, otherwise `false_type`

Definition at line 169 of file `bits/alloc_traits.h`.

4.492.2.7 `template<typename _Alloc> typedef __propagate_on_container_move_assignment std::allocator_traits< _Alloc >::propagate_on_container_move_assignment`

How the allocator is propagated on move assignment.

`Alloc::propagate_on_container_move_assignment` if that type exists, otherwise `false_type`

Definition at line 181 of file `bits/alloc_traits.h`.

4.492.2.8 `template<typename _Alloc> typedef __propagate_on_container_swap std::allocator_traits< _Alloc >::propagate_on_container_swap`

How the allocator is propagated on swap.

`Alloc::propagate_on_container_swap` if that type exists, otherwise `false_type`

Definition at line 192 of file `bits/alloc_traits.h`.

4.492.2.9 `template<typename _Alloc> typedef __size_type std::allocator_traits< _Alloc >::size_type`

The allocator's size type.

`Alloc::size_type` if that type exists, otherwise `make_unsigned<difference_type>::type`

Definition at line 157 of file `bits/alloc_traits.h`.

4.492.2.10 `template<typename _Alloc> typedef _Alloc::value_type std::allocator_traits< _Alloc >::value_type`

The allocated type.

Definition at line 85 of file `bits/alloc_traits.h`.

4.492.2.11 `template<typename _Alloc> typedef __void_pointer std::allocator_traits< _Alloc >::void_pointer`

The allocator's void pointer type.

`Alloc::void_pointer` if that type exists, otherwise `pointer_traits<pointer>::rebind<void>`

Definition at line 124 of file `bits/alloc_traits.h`.

4.492.3 Member Function Documentation

4.492.3.1 `template<typename _Alloc> static pointer std::allocator_traits< _Alloc >::allocate (_Alloc & __a, size_type __n) [inline],[static]`

Allocate memory.

Parameters

$_a$	An allocator.
$_n$	The number of objects to allocate space for.

Calls `a.allocate(n)`

Definition at line 356 of file `bits/alloc_traits.h`.

4.492.3.2 `template<typename _Alloc> static pointer std::allocator_traits<_Alloc>::allocate (_Alloc & __a, size_type __n, const_void_pointer __hint) [inline],[static]`

Allocate memory.

Parameters

<code>__a</code>	An allocator.
<code>__n</code>	The number of objects to allocate space for.
<code>__hint</code>	Aid to locality.

Returns

Memory of suitable size and alignment for n objects of type `value_type`

Returns `a.allocate(n, hint)` if that expression is well-formed, otherwise returns `a.allocate(n)`

Definition at line 371 of file `bits/alloc_traits.h`.

4.492.3.3 `template<typename _Alloc> template<typename _Tp, typename... _Args> static auto std::allocator_traits<_Alloc>::construct (_Alloc & __a, _Tp * __p, _Args &&... __args) -> decltype(_S_construct(__a, __p, std::forward<_Args>(__args)...)) [inline],[static]`

Construct an object of type `_Tp`.

Parameters

<code>__a</code>	An allocator.
<code>__p</code>	Pointer to memory of suitable size and alignment for <code>Tp</code>
<code>__args</code>	Constructor arguments.

Calls `__a.construct(__p, std::forward<Args>(__args)...) if that expression is well-formed, otherwise uses placement-new to construct an object of type _Tp at location __p from the arguments __args...`

Definition at line 397 of file `bits/alloc_traits.h`.

4.492.3.4 `template<typename _Alloc> static void std::allocator_traits<_Alloc>::deallocate (_Alloc & __a, pointer __p, size_type __n) [inline],[static]`

Deallocate memory.

Parameters

<code>__a</code>	An allocator.
<code>__p</code>	Pointer to the memory to deallocate.
<code>__n</code>	The number of objects space was allocated for.

Calls `a.deallocate(p, n)`

Definition at line 382 of file `bits/alloc_traits.h`.

4.492.3.5 `template<typename _Alloc> template<typename _Tp> static void std::allocator_traits<_Alloc>::destroy (_Alloc & __a, _Tp* __p) [inline],[static]`

Destroy an object of type `_Tp`.

Parameters

<code>__a</code>	An allocator.
<code>__p</code>	Pointer to the object to destroy

Calls `__a.destroy(__p)` if that expression is well-formed, otherwise calls `__p->~_Tp()`

Definition at line 410 of file `bits/alloc_traits.h`.

4.492.3.6 `template<typename _Alloc> static size_type std::allocator_traits<_Alloc>::max_size (const _Alloc & __a) [inline],[static],[noexcept]`

The maximum supported allocation size.

Parameters

<code>__a</code>	An allocator.
------------------	---------------

Returns

`__a.max_size()` or `numeric_limits<size_type>::max()`

Returns `__a.max_size()` if that expression is well-formed, otherwise returns `numeric_limits<size_↵type>::max()`

Definition at line 421 of file `bits/alloc_traits.h`.

4.492.3.7 `template<typename _Alloc> static _Alloc std::allocator_traits< _Alloc >::select_on_container_copy_construction (const _Alloc & __rhs) [inline], [static]`

Obtain an allocator to use when copying a container.

Parameters

<code>__rhs</code>	An allocator.
--------------------	---------------

Returns

`__rhs.select_on_container_copy_construction()` or `__rhs`

Returns `__rhs.select_on_container_copy_construction()` if that expression is well-formed, otherwise returns `__rhs`

Definition at line 433 of file `bits/alloc_traits.h`.

The documentation for this struct was generated from the following file:

- [bits/alloc_traits.h](#)

4.493 `std::allocator_traits< allocator< _Tp > >` Struct Template Reference

Public Types

- using `allocator_type` = `allocator< _Tp >`
- using `const_pointer` = `const _Tp *`
- using `const_void_pointer` = `const void *`
- using `difference_type` = `std::ptrdiff_t`
- using `pointer` = `_Tp *`
- using `propagate_on_container_copy_assignment` = `false_type`
- using `propagate_on_container_move_assignment` = `true_type`
- using `propagate_on_container_swap` = `false_type`
- `template<typename _Up >`
using `rebind_alloc` = `allocator< _Up >`
- `template<typename _Up >`
using `rebind_traits` = `allocator_traits< allocator< _Up >>`
- using `size_type` = `std::size_t`
- using `value_type` = `_Tp`
- using `void_pointer` = `void *`

Static Public Member Functions

- static [pointer allocate](#) ([allocator_type](#) &__a, [size_type](#) __n)
- static [pointer allocate](#) ([allocator_type](#) &__a, [size_type](#) __n, [const_void_pointer](#) __hint)
- template<typename _Up, typename... _Args>
static void [construct](#) ([allocator_type](#) &__a, _Up *__p, _Args &&...__args)
- static void [deallocate](#) ([allocator_type](#) &__a, [pointer](#) __p, [size_type](#) __n)
- template<typename _Up >
static void [destroy](#) ([allocator_type](#) &__a, _Up *__p)
- static [size_type max_size](#) (const [allocator_type](#) &__a) noexcept
- static [allocator_type select_on_container_copy_construction](#) (const [allocator_type](#) &__rhs)

4.493.1 Detailed Description

```
template<typename _Tp>
struct std::allocator_traits< allocator< _Tp > >
```

Partial specialization for std::allocator.

Definition at line 439 of file bits/alloc_traits.h.

4.493.2 Member Typedef Documentation

4.493.2.1 `template<typename _Tp > using std::allocator_traits< allocator< _Tp > >::allocator_type = allocator<_Tp>`

The allocator type.

Definition at line 442 of file bits/alloc_traits.h.

4.493.2.2 `template<typename _Tp > using std::allocator_traits< allocator< _Tp > >::const_pointer = const _Tp*`

The allocator's const pointer type.

Definition at line 450 of file bits/alloc_traits.h.

4.493.2.3 `template<typename _Tp > using std::allocator_traits< allocator< _Tp > >::const_void_pointer = const void*`

The allocator's const void pointer type.

Definition at line 456 of file bits/alloc_traits.h.

4.493.2.4 `template<typename _Tp > using std::allocator_traits< allocator< _Tp > >::difference_type = std::ptrdiff_t`

The allocator's difference type.

Definition at line 459 of file bits/alloc_traits.h.

4.493.2.5 `template<typename _Tp> using std::allocator_traits< allocator< _Tp > >::pointer = _Tp*`

The allocator's pointer type.

Definition at line 447 of file `bits/alloc_traits.h`.

4.493.2.6 `template<typename _Tp> using std::allocator_traits< allocator< _Tp > >::propagate_on_container_↵
copy_assignment = false_type`

How the allocator is propagated on copy assignment.

Definition at line 465 of file `bits/alloc_traits.h`.

4.493.2.7 `template<typename _Tp> using std::allocator_traits< allocator< _Tp > >::propagate_on_container_↵
move_assignment = true_type`

How the allocator is propagated on move assignment.

Definition at line 468 of file `bits/alloc_traits.h`.

4.493.2.8 `template<typename _Tp> using std::allocator_traits< allocator< _Tp > >::propagate_on_container_swap
= false_type`

How the allocator is propagated on swap.

Definition at line 471 of file `bits/alloc_traits.h`.

4.493.2.9 `template<typename _Tp> using std::allocator_traits< allocator< _Tp > >::size_type = std::size_t`

The allocator's size type.

Definition at line 462 of file `bits/alloc_traits.h`.

4.493.2.10 `template<typename _Tp> using std::allocator_traits< allocator< _Tp > >::value_type = _Tp`

The allocated type.

Definition at line 444 of file `bits/alloc_traits.h`.

4.493.2.11 `template<typename _Tp> using std::allocator_traits< allocator< _Tp > >::void_pointer = void*`

The allocator's void pointer type.

Definition at line 453 of file `bits/alloc_traits.h`.

4.493.3 Member Function Documentation

4.493.3.1 `template<typename _Tp> static pointer std::allocator_traits< allocator< _Tp > >::allocate (allocator_type
&__a, size_type __n) [inline],[static]`

Allocate memory.

Parameters

\leftrightarrow _a	An allocator.
\leftrightarrow _n	The number of objects to allocate space for.

Calls `a.allocate(n)`

Definition at line 487 of file `bits/alloc_traits.h`.

4.493.3.2 `template<typename _Tp> static pointer std::allocator_traits< allocator<_Tp>>::allocate (allocator_type &_a, size_type _n, const_void_pointer __hint) [inline],[static]`

Allocate memory.

Parameters

__a	An allocator.
__n	The number of objects to allocate space for.
__hint	Aid to locality.

Returns

Memory of suitable size and alignment for *n* objects of type `value_type`

Returns `a.allocate(n, hint)`

Definition at line 501 of file `bits/alloc_traits.h`.

4.493.3.3 `template<typename _Tp> template<typename _Up, typename... _Args> static void std::allocator_traits< allocator<_Tp>>::construct (allocator_type &_a, _Up * __p, _Args &&... __args) [inline],[static]`

Construct an object of type `_Up`.

Parameters

__a	An allocator.
__p	Pointer to memory of suitable size and alignment for <code>Tp</code>
__args	Constructor arguments.

Calls `__a.construct(__p, std::forward<Args>(__args)...)`

Definition at line 526 of file `bits/alloc_traits.h`.

4.493.3.4 `template<typename _Tp> static void std::allocator_traits< allocator< _Tp > >::deallocate (allocator_type & __a, pointer __p, size_type __n) [inline],[static]`

Deallocate memory.

Parameters

<code>__a</code>	An allocator.
<code>__p</code>	Pointer to the memory to deallocate.
<code>__n</code>	The number of objects space was allocated for.

Calls `a.deallocate(p, n)`

Definition at line 513 of file `bits/alloc_traits.h`.

4.493.3.5 `template<typename _Tp> template<typename _Up> static void std::allocator_traits< allocator< _Tp > >::destroy (allocator_type & __a, _Up * __p) [inline],[static]`

Destroy an object of type `_Up`.

Parameters

<code>__a</code>	An allocator.
<code>__p</code>	Pointer to the object to destroy

Calls `__a.destroy(__p)`.

Definition at line 538 of file `bits/alloc_traits.h`.

4.493.3.6 `template<typename _Tp> static size_type std::allocator_traits< allocator< _Tp > >::max_size (const allocator_type & __a) [inline],[static],[noexcept]`

The maximum supported allocation size.

Parameters

<code>__a</code>	An allocator.
------------------	---------------

Returns

`__a.max_size()`

Definition at line 547 of file `bits/alloc_traits.h`.

```
4.493.3.7  template<typename _Tp > static allocator_type std::allocator_traits< allocator< _Tp >
           >::select_on_container_copy_construction ( const allocator_type & __rhs )  [inline],[static]
```

Obtain an allocator to use when copying a container.

Parameters

<code>__rhs</code>	An allocator.
--------------------	---------------

Returns

`__rhs`

Definition at line 556 of file `bits/alloc_traits.h`.

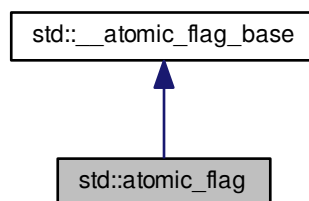
References `std::move()`, and `std::swap()`.

The documentation for this struct was generated from the following file:

- [bits/alloc_traits.h](#)

4.494 std::atomic_flag Struct Reference

Inheritance diagram for `std::atomic_flag`:



Public Member Functions

- **atomic_flag** (const [atomic_flag](#) &)=delete
- constexpr **atomic_flag** (bool __i) noexcept
- [atomic_flag](#) & **operator=** (const [atomic_flag](#) &)=delete
- [atomic_flag](#) & **operator=** (const [atomic_flag](#) &) volatile=delete

Public Attributes

- `__atomic_flag_data_type _M_i`

4.494.1 Detailed Description

`atomic_flag`

Definition at line 275 of file `atomic_base.h`.

The documentation for this struct was generated from the following file:

- [atomic_base.h](#)

4.495 `std::atomic_ptr<_Tp>` Class Template Reference

Public Types

- `typedef _Tp element_type`

Public Member Functions

- `atomic_ptr(element_type *__p=0) throw ()`
- `atomic_ptr(auto_ptr &__a) throw ()`
- `template<typename _Tp1 >
atomic_ptr(auto_ptr<_Tp1> &__a) throw ()`
- `atomic_ptr(auto_ptr_ref<element_type> &__ref) throw ()`
- `~atomic_ptr()`
- `element_type * get() const throw ()`
- `template<typename _Tp1 >
operator auto_ptr<_Tp1>() throw ()`
- `template<typename _Tp1 >
operator auto_ptr_ref<_Tp1>() throw ()`
- `element_type & operator*() const throw ()`
- `element_type * operator->() const throw ()`
- `atomic_ptr & operator=(auto_ptr &__a) throw ()`
- `template<typename _Tp1 >
atomic_ptr & operator=(auto_ptr<_Tp1> &__a) throw ()`
- `atomic_ptr & operator=(auto_ptr_ref<element_type> &__ref) throw ()`
- `element_type * release() throw ()`
- `void reset(element_type *__p=0) throw ()`

4.495.1 Detailed Description

```
template<typename _Tp>  
class std::auto_ptr<_Tp>
```

A simple smart pointer providing strict ownership semantics.

The Standard says:

An `auto_ptr` owns the object it holds a pointer to. Copying an `auto_ptr` copies the pointer and transfers ownership to the destination. If more than one `auto_ptr` owns the same object at the same time the behavior of the program is undefined.

The uses of `auto_ptr` include providing temporary exception-safety for dynamically allocated memory, passing ownership of dynamically allocated memory to a function, and returning dynamically allocated memory from a function. `auto_ptr` does not meet the CopyConstructible requirements for Standard Library `container` elements and thus instantiating a Standard Library container with an `auto_ptr` results in undefined behavior.

Quoted from [20.4.5]/3.

Good examples of what can and cannot be done with `auto_ptr` can be found in the `libstdc++` testsuite.

`_GLIBCXX_RESOLVE_LIB_DEFECTS` 127. `auto_ptr<>` conversion issues These resolutions have all been incorporated.

Definition at line 87 of file `auto_ptr.h`.

4.495.2 Member Typedef Documentation

4.495.2.1 `template<typename _Tp> typedef _Tp std::auto_ptr<_Tp>::element_type`

The pointed-to type.

Definition at line 94 of file `auto_ptr.h`.

4.495.3 Constructor & Destructor Documentation

4.495.3.1 `template<typename _Tp> std::auto_ptr<_Tp>::auto_ptr (element_type *__p = 0) throw () [inline], [explicit]`

An `auto_ptr` is usually constructed from a raw pointer.

Parameters

\leftrightarrow __p	A pointer (defaults to NULL).
--------------------------	-------------------------------

This object now *owns* the object pointed to by __p.

Definition at line 103 of file auto_ptr.h.

4.495.3.2 `template<typename _Tp> std::auto_ptr<_Tp>::auto_ptr (auto_ptr<_Tp> &__a) throw () [inline]`

An auto_ptr can be constructed from another auto_ptr.

Parameters

\leftrightarrow __a	Another auto_ptr of the same type.
--------------------------	------------------------------------

This object now *owns* the object previously owned by __a, which has given up ownership.

Definition at line 112 of file auto_ptr.h.

4.495.3.3 `template<typename _Tp> template<typename _Tp1> std::auto_ptr<_Tp>::auto_ptr (auto_ptr<_Tp1> &__a) throw () [inline]`

An auto_ptr can be constructed from another auto_ptr.

Parameters

\leftrightarrow __a	Another auto_ptr of a different but related type.
--------------------------	---

A pointer-to-Tp1 must be convertible to a pointer-to-Tp/element_type.

This object now *owns* the object previously owned by __a, which has given up ownership.

Definition at line 125 of file auto_ptr.h.

4.495.3.4 `template<typename _Tp> std::auto_ptr<_Tp>::~~auto_ptr () [inline]`

When the auto_ptr goes out of scope, the object it owns is deleted. If it no longer owns anything (i.e., get () is NULL), then this has no effect.

The C++ standard says there is supposed to be an empty throw specification here, but omitting it is standard conforming. Its presence can be detected only if _Tp::~~_Tp() throws, but this is prohibited. [17.4.3.6]/2

Definition at line 170 of file auto_ptr.h.

4.495.3.5 `template<typename _Tp> std::auto_ptr<_Tp>::auto_ptr (auto_ptr_ref< element_type > __ref) throw)`
`[inline]`

Automatic conversions.

These operations convert an `auto_ptr` into and from an `auto_ptr_ref` automatically as needed. This allows constructs such as

```
auto_ptr<Derived> func_returning_auto_ptr(....);
...
auto_ptr<Base> ptr = func_returning_auto_ptr(....);
```

Definition at line 260 of file `auto_ptr.h`.

References `std::move()`, `std::shared_ptr<_Tp>::shared_ptr()`, and `std::unique_ptr<_Tp, _Dp>::unique_ptr()`.

4.495.4 Member Function Documentation

4.495.4.1 `template<typename _Tp> element_type* std::auto_ptr<_Tp>::get () const throw)` `[inline]`

Bypassing the smart pointer.

Returns

The raw pointer being managed.

You can get a copy of the pointer that this object owns, for situations such as passing to a function which only accepts a raw pointer.

Note

This `auto_ptr` still owns the memory.

Definition at line 211 of file `auto_ptr.h`.

4.495.4.2 `template<typename _Tp> element_type& std::auto_ptr<_Tp>::operator* () const throw)` `[inline]`

Smart pointer dereferencing.

If this `auto_ptr` no longer owns anything, then this operation will crash. (For a smart pointer, *no longer owns anything* is the same as being a null pointer, and you know what happens when you dereference one of those...)

Definition at line 181 of file `auto_ptr.h`.

4.495.4.3 `template<typename _Tp> element_type* std::auto_ptr<_Tp>::operator-> () const throw)` `[inline]`

Smart pointer dereferencing.


This returns the pointer itself, which the language then will automatically cause to be dereferenced.

Definition at line 194 of file `auto_ptr.h`.

4.495.4.4 `template<typename _Tp> auto_ptr& std::auto_ptr<_Tp>::operator= (auto_ptr<_Tp> & __a) throw)`
`[inline]`

`auto_ptr` assignment operator.

Parameters

	Another auto_ptr of the same type.
<code>__a</code>	


This object now *owns* the object previously owned by `__a`, which has given up ownership. The object that this one *used* to own and track has been deleted.

Definition at line 136 of file auto_ptr.h.

4.495.4.5 `template<typename _Tp> template<typename _Tp1 > auto_ptr& std::auto_ptr<_Tp>::operator=(auto_ptr<_Tp1 > &__a) throw) [inline]`

auto_ptr assignment operator.

Parameters

	Another auto_ptr of a different but related type.
<code>__a</code>	

A pointer-to-Tp1 must be convertible to a pointer-to-Tp/element_type.

This object now *owns* the object previously owned by `__a`, which has given up ownership. The object that this one *used* to own and track has been deleted.

Definition at line 154 of file auto_ptr.h.

4.495.4.6 `template<typename _Tp> element_type* std::auto_ptr<_Tp>::release () throw) [inline]`

Bypassing the smart pointer.

Returns

The raw pointer being managed.

You can get a copy of the pointer that this object owns, for situations such as passing to a function which only accepts a raw pointer.

Note

This auto_ptr no longer owns the memory. When this object goes out of scope, nothing will happen.

Definition at line 225 of file auto_ptr.h.

4.495.4.7 `template<typename _Tp> void std::auto_ptr<_Tp>::reset (element_type * __p = 0) throw) [inline]`

Forcibly deletes the managed object.

Parameters

<code>__p</code>	A pointer (defaults to NULL).
------------------	-------------------------------

This object now *owns* the object pointed to by `__p`. The previous object has been deleted.

Definition at line 240 of file `auto_ptr.h`.

The documentation for this class was generated from the following file:

- [auto_ptr.h](#)

4.496 `std::auto_ptr_ref<_Tp1>` Struct Template Reference

Public Member Functions

- `auto_ptr_ref` (`_Tp1 *__p`)

Public Attributes

- `_Tp1 * _M_ptr`

4.496.1 Detailed Description

```
template<typename _Tp1>
struct std::auto_ptr_ref<_Tp1>
```

A wrapper class to provide `auto_ptr` with reference semantics. For example, an `auto_ptr` can be assigned (or constructed from) the result of a function which returns an `auto_ptr` by value.

All the `auto_ptr_ref` stuff should happen behind the scenes.

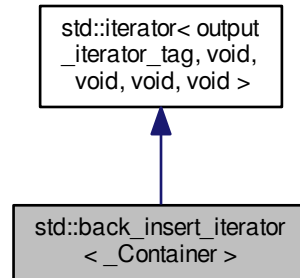
Definition at line 48 of file `auto_ptr.h`.

The documentation for this struct was generated from the following file:

- [auto_ptr.h](#)

4.497 `std::back_insert_iterator<_Container>` Class Template Reference

Inheritance diagram for `std::back_insert_iterator<_Container>`:



Public Types

- typedef `_Container` `container_type`
- typedef void `difference_type`
- typedef `output_iterator_tag` `iterator_category`
- typedef void `pointer`
- typedef void `reference`
- typedef void `value_type`

Public Member Functions

- `back_insert_iterator` (`_Container &__x`)
- `back_insert_iterator` & `operator*` ()
- `back_insert_iterator` & `operator++` ()
- `back_insert_iterator` `operator++` (int)
- `back_insert_iterator` & `operator=` (const typename `_Container::value_type` &__value)
- `back_insert_iterator` & **`operator=`** (typename `_Container::value_type` &&__value)

Protected Attributes

- `_Container *` **`container`**

4.497.1 Detailed Description

```
template<typename _Container>
class std::back_insert_iterator< _Container >
```

Turns assignment into insertion.

These are output iterators, constructed from a container-of-T. Assigning a T to the iterator appends it to the container using `push_back`.

Tip: Using the `back_inserter` function to create these iterators can save typing.

Definition at line 403 of file `stl_iterator.h`.

4.497.2 Member Typedef Documentation

4.497.2.1 `template<typename _Container > typedef _Container std::back_insert_iterator< _Container >::container_type`

A nested typedef for the type of whatever container you used.

Definition at line 411 of file `stl_iterator.h`.

4.497.2.2 `typedef void std::iterator< output_iterator_tag , void , void , void , void >::difference_type` `[inherited]`

Distance between iterators is represented as this type.

Definition at line 125 of file `stl_iterator_base_types.h`.

4.497.2.3 `typedef output_iterator_tag std::iterator< output_iterator_tag , void , void , void , void >::iterator_category` `[inherited]`

One of the [tag types](#).

Definition at line 121 of file `stl_iterator_base_types.h`.

4.497.2.4 `typedef void std::iterator< output_iterator_tag , void , void , void , void >::pointer` `[inherited]`

This type represents a pointer-to-value_type.

Definition at line 127 of file `stl_iterator_base_types.h`.

4.497.2.5 `typedef void std::iterator< output_iterator_tag , void , void , void , void >::reference` `[inherited]`

This type represents a reference-to-value_type.

Definition at line 129 of file `stl_iterator_base_types.h`.

4.497.2.6 typedef void std::iterator< output_iterator_tag, void, void, void, void >::value_type [inherited]

The type "pointed to" by the iterator.

Definition at line 123 of file stl_iterator_base_types.h.

4.497.3 Constructor & Destructor Documentation

4.497.3.1 template<typename _Container> std::back_insert_iterator<_Container>::back_insert_iterator(_Container &__x) [inline], [explicit]

The only way to create this iterator is with a container.

Definition at line 415 of file stl_iterator.h.

4.497.4 Member Function Documentation

4.497.4.1 template<typename _Container> back_insert_iterator& std::back_insert_iterator<_Container>::operator*() [inline]

Simply returns *this.

Definition at line 453 of file stl_iterator.h.

4.497.4.2 template<typename _Container> back_insert_iterator& std::back_insert_iterator<_Container>::operator++() [inline]

Simply returns *this. (This iterator does not *move*.)

Definition at line 458 of file stl_iterator.h.

4.497.4.3 template<typename _Container> back_insert_iterator std::back_insert_iterator<_Container>::operator++(int) [inline]

Simply returns *this. (This iterator does not *move*.)

Definition at line 463 of file stl_iterator.h.

4.497.4.4 template<typename _Container> back_insert_iterator& std::back_insert_iterator<_Container>::operator=(const typename _Container::value_type &__value) [inline]

Parameters

__value	An instance of whatever type container_type::const_reference is; presumably a reference-to-const T for container<T>.
---------	--

Returns

This iterator, for chained operations.

This kind of iterator doesn't really have a *position* in the container (you can think of the position as being permanently at the end, if you like). Assigning a value to the iterator will always append the value to the end of the container.

Definition at line 437 of file `stl_iterator.h`.

References `std::move()`.

The documentation for this class was generated from the following file:

- [stl_iterator.h](#)

4.498 `std::bad_weak_ptr` Class Reference

Inherits exception.

Public Member Functions

- virtual `char const * what () const` noexcept

4.498.1 Detailed Description

Exception possibly thrown by `shared_ptr`.

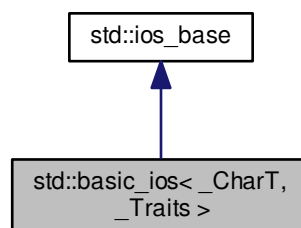
Definition at line 66 of file `shared_ptr_base.h`.

The documentation for this class was generated from the following file:

- [shared_ptr_base.h](#)

4.499 `std::basic_ios<_CharT, _Traits>` Class Template Reference

Inheritance diagram for `std::basic_ios<_CharT, _Traits>`:



Public Types

- enum [event](#) { [erase_event](#), [imbue_event](#), [copyfmt_event](#) }
- typedef void(* [event_callback](#)) ([event](#) __e, [ios_base](#) &__b, int __i)
- typedef [_ios_Fmtflags](#) [fmtflags](#)
- typedef int [io_state](#)
- typedef [_ios_iostate](#) [iostate](#)
- typedef int [open_mode](#)
- typedef [_ios_Openmode](#) [openmode](#)
- typedef int [seek_dir](#)
- typedef [_ios_Seekdir](#) [seekdir](#)
- typedef [std::streamoff](#) [streamoff](#)
- typedef [std::streampos](#) [streampos](#)
- typedef [_CharT](#) [char_type](#)
- typedef [_Traits::int_type](#) [int_type](#)
- typedef [_Traits::pos_type](#) [pos_type](#)
- typedef [_Traits::off_type](#) [off_type](#)
- typedef [_Traits](#) [traits_type](#)
- typedef [ctype](#)< [_CharT](#) > [__ctype_type](#)
- typedef [num_put](#)< [_CharT](#), [ostreambuf_iterator](#)< [_CharT](#), [_Traits](#) > > [__num_put_type](#)
- typedef [num_get](#)< [_CharT](#), [istreambuf_iterator](#)< [_CharT](#), [_Traits](#) > > [__num_get_type](#)

Public Member Functions

- [basic_ios](#) ([basic_streambuf](#)< [_CharT](#), [_Traits](#) > *__sb)
- virtual [~basic_ios](#) ()
- const [locale](#) & [_M_getloc](#) () const
- void [_M_setstate](#) ([iostate](#) __state)
- bool [bad](#) () const
- void [clear](#) ([iostate](#) __state=[goodbit](#))
- [basic_ios](#) & [copyfmt](#) (const [basic_ios](#) &__rhs)
- bool [eof](#) () const
- [iostate](#) [exceptions](#) () const
- void [exceptions](#) ([iostate](#) __except)
- bool [fail](#) () const
- [char_type](#) [fill](#) () const
- [char_type](#) [fill](#) ([char_type](#) __ch)
- [fmtflags](#) [flags](#) () const
- [fmtflags](#) [flags](#) ([fmtflags](#) __fmtfl)
- [locale](#) [getloc](#) () const
- bool [good](#) () const
- [locale](#) [imbue](#) (const [locale](#) &__loc)
- long & [iword](#) (int __ix)
- char [narrow](#) ([char_type](#) __c, char __dfault) const
- [streamsize](#) [precision](#) () const
- [streamsize](#) [precision](#) ([streamsize](#) __prec)
- void *& [pword](#) (int __ix)

- `basic_streambuf< _CharT, _Traits > * rdbuf () const`
 - `basic_streambuf< _CharT, _Traits > * rdbuf (basic_streambuf< _CharT, _Traits > * __sb)`
 - `iosstate rdstate () const`
 - `void register_callback (event_callback __fn, int __index)`
 - `fmtflags setf (fmtflags __fmtfl)`
 - `fmtflags setf (fmtflags __fmtfl, fmtflags __mask)`
 - `void setstate (iosstate __state)`
 - `basic_ostream< _CharT, _Traits > * tie () const`
 - `basic_ostream< _CharT, _Traits > * tie (basic_ostream< _CharT, _Traits > * __tiestr)`
 - `void unsetf (fmtflags __mask)`
 - `char_type widen (char __c) const`
 - `streamsize width () const`
 - `streamsize width (streamsize __wide)`
-
- `operator void * () const`
 - `bool operator! () const`

Static Public Member Functions

- `static bool sync_with_stdio (bool __sync=true)`
- `static int xalloc () throw ()`

Static Public Attributes

- `static const fmtflags adjustfield`
- `static const openmode app`
- `static const openmode ate`
- `static const iosstate badbit`
- `static const fmtflags basefield`
- `static const seekdir beg`
- `static const openmode binary`
- `static const fmtflags boolalpha`
- `static const seekdir cur`
- `static const fmtflags dec`
- `static const seekdir end`
- `static const iosstate eofbit`
- `static const iosstate failbit`
- `static const fmtflags fixed`
- `static const fmtflags floatfield`
- `static const iosstate goodbit`
- `static const fmtflags hex`
- `static const openmode in`
- `static const fmtflags internal`
- `static const fmtflags left`
- `static const fmtflags oct`
- `static const openmode out`
- `static const fmtflags right`
- `static const fmtflags scientific`

- static const [fmtflags](#) showbase
- static const [fmtflags](#) showpoint
- static const [fmtflags](#) showpos
- static const [fmtflags](#) skipws
- static const [openmode](#) trunc
- static const [fmtflags](#) unitbuf
- static const [fmtflags](#) uppercase

Protected Types

- enum { [_S_local_word_size](#) }

Protected Member Functions

- [basic_ios](#) ()
- void [_M_cache_locale](#) (const [locale](#) &__loc)
- void [_M_call_callbacks](#) ([event](#) __ev) throw ()
- void [_M_dispose_callbacks](#) (void) throw ()
- [_Words](#) & [_M_grow_words](#) (int __index, bool __iword)
- void [_M_init](#) () throw ()
- void [init](#) (basic_streambuf< _CharT, _Traits > * __sb)

Protected Attributes

- [_Callback_list](#) * [_M_callbacks](#)
- const [__ctype_type](#) * [_M_ctype](#)
- [iostate](#) [_M_exception](#)
- [char_type](#) [_M_fill](#)
- bool [_M_fill_init](#)
- [fmtflags](#) [_M_flags](#)
- [locale](#) [_M_ios_locale](#)
- [_Words](#) [_M_local_word](#) [[_S_local_word_size](#)]
- const [__num_get_type](#) * [_M_num_get](#)
- const [__num_put_type](#) * [_M_num_put](#)
- [streamsize](#) [_M_precision](#)
- basic_streambuf< _CharT, _Traits > * [_M_streambuf](#)
- [iostate](#) [_M_streambuf_state](#)
- basic_ostream< _CharT, _Traits > * [_M_tie](#)
- [streamsize](#) [_M_width](#)
- [_Words](#) * [_M_word](#)
- int [_M_word_size](#)
- [_Words](#) [_M_word_zero](#)

4.499.1 Detailed Description

```
template<typename _CharT, typename _Traits>
class std::basic_ios< _CharT, _Traits >
```

Template class basic_ios, virtual base class for all stream classes.

Template Parameters

<code>_CharT</code>	Type of character stream.
<code>_Traits</code>	Traits for character type, defaults to <code>char_traits<_CharT></code> .

Most of the member functions called dispatched on stream objects (e.g., `std::cout.foo(bar);`) are consolidated in this class.

Definition at line 66 of file `basic_ios.h`.

4.499.2 Member Typedef Documentation

4.499.2.1 `template<typename _CharT, typename _Traits> typedef ctype<_CharT> std::basic_ios<_CharT, _Traits>::__ctype_type`

These are non-standard types.

Definition at line 86 of file `basic_ios.h`.

4.499.2.2 `template<typename _CharT, typename _Traits> typedef num_get<_CharT, istreambuf_iterator<_CharT, _Traits>> std::basic_ios<_CharT, _Traits>::__num_get_type`

These are non-standard types.

Definition at line 90 of file `basic_ios.h`.

4.499.2.3 `template<typename _CharT, typename _Traits> typedef num_put<_CharT, ostreambuf_iterator<_CharT, _Traits>> std::basic_ios<_CharT, _Traits>::__num_put_type`

These are non-standard types.

Definition at line 88 of file `basic_ios.h`.

4.499.2.4 `template<typename _CharT, typename _Traits> typedef _CharT std::basic_ios<_CharT, _Traits>::__char_type`




These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

Definition at line 75 of file `basic_ios.h`.

4.499.2.5 `typedef void(* std::ios_base::event_callback)(event __e, ios_base &__b, int __i) [inherited]`

The type of an event callback function.

Parameters

 _e	One of the members of the event enum.
 _b	Reference to the ios_base object.
 _i	The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several ios_base and basic_ios functions, specifically imbue(), copyfmt(), and ~ios().

Definition at line 436 of file ios_base.h.

4.499.2.6 typedef _Ios_Fmtflags std::ios_base::fmtflags [inherited]

This is a bitmask type.

_Ios_Fmtflags is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type fmtflags are:

- boolalpha
- dec
- fixed
- hex
- internal
- left
- oct
- right
- scientific
- showbase
- showpoint
- showpos
- skipws
- unitbuf
- uppercase
- adjustfield
- basefield
- floatfield

Definition at line 255 of file ios_base.h.

4.499.2.7 `template<typename _CharT, typename _Traits > typedef _Traits::int_type std::basic_ios< _CharT, _Traits >::int_type`

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

Definition at line 76 of file `basic_ios.h`.

4.499.2.8 `typedef _ios_istate std::ios_base::istate [inherited]`

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `istate` are:

- `badbit`
- `eofbit`
- `failbit`
- `goodbit`

Definition at line 330 of file `ios_base.h`.

4.499.2.9 `template<typename _CharT, typename _Traits > typedef _Traits::off_type std::basic_ios< _CharT, _Traits >::off_type`

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

Definition at line 78 of file `basic_ios.h`.

4.499.2.10 `typedef _ios_Openmode std::ios_base::openmode [inherited]`

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `openmode` are:

- `app`
- `ate`
- `binary`
- `in`
- `out`
- `trunc`

Definition at line 361 of file `ios_base.h`.

4.499.2.11 `template<typename _CharT, typename _Traits > typedef _Traits::pos_type std::basic_ios< _CharT, _Traits >::pos_type`

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

Definition at line 77 of file `basic_ios.h`.

4.499.2.12 `typedef _Ios_Seekdir std::ios_base::seekdir` `[inherited]`

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- `beg`
- `cur`, equivalent to `SEEK_CUR` in the C standard library.
- `end`, equivalent to `SEEK_END` in the C standard library.

Definition at line 393 of file `ios_base.h`.

4.499.2.13 `template<typename _CharT, typename _Traits > typedef _Traits std::basic_ios< _CharT, _Traits >::traits_type`

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

Definition at line 79 of file `basic_ios.h`.

4.499.3 Member Enumeration Documentation

4.499.3.1 `enum std::ios_base::event` `[inherited]`

The set of events that may be passed to an event callback.

`erase_event` is used during `~ios()` and `copyfmt()`. `imbue_event` is used during `imbue()`. `copyfmt_event` is used during `copyfmt()`.

Definition at line 419 of file `ios_base.h`.

4.499.4 Constructor & Destructor Documentation

4.499.4.1 `template<typename _CharT, typename _Traits > std::basic_ios< _CharT, _Traits >::basic_ios (basic_streambuf< _CharT, _Traits > * __sb)` `[inline], [explicit]`

Constructor performs initialization.

The parameter is passed by derived streams.

Definition at line 264 of file `basic_ios.h`.

References `std::basic_ios< _CharT, _Traits >::init()`.

4.499.4.2 `template<typename _CharT, typename _Traits > virtual std::basic_ios< _CharT, _Traits >::~basic_ios ()`
`[inline], [virtual]`

Empty.

The destructor does nothing. More specifically, it does not destroy the streambuf held by rdbuf().

Definition at line 276 of file basic_ios.h.

4.499.4.3 `template<typename _CharT, typename _Traits > std::basic_ios< _CharT, _Traits >::~basic_ios ()` `[inline],`
`[protected]`

Empty.

The default constructor does nothing and is not normally accessible to users.

Definition at line 454 of file basic_ios.h.

References `std::basic_ios< _CharT, _Traits >::init()`.

4.499.5 Member Function Documentation

4.499.5.1 `const locale& std::ios_base::_M_getloc () const` `[inline], [inherited]`

Locale access.

Returns

A reference to the current locale.

Like `getloc` above, but returns a reference instead of generating a copy.

Definition at line 706 of file ios_base.h.

References `std::ios_base::xalloc()`.

4.499.5.2 `template<typename _CharT, typename _Traits > bool std::basic_ios< _CharT, _Traits >::bad () const`
`[inline]`

Fast error checking.

Returns

True if the badbit is set.

Note that other iostate flags may also be set.

Definition at line 205 of file basic_ios.h.

References `std::ios_base::badbit`, and `std::basic_ios< _CharT, _Traits >::rdstate()`.

4.499.5.3 `template<typename _CharT, typename _Traits > void std::basic_ios< _CharT, _Traits >::clear (iostate __state =`
`goodbit)`

[Re]sets the error state.

Parameters

<code>__state</code>	The new state flag(s) to set.
----------------------	-------------------------------

See `std::ios_base::iostate` for the possible bit values. Most users will not need to pass an argument.

Referenced by `std::basic_ios< _CharT, _Traits >::exceptions()`, `std::basic_ios< _CharT, _Traits >::rdstate()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

4.499.5.4 `template<typename _CharT, typename _Traits> basic_ios& std::basic_ios< _CharT, _Traits >::copyfmt (const basic_ios< _CharT, _Traits > &__rhs)`

Copies fields of `__rhs` into this.

Parameters

<code>__rhs</code>	The source values for the copies.
--------------------	-----------------------------------

Returns

Reference to this object.

All fields of `__rhs` are copied into this object except that `rdbuf()` and `rdstate()` remain unchanged. All values in the `pword` and `iword` arrays are copied. Before copying, each callback is invoked with `erase_event`. After copying, each (new) callback is invoked with `copyfmt_event`. The final step is to copy `exceptions()`.

Referenced by `std::basic_ios< _CharT, _Traits >::rdbuf()`.

4.499.5.5 `template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::eof () const [inline]`

Fast error checking.

Returns

True if the eofbit is set.

Note that other `iostate` flags may also be set.

Definition at line 184 of file `basic_ios.h`.

References `std::ios_base::eofbit`, and `std::basic_ios< _CharT, _Traits >::rdstate()`.

4.499.5.6 `template<typename _CharT, typename _Traits > ios_base::basic_ios< _CharT, _Traits >::exceptions () const`
`[inline]`

Throwing exceptions on errors.

Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of `exceptions(ios_base)` for the meaning of the return value.

Definition at line 216 of file `basic_ios.h`.

Referenced by `std::basic_ios< _CharT, _Traits >::setstate()`.

4.499.5.7 `template<typename _CharT, typename _Traits > void std::basic_ios< _CharT, _Traits >::exceptions (ios_base::__except)`
`[inline]`

Throwing exceptions on errors.

Parameters

<code>__except</code>	The new exceptions mask.
-----------------------	--------------------------

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type `std::ios_base::failure` is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
    std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

    std::ifstream f ("/etc/motd");

    std::cerr << "Setting badbit\n";
    f.setstate (std::ios_base::badbit);

    std::cerr << "Setting exception mask\n";
    f.exceptions (std::ios_base::badbit);
}
```

Definition at line 251 of file `basic_ios.h`.

References `std::basic_ios< _CharT, _Traits >::clear()`.

4.499.5.8 `template<typename _CharT, typename _Traits > bool std::basic_ios< _CharT, _Traits >::fail () const`
`[inline]`

Fast error checking.

Returns

True if either the badbit or the failbit is set.

Checking the badbit in fail() is historical practice. Note that other iostate flags may also be set.

Definition at line 195 of file basic_ios.h.

References std::ios_base::badbit, std::ios_base::failbit, and std::basic_ios< _CharT, _Traits >::rdstate().

Referenced by std::basic_ios< _CharT, _Traits >::operator void *(), and std::basic_ios< _CharT, _Traits >::operator!().

4.499.5.9 `template<typename _CharT, typename _Traits > char_type std::basic_ios< _CharT, _Traits >::fill () const`
`[inline]`

Retrieves the *empty* character.

Returns

The current fill character.

It defaults to a space (' ') in the current locale.

Definition at line 364 of file basic_ios.h.

References std::basic_ios< _CharT, _Traits >::widen().

Referenced by std::basic_ios< _CharT, _Traits >::fill().

4.499.5.10 `template<typename _CharT, typename _Traits > char_type std::basic_ios< _CharT, _Traits >::fill (char_type`
`__ch) [inline]`

Sets a new *empty* character.

Parameters

<code>__ch</code>	The new character.
-------------------	--------------------

Returns

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via setw), Q characters are actually used, and Q<P. It defaults to a space (' ') in the current locale.

Definition at line 384 of file `basic_ios.h`.

References `std::basic_ios<_CharT, _Traits>::fill()`, and `std::basic_ios<_CharT, _Traits>::imbue()`.

4.499.5.11 `fmtflags std::ios_base::flags () const` `[inline]`, `[inherited]`

Access to format flags.

Returns

The format control flags for both input and output.

Definition at line 551 of file `ios_base.h`.

4.499.5.12 `fmtflags std::ios_base::flags (fmtflags __fmtfl)` `[inline]`, `[inherited]`

Setting new format flags all at once.

Parameters

<code>__fmtfl</code>	The new flags to set.
----------------------	-----------------------

Returns

The previous format control flags.

This function overwrites all the format flags with `__fmtfl`.

Definition at line 562 of file `ios_base.h`.

4.499.5.13 `locale std::ios_base::getloc () const` `[inline]`, `[inherited]`

Locale access.

Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Definition at line 695 of file `ios_base.h`.

4.499.5.14 `template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::good () const`
`[inline]`

Fast error checking.

Returns

True if no error flags are set.

A wrapper around `rdstate`.

Definition at line 174 of file `basic_ios.h`.

References `std::basic_ios< _CharT, _Traits >::rdstate()`.

4.499.5.15 `template<typename _CharT, typename _Traits> locale std::basic_ios< _CharT, _Traits >::imbue (const locale & __loc)`

Moves to a new locale.

Parameters

<code>__loc</code>	The new locale.
--------------------	-----------------

Returns

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>.

Referenced by `std::basic_ios< _CharT, _Traits >::fill()`.

4.499.5.16 `template<typename _CharT, typename _Traits> void std::basic_ios< _CharT, _Traits >::init (basic_streambuf< _CharT, _Traits > * __sb)` `[protected]`

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Referenced by `std::basic_ios< _CharT, _Traits >::basic_ios()`.

4.499.5.17 `long& std::ios_base::iword (int __ix)` `[inline]`, `[inherited]`

Access to integer array.

Parameters

<code>_↵</code> <code>_ix</code>	Index into the array.
-------------------------------------	-----------------------

Returns

A reference to an integer associated with the index.

The `iword` function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 741 of file `ios_base.h`.

4.499.5.18 `template<typename _CharT, typename _Traits> char std::basic_ios<_CharT, _Traits>::narrow (char_type __c, char __default) const [inline]`

Squeezes characters.

Parameters

<code>__c</code>	The character to narrow.
<code>__default</code>	The character to narrow.

Returns

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).narrow(c, default)
```

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.↵html>

Definition at line 424 of file `basic_ios.h`.

4.499.5.19 `template<typename _CharT, typename _Traits> std::basic_ios<_CharT, _Traits>::operator void * () const [inline]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

Definition at line 115 of file `basic_ios.h`.

References `std::basic_ios<_CharT, _Traits>::fail()`.

4.499.5.20 `template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::operator! () const`
`[inline]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 119 of file `basic_ios.h`.

References `std::basic_ios< _CharT, _Traits >::fail()`.

4.499.5.21 `streamsize std::ios_base::precision () const` `[inline],[inherited]`

Flags access.

Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 621 of file `ios_base.h`.

4.499.5.22 `streamsize std::ios_base::precision (streamsize __prec)` `[inline],[inherited]`

Changing flags.

Parameters

<code>__prec</code>	The new precision value.
---------------------	--------------------------

Returns

The previous value of `precision()`.

Definition at line 630 of file `ios_base.h`.

4.499.5.23 `void*& std::ios_base::pword (int __ix)` `[inline],[inherited]`

Access to void pointer array.

Parameters

<code>__ix</code>	Index into the array.
-------------------	-----------------------

Returns

A reference to a `void*` associated with the index.

The `pwd` function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 762 of file `ios_base.h`.

References `std::ios_base::~~ios_base()`.

4.499.5.24 `template<typename _CharT, typename _Traits> basic_streambuf<_CharT, _Traits>* std::basic_ios<_CharT, _Traits>::rdbuf () const [inline]`

Accessing the underlying buffer.

Returns

The current stream buffer.

This does not change the state of the stream.

Definition at line 315 of file `basic_ios.h`.

References `std::basic_ios<_CharT, _Traits>::copyfmt()`.

4.499.5.25 `template<typename _CharT, typename _Traits> basic_streambuf<_CharT, _Traits>* std::basic_ios<_CharT, _Traits>::rdbuf (basic_streambuf<_CharT, _Traits> * __sb)`

Changing the underlying buffer.

Parameters

<code>__sb</code>	The new stream buffer.
-------------------	------------------------

Returns

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream    foo;           // or some other derived type
std::streambuf* p = .....;

foo.ios::rdbuf(p);             // ios == basic_ios<char>
```


4.499.5.26 `template<typename _CharT, typename _Traits> iostate std::basic_ios< _CharT, _Traits >::rdstate () const`
`[inline]`

Returns the error state of the stream buffer.

Returns

A bit pattern (well, isn't everything?)

See `std::ios_base::iostate` for the possible bit values. Most users will call one of the interpreting wrappers, e.g., `good()`.

Definition at line 131 of file `basic_ios.h`.

References `std::basic_ios< _CharT, _Traits >::clear()`, and `std::ios_base::goodbit`.

Referenced by `std::basic_ios< _CharT, _Traits >::bad()`, `std::basic_ios< _CharT, _Traits >::eof()`, `std::basic_ios< _CharT, _Traits >::fail()`, `std::basic_ios< _CharT, _Traits >::good()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

4.499.5.27 `void std::ios_base::register_callback (event_callback __fn, int __index)` `[inherited]`

Add the callback `__fn` with parameter `__index`.

Parameters

<code>__fn</code>	The function to add.
<code>__index</code>	The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

4.499.5.28 `fmtflags std::ios_base::setf (fmtflags __fmtfl)` `[inline],[inherited]`

Setting new format flags.

Parameters

<code>__fmtfl</code>	Additional flags to set.
----------------------	--------------------------

Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 578 of file `ios_base.h`.

Referenced by `std::boolalpha()`, `std::dec()`, `std::fixed()`, `std::hex()`, `std::left()`, `std::oct()`, `std::right()`, `std::scientific()`, `std::showbase()`, `std::showpoint()`, `std::showpos()`, `std::skipws()`, `std::unitbuf()`, and `std::uppercase()`.

4.499.5.29 `fmtflags std::ios_base::set(fmtflags __fmtfl, fmtflags __mask)` `[inline],[inherited]`

Setting new format flags.

Parameters

<code>__fmtfl</code>	Additional flags to set.
<code>__mask</code>	The flags mask for <code>fmtfl</code> .

Returns

The previous format control flags.

This function clears `mask` in the format flags, then sets `fmtfl` & `mask`. An example mask is `ios_base::adjustfield`.

Definition at line 595 of file `ios_base.h`.

4.499.5.30 `template<typename _CharT, typename _Traits> void std::basic_ios<_CharT, _Traits>::setstate(iostate __state)` `[inline]`

Sets additional flags in the error state.

Parameters

<code>__state</code>	The additional state flag(s) to set.
----------------------	--------------------------------------

See `std::ios_base::iostate` for the possible bit values.

Definition at line 151 of file `basic_ios.h`.

References `std::basic_ios<_CharT, _Traits>::clear()`, `std::basic_ios<_CharT, _Traits>::exceptions()`, and `std::basic_ios<_CharT, _Traits>::rdstate()`.

4.499.5.31 `static bool std::ios_base::sync_with_stdio(bool __sync = true)` `[static],[inherited]`

Interaction with the standard C I/O objects.

Parameters

<code>__sync</code>	Whether to synchronize or not.
---------------------	--------------------------------

Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., stdout) and the standard C++ objects (e.g., cout). User-declared streams are unaffected. See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt1.html>

Referenced by std::ios_base::width().

4.499.5.32 `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits>::tie() const [inline]`

Fetches the current *tied* stream.

Returns

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Definition at line 289 of file basic_ios.h.

4.499.5.33 `template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits>::tie(basic_ostream<_CharT, _Traits> * __tiestr) [inline]`

Ties this stream to an output stream.

Parameters

<code>__tiestr</code>	The output stream.
-----------------------	--------------------

Returns

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see tie() for more.

Definition at line 301 of file basic_ios.h.

4.499.5.34 `void std::ios_base::unsetf(fmtflags __mask) [inline], [inherited]`

Clearing format flags.

Parameters

<code>__mask</code>	The flags to unset.
---------------------	---------------------

This function clears `__mask` in the format flags.

Definition at line 610 of file `ios_base.h`.

Referenced by `std::noboolalpha()`, `std::noshowbase()`, `std::noshowpoint()`, `std::noshowpos()`, `std::noskipws()`, `std::nunitbuf()`, and `std::nouppercase()`.

4.499.5.35 `template<typename _CharT, typename _Traits> char_type std::basic_ios<_CharT, _Traits>::widen (char __c)`
`const [inline]`

Widens characters.

Parameters

<code>__c</code>	The character to widen.
------------------	-------------------------

Returns

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).widen(c)
```

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 443 of file `basic_ios.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::fill()`.

4.499.5.36 `streamsize std::ios_base::width () const [inline], [inherited]`

Flags access.

Returns

The minimum field width to generate on output operations.

Minimum field width refers to the number of characters.

Definition at line 644 of file `ios_base.h`.

4.499.5.37 `streamsize std::ios_base::width (streamsize __wide) [inline], [inherited]`

Changing flags.

Parameters

<code>__wide</code>	The new width value.
---------------------	----------------------

Returns

The previous value of `width()`.

Definition at line 653 of file `ios_base.h`.

References `std::ios_base::imbue()`, and `std::ios_base::sync_with_stdio()`.

4.499.5.38 `static int std::ios_base::xalloc () throw` `[static], [inherited]`

Access to unique indices.

Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

Referenced by `std::ios_base::_M_getloc()`.

4.499.6 Member Data Documentation

4.499.6.1 `const fmtflags std::ios_base::adjustfield` `[static], [inherited]`

A mask of `left|right|internal`. Useful for the 2-arg form of `setf`.

Definition at line 310 of file `ios_base.h`.

Referenced by `std::internal()`, `std::left()`, and `std::right()`.

4.499.6.2 `const openmode std::ios_base::app` `[static], [inherited]`

Seek to end before each write.

Definition at line 364 of file `ios_base.h`.

4.499.6.3 `const openmode std::ios_base::ate` `[static], [inherited]`

Open and seek to end immediately after opening.

Definition at line 367 of file `ios_base.h`.

4.499.6.4 `const iostate std::ios_base::badbit` `[static], [inherited]`

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Definition at line 334 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::bad()`, and `std::basic_ios<_CharT, _Traits>::fail()`.

4.499.6.5 `const fmtflags std::ios_base::basefield` `[static], [inherited]`

A mask of `dec|oct|hex`. Useful for the 2-arg form of `setf`.

Definition at line 313 of file `ios_base.h`.

Referenced by `std::dec()`, `std::hex()`, and `std::oct()`.

4.499.6.6 `const seekdir std::ios_base::beg` `[static], [inherited]`

Request a seek relative to the beginning of the stream.

Definition at line 396 of file `ios_base.h`.

Referenced by `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>::file()`.

4.499.6.7 `const openmode std::ios_base::binary` `[static], [inherited]`

Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch27s02.html>.

Definition at line 372 of file `ios_base.h`.

4.499.6.8 `const fmtflags std::ios_base::boolalpha` `[static], [inherited]`

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 258 of file `ios_base.h`.

Referenced by `std::boolalpha()`, and `std::noboolalpha()`.

4.499.6.9 `const seekdir std::ios_base::cur` `[static], [inherited]`

Request a seek relative to the current position within the sequence.

Definition at line 399 of file `ios_base.h`.

Referenced by `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>::file()`.

4.499.6.10 `const fmtflags std::ios_base::dec` `[static], [inherited]`

Converts integer input or generates integer output in decimal base.

Definition at line 261 of file `ios_base.h`.

Referenced by `std::dec()`.

4.499.6.11 `const seekdir std::ios_base::end` `[static], [inherited]`

Request a seek relative to the current end of the sequence.

Definition at line 402 of file `ios_base.h`.

4.499.6.12 `const iostate std::ios_base::eofbit` `[static], [inherited]`

Indicates that an input operation reached the end of an input sequence.

Definition at line 337 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::eof()`.

4.499.6.13 `const iostate std::ios_base::failbit` `[static], [inherited]`

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.

Definition at line 342 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::fail()`.

4.499.6.14 `const fmtflags std::ios_base::fixed` `[static], [inherited]`

Generate floating-point output in fixed-point notation.

Definition at line 264 of file `ios_base.h`.

Referenced by `std::fixed()`.

4.499.6.15 `const fmtflags std::ios_base::floatfield` `[static], [inherited]`

A mask of `scientific|fixed`. Useful for the 2-arg form of `setf`.

Definition at line 316 of file `ios_base.h`.

Referenced by `std::fixed()`, and `std::scientific()`.

4.499.6.16 `const ios_base::goodbit` `[static], [inherited]`

Indicates all is well.

Definition at line 345 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::rdstate()`.

4.499.6.17 `const fmtflags std::ios_base::hex` `[static], [inherited]`

Converts integer input or generates integer output in hexadecimal base.

Definition at line 267 of file `ios_base.h`.

Referenced by `std::hex()`.

4.499.6.18 `const openmode std::ios_base::in` `[static], [inherited]`

Open for input. Default for `ifstream` and `fstream`.

Definition at line 375 of file `ios_base.h`.

Referenced by `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>::file()`.

4.499.6.19 `const fmtflags std::ios_base::internal` `[static], [inherited]`

Adds fill characters at a designated internal point in certain generated output, or identical to `right` if no such point is designated.

Definition at line 272 of file `ios_base.h`.

Referenced by `std::internal()`.

4.499.6.20 `const fmtflags std::ios_base::left` `[static], [inherited]`

Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.)

Definition at line 276 of file `ios_base.h`.

Referenced by `std::left()`.

4.499.6.21 `const fmtflags std::ios_base::oct` `[static], [inherited]`

Converts integer input or generates integer output in octal base.

Definition at line 279 of file `ios_base.h`.

Referenced by `std::oct()`.

4.499.6.22 `const openmode std::ios_base::out` `[static], [inherited]`

Open for output. Default for `ofstream` and `fstream`.

Definition at line 378 of file `ios_base.h`.

Referenced by `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>::file()`.

4.499.6.23 `const fmtflags std::ios_base::right` `[static], [inherited]`

Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.)

Definition at line 283 of file `ios_base.h`.

Referenced by `std::right()`.

4.499.6.24 `const fmtflags std::ios_base::scientific` `[static], [inherited]`

Generates floating-point output in scientific notation.

Definition at line 286 of file `ios_base.h`.

Referenced by `std::scientific()`.

4.499.6.25 `const fmtflags std::ios_base::showbase` `[static], [inherited]`

Generates a prefix indicating the numeric base of generated integer output.

Definition at line 290 of file `ios_base.h`.

Referenced by `std::noshowbase()`, and `std::showbase()`.

4.499.6.26 `const fmtflags std::ios_base::showpoint` `[static], [inherited]`

Generates a decimal-point character unconditionally in generated floating-point output.

Definition at line 294 of file `ios_base.h`.

Referenced by `std::noshowpoint()`, and `std::showpoint()`.

4.499.6.27 `const fmtflags std::ios_base::showpos` `[static], [inherited]`

Generates a + sign in non-negative generated numeric output.

Definition at line 297 of file `ios_base.h`.

Referenced by `std::noshowpos()`, and `std::showpos()`.

4.499.6.28 `const fmtflags std::ios_base::skipws` `[static], [inherited]`

Skips leading white space before certain input operations.

Definition at line 300 of file `ios_base.h`.

Referenced by `std::noskipws()`, and `std::skipws()`.

4.499.6.29 `const openmode std::ios_base::trunc` `[static], [inherited]`

Open for input. Default for `ofstream`.

Definition at line 381 of file `ios_base.h`.

4.499.6.30 `const fmtflags std::ios_base::unitbuf` `[static], [inherited]`

Flushes output after each output operation.

Definition at line 303 of file `ios_base.h`.

Referenced by `std::nounitbuf()`, and `std::unitbuf()`.

4.499.6.31 `const fmtflags std::ios_base::uppercase` `[static], [inherited]`

Replaces certain lowercase letters with their uppercase equivalents in generated output.

Definition at line 307 of file `ios_base.h`.

Referenced by `std::nouppercase()`, and `std::uppercase()`.

The documentation for this class was generated from the following file:

- [basic_ios.h](#)

4.500 `std::basic_regex<_Ch_type, _Rx_traits>` Class Template Reference

Public Types

- typedef [regex_constants::syntax_option_type](#) **flag_type**
- typedef `traits_type::locale_type` **locale_type**
- typedef `traits_type::string_type` **string_type**
- typedef `_Rx_traits` **traits_type**
- typedef `_Ch_type` **value_type**

Public Member Functions

- `basic_regex` ()
- `basic_regex` (const `_Ch_type` * __p, `flag_type` __f=ECMAScript)
- `basic_regex` (const `_Ch_type` * __p, `std::size_t` __len, `flag_type` __f=ECMAScript)
- `basic_regex` (const `basic_regex` & __rhs)
- `basic_regex` (`basic_regex` && __rhs)
- `template<typename _Ch_traits, typename _Ch_alloc >`
`basic_regex` (const `std::basic_string`< `_Ch_type`, `_Ch_traits`, `_Ch_alloc` > & __s, `flag_type` __f=ECMAScript)
- `template<typename _FwdIter >`
`basic_regex` (_FwdIter __first, _FwdIter __last, `flag_type` __f=ECMAScript)
- `basic_regex` (initializer_list< `_Ch_type` > __l, `flag_type` __f=ECMAScript)
- `~basic_regex` ()
- `basic_regex` & `assign` (const `basic_regex` & __rhs)
- `basic_regex` & `assign` (`basic_regex` && __rhs)
- `basic_regex` & `assign` (const `_Ch_type` * __p, `flag_type` __flags=ECMAScript)
- `basic_regex` & `assign` (const `_Ch_type` * __p, `std::size_t` __len, `flag_type` __flags)
- `template<typename _Ch_traits, typename _Alloc >`
`basic_regex` & `assign` (const `basic_string`< `_Ch_type`, `_Ch_traits`, `_Alloc` > & __s, `flag_type` __flags=ECMAScript)
- `template<typename _InputIterator >`
`basic_regex` & `assign` (_InputIterator __first, _InputIterator __last, `flag_type` __flags=ECMAScript)
- `basic_regex` & `assign` (initializer_list< `_Ch_type` > __l, `flag_type` __flags=ECMAScript)
- `flag_type flags` () const
- `locale_type getloc` () const
- `locale_type imbue` (`locale_type` __loc)
- `unsigned int mark_count` () const
- `basic_regex` & `operator=` (const `basic_regex` & __rhs)
- `basic_regex` & `operator=` (`basic_regex` && __rhs)
- `basic_regex` & `operator=` (const `_Ch_type` * __p)
- `basic_regex` & `operator=` (initializer_list< `_Ch_type` > __l)
- `template<typename _Ch_traits, typename _Alloc >`
`basic_regex` & `operator=` (const `basic_string`< `_Ch_type`, `_Ch_traits`, `_Alloc` > & __s)
- `void swap` (`basic_regex` & __rhs)

Static Public Attributes

Constants

std [28.8.1](1)

- static constexpr `flag_type` `icase`
- static constexpr `flag_type` `nosubs`
- static constexpr `flag_type` `optimize`
- static constexpr `flag_type` `collate`
- static constexpr `flag_type` `ECMAScript`
- static constexpr `flag_type` `basic`
- static constexpr `flag_type` `extended`
- static constexpr `flag_type` `awk`
- static constexpr `flag_type` `grep`
- static constexpr `flag_type` `egrep`

Friends

- `template<typename _Bp, typename _Ap, typename _Cp, typename _Rp, __detail::_RegexExecutorPolicy, bool >`
`bool __detail::_regex_algo_impl (_Bp, _Bp, match_results< _Bp, _Ap > &, const basic_regex< _Cp, _Rp >`
`&, regex_constants::match_flag_type)`
- `template<typename, typename, typename, bool >`
`class __detail::_Executor`

4.500.1 Detailed Description

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
class std::basic_regex<_Ch_type, _Rx_traits >
```

Objects of specializations of this class represent regular expressions constructed from sequences of character type `_Ch_type`.

Storage for the regular expression is allocated and deallocated as necessary by the member functions of this class.

Definition at line 35 of file `regex.h`.

4.500.2 Constructor & Destructor Documentation

4.500.2.1 `template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> std::basic_regex<_Ch_type,`
`_Rx_traits >::basic_regex () [inline]`

Constructs a basic regular expression that does not match any character sequence.

Definition at line 435 of file `regex.h`.

4.500.2.2 `template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> std::basic_regex<_Ch_type,`
`_Rx_traits >::basic_regex (const _Ch_type * __p, flag_type __f = ECMAScript) [inline],`
`[explicit]`

Constructs a basic regular expression from the sequence `[__p, __p + char_traits<_Ch_type>::length(__p))` interpreted according to the flags in `__f`.

Parameters

<code>__p</code>	A pointer to the start of a C-style null-terminated string containing a regular expression.
<code>__f</code>	Flags indicating the syntax rules and options.

Exceptions

<code>regex_error</code>	if <code>__p</code> is not a valid regular expression.
--------------------------	--

Definition at line 451 of file regex.h.

```
4.500.2.3 template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> std::basic_regex<_Ch_type,
    _Rx_traits>::basic_regex ( const _Ch_type * __p, std::size_t __len, flag_type __f = ECMAScript )
    [inline]
```

Constructs a basic regular expression from the sequence [p, p + len) interpreted according to the flags in f.

Parameters

<code>__p</code>	A pointer to the start of a string containing a regular expression.
<code>__len</code>	The length of the string containing the regular expression.
<code>__f</code>	Flags indicating the syntax rules and options.

Exceptions

<code>regex_error</code>	if <code>__p</code> is not a valid regular expression.
--------------------------	--

Definition at line 467 of file regex.h.

```
4.500.2.4 template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> std::basic_regex<_Ch_type,
    _Rx_traits>::basic_regex ( const basic_regex<_Ch_type, _Rx_traits> & __rhs ) [inline]
```

Copy-constructs a basic regular expression.

Parameters

<code>__rhs</code>	A regex object.
--------------------	-----------------

Definition at line 477 of file regex.h.

References `std::basic_regex< _Ch_type, _Rx_traits >::getloc()`.

```
4.500.2.5 template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> std::basic_regex<_Ch_type,
    _Rx_traits>::basic_regex ( basic_regex<_Ch_type, _Rx_traits> && __rhs ) [inline]
```

Move-constructs a basic regular expression.

Parameters

<code>__rhs</code>	A regex object.
--------------------	-----------------

The implementation is a workaround concerning ABI compatibility. See: <https://gcc.gnu.org/ml/libstdc++/2014-09/msg00001.html>

Definition at line 492 of file regex.h.

```
4.500.2.6  template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>> template<typename _Ch_traits ,
            typename _Ch_alloc > std::basic_regex<_Ch_type, _Rx_traits >::basic_regex ( const std::basic_string<
            _Ch_type, _Ch_traits, _Ch_alloc > &__s, flag_type __f=ECMAScript ) [inline],[explicit]
```

Constructs a basic regular expression from the string `s` interpreted according to the flags in `f`.

Parameters

<code>__s</code>	A string containing a regular expression.
<code>__f</code>	Flags indicating the syntax rules and options.

Exceptions

<code>regex_error</code>	if <code>__s</code> is not a valid regular expression.
--------------------------	--

Definition at line 512 of file `regex.h`.

```
4.500.2.7  template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>> template<typename _Fwdlter >
            std::basic_regex<_Ch_type, _Rx_traits >::basic_regex ( _Fwdlter __first, _Fwdlter __last, flag_type __f =
            ECMAScript ) [inline]
```

Constructs a basic regular expression from the range `[first, last)` interpreted according to the flags in `f`.

Parameters

<code>__first</code>	The start of a range containing a valid regular expression.
<code>__last</code>	The end of a range containing a valid regular expression.
<code>__f</code>	The format flags of the regular expression.

Exceptions











<code>regex_error</code>	if <code>[__first, __last)</code> is not a valid regular expression.
--------------------------	--

Definition at line 532 of file `regex.h`.

```
4.500.2.8  template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>> std::basic_regex<_Ch_type,
            _Rx_traits >::basic_regex ( initializer_list<_Ch_type > __l, flag_type __f=ECMAScript ) [inline]
```

Constructs a basic regular expression from an initializer list.

Parameters

	The initializer list.
	
	
	
	
	The format flags of the regular expression.
	
	
	
	

Exceptions

<i>regex_error</i>	if <code>__l</code> is not a valid regular expression.
--------------------	--

Definition at line 551 of file regex.h.

4.500.2.9 `template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> std::basic_regex<_Ch_type, _Rx_traits>::~basic_regex () [inline]`

Destroys a basic regular expression.

Definition at line 558 of file regex.h.

4.500.3 Member Function Documentation

4.500.3.1 `template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> basic_regex& std::basic_regex<_Ch_type, _Rx_traits>::assign (const basic_regex<_Ch_type, _Rx_traits> &__rhs) [inline]`

the real assignment operator.

Parameters

<code>__rhs</code>	Another regular expression object.
--------------------	------------------------------------

Definition at line 619 of file regex.h.

References `std::basic_regex<_Ch_type, _Rx_traits>::getloc()`.

Referenced by `std::basic_regex<_Ch_type, _Rx_traits>::swap()`.

4.500.3.2 `template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> basic_regex& std::basic_regex<_Ch_type, _Rx_traits>::assign (basic_regex<_Ch_type, _Rx_traits> &&__rhs) [inline]`

The move-assignment operator.

Parameters

<code>__rhs</code>	Another regular expression object.
--------------------	------------------------------------

The implementation is a workaround concerning ABI compatibility. See: <https://gcc.gnu.org/ml/libstdc++/2014-09/msg00001.html>

Definition at line 637 of file `regex.h`.

References `std::move()`.

4.500.3.3 `template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> basic_regex& std::basic_regex<_Ch_type, _Rx_traits>::assign (const _Ch_type * __p, flag_type __flags = ECMAScript) [inline]`

Assigns a new regular expression to a regex object from a C-style null-terminated string containing a regular expression pattern.

Parameters

<code>__p</code>	A pointer to a C-style null-terminated string containing a regular expression pattern.
<code>__flags</code>	Syntax option flags.

Exceptions

<code>regex_error</code>	if <code>__p</code> does not contain a valid regular expression pattern interpreted according to <code>__flags</code> . If <code>regex_error</code> is thrown, <code>*this</code> remains unchanged.
--------------------------	--

Definition at line 661 of file `regex.h`.

4.500.3.4 `template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> basic_regex& std::basic_regex<_Ch_type, _Rx_traits>::assign (const _Ch_type * __p, std::size_t __len, flag_type __flags) [inline]`

Assigns a new regular expression to a regex object from a C-style string containing a regular expression pattern.

Parameters

<code>__p</code>	A pointer to a C-style string containing a regular expression pattern.
<code>__len</code>	The length of the regular expression pattern string.
<code>__flags</code>	Syntax option flags.

Exceptions

<code>regex_error</code>	if <code>p</code> does not contain a valid regular expression pattern interpreted according to <code>__flags</code> . If <code>regex_error</code> is thrown, <code>*this</code> remains unchanged.
--------------------------	--

Definition at line 678 of file regex.h.

4.500.3.5 `template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> template<typename _Ch_type, typename _Alloc > basic_regex& std::basic_regex<_Ch_type, _Rx_traits>::assign (const basic_string<_Ch_type, _Ch_type_traits, _Alloc> &__s, flag_type __flags = ECMAScript) [inline]`

Assigns a new regular expression to a regex object from a string containing a regular expression pattern.

Parameters

<code>__s</code>	A string containing a regular expression pattern.
<code>__flags</code>	Syntax option flags.

Exceptions

<code>regex_error</code>	if <code>__s</code> does not contain a valid regular expression pattern interpreted according to <code>__flags</code> . If <code>regex_error</code> is thrown, <code>*this</code> remains unchanged.
--------------------------	--

Definition at line 694 of file regex.h.

References `std::basic_string<_CharT, _Traits, _Alloc>::data()`, and `std::basic_string<_CharT, _Traits, _Alloc>::size()`.

4.500.3.6 `template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> template<typename _InputIterator > basic_regex& std::basic_regex<_Ch_type, _Rx_traits>::assign (_InputIterator __first, _InputIterator __last, flag_type __flags = ECMAScript) [inline]`

Assigns a new regular expression to a regex object.

Parameters

<code>__first</code>	The start of a range containing a valid regular expression.
<code>__last</code>	The end of a range containing a valid regular expression.
<code>__flags</code>	Syntax option flags.

Exceptions

<code>regex_error</code>	if <code>p</code> does not contain a valid regular expression pattern interpreted according to <code>__flags</code> . If <code>regex_error</code> is thrown, the object remains unchanged.
--------------------------	--

Definition at line 719 of file regex.h.

4.500.3.7 `template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> basic_regex& std::basic_regex<_Ch_type, _Rx_traits>::assign (initializer_list<_Ch_type> __l, flag_type __flags = ECMAScript) [inline]`

Assigns a new regular expression to a regex object.

Parameters

<code>__l</code>	An initializer list representing a regular expression.
<code>__flags</code>	Syntax option flags.

Exceptions

<code>regex_error</code>	if <code>__l</code> does not contain a valid regular expression pattern interpreted according to <code>__flags</code> . If <code>regex_error</code> is thrown, the object remains unchanged.
--------------------------	--

Definition at line 735 of file `regex.h`.

4.500.3.8 `template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> flag_type std::basic_regex<_Ch_type, _Rx_traits>::flags() const [inline]`

Gets the flags used to construct the regular expression or in the last call to `assign()`.

Definition at line 756 of file `regex.h`.

4.500.3.9 `template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> locale_type std::basic_regex<_Ch_type, _Rx_traits>::getloc() const [inline]`

Gets the locale currently imbued in the regular expression object.

Definition at line 777 of file `regex.h`.

Referenced by `std::basic_regex<_Ch_type, _Rx_traits>::assign()`, and `std::basic_regex<_Ch_type, _Rx_traits>::basic_regex()`.

4.500.3.10 `template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> locale_type std::basic_regex<_Ch_type, _Rx_traits>::imbue(locale_type __loc) [inline]`

Imbues the regular expression object with the given locale.

Parameters

<code>__loc</code>	A locale.
--------------------	-----------

Definition at line 766 of file `regex.h`.

4.500.3.11 `template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> unsigned int std::basic_regex<_Ch_type, _Rx_traits>::mark_count() const [inline]`

Gets the number of marked subexpressions within the regular expression.

Definition at line 744 of file `regex.h`.

```
4.500.3.12 template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> basic_regex&
std::basic_regex< _Ch_type, _Rx_traits >::operator= ( const basic_regex< _Ch_type, _Rx_traits > & __rhs )
[inline]
```

Assigns one regular expression to another.

Definition at line 565 of file regex.h.

```
4.500.3.13 template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> basic_regex&
std::basic_regex< _Ch_type, _Rx_traits >::operator= ( basic_regex< _Ch_type, _Rx_traits > && __rhs )
[inline]
```

Move-assigns one regular expression to another.

The implementation is a workaround concerning ABI compatibility. See: <https://gcc.gnu.org/ml/libstdc++/2014-09/msg00001.html>

Definition at line 575 of file regex.h.

References std::move().

```
4.500.3.14 template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> basic_regex&
std::basic_regex< _Ch_type, _Rx_traits >::operator= ( const _Ch_type * __p ) [inline]
```

Replaces a regular expression with a new one constructed from a C-style null-terminated string.

Parameters

<code>__p</code>	A pointer to the start of a null-terminated C-style string containing a regular expression.
------------------	---

Definition at line 586 of file regex.h.

```
4.500.3.15 template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> basic_regex&
std::basic_regex< _Ch_type, _Rx_traits >::operator= ( initializer_list<_Ch_type> __l ) [inline]
```

Replaces a regular expression with a new one constructed from an initializer list.

Parameters

<code>__l</code>	The initializer list.
------------------	-----------------------

Exceptions

<code>regex_error</code>	if <code>__l</code> is not a valid regular expression.
--------------------------	--

Definition at line 598 of file regex.h.

```
4.500.3.16  template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> template<typename _Ch_typeraits ,
            typename _Alloc > basic_regex& std::basic_regex< _Ch_type, _Rx_traits >::operator= ( const basic_string<
            _Ch_type, _Ch_typeraits, _Alloc > & __s )  [inline]
```

Replaces a regular expression with a new one constructed from a string.

Parameters

<code>__s</code>	A pointer to a string containing a regular expression.
------------------	--

Definition at line 609 of file regex.h.

```
4.500.3.17  template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> void std::basic_regex< _Ch_type,
            _Rx_traits >::swap ( basic_regex< _Ch_type, _Rx_traits > & __rhs )  [inline]
```

Swaps the contents of two regular expression objects.

Parameters

<code>__rhs</code>	Another regular expression object.
--------------------	------------------------------------

Definition at line 787 of file regex.h.

References `std::basic_regex< _Ch_type, _Rx_traits >::assign()`, `std::move()`, and `std::swap()`.

Referenced by `std::swap()`.

The documentation for this class was generated from the following file:

- [regex.h](#)

4.501 `std::basic_string< _CharT, _Traits, _Alloc >` Class Template Reference

Public Types

- typedef `_Alloc` **allocator_type**
- typedef `__gnu_cxx::__normal_iterator< const_pointer, basic_string >` **const_iterator**
- typedef `_CharT_alloc_type::const_pointer` **const_pointer**
- typedef `_CharT_alloc_type::const_reference` **const_reference**
- typedef `std::reverse_iterator< const_iterator >` **const_reverse_iterator**
- typedef `_CharT_alloc_type::difference_type` **difference_type**
- typedef `__gnu_cxx::__normal_iterator< pointer, basic_string >` **iterator**
- typedef `_CharT_alloc_type::pointer` **pointer**
- typedef `_CharT_alloc_type::reference` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse_iterator**
- typedef `_CharT_alloc_type::size_type` **size_type**
- typedef `_Traits` **traits_type**
- typedef `_Traits::char_type` **value_type**

Public Member Functions

- [basic_string](#) ()
- [basic_string](#) (const _Alloc & __a)
- [basic_string](#) (const [basic_string](#) & __str)
- [basic_string](#) (const [basic_string](#) & __str, size_type __pos, size_type __n=npos)
- [basic_string](#) (const [basic_string](#) & __str, size_type __pos, size_type __n, const _Alloc & __a)
- [basic_string](#) (const _CharT * __s, size_type __n, const _Alloc & __a=_Alloc())
- [basic_string](#) (const _CharT * __s, const _Alloc & __a=_Alloc())
- [basic_string](#) (size_type __n, _CharT __c, const _Alloc & __a=_Alloc())
- [basic_string](#) ([basic_string](#) && __str) noexcept
- [basic_string](#) (initializer_list< _CharT > __l, const _Alloc & __a=_Alloc())
- template<class _InputIterator >
 [basic_string](#) (_InputIterator __beg, _InputIterator __end, const _Alloc & __a=_Alloc())
- ~[basic_string](#) () noexcept
- [basic_string](#) & [append](#) (const [basic_string](#) & __str)
- [basic_string](#) & [append](#) (const [basic_string](#) & __str, size_type __pos, size_type __n)
- [basic_string](#) & [append](#) (const _CharT * __s, size_type __n)
- [basic_string](#) & [append](#) (const _CharT * __s)
- [basic_string](#) & [append](#) (size_type __n, _CharT __c)
- [basic_string](#) & [append](#) (initializer_list< _CharT > __l)
- template<class _InputIterator >
 [basic_string](#) & [append](#) (_InputIterator __first, _InputIterator __last)
- [basic_string](#) & [assign](#) (const [basic_string](#) & __str)
- [basic_string](#) & [assign](#) ([basic_string](#) && __str)
- [basic_string](#) & [assign](#) (const [basic_string](#) & __str, size_type __pos, size_type __n)
- [basic_string](#) & [assign](#) (const _CharT * __s, size_type __n)
- [basic_string](#) & [assign](#) (const _CharT * __s)
- [basic_string](#) & [assign](#) (size_type __n, _CharT __c)
- template<class _InputIterator >
 [basic_string](#) & [assign](#) (_InputIterator __first, _InputIterator __last)
- [basic_string](#) & [assign](#) (initializer_list< _CharT > __l)
- const_reference [at](#) (size_type __n) const
- reference [at](#) (size_type __n)
- reference [back](#) ()
- const_reference [back](#) () const noexcept
- iterator [begin](#) ()
- const_iterator [begin](#) () const noexcept
- const _CharT * [c_str](#) () const noexcept
- size_type [capacity](#) () const noexcept
- const_iterator [cbegin](#) () const noexcept
- const_iterator [cend](#) () const noexcept
- void [clear](#) ()
- int [compare](#) (const [basic_string](#) & __str) const
- int [compare](#) (size_type __pos, size_type __n, const [basic_string](#) & __str) const
- int [compare](#) (size_type __pos1, size_type __n1, const [basic_string](#) & __str, size_type __pos2, size_type __n2) const
- int [compare](#) (const _CharT * __s) const
- int [compare](#) (size_type __pos, size_type __n1, const _CharT * __s) const
- int [compare](#) (size_type __pos, size_type __n1, const _CharT * __s, size_type __n2) const
- size_type [copy](#) (_CharT * __s, size_type __n, size_type __pos=0) const

- `const_reverse_iterator crbegin` () const noexcept
- `const_reverse_iterator crend` () const noexcept
- `const _CharT * data` () const noexcept
- `bool empty` () const noexcept
- `iterator end` ()
- `const_iterator end` () const noexcept
- `basic_string & erase` (size_type __pos=0, size_type __n=npos)
- `iterator erase` (iterator __position)
- `iterator erase` (iterator __first, iterator __last)
- `size_type find` (const _CharT *__s, size_type __pos, size_type __n) const
- `size_type find` (const basic_string & __str, size_type __pos=0) const noexcept
- `size_type find` (const _CharT *__s, size_type __pos=0) const
- `size_type find` (_CharT __c, size_type __pos=0) const noexcept
- `size_type find_first_not_of` (const basic_string & __str, size_type __pos=0) const noexcept
- `size_type find_first_not_of` (const _CharT *__s, size_type __pos, size_type __n) const
- `size_type find_first_not_of` (const _CharT *__s, size_type __pos=0) const
- `size_type find_first_not_of` (_CharT __c, size_type __pos=0) const noexcept
- `size_type find_first_of` (const basic_string & __str, size_type __pos=0) const noexcept
- `size_type find_first_of` (const _CharT *__s, size_type __pos, size_type __n) const
- `size_type find_first_of` (const _CharT *__s, size_type __pos=0) const
- `size_type find_first_of` (_CharT __c, size_type __pos=0) const noexcept
- `size_type find_last_not_of` (const basic_string & __str, size_type __pos=npos) const noexcept
- `size_type find_last_not_of` (const _CharT *__s, size_type __pos, size_type __n) const
- `size_type find_last_not_of` (const _CharT *__s, size_type __pos=npos) const
- `size_type find_last_not_of` (_CharT __c, size_type __pos=npos) const noexcept
- `size_type find_last_of` (const basic_string & __str, size_type __pos=npos) const noexcept
- `size_type find_last_of` (const _CharT *__s, size_type __pos, size_type __n) const
- `size_type find_last_of` (const _CharT *__s, size_type __pos=npos) const
- `size_type find_last_of` (_CharT __c, size_type __pos=npos) const noexcept
- `reference front` ()
- `const_reference front` () const noexcept
- `allocator_type get_allocator` () const noexcept
- `void insert` (iterator __p, size_type __n, _CharT __c)
- `template<class _InputIterator >`
`void insert` (iterator __p, _InputIterator __beg, _InputIterator __end)
- `void insert` (iterator __p, initializer_list< _CharT > __l)
- `basic_string & insert` (size_type __pos1, const basic_string & __str)
- `basic_string & insert` (size_type __pos1, const basic_string & __str, size_type __pos2, size_type __n)
- `basic_string & insert` (size_type __pos, const _CharT *__s, size_type __n)
- `basic_string & insert` (size_type __pos, const _CharT *__s)
- `basic_string & insert` (size_type __pos, size_type __n, _CharT __c)
- `iterator insert` (iterator __p, _CharT __c)
- `size_type length` () const noexcept
- `size_type max_size` () const noexcept
- `basic_string & operator+=` (const basic_string & __str)
- `basic_string & operator+=` (const _CharT *__s)
- `basic_string & operator+=` (_CharT __c)
- `basic_string & operator+=` (initializer_list< _CharT > __l)
- `basic_string & operator=` (const basic_string & __str)
- `basic_string & operator=` (const _CharT *__s)
- `basic_string & operator=` (_CharT __c)

- [basic_string](#) & [operator=](#) ([basic_string](#) && __str)
- [basic_string](#) & [operator=](#) (initializer_list< _CharT > __l)
- const_reference [operator\[\]](#) (size_type __pos) const noexcept
- reference [operator\[\]](#) (size_type __pos)
- void [pop_back](#) ()
- void [push_back](#) (_CharT __c)
- [reverse_iterator](#) [rbegin](#) ()
- const [reverse_iterator](#) [rbegin](#) () const noexcept
- [reverse_iterator](#) [rend](#) ()
- const [reverse_iterator](#) [rend](#) () const noexcept
- [basic_string](#) & [replace](#) (size_type __pos, size_type __n, const [basic_string](#) & __str)
- [basic_string](#) & [replace](#) (size_type __pos1, size_type __n1, const [basic_string](#) & __str, size_type __pos2, size_type __n2)
- [basic_string](#) & [replace](#) (size_type __pos, size_type __n1, const _CharT * __s, size_type __n2)
- [basic_string](#) & [replace](#) (size_type __pos, size_type __n1, const _CharT * __s)
- [basic_string](#) & [replace](#) (size_type __pos, size_type __n1, size_type __n2, _CharT __c)
- [basic_string](#) & [replace](#) (iterator __i1, iterator __i2, const [basic_string](#) & __str)
- [basic_string](#) & [replace](#) (iterator __i1, iterator __i2, const _CharT * __s, size_type __n)
- [basic_string](#) & [replace](#) (iterator __i1, iterator __i2, const _CharT * __s)
- [basic_string](#) & [replace](#) (iterator __i1, iterator __i2, size_type __n, _CharT __c)
- template<class _InputIterator >
[basic_string](#) & [replace](#) (iterator __i1, iterator __i2, _InputIterator __k1, _InputIterator __k2)
- [basic_string](#) & [replace](#) (iterator __i1, iterator __i2, _CharT * __k1, _CharT * __k2)
- [basic_string](#) & [replace](#) (iterator __i1, iterator __i2, const _CharT * __k1, const _CharT * __k2)
- [basic_string](#) & [replace](#) (iterator __i1, iterator __i2, iterator __k1, iterator __k2)
- [basic_string](#) & [replace](#) (iterator __i1, iterator __i2, const_iterator __k1, const_iterator __k2)
- [basic_string](#) & [replace](#) (iterator __i1, iterator __i2, initializer_list< _CharT > __l)
- void [reserve](#) (size_type __res_arg=0)
- void [resize](#) (size_type __n, _CharT __c)
- void [resize](#) (size_type __n)
- size_type [rfind](#) (const [basic_string](#) & __str, size_type __pos=[npos](#)) const noexcept
- size_type [rfind](#) (const _CharT * __s, size_type __pos, size_type __n) const
- size_type [rfind](#) (const _CharT * __s, size_type __pos=[npos](#)) const
- size_type [rfind](#) (_CharT __c, size_type __pos=[npos](#)) const noexcept
- void [shrink_to_fit](#) () noexcept
- size_type [size](#) () const noexcept
- [basic_string](#) substr (size_type __pos=0, size_type __n=[npos](#)) const
- void [swap](#) ([basic_string](#) & __s)

Static Public Attributes

- static const size_type [npos](#)

4.501.1 Detailed Description

```
template<typename _CharT, typename _Traits, typename _Alloc>
class std::basic_string< _CharT, _Traits, _Alloc >
```

Managing sequences of characters and character-like objects.

Template Parameters

<code>_CharT</code>	Type of character
<code>_Traits</code>	Traits for character type, defaults to <code>char_traits<_CharT></code> .
<code>_Alloc</code>	Allocator type, defaults to <code>allocator<_CharT></code> .

Meets the requirements of a [container](#), a [reversible container](#), and a [sequence](#). Of the [optional sequence requirements](#), only `push_back`, `at`, and `array` access are supported.

Todo Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html

Documentation? What's that? Nathan Myers ncm@cantrip.org.

A string looks like this:

```

[basic_string<char_type>]
_M_dataplus
_M_p ----->
                                     [_Rep]
                                     _M_length
                                     _M_capacity
                                     _M_refcount
                                     unnamed array of char_type
```

Where the `_M_p` points to the first character in the string, and you cast it to a pointer-to-`_Rep` and subtract 1 to get a pointer to the header.

This approach has the enormous advantage that a string object requires only one allocation. All the ugliness is confined within a single pair of inline functions, which each compile to a single *add* instruction: `_Rep::_M_data()`, and `string::_M_rep()`; and the allocation function which gets a block of raw bytes and with room enough and constructs a `_Rep` object at the front.

The reason you want `_M_data` pointing to the character array and not the `_Rep` is so that the debugger can see the string contents. (Probably we should add a non-inline member to get the `_Rep` for the debugger to use, so users can check the actual string length.)

Note that the `_Rep` object is a POD so that you can have a static *empty string* `_Rep` object already *constructed* before static constructors have run. The reference-count encoding is chosen so that a 0 indicates one reference, so you never try to destroy the empty-string `_Rep` object.

All but the last paragraph is considered pretty conventional for a C++ string implementation.

Definition at line 112 of file `basic_string.h`.

4.501.2 Constructor & Destructor Documentation

4.501.2.1 `template<typename _CharT, typename _Traits, typename _Alloc> std::basic_string<_CharT, _Traits, _Alloc>::basic_string() [inline]`

Default constructor creates an empty string.

Definition at line 441 of file `basic_string.h`.

Referenced by `std::basic_string<_Ch_type>::basic_string()`, and `std::basic_string<_Ch_type>::substr()`.

4.501.2.2 `template<typename _CharT, typename _Traits, typename _Alloc> std::basic_string<_CharT, _Traits, _Alloc>::basic_string (const _Alloc &__a) [explicit]`

Construct an empty string using allocator *a*.

4.501.2.3 `template<typename _CharT, typename _Traits, typename _Alloc> std::basic_string<_CharT, _Traits, _Alloc>::basic_string (const basic_string<_CharT, _Traits, _Alloc> &__str)`

Construct string with copy of value of *str*.

Parameters

<code>__str</code>	Source string.
--------------------	----------------

4.501.2.4 `template<typename _CharT, typename _Traits, typename _Alloc> std::basic_string< _CharT, _Traits, _Alloc >::basic_string (const basic_string< _CharT, _Traits, _Alloc > & __str, size_type __pos, size_type __n = npos)`

Construct string as copy of a substring.

Parameters

<code>__str</code>	Source string.
<code>__pos</code>	Index of first character to copy from.
<code>__n</code>	Number of characters to copy (default remainder).

4.501.2.5 `template<typename _CharT, typename _Traits, typename _Alloc> std::basic_string< _CharT, _Traits, _Alloc >::basic_string (const basic_string< _CharT, _Traits, _Alloc > & __str, size_type __pos, size_type __n, const _Alloc & __a)`

Construct string as copy of a substring.

Parameters

<code>__str</code>	Source string.
<code>__pos</code>	Index of first character to copy from.
<code>__n</code>	Number of characters to copy.
<code>__a</code>	Allocator to use.

4.501.2.6 `template<typename _CharT, typename _Traits, typename _Alloc> std::basic_string< _CharT, _Traits, _Alloc >::basic_string (const _CharT * __s, size_type __n, const _Alloc & __a = _Alloc())`

Construct string initialized by a character array.

Parameters

<code>__s</code>	Source character array.
<code>__n</code>	Number of characters to copy.
<code>__a</code>	Allocator to use (default is default allocator).

NB: `__s` must have at least `__n` characters, `'\0'` has no special meaning.

4.501.2.7 `template<typename _CharT, typename _Traits, typename _Alloc> std::basic_string<_CharT, _Traits, _Alloc>::basic_string (const _CharT * __s, const _Alloc & __a = _Alloc())`

Construct string as copy of a C string.

Parameters

<code>__s</code>	Source C string.
<code>__a</code>	Allocator to use (default is default allocator).

4.501.2.8 `template<typename _CharT, typename _Traits, typename _Alloc> std::basic_string<_CharT, _Traits, _Alloc>::basic_string (size_type __n, _CharT __c, const _Alloc & __a = _Alloc())`

Construct string as multiple characters.

Parameters

<code>__n</code>	Number of characters.
<code>__c</code>	Character to use.
<code>__a</code>	Allocator to use (default is default allocator).

4.501.2.9 `template<typename _CharT, typename _Traits, typename _Alloc> std::basic_string<_CharT, _Traits, _Alloc>::basic_string (basic_string<_CharT, _Traits, _Alloc> && __str) [inline], [noexcept]`

Move construct string.

Parameters

<code>__str</code>	Source string.
--------------------	----------------

The newly-created string contains the exact contents of `__str`. `__str` is a valid, but unspecified string.

Definition at line 511 of file `basic_string.h`.

4.501.2.10 `template<typename _CharT, typename _Traits, typename _Alloc> std::basic_string<_CharT, _Traits, _Alloc>::basic_string (initializer_list<_CharT> __l, const _Alloc & __a = _Alloc())`

Construct string from an initializer list.

Parameters

<code>_l</code>	<code>std::initializer_list</code> of characters.
<code>_a</code>	Allocator to use (default is default allocator).

4.501.2.11 `template<typename _CharT, typename _Traits, typename _Alloc> template<class _InputIterator >
std::basic_string<_CharT, _Traits, _Alloc>::basic_string (_InputIterator __beg, _InputIterator __end, const
_Alloc & __a = _Alloc())`

Construct string as copy of a range.

Parameters

<code>__beg</code>	Start of range.
<code>__end</code>	End of range.
<code>__a</code>	Allocator to use (default is default allocator).

4.501.2.12 `template<typename _CharT, typename _Traits, typename _Alloc> std::basic_string<_CharT, _Traits, _Alloc
>::~~basic_string () [inline], [noexcept]`

Destroy the string instance.

Definition at line 545 of file `basic_string.h`.

4.501.3 Member Function Documentation

4.501.3.1 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string<_CharT,
_Traits, _Alloc>::append (const basic_string<_CharT, _Traits, _Alloc> & __str)`

Append a string to this string.

Parameters

<code>__str</code>	The string to append.
--------------------	-----------------------

Returns

Reference to this string.

Referenced by `std::basic_string<_Ch_type>::append()`, `std::operator+()`, and `std::basic_string<_Ch_type>::operator+=()`.

4.501.3.2 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT, _Traits, _Alloc >::append (const basic_string< _CharT, _Traits, _Alloc > & __str, size_type __pos, size_type __n)`

Append a substring.

Parameters

<code>__str</code>	The string to append.
<code>__pos</code>	Index of the first character of <code>str</code> to append.
<code>__n</code>	The number of characters to append.

Returns

Reference to this string.

Exceptions

<code>std::out_of_range</code>	if <code>__pos</code> is not a valid index.
--------------------------------	---

This function appends `__n` characters from `__str` starting at `__pos` to this string. If `__n` is larger than the number of available characters in `__str`, the remainder of `__str` is appended.

4.501.3.3 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT, _Traits, _Alloc >::append (const _CharT * __s, size_type __n)`

Append a C substring.

Parameters

<code>__s</code>	The C string to append.
<code>__n</code>	The number of characters to append.

Returns

Reference to this string.

4.501.3.4 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT, _Traits, _Alloc >::append (const _CharT * __s) [inline]`

Append a C string.

Parameters

<code>__s</code>	The C string to append.
------------------	-------------------------

Returns

Reference to this string.

Definition at line 1022 of file `basic_string.h`.

4.501.3.5 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT, _Traits, _Alloc >::append (size_type __n, _CharT __c)`

Append multiple characters.

Parameters

<code>__n</code>	The number of characters to append.
<code>__c</code>	The character to use.

Returns

Reference to this string.

Appends `__n` copies of `__c` to this string.

4.501.3.6 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT, _Traits, _Alloc >::append (initializer_list< _CharT > __l) [inline]`

Append an `initializer_list` of characters.

Parameters

<code>__l</code>	The <code>initializer_list</code> of characters to append.
------------------	--

Returns

Reference to this string.

Definition at line 1046 of file `basic_string.h`.

4.501.3.7 `template<typename _CharT, typename _Traits, typename _Alloc> template<class _InputIterator > basic_string& std::basic_string< _CharT, _Traits, _Alloc >::append (_InputIterator __first, _InputIterator __last) [inline]`

Append a range of characters.

Parameters

<code>__first</code>	Iterator referencing the first character to append.
<code>__last</code>	Iterator marking the end of the range.

Returns

Reference to this string.

Appends characters in the range [`__first`,`__last`) to this string.

Definition at line 1060 of file `basic_string.h`.

4.501.3.8 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT, _Traits, _Alloc >::assign (const basic_string< _CharT, _Traits, _Alloc > & __str)`

Set value to contents of another string.

Parameters

<code>__str</code>	Source string to use.
--------------------	-----------------------

Returns

Reference to this string.

Referenced by `std::basic_string< _Ch_type >::assign()`, `std::basic_string< _Ch_type >::operator=()`, and `std::basic_string< _Ch_type >::push_back()`.

4.501.3.9 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT, _Traits, _Alloc >::assign (basic_string< _CharT, _Traits, _Alloc > && __str) [inline]`

Set value to contents of another string.

Parameters

<code>__str</code>	Source string to use.
--------------------	-----------------------

Returns

Reference to this string.

This function sets this string to the exact contents of `__str`. `__str` is a valid, but unspecified string.

Definition at line 1096 of file `basic_string.h`.

```
4.501.3.10 template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT,
    _Traits, _Alloc >::assign ( const basic_string< _CharT, _Traits, _Alloc > & __str, size_type __pos, size_type __n )
    [inline]
```

Set value to a substring of a string.

Parameters

<code>__str</code>	The string to use.
<code>__pos</code>	Index of the first character of str.
<code>__n</code>	Number of characters to use.

Returns

Reference to this string.

Exceptions

<code>std::out_of_range</code>	if <code>pos</code> is not a valid index.
--------------------------------	---

This function sets this string to the substring of `__str` consisting of `__n` characters at `__pos`. If `__n` is larger than the number of available characters in `__str`, the remainder of `__str` is used.

Definition at line 1117 of file `basic_string.h`.

```
4.501.3.11 template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT,
    _Traits, _Alloc >::assign ( const _CharT * __s, size_type __n )
```

Set value to a C substring.

Parameters

<code>__s</code>	The C string to use.
<code>__n</code>	Number of characters to use.

Returns

Reference to this string.

This function sets the value of this string to the first `__n` characters of `__s`. If `__n` is larger than the number of available characters in `__s`, the remainder of `__s` is used.

4.501.3.12 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT, _Traits, _Alloc >::assign (const _CharT * __s) [inline]`

Set value to contents of a C string.

Parameters

<code>__s</code>	The C string to use.
------------------	----------------------

Returns

Reference to this string.

This function sets the value of this string to the value of `__s`. The data is copied, so there is no dependence on `__s` once the function returns.

Definition at line 1145 of file `basic_string.h`.

4.501.3.13 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT, _Traits, _Alloc >::assign (size_type __n, _CharT __c) [inline]`

Set value to multiple characters.

Parameters

<code>__n</code>	Length of the resulting string.
<code>__c</code>	The character to use.

Returns

Reference to this string.

This function sets the value of this string to `__n` copies of character `__c`.

Definition at line 1161 of file `basic_string.h`.

4.501.3.14 `template<typename _CharT, typename _Traits, typename _Alloc> template<class _InputIterator > basic_string& std::basic_string< _CharT, _Traits, _Alloc >::assign (_InputIterator __first, _InputIterator __last) [inline]`

Set value to a range of characters.

Parameters

<code>__first</code>	Iterator referencing the first character to append.
<code>__last</code>	Iterator marking the end of the range.

Returns

Reference to this string.

Sets value of string to characters in the range [`__first`,`__last`).

Definition at line 1174 of file `basic_string.h`.

4.501.3.15 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT, _Traits, _Alloc >::assign (initializer_list< _CharT > __l) [inline]`

Set value to an `initializer_list` of characters.

Parameters

<code>__l</code>	The <code>initializer_list</code> of characters to assign.
<code>__l</code>	
<code>__l</code>	
<code>__l</code>	
<code>__l</code>	

Returns

Reference to this string.

Definition at line 1184 of file `basic_string.h`.

4.501.3.16 `template<typename _CharT, typename _Traits, typename _Alloc> const_reference std::basic_string< _CharT, _Traits, _Alloc >::at (size_type __n) const [inline]`

Provides access to the data contained in the string.

Parameters

<code>__n</code>	The index of the character to access.
<code>__n</code>	

Returns

Read-only (`const`) reference to the character.

Exceptions

<code>std::out_of_range</code>	If <code>n</code> is an invalid index.
--------------------------------	--

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws `out_of_range` if the check fails.

Definition at line 874 of file basic_string.h.

4.501.3.17 `template<typename _CharT, typename _Traits, typename _Alloc> reference std::basic_string<_CharT, _Traits, _Alloc>::at(size_type __n) [inline]`

Provides access to the data contained in the string.

Parameters

<code>__n</code>	The index of the character to access.
------------------	---------------------------------------

Returns

Read/write reference to the character.

Exceptions

<code>std::out_of_range</code>	If <i>n</i> is an invalid index.
--------------------------------	----------------------------------

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws `out_of_range` if the check fails. Success results in unsharing the string.

Definition at line 896 of file basic_string.h.

4.501.3.18 `template<typename _CharT, typename _Traits, typename _Alloc> reference std::basic_string<_CharT, _Traits, _Alloc>::back() [inline]`

Returns a read/write reference to the data at the last element of the string.

Definition at line 929 of file basic_string.h.

4.501.3.19 `template<typename _CharT, typename _Traits, typename _Alloc> const_reference std::basic_string<_CharT, _Traits, _Alloc>::back() const [inline], [noexcept]`

Returns a read-only (constant) reference to the data at the last element of the string.

Definition at line 937 of file basic_string.h.

4.501.3.20 `template<typename _CharT, typename _Traits, typename _Alloc> iterator std::basic_string<_CharT, _Traits, _Alloc>::begin() [inline]`

Returns a read/write iterator that points to the first character in the string. Unshares the string.

Definition at line 613 of file basic_string.h.

Referenced by `__gnu_profile::__report()`, `std::basic_string<_Ch_type>::crend()`, `std::regex_match()`, `std::regex_replace()`, `std::regex_search()`, and `std::basic_string<_Ch_type>::rend()`.

4.501.3.21 `template<typename _CharT, typename _Traits, typename _Alloc> const_iterator std::basic_string< _CharT, _Traits, _Alloc >::begin () const [inline], [noexcept]`

Returns a read-only (constant) iterator that points to the first character in the string.

Definition at line 624 of file `basic_string.h`.

4.501.3.22 `template<typename _CharT, typename _Traits, typename _Alloc> const _CharT* std::basic_string< _CharT, _Traits, _Alloc >::c_str () const [inline], [noexcept]`

Return const pointer to null-terminated contents.

This is a handle to internal data. Do not modify or dire things may happen.

Definition at line 1818 of file `basic_string.h`.

Referenced by `__gnu_profile::__report()`, `std::getline()`, `std::messages< _CharT >::messages()`, `std::operator==()`, `std::regex_replace()`, and `std::messages< _CharT >::~~messages()`.

4.501.3.23 `template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string< _CharT, _Traits, _Alloc >::capacity () const [inline], [noexcept]`

Returns the total number of characters that the string can hold before needing to allocate more memory.

Definition at line 784 of file `basic_string.h`.

Referenced by `std::operator+()`, `std::basic_string< _Ch_type >::push_back()`, and `std::basic_string< _Ch_type >::shrink_to_fit()`.

4.501.3.24 `template<typename _CharT, typename _Traits, typename _Alloc> const_iterator std::basic_string< _CharT, _Traits, _Alloc >::cbegin () const [inline], [noexcept]`

Returns a read-only (constant) iterator that points to the first character in the string.

Definition at line 688 of file `basic_string.h`.

4.501.3.25 `template<typename _CharT, typename _Traits, typename _Alloc> const_iterator std::basic_string< _CharT, _Traits, _Alloc >::cend () const [inline], [noexcept]`

Returns a read-only (constant) iterator that points one past the last character in the string.

Definition at line 696 of file `basic_string.h`.

4.501.3.26 `template<typename _CharT, typename _Traits, typename _Alloc> void std::basic_string< _CharT, _Traits, _Alloc >::clear () [inline]`

Erases the string, making it empty.

Definition at line 812 of file `basic_string.h`.

4.501.3.27 `template<typename _CharT, typename _Traits, typename _Alloc> int std::basic_string< _CharT, _Traits, _Alloc >::compare (const basic_string< _CharT, _Traits, _Alloc > & __str) const [inline]`

Compare to a string.

Parameters

<code>__str</code>	String to compare against.
--------------------	----------------------------

Returns

Integer < 0, 0, or > 0.

Returns an integer < 0 if this string is ordered before `__str`, 0 if their values are equivalent, or > 0 if this string is ordered after `__str`. Determines the effective length `rlen` of the strings to compare as the smallest of `size()` and `str.size()`. The function then compares the two strings by calling `traits::compare(data(), str.data(), rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 2243 of file `basic_string.h`.

Referenced by `std::basic_string<_Ch_type>::compare()`, `std::operator<()`, `std::operator<=()`, `std::operator==()`, `std::operator>()`, and `std::operator>=()`.

4.501.3.28 `template<typename _CharT, typename _Traits, typename _Alloc> int std::basic_string<_CharT, _Traits, _Alloc>::compare (size_type __pos, size_type __n, const basic_string<_CharT, _Traits, _Alloc> & __str) const`

Compare substring to a string.

Parameters

<code>__pos</code>	Index of first character of substring.
<code>__n</code>	Number of characters in substring.
<code>__str</code>	String to compare against.

Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the `__n` characters starting at `__pos`. Returns an integer < 0 if the substring is ordered before `__str`, 0 if their values are equivalent, or > 0 if the substring is ordered after `__str`. Determines the effective length `rlen` of the strings to compare as the smallest of the length of the substring and `__str.size()`. The function then compares the two strings by calling `traits::compare(substring.data(), str.data(), rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

4.501.3.29 `template<typename _CharT, typename _Traits, typename _Alloc> int std::basic_string<_CharT, _Traits, _Alloc>::compare (size_type __pos1, size_type __n1, const basic_string<_CharT, _Traits, _Alloc> & __str, size_type __pos2, size_type __n2) const`

Compare substring to a substring.

Parameters

<code>__pos1</code>	Index of first character of substring.
---------------------	--

Parameters

<code>__n1</code>	Number of characters in substring.
<code>__str</code>	String to compare against.
<code>__pos2</code>	Index of first character of substring of <code>str</code> .
<code>__n2</code>	Number of characters in substring of <code>str</code> .

Returns

Integer < 0 , 0 , or > 0 .

Form the substring of this string from the `__n1` characters starting at `__pos1`. Form the substring of `__str` from the `__n2` characters starting at `__pos2`. Returns an integer < 0 if this substring is ordered before the substring of `__str`, 0 if their values are equivalent, or > 0 if this substring is ordered after the substring of `__str`. Determines the effective length `rlen` of the strings to compare as the smallest of the lengths of the substrings. The function then compares the two strings by calling `traits::compare(substring.data(),str.substr(pos2,n2).data(),rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

4.501.3.30 `template<typename _CharT, typename _Traits, typename _Alloc> int std::basic_string< _CharT, _Traits, _Alloc >::compare (const _CharT * __s) const`

Compare to a C string.

Parameters

<code>__s</code>	C string to compare against.
------------------	------------------------------

Returns

Integer < 0 , 0 , or > 0 .

Returns an integer < 0 if this string is ordered before `__s`, 0 if their values are equivalent, or > 0 if this string is ordered after `__s`. Determines the effective length `rlen` of the strings to compare as the smallest of `size()` and the length of a string constructed from `__s`. The function then compares the two strings by calling `traits::compare(data(),s,rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

4.501.3.31 `template<typename _CharT, typename _Traits, typename _Alloc> int std::basic_string< _CharT, _Traits, _Alloc >::compare (size_type __pos, size_type __n1, const _CharT * __s) const`

Compare substring to a C string.

Parameters

<code>__pos</code>	Index of first character of substring.
<code>__n1</code>	Number of characters in substring.
<code>__s</code>	C string to compare against.

Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the `__n1` characters starting at `pos`. Returns an integer < 0 if the substring is ordered before `__s`, 0 if their values are equivalent, or > 0 if the substring is ordered after `__s`. Determines the effective length `rlen` of the strings to compare as the smallest of the length of the substring and the length of a string constructed from `__s`. The function then compares the two string by calling `traits::compare(substring.data(),__s,rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

4.501.3.32 `template<typename _CharT, typename _Traits, typename _Alloc> int std::basic_string< _CharT, _Traits, _Alloc >::compare (size_type __pos, size_type __n1, const _CharT * __s, size_type __n2) const`

Compare substring against a character array.

Parameters

<code>__pos</code>	Index of first character of substring.
<code>__n1</code>	Number of characters in substring.
<code>__s</code>	character array to compare against.
<code>__n2</code>	Number of characters of <code>s</code> .

Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the `__n1` characters starting at `__pos`. Form a string from the first `__n2` characters of `__s`. Returns an integer < 0 if this substring is ordered before the string from `__s`, 0 if their values are equivalent, or > 0 if this substring is ordered after the string from `__s`. Determines the effective length `rlen` of the strings to compare as the smallest of the length of the substring and `__n2`. The function then compares the two strings by calling `traits::compare(substring.data(),s,rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

NB: `s` must have at least `n2` characters, `'\0'` has no special meaning.

4.501.3.33 `template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string< _CharT, _Traits, _Alloc >::copy (_CharT * __s, size_type __n, size_type __pos = 0) const`

Copy substring into C string.

Parameters

<code>__s</code>	C string to copy value into.
<code>__n</code>	Number of characters to copy.
<code>__pos</code>	Index of first character to copy.

Returns

Number of characters actually copied

Exceptions

<code>std::out_of_range</code>	If <code>__pos > size()</code> .
--------------------------------	-------------------------------------

Copies up to `__n` characters starting at `__pos` into the C string `__s`. If `__pos` is greater than `size()`, `out_of_range` is thrown.

Referenced by `std::basic_string<_Ch_type>::replace()`.

4.501.3.34 `template<typename _CharT, typename _Traits, typename _Alloc> const_reverse_iterator std::basic_string<_CharT, _Traits, _Alloc>::crbegin() const [inline], [noexcept]`

Returns a read-only (constant) reverse iterator that points to the last character in the string. Iteration is done in reverse element order.

Definition at line 705 of file `basic_string.h`.

4.501.3.35 `template<typename _CharT, typename _Traits, typename _Alloc> const_reverse_iterator std::basic_string<_CharT, _Traits, _Alloc>::crend() const [inline], [noexcept]`

Returns a read-only (constant) reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order.

Definition at line 714 of file `basic_string.h`.

4.501.3.36 `template<typename _CharT, typename _Traits, typename _Alloc> const _CharT* std::basic_string<_CharT, _Traits, _Alloc>::data() const [inline], [noexcept]`

Return const pointer to contents.

This is a handle to internal data. Do not modify or dire things may happen.

Definition at line 1828 of file `basic_string.h`.

Referenced by `std::basic_regex<_Ch_type, _Rx_traits>::assign()`, `std::basic_string<_Ch_type>::compare()`, `std::basic_string<_Ch_type>::find()`, `std::basic_string<_Ch_type>::find_first_not_of()`, `std::basic_string<_Ch_type>::find_first_of()`, `std::basic_string<_Ch_type>::find_last_not_of()`, `std::basic_string<_Ch_type>::find_last_of()`, `std::match_results<_Bi_iter>::format()`, `std::operator<<()`, `std::operator==()`, `std::basic_string<_Ch_type>::rfind()`, and `std::regex_traits<_Ch_type>::transform()`.

4.501.3.37 `template<typename _CharT, typename _Traits, typename _Alloc> bool std::basic_string<_CharT, _Traits, _Alloc>::empty() const [inline], [noexcept]`

Returns true if the string is empty. Equivalent to `*this == ""`.

Definition at line 820 of file `basic_string.h`.

4.501.338 `template<typename _CharT, typename _Traits, typename _Alloc> iterator std::basic_string< _CharT, _Traits, _Alloc >::end() [inline]`

Returns a read/write iterator that points one past the last character in the string. Unshares the string.

Definition at line 632 of file `basic_string.h`.

Referenced by `__gnu_profile::__report()`, `std::basic_string< _Ch_type >::crbegin()`, `std::basic_string< _Ch_type >::begin()`, `std::regex_match()`, `std::regex_replace()`, and `std::regex_search()`.

4.501.339 `template<typename _CharT, typename _Traits, typename _Alloc> const_iterator std::basic_string< _CharT, _Traits, _Alloc >::end() const [inline], [noexcept]`

Returns a read-only (constant) iterator that points one past the last character in the string.

Definition at line 643 of file `basic_string.h`.

4.501.340 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT, _Traits, _Alloc >::erase (size_type __pos = 0, size_type __n = npos) [inline]`

Remove characters.

Parameters

<code>__pos</code>	Index of first character to remove (default 0).
<code>__n</code>	Number of characters to remove (default remainder).

Returns

Reference to this string.

Exceptions

<code>std::out_of_range</code>	If <code>pos</code> is beyond the end of this string.
--------------------------------	---

Removes `__n` characters from this string starting at `__pos`. The length of the string is reduced by `__n`. If there are `< __n` characters to remove, the remainder of the string is truncated. If `__p` is beyond end of string, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1379 of file `basic_string.h`.

Referenced by `__gnu_profile::__report()`, `std::basic_string< _Ch_type >::erase()`, and `std::basic_string< _Ch_type >::pop_back()`.

4.501.341 `template<typename _CharT, typename _Traits, typename _Alloc> iterator std::basic_string< _CharT, _Traits, _Alloc >::erase (iterator __position) [inline]`

Remove one character.

Parameters

<code>__position</code>	Iterator referencing the character to remove.
-------------------------	---

Returns

iterator referencing same location after removal.

Removes the character at `__position` from this string. The value of the string doesn't change if an error is thrown.

Definition at line 1395 of file `basic_string.h`.

4.501.3.42 `template<typename _CharT, typename _Traits, typename _Alloc> iterator std::basic_string< _CharT, _Traits, _Alloc >::erase (iterator __first, iterator __last)`

Remove a range of characters.

Parameters

<code>__first</code>	Iterator referencing the first character to remove.
<code>__last</code>	Iterator referencing the end of the range.

Returns

Iterator referencing location of first after removal.

Removes the characters in the range `[first,last)` from this string. The value of the string doesn't change if an error is thrown.

4.501.3.43 `template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string< _CharT, _Traits, _Alloc >::find (const _CharT * __s, size_type __pos, size_type __n) const`

Find position of a C substring.

Parameters

<code>__s</code>	C string to locate.
<code>__pos</code>	Index of character to search from.
<code>__n</code>	Number of characters from <code>s</code> to search for.

Returns

Index of start of first occurrence.

Starting from `__pos`, searches forward for the first `__n` characters in `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Referenced by `__gnu_profile::__report()`, `std::basic_string< _Ch_type >::find()`, `std::basic_string< _Ch_type >::find←_first_of()`, and `std::basic_string< _Ch_type >::get_allocator()`.

4.501.3.44 `template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string< _CharT, _Traits, _Alloc >::find (const basic_string< _CharT, _Traits, _Alloc > & __str, size_type __pos = 0) const [inline], [noexcept]`

Find position of a string.

Parameters

<code>__str</code>	String to locate.
<code>__pos</code>	Index of character to search from (default 0).

Returns

Index of start of first occurrence.

Starting from `__pos`, searches forward for value of `__str` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 1864 of file `basic_string.h`.

4.501.3.45 `template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string< _CharT, _Traits, _Alloc >::find (const _CharT * __s, size_type __pos = 0) const [inline]`

Find position of a C string.

Parameters

<code>__s</code>	C string to locate.
<code>__pos</code>	Index of character to search from (default 0).

Returns

Index of start of first occurrence.

Starting from `__pos`, searches forward for the value of `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 1879 of file `basic_string.h`.

4.501.3.46 `template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string< _CharT, _Traits, _Alloc >::find (_CharT __c, size_type __pos = 0) const [noexcept]`

Find position of a character.

Parameters

<code>__c</code>	Character to locate.
<code>__pos</code>	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from `__pos`, searches forward for `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

4.501.3.47 `template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_not_of (const basic_string< _CharT, _Traits, _Alloc > & __str, size_type __pos = 0) const`
`[inline], [noexcept]`

Find position of a character not in string.

Parameters

<code>__str</code>	String containing characters to avoid.
<code>__pos</code>	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from `__pos`, searches forward for a character not contained in `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 2097 of file `basic_string.h`.

Referenced by `__gnu_profile::__report()`, and `std::basic_string< _Ch_type >::find_first_not_of()`.

4.501.3.48 `template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_not_of (const _CharT * __s, size_type __pos, size_type __n) const`

Find position of a character not in C substring.

Parameters

<code>__s</code>	C string containing characters to avoid.
<code>__pos</code>	Index of character to search from.
<code>__n</code>	Number of characters from <code>__s</code> to consider.

Returns

Index of first occurrence.

Starting from `__pos`, searches forward for a character not contained in the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

4.501.3.49 `template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_not_of (const _CharT* __s, size_type __pos = 0) const [inline]`

Find position of a character not in C string.

Parameters

<code>__s</code>	C string containing characters to avoid.
<code>__pos</code>	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from `__pos`, searches forward for a character not contained in `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 2128 of file `basic_string.h`.

4.501.3.50 `template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_not_of (_CharT __c, size_type __pos = 0) const [noexcept]`

Find position of a different character.

Parameters

<code>__c</code>	Character to avoid.
<code>__pos</code>	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from `__pos`, searches forward for a character other than `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

4.501.3.51 `template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_of (const basic_string< _CharT, _Traits, _Alloc > & __str, size_type __pos = 0) const [inline], [noexcept]`

Find position of a character of string.

Parameters

<code>__str</code>	String containing characters to locate.
<code>__pos</code>	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from `__pos`, searches forward for one of the characters of `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1970 of file `basic_string.h`.

Referenced by `__gnu_profile::__report()`, and `std::basic_string<_Ch_type>::find_first_of()`.

4.501.3.52 `template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string<_CharT, _Traits, _Alloc>::find_first_of(const _CharT * __s, size_type __pos, size_type __n) const`

Find position of a character of C substring.

Parameters

<code>__s</code>	String containing characters to locate.
<code>__pos</code>	Index of character to search from.
<code>__n</code>	Number of characters from <code>s</code> to search for.

Returns

Index of first occurrence.

Starting from `__pos`, searches forward for one of the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

4.501.3.53 `template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string<_CharT, _Traits, _Alloc>::find_first_of(const _CharT * __s, size_type __pos = 0) const [inline]`

Find position of a character of C string.

Parameters

<code>__s</code>	String containing characters to locate.
<code>__pos</code>	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from `__pos`, searches forward for one of the characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 2000 of file `basic_string.h`.

```
4.501.3.54 template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string< _CharT, _Traits,
    _Alloc >::find_first_of( _CharT __c, size_type __pos = 0 ) const [inline], [noexcept]
```

Find position of a character.

Parameters

<code>__c</code>	Character to locate.
<code>__pos</code>	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from `__pos`, searches forward for the character `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Note: equivalent to `find(__c, __pos)`.

Definition at line 2019 of file `basic_string.h`.

```
4.501.3.55 template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string< _CharT, _Traits,
    _Alloc >::find_last_not_of( const basic_string< _CharT, _Traits, _Alloc > & __str, size_type __pos = npos ) const
    [inline], [noexcept]
```

Find last position of a character not in string.

Parameters

<code>__str</code>	String containing characters to avoid.
<code>__pos</code>	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from `__pos`, searches backward for a character not contained in `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 2160 of file `basic_string.h`.

Referenced by `std::basic_string< _Ch_type >::find_last_not_of()`.

4.501.3.56 `template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_not_of (const _CharT * __s, size_type __pos, size_type __n) const`

Find last position of a character not in C substring.

Parameters

<code>__s</code>	C string containing characters to avoid.
<code>__pos</code>	Index of character to search back from.
<code>__n</code>	Number of characters from s to consider.

Returns

Index of last occurrence.

Starting from `__pos`, searches backward for a character not contained in the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

4.501.3.57 `template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_not_of (const _CharT * __s, size_type __pos = npos) const [inline]`

Find last position of a character not in C string.

Parameters

<code>__s</code>	C string containing characters to avoid.
<code>__pos</code>	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from `__pos`, searches backward for a character not contained in `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 2191 of file `basic_string.h`.

4.501.3.58 `template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_not_of (_CharT __c, size_type __pos = npos) const [noexcept]`

Find last position of a different character.

Parameters

<code>__c</code>	Character to avoid.
<code>__pos</code>	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from `__pos`, searches backward for a character other than `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

4.501.3.59 `template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string<_CharT, _Traits, _Alloc>::find_last_of(const basic_string<_CharT, _Traits, _Alloc> & __str, size_type __pos = npos) const`
`[inline], [noexcept]`

Find last position of a character of string.

Parameters

<code>__str</code>	String containing characters to locate.
<code>__pos</code>	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from `__pos`, searches backward for one of the characters of `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 2034 of file `basic_string.h`.

Referenced by `std::basic_string<_Ch_type>::find_last_of()`.

4.501.3.60 `template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string<_CharT, _Traits, _Alloc>::find_last_of(const _CharT* __s, size_type __pos, size_type __n) const`

Find last position of a character of C substring.

Parameters

<code>__s</code>	C string containing characters to locate.
<code>__pos</code>	Index of character to search back from.
<code>__n</code>	Number of characters from <code>s</code> to search for.

Returns

Index of last occurrence.

Starting from `__pos`, searches backward for one of the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

4.501.3.61 `template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_of(const _CharT* __s, size_type __pos = npos) const [inline]`

Find last position of a character of C string.

Parameters

<code>__s</code>	C string containing characters to locate.
<code>__pos</code>	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from `__pos`, searches backward for one of the characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 2064 of file `basic_string.h`.

4.501.3.62 `template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_of(_CharT __c, size_type __pos = npos) const [inline], [noexcept]`

Find last position of a character.

Parameters

<code>__c</code>	Character to locate.
<code>__pos</code>	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from `__pos`, searches backward for `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Note: equivalent to `rfind(__c, __pos)`.

Definition at line 2083 of file `basic_string.h`.

4.501.3.63 `template<typename _CharT, typename _Traits, typename _Alloc> reference std::basic_string< _CharT, _Traits, _Alloc >::front() [inline]`

Returns a read/write reference to the data at the first element of the string.

Definition at line 913 of file `basic_string.h`.

4.501.3.64 `template<typename _CharT, typename _Traits, typename _Alloc> const_reference std::basic_string< _CharT, _Traits, _Alloc >::front () const [inline], [noexcept]`

Returns a read-only (constant) reference to the data at the first element of the string.

Definition at line 921 of file basic_string.h.

4.501.3.65 `template<typename _CharT, typename _Traits, typename _Alloc> allocator_type std::basic_string< _CharT, _Traits, _Alloc >::get_allocator () const [inline], [noexcept]`

Return copy of allocator used to construct this string.

Definition at line 1835 of file basic_string.h.

Referenced by `std::basic_string< _Ch_type >::basic_string()`, and `std::basic_string< _Ch_type >::~~basic_string()`.

4.501.3.66 `template<typename _CharT, typename _Traits, typename _Alloc> void std::basic_string< _CharT, _Traits, _Alloc >::insert (iterator __p, size_type __n, _CharT __c) [inline]`

Insert multiple characters.

Parameters

<code>__p</code>	Iterator referencing location in string to insert at.
<code>__n</code>	Number of characters to insert
<code>__c</code>	The character to insert.

Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Inserts `__n` copies of character `__c` starting at the position referenced by iterator `__p`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1202 of file basic_string.h.

Referenced by `std::basic_string< _Ch_type >::insert()`, and `std::operator+()`.

4.501.3.67 `template<typename _CharT, typename _Traits, typename _Alloc> template<class _InputIterator > void std::basic_string< _CharT, _Traits, _Alloc >::insert (iterator __p, _InputIterator __beg, _InputIterator __end) [inline]`

Insert a range of characters.

Parameters

<code>__p</code>	Iterator referencing location in string to insert at.
<code>__beg</code>	Start of range.
<code>__end</code>	End of range.

Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Inserts characters in range `[__beg, __end)`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1219 of file `basic_string.h`.

```
4.501.3.68  template<typename _CharT, typename _Traits, typename _Alloc> void std::basic_string<_CharT, _Traits, _Alloc>::insert( iterator __p, initializer_list<_CharT> __l ) [inline]
```

Insert an `initializer_list` of characters.

Parameters

<code>__p</code>	Iterator referencing location in string to insert at.
<code>__l</code>	The <code>initializer_list</code> of characters to insert.

Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Definition at line 1230 of file `basic_string.h`.

```
4.501.3.69  template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string<_CharT, _Traits, _Alloc>::insert( size_type __pos1, const basic_string<_CharT, _Traits, _Alloc> & __str ) [inline]
```

Insert value of a string.

Parameters

<code>__pos1</code>	Iterator referencing location in string to insert at.
<code>__str</code>	The string to insert.

Returns

Reference to this string.

Exceptions

<i>std::length_error</i>	If new length exceeds <code>max_size()</code> .
--------------------------	---

Inserts value of `__str` starting at `__pos1`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1250 of file `basic_string.h`.

4.501.3.70 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string<_CharT, _Traits, _Alloc>::insert(size_type __pos1, const basic_string<_CharT, _Traits, _Alloc> & __str, size_type __pos2, size_type __n) [inline]`

Insert a substring.

Parameters

<code>__pos1</code>	Iterator referencing location in string to insert at.
<code>__str</code>	The string to insert.
<code>__pos2</code>	Start of characters in <code>str</code> to insert.
<code>__n</code>	Number of characters to insert.

Returns

Reference to this string.

Exceptions

<i>std::length_error</i>	If new length exceeds <code>max_size()</code> .
<i>std::out_of_range</i>	If <code>pos1 > size()</code> or <code>__pos2 > str.size()</code> .

Starting at `pos1`, insert `__n` character of `__str` beginning with `__pos2`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos1` is beyond the end of this string or `__pos2` is beyond the end of `__str`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1272 of file `basic_string.h`.

4.501.3.71 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string<_CharT, _Traits, _Alloc>::insert(size_type __pos, const _CharT* __s, size_type __n)`

Insert a C substring.

Parameters

<code>__pos</code>	Iterator referencing location in string to insert at.
<code>__s</code>	The C string to insert.
<code>__n</code>	The number of characters to insert.

Returns

Reference to this string.

Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
<code>std::out_of_range</code>	If <code>__pos</code> is beyond the end of this string.

Inserts the first `__n` characters of `__s` starting at `__pos`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos` is beyond `end()`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

4.501.3.72 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string<_CharT, _Traits, _Alloc>::insert (size_type __pos, const _CharT* __s) [inline]`

Insert a C string.

Parameters

<code>__pos</code>	Iterator referencing location in string to insert at.
<code>__s</code>	The C string to insert.

Returns

Reference to this string.

Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
<code>std::out_of_range</code>	If <code>pos</code> is beyond the end of this string.

Inserts the first `n` characters of `__s` starting at `__pos`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos` is beyond `end()`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1313 of file `basic_string.h`.

4.501.3.73 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string<_CharT, _Traits, _Alloc>::insert (size_type __pos, size_type __n, _CharT __c) [inline]`

Insert multiple characters.

Parameters

<code>__pos</code>	Index in string to insert at.
<code>__n</code>	Number of characters to insert
<code>__c</code>	The character to insert.

Returns

Reference to this string.

Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
<code>std::out_of_range</code>	If <code>__pos</code> is beyond the end of this string.

Inserts `__n` copies of character `__c` starting at index `__pos`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos > length()`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1336 of file `basic_string.h`.

4.501.3.74 `template<typename _CharT, typename _Traits, typename _Alloc> iterator std::basic_string<_CharT, _Traits, _Alloc>::insert (iterator __p, _CharT __c) [inline]`

Insert one character.

Parameters

<code>__p</code>	Iterator referencing position in string to insert at.
<code>__c</code>	The character to insert.

Returns

Iterator referencing newly inserted char.

Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Inserts character `__c` at position referenced by `__p`. If adding character causes the length to exceed `max_size()`, `length_error` is thrown. If `__p` is beyond end of string, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1354 of file `basic_string.h`.

```
4.501.3.75  template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string< _CharT, _Traits,
            _Alloc >::length ( ) const    [inline], [noexcept]
```

Returns the number of characters in the string, not including any null-termination.

Definition at line 729 of file `basic_string.h`.

Referenced by `__gnu_profile::__report()`.

```
4.501.3.76  template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string< _CharT, _Traits,
            _Alloc >::max_size ( ) const    [inline], [noexcept]
```

Returns the `size()` of the largest possible string.

Definition at line 734 of file `basic_string.h`.

```
4.501.3.77  template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT,
            _Traits, _Alloc >::operator+=( const basic_string< _CharT, _Traits, _Alloc > & __str )    [inline]
```

Append a string to this string.

Parameters

<code>__str</code>	The string to append.
--------------------	-----------------------

Returns

Reference to this string.

Definition at line 948 of file `basic_string.h`.

```
4.501.3.78  template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT,
            _Traits, _Alloc >::operator+=( const _CharT * __s )    [inline]
```

Append a C string.

Parameters

<code>__s</code>	The C string to append.
------------------	-------------------------

Returns

Reference to this string.

Definition at line 957 of file basic_string.h.

```
4.501.3.79 template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT,
    _Traits, _Alloc >::operator+=( _CharT __c ) [inline]
```

Append a character.

Parameters

<code>__c</code>	The character to append.
-------------------------	--------------------------

Returns

Reference to this string.

Definition at line 966 of file basic_string.h.

```
4.501.3.80 template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT,
    _Traits, _Alloc >::operator+=( initializer_list< _CharT > __l ) [inline]
```

Append an initializer_list of characters.

Parameters

<code>__l</code>	The initializer_list of characters to be appended.
-------------------------	--

Returns

Reference to this string.

Definition at line 979 of file basic_string.h.

```
4.501.3.81 template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT,
    _Traits, _Alloc >::operator=( const basic_string< _CharT, _Traits, _Alloc > & __str ) [inline]
```

Assign the value of *str* to this string.

Parameters

<code>__str</code>	Source string.
--------------------	----------------

Definition at line 553 of file `basic_string.h`.

4.501.3.82 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT, _Traits, _Alloc >::operator= (const _CharT * __s) [inline]`

Copy contents of `s` into this string.

Parameters

<code>__s</code>	Source null-terminated string.
------------------	--------------------------------

Definition at line 561 of file `basic_string.h`.

4.501.3.83 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT, _Traits, _Alloc >::operator= (_CharT __c) [inline]`

Set value to string of length 1.

Parameters

<code>__c</code>	Source character.
------------------	-------------------

Assigning to a character makes this string length 1 and `(*this)[0] == c`.

Definition at line 572 of file `basic_string.h`.

4.501.3.84 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT, _Traits, _Alloc >::operator= (basic_string< _CharT, _Traits, _Alloc > && __str) [inline]`

Move assign the value of `str` to this string.

Parameters

<code>__str</code>	Source string.
--------------------	----------------

The contents of `str` are moved into this string (without copying). `str` is a valid, but unspecified string.

Definition at line 588 of file `basic_string.h`.

4.501.3.85 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT, _Traits, _Alloc >::operator=(initializer_list< _CharT > __l) [inline]`

Set value to string constructed from initializer list.

Parameters

↩	std::initializer_list.
↩	
↩	
↩	
/	

Definition at line 600 of file basic_string.h.

4.501.3.86 `template<typename _CharT, typename _Traits, typename _Alloc> const_reference std::basic_string< _CharT, _Traits, _Alloc >::operator[](size_type __pos) const [inline], [noexcept]`

Subscript access to the data contained in the string.

Parameters

__pos	The index of the character to access.
-------	---------------------------------------

Returns

Read-only (constant) reference to the character.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and out_of_range lookups are not defined. (For checked lookups see at().)

Definition at line 835 of file basic_string.h.

Referenced by std::basic_string< _Ch_type >::back(), and std::basic_string< _Ch_type >::front().

4.501.3.87 `template<typename _CharT, typename _Traits, typename _Alloc> reference std::basic_string< _CharT, _Traits, _Alloc >::operator[](size_type __pos) [inline]`

Subscript access to the data contained in the string.

Parameters

__pos	The index of the character to access.
-------	---------------------------------------

Returns

Read/write reference to the character.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and `out_of_range` lookups are not defined. (For checked lookups see `at()`.) Unshares the string.

Definition at line 852 of file `basic_string.h`.

```
4.501.3.88  template<typename _CharT, typename _Traits, typename _Alloc> void std::basic_string< _CharT, _Traits, _Alloc
>::pop_back( ) [inline]
```

Remove the last character.

The string must be non-empty.

Definition at line 1424 of file `basic_string.h`.

```
4.501.3.89  template<typename _CharT, typename _Traits, typename _Alloc> void std::basic_string< _CharT, _Traits, _Alloc
>::push_back( _CharT __c ) [inline]
```

Append a single character.

Parameters

<code>__c</code>	Character to append.
------------------	----------------------

Definition at line 1068 of file `basic_string.h`.

Referenced by `std::basic_string< _Ch_type >::operator+=()`.

```
4.501.3.90  template<typename _CharT, typename _Traits, typename _Alloc> reverse_iterator std::basic_string< _CharT,
_Traits, _Alloc >::rbegin( ) [inline]
```

Returns a read/write reverse iterator that points to the last character in the string. Iteration is done in reverse element order. Unshares the string.

Definition at line 652 of file `basic_string.h`.

```
4.501.3.91  template<typename _CharT, typename _Traits, typename _Alloc> const_reverse_iterator std::basic_string<
_CharT, _Traits, _Alloc >::rbegin( ) const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last character in the string. Iteration is done in reverse element order.

Definition at line 661 of file `basic_string.h`.

```
4.501.3.92  template<typename _CharT, typename _Traits, typename _Alloc> reverse_iterator std::basic_string< _CharT,
_Traits, _Alloc >::rend( ) [inline]
```

Returns a read/write reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order. Unshares the string.

Definition at line 670 of file `basic_string.h`.

4.501.3.93 `template<typename _CharT, typename _Traits, typename _Alloc> const_reverse_iterator std::basic_string< _CharT, _Traits, _Alloc >::rend() const [inline], [noexcept]`

Returns a read-only (constant) reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order.

Definition at line 679 of file basic_string.h.

4.501.3.94 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT, _Traits, _Alloc >::replace(size_type __pos, size_type __n, const basic_string< _CharT, _Traits, _Alloc > & __str) [inline]`

Replace characters with value from another string.

Parameters

<code>__pos</code>	Index of first character to replace.
<code>__n</code>	Number of characters to be replaced.
<code>__str</code>	String to insert.

Returns

Reference to this string.

Exceptions

<code>std::out_of_range</code>	If <code>pos</code> is beyond the end of this string.
<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .

Removes the characters in the range `[__pos, __pos+__n)` from this string. In place, the value of `__str` is inserted. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of the result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1446 of file basic_string.h.

Referenced by `std::basic_string< _Ch_type >::append()`, `std::basic_string< _Ch_type >::assign()`, `std::basic_string< _Ch_type >::insert()`, and `std::basic_string< _Ch_type >::replace()`.

4.501.3.95 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT, _Traits, _Alloc >::replace(size_type __pos1, size_type __n1, const basic_string< _CharT, _Traits, _Alloc > & __str, size_type __pos2, size_type __n2) [inline]`

Replace characters with value from another string.

Parameters

<code>__pos1</code>	Index of first character to replace.
<code>__n1</code>	Number of characters to be replaced.
<code>__str</code>	String to insert.
<code>__pos2</code>	Index of first character of <code>str</code> to use.
<code>__n2</code>	Number of characters from <code>str</code> to use.

Returns

Reference to this string.

Exceptions

<i>std::out_of_range</i>	If <code>__pos1 > size()</code> or <code>__pos2 > __str.size()</code> .
<i>std::length_error</i>	If new length exceeds <code>max_size()</code> .

Removes the characters in the range `[__pos1, __pos1 + n)` from this string. In place, the value of `__str` is inserted. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of the result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1468 of file `basic_string.h`.

4.501.3.96 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string<_CharT, _Traits, _Alloc>::replace (size_type __pos, size_type __n1, const _CharT * __s, size_type __n2)`

Replace characters with value of a C substring.

Parameters

<code>__pos</code>	Index of first character to replace.
<code>__n1</code>	Number of characters to be replaced.
<code>__s</code>	C string to insert.
<code>__n2</code>	Number of characters from <code>s</code> to use.

Returns

Reference to this string.

Exceptions

<i>std::out_of_range</i>	If <code>pos1 > size()</code> .
<i>std::length_error</i>	If new length exceeds <code>max_size()</code> .

Removes the characters in the range `[__pos, __pos + __n1)` from this string. In place, the first `__n2` characters of `__s` are inserted, or all of `__s` if `__n2` is too large. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

4.501.3.97 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string<_CharT, _Traits, _Alloc>::replace (size_type __pos, size_type __n1, const _CharT * __s) [inline]`

Replace characters with value of a C string.

Parameters

<code>__pos</code>	Index of first character to replace.
<code>__n1</code>	Number of characters to be replaced.
<code>__s</code>	C string to insert.

Returns

Reference to this string.

Exceptions

<code>std::out_of_range</code>	If <code>pos > size()</code> .
<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .

Removes the characters in the range `[__pos, __pos + __n1)` from this string. In place, the characters of `__s` are inserted. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1513 of file `basic_string.h`.

4.501.3.98 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string<_CharT, _Traits, _Alloc>::replace (size_type __pos, size_type __n1, size_type __n2, _CharT __c) [inline]`

Replace characters with multiple characters.

Parameters

<code>__pos</code>	Index of first character to replace.
<code>__n1</code>	Number of characters to be replaced.
<code>__n2</code>	Number of characters to insert.
<code>__c</code>	Character to insert.

Returns

Reference to this string.

Exceptions

<code>std::out_of_range</code>	If <code>__pos > size()</code> .
<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .

Removes the characters in the range `[pos, pos + n1)` from this string. In place, `__n2` copies of `__c` are inserted. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1537 of file basic_string.h.

```
4.501.3.99  template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT,
    _Traits, _Alloc >::replace ( iterator __i1, iterator __i2, const basic_string< _CharT, _Traits, _Alloc > & __str )
    [inline]
```

Replace range of characters with string.

Parameters

<code>__i1</code>	Iterator referencing start of range to replace.
<code>__i2</code>	Iterator referencing end of range to replace.
<code>__str</code>	String value to insert.

Returns

Reference to this string.

Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Removes the characters in the range `[__i1, __i2)`. In place, the value of `__str` is inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1555 of file basic_string.h.

```
4.501.3.100  template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT,
    _Traits, _Alloc >::replace ( iterator __i1, iterator __i2, const _CharT* __s, size_type __n ) [inline]
```

Replace range of characters with C substring.

Parameters

<code>__i1</code>	Iterator referencing start of range to replace.
<code>__i2</code>	Iterator referencing end of range to replace.
<code>__s</code>	C string value to insert.
<code>__n</code>	Number of characters from <code>s</code> to insert.

Returns

Reference to this string.

Exceptions

<i>std::length_error</i>	If new length exceeds <code>max_size()</code> .
--------------------------	---

Removes the characters in the range `[__i1, __i2)`. In place, the first `__n` characters of `__s` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1574 of file `basic_string.h`.

4.501.3.101 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string<_CharT, _Traits, _Alloc>::replace (iterator __i1, iterator __i2, const _CharT *__s) [inline]`

Replace range of characters with C string.

Parameters

<code>__i1</code>	Iterator referencing start of range to replace.
<code>__i2</code>	Iterator referencing end of range to replace.
<code>__s</code>	C string value to insert.

Returns

Reference to this string.

Exceptions

<i>std::length_error</i>	If new length exceeds <code>max_size()</code> .
--------------------------	---

Removes the characters in the range `[__i1, __i2)`. In place, the characters of `__s` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1595 of file `basic_string.h`.

4.501.3.102 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string<_CharT, _Traits, _Alloc>::replace (iterator __i1, iterator __i2, size_type __n, _CharT __c) [inline]`

Replace range of characters with multiple characters.

Parameters

<code>__i1</code>	Iterator referencing start of range to replace.
<code>__i2</code>	Iterator referencing end of range to replace.

Parameters

<code>__n</code>	Number of characters to insert.
<code>__c</code>	Character to insert.

Returns

Reference to this string.

Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Removes the characters in the range `[__i1,__i2)`. In place, `__n` copies of `__c` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1616 of file `basic_string.h`.

4.501.3.103 `template<typename _CharT, typename _Traits, typename _Alloc> template<class _InputIterator > basic_string& std::basic_string<_CharT, _Traits, _Alloc>::replace (iterator __i1, iterator __i2, _InputIterator __k1, _InputIterator __k2) [inline]`

Replace range of characters with range.

Parameters

<code>__i1</code>	Iterator referencing start of range to replace.
<code>__i2</code>	Iterator referencing end of range to replace.
<code>__k1</code>	Iterator referencing start of range to insert.
<code>__k2</code>	Iterator referencing end of range to insert.

Returns

Reference to this string.

Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Removes the characters in the range `[__i1,__i2)`. In place, characters in the range `[__k1,__k2)` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1640 of file `basic_string.h`.

4.501.3.104 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string<_CharT, _Traits, _Alloc>::replace (iterator __i1, iterator __i2, initializer_list<_CharT> __l) [inline]`

Replace range of characters with initializer_list.

Parameters

<code>__i1</code>	Iterator referencing start of range to replace.
<code>__i2</code>	Iterator referencing end of range to replace.
<code>__l</code>	The initializer_list of characters to insert.

Returns

Reference to this string.

Exceptions

<code>std::length_error</code>	If new length exceeds <code>max_size()</code> .
--------------------------------	---

Removes the characters in the range `[__i1, __i2)`. In place, characters in the range `[__k1, __k2)` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1709 of file `basic_string.h`.

4.501.3.105 `template<typename _CharT, typename _Traits, typename _Alloc> void std::basic_string<_CharT, _Traits, _Alloc>::reserve (size_type __res_arg = 0)`

Attempt to preallocate enough memory for specified number of characters.

Parameters

<code>__res_arg</code>	Number of characters required.
------------------------	--------------------------------

Exceptions

<code>std::length_error</code>	If <code>__res_arg</code> exceeds <code>max_size()</code> .
--------------------------------	---

This function attempts to reserve enough memory for the string to hold the specified number of characters. If the number requested is more than `max_size()`, `length_error` is thrown.

The advantage of this function is that if optimal code is a necessity and the user can determine the string length that will be required, the user can reserve the memory in advance, and thus prevent a possible reallocation of memory and copying of string data.

Referenced by `std::basic_string< _Ch_type >::capacity()`, `std::basic_string< _Ch_type >::push_back()`, and `std::basic_string< _Ch_type >::shrink_to_fit()`.

4.501.3.106 `template<typename _CharT, typename _Traits, typename _Alloc> void std::basic_string< _CharT, _Traits, _Alloc >::resize (size_type __n, _CharT __c)`

Resizes the string to the specified number of characters.

Parameters

<code>__n</code>	Number of characters the string should contain.
<code>__c</code>	Character to fill any new elements.

This function will resize the string to the specified number of characters. If the number is smaller than the string's current size the string is truncated, otherwise the string is extended and new elements are set to `__c`.

Referenced by `std::basic_string< _Ch_type >::max_size()`, and `std::basic_string< _Ch_type >::resize()`.

4.501.3.107 `template<typename _CharT, typename _Traits, typename _Alloc> void std::basic_string< _CharT, _Traits, _Alloc >::resize (size_type __n) [inline]`

Resizes the string to the specified number of characters.

Parameters

<code>__n</code>	Number of characters the string should contain.
------------------	---

This function will resize the string to the specified length. If the new size is smaller than the string's current size the string is truncated, otherwise the string is extended and new characters are default-constructed. For basic types such as `char`, this means setting them to 0.

Definition at line 761 of file `basic_string.h`.

4.501.3.108 `template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string< _CharT, _Traits, _Alloc >::rfind (const basic_string< _CharT, _Traits, _Alloc > & __str, size_type __pos = npos) const [inline], [noexcept]`

Find last position of a string.

Parameters

<code>__str</code>	String to locate.
<code>__pos</code>	Index of character to search back from (default end).

Returns

Index of start of last occurrence.

Starting from `__pos`, searches backward for value of `__str` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 1909 of file `basic_string.h`.

Referenced by `std::basic_string<_Ch_type>::find_last_of()`, and `std::basic_string<_Ch_type>::rfind()`.

4.501.3.109 `template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string<_CharT, _Traits, _Alloc>::rfind (const _CharT * __s, size_type __pos, size_type __n) const`

Find last position of a C substring.

Parameters

<code>__s</code>	C string to locate.
<code>__pos</code>	Index of character to search back from.
<code>__n</code>	Number of characters from <code>s</code> to search for.

Returns

Index of start of last occurrence.

Starting from `__pos`, searches backward for the first `__n` characters in `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

4.501.3.110 `template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string<_CharT, _Traits, _Alloc>::rfind (const _CharT * __s, size_type __pos = npos) const [inline]`

Find last position of a C string.

Parameters

<code>__s</code>	C string to locate.
<code>__pos</code>	Index of character to start search at (default end).

Returns

Index of start of last occurrence.

Starting from `__pos`, searches backward for the value of `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 1939 of file `basic_string.h`.

4.501.3.111 `template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string< _CharT, _Traits, _Alloc >::rfind (_CharT __c, size_type __pos = npos) const [noexcept]`

Find last position of a character.

Parameters

<code>__c</code>	Character to locate.
<code>__pos</code>	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from `__pos`, searches backward for `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

4.501.3.112 `template<typename _CharT, typename _Traits, typename _Alloc> void std::basic_string< _CharT, _Traits, _Alloc >::shrink_to_fit () [inline], [noexcept]`

A non-binding request to reduce `capacity()` to `size()`.

Definition at line 767 of file `basic_string.h`.

4.501.3.113 `template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string< _CharT, _Traits, _Alloc >::size () const [inline], [noexcept]`

Returns the number of characters in the string, not including any null-termination.

Definition at line 723 of file `basic_string.h`.

Referenced by `std::basic_regex< _Ch_type, _Rx_traits >::assign()`, `std::basic_string< _Ch_type >::assign()`, `std::basic_string< _Ch_type >::at()`, `std::basic_string< _Ch_type >::back()`, `std::basic_string< _Ch_type >::cend()`, `std::basic_string< _Ch_type >::clear()`, `std::basic_string< _Ch_type >::compare()`, `std::basic_string< _Ch_type >::empty()`, `std::basic_string< _Ch_type >::end()`, `std::basic_string< _Ch_type >::find()`, `std::basic_string< _Ch_type >::find_first_not_of()`, `std::basic_string< _Ch_type >::find_first_of()`, `std::basic_string< _Ch_type >::find_last_not_of()`, `std::basic_string< _Ch_type >::find_last_of()`, `std::match_results< _Bi_iter >::format()`, `std::basic_string< _Ch_type >::insert()`, `std::operator+`, `std::operator<<`, `std::operator==()`, `std::basic_string< _Ch_type >::operator[]()`, `std::basic_string< _Ch_type >::pop_back()`, `std::basic_string< _Ch_type >::push_back()`, `std::basic_string< _Ch_type >::replace()`, `std::basic_string< _Ch_type >::rfind()`, `std::basic_string< _Ch_type >::shrink_to_fit()`, and `std::basic_regex_traits< _Ch_type >::transform()`.

4.501.3.114 `template<typename _CharT, typename _Traits, typename _Alloc> basic_string std::basic_string< _CharT, _Traits, _Alloc >::substr (size_type __pos = 0, size_type __n = npos) const [inline]`

Get a substring.

Parameters

<code>__pos</code>	Index of first character (default 0).
<code>__n</code>	Number of characters in substring (default remainder).

Returns

The new string.

Exceptions

<code>std::out_of_range</code>	If <code>__pos > size()</code> .
--------------------------------	-------------------------------------

Construct and return a new string using the `__n` characters starting at `__pos`. If the string is too short, use the remainder of the characters. If `__pos` is beyond the end of the string, `out_of_range` is thrown.

Definition at line 2224 of file `basic_string.h`.

Referenced by `__gnu_profile::__report()`.

4.501.3.115 `template<typename _CharT, typename _Traits, typename _Alloc> void std::basic_string< _CharT, _Traits, _Alloc >::swap (basic_string< _CharT, _Traits, _Alloc > & __s)`

Swap contents with another string.

Parameters

<code>__s</code>	String to swap with.
------------------	----------------------

Exchanges the contents of this string with that of `__s` in constant time.

Referenced by `std::basic_string< _Ch_type >::assign()`, `std::basic_string< _Ch_type >::operator=()`, `std::basic_string< _Ch_type >::replace()`, and `std::swap()`.

4.501.4 Member Data Documentation

4.501.4.1 `template<typename _CharT, typename _Traits, typename _Alloc> const size_type std::basic_string< _CharT, _Traits, _Alloc >::npos [static]`

Value returned by various member functions when they fail.

Definition at line 285 of file `basic_string.h`.

The documentation for this class was generated from the following file:

- [basic_string.h](#)

4.502 std::bernoulli_distribution Class Reference

Classes

- struct [param_type](#)

Public Types

- typedef bool [result_type](#)

Public Member Functions

- [bernoulli_distribution](#) (double __p=0.5)
- [bernoulli_distribution](#) (const [param_type](#) &__p)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator >
void [__generate](#) (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator >
void [__generate](#) (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const [param_type](#) &__p)
- template<typename _UniformRandomNumberGenerator >
void [__generate](#) ([result_type](#) *__f, [result_type](#) *__t, _UniformRandomNumberGenerator &__urng, const [param_type](#) &__p)
- [result_type](#) max () const
- [result_type](#) min () const
- template<typename _UniformRandomNumberGenerator >
[result_type](#) [operator\(\)](#) (_UniformRandomNumberGenerator &__urng)
- template<typename _UniformRandomNumberGenerator >
[result_type](#) [operator\(\)](#) (_UniformRandomNumberGenerator &__urng, const [param_type](#) &__p)
- double [p](#) () const
- [param_type](#) [param](#) () const
- void [param](#) (const [param_type](#) &__param)
- void [reset](#) ()

Friends

- bool [operator==](#) (const [bernoulli_distribution](#) &__d1, const [bernoulli_distribution](#) &__d2)

4.502.1 Detailed Description

A Bernoulli random number distribution.

Generates a sequence of true and false values with likelihood p that true will come up and $(1 - p)$ that false will appear.

Definition at line 3574 of file random.h.

4.502.2 Member Typedef Documentation

4.502.2.1 typedef bool std::bernoulli_distribution::result_type

The type of the range of the distribution.

Definition at line 3578 of file random.h.

4.502.3 Constructor & Destructor Documentation

4.502.3.1 std::bernoulli_distribution::bernoulli_distribution (double __p = 0.5) [inline],[explicit]

Constructs a Bernoulli distribution with likelihood p .

Parameters

\leftarrow __p	[IN] The likelihood of a true result being returned. Must be in the interval $[0, 1]$.
---------------------	---

Definition at line 3611 of file random.h.

4.502.4 Member Function Documentation

4.502.4.1 result_type std::bernoulli_distribution::max () const [inline]

Returns the least upper bound value of the distribution.

Definition at line 3661 of file random.h.

References std::max().

4.502.4.2 result_type std::bernoulli_distribution::min () const [inline]

Returns the greatest lower bound value of the distribution.

Definition at line 3654 of file random.h.

References std::min().

4.502.4.3 template<typename _UniformRandomNumberGenerator > result_type std::bernoulli_distribution::operator() (_UniformRandomNumberGenerator & __urng) [inline]

Generating functions.

Definition at line 3669 of file random.h.

4.502.4.4 `double std::bernoulli_distribution::p () const [inline]`

Returns the `p` parameter of the distribution.

Definition at line 3632 of file `random.h`.

4.502.4.5 `param_type std::bernoulli_distribution::param () const [inline]`

Returns the parameter set of the distribution.

Definition at line 3639 of file `random.h`.

Referenced by `std::operator>>()`.

4.502.4.6 `void std::bernoulli_distribution::param (const param_type & __param) [inline]`

Sets the parameter set of the distribution.

Parameters

<code>__param</code>	The new parameter set of the distribution.
----------------------	--

Definition at line 3647 of file `random.h`.

4.502.4.7 `void std::bernoulli_distribution::reset () [inline]`

Resets the distribution state.

Does nothing for a Bernoulli distribution.

Definition at line 3626 of file `random.h`.

4.502.5 Friends And Related Function Documentation

4.502.5.1 `bool operator== (const bernoulli_distribution & __d1, const bernoulli_distribution & __d2) [friend]`

Return true if two Bernoulli distributions have the same parameters.

Definition at line 3711 of file `random.h`.

The documentation for this class was generated from the following file:

- [random.h](#)

4.503 `std::bernoulli_distribution::param_type` Struct Reference

Public Types

- typedef [bernoulli_distribution](#) **distribution_type**

Public Member Functions

- **param_type** (double __p=0.5)
- double **p** () const

Friends

- bool **operator==** (const [param_type](#) &__p1, const [param_type](#) &__p2)

4.503.1 Detailed Description

Parameter type.

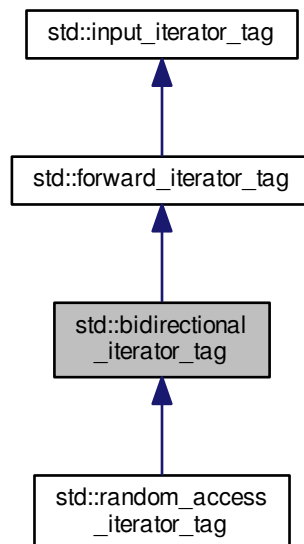
Definition at line 3580 of file `random.h`.

The documentation for this struct was generated from the following file:

- [random.h](#)

4.504 `std::bidirectional_iterator_tag` Struct Reference

Inheritance diagram for `std::bidirectional_iterator_tag`:



4.504.1 Detailed Description

Bidirectional iterators support a superset of forward iterator operations.

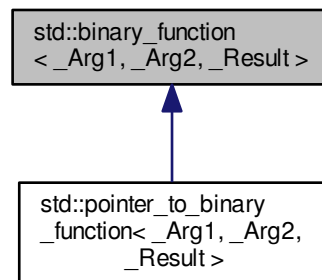
Definition at line 99 of file `stl_iterator_base_types.h`.

The documentation for this struct was generated from the following file:

- [stl_iterator_base_types.h](#)

4.505 `std::binary_function<_Arg1, _Arg2, _Result>` Struct Template Reference

Inheritance diagram for `std::binary_function<_Arg1, _Arg2, _Result>`:



Public Types

- `typedef _Arg1` [first_argument_type](#)
- `typedef _Result` [result_type](#)
- `typedef _Arg2` [second_argument_type](#)

4.505.1 Detailed Description

```
template<typename _Arg1, typename _Arg2, typename _Result>
struct std::binary_function<_Arg1, _Arg2, _Result>
```

This is one of the [functor base classes](#).

Definition at line 118 of file `stl_function.h`.

4.505.2 Member Typedef Documentation

4.505.2.1 `template<typename _Arg1, typename _Arg2, typename _Result> typedef _Arg1 std::binary_function< _Arg1, _Arg2, _Result >::first_argument_type`

`first_argument_type` is the type of the first argument

Definition at line 121 of file `stl_function.h`.

4.505.2.2 `template<typename _Arg1, typename _Arg2, typename _Result> typedef _Result std::binary_function< _Arg1, _Arg2, _Result >::result_type`

`result_type` is the return type

Definition at line 127 of file `stl_function.h`.

4.505.2.3 `template<typename _Arg1, typename _Arg2, typename _Result> typedef _Arg2 std::binary_function< _Arg1, _Arg2, _Result >::second_argument_type`

`second_argument_type` is the type of the second argument

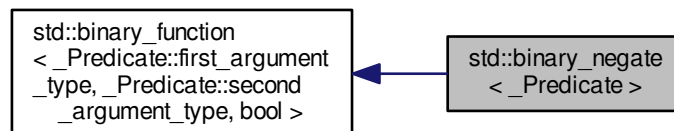
Definition at line 124 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl_function.h](#)

4.506 std::binary_negate< _Predicate > Class Template Reference

Inheritance diagram for `std::binary_negate< _Predicate >`:



Public Types

- typedef `_Predicate::first_argument_type` [first_argument_type](#)
- typedef `bool` [result_type](#)
- typedef `_Predicate::second_argument_type` [second_argument_type](#)

Public Member Functions

- **binary_negate** (const _Predicate &__x)
- bool **operator()** (const typename _Predicate::first_argument_type &__x, const typename _Predicate::second_argument_type &__y) const

Protected Attributes

- _Predicate **_M_pred**

4.506.1 Detailed Description

```
template<typename _Predicate>
class std::binary_negate<_Predicate >
```

One of the [negation functors](#).

Definition at line 727 of file `stl_function.h`.

4.506.2 Member Typedef Documentation

4.506.2.1 `typedef _Predicate::first_argument_type std::binary_function< _Predicate::first_argument_type , _Predicate::second_argument_type , bool >::first_argument_type` [\[inherited\]](#)

`first_argument_type` is the type of the first argument

Definition at line 121 of file `stl_function.h`.

4.506.2.2 `typedef bool std::binary_function< _Predicate::first_argument_type , _Predicate::second_argument_type , bool >::result_type` [\[inherited\]](#)

`result_type` is the return type

Definition at line 127 of file `stl_function.h`.

4.506.2.3 `typedef _Predicate::second_argument_type std::binary_function< _Predicate::first_argument_type , _Predicate::second_argument_type , bool >::second_argument_type` [\[inherited\]](#)

`second_argument_type` is the type of the second argument

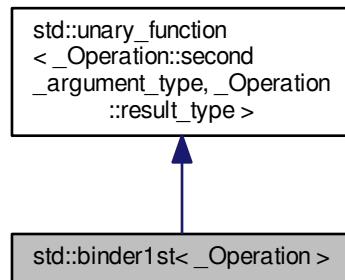
Definition at line 124 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [stl_function.h](#)

4.507 std::binder1st< _Operation > Class Template Reference

Inheritance diagram for std::binder1st< _Operation >:



Public Types

- typedef `_Operation::second_argument_type` [argument_type](#)
- typedef `_Operation::result_type` [result_type](#)

Public Member Functions

- **binder1st** (const `_Operation` &__x, const typename `_Operation::first_argument_type` &__y)
- `_Operation::result_type` **operator()** (const typename `_Operation::second_argument_type` &__x) const
- `_Operation::result_type` **operator()** (typename `_Operation::second_argument_type` &__x) const

Protected Attributes

- `_Operation` **op**
- `_Operation::first_argument_type` **value**

4.507.1 Detailed Description

```
template<typename _Operation>
class std::binder1st< _Operation >
```

One of the [binder functors](#).

Definition at line 104 of file `binders.h`.

4.507.2 Member Typedef Documentation

4.507.2.1 `typedef _Operation::second_argument_type std::unary_function< _Operation::second_argument_type ,
_Operation::result_type >::argument_type` [inherited]

`argument_type` is the type of the argument

Definition at line 108 of file `stl_function.h`.

4.507.2.2 `typedef _Operation::result_type std::unary_function< _Operation::second_argument_type , _Operation::result_type
>::result_type` [inherited]

`result_type` is the return type

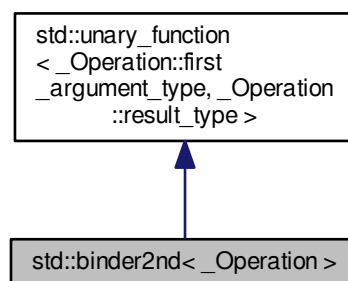
Definition at line 111 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [binders.h](#)

4.508 `std::binder2nd< _Operation >` Class Template Reference

Inheritance diagram for `std::binder2nd< _Operation >`:



Public Types

- `typedef _Operation::first_argument_type` [argument_type](#)
- `typedef _Operation::result_type` [result_type](#)

Public Member Functions

- **binder2nd** (const `_Operation` &__x, const typename `_Operation::second_argument_type` &__y)
- `_Operation::result_type` **operator()** (const typename `_Operation::first_argument_type` &__x) const
- `_Operation::result_type` **operator()** (typename `_Operation::first_argument_type` &__x) const

Protected Attributes

- `_Operation` **op**
- `_Operation::second_argument_type` **value**

4.508.1 Detailed Description

```
template<typename _Operation>
class std::binder2nd<_Operation>
```

One of the [binder functors](#).

Definition at line 139 of file `binders.h`.

4.508.2 Member Typedef Documentation

4.508.2.1 `typedef _Operation::first_argument_type std::unary_function<_Operation::first_argument_type ,
_Operation::result_type>::argument_type` `[inherited]`

`argument_type` is the type of the argument

Definition at line 108 of file `stl_function.h`.

4.508.2.2 `typedef _Operation::result_type std::unary_function<_Operation::first_argument_type ,_Operation::result_type
>::result_type` `[inherited]`

`result_type` is the return type

Definition at line 111 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [binders.h](#)

4.509 `std::binomial_distribution<_IntType>` Class Template Reference

Classes

- struct [param_type](#)

Public Types

- typedef `_IntType` [result_type](#)

Public Member Functions

- **binomial_distribution** (`_IntType __t=_IntType(1)`, `double __p=0.5`)
- **binomial_distribution** (const [param_type](#) &__p)
- template<typename `_ForwardIterator` , typename `_UniformRandomNumberGenerator` >
void **generate** (`_ForwardIterator __f`, `_ForwardIterator __t`, `_UniformRandomNumberGenerator &__urng`)
- template<typename `_ForwardIterator` , typename `_UniformRandomNumberGenerator` >
void **generate** (`_ForwardIterator __f`, `_ForwardIterator __t`, `_UniformRandomNumberGenerator &__urng`, const [param_type](#) &__p)
- template<typename `_UniformRandomNumberGenerator` >
void **generate** ([result_type](#) *__f, [result_type](#) *__t, `_UniformRandomNumberGenerator &__urng`, const [param_type](#) &__p)
- [result_type](#) **max** () const
- [result_type](#) **min** () const
- template<typename `_UniformRandomNumberGenerator` >
[result_type](#) **operator()** (`_UniformRandomNumberGenerator &__urng`)
- template<typename `_UniformRandomNumberGenerator` >
[result_type](#) **operator()** (`_UniformRandomNumberGenerator &__urng`, const [param_type](#) &__p)
- double **p** () const
- [param_type](#) **param** () const
- void **param** (const [param_type](#) &__param)
- void **reset** ()
- `_IntType` **t** () const

Friends

- template<typename `_IntType1` , typename `_CharT` , typename `_Traits` >
std::basic_ostream< `_CharT`, `_Traits` > & **operator<<** (std::basic_ostream< `_CharT`, `_Traits` > &__os, const [std::binomial_distribution](#)< `_IntType1` > &__x)
- bool **operator==** (const [binomial_distribution](#) &__d1, const [binomial_distribution](#) &__d2)
- template<typename `_IntType1` , typename `_CharT` , typename `_Traits` >
std::basic_istream< `_CharT`, `_Traits` > & **operator>>** (std::basic_istream< `_CharT`, `_Traits` > &__is, [std::binomial_distribution](#)< `_IntType1` > &__x)

4.509.1 Detailed Description

```
template<typename _IntType = int>
class std::binomial_distribution< _IntType >
```

A discrete binomial random number distribution.

The formula for the binomial probability density function is $p(i|t, p) = \binom{t}{i} p^i (1-p)^{t-i}$ where t and p are the parameters of the distribution.

Definition at line 3779 of file random.h.

4.509.2 Member Typedef Documentation

4.509.2.1 template<typename _IntType = int> typedef _IntType std::binomial_distribution< _IntType >::result_type

The type of the range of the distribution.

Definition at line 3782 of file random.h.

4.509.3 Member Function Documentation

4.509.3.1 template<typename _IntType = int> result_type std::binomial_distribution< _IntType >::max () const
[inline]

Returns the least upper bound value of the distribution.

Definition at line 3889 of file random.h.

4.509.3.2 template<typename _IntType = int> result_type std::binomial_distribution< _IntType >::min () const
[inline]

Returns the greatest lower bound value of the distribution.

Definition at line 3882 of file random.h.

4.509.3.3 template<typename _IntType = int> template<typename _UniformRandomNumberGenerator > result_type
std::binomial_distribution< _IntType >::operator() (_UniformRandomNumberGenerator & __urng) [inline]

Generating functions.

Definition at line 3897 of file random.h.

4.509.3.4 template<typename _IntType = int> double std::binomial_distribution< _IntType >::p () const [inline]

Returns the distribution p parameter.

Definition at line 3860 of file random.h.

4.509.3.5 template<typename _IntType = int> param_type std::binomial_distribution< _IntType >::param () const
[inline]

Returns the parameter set of the distribution.

Definition at line 3867 of file random.h.

4.509.3.6 template<typename _IntType = int> void std::binomial_distribution< _IntType >::param (const param_type &
__param) [inline]

Sets the parameter set of the distribution.

Parameters

<code>__param</code>	The new parameter set of the distribution.
----------------------	--

Definition at line 3875 of file random.h.

4.509.3.7 `template<typename _IntType = int> void std::binomial_distribution<_IntType>::reset () [inline]`

Resets the distribution state.

Definition at line 3846 of file random.h.

4.509.3.8 `template<typename _IntType = int> _IntType std::binomial_distribution<_IntType>::t () const [inline]`

Returns the distribution `t` parameter.

Definition at line 3853 of file random.h.

4.509.4 Friends And Related Function Documentation

4.509.4.1 `template<typename _IntType = int> template<typename _IntType1, typename _CharT, typename _Traits > std::basic_ostream<_CharT, _Traits>& operator<< (std::basic_ostream<_CharT, _Traits> & __os, const std::binomial_distribution<_IntType1> & __x) [friend]`

Inserts a `binomial_distribution` random number distribution `__x` into the output stream `__os`.

Parameters

<code>__os</code>	An output stream.
<code>__x</code>	A <code>binomial_distribution</code> random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

4.509.4.2 `template<typename _IntType = int> bool operator== (const binomial_distribution<_IntType> & __d1, const binomial_distribution<_IntType> & __d2) [friend]`

Return true if two binomial distributions have the same parameters and the sequences that would be generated are equal.

Definition at line 3933 of file random.h.

4.509.4.3 `template<typename _IntType = int> template<typename _IntType1, typename _CharT, typename _Traits > std::basic_istream<_CharT, _Traits>& operator>> (std::basic_istream<_CharT, _Traits> & __is, std::binomial_distribution<_IntType1> & __x) [friend]`

Extracts a `binomial_distribution` random number distribution `__x` from the input stream `__is`.

Parameters

<code>__is</code>	An input stream.
<code>__x</code>	A <code>binomial_distribution</code> random number generator engine.

Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following file:

- [random.h](#)

4.510 `std::binomial_distribution<_IntType>::param_type` Struct Reference

Public Types

- typedef `binomial_distribution<_IntType>` **distribution_type**

Public Member Functions

- **param_type** (`_IntType __t=_IntType(1)`, `double __p=0.5`)
- `double p () const`
- `_IntType t () const`

Friends

- class **binomial_distribution<_IntType>**
- `bool operator== (const param_type &__p1, const param_type &__p2)`

4.510.1 Detailed Description

```
template<typename _IntType = int>
struct std::binomial_distribution<_IntType>::param_type
```

Parameter type.

Definition at line 3788 of file `random.h`.

The documentation for this struct was generated from the following file:

- [random.h](#)

4.511 std::cauchy_distribution<_RealType> Class Template Reference

Classes

- struct [param_type](#)

Public Types

- typedef _RealType [result_type](#)

Public Member Functions

- **cauchy_distribution** (_RealType __a=_RealType(0), _RealType __b=_RealType(1))
- **cauchy_distribution** (const [param_type](#) &__p)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator >
void **generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator >
void **generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const [param_type](#) &__p)
- template<typename _UniformRandomNumberGenerator >
void **generate** ([result_type](#) *__f, [result_type](#) *__t, _UniformRandomNumberGenerator &__urng, const [param_type](#) &__p)
- _RealType **a** () const
- _RealType **b** () const
- [result_type](#) **max** () const
- [result_type](#) **min** () const
- template<typename _UniformRandomNumberGenerator >
[result_type](#) **operator()** (_UniformRandomNumberGenerator &__urng)
- template<typename _UniformRandomNumberGenerator >
[result_type](#) **operator()** (_UniformRandomNumberGenerator &__urng, const [param_type](#) &__p)
- [param_type](#) **param** () const
- void **param** (const [param_type](#) &__param)
- void **reset** ()

Friends

- bool **operator==** (const [cauchy_distribution](#) &__d1, const [cauchy_distribution](#) &__d2)

4.511.1 Detailed Description

```
template<typename _RealType = double>
class std::cauchy_distribution<_RealType>
```

A `cauchy_distribution` random number distribution.

The formula for the normal probability mass function is $p(x|a, b) = (\pi b(1 + (\frac{x-a}{b})^2))^{-1}$

Definition at line 2931 of file `random.h`.

4.511.2 Member Typedef Documentation

4.511.2.1 `template<typename _RealType = double> typedef _RealType std::cauchy_distribution<_RealType>::result_type`

The type of the range of the distribution.

Definition at line 2934 of file random.h.

4.511.3 Member Function Documentation

4.511.3.1 `template<typename _RealType = double> result_type std::cauchy_distribution<_RealType>::max () const [inline]`

Returns the least upper bound value of the distribution.

Definition at line 3022 of file random.h.

References `std::max()`.

4.511.3.2 `template<typename _RealType = double> result_type std::cauchy_distribution<_RealType>::min () const [inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 3015 of file random.h.

4.511.3.3 `template<typename _RealType = double> template<typename _UniformRandomNumberGenerator> result_type std::cauchy_distribution<_RealType>::operator() (_UniformRandomNumberGenerator & __urng) [inline]`

Generating functions.

Definition at line 3030 of file random.h.

4.511.3.4 `template<typename _RealType = double> param_type std::cauchy_distribution<_RealType>::param () const [inline]`

Returns the parameter set of the distribution.

Definition at line 3000 of file random.h.

4.511.3.5 `template<typename _RealType = double> void std::cauchy_distribution<_RealType>::param (const param_type & __param) [inline]`

Sets the parameter set of the distribution.

Parameters

<code>__param</code>	The new parameter set of the distribution.
----------------------	--

Definition at line 3008 of file random.h.

4.511.3.6 `template<typename _RealType = double> void std::cauchy_distribution<_RealType>::reset () [inline]`

Resets the distribution state.

Definition at line 2982 of file random.h.

4.511.4 Friends And Related Function Documentation

4.511.4.1 `template<typename _RealType = double> bool operator==(const cauchy_distribution<_RealType> &__d1, const cauchy_distribution<_RealType> &__d2) [friend]`

Return true if two Cauchy distributions have the same parameters.

Definition at line 3065 of file random.h.

The documentation for this class was generated from the following file:

- [random.h](#)

4.512 `std::cauchy_distribution<_RealType>::param_type` Struct Reference

Public Types

- typedef `cauchy_distribution<_RealType>` **distribution_type**

Public Member Functions

- **param_type** (`_RealType __a=_RealType(0), _RealType __b=_RealType(1)`)
- `_RealType a` () const
- `_RealType b` () const

Friends

- bool **operator==** (const `param_type` &__p1, const `param_type` &__p2)

4.512.1 Detailed Description

```
template<typename _RealType = double>
struct std::cauchy_distribution<_RealType>::param_type
```

Parameter type.

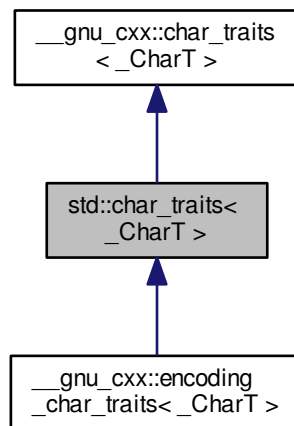
Definition at line 2940 of file random.h.

The documentation for this struct was generated from the following file:

- [random.h](#)

4.513 std::char_traits<_CharT> Struct Template Reference

Inheritance diagram for std::char_traits<_CharT>:



Public Types

- typedef _CharT **char_type**
- typedef _Char_types<_CharT>::int_type **int_type**
- typedef _Char_types<_CharT>::off_type **off_type**
- typedef _Char_types<_CharT>::pos_type **pos_type**
- typedef _Char_types<_CharT>::state_type **state_type**

Static Public Member Functions

- static void **assign** (char_type &__c1, const char_type &__c2)
- static char_type * **assign** (char_type * __s, std::size_t __n, char_type __a)
- static int **compare** (const char_type * __s1, const char_type * __s2, std::size_t __n)
- static char_type * **copy** (char_type * __s1, const char_type * __s2, std::size_t __n)
- static constexpr int_type **eof** ()
- static constexpr bool **eq** (const char_type &__c1, const char_type &__c2)
- static constexpr bool **eq_int_type** (const int_type &__c1, const int_type &__c2)
- static const char_type * **find** (const char_type * __s, std::size_t __n, const char_type &__a)
- static std::size_t **length** (const char_type * __s)
- static constexpr bool **lt** (const char_type &__c1, const char_type &__c2)
- static char_type * **move** (char_type * __s1, const char_type * __s2, std::size_t __n)
- static constexpr int_type **not_eof** (const int_type &__c)
- static constexpr char_type **to_char_type** (const int_type &__c)
- static constexpr int_type **to_int_type** (const char_type &__c)

4.513.1 Detailed Description

```
template<class _CharT>
struct std::char_traits< _CharT >
```

Basis for explicit traits specializations.

Note

For any given actual character type, this definition is probably wrong. Since this is just a thin wrapper around `__gnu_cxx::char_traits`, it is possible to achieve a more appropriate definition by specializing `__gnu_cxx::char_traits`.

See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt05ch13s03.html> for advice on how to make use of this class for *unusual* character types. Also, check out `include/ext/pod_char_traits.h`.

Definition at line 227 of file `char_traits.h`.

The documentation for this struct was generated from the following file:

- [char_traits.h](#)

4.514 `std::char_traits< __gnu_cxx::character< _Value, _Int, _St > >` Struct Template Reference

Public Types

- typedef `__gnu_cxx::character< _Value, _Int, _St >` **char_type**
- typedef `char_type::int_type` **int_type**
- typedef `streamoff` **off_type**
- typedef `fpos< state_type >` **pos_type**
- typedef `char_type::state_type` **state_type**

Static Public Member Functions

- static void **assign** ([char_type](#) &__c1, const [char_type](#) &__c2)
- static [char_type](#) * **assign** ([char_type](#) * __s, size_t __n, [char_type](#) __a)
- static int **compare** (const [char_type](#) * __s1, const [char_type](#) * __s2, size_t __n)
- static [char_type](#) * **copy** ([char_type](#) * __s1, const [char_type](#) * __s2, size_t __n)
- static int_type **eof** ()
- static bool **eq** (const [char_type](#) &__c1, const [char_type](#) &__c2)
- static bool **eq_int_type** (const int_type &__c1, const int_type &__c2)
- static const [char_type](#) * **find** (const [char_type](#) * __s, size_t __n, const [char_type](#) &__a)
- static size_t **length** (const [char_type](#) * __s)
- static bool **lt** (const [char_type](#) &__c1, const [char_type](#) &__c2)
- static [char_type](#) * **move** ([char_type](#) * __s1, const [char_type](#) * __s2, size_t __n)
- static int_type **not_eof** (const int_type &__c)
- static [char_type](#) **to_char_type** (const int_type &__i)
- static int_type **to_int_type** (const [char_type](#) &__c)

4.514.1 Detailed Description

```
template<typename _Value, typename _Int, typename _St>
struct std::char_traits< __gnu_cxx::character< _Value, _Int, _St > >
```

char_traits<__gnu_cxx::character> specialization.

Definition at line 97 of file pod_char_traits.h.

The documentation for this struct was generated from the following file:

- [pod_char_traits.h](#)

4.515 std::char_traits< char > Struct Template Reference

Public Types

- typedef char **char_type**
- typedef int **int_type**
- typedef [streamoff](#) **off_type**
- typedef [streampos](#) **pos_type**
- typedef mbstate_t **state_type**

Static Public Member Functions

- static void **assign** (char_type &__c1, const char_type &__c2) noexcept
- static char_type * **assign** (char_type *__s, size_t __n, char_type __a)
- static int **compare** (const char_type *__s1, const char_type *__s2, size_t __n)
- static char_type * **copy** (char_type *__s1, const char_type *__s2, size_t __n)
- static constexpr int_type **eof** () noexcept
- static constexpr bool **eq** (const char_type &__c1, const char_type &__c2) noexcept
- static constexpr bool **eq_int_type** (const int_type &__c1, const int_type &__c2) noexcept
- static const char_type * **find** (const char_type *__s, size_t __n, const char_type &__a)
- static size_t **length** (const char_type *__s)
- static constexpr bool **lt** (const char_type &__c1, const char_type &__c2) noexcept
- static char_type * **move** (char_type *__s1, const char_type *__s2, size_t __n)
- static constexpr int_type **not_eof** (const int_type &__c) noexcept
- static constexpr char_type **to_char_type** (const int_type &__c) noexcept
- static constexpr int_type **to_int_type** (const char_type &__c) noexcept

4.515.1 Detailed Description

```
template<>
struct std::char_traits< char >
```

21.1.3.1 char_traits specializations

Definition at line 233 of file char_traits.h.

The documentation for this struct was generated from the following file:

- [char_traits.h](#)

4.516 std::char_traits< wchar_t > Struct Template Reference

Public Types

- typedef wchar_t **char_type**
- typedef wint_t **int_type**
- typedef [streamoff](#) **off_type**
- typedef [wstreampos](#) **pos_type**
- typedef mbstate_t **state_type**

Static Public Member Functions

- static void **assign** (char_type &__c1, const char_type &__c2) noexcept
- static char_type * **assign** (char_type *__s, size_t __n, char_type __a)
- static int **compare** (const char_type *__s1, const char_type *__s2, size_t __n)
- static char_type * **copy** (char_type *__s1, const char_type *__s2, size_t __n)
- static constexpr int_type **eof** () noexcept
- static constexpr bool **eq** (const char_type &__c1, const char_type &__c2) noexcept
- static constexpr bool **eq_int_type** (const int_type &__c1, const int_type &__c2) noexcept
- static const char_type * **find** (const char_type *__s, size_t __n, const char_type &__a)
- static size_t **length** (const char_type *__s)
- static constexpr bool **lt** (const char_type &__c1, const char_type &__c2) noexcept
- static char_type * **move** (char_type *__s1, const char_type *__s2, size_t __n)
- static constexpr int_type **not_eof** (const int_type &__c) noexcept
- static constexpr char_type **to_char_type** (const int_type &__c) noexcept
- static constexpr int_type **to_int_type** (const char_type &__c) noexcept

4.516.1 Detailed Description

```
template<>
struct std::char_traits< wchar_t >
```

21.1.3.2 char_traits specializations

Definition at line 308 of file char_traits.h.

The documentation for this struct was generated from the following file:

- [char_traits.h](#)

4.517 `std::chi_squared_distribution<_RealType>` Class Template Reference

Classes

- struct [param_type](#)

Public Types

- typedef _RealType [result_type](#)

Public Member Functions

- **chi_squared_distribution** (_RealType __n=_RealType(1))
- **chi_squared_distribution** (const [param_type](#) &__p)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator >
void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator >
void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const [param_type](#) &__p)
- template<typename _UniformRandomNumberGenerator >
void **__generate** ([result_type](#) *__f, [result_type](#) *__t, _UniformRandomNumberGenerator &__urng)
- template<typename _UniformRandomNumberGenerator >
void **__generate** ([result_type](#) *__f, [result_type](#) *__t, _UniformRandomNumberGenerator &__urng, const [param_type](#) &__p)
- [result_type](#) **max** () const
- [result_type](#) **min** () const
- _RealType **n** () const
- template<typename _UniformRandomNumberGenerator >
[result_type](#) **operator()** (_UniformRandomNumberGenerator &__urng)
- template<typename _UniformRandomNumberGenerator >
[result_type](#) **operator()** (_UniformRandomNumberGenerator &__urng, const [param_type](#) &__p)
- [param_type](#) **param** () const
- void **param** (const [param_type](#) &__param)
- void **reset** ()

Friends

- template<typename _RealType1, typename _CharT, typename _Traits >
std::basic_ostream< _CharT, _Traits > & **operator<<** (std::basic_ostream< _CharT, _Traits > &__os, const [std::chi_squared_distribution](#)< _RealType1 > &__x)
- bool **operator==** (const [chi_squared_distribution](#) &__d1, const [chi_squared_distribution](#) &__d2)
- template<typename _RealType1, typename _CharT, typename _Traits >
std::basic_istream< _CharT, _Traits > & **operator>>** (std::basic_istream< _CharT, _Traits > &__is, [std::chi_squared_distribution](#)< _RealType1 > &__x)

4.517.1 Detailed Description

```
template<typename _RealType = double>
class std::chi_squared_distribution< _RealType >
```

A [chi_squared_distribution](#) random number distribution.

The formula for the normal probability mass function is $p(x|n) = \frac{x^{(n/2)-1} e^{-x/2}}{\Gamma(n/2) 2^{n/2}}$

Definition at line 2721 of file random.h.

4.517.2 Member Typedef Documentation

4.517.2.1 `template<typename _RealType = double> typedef _RealType std::chi_squared_distribution<_RealType>::result_type`

The type of the range of the distribution.

Definition at line 2724 of file random.h.

4.517.3 Member Function Documentation

4.517.3.1 `template<typename _RealType = double> result_type std::chi_squared_distribution<_RealType>::max ()
const [inline]`

Returns the least upper bound value of the distribution.

Definition at line 2801 of file random.h.

References `std::max()`.

4.517.3.2 `template<typename _RealType = double> result_type std::chi_squared_distribution<_RealType>::min ()
const [inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 2794 of file random.h.

4.517.3.3 `template<typename _RealType = double> template<typename _UniformRandomNumberGenerator> result_type
std::chi_squared_distribution<_RealType>::operator() (_UniformRandomNumberGenerator & __urng)
[inline]`

Generating functions.

Definition at line 2809 of file random.h.

4.517.3.4 `template<typename _RealType = double> param_type std::chi_squared_distribution<_RealType>::param ()
const [inline]`

Returns the parameter set of the distribution.

Definition at line 2779 of file random.h.

4.517.3.5 `template<typename _RealType = double> void std::chi_squared_distribution<_RealType>::param (const
param_type & __param) [inline]`

Sets the parameter set of the distribution.

Parameters

<code>__param</code>	The new parameter set of the distribution.
----------------------	--

Definition at line 2787 of file random.h.

4.517.3.6 `template<typename _RealType = double> void std::chi_squared_distribution<_RealType>::reset ()`
`[inline]`

Resets the distribution state.

Definition at line 2765 of file random.h.

4.517.4 Friends And Related Function Documentation

4.517.4.1 `template<typename _RealType = double> template<typename _RealType1 , typename _CharT , typename _Traits`
`> std::basic_ostream<_CharT, _Traits>& operator<< (std::basic_ostream<_CharT, _Traits> & __os, const`
`std::chi_squared_distribution<_RealType1> & __x) [friend]`

Inserts a `chi_squared_distribution` random number distribution `__x` into the output stream `__os`.

Parameters

<code>__os</code>	An output stream.
<code>__x</code>	A <code>chi_squared_distribution</code> random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

4.517.4.2 `template<typename _RealType = double> bool operator==(const chi_squared_distribution<_RealType> & __d1,`
`const chi_squared_distribution<_RealType> & __d2) [friend]`

Return true if two Chi-squared distributions have the same parameters and the sequences that would be generated are equal.

Definition at line 2860 of file random.h.

4.517.4.3 `template<typename _RealType = double> template<typename _RealType1 , typename _CharT , typename`
`_Traits> std::basic_istream<_CharT, _Traits>& operator>> (std::basic_istream<_CharT, _Traits> & __is,`
`std::chi_squared_distribution<_RealType1> & __x) [friend]`

Extracts a `chi_squared_distribution` random number distribution `__x` from the input stream `__is`.

Parameters

<code>_is</code>	An input stream.
<code>_x</code>	A chi_squared_distribution random number generator engine.

Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following file:

- [random.h](#)

4.518 std::chi_squared_distribution<_RealType>::param_type Struct Reference

Public Types

- typedef [chi_squared_distribution](#)<_RealType> **distribution_type**

Public Member Functions

- **param_type** (_RealType __n=_RealType(1))
- _RealType **n** () const

Friends

- bool **operator==** (const [param_type](#) &__p1, const [param_type](#) &__p2)

4.518.1 Detailed Description

```
template<typename _RealType = double>
struct std::chi_squared_distribution<_RealType>::param_type
```

Parameter type.

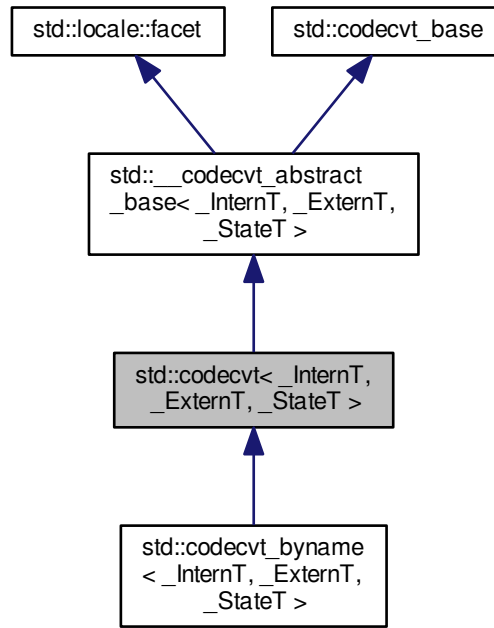
Definition at line 2730 of file random.h.

The documentation for this struct was generated from the following file:

- [random.h](#)

4.519 `std::codecvt<_InternT,_ExternT,_StateT>` Class Template Reference

Inheritance diagram for `std::codecvt<_InternT,_ExternT,_StateT>`:



Public Types

- typedef `_ExternT` **extern_type**
- typedef `_InternT` **intern_type**
- typedef `codecvt_base::result` **result**
- typedef `_StateT` **state_type**

Public Member Functions

- **codecvt** (`size_t __refs=0`)
- **codecvt** (`__c_locale __cloc, size_t __refs=0`)
- bool **always_noconv** () const throw ()
- int **encoding** () const throw ()
- result **in** (`state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *__&__from_next, intern_type *__to, intern_type *__to_end, intern_type *__&__to_next`) const
- int **length** (`state_type &__state, const extern_type *__from, const extern_type *__end, size_t __max`) const
- int **max_length** () const throw ()
- result **out** (`state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *__&__from_next, extern_type *__to, extern_type *__to_end, extern_type *__&__to_next`) const
- result **unshift** (`state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *__&__to_next`) const

Static Public Attributes

- static [locale::id](#) id

Protected Member Functions

- virtual bool **do_always_noconv** () const throw ()
- virtual int **do_encoding** () const throw ()
- virtual result **do_in** (state_type &__state, const extern_type * __from, const extern_type * __from_end, const extern_type * & __from_next, intern_type * __to, intern_type * __to_end, intern_type * & __to_next) const
- virtual int **do_length** (state_type &, const extern_type * __from, const extern_type * __end, size_t __max) const
- virtual int **do_max_length** () const throw ()
- virtual result **do_out** (state_type &__state, const intern_type * __from, const intern_type * __from_end, const intern_type * & __from_next, extern_type * __to, extern_type * __to_end, extern_type * & __to_next) const
- virtual result **do_unshift** (state_type &__state, extern_type * __to, extern_type * __to_end, extern_type * & __to↵__next) const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char * __s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char * __s)

Protected Attributes

- __c_locale **_M_c_locale_codecvt**

4.519.1 Detailed Description

```
template<typename _InternT, typename _ExternT, typename _StateT>
class std::codecvt<_InternT, _ExternT, _StateT >
```

Primary class template codecvt.

NB: Generic, mostly useless implementation.

Definition at line 276 of file codecvt.h.

4.519.2 Member Function Documentation

4.519.2.1 `template<typename _InternT, typename _ExternT, typename _StateT> virtual result std::codecvt< _InternT, _ExternT, _StateT >::do_out (state_type & __state, const intern_type * __from, const intern_type * __from_end, const intern_type * & __from_next, extern_type * __to, extern_type * __to_end, extern_type * & __to_next) const [protected], [virtual]`

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This function is a hook for derived classes to change the value returned.

See also

out for more information.

Implements [std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >](#).

4.519.2.2 `template<typename _InternT, typename _ExternT, typename _StateT> result std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >::in (state_type & __state, const extern_type * __from, const extern_type * __from_end, const extern_type * & __from_next, intern_type * __to, intern_type * __to_end, intern_type * & __to_next) const [inline], [inherited]`

Convert from external to internal character set.

Converts input string of `extern_type` to output string of `intern_type`. This is analogous to `mbsrtowcs`. It does this by calling `codecvt::do_in`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The `state` argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how `state` is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

Parameters

<code>__state</code>	Persistent conversion state data.
<code>__from</code>	Start of input.
<code>__from_end</code>	End of input.
<code>__from_next</code>	Returns start of unconverted data.
<code>__to</code>	Start of output buffer.
<code>__to_end</code>	End of output buffer.
<code>__to_next</code>	Returns start of unused output area.

Returns

codecvt_base::result.

Definition at line 196 of file codecvt.h.

4.519.2.3 template<typename _InternT, typename _ExternT, typename _StateT> result std::__codecvt_abstract_base<_InternT, _ExternT, _StateT>::out (state_type & __state, const intern_type * __from, const intern_type * __from_end, const intern_type * & __from_next, extern_type * __to, extern_type * __to_end, extern_type * & __to_next) const [inline], [inherited]

Convert from internal to external character set.

Converts input string of intern_type to output string of extern_type. This is analogous to wcsrtombs. It does this by calling codecvt::do_out.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from_end) are converted and written to [to,to_end). from_next and to_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from_next and to_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt_base::result. If all the input is converted, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt_base::partial. Otherwise the conversion failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__from	Start of input.
__from_end	End of input.
__from_next	Returns start of unconverted data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

Definition at line 116 of file codecvt.h.

4.519.2.4 template<typename _InternT, typename _ExternT, typename _StateT> result std::__codecvt_abstract_base<_InternT, _ExternT, _StateT>::unshift (state_type & __state, extern_type * __to, extern_type * __to_end, extern_type * & __to_next) const [inline], [inherited]

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling `codecv_t::do_unshift()`.

For example, if 4 external characters always converted to 1 internal character, and input to `in()` had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of `codecv_t_base::result`. If the state could be reset and data written, returns `codecv_t_base::ok`. If no conversion is necessary, returns `codecv_t_base::noconv`. If the output has insufficient space, returns `codecv_t_base::partial`. Otherwise the reset failed and `codecv_t_base::error` is returned.

Parameters

<code>__state</code>	Persistent conversion state data.
<code>__to</code>	Start of output buffer.
<code>__to_end</code>	End of output buffer.
<code>__to_next</code>	Returns start of unused output area.

Returns

`codecv_t_base::result`.

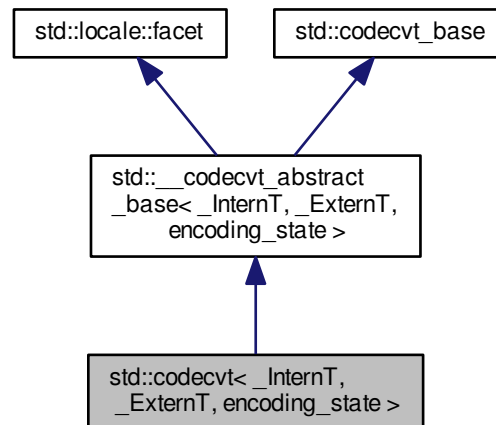
Definition at line 155 of file `codecv_t.h`.

The documentation for this class was generated from the following file:

- [codecv_t.h](#)

4.520 std::codecvt< _InternT, _ExternT, encoding_state > Class Template Reference

Inheritance diagram for std::codecvt< _InternT, _ExternT, encoding_state >:



Public Types

- typedef `state_type::descriptor_type` **descriptor_type**
- typedef `_ExternT` **extern_type**
- typedef `_InternT` **intern_type**
- typedef `codecvt_base::result` **result**
- typedef `__gnu_cxx::encoding_state` **state_type**

Public Member Functions

- **codecvt** (`size_t __refs=0`)
- **codecvt** (`state_type &__enc, size_t __refs=0`)
- **bool always_noconv** () const throw ()
- **int encoding** () const throw ()
- **result in** (`state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *__&__from_next, intern_type *__to, intern_type *__to_end, intern_type *__&__to_next`) const
- **int length** (`state_type &__state, const extern_type *__from, const extern_type *__end, size_t __max`) const
- **int max_length** () const throw ()
- **result out** (`state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *__&__from_next, extern_type *__to, extern_type *__to_end, extern_type *__&__to_next`) const
- **result unshift** (`state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *__&__to_next`) const

Static Public Attributes

- static `locale::id` **id**

Protected Member Functions

- virtual bool **do_always_noconv** () const throw ()
- virtual int **do_encoding** () const throw ()
- virtual result **do_in** (state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *&__from_next, intern_type *__to, intern_type *__to_end, intern_type *&__to_next) const
- virtual int **do_length** (state_type &, const extern_type *__from, const extern_type *__end, size_t __max) const
- virtual int **do_max_length** () const throw ()
- virtual result **do_out** (state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- virtual result **do_unshift** (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

4.520.1 Detailed Description

```
template<typename _InternT, typename _ExternT>
class std::codecvt< _InternT, _ExternT, encoding_state >
```

codecvt<InternT, _ExternT, encoding_state> specialization.

Definition at line 230 of file codecvt_specializations.h.

4.520.2 Member Function Documentation

4.520.2.1 `template<typename _InternT, typename _ExternT > codecvt_base::result std::codecvt< _InternT, _ExternT, encoding_state >::do_out (state_type & __state, const intern_type * __from, const intern_type * __from_end, const intern_type *& __from_next, extern_type * __to, extern_type * __to_end, extern_type *& __to_next) const` [protected], [virtual]

Convert from internal to external character set.

Converts input string of intern_type to output string of extern_type. This function is a hook for derived classes to change the value returned.

See also

out for more information.

Implements `std::__codecvt_abstract_base< _InternT, _ExternT, encoding_state >`.

Definition at line 306 of file codecvt_specializations.h.

References `std::min()`.

4.520.2.2 **result** std::__codecvt_abstract_base<_InternT, _ExternT, encoding_state >::in (state_type & __state, const extern_type * __from, const extern_type * __from_end, const extern_type *& __from_next, intern_type * __to, intern_type * __to_end, intern_type *& __to_next) const [inline],[inherited]

Convert from external to internal character set.

Converts input string of extern_type to output string of intern_type. This is analogous to mbsrtowcs. It does this by calling codecvt::do_in.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from_end) are converted and written to [to,to_end). from_next and to_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from_next and to_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt_base::result. If all the input is converted, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt_base::partial. Otherwise the conversion failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__from	Start of input.
__from_end	End of input.
__from_next	Returns start of unconverted data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

Definition at line 196 of file codecvt.h.

4.520.2.3 **result** std::__codecvt_abstract_base<_InternT, _ExternT, encoding_state >::out (state_type & __state, const intern_type * __from, const intern_type * __from_end, const intern_type *& __from_next, extern_type * __to, extern_type * __to_end, extern_type *& __to_next) const [inline],[inherited]

Convert from internal to external character set.

Converts input string of intern_type to output string of extern_type. This is analogous to wcsrtombs. It does this by calling codecvt::do_out.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

Parameters

<code>__state</code>	Persistent conversion state data.
<code>__from</code>	Start of input.
<code>__from_end</code>	End of input.
<code>__from_next</code>	Returns start of unconverted data.
<code>__to</code>	Start of output buffer.
<code>__to_end</code>	End of output buffer.
<code>__to_next</code>	Returns start of unused output area.

Returns

`codecvt_base::result`.

Definition at line 116 of file `codecvt.h`.

4.520.2.4 `result std::__codecvt_abstract_base<_InternT,_ExternT,encoding_state>::unshift (state_type & __state, extern_type * __to, extern_type * __to_end, extern_type *& __to_next) const` `[inline],[inherited]`

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling `codecvt::do_unshift()`.

For example, if 4 external characters always converted to 1 internal character, and input to `in()` had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of `codecvt_base::result`. If the state could be reset and data written, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the output has insufficient space, returns `codecvt_base::partial`. Otherwise the reset failed and `codecvt_base::error` is returned.

Parameters

<code>__state</code>	Persistent conversion state data.
<code>__to</code>	Start of output buffer.
<code>__to_end</code>	End of output buffer.
<code>__to_next</code>	Returns start of unused output area.

Returns

codecvt_base::result.

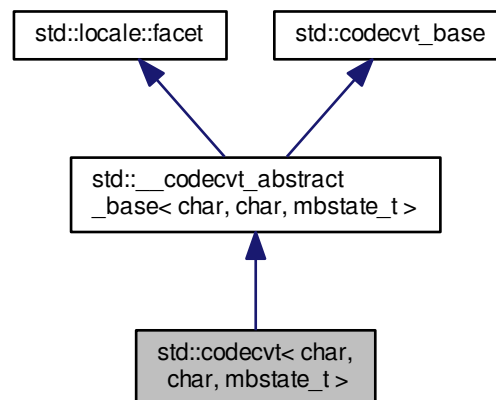
Definition at line 155 of file codecvt.h.

The documentation for this class was generated from the following file:

- [codecvt_specializations.h](#)

4.521 std::codecvt< char, char, mbstate_t > Class Template Reference

Inheritance diagram for std::codecvt< char, char, mbstate_t >:



Public Types

- typedef char **extern_type**
- typedef char **intern_type**
- typedef codecvt_base::result **result**
- typedef mbstate_t **state_type**

Public Member Functions

- **codecvt** (size_t __refs=0)
- **codecvt** (__c_locale __cloc, size_t __refs=0)
- bool **always_noconv** () const throw ()
- int **encoding** () const throw ()
- result **in** (state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *__&__from_next, intern_type *__to, intern_type *__to_end, intern_type *__&__to_next) const
- int **length** (state_type &__state, const extern_type *__from, const extern_type *__end, size_t __max) const
- int **max_length** () const throw ()
- result **out** (state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *__&__from_next, extern_type *__to, extern_type *__to_end, extern_type *__&__to_next) const
- result **unshift** (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *__&__to_next) const

Static Public Attributes

- static [locale::id](#) id

Protected Member Functions

- virtual bool **do_always_noconv** () const throw ()
- virtual int **do_encoding** () const throw ()
- virtual result **do_in** (state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *&__from_next, intern_type *__to, intern_type *__to_end, intern_type *&__to_next) const
- virtual int **do_length** (state_type &, const extern_type *__from, const extern_type *__end, size_t __max) const
- virtual int **do_max_length** () const throw ()
- virtual result [do_out](#) (state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- virtual result **do_unshift** (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *&__to↵__next) const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

Protected Attributes

- __c_locale **_M_c_locale_codecvt**

4.521.1 Detailed Description

```
template<>
class std::codecvt< char, char, mbstate_t >
```

class codecvt<char, char, mbstate_t> specialization.

Definition at line 340 of file codecvt.h.

4.521.2 Member Function Documentation

4.521.2.1 virtual result `std::codecvt< char, char, mbstate_t >::do_out (state_type & __state, const intern_type * __from, const intern_type * __from_end, const intern_type * & __from_next, extern_type * __to, extern_type * __to_end, extern_type * & __to_next) const` `[protected]`, `[virtual]`

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This function is a hook for derived classes to change the value returned.

See also

out for more information.

Implements [std::__codecvt_abstract_base< char, char, mbstate_t >](#).

4.521.2.2 result `std::__codecvt_abstract_base< char, char, mbstate_t >::in (state_type & __state, const extern_type * __from, const extern_type * __from_end, const extern_type * & __from_next, intern_type * __to, intern_type * __to_end, intern_type * & __to_next) const` `[inline]`, `[inherited]`

Convert from external to internal character set.

Converts input string of `extern_type` to output string of `intern_type`. This is analogous to `mbsrtowcs`. It does this by calling `codecvt::do_in`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The `state` argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how `state` is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

Parameters

<code>__state</code>	Persistent conversion state data.
<code>__from</code>	Start of input.
<code>__from_end</code>	End of input.
<code>__from_next</code>	Returns start of unconverted data.
<code>__to</code>	Start of output buffer.
<code>__to_end</code>	End of output buffer.
<code>__to_next</code>	Returns start of unused output area.

Returns

codecvt_base::result.

Definition at line 196 of file codecvt.h.

4.521.2.3 **result** std::__codecvt_abstract_base< char , char , mbstate_t >::out (state_type & __state, const intern_type * __from, const intern_type * __from_end, const intern_type *& __from_next, extern_type * __to, extern_type * __to_end, extern_type *& __to_next) const [inline],[inherited]

Convert from internal to external character set.

Converts input string of intern_type to output string of extern_type. This is analogous to wcsrtombs. It does this by calling codecvt::do_out.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from_end) are converted and written to [to,to_end). from_next and to_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from_next and to_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt_base::result. If all the input is converted, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt_base::partial. Otherwise the conversion failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__from	Start of input.
__from_end	End of input.
__from_next	Returns start of unconverted data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

Definition at line 116 of file codecvt.h.

4.521.2.4 **result** std::__codecvt_abstract_base< char , char , mbstate_t >::unshift (state_type & __state, extern_type * __to, extern_type * __to_end, extern_type *& __to_next) const [inline],[inherited]

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling `codecvt::do_unshift()`.

For example, if 4 external characters always converted to 1 internal character, and input to `in()` had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of `codecvt_base::result`. If the state could be reset and data written, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the output has insufficient space, returns `codecvt_base::partial`. Otherwise the reset failed and `codecvt_base::error` is returned.

Parameters

<code>__state</code>	Persistent conversion state data.
<code>__to</code>	Start of output buffer.
<code>__to_end</code>	End of output buffer.
<code>__to_next</code>	Returns start of unused output area.

Returns

`codecvt_base::result`.

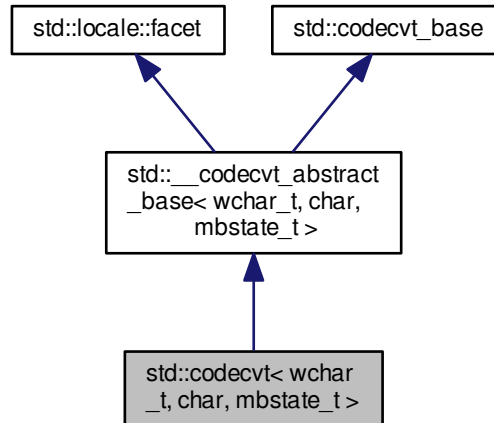
Definition at line 155 of file `codecvt.h`.

The documentation for this class was generated from the following file:

- [codecvt.h](#)

4.522 `std::codecvt< wchar_t, char, mbstate_t >` Class Template Reference

Inheritance diagram for `std::codecvt< wchar_t, char, mbstate_t >`:



Public Types

- typedef char **extern_type**
- typedef wchar_t **intern_type**
- typedef `codecvt_base::result` **result**
- typedef `mbstate_t` **state_type**

Public Member Functions

- **codecvt** (size_t __refs=0)
- **codecvt** (__c_locale __cloc, size_t __refs=0)
- bool **always_noconv** () const throw ()
- int **encoding** () const throw ()
- result **in** (state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *__from_next, intern_type *__to, intern_type *__to_end, intern_type *__to_next) const
- int **length** (state_type &__state, const extern_type *__from, const extern_type *__end, size_t __max) const
- int **max_length** () const throw ()
- result **out** (state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *__from_next, extern_type *__to, extern_type *__to_end, extern_type *__to_next) const
- result **unshift** (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *__to_next) const

Static Public Attributes

- static `locale::id` **id**

Protected Member Functions

- virtual bool **do_always_noconv** () const throw ()
- virtual int **do_encoding** () const throw ()
- virtual result **do_in** (state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *&__from_next, intern_type *__to, intern_type *__to_end, intern_type *&__to_next) const
- virtual int **do_length** (state_type &, const extern_type *__from, const extern_type *__end, size_t __max) const
- virtual int **do_max_length** () const throw ()
- virtual result **do_out** (state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- virtual result **do_unshift** (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

Protected Attributes

- __c_locale **_M_c_locale_codecvt**

4.522.1 Detailed Description

template<>

class std::codecvt< wchar_t, char, mbstate_t >

class codecvt<wchar_t, char, mbstate_t> specialization.

Definition at line 398 of file codecvt.h.

4.522.2 Member Function Documentation

4.522.2.1 virtual result std::codecvt< wchar_t, char, mbstate_t >::do_out (state_type & __state, const intern_type * __from, const intern_type * __from_end, const intern_type *& __from_next, extern_type * __to, extern_type * __to_end, extern_type *& __to_next) const [protected], [virtual]

Convert from internal to external character set.

Converts input string of intern_type to output string of extern_type. This function is a hook for derived classes to change the value returned.

See also

out for more information.

Implements [std::__codecvt_abstract_base< wchar_t, char, mbstate_t >](#).

4.522.2.2 **result** **std::__codecvt_abstract_base**< **wchar_t**, **char**, **mbstate_t** >::**in** (**state_type** & **__state**, **const extern_type** * **__from**, **const extern_type** * **__from_end**, **const extern_type** * & **__from_next**, **intern_type** * **__to**, **intern_type** * **__to_end**, **intern_type** * & **__to_next**) **const** [inline],[inherited]

Convert from external to internal character set.

Converts input string of **extern_type** to output string of **intern_type**. This is analogous to **mbsrtowcs**. It does this by calling **codecvt::do_in**.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [**from**,**from_end**) are converted and written to [**to**,**to_end**). **from_next** and **to_next** are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, **from_next** and **to_next** are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of **codecvt_base::result**. If all the input is converted, returns **codecvt_base::ok**. If no conversion is necessary, returns **codecvt_base::noconv**. If the input ends early or there is insufficient space in the output, returns **codecvt_base::partial**. Otherwise the conversion failed and **codecvt_base::error** is returned.

Parameters

__state	Persistent conversion state data.
__from	Start of input.
__from_end	End of input.
__from_next	Returns start of unconverted data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

Definition at line 196 of file **codecvt.h**.

4.522.2.3 **result** **std::__codecvt_abstract_base**< **wchar_t**, **char**, **mbstate_t** >::**out** (**state_type** & **__state**, **const intern_type** * **__from**, **const intern_type** * **__from_end**, **const intern_type** * & **__from_next**, **extern_type** * **__to**, **extern_type** * **__to_end**, **extern_type** * & **__to_next**) **const** [inline],[inherited]

Convert from internal to external character set.

Converts input string of **intern_type** to output string of **extern_type**. This is analogous to **wcsrtombs**. It does this by calling **codecvt::do_out**.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

Parameters

<code>__state</code>	Persistent conversion state data.
<code>__from</code>	Start of input.
<code>__from_end</code>	End of input.
<code>__from_next</code>	Returns start of unconverted data.
<code>__to</code>	Start of output buffer.
<code>__to_end</code>	End of output buffer.
<code>__to_next</code>	Returns start of unused output area.

Returns

`codecvt_base::result`.

Definition at line 116 of file `codecvt.h`.

4.522.2.4 `result std::__codecvt_abstract_base< wchar_t, char, mbstate_t >::unshift (state_type & __state, extern_type * __to, extern_type * __to_end, extern_type *& __to_next) const` `[inline],[inherited]`

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling `codecvt::do_unshift()`.

For example, if 4 external characters always converted to 1 internal character, and input to `in()` had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of `codecvt_base::result`. If the state could be reset and data written, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the output has insufficient space, returns `codecvt_base::partial`. Otherwise the reset failed and `codecvt_base::error` is returned.

Parameters

<code>__state</code>	Persistent conversion state data.
<code>__to</code>	Start of output buffer.
<code>__to_end</code>	End of output buffer.
<code>__to_next</code>	Returns start of unused output area.

Returns

`codecvt_base::result`.

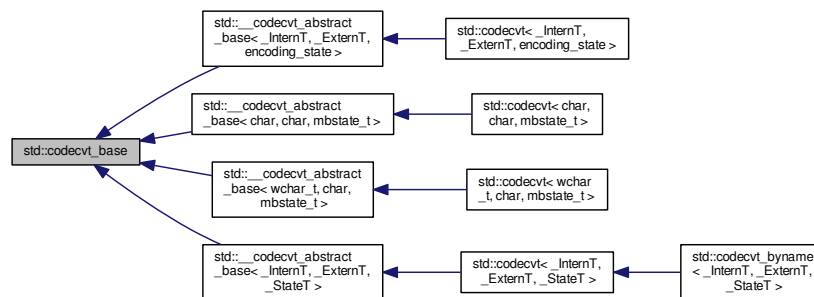
Definition at line 155 of file `codecvt.h`.

The documentation for this class was generated from the following file:

- [codecvt.h](#)

4.523 std::codecvt_base Class Reference

Inheritance diagram for `std::codecvt_base`:

**Public Types**

- enum **result** { **ok**, **partial**, **error**, **noconv** }

4.523.1 Detailed Description

Empty base class for `codecvt` facet [22.2.1.5].

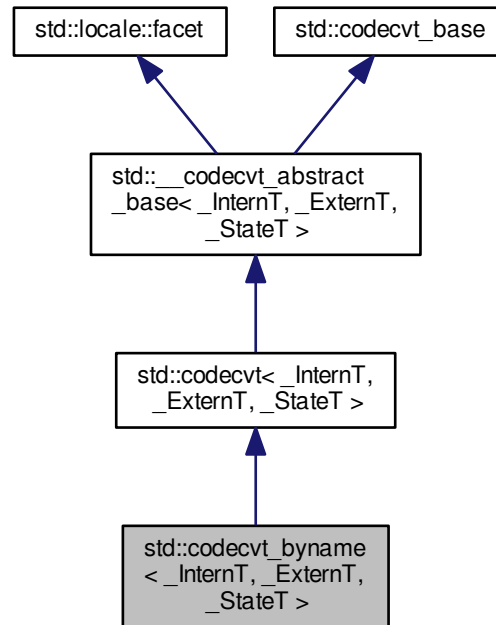
Definition at line 46 of file `codecvt.h`.

The documentation for this class was generated from the following file:

- [codecvt.h](#)

4.524 std::codecvt_byname< _InternT, _ExternT, _StateT > Class Template Reference

Inheritance diagram for std::codecvt_byname< _InternT, _ExternT, _StateT >:



Public Types

- typedef `_ExternT` **extern_type**
- typedef `_InternT` **intern_type**
- typedef `codecvt_base::result` **result**
- typedef `_StateT` **state_type**

Public Member Functions

- **codecvt_byname** (const char *__s, size_t __refs=0)
- bool **always_noconv** () const throw ()
- int **encoding** () const throw ()
- result **in** (state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *__&__from_next, intern_type *__to, intern_type *__to_end, intern_type *__&__to_next) const
- int **length** (state_type &__state, const extern_type *__from, const extern_type *__end, size_t __max) const
- int **max_length** () const throw ()
- result **out** (state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *__&__from_next, extern_type *__to, extern_type *__to_end, extern_type *__&__to_next) const
- result **unshift** (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *__&__to_next) const

Static Public Attributes

- static [locale::id](#)

Protected Member Functions

- virtual bool **do_always_noconv** () const throw ()
- virtual int **do_encoding** () const throw ()
- virtual result **do_in** (state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *&__from_next, intern_type *__to, intern_type *__to_end, intern_type *&__to_next) const
- virtual int **do_length** (state_type &, const extern_type *__from, const extern_type *__end, size_t __max) const
- virtual int **do_max_length** () const throw ()
- virtual result [do_out](#) (state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- virtual result **do_unshift** (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *&__to←__next) const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

Protected Attributes

- __c_locale **_M_c_locale_codecvt**

4.524.1 Detailed Description

```
template<typename _InternT, typename _ExternT, typename _StateT>
class std::codecvt_byname< _InternT, _ExternT, _StateT >
```

class codecvt_byname [22.2.1.6].

Definition at line 458 of file codecvt.h.

4.524.2 Member Function Documentation

4.524.2.1 `template<typename _InternT, typename _ExternT, typename _StateT> virtual result std::codecvt< _InternT, _ExternT, _StateT >::do_out (state_type & __state, const intern_type * __from, const intern_type * __from_end, const intern_type * & __from_next, extern_type * __to, extern_type * __to_end, extern_type * & __to_next) const` [protected], [virtual], [inherited]

Convert from internal to external character set.

Converts input string of intern_type to output string of extern_type. This function is a hook for derived classes to change the value returned.

See also

out for more information.

Implements [std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >](#).

4.524.2.2 `template<typename _InternT, typename _ExternT, typename _StateT> result std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >::in (state_type & __state, const extern_type * __from, const extern_type * __from_end, const extern_type * & __from_next, intern_type * __to, intern_type * __to_end, intern_type * & __to_next) const` [inline], [inherited]

Convert from external to internal character set.

Converts input string of extern_type to output string of intern_type. This is analogous to mbsrtowcs. It does this by calling codecvt::do_in.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from_end) are converted and written to [to,to_end). from_next and to_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from_next and to_next are not affected.

The state argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how state is used.

The result returned is a member of codecvt_base::result. If all the input is converted, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt_base::partial. Otherwise the conversion failed and codecvt_base::error is returned.

Parameters

<code>__state</code>	Persistent conversion state data.
<code>__from</code>	Start of input.
<code>__from_end</code>	End of input.
<code>__from_next</code>	Returns start of unconverted data.
<code>__to</code>	Start of output buffer.
<code>__to_end</code>	End of output buffer.
<code>__to_next</code>	Returns start of unused output area.

Returns

codecvt_base::result.

Definition at line 196 of file codecvt.h.

```
4.524.2.3 template<typename _InternT, typename _ExternT, typename _StateT> result std::__codecvt_abstract_base<
    _InternT, _ExternT, _StateT>::out ( state_type & __state, const intern_type * __from, const intern_type * __from_end,
    const intern_type * & __from_next, extern_type * __to, extern_type * __to_end, extern_type * & __to_next ) const
    [inline], [inherited]
```

Convert from internal to external character set.

Converts input string of intern_type to output string of extern_type. This is analogous to wcsrtombs. It does this by calling codecvt::do_out.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from_end) are converted and written to [to,to_end). from_next and to_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from_next and to_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt_base::result. If all the input is converted, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt_base::partial. Otherwise the conversion failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__from	Start of input.
__from_end	End of input.
__from_next	Returns start of unconverted data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

Definition at line 116 of file codecvt.h.

```
4.524.2.4 template<typename _InternT, typename _ExternT, typename _StateT> result std::__codecvt_abstract_base<
    _InternT, _ExternT, _StateT>::unshift ( state_type & __state, extern_type * __to, extern_type * __to_end, extern_type
    * & __to_next ) const [inline], [inherited]
```

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling `codecvt::do_unshift()`.

For example, if 4 external characters always converted to 1 internal character, and input to `in()` had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of `codecvt_base::result`. If the state could be reset and data written, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the output has insufficient space, returns `codecvt_base::partial`. Otherwise the reset failed and `codecvt_base::error` is returned.

Parameters

<code>__state</code>	Persistent conversion state data.
<code>__to</code>	Start of output buffer.
<code>__to_end</code>	End of output buffer.
<code>__to_next</code>	Returns start of unused output area.

Returns

`codecvt_base::result`.

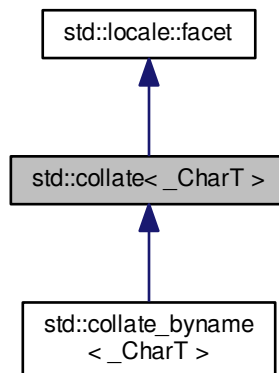
Definition at line 155 of file `codecvt.h`.

The documentation for this class was generated from the following file:

- [codecvt.h](#)

4.525 `std::collate<_CharT>` Class Template Reference

Inheritance diagram for `std::collate<_CharT>`:



Public Types

- typedef `_CharT` `char_type`
- typedef `basic_string<_CharT>` `string_type`

Public Member Functions

- `collate` (`size_t __refs=0`)
- `collate` (`__c_locale __cloc, size_t __refs=0`)
- `int _M_compare` (`const _CharT *`, `const _CharT *`) `const throw ()`
- `template<>`
`int _M_compare` (`const char *`, `const char *`) `const throw()`
- `template<>`
`int _M_compare` (`const wchar_t *`, `const wchar_t *`) `const throw()`
- `size_t _M_transform` (`_CharT *`, `const _CharT *`, `size_t`) `const throw ()`
- `template<>`
`size_t _M_transform` (`char *`, `const char *`, `size_t`) `const throw()`
- `template<>`
`size_t _M_transform` (`wchar_t *`, `const wchar_t *`, `size_t`) `const throw()`
- `int compare` (`const _CharT * __lo1, const _CharT * __hi1, const _CharT * __lo2, const _CharT * __hi2`) `const`
- `long hash` (`const _CharT * __lo, const _CharT * __hi`) `const`
- `string_type transform` (`const _CharT * __lo, const _CharT * __hi`) `const`

Static Public Attributes

- static `locale::id` `id`

Protected Member Functions

- virtual `~collate` ()
- virtual `int do_compare` (`const _CharT * __lo1, const _CharT * __hi1, const _CharT * __lo2, const _CharT * __hi2`) `const`
- virtual `long do_hash` (`const _CharT * __lo, const _CharT * __hi`) `const`
- virtual `string_type do_transform` (`const _CharT * __lo, const _CharT * __hi`) `const`

Static Protected Member Functions

- static `__c_locale _S_clone_c_locale` (`__c_locale & __cloc`) `throw ()`
- static `void _S_create_c_locale` (`__c_locale & __cloc, const char * __s, __c_locale __old=0`)
- static `void _S_destroy_c_locale` (`__c_locale & __cloc`)
- static `__c_locale _S_get_c_locale` ()
- static `const char * _S_get_c_name` () `throw ()`
- static `__c_locale _S_lc_ctype_c_locale` (`__c_locale __cloc, const char * __s`)

Protected Attributes

- `__c_locale _M_c_locale_collate`

4.525.1 Detailed Description

```
template<typename _CharT>
class std::collate<_CharT>
```

Facet for localized string comparison.

This facet encapsulates the code to compare strings in a localized manner.

The collate template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the collate facet.

Definition at line 584 of file locale_classes.h.

4.525.2 Member Typedef Documentation

4.525.2.1 `template<typename _CharT> typedef _CharT std::collate<_CharT>::char_type`

Public typedefs.

Definition at line 590 of file locale_classes.h.

4.525.2.2 `template<typename _CharT> typedef basic_string<_CharT> std::collate<_CharT>::string_type`

Public typedefs.

Definition at line 591 of file locale_classes.h.

4.525.3 Constructor & Destructor Documentation

4.525.3.1 `template<typename _CharT> std::collate<_CharT>::collate (size_t __refs = 0) [inline],[explicit]`

Constructor performs initialization.

This is the constructor provided by the standard.

Parameters

<code>__refs</code>	Passed to the base facet class.
---------------------	---------------------------------

Definition at line 611 of file locale_classes.h.

4.525.3.2 `template<typename _CharT> std::collate<_CharT>::collate (__c_locale __cloc, size_t __refs = 0) [inline],[explicit]`

Internal constructor. Not for general use.

This is a constructor for use by the library itself to set up new locales.

Parameters

<code>__cloc</code>	The C locale.
<code>__refs</code>	Passed to the base facet class.

Definition at line 625 of file `locale_classes.h`.

4.525.3.3 `template<typename _CharT> virtual std::collate<_CharT>::~~collate () [inline], [protected], [virtual]`

Destructor.

Definition at line 688 of file `locale_classes.h`.

4.525.4 Member Function Documentation

4.525.4.1 `template<typename _CharT> int std::collate<_CharT>::compare (const _CharT * __lo1, const _CharT * __hi1, const _CharT * __lo2, const _CharT * __hi2) const [inline]`

Compare two strings.

This function compares two strings and returns the result by calling `collate::do_compare()`.

Parameters

<code>__lo1</code>	Start of string 1.
<code>__hi1</code>	End of string 1.
<code>__lo2</code>	Start of string 2.
<code>__hi2</code>	End of string 2.

Returns

1 if `string1 > string2`, -1 if `string1 < string2`, else 0.

Definition at line 642 of file `locale_classes.h`.

4.525.4.2 `template<typename _CharT> virtual int std::collate<_CharT>::do_compare (const _CharT * __lo1, const _CharT * __hi1, const _CharT * __lo2, const _CharT * __hi2) const [protected], [virtual]`

Compare two strings.

This function is a hook for derived classes to change the value returned.

See also

`compare()`.

Parameters

<code>__lo1</code>	Start of string 1.
<code>__hi1</code>	End of string 1.
<code>__lo2</code>	Start of string 2.
<code>__hi2</code>	End of string 2.

Returns

1 if string1 > string2, -1 if string1 < string2, else 0.

4.525.4.3 `template<typename _CharT> virtual long std::collate<_CharT>::do_hash (const _CharT * __lo, const _CharT * __hi) const` [protected],[virtual]

Return hash of a string.

This function computes and returns a hash on the input string. This function is a hook for derived classes to change the value returned.

Parameters

<code>__↔ __lo</code>	Start of string.
<code>__↔ __hi</code>	End of string.

Returns

Hash value.

4.525.4.4 `template<typename _CharT> virtual string_type std::collate<_CharT>::do_transform (const _CharT * __lo, const _CharT * __hi) const` [protected],[virtual]

Transform string to comparable form.

This function is a hook for derived classes to change the value returned.

Parameters

<code>__↔ __lo</code>	Start.
<code>__↔ __hi</code>	End.

Returns

transformed string.

4.525.4.5 `template<typename _CharT> long std::collate<_CharT>::hash (const _CharT * __lo, const _CharT * __hi) const [inline]`

Return hash of a string.

This function computes and returns a hash on the input string. It does so by returning `collate::do_hash()`.

Parameters

<code>__lo</code>	Start of string.
<code>__hi</code>	End of string.

Returns

Hash value.

Definition at line 675 of file `locale_classes.h`.

4.525.4.6 `template<typename _CharT> string_type std::collate<_CharT>::transform (const _CharT * __lo, const _CharT * __hi) const [inline]`

Transform string to comparable form.

This function is a wrapper for `strxfrm` functionality. It takes the input string and returns a modified string that can be directly compared to other transformed strings. In the C locale, this function just returns a copy of the input string. In some other locales, it may replace two chars with one, change a char for another, etc. It does so by returning `collate::do_transform()`.

Parameters

<code>__lo</code>	Start of string.
<code>__hi</code>	End of string.

Returns

Transformed `string_type`.

Definition at line 661 of file `locale_classes.h`.

4.525.5 Member Data Documentation

4.525.5.1 template<typename _CharT> locale::id std::collate<_CharT>::id [static]

Numpunct facet id.

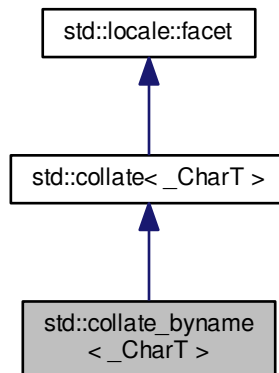
Definition at line 601 of file locale_classes.h.

The documentation for this class was generated from the following file:

- [locale_classes.h](#)

4.526 std::collate_byname<_CharT> Class Template Reference

Inheritance diagram for std::collate_byname<_CharT>:



Public Types

- typedef _CharT [char_type](#)
- typedef [basic_string](#)<_CharT> [string_type](#)

Public Member Functions

- **collate_byname** (const char *__s, size_t __refs=0)
- **int _M_compare** (const _CharT *, const _CharT *) const throw ()
- **template<>**
int _M_compare (const char *, const char *) const throw()
- **template<>**
int _M_compare (const wchar_t *, const wchar_t *) const throw()
- **size_t _M_transform** (_CharT *, const _CharT *, size_t) const throw ()
- **template<>**
size_t _M_transform (char *, const char *, size_t) const throw()
- **template<>**
size_t _M_transform (wchar_t *, const wchar_t *, size_t) const throw()
- **int compare** (const _CharT *__lo1, const _CharT *__hi1, const _CharT *__lo2, const _CharT *__hi2) const
- **long hash** (const _CharT *__lo, const _CharT *__hi) const
- **string_type transform** (const _CharT *__lo, const _CharT *__hi) const

Static Public Attributes

- static **locale::id** id

Protected Member Functions

- **virtual int do_compare** (const _CharT *__lo1, const _CharT *__hi1, const _CharT *__lo2, const _CharT *__hi2) const
- **virtual long do_hash** (const _CharT *__lo, const _CharT *__hi) const
- **virtual string_type do_transform** (const _CharT *__lo, const _CharT *__hi) const

Static Protected Member Functions

- **static __c_locale _S_clone_c_locale** (__c_locale &__cloc) throw ()
- **static void _S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- **static void _S_destroy_c_locale** (__c_locale &__cloc)
- **static __c_locale _S_get_c_locale** ()
- **static const char * _S_get_c_name** () throw ()
- **static __c_locale _S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

Protected Attributes

- **__c_locale _M_c_locale_collate**

4.526.1 Detailed Description

```
template<typename _CharT>
class std::collate_byname<_CharT>
```

class collate_byname [22.2.4.2].

Definition at line 758 of file locale_classes.h.

4.526.2 Member Typedef Documentation

4.526.2.1 `template<typename _CharT> typedef _CharT std::collate_byname<_CharT>::char_type`

Public typedefs.

Definition at line 763 of file locale_classes.h.

4.526.2.2 `template<typename _CharT> typedef basic_string<_CharT> std::collate_byname<_CharT>::string_type`

Public typedefs.

Definition at line 764 of file locale_classes.h.

4.526.3 Member Function Documentation

4.526.3.1 `template<typename _CharT> int std::collate<_CharT>::compare(const _CharT * __lo1, const _CharT * __hi1, const _CharT * __lo2, const _CharT * __hi2) const` [inline], [inherited]

Compare two strings.

This function compares two strings and returns the result by calling `collate::do_compare()`.

Parameters

<code>__lo1</code>	Start of string 1.
<code>__hi1</code>	End of string 1.
<code>__lo2</code>	Start of string 2.
<code>__hi2</code>	End of string 2.

Returns

1 if `string1 > string2`, -1 if `string1 < string2`, else 0.

Definition at line 642 of file locale_classes.h.

4.526.3.2 `template<typename _CharT> virtual int std::collate<_CharT>::do_compare(const _CharT * __lo1, const _CharT * __hi1, const _CharT * __lo2, const _CharT * __hi2) const` [protected], [virtual], [inherited]

Compare two strings.

This function is a hook for derived classes to change the value returned.

See also

`compare()`.

Parameters

<code>__lo1</code>	Start of string 1.
<code>__hi1</code>	End of string 1.
<code>__lo2</code>	Start of string 2.
<code>__hi2</code>	End of string 2.

Returns

1 if `string1 > string2`, -1 if `string1 < string2`, else 0.

4.526.3.3 `template<typename _CharT> virtual long std::collate<_CharT>::do_hash (const _CharT * __lo, const _CharT * __hi) const` [protected],[virtual],[inherited]

Return hash of a string.

This function computes and returns a hash on the input string. This function is a hook for derived classes to change the value returned.

Parameters

<code>__↔ __lo</code>	Start of string.
<code>__↔ __hi</code>	End of string.

Returns

Hash value.

4.526.3.4 `template<typename _CharT> virtual string_type std::collate<_CharT>::do_transform (const _CharT * __lo, const _CharT * __hi) const` [protected],[virtual],[inherited]

Transform string to comparable form.

This function is a hook for derived classes to change the value returned.

Parameters

<code>__↔ __lo</code>	Start.
<code>__↔ __hi</code>	End.

Returns

transformed string.

4.526.3.5 `template<typename _CharT> long std::collate<_CharT>::hash (const _CharT * __lo, const _CharT * __hi) const`
`[inline],[inherited]`

Return hash of a string.

This function computes and returns a hash on the input string. It does so by returning `collate::do_hash()`.

Parameters

<code>__lo</code>	Start of string.
<code>__hi</code>	End of string.

Returns

Hash value.

Definition at line 675 of file `locale_classes.h`.

4.526.3.6 `template<typename _CharT> string_type std::collate<_CharT>::transform (const _CharT * __lo, const _CharT * __hi) const`
`[inline],[inherited]`

Transform string to comparable form.

This function is a wrapper for `strxfrm` functionality. It takes the input string and returns a modified string that can be directly compared to other transformed strings. In the C locale, this function just returns a copy of the input string. In some other locales, it may replace two chars with one, change a char for another, etc. It does so by returning `collate::do_transform()`.

Parameters

<code>__lo</code>	Start of string.
<code>__hi</code>	End of string.

Returns

Transformed `string_type`.

Definition at line 661 of file `locale_classes.h`.

4.526.4 Member Data Documentation

4.526.4.1 `template<typename _CharT> locale::id std::collate<_CharT>::id` `[static], [inherited]`

Numpunct facet id.

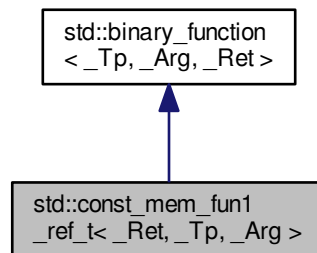
Definition at line 601 of file `locale_classes.h`.

The documentation for this class was generated from the following file:

- [locale_classes.h](#)

4.527 `std::const_mem_fun1_ref_t<_Ret, _Tp, _Arg>` Class Template Reference

Inheritance diagram for `std::const_mem_fun1_ref_t<_Ret, _Tp, _Arg>`:



Public Types

- typedef `_Tp` [first_argument_type](#)
- typedef `_Ret` [result_type](#)
- typedef `_Arg` [second_argument_type](#)

Public Member Functions

- `const_mem_fun1_ref_t` (`_Ret` (`_Tp::*_pf`) (`_Arg`) `const`)
- `_Ret operator()` (`const _Tp &__r`, `_Arg __x`) `const`

4.527.1 Detailed Description

```
template<typename _Ret, typename _Tp, typename _Arg>
class std::const_mem_fun1_ref_t<_Ret, _Tp, _Arg>
```

One of the [adaptors for member pointers](#).

Definition at line 1021 of file `stl_function.h`.

4.527.2 Member Typedef Documentation

4.527.2.1 `typedef _Tp std::binary_function<_Tp, _Arg, _Ret>::first_argument_type` [\[inherited\]](#)

`first_argument_type` is the type of the first argument

Definition at line 121 of file `stl_function.h`.

4.527.2.2 `typedef _Ret std::binary_function<_Tp, _Arg, _Ret>::result_type` [\[inherited\]](#)

`result_type` is the return type

Definition at line 127 of file `stl_function.h`.

4.527.2.3 `typedef _Arg std::binary_function<_Tp, _Arg, _Ret>::second_argument_type` [\[inherited\]](#)

`second_argument_type` is the type of the second argument

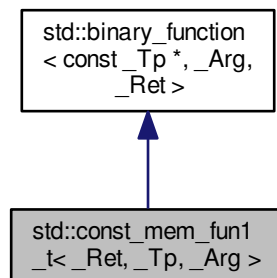
Definition at line 124 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [stl_function.h](#)

4.528 `std::const_mem_fun1_t<_Ret, _Tp, _Arg>` Class Template Reference

Inheritance diagram for `std::const_mem_fun1_t<_Ret, _Tp, _Arg>`:



Public Types

- `typedef const _Tp * first_argument_type`
- `typedef _Ret result_type`
- `typedef _Arg second_argument_type`

Public Member Functions

- `const_mem_fun1_t (_Ret(_Tp::*__pf)(_Arg) const)`
- `_Ret operator() (const _Tp *__p, _Arg __x) const`

4.528.1 Detailed Description

```
template<typename _Ret, typename _Tp, typename _Arg>
class std::const_mem_fun1_t<_Ret, _Tp, _Arg>
```

One of the [adaptors for member pointers](#).

Definition at line 985 of file `stl_function.h`.

4.528.2 Member Typedef Documentation

4.528.2.1 `typedef const _Tp * std::binary_function< const _Tp *, _Arg, _Ret >::first_argument_type` [\[inherited\]](#)

`first_argument_type` is the type of the first argument

Definition at line 121 of file `stl_function.h`.

4.528.2.2 `typedef _Ret std::binary_function< const _Tp *, _Arg, _Ret >::result_type` [\[inherited\]](#)

`result_type` is the return type

Definition at line 127 of file `stl_function.h`.

4.528.2.3 `typedef _Arg std::binary_function< const _Tp *, _Arg, _Ret >::second_argument_type` [\[inherited\]](#)

`second_argument_type` is the type of the second argument

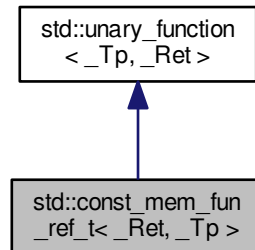
Definition at line 124 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [stl_function.h](#)

4.529 `std::const_mem_fun_ref_t<_Ret, _Tp>` Class Template Reference

Inheritance diagram for `std::const_mem_fun_ref_t<_Ret, _Tp>`:



Public Types

- typedef `_Tp` [argument_type](#)
- typedef `_Ret` [result_type](#)

Public Member Functions

- **`const_mem_fun_ref_t`** (`_Ret` (`_Tp::*__pf`)() const)
- `_Ret` **`operator()`** (const `_Tp` &`__r`) const

4.529.1 Detailed Description

```
template<typename _Ret, typename _Tp>
class std::const_mem_fun_ref_t<_Ret, _Tp>
```

One of the [adaptors for member pointers](#).

Definition at line 949 of file `stl_function.h`.

4.529.2 Member Typedef Documentation

4.529.2.1 typedef `_Tp` `std::unary_function<_Tp, _Ret>::argument_type` [inherited]

`argument_type` is the type of the argument

Definition at line 108 of file `stl_function.h`.

4.529.2.2 `typedef _Ret std::unary_function<_Tp, _Ret>::result_type` [inherited]

`result_type` is the return type

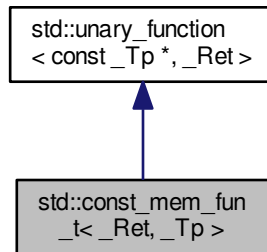
Definition at line 111 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [stl_function.h](#)

4.530 `std::const_mem_fun_t<_Ret, _Tp>` Class Template Reference

Inheritance diagram for `std::const_mem_fun_t<_Ret, _Tp>`:



Public Types

- `typedef const _Tp *` [argument_type](#)
- `typedef _Ret` [result_type](#)

Public Member Functions

- `const_mem_fun_t(_Ret(_Tp::*__pf)()) const`
- `_Ret operator() (const _Tp *__p) const`

4.530.1 Detailed Description

```
template<typename _Ret, typename _Tp>
class std::const_mem_fun_t<_Ret, _Tp>
```

One of the [adaptors for member pointers](#).

Definition at line 913 of file `stl_function.h`.

4.530.2 Member Typedef Documentation

4.530.2.1 `typedef const_Tp * std::unary_function< const_Tp *, _Ret >::argument_type` [inherited]

`argument_type` is the type of the argument

Definition at line 108 of file `stl_function.h`.

4.530.2.2 `typedef _Ret std::unary_function< const_Tp *, _Ret >::result_type` [inherited]

`result_type` is the return type

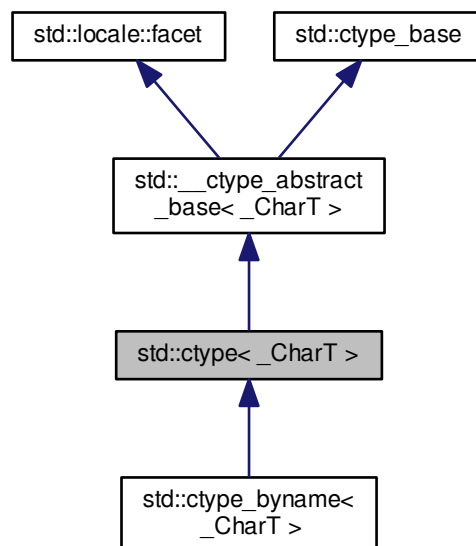
Definition at line 111 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [stl_function.h](#)

4.531 std::ctype<_CharT> Class Template Reference

Inheritance diagram for `std::ctype<_CharT>`:



Public Types

- typedef const int * **__to_type**
- typedef _CharT **char_type**
- typedef **__ctype_abstract_base**< _CharT >::mask **mask**

Public Member Functions

- **ctype** (size_t __refs=0)
- bool **is** (mask __m, **char_type** __c) const
- const **char_type** * **is** (const **char_type** * __lo, const **char_type** * __hi, mask * __vec) const
- char **narrow** (**char_type** __c, char __dfault) const
- const **char_type** * **narrow** (const **char_type** * __lo, const **char_type** * __hi, char __dfault, char * __to) const
- const **char_type** * **scan_is** (mask __m, const **char_type** * __lo, const **char_type** * __hi) const
- const **char_type** * **scan_not** (mask __m, const **char_type** * __lo, const **char_type** * __hi) const
- **char_type** **tolower** (**char_type** __c) const
- const **char_type** * **tolower** (**char_type** * __lo, const **char_type** * __hi) const
- **char_type** **toupper** (**char_type** __c) const
- const **char_type** * **toupper** (**char_type** * __lo, const **char_type** * __hi) const
- **char_type** **widen** (char __c) const
- const char * **widen** (const char * __lo, const char * __hi, **char_type** * __to) const

Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static **locale::id** **id**
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const mask **upper**
- static const mask **xdigit**

Protected Member Functions

- virtual bool **do_is** (mask __m, **char_type** __c) const
- virtual const **char_type** * **do_is** (const **char_type** * __lo, const **char_type** * __hi, mask * __vec) const
- virtual char **do_narrow** (**char_type**, char __dfault) const
- virtual const **char_type** * **do_narrow** (const **char_type** * __lo, const **char_type** * __hi, char __dfault, char * __to) const
- virtual const **char_type** * **do_scan_is** (mask __m, const **char_type** * __lo, const **char_type** * __hi) const
- virtual const **char_type** * **do_scan_not** (mask __m, const **char_type** * __lo, const **char_type** * __hi) const
- virtual **char_type** **do_tolower** (**char_type** __c) const
- virtual const **char_type** * **do_tolower** (**char_type** * __lo, const **char_type** * __hi) const
- virtual **char_type** **do_toupper** (**char_type** __c) const
- virtual const **char_type** * **do_toupper** (**char_type** * __lo, const **char_type** * __hi) const
- virtual **char_type** **do_widen** (char __c) const
- virtual const char * **do_widen** (const char * __lo, const char * __hi, **char_type** * __dest) const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

4.531.1 Detailed Description

```
template<typename _CharT>
class std::ctype<_CharT>
```

Primary class template ctype facet.

This template class defines classification and conversion functions for character sets. It wraps ctype functionality. Ctype gets used by streams for many I/O operations.

This template provides the protected virtual functions the developer will have to replace in a derived class or specialization to make a working facet. The public functions that access them are defined in __ctype_abstract_base, to allow for implementation flexibility. See ctype<wchar_t> for an example. The functions are documented in __ctype_abstract_↵base.

Note: implementations are provided for all the protected virtual functions, but will likely not be useful.

Definition at line 605 of file locale_facets.h.

4.531.2 Member Function Documentation

4.531.2.1 `template<typename _CharT> virtual bool std::ctype<_CharT>::do_is (mask __m, char_type __c) const`
`[protected], [virtual]`

Test char_type classification.

This function finds a mask M for c and compares it to mask m.

do_is() is a hook for a derived facet to change the behavior of classifying. do_is() must always return the same result for the same input.

Parameters

<code>↵ __c</code>	The char_type to find the mask of.
<code>↵ __m</code>	The mask to compare against.

Returns

(M & __m) != 0.

Implements [std::__ctype_abstract_base<_CharT>](#).

4.531.2.2 `template<typename _CharT> virtual const char_type* std::ctype<_CharT>::do_is (const char_type * __lo, const char_type * __hi, mask * __vec) const` `[protected], [virtual]`

Return a mask array.

This function finds the mask for each char_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the input.

do_is() is a hook for a derived facet to change the behavior of classifying. do_is() must always return the same result for the same input.

Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.
<code>__vec</code>	Pointer to an array of mask storage.

Returns

`__hi`.

Implements [std::__ctype_abstract_base<_CharT>](#).

4.531.2.3 `template<typename _CharT> virtual char std::ctype<_CharT>::do_narrow (char_type __c, char __dfault) const` `[protected], [virtual]`

Narrow char_type to char.

This virtual function converts the argument to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead.

do_narrow() is a hook for a derived facet to change the behavior of narrowing. do_narrow() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

<code>__c</code>	The char_type to convert.
<code>__dfault</code>	Char to return if conversion fails.

Returns

The converted char.

Implements [std::__ctype_abstract_base<_CharT>](#).

4.531.2.4 `template<typename _CharT> virtual const char_type* std::ctype<_CharT>::do_narrow (const char_type *
__lo, const char_type * __hi, char __default, char * __to) const [protected], [virtual]`

Narrow char_type array to char.

This virtual function converts each char_type in the range [`__lo`,`__hi`) to char using the simplest reasonable transformation and writes the results to the destination array. For any element in the input that cannot be converted, `__default` is used instead.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.
<code>__default</code>	Char to use if conversion fails.
<code>__to</code>	Pointer to the destination array.

Returns

`__hi`.

Implements [std::__ctype_abstract_base<_CharT>](#).

4.531.2.5 `template<typename _CharT> virtual const char_type* std::ctype<_CharT>::do_scan_is (mask __m, const
char_type * __lo, const char_type * __hi) const [protected], [virtual]`

Find char_type matching mask.

This function searches for and returns the first char_type `c` in [`__lo`,`__hi`) for which `is(__m,c)` is true.

`do_scan_is()` is a hook for a derived facet to change the behavior of match searching. `do_is()` must always return the same result for the same input.

Parameters

<code>__m</code>	The mask to compare against.
<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.

Returns

Pointer to a matching char_type if found, else __hi.

Implements [std::__ctype_abstract_base<_CharT>](#).

4.531.2.6 `template<typename _CharT> virtual const char_type* std::ctype<_CharT>::do_scan_not (mask __m, const char_type * __lo, const char_type * __hi) const` [protected], [virtual]

Find char_type not matching mask.

This function searches for and returns a pointer to the first char_type c of [lo,hi) for which is(m,c) is false.

do_scan_is() is a hook for a derived facet to change the behavior of match searching. do_is() must always return the same result for the same input.

Parameters

<code>__m</code>	The mask to compare against.
<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.

Returns

Pointer to a non-matching char_type if found, else __hi.

Implements [std::__ctype_abstract_base<_CharT>](#).

4.531.2.7 `template<typename _CharT> virtual char_type std::ctype<_CharT>::do_tolower (char_type __c) const` [protected], [virtual]

Convert to lowercase.

This virtual function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

do_tolower() is a hook for a derived facet to change the behavior of lowercasing. do_tolower() must always return the same result for the same input.

Parameters

<code>__c</code>	The char_type to convert.
----------------------------------	---------------------------

Returns

The lowercase char_type if convertible, else __c.

Implements [std::__ctype_abstract_base<_CharT>](#).



4.531.2.8 `template<typename _CharT> virtual const char_type* std::ctype<_CharT>::do_tolower (char_type * __lo, const char_type * __hi) const` [protected], [virtual]

Convert array to lowercase.

This virtual function converts each char_type in the range [__lo,__hi) to lowercase if possible. Other elements remain untouched.

do_tolower() is a hook for a derived facet to change the behavior of lowercasing. do_tolower() must always return the same result for the same input.

Parameters

 __lo	Pointer to start of range.
 __hi	Pointer to end of range.

Returns

[!\[\]\(c444627dab9fee9a1550c053ffaaaae2_img.jpg\) __hi](#).

Implements [std::__ctype_abstract_base<_CharT>](#).


4.531.2.9 `template<typename _CharT> virtual char_type std::ctype<_CharT>::do_toupper (char_type __c) const` [protected], [virtual]

Convert to uppercase.

This virtual function converts the char_type argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

do_toupper() is a hook for a derived facet to change the behavior of uppercasing. do_toupper() must always return the same result for the same input.

Parameters

 __c	The char_type to convert.
---	---------------------------

Returns

The uppercase `char_type` if convertible, else `__c`.

Implements [std::__ctype_abstract_base<_CharT>](#).

```
4.531.2.10 template<typename _CharT> virtual const char_type* std::ctype<_CharT>::do_toupper ( char_type * __lo,
const char_type * __hi ) const [protected], [virtual]
```

Convert array to uppercase.

This virtual function converts each `char_type` in the range `[__lo,__hi)` to uppercase if possible. Other elements remain untouched.

`do_toupper()` is a hook for a derived facet to change the behavior of uppercasing. `do_toupper()` must always return the same result for the same input.

Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.

Returns

`__hi`.

Implements [std::__ctype_abstract_base<_CharT>](#).

```
4.531.2.11 template<typename _CharT> virtual char_type std::ctype<_CharT>::do_widen ( char __c ) const
[protected], [virtual]
```

Widen char.

This virtual function converts the `char` to `char_type` using the simplest reasonable transformation.

`do_widen()` is a hook for a derived facet to change the behavior of widening. `do_widen()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

<code>__c</code>	The char to convert.
------------------	----------------------

Returns

The converted char_type

Implements [std::__ctype_abstract_base<_CharT>](#).

4.531.2.12 `template<typename _CharT> virtual const char* std::ctype<_CharT>::do_widen (const char * __lo, const char * __hi, char_type * __to) const` `[protected],[virtual]`

Widen char array.

This function converts each char in the input to char_type using the simplest reasonable transformation.

do_widen() is a hook for a derived facet to change the behavior of widening. do_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

<code>__lo</code>	Pointer to start range.
<code>__hi</code>	Pointer to end of range.
<code>__to</code>	Pointer to the destination array.

Returns

[`__hi`](#).

Implements [std::__ctype_abstract_base<_CharT>](#).

4.531.2.13 `template<typename _CharT> bool std::__ctype_abstract_base<_CharT>::is (mask __m, char_type __c) const` `[inline],[inherited]`

Test char_type classification.

This function finds a mask M for `__c` and compares it to mask `__m`. It does so by returning the value of `ctype<char_type>::do_is()`.

Parameters

<code>__c</code>	The char_type to compare the mask of.
<code>__m</code>	The mask to compare against.

Returns

(M & __m) != 0.

Definition at line 162 of file locale_facets.h.

Referenced by `std::ctype<char>::is()`, `std::isalnum()`, `std::isalpha()`, `std::iscntrl()`, `std::isdigit()`, `std::isgraph()`, `std::islower()`, `std::isprint()`, `std::ispunct()`, `std::isspace()`, `std::isupper()`, and `std::isxdigit()`.

4.531.2.14 `template<typename _CharT> const char_type* std::__ctype_abstract_base<_CharT>::is(const char_type * __lo, const char_type * __hi, mask * __vec) const` `[inline]`, `[inherited]`

Return a mask array.

This function finds the mask for each `char_type` in the range `[lo,hi)` and successively writes it to `vec`. `vec` must have as many elements as the char array. It does so by returning the value of `ctype<char_type>::do_is()`.

Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.
<code>__vec</code>	Pointer to an array of mask storage.

Returns

`__hi`.

Definition at line 179 of file locale_facets.h.

4.531.2.15 `template<typename _CharT> char std::__ctype_abstract_base<_CharT>::narrow(char_type __c, char __default) const` `[inline]`, `[inherited]`

Narrow `char_type` to `char`.

This function converts the `char_type` to `char` using the simplest reasonable transformation. If the conversion fails, `default` is returned instead. It does so by returning `ctype<char_type>::do_narrow(__c)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

<code>__c</code>	The <code>char_type</code> to convert.
<code>__default</code>	Char to return if conversion fails.

Returns

The converted `char`.

Definition at line 324 of file locale_facets.h.

4.531.2.16 `template<typename _CharT> const char_type* std::__ctype_abstract_base<_CharT>::narrow (const char_type * __lo, const char_type * __hi, char __default, char * __to) const` `[inline],[inherited]`

Narrow array to char array.

This function converts each `char_type` in the input to `char` using the simplest reasonable transformation and writes the results to the destination array. For any `char_type` in the input that cannot be converted, *default* is used instead. It does so by returning `ctype<char_type>::do_narrow(__lo, __hi, __default, __to)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.
<code>__default</code>	Char to use if conversion fails.
<code>__to</code>	Pointer to the destination array.

Returns

`__hi`.

Definition at line 346 of file `locale_facets.h`.

4.531.2.17 `template<typename _CharT> const char_type* std::__ctype_abstract_base<_CharT>::scan_is (mask __m, const char_type * __lo, const char_type * __hi) const` `[inline],[inherited]`

Find `char_type` matching a mask.

This function searches for and returns the first `char_type` `c` in `[lo,hi)` for which `is(m,c)` is true. It does so by returning `ctype<char_type>::do_scan_is()`.

Parameters

<code>__m</code>	The mask to compare against.
<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.

Returns

Pointer to matching `char_type` if found, else `__hi`.

Definition at line 195 of file `locale_facets.h`.

Referenced by `std::ctype< char >::is()`.

4.531.2.18 `template<typename _CharT> const char_type* std::__ctype_abstract_base<_CharT>::scan_not (mask __m, const char_type * __lo, const char_type * __hi) const` `[inline],[inherited]`

Find char_type not matching a mask.

This function searches for and returns the first char_type c in [lo,hi) for which is(m,c) is false. It does so by returning ctype<char_type>::do_scan_not().

Parameters

<code>__m</code>	The mask to compare against.
<code>__lo</code>	Pointer to first char in range.
<code>__hi</code>	Pointer to end of range.

Returns

Pointer to non-matching char if found, else `__hi`.

Definition at line 211 of file locale_facets.h.

Referenced by std::ctype<char>::scan_is().

4.531.2.19 `template<typename _CharT> char_type std::__ctype_abstract_base<_CharT>::tolower (char_type __c) const` `[inline],[inherited]`

Convert to lowercase.

This function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning ctype<char_type>::do_tolower(c).

Parameters

<code>__c</code>	The char_type to convert.
------------------	---------------------------

Returns

The lowercase char_type if convertible, else `__c`.

Definition at line 254 of file locale_facets.h.

Referenced by std::tolower().

4.531.2.20 `template<typename _CharT> const char_type* std::__ctype_abstract_base<_CharT>::tolower (char_type
* __lo, const char_type * __hi) const [inline],[inherited]`

Convert array to lowercase.

This function converts each `char_type` in the range `[__lo,__hi)` to lowercase if possible. Other elements remain untouched. It does so by returning `ctype<char_type>::do_tolower(__lo,__hi)`.

Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.

Returns

`__hi`.

Definition at line 269 of file `locale_facets.h`.

4.531.2.21 `template<typename _CharT> char_type std::__ctype_abstract_base<_CharT>::toupper (char_type __c)
const [inline],[inherited]`

Convert to uppercase.

This function converts the argument to uppercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning `ctype<char_type>::do_toupper()`.

Parameters

<code>__c</code>	The <code>char_type</code> to convert.
------------------	--

Returns

The uppercase `char_type` if convertible, else `__c`.

Definition at line 225 of file `locale_facets.h`.



Referenced by `std::toupper()`.

4.531.2.22 `template<typename _CharT> const char_type* std::__ctype_abstract_base<_CharT>::toupper (char_type
* __lo, const char_type * __hi) const [inline],[inherited]`

Convert array to uppercase.

This function converts each `char_type` in the range `[lo,hi)` to uppercase if possible. Other elements remain untouched. It does so by returning `ctype<char_type>::do_toupper(lo, hi)`.

Parameters

 <code>__lo</code>	Pointer to start of range.
 <code>__hi</code>	Pointer to end of range.

Returns

`__hi`.

Definition at line 240 of file locale_facets.h.


```
4.531.2.23  template<typename _CharT> char_type std::__ctype_abstract_base<_CharT>::widen ( char __c ) const
            [inline],[inherited]
```

Widen char to char_type.

This function converts the char argument to char_type using the simplest reasonable transformation. It does so by returning ctype<char_type>::do_widen(c).

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

 <code>__c</code>	The char to convert.
---	----------------------

Returns

The converted char_type.

Definition at line 286 of file locale_facets.h.




```
4.531.2.24  template<typename _CharT> const char* std::__ctype_abstract_base<_CharT>::widen ( const char * __lo,
            const char * __hi, char_type * __to ) const  [inline],[inherited]
```

Widen array to char_type.

This function converts each char in the input to char_type using the simplest reasonable transformation. It does so by returning ctype<char_type>::do_widen(c).

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

 <code>__lo</code>	Pointer to start of range.
 <code>__hi</code>	Pointer to end of range.
 <code>__to</code>	Pointer to the destination array.

Returns

`__hi`.

Definition at line 305 of file `locale_facets.h`.

4.531.3 Member Data Documentation

4.531.3.1 `template<typename _CharT> locale::id std::ctype<_CharT>::id` `[static]`

The facet id for `ctype<char_type>`

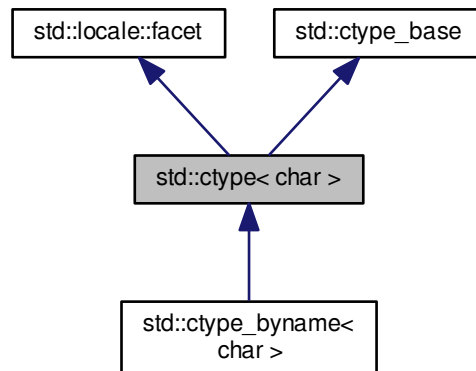
Definition at line 613 of file `locale_facets.h`.

The documentation for this class was generated from the following file:

- [locale_facets.h](#)

4.532 std::ctype< char > Class Template Reference

Inheritance diagram for `std::ctype< char >`:



Public Types

- `typedef const int * __to_type`
- `typedef char char_type`
- `typedef unsigned short mask`

Public Member Functions

- `ctype` (const mask *__table=0, bool __del=false, size_t __refs=0)
- `ctype` (__c_locale __cloc, const mask *__table=0, bool __del=false, size_t __refs=0)
- bool `is` (mask __m, char __c) const
- const char * `is` (const char *__lo, const char *__hi, mask *__vec) const
- char `narrow` (char_type __c, char __dfault) const
- const char_type * `narrow` (const char_type *__lo, const char_type *__hi, char __dfault, char *__to) const
- const char * `scan_is` (mask __m, const char *__lo, const char *__hi) const
- const char * `scan_not` (mask __m, const char *__lo, const char *__hi) const
- const mask * `table` () const throw ()
- char_type `tolower` (char_type __c) const
- const char_type * `tolower` (char_type *__lo, const char_type *__hi) const
- char_type `toupper` (char_type __c) const
- const char_type * `toupper` (char_type *__lo, const char_type *__hi) const
- char_type `widen` (char __c) const
- const char * `widen` (const char *__lo, const char *__hi, char_type *__to) const

Static Public Member Functions

- static const mask * `classic_table` () throw ()

Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static `locale::id` `id`
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const size_t `table_size`
- static const mask **upper**
- static const mask **xdigit**

Protected Member Functions

- virtual `~ctype` ()
- virtual char `do_narrow` (char_type __c, char __dfault) const
- virtual const char_type * `do_narrow` (const char_type *__lo, const char_type *__hi, char __dfault, char *__to) const
- virtual char_type `do_tolower` (char_type __c) const
- virtual const char_type * `do_tolower` (char_type *__lo, const char_type *__hi) const
- virtual char_type `do_toupper` (char_type __c) const
- virtual const char_type * `do_toupper` (char_type *__lo, const char_type *__hi) const
- virtual char_type `do_widen` (char __c) const
- virtual const char * `do_widen` (const char *__lo, const char *__hi, char_type *__to) const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

Protected Attributes

- __c_locale **_M_c_locale_ctype**
- bool **_M_del**
- char **_M_narrow** [1+static_cast< unsigned char >(-1)]
- char **_M_narrow_ok**
- const mask * **_M_table**
- __to_type **_M_tolower**
- __to_type **_M_toupper**
- char **_M_widen** [1+static_cast< unsigned char >(-1)]
- char **_M_widen_ok**

4.532.1 Detailed Description

```
template<>
class std::ctype< char >
```

The ctype<char> specialization.

This class defines classification and conversion functions for the char type. It gets used by char streams for many I/O operations. The char specialization provides a number of optimizations as well.

Definition at line 674 of file locale_facets.h.

4.532.2 Member Typedef Documentation

4.532.2.1 typedef char std::ctype< char >::char_type

Typedef for the template parameter char.

Definition at line 679 of file locale_facets.h.

4.532.3 Constructor & Destructor Documentation

4.532.3.1 std::ctype< char >::ctype (const mask * __table = 0, bool __del = false, size_t __refs = 0) [explicit]

Constructor performs initialization.

This is the constructor provided by the standard.

Parameters

<code>__table</code>	If non-zero, table is used as the per-char mask. Else <code>classic_table()</code> is used.
<code>__del</code>	If true, passes ownership of table to this facet.
<code>__refs</code>	Passed to the base facet class.

4.532.3.2 `std::ctype< char >::ctype (__c_locale __cloc, const mask * __table = 0, bool __del = false, size_t __refs = 0)`
`[explicit]`

Constructor performs static initialization.

This constructor is used to construct the initial C locale facet.

Parameters

<code>__cloc</code>	Handle to C locale data.
<code>__table</code>	If non-zero, table is used as the per-char mask.
<code>__del</code>	If true, passes ownership of table to this facet.
<code>__refs</code>	Passed to the base facet class.

4.532.3.3 `virtual std::ctype< char >::~~ctype ()` `[protected], [virtual]`

Destructor.

This function deletes `table()` if `del` was true in the constructor.

4.532.4 Member Function Documentation

4.532.4.1 `static const mask* std::ctype< char >::classic_table () throw` `[static]`

Returns a pointer to the C locale mask table.

4.532.4.2 `virtual char std::ctype< char >::do_narrow (char_type __c, char __dfault) const` `[inline], [protected], [virtual]`

Narrow char.

This virtual function converts the `char` to `char` using the simplest reasonable transformation. If the conversion fails, `dfault` is returned instead. For an undervied `ctype<char>` facet, `c` will be returned unchanged.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

<code>__c</code>	The char to convert.
<code>__default</code>	Char to return if conversion fails.

Returns

The converted char.

Definition at line 1124 of file locale_facets.h.

4.532.4.3 `virtual const char_type* std::ctype< char >::do_narrow (const char_type * __lo, const char_type * __hi, char __default, char * __to) const` `[inline], [protected], [virtual]`

Narrow char array to char array.

This virtual function converts each char in the range [lo,hi) to char using the simplest reasonable transformation and writes the results to the destination array. For any char in the input that cannot be converted, *default* is used instead. For an underived ctype<char> facet, the argument will be copied unchanged.

do_narrow() is a hook for a derived facet to change the behavior of narrowing. do_narrow() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.
<code>__default</code>	Char to use if conversion fails.
<code>__to</code>	Pointer to the destination array.

Returns

`__hi`.

Definition at line 1150 of file locale_facets.h.

4.532.4.4 `virtual char_type std::ctype< char >::do_tolower (char_type __c) const` `[protected], [virtual]`

Convert to lowercase.

This virtual function converts the char argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

do_tolower() is a hook for a derived facet to change the behavior of lowercasing. do_tolower() must always return the same result for the same input.

Parameters

\leftrightarrow _c	The char to convert.
-------------------------	----------------------

Returns

The lowercase char if convertible, else __c.

4.532.4.5 `virtual const char_type* std::ctype< char >::do_tolower (char_type * __lo, const char_type * __hi) const` `[protected], [virtual]`

Convert array to lowercase.

This virtual function converts each char in the range [lo,hi) to lowercase if possible. Other chars remain untouched.

do_tolower() is a hook for a derived facet to change the behavior of lowercasing. do_tolower() must always return the same result for the same input.

Parameters

\leftrightarrow _lo	Pointer to first char in range.
\leftrightarrow _hi	Pointer to end of range.

Returns

__hi.

4.532.4.6 `virtual char_type std::ctype< char >::do_toupper (char_type __c) const` `[protected], [virtual]`

Convert to uppercase.

This virtual function converts the char argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

do_toupper() is a hook for a derived facet to change the behavior of uppercasing. do_toupper() must always return the same result for the same input.

Parameters

\leftrightarrow _c	The char to convert.
-------------------------	----------------------

Returns

The uppercase char if convertible, else `__c`.

4.532.4.7 `virtual const char_type* std::ctype< char >::do_toupper (char_type * __lo, const char_type * __hi) const` `[protected]`, `[virtual]`

Convert array to uppercase.

This virtual function converts each char in the range `[lo,hi)` to uppercase if possible. Other chars remain untouched.

`do_toupper()` is a hook for a derived facet to change the behavior of uppercasing. `do_toupper()` must always return the same result for the same input.

Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.

Returns

`__hi`.

4.532.4.8 `virtual char_type std::ctype< char >::do_widen (char __c) const` `[inline]`, `[protected]`, `[virtual]`

Widen char.

This virtual function converts the char to char using the simplest reasonable transformation. For an underived `ctype<char>` facet, the argument will be returned unchanged.

`do_widen()` is a hook for a derived facet to change the behavior of widening. `do_widen()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

<code>__c</code>	The char to convert.
------------------	----------------------

Returns

The converted character.

Definition at line 1075 of file `locale_facets.h`.

4.532.4.9 `virtual const char* std::ctype< char >::do_widen (const char * __lo, const char * __hi, char_type * __to) const`
`[inline], [protected], [virtual]`

Widen char array.

This function converts each char in the range [lo,hi) to char using the simplest reasonable transformation. For an undervied ctype<char> facet, the argument will be copied unchanged.

do_widen() is a hook for a derived facet to change the behavior of widening. do_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.
<code>__to</code>	Pointer to the destination array.

Returns

`__hi`.

Definition at line 1098 of file locale_facets.h.

4.532.4.10 `bool std::ctype< char >::is (mask __m, char __c) const` `[inline]`

Test char classification.

This function compares the mask table[c] to `__m`.

Parameters

<code>__c</code>	The char to compare the mask of.
<code>__m</code>	The mask to compare against.

Returns

True if `__m & table[__c]` is true, false otherwise.

Definition at line 43 of file ctype_inline.h.

References `std::__ctype_abstract_base< _CharT >::is()`.

4.532.4.11 `const char * std::ctype< char >::is (const char * __lo, const char * __hi, mask * __vec) const` `[inline]`

Return a mask array.

This function finds the mask for each char in the range [lo, hi) and successively writes it to vec. vec must have as many elements as the char array.

Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.
<code>__vec</code>	Pointer to an array of mask storage.

Returns

`__hi`.

Definition at line 48 of file `ctype_inline.h`.

References `std::__ctype_abstract_base< _CharT >::scan_is()`.

4.532.4.12 `char std::ctype< char >::narrow (char_type __c, char __dfault) const` `[inline]`

Narrow char.

This function converts the char to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead. For an underived ctype<char> facet, c will be returned unchanged.

This function works as if it returns `ctype<char>::do_narrow(c)`. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

<code>__c</code>	The char to convert.
<code>__dfault</code>	Char to return if conversion fails.

Returns

The converted character.

Definition at line 923 of file `locale_facets.h`.

References `std::__ctype_abstract_base< _CharT >::do_narrow()`.

4.532.4.13 `const char_type* std::ctype< char >::narrow (const char_type * __lo, const char_type * __hi, char __default, char * __to) const` `[inline]`

Narrow char array.

This function converts each char in the input to char using the simplest reasonable transformation and writes the results to the destination array. For any char in the input that cannot be converted, *default* is used instead. For an underived `ctype<char>` facet, the argument will be copied unchanged.

This function works as if it returns `ctype<char>::do_narrow(lo, hi, default, to)`. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.
<code>__default</code>	Char to use if conversion fails.
<code>__to</code>	Pointer to the destination array.

Returns

`__hi`.

Definition at line 956 of file `locale_facets.h`.

References `std::__ctype_abstract_base< _CharT >::do_narrow()`.

4.532.4.14 `const char * std::ctype< char >::scan_is (mask __m, const char * __lo, const char * __hi) const` `[inline]`

Find char matching a mask.

This function searches for and returns the first char in `[lo,hi)` for which `is(m,char)` is true.

Parameters

<code>__m</code>	The mask to compare against.
<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.

Returns

Pointer to a matching char if found, else `__hi`.

Definition at line 57 of file `ctype_inline.h`.

References std::__ctype_abstract_base< _CharT >::scan_not().

4.532.4.15 `const char * std::ctype< char >::scan_not (mask __m, const char * __lo, const char * __hi) const` [inline]

Find char not matching a mask.

This function searches for and returns a pointer to the first char in [`__lo`,`__hi`) for which `is(m,char)` is false.

Parameters

<code>↔</code> <code>__m</code>	The mask to compare against.
<code>↔</code> <code>__lo</code>	Pointer to start of range.
<code>↔</code> <code>__hi</code>	Pointer to end of range.

Returns

Pointer to a non-matching char if found, else `__hi`.

Definition at line 67 of file `ctype_inline.h`.

4.532.4.16 `const mask* std::ctype< char >::table () const throw` [inline]

Returns a pointer to the mask table provided to the constructor, or the default from `classic_table()` if none was provided.

Definition at line 974 of file `locale_facets.h`.

References std::__ctype_abstract_base< _CharT >::do_tolower(), and std::__ctype_abstract_base< _CharT >::do_↔
_toupper().

4.532.4.17 `char_type std::ctype< char >::tolower (char_type __c) const` [inline]

Convert to lowercase.

This function converts the char argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

`tolower()` acts as if it returns `ctype<char>::do_tolower(__c)`. `do_tolower()` must always return the same result for the same input.

Parameters

<code>↔</code> <code>__c</code>	The char to convert.
------------------------------------	----------------------

Returns

The lowercase char if convertible, else `__c`.

Definition at line 828 of file `locale_facets.h`.

References `std::__ctype_abstract_base<_CharT>::do_tolower()`.

4.532.4.18 `const char_type* std::ctype<char>::tolower (char_type * __lo, const char_type * __hi) const`
`[inline]`

Convert array to lowercase.

This function converts each char in the range `[lo,hi)` to lowercase if possible. Other chars remain untouched.

`tolower()` acts as if it returns `ctype<char>::do_tolower(__lo, __hi)`. `do_tolower()` must always return the same result for the same input.

Parameters

<code>__lo</code>	Pointer to first char in range.
<code>__hi</code>	Pointer to end of range.

Returns

`__hi`.

Definition at line 845 of file `locale_facets.h`.

References `std::__ctype_abstract_base<_CharT>::do_tolower()`.

4.532.4.19 `char_type std::ctype<char>::toupper (char_type __c) const` `[inline]`

Convert to uppercase.

This function converts the char argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

`toupper()` acts as if it returns `ctype<char>::do_toupper(c)`. `do_toupper()` must always return the same result for the same input.

Parameters

<code>__c</code>	The char to convert.
------------------	----------------------

Returns

The uppercase char if convertible, else `__c`.

Definition at line 795 of file locale_facets.h.

References `std::__ctype_abstract_base<_CharT>::do_toupper()`.

4.532.4.20 `const char_type* std::ctype< char >::toupper (char_type * __lo, const char_type * __hi) const`
`[inline]`

Convert array to uppercase.

This function converts each char in the range `[__lo,__hi)` to uppercase if possible. Other chars remain untouched.

`toupper()` acts as if it returns `ctype<char>::do_toupper(__lo, __hi)`. `do_toupper()` must always return the same result for the same input.

Parameters

<code>__lo</code>	Pointer to first char in range.
<code>__hi</code>	Pointer to end of range.

Returns

`__hi`.

Definition at line 812 of file locale_facets.h.

References `std::__ctype_abstract_base<_CharT>::do_toupper()`.

4.532.4.21 `char_type std::ctype< char >::widen (char __c) const` `[inline]`

Widen char.

This function converts the `char` to `char_type` using the simplest reasonable transformation. For an underived `ctype<char>` facet, the argument will be returned unchanged.

This function works as if it returns `ctype<char>::do_widen(c)`. `do_widen()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

<code>__c</code>	The char to convert.
------------------	----------------------

Returns

The converted character.

Definition at line 865 of file locale_facets.h.

References `std::__ctype_abstract_base<_CharT>::do_widen()`.

4.532.4.22 `const char* std::ctype<char>::widen (const char * __lo, const char * __hi, char_type * __to) const`
`[inline]`

Widen char array.

This function converts each char in the input to char using the simplest reasonable transformation. For an underived `ctype<char>` facet, the argument will be copied unchanged.

This function works as if it returns `ctype<char>::do_widen(c)`. `do_widen()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

<code>__lo</code>	Pointer to first char in range.
<code>__hi</code>	Pointer to end of range.
<code>__to</code>	Pointer to the destination array.

Returns

`__hi`.

Definition at line 892 of file locale_facets.h.

References `std::__ctype_abstract_base<_CharT>::do_widen()`.

4.532.5 Member Data Documentation

4.532.5.1 `locale::id std::ctype<char>::id` `[static]`

The facet id for `ctype<char>`

Definition at line 696 of file locale_facets.h.

4.532.5.2 const size_t std::ctype< char >::table_size [static]

The size of the mask table. It is SCHAR_MAX + 1.

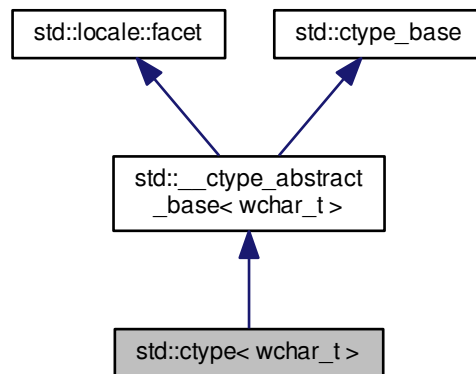
Definition at line 698 of file locale_facets.h.

The documentation for this class was generated from the following files:

- [locale_facets.h](#)
- [ctype_inline.h](#)

4.533 std::ctype< wchar_t > Class Template Reference

Inheritance diagram for std::ctype< wchar_t >:



Public Types

- typedef const int * **__to_type**
- typedef wctype_t **__wmask_type**
- typedef wchar_t **char_type**
- typedef unsigned short **mask**

Public Member Functions

- `ctype` (size_t __refs=0)
- `ctype` (__c_locale __cloc, size_t __refs=0)
- `bool is` (mask __m, `char_type` __c) const
- `const char_type * is` (const `char_type` *__lo, const `char_type` *__hi, mask *__vec) const
- `char narrow` (`char_type` __c, char __dfault) const
- `const char_type * narrow` (const `char_type` *__lo, const `char_type` *__hi, char __dfault, char *__to) const
- `const char_type * scan_is` (mask __m, const `char_type` *__lo, const `char_type` *__hi) const
- `const char_type * scan_not` (mask __m, const `char_type` *__lo, const `char_type` *__hi) const
- `char_type tolower` (`char_type` __c) const
- `const char_type * tolower` (`char_type` *__lo, const `char_type` *__hi) const
- `char_type toupper` (`char_type` __c) const
- `const char_type * toupper` (`char_type` *__lo, const `char_type` *__hi) const
- `char_type widen` (char __c) const
- `const char * widen` (const char *__lo, const char *__hi, `char_type` *__to) const

Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static `locale::id` **id**
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const mask **upper**
- static const mask **xdigit**

Protected Member Functions

- virtual `~ctype` ()
- `__wmask_type _M_convert_to_wmask` (const mask __m) const throw ()
- `void _M_initialize_ctype` () throw ()
- virtual `bool do_is` (mask __m, `char_type` __c) const
- virtual `const char_type * do_is` (const `char_type` *__lo, const `char_type` *__hi, mask *__vec) const
- virtual `char do_narrow` (`char_type` __c, char __dfault) const
- virtual `const char_type * do_narrow` (const `char_type` *__lo, const `char_type` *__hi, char __dfault, char *__to) const
- virtual `const char_type * do_scan_is` (mask __m, const `char_type` *__lo, const `char_type` *__hi) const
- virtual `const char_type * do_scan_not` (mask __m, const `char_type` *__lo, const `char_type` *__hi) const
- virtual `char_type do_tolower` (`char_type` __c) const
- virtual `const char_type * do_tolower` (`char_type` *__lo, const `char_type` *__hi) const
- virtual `char_type do_toupper` (`char_type` __c) const
- virtual `const char_type * do_toupper` (`char_type` *__lo, const `char_type` *__hi) const
- virtual `char_type do_widen` (char __c) const
- virtual `const char * do_widen` (const char *__lo, const char *__hi, `char_type` *__to) const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

Protected Attributes

- mask **_M_bit** [16]
- __c_locale **_M_c_locale_ctype**
- char **_M_narrow** [128]
- bool **_M_narrow_ok**
- wint_t **_M_widen** [1+static_cast< unsigned char >(-1)]
- __wmask_type **_M_wmask** [16]

4.533.1 Detailed Description

```
template<>
class std::ctype< wchar_t >
```

The ctype<wchar_t> specialization.

This class defines classification and conversion functions for the wchar_t type. It gets used by wchar_t streams for many I/O operations. The wchar_t specialization provides a number of optimizations as well.

ctype<wchar_t> inherits its public methods from __ctype_abstract_base<wchar_t>.

Definition at line 1175 of file locale_facets.h.

4.533.2 Member Typedef Documentation

4.533.2.1 typedef wchar_t std::ctype< wchar_t >::char_type

Typedef for the template parameter wchar_t.

Definition at line 1180 of file locale_facets.h.

4.533.3 Constructor & Destructor Documentation

4.533.3.1 std::ctype< wchar_t >::ctype (size_t __refs = 0) [explicit]

Constructor performs initialization.

This is the constructor provided by the standard.

Parameters

<code>__refs</code>	Passed to the base facet class.
---------------------	---------------------------------

4.533.3.2 `std::ctype<wchar_t>::ctype (__c_locale __cloc, size_t __refs = 0)` `[explicit]`

Constructor performs static initialization.

This constructor is used to construct the initial C locale facet.

Parameters

<code>__cloc</code>	Handle to C locale data.
<code>__refs</code>	Passed to the base facet class.

4.533.3.3 `virtual std::ctype<wchar_t>::~~ctype ()` `[protected]`, `[virtual]`

Destructor.

4.533.4 Member Function Documentation

4.533.4.1 `virtual bool std::ctype<wchar_t>::do_is (mask __m, char_type __c) const` `[protected]`, `[virtual]`

Test `wchar_t` classification.

This function finds a mask `M` for `c` and compares it to mask `m`.

`do_is()` is a hook for a derived facet to change the behavior of classifying. `do_is()` must always return the same result for the same input.

Parameters

<code>__c</code>	The <code>wchar_t</code> to find the mask of.
<code>__m</code>	The mask to compare against.

Returns

`(M & __m) != 0.`

Implements [std::__ctype_abstract_base<wchar_t>](#).

4.533.4.2 `virtual const char_type* std::ctype< wchar_t >::do_is (const char_type * __lo, const char_type * __hi, mask * __vec) const` [protected], [virtual]

Return a mask array.

This function finds the mask for each `wchar_t` in the range `[lo,hi)` and successively writes it to `vec`. `vec` must have as many elements as the input.

`do_is()` is a hook for a derived facet to change the behavior of classifying. `do_is()` must always return the same result for the same input.

Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.
<code>__vec</code>	Pointer to an array of mask storage.

Returns

`__hi`.

Implements [std::__ctype_abstract_base< wchar_t >](#).

4.533.4.3 `virtual char std::ctype< wchar_t >::do_narrow (char_type __c, char __dfault) const` [protected], [virtual]

Narrow `wchar_t` to `char`.

This virtual function converts the argument to `char` using the simplest reasonable transformation. If the conversion fails, `dfault` is returned instead. For an underived `ctype<wchar_t>` facet, `c` will be cast to `char` and returned.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

<code>__c</code>	The <code>wchar_t</code> to convert.
<code>__dfault</code>	Char to return if conversion fails.

Returns

The converted `char`.

Implements [std::__ctype_abstract_base< wchar_t >](#).

4.533.4.4 `virtual const char_type* std::ctype<wchar_t>::do_narrow(const char_type * __lo, const char_type * __hi, char __dfault, char * __to) const` [protected], [virtual]

Narrow `wchar_t` array to `char` array.

This virtual function converts each `wchar_t` in the range `[lo,hi)` to `char` using the simplest reasonable transformation and writes the results to the destination array. For any `wchar_t` in the input that cannot be converted, *dfault* is used instead. For an underived `ctype<wchar_t>` facet, the argument will be copied, casting each element to `char`.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.
<code>__dfault</code>	Char to use if conversion fails.
<code>__to</code>	Pointer to the destination array.

Returns

`__hi`.

Implements [std::__ctype_abstract_base<wchar_t>](#).

4.533.4.5 `virtual const char_type* std::ctype<wchar_t>::do_scan_is(mask __m, const char_type * __lo, const char_type * __hi) const` [protected], [virtual]

Find `wchar_t` matching mask.

This function searches for and returns the first `wchar_t` `c` in `[__lo,__hi)` for which `is(__m,c)` is true.

`do_scan_is()` is a hook for a derived facet to change the behavior of match searching. `do_is()` must always return the same result for the same input.

Parameters

<code>__m</code>	The mask to compare against.
<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.

Returns

Pointer to a matching wchar_t if found, else __hi.

Implements [std::__ctype_abstract_base< wchar_t >](#).




4.533.4.6 virtual const char_type* std::ctype< wchar_t >::do_scan_not (mask __m, const char_type * __lo, const char_type * __hi) const [protected], [virtual]

Find wchar_t not matching mask.

This function searches for and returns a pointer to the first wchar_t c of [__lo,__hi) for which is(__m,c) is false.

do_scan_is() is a hook for a derived facet to change the behavior of match searching. do_is() must always return the same result for the same input.

Parameters

 __m	The mask to compare against.
 __lo	Pointer to start of range.
 __hi	Pointer to end of range.

Returns

Pointer to a non-matching wchar_t if found, else __hi.

Implements [std::__ctype_abstract_base< wchar_t >](#).


4.533.4.7 virtual char_type std::ctype< wchar_t >::do_tolower (char_type __c) const [protected], [virtual]

Convert to lowercase.

This virtual function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

do_tolower() is a hook for a derived facet to change the behavior of lowercasing. do_tolower() must always return the same result for the same input.

Parameters

 __c	The wchar_t to convert.
---	-------------------------

Returns

The lowercase `wchar_t` if convertible, else `__c`.

Implements [std::__ctype_abstract_base< wchar_t >](#).

4.533.4.8 `virtual const char_type* std::ctype< wchar_t >::do_tolower (char_type * __lo, const char_type * __hi) const` `[protected], [virtual]`

Convert array to lowercase.

This virtual function converts each `wchar_t` in the range `[lo,hi)` to lowercase if possible. Other elements remain untouched.

`do_tolower()` is a hook for a derived facet to change the behavior of lowercasing. `do_tolower()` must always return the same result for the same input.

Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.

Returns

`__hi`.

Implements [std::__ctype_abstract_base< wchar_t >](#).

4.533.4.9 `virtual char_type std::ctype< wchar_t >::do_toupper (char_type __c) const` `[protected], [virtual]`

Convert to uppercase.

This virtual function converts the `wchar_t` argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

`do_toupper()` is a hook for a derived facet to change the behavior of uppercasing. `do_toupper()` must always return the same result for the same input.

Parameters

<code>__c</code>	The <code>wchar_t</code> to convert.
------------------	--------------------------------------

Returns

The uppercase `wchar_t` if convertible, else `__c`.

Implements [std::__ctype_abstract_base< wchar_t >](#).



4.533.4.10 **virtual const char_type* std::ctype< wchar_t >::do_toupper (char_type * __lo, const char_type * __hi) const** [protected], [virtual]

Convert array to uppercase.

This virtual function converts each wchar_t in the range [lo,hi) to uppercase if possible. Other elements remain untouched.

do_toupper() is a hook for a derived facet to change the behavior of uppercasing. do_toupper() must always return the same result for the same input.

Parameters

 __lo	Pointer to start of range.
 __hi	Pointer to end of range.

Returns

__hi.

Implements [std::__ctype_abstract_base< wchar_t >](#).

4.533.4.11 **virtual char_type std::ctype< wchar_t >::do_widen (char __c) const** [protected], [virtual]


Widen char to wchar_t.

This virtual function converts the char to wchar_t using the simplest reasonable transformation. For an underived ctype<wchar_t> facet, the argument will be cast to wchar_t.

do_widen() is a hook for a derived facet to change the behavior of widening. do_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

 __c	The char to convert.
--	----------------------

Returns

The converted wchar_t.

Implements [std::__ctype_abstract_base< wchar_t >](#).

4.533.4.12 `virtual const char* std::ctype< wchar_t >::do_widen (const char * __lo, const char * __hi, char_type * __to) const` `[protected]`, `[virtual]`

Widen char array to wchar_t array.

This function converts each char in the input to wchar_t using the simplest reasonable transformation. For an underived ctype<wchar_t> facet, the argument will be copied, casting each element to wchar_t.

do_widen() is a hook for a derived facet to change the behavior of widening. do_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

<code>__lo</code>	Pointer to start range.
<code>__hi</code>	Pointer to end of range.
<code>__to</code>	Pointer to the destination array.

Returns

`__hi`.

Implements `std::__ctype_abstract_base< wchar_t >`.

4.533.4.13 `bool std::__ctype_abstract_base< wchar_t >::is (mask __m, char_type __c) const` `[inline]`, `[inherited]`

Test char_type classification.

This function finds a mask M for __c and compares it to mask __m. It does so by returning the value of ctype<char_type>::do_is().

Parameters

<code>__c</code>	The char_type to compare the mask of.
<code>__m</code>	The mask to compare against.

Returns

`(M & __m) != 0`.

Definition at line 162 of file locale_facets.h.

References `std::__ctype_abstract_base< _CharT >::do_is()`.

4.533.4.14 `const char_type* std::__ctype_abstract_base< wchar_t >::is (const char_type * __lo, const char_type * __hi, mask * __vec) const` [inline],[inherited]

Return a mask array.

This function finds the mask for each char_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the char array. It does so by returning the value of ctype<char_type>::do_is().

Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.
<code>__vec</code>	Pointer to an array of mask storage.

Returns

`__hi`.

Definition at line 179 of file locale_facets.h.

References std::__ctype_abstract_base< _CharT >::do_is().

4.533.4.15 `char std::__ctype_abstract_base< wchar_t >::narrow (char_type __c, char __dfault) const` [inline],[inherited]

Narrow char_type to char.

This function converts the char_type to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead. It does so by returning ctype<char_type>::do_narrow(__c).

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

<code>__c</code>	The char_type to convert.
<code>__dfault</code>	Char to return if conversion fails.

Returns

The converted char.

Definition at line 324 of file locale_facets.h.

References std::__ctype_abstract_base< _CharT >::do_narrow().

4.533.4.16 **const char_type* std::__ctype_abstract_base<wchar_t>::narrow (const char_type * __lo, const char_type * __hi, char __default, char * __to) const** [inline],[inherited]

Narrow array to char array.

This function converts each char_type in the input to char using the simplest reasonable transformation and writes the results to the destination array. For any char_type in the input that cannot be converted, *default* is used instead. It does so by returning ctype<char_type>::do_narrow(__lo, __hi, __default, __to).

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.
<code>__default</code>	Char to use if conversion fails.
<code>__to</code>	Pointer to the destination array.

Returns

`__hi`.

Definition at line 346 of file locale_facets.h.

References std::__ctype_abstract_base<_CharT>::do_is(), std::__ctype_abstract_base<_CharT>::do_narrow(), std::__ctype_abstract_base<_CharT>::do_scan_is(), std::__ctype_abstract_base<_CharT>::do_scan_not(), std::__ctype_abstract_base<_CharT>::do_tolower(), std::__ctype_abstract_base<_CharT>::do_toupper(), std::__ctype_abstract_base<_CharT>::do_widen(), and std::locale::facet::facet().

4.533.4.17 **const char_type* std::__ctype_abstract_base<wchar_t>::scan_is (mask __m, const char_type * __lo, const char_type * __hi) const** [inline],[inherited]

Find char_type matching a mask.

This function searches for and returns the first char_type c in [lo,hi) for which is(m,c) is true. It does so by returning ctype<char_type>::do_scan_is().

Parameters

<code>__m</code>	The mask to compare against.
<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.

Returns

Pointer to matching char_type if found, else __hi.

Definition at line 195 of file locale_facets.h.

References std::__ctype_abstract_base<_CharT>::do_scan_is().

4.533.4.18 `const char_type* std::__ctype_abstract_base<wchar_t>::scan_not (mask __m, const char_type * __lo, const char_type * __hi) const` [inline],[inherited]

Find char_type not matching a mask.

This function searches for and returns the first char_type c in [lo,hi) for which is(m,c) is false. It does so by returning ctype<char_type>::do_scan_not().

Parameters

<code>__m</code>	The mask to compare against.
<code>__lo</code>	Pointer to first char in range.
<code>__hi</code>	Pointer to end of range.

Returns

Pointer to non-matching char if found, else __hi.

Definition at line 211 of file locale_facets.h.

References std::__ctype_abstract_base<_CharT>::do_scan_not().

4.533.4.19 `char_type std::__ctype_abstract_base<wchar_t>::tolower (char_type __c) const` [inline],[inherited]

Convert to lowercase.

This function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning ctype<char_type>::do_tolower(c).

Parameters

<code>__c</code>	The char_type to convert.
------------------	---------------------------

Returns

The lowercase `char_type` if convertible, else `__c`.

Definition at line 254 of file `locale_facets.h`.

References `std::__ctype_abstract_base<_CharT>::do_tolower()`.

4.533.4.20 `const char_type* std::__ctype_abstract_base<wchar_t>::tolower (char_type * __lo, const char_type * __hi) const` `[inline]`, `[inherited]`

Convert array to lowercase.

This function converts each `char_type` in the range `[__lo,__hi)` to lowercase if possible. Other elements remain untouched. It does so by returning `ctype<char_type>::do_tolower(__lo, __hi)`.

Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.

Returns

`__hi`.

Definition at line 269 of file `locale_facets.h`.

References `std::__ctype_abstract_base<_CharT>::do_tolower()`.

4.533.4.21 `char_type std::__ctype_abstract_base<wchar_t>::toupper (char_type __c) const` `[inline]`, `[inherited]`

Convert to uppercase.

This function converts the argument to uppercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning `ctype<char_type>::do_toupper()`.

Parameters

<code>__c</code>	The <code>char_type</code> to convert.
------------------	--

Returns

The uppercase `char_type` if convertible, else `__c`.

Definition at line 225 of file `locale_facets.h`.

References std::__ctype_abstract_base< _CharT >::do_toupper().

4.533.4.22 `const char_type* std::__ctype_abstract_base< wchar_t >::toupper (char_type * __lo, const char_type * __hi) const` `[inline], [inherited]`

Convert array to uppercase.

This function converts each char_type in the range [lo,hi) to uppercase if possible. Other elements remain untouched. It does so by returning ctype<char_type>::do_toupper(lo, hi).

Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.

Returns

`__hi`.

Definition at line 240 of file locale_facets.h.

References std::__ctype_abstract_base< _CharT >::do_toupper().

4.533.4.23 `char_type std::__ctype_abstract_base< wchar_t >::widen (char __c) const` `[inline], [inherited]`

Widen char to char_type.

This function converts the char argument to char_type using the simplest reasonable transformation. It does so by returning ctype<char_type>::do_widen(c).

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

<code>__c</code>	The char to convert.
------------------	----------------------

Returns

The converted char_type.

Definition at line 286 of file locale_facets.h.

References std::__ctype_abstract_base< _CharT >::do_widen().

4.533.4.24 `const char* std::__ctype_abstract_base<wchar_t>::widen (const char * __lo, const char * __hi, char_type * __to) const` `[inline],[inherited]`

Widen array to `char_type`.

This function converts each char in the input to `char_type` using the simplest reasonable transformation. It does so by returning `ctype<char_type>::do_widen(c)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.
<code>__to</code>	Pointer to the destination array.

Returns

`__hi`.

Definition at line 305 of file `locale_facets.h`.

References `std::__ctype_abstract_base<_CharT>::do_widen()`.

4.533.5 Member Data Documentation

4.533.5.1 `locale::id std::ctype<wchar_t>::id` `[static]`

The facet id for `ctype<wchar_t>`

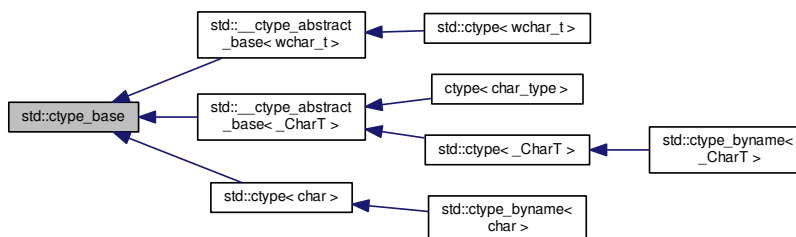
Definition at line 1198 of file `locale_facets.h`.

The documentation for this class was generated from the following file:

- [locale_facets.h](#)

4.534 std::ctype_base Struct Reference

Inheritance diagram for `std::ctype_base`:



Public Types

- typedef const int * **__to_type**
- typedef unsigned short **mask**

Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const mask **upper**
- static const mask **xdigit**

4.534.1 Detailed Description

Base class for ctype.

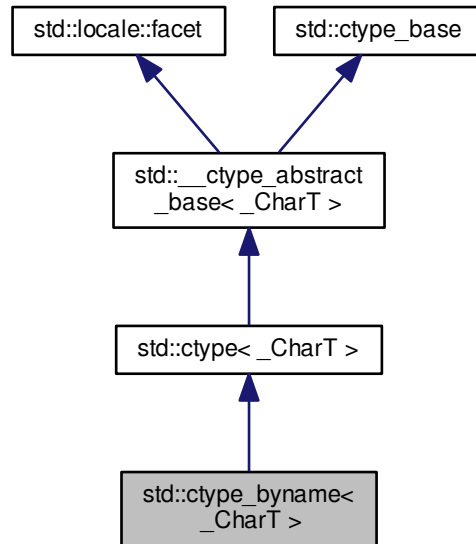
Definition at line 41 of file `ctype_base.h`.

The documentation for this struct was generated from the following file:

- [ctype_base.h](#)

4.535 std::ctype_byname<_CharT> Class Template Reference

Inheritance diagram for std::ctype_byname<_CharT>:



Public Types

- typedef const int * **__to_type**
- typedef _CharT **char_type**
- typedef `ctype<_CharT>::mask` **mask**

Public Member Functions

- **ctype_byname** (const char * __s, size_t __refs=0)
- bool **is** (mask __m, `char_type` __c) const
- const `char_type` * **is** (const `char_type` * __lo, const `char_type` * __hi, mask * __vec) const
- char **narrow** (`char_type` __c, char __default) const
- const `char_type` * **narrow** (const `char_type` * __lo, const `char_type` * __hi, char __default, char * __to) const
- const `char_type` * **scan_is** (mask __m, const `char_type` * __lo, const `char_type` * __hi) const
- const `char_type` * **scan_not** (mask __m, const `char_type` * __lo, const `char_type` * __hi) const
- `char_type` **tolower** (`char_type` __c) const
- const `char_type` * **tolower** (`char_type` * __lo, const `char_type` * __hi) const
- `char_type` **toupper** (`char_type` __c) const
- const `char_type` * **toupper** (`char_type` * __lo, const `char_type` * __hi) const
- `char_type` **widen** (char __c) const
- const char * **widen** (const char * __lo, const char * __hi, `char_type` * __to) const

Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static [locale::id](#) **id**
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const mask **upper**
- static const mask **xdigit**

Protected Member Functions

- virtual bool [do_is](#) (mask __m, [char_type](#) __c) const
- virtual const [char_type](#) * [do_is](#) (const [char_type](#) * __lo, const [char_type](#) * __hi, mask * __vec) const
- virtual [char](#) [do_narrow](#) ([char_type](#), [char](#) __dfault) const
- virtual const [char_type](#) * [do_narrow](#) (const [char_type](#) * __lo, const [char_type](#) * __hi, [char](#) __dfault, [char](#) * __to) const
- virtual const [char_type](#) * [do_scan_is](#) (mask __m, const [char_type](#) * __lo, const [char_type](#) * __hi) const
- virtual const [char_type](#) * [do_scan_not](#) (mask __m, const [char_type](#) * __lo, const [char_type](#) * __hi) const
- virtual [char_type](#) [do_tolower](#) ([char_type](#) __c) const
- virtual const [char_type](#) * [do_tolower](#) ([char_type](#) * __lo, const [char_type](#) * __hi) const
- virtual [char_type](#) [do_toupper](#) ([char_type](#) __c) const
- virtual const [char_type](#) * [do_toupper](#) ([char_type](#) * __lo, const [char_type](#) * __hi) const
- virtual [char_type](#) [do_widen](#) ([char](#) __c) const
- virtual const [char](#) * [do_widen](#) (const [char](#) * __lo, const [char](#) * __hi, [char_type](#) * __dest) const

Static Protected Member Functions

- static [__c_locale](#) [_S_clone_c_locale](#) ([__c_locale](#) & __cloc) throw ()
- static void [_S_create_c_locale](#) ([__c_locale](#) & __cloc, const [char](#) * __s, [__c_locale](#) __old=0)
- static void [_S_destroy_c_locale](#) ([__c_locale](#) & __cloc)
- static [__c_locale](#) [_S_get_c_locale](#) ()
- static const [char](#) * [_S_get_c_name](#) () throw ()
- static [__c_locale](#) [_S_lc_ctype_c_locale](#) ([__c_locale](#) __cloc, const [char](#) * __s)

4.535.1 Detailed Description

```
template<typename _CharT>
class std::ctype_byname<_CharT>
```

class [ctype_byname](#) [22.2.1.2].

Definition at line 1467 of file [locale_facets.h](#).

4.535.2 Member Function Documentation

4.535.2.1 `template<typename _CharT> virtual bool std::ctype<_CharT>::do_is (mask __m, char_type __c) const`
`[protected], [virtual], [inherited]`

Test `char_type` classification.

This function finds a mask `M` for `c` and compares it to mask `m`.

`do_is()` is a hook for a derived facet to change the behavior of classifying. `do_is()` must always return the same result for the same input.

Parameters

<code>__c</code>	The <code>char_type</code> to find the mask of.
<code>__m</code>	The mask to compare against.

Returns

$(M \& \text{__m}) \neq 0$.

Implements [std::__ctype_abstract_base<_CharT>](#).

4.535.2.2 `template<typename _CharT> virtual const char_type* std::ctype<_CharT>::do_is (const char_type * __lo, const char_type * __hi, mask * __vec) const`
`[protected], [virtual], [inherited]`

Return a mask array.

This function finds the mask for each `char_type` in the range `[lo,hi)` and successively writes it to `vec`. `vec` must have as many elements as the input.

`do_is()` is a hook for a derived facet to change the behavior of classifying. `do_is()` must always return the same result for the same input.

Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.
<code>__vec</code>	Pointer to an array of mask storage.

Returns

`__hi`.

Implements [std::__ctype_abstract_base<_CharT>](#).

4.535.2.3 `template<typename _CharT> virtual char std::ctype<_CharT>::do_narrow (char_type __c, char __dfault) const` `[protected]`, `[virtual]`, `[inherited]`

Narrow `char_type` to `char`.

This virtual function converts the argument to `char` using the simplest reasonable transformation. If the conversion fails, `dfault` is returned instead.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

<code>__c</code>	The <code>char_type</code> to convert.
<code>__dfault</code>	Char to return if conversion fails.

Returns

The converted `char`.

Implements [std::__ctype_abstract_base<_CharT>](#).

4.535.2.4 `template<typename _CharT> virtual const char_type* std::ctype<_CharT>::do_narrow (const char_type* __lo, const char_type* __hi, char __dfault, char* __to) const` `[protected]`, `[virtual]`, `[inherited]`

Narrow `char_type` array to `char`.

This virtual function converts each `char_type` in the range `[__lo,__hi)` to `char` using the simplest reasonable transformation and writes the results to the destination array. For any element in the input that cannot be converted, `__dfault` is used instead.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.
<code>__dfault</code>	Char to use if conversion fails.
<code>__to</code>	Pointer to the destination array.

Returns

`__hi`.

Implements [std::__ctype_abstract_base<_CharT>](#).

4.535.2.5 `template<typename _CharT> virtual const char_type* std::ctype<_CharT>::do_scan_is (mask __m, const char_type * __lo, const char_type * __hi) const` [protected],[virtual],[inherited]

Find char_type matching mask.

This function searches for and returns the first char_type c in [__lo,__hi) for which is(__m,c) is true.

do_scan_is() is a hook for a derived facet to change the behavior of match searching. do_is() must always return the same result for the same input.

Parameters

<code>__m</code>	The mask to compare against.
<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.

Returns

Pointer to a matching char_type if found, else __hi.

Implements [std::__ctype_abstract_base<_CharT>](#).

4.535.2.6 `template<typename _CharT> virtual const char_type* std::ctype<_CharT>::do_scan_not (mask __m, const char_type * __lo, const char_type * __hi) const` [protected],[virtual],[inherited]

Find char_type not matching mask.

This function searches for and returns a pointer to the first char_type c of [lo,hi) for which is(m,c) is false.

do_scan_is() is a hook for a derived facet to change the behavior of match searching. do_is() must always return the same result for the same input.

Parameters

<code>__m</code>	The mask to compare against.
<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.

Returns

Pointer to a non-matching char_type if found, else __hi.

Implements [std::__ctype_abstract_base<_CharT>](#).

4.535.2.7 `template<typename _CharT> virtual char_type std::ctype<_CharT>::do_tolower (char_type __c) const`
`[protected], [virtual], [inherited]`

Convert to lowercase.

This virtual function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

do_tolower() is a hook for a derived facet to change the behavior of lowercasing. do_tolower() must always return the same result for the same input.

Parameters

<code>__c</code>	The char_type to convert.
------------------	---------------------------

Returns

The lowercase char_type if convertible, else __c.

Implements [std::__ctype_abstract_base<_CharT>](#).

4.535.2.8 `template<typename _CharT> virtual const char_type* std::ctype<_CharT>::do_tolower (char_type * __lo, const char_type * __hi) const`
`[protected], [virtual], [inherited]`

Convert array to lowercase.

This virtual function converts each char_type in the range [__lo,__hi) to lowercase if possible. Other elements remain untouched.

do_tolower() is a hook for a derived facet to change the behavior of lowercasing. do_tolower() must always return the same result for the same input.

Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.

Returns

`__hi`.

Implements [std::__ctype_abstract_base<_CharT>](#).

4.535.2.9 `template<typename _CharT> virtual char_type std::ctype<_CharT>::do_toupper (char_type __c) const`
`[protected], [virtual], [inherited]`

Convert to uppercase.

This virtual function converts the `char_type` argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

`do_toupper()` is a hook for a derived facet to change the behavior of uppercasing. `do_toupper()` must always return the same result for the same input.

Parameters

<code>__c</code>	The <code>char_type</code> to convert.
------------------	--

Returns

The uppercase `char_type` if convertible, else `__c`.

Implements [std::__ctype_abstract_base<_CharT>](#).

4.535.2.10 `template<typename _CharT> virtual const char_type* std::ctype<_CharT>::do_toupper (char_type * __lo,`
`const char_type * __hi) const` `[protected], [virtual], [inherited]`

Convert array to uppercase.

This virtual function converts each `char_type` in the range `[__lo, __hi)` to uppercase if possible. Other elements remain untouched.

`do_toupper()` is a hook for a derived facet to change the behavior of uppercasing. `do_toupper()` must always return the same result for the same input.

Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.

Returns`__hi`.Implements [std::__ctype_abstract_base<_CharT>](#).

4.535.2.11 `template<typename _CharT> virtual char_type std::ctype<_CharT>::do_widen (char __c) const`
`[protected], [virtual], [inherited]`

Widen char.

This virtual function converts the char to char_type using the simplest reasonable transformation.

do_widen() is a hook for a derived facet to change the behavior of widening. do_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

<code>__c</code>	The char to convert.
------------------	----------------------

Returns

The converted char_type

Implements [std::__ctype_abstract_base<_CharT>](#).

4.535.2.12 `template<typename _CharT> virtual const char* std::ctype<_CharT>::do_widen (const char * __lo, const char * __hi, char_type * __to) const`
`[protected], [virtual], [inherited]`

Widen char array.

This function converts each char in the input to char_type using the simplest reasonable transformation.

do_widen() is a hook for a derived facet to change the behavior of widening. do_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

<code>__lo</code>	Pointer to start range.
<code>__hi</code>	Pointer to end of range.
<code>__to</code>	Pointer to the destination array.

Returns

`__hi`.

Implements `std::__ctype_abstract_base<_CharT>`.

4.535.2.13 `template<typename _CharT> bool std::__ctype_abstract_base<_CharT>::is (mask __m, char_type __c)
const [inline],[inherited]`

Test `char_type` classification.

This function finds a mask `M` for `__c` and compares it to mask `__m`. It does so by returning the value of `ctype<char_type>::do_is()`.

Parameters

<code>__c</code>	The <code>char_type</code> to compare the mask of.
<code>__m</code>	The mask to compare against.

Returns

`(M & __m) != 0`.

Definition at line 162 of file `locale_facets.h`.

Referenced by `std::ctype<char>::is()`, `std::isalnum()`, `std::isalpha()`, `std::iscntrl()`, `std::isdigit()`, `std::isgraph()`, `std::islower()`, `std::isprint()`, `std::ispunct()`, `std::isspace()`, `std::isupper()`, and `std::isxdigit()`.

4.535.2.14 `template<typename _CharT> const char_type* std::__ctype_abstract_base<_CharT>::is (const char_type
* __lo, const char_type * __hi, mask * __vec) const [inline],[inherited]`

Return a mask array.

This function finds the mask for each `char_type` in the range `[lo,hi)` and successively writes it to `vec`. `vec` must have as many elements as the char array. It does so by returning the value of `ctype<char_type>::do_is()`.

Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.
<code>__vec</code>	Pointer to an array of mask storage.

Returns

`__hi`.

Definition at line 179 of file `locale_facets.h`.

4.535.2.15 `template<typename _CharT> char std::__ctype_abstract_base<_CharT>::narrow (char_type __c, char __dfault) const [inline],[inherited]`

Narrow char_type to char.

This function converts the char_type to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead. It does so by returning ctype<char_type>::do_narrow(__c).

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

<code>__c</code>	The char_type to convert.
<code>__dfault</code>	Char to return if conversion fails.

Returns

The converted char.

Definition at line 324 of file locale_facets.h.

4.535.2.16 `template<typename _CharT> const char_type* std::__ctype_abstract_base<_CharT>::narrow (const char_type * __lo, const char_type * __hi, char __dfault, char * __to) const [inline],[inherited]`

Narrow array to char array.

This function converts each char_type in the input to char using the simplest reasonable transformation and writes the results to the destination array. For any char_type in the input that cannot be converted, dfault is used instead. It does so by returning ctype<char_type>::do_narrow(__lo, __hi, __dfault, __to).

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.
<code>__dfault</code>	Char to use if conversion fails.
<code>__to</code>	Pointer to the destination array.

Returns

`__hi`.

Definition at line 346 of file locale_facets.h.

4.535.2.17 `template<typename _CharT> const char_type* std::__ctype_abstract_base<_CharT>::scan_is (mask __m, const char_type * __lo, const char_type * __hi) const [inline],[inherited]`

Find char_type matching a mask.

This function searches for and returns the first `char_type` `c` in `[lo,hi)` for which `is(m,c)` is true. It does so by returning `ctype<char_type>::do_scan_is()`.

Parameters

<code>__m</code>	The mask to compare against.
<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.

Returns

Pointer to matching `char_type` if found, else `__hi`.

Definition at line 195 of file `locale_facets.h`.

Referenced by `std::ctype<char>::is()`.

```
4.535.2.18 template<typename _CharT> const char_type* std::__ctype_abstract_base<_CharT>::scan_not( mask __m,
const char_type * __lo, const char_type * __hi ) const [inline],[inherited]
```

Find `char_type` not matching a mask.

This function searches for and returns the first `char_type` `c` in `[lo,hi)` for which `is(m,c)` is false. It does so by returning `ctype<char_type>::do_scan_not()`.

Parameters

<code>__m</code>	The mask to compare against.
<code>__lo</code>	Pointer to first char in range.
<code>__hi</code>	Pointer to end of range.

Returns

Pointer to non-matching char if found, else `__hi`.

Definition at line 211 of file `locale_facets.h`.

Referenced by `std::ctype<char>::scan_is()`.

```
4.535.2.19 template<typename _CharT> char_type std::__ctype_abstract_base<_CharT>::tolower( char_type __c )
const [inline],[inherited]
```

Convert to lowercase.

This function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning `ctype<char_type>::do_tolower(c)`.

Parameters

<code>__c</code>	The char_type to convert.
------------------	---------------------------

Returns

The lowercase char_type if convertible, else `__c`.

Definition at line 254 of file locale_facets.h.

Referenced by `std::tolower()`.

4.535.2.20 `template<typename _CharT> const char_type* std::__ctype_abstract_base<_CharT>::tolower (char_type * __lo, const char_type * __hi) const [inline],[inherited]`

Convert array to lowercase.

This function converts each char_type in the range `[__lo,__hi)` to lowercase if possible. Other elements remain untouched. It does so by returning `ctype<char_type>::do_tolower(__lo,__hi)`.

Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.

Returns

`__hi`.

Definition at line 269 of file locale_facets.h.

4.535.2.21 `template<typename _CharT> char_type std::__ctype_abstract_base<_CharT>::toupper (char_type __c) const [inline],[inherited]`

Convert to uppercase.

This function converts the argument to uppercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning `ctype<char_type>::do_toupper()`.

Parameters

<code>__c</code>	The char_type to convert.
------------------	---------------------------

Returns

The uppercase char_type if convertible, else __c.

Definition at line 225 of file locale_facets.h.

Referenced by std::toupper().

4.535.2.22 `template<typename _CharT> const char_type* std::__ctype_abstract_base<_CharT>::toupper (char_type *__lo, const char_type *__hi) const [inline],[inherited]`

Convert array to uppercase.

This function converts each char_type in the range [lo,hi) to uppercase if possible. Other elements remain untouched. It does so by returning ctype<char_type>::do_toupper(lo, hi).

Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.

Returns

`__hi`.

Definition at line 240 of file locale_facets.h.

4.535.2.23 `template<typename _CharT> char_type std::__ctype_abstract_base<_CharT>::widen (char __c) const [inline],[inherited]`

Widen char to char_type.

This function converts the char argument to char_type using the simplest reasonable transformation. It does so by returning ctype<char_type>::do_widen(c).

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

<code>__c</code>	The char to convert.
------------------	----------------------

Returns

The converted char_type.

Definition at line 286 of file locale_facets.h.

4.535.2.24 `template<typename _CharT> const char* std::__ctype_abstract_base<_CharT>::widen (const char * __lo, const char * __hi, char_type * __to) const` `[inline], [inherited]`

Widen array to char_type.

This function converts each char in the input to char_type using the simplest reasonable transformation. It does so by returning `ctype<char_type>::do_widen(c)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.
<code>__to</code>	Pointer to the destination array.

Returns

`__hi`.

Definition at line 305 of file `locale_facets.h`.

4.535.3 Member Data Documentation

4.535.3.1 `template<typename _CharT> locale::id std::ctype<_CharT>::id` `[static], [inherited]`

The facet id for `ctype<char_type>`

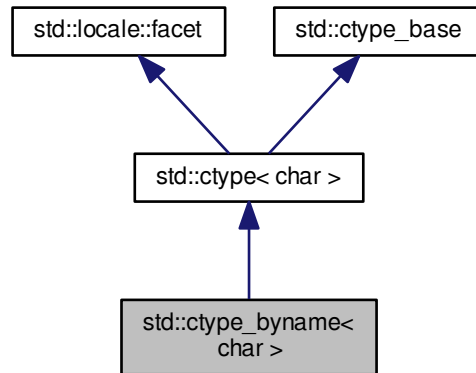
Definition at line 613 of file `locale_facets.h`.

The documentation for this class was generated from the following file:

- [locale_facets.h](#)

4.536 std::ctype_byname< char > Class Template Reference

Inheritance diagram for std::ctype_byname< char >:



Public Types

- typedef const int * **__to_type**
- typedef char [char_type](#)
- typedef unsigned short **mask**

Public Member Functions

- **ctype_byname** (const char * __s, size_t __refs=0)
- bool [is](#) (mask __m, char __c) const
- const char * [is](#) (const char * __lo, const char * __hi, mask * __vec) const
- char [narrow](#) (char_type __c, char __default) const
- const char_type * [narrow](#) (const char_type * __lo, const char_type * __hi, char __default, char * __to) const
- const char * [scan_is](#) (mask __m, const char * __lo, const char * __hi) const
- const char * [scan_not](#) (mask __m, const char * __lo, const char * __hi) const
- const mask * [table](#) () const throw ()
- char_type [tolower](#) (char_type __c) const
- const char_type * [tolower](#) (char_type * __lo, const char_type * __hi) const
- char_type [toupper](#) (char_type __c) const
- const char_type * [toupper](#) (char_type * __lo, const char_type * __hi) const
- char_type [widen](#) (char __c) const
- const char * [widen](#) (const char * __lo, const char * __hi, char_type * __to) const

Static Public Member Functions

- static const mask * [classic_table](#) () throw ()

Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static [locale::id](#) **id**
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const size_t [table_size](#)
- static const mask **upper**
- static const mask **xdigit**

Protected Member Functions

- virtual char [do_narrow](#) (char_type __c, char __dfault) const
- virtual const char_type * [do_narrow](#) (const char_type *__lo, const char_type *__hi, char __dfault, char *__to) const
- virtual char_type [do_tolower](#) (char_type __c) const
- virtual const char_type * [do_tolower](#) (char_type *__lo, const char_type *__hi) const
- virtual char_type [do_toupper](#) (char_type __c) const
- virtual const char_type * [do_toupper](#) (char_type *__lo, const char_type *__hi) const
- virtual char_type [do_widen](#) (char __c) const
- virtual const char * [do_widen](#) (const char *__lo, const char *__hi, char_type *__to) const

Static Protected Member Functions

- static __c_locale [_S_clone_c_locale](#) (__c_locale &__cloc) throw ()
- static void [_S_create_c_locale](#) (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void [_S_destroy_c_locale](#) (__c_locale &__cloc)
- static __c_locale [_S_get_c_locale](#) ()
- static const char * [_S_get_c_name](#) () throw ()
- static __c_locale [_S_lc_ctype_c_locale](#) (__c_locale __cloc, const char *__s)

Protected Attributes

- __c_locale **_M_c_locale_ctype**
- bool **_M_del**
- char **_M_narrow** [1+static_cast< unsigned char >(-1)]
- char **_M_narrow_ok**
- const mask * **_M_table**
- __to_type **_M_tolower**
- __to_type **_M_toupper**
- char **_M_widen** [1+static_cast< unsigned char >(-1)]
- char **_M_widen_ok**

4.536.1 Detailed Description

```
template<>
class std::ctype_byname< char >
```

22.2.1.4 Class ctype_byname specializations.

Definition at line 1482 of file locale_facets.h.

4.536.2 Member Typedef Documentation

4.536.2.1 `typedef char std::ctype< char >::char_type` [inherited]

Typedef for the template parameter char.

Definition at line 679 of file locale_facets.h.

4.536.3 Member Function Documentation

4.536.3.1 `static const mask* std::ctype< char >::classic_table () throw` [static],[inherited]

Returns a pointer to the C locale mask table.

4.536.3.2 `virtual char std::ctype< char >::do_narrow (char_type __c, char __dfault) const` [inline],
[protected],[virtual],[inherited]

Narrow char.

This virtual function converts the char to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead. For an underived ctype<char> facet, c will be returned unchanged.

do_narrow() is a hook for a derived facet to change the behavior of narrowing. do_narrow() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

<code>__c</code>	The char to convert.
<code>__dfault</code>	Char to return if conversion fails.

Returns

The converted char.

Definition at line 1124 of file locale_facets.h.

4.536.3.3 `virtual const char_type* std::ctype< char >::do_narrow (const char_type * __lo, const char_type * __hi, char __default, char * __to) const` `[inline], [protected], [virtual], [inherited]`

Narrow char array to char array.

This virtual function converts each char in the range [lo,hi) to char using the simplest reasonable transformation and writes the results to the destination array. For any char in the input that cannot be converted, *default* is used instead. For an undervied `ctype<char>` facet, the argument will be copied unchanged.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.
<code>__default</code>	Char to use if conversion fails.
<code>__to</code>	Pointer to the destination array.

Returns

`__hi`.

Definition at line 1150 of file `locale_facets.h`.

4.536.3.4 `virtual char_type std::ctype< char >::do_tolower (char_type __c) const` `[protected], [virtual], [inherited]`

Convert to lowercase.

This virtual function converts the char argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

`do_tolower()` is a hook for a derived facet to change the behavior of lowercasing. `do_tolower()` must always return the same result for the same input.

Parameters

<code>__c</code>	The char to convert.
------------------	----------------------

Returns

The lowercase char if convertible, else `__c`.

4.536.3.5 **virtual const char_type* std::ctype< char >::do_toupper (char_type * __lo, const char_type * __hi) const** [protected], [virtual], [inherited]

Convert array to lowercase.

This virtual function converts each char in the range [lo,hi) to lowercase if possible. Other chars remain untouched.

do_toupper() is a hook for a derived facet to change the behavior of lowercasing. do_toupper() must always return the same result for the same input.

Parameters

\leftrightarrow __lo	Pointer to first char in range.
\leftrightarrow __hi	Pointer to end of range.

Returns

__hi.

4.536.3.6 **virtual char_type std::ctype< char >::do_toupper (char_type __c) const** [protected], [virtual], [inherited]

Convert to uppercase.

This virtual function converts the char argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

do_toupper() is a hook for a derived facet to change the behavior of uppercasing. do_toupper() must always return the same result for the same input.

Parameters

\leftrightarrow __c	The char to convert.
--------------------------	----------------------

Returns

The uppercase char if convertible, else __c.

4.536.3.7 **virtual const char_type* std::ctype< char >::do_toupper (char_type * __lo, const char_type * __hi) const** [protected], [virtual], [inherited]

Convert array to uppercase.

This virtual function converts each char in the range [lo,hi) to uppercase if possible. Other chars remain untouched.

do_toupper() is a hook for a derived facet to change the behavior of uppercasing. do_toupper() must always return the same result for the same input.

Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.

Returns

`__hi`.

4.536.3.8 `virtual char_type std::ctype< char >::do_widen (char __c) const` `[inline], [protected], [virtual], [inherited]`

Widen char.

This virtual function converts the char to char using the simplest reasonable transformation. For an underived `ctype<char>` facet, the argument will be returned unchanged.

`do_widen()` is a hook for a derived facet to change the behavior of widening. `do_widen()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

<code>__c</code>	The char to convert.
------------------	----------------------

Returns

The converted character.

Definition at line 1075 of file `locale_facets.h`.

4.536.3.9 `virtual const char* std::ctype< char >::do_widen (const char * __lo, const char * __hi, char_type * __to) const` `[inline], [protected], [virtual], [inherited]`

Widen char array.

This function converts each char in the range `[lo,hi)` to char using the simplest reasonable transformation. For an underived `ctype<char>` facet, the argument will be copied unchanged.

`do_widen()` is a hook for a derived facet to change the behavior of widening. `do_widen()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.
<code>__to</code>	Pointer to the destination array.

Returns

`__hi`.

Definition at line 1098 of file locale_facets.h.

4.536.3.10 `bool std::ctype< char >::is (mask __m, char __c) const` `[inline]`, `[inherited]`

Test char classification.

This function compares the mask table[c] to `__m`.

Parameters

<code>__c</code>	The char to compare the mask of.
<code>__m</code>	The mask to compare against.

Returns

True if `__m & table[__c]` is true, false otherwise.

Definition at line 43 of file ctype_inline.h.

References `std::__ctype_abstract_base< _CharT >::is()`.

4.536.3.11 `const char * std::ctype< char >::is (const char * __lo, const char * __hi, mask * __vec) const` `[inline]`, `[inherited]`

Return a mask array.

This function finds the mask for each char in the range [lo, hi) and successively writes it to vec. vec must have as many elements as the char array.

Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.
<code>__vec</code>	Pointer to an array of mask storage.

Returns

`__hi`.

Definition at line 48 of file `ctype_inline.h`.

References `std::__ctype_abstract_base<_CharT>::scan_is()`.

4.536.3.12 `char std::ctype<char>::narrow (char_type __c, char __dfault) const` `[inline]`, `[inherited]`

Narrow char.

This function converts the char to char using the simplest reasonable transformation. If the conversion fails, *dfault* is returned instead. For an underived `ctype<char>` facet, *c* will be returned unchanged.

This function works as if it returns `ctype<char>::do_narrow(c)`. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

<code>__c</code>	The char to convert.
<code>__dfault</code>	Char to return if conversion fails.

Returns

The converted character.

Definition at line 923 of file `locale_facets.h`.

References `std::__ctype_abstract_base<_CharT>::do_narrow()`.

4.536.3.13 `const char_type* std::ctype<char>::narrow (const char_type * __lo, const char_type * __hi, char __dfault, char * __to) const` `[inline]`, `[inherited]`

Narrow char array.

This function converts each char in the input to char using the simplest reasonable transformation and writes the results to the destination array. For any char in the input that cannot be converted, *dfault* is used instead. For an underived `ctype<char>` facet, the argument will be copied unchanged.

This function works as if it returns `ctype<char>::do_narrow(lo, hi, dfault, to)`. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.
<code>__dfault</code>	Char to use if conversion fails.
<code>__to</code>	Pointer to the destination array.

Returns`__hi`.

Definition at line 956 of file locale_facets.h.

References `std::__ctype_abstract_base<_CharT>::do_narrow()`.

4.536.3.14 `const char * std::ctype< char >::scan_is (mask __m, const char * __lo, const char * __hi) const` `[inline]`,
`[inherited]`

Find char matching a mask.

This function searches for and returns the first char in [lo,hi) for which is(m,char) is true.

Parameters

<code>__m</code>	The mask to compare against.
<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.

ReturnsPointer to a matching char if found, else `__hi`.

Definition at line 57 of file ctype_inline.h.

References `std::__ctype_abstract_base<_CharT>::scan_not()`.

4.536.3.15 `const char * std::ctype< char >::scan_not (mask __m, const char * __lo, const char * __hi) const` `[inline]`,
`[inherited]`

Find char not matching a mask.

This function searches for and returns a pointer to the first char in [__lo,__hi) for which is(m,char) is false.

Parameters

<code>__m</code>	The mask to compare against.
<code>__lo</code>	Pointer to start of range.
<code>__hi</code>	Pointer to end of range.

Returns

Pointer to a non-matching char if found, else `__hi`.

Definition at line 67 of file `ctype_inline.h`.

4.536.3.16 `const mask* std::ctype< char >::table () const throw` `[inline], [inherited]`

Returns a pointer to the mask table provided to the constructor, or the default from `classic_table()` if none was provided.

Definition at line 974 of file `locale_facets.h`.

References `std::__ctype_abstract_base< _CharT >::do_tolower()`, and `std::__ctype_abstract_base< _CharT >::do_↵_toupper()`.

4.536.3.17 `char_type std::ctype< char >::tolower (char_type __c) const` `[inline], [inherited]`

Convert to lowercase.

This function converts the char argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

`tolower()` acts as if it returns `ctype<char>::do_tolower(__c)`. `do_tolower()` must always return the same result for the same input.

Parameters

<code>↵ __c</code>	The char to convert.
------------------------	----------------------

Returns

The lowercase char if convertible, else `__c`.

Definition at line 828 of file `locale_facets.h`.

References `std::__ctype_abstract_base< _CharT >::do_tolower()`.

4.536.3.18 `const char_type* std::ctype< char >::tolower (char_type * __lo, const char_type * __hi) const`
`[inline], [inherited]`

Convert array to lowercase.

This function converts each char in the range `[lo,hi)` to lowercase if possible. Other chars remain untouched.

`tolower()` acts as if it returns `ctype<char>::do_tolower(__lo, __hi)`. `do_tolower()` must always return the same result for the same input.

Parameters

<code>__lo</code>	Pointer to first char in range.
<code>__hi</code>	Pointer to end of range.

Returns

`__hi`.

Definition at line 845 of file locale_facets.h.

References std::__ctype_abstract_base< _CharT >::do_tolower().

4.536.3.19 `char_type std::ctype< char >::toupper (char_type __c) const` [inline],[inherited]

Convert to uppercase.

This function converts the char argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

toupper() acts as if it returns ctype<char>::do_toupper(c). do_toupper() must always return the same result for the same input.

Parameters

<code>__c</code>	The char to convert.
------------------	----------------------

Returns

The uppercase char if convertible, else `__c`.

Definition at line 795 of file locale_facets.h.

References std::__ctype_abstract_base< _CharT >::do_toupper().

4.536.3.20 `const char_type* std::ctype< char >::toupper (char_type * __lo, const char_type * __hi) const` [inline],[inherited]

Convert array to uppercase.

This function converts each char in the range [`__lo`,`__hi`) to uppercase if possible. Other chars remain untouched.

toupper() acts as if it returns ctype<char>:: do_toupper(`__lo`, `__hi`). do_toupper() must always return the same result for the same input.

Parameters

<code>__lo</code>	Pointer to first char in range.
<code>__hi</code>	Pointer to end of range.

Returns

`__hi`.

Definition at line 812 of file `locale_facets.h`.

References `std::__ctype_abstract_base<_CharT>::do_toupper()`.

4.536.3.21 `char_type std::ctype<char>::widen (char __c) const` `[inline],[inherited]`

Widen char.

This function converts the `char` to `char_type` using the simplest reasonable transformation. For an underived `ctype<char>` facet, the argument will be returned unchanged.

This function works as if it returns `ctype<char>::do_widen(c)`. `do_widen()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

<code>__c</code>	The char to convert.
------------------	----------------------

Returns

The converted character.

Definition at line 865 of file `locale_facets.h`.

References `std::__ctype_abstract_base<_CharT>::do_widen()`.

4.536.3.22 `const char* std::ctype<char>::widen (const char * __lo, const char * __hi, char_type * __to) const` `[inline],[inherited]`

Widen char array.

This function converts each `char` in the input to `char` using the simplest reasonable transformation. For an underived `ctype<char>` facet, the argument will be copied unchanged.

This function works as if it returns `ctype<char>::do_widen(c)`. `do_widen()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

Parameters

<code>__lo</code>	Pointer to first char in range.
<code>__hi</code>	Pointer to end of range.
<code>__to</code>	Pointer to the destination array.

Returns

`__hi`.

Definition at line 892 of file `locale_facets.h`.

References `std::__ctype_abstract_base<_CharT>::do_widen()`.

4.536.4 Member Data Documentation

4.536.4.1 `locale::id std::ctype<char>::id` `[static]`, `[inherited]`

The facet id for `ctype<char>`

Definition at line 696 of file `locale_facets.h`.

4.536.4.2 `const size_t std::ctype<char>::table_size` `[static]`, `[inherited]`

The size of the mask table. It is `SCHAR_MAX + 1`.

Definition at line 698 of file `locale_facets.h`.

The documentation for this class was generated from the following file:

- [locale_facets.h](#)

4.537 `std::default_delete<_Tp>` Struct Template Reference

Public Member Functions

- constexpr [default_delete](#) () noexcept=default
- template<typename `_Up` , typename = typename enable_if<is_convertible<_Up*, _Tp*>::value>::type> [default_delete](#) (const [default_delete](#)<_Up> &) noexcept
- void [operator\(\)](#) (_Tp *__ptr) const

4.537.1 Detailed Description

```
template<typename _Tp>
struct std::default_delete< _Tp >
```

Primary template of `default_delete`, used by `unique_ptr`.

Definition at line 54 of file `unique_ptr.h`.

4.537.2 Constructor & Destructor Documentation

4.537.2.1 `template<typename _Tp> constexpr std::default_delete< _Tp >::default_delete () [default], [noexcept]`

Default constructor.

4.537.2.2 `template<typename _Tp> template<typename _Up, typename = typename enable_if<is_convertible<_Up*, _Tp*>::value>::type> std::default_delete< _Tp >::default_delete (const default_delete< _Up > &) [inline], [noexcept]`

Converting constructor.

Allows conversion from a deleter for arrays of another type, `_Up`, only if `_Up*` is convertible to `_Tp*`.

Definition at line 66 of file `unique_ptr.h`.

4.537.3 Member Function Documentation

4.537.3.1 `template<typename _Tp> void std::default_delete< _Tp >::operator() (_Tp * __ptr) const [inline]`

Calls `delete __ptr`.

Definition at line 70 of file `unique_ptr.h`.

Referenced by `std::default_delete< _Tp[]>::operator()()`.

The documentation for this struct was generated from the following file:

- [unique_ptr.h](#)

4.538 `std::default_delete< _Tp[]>` Struct Template Reference

Public Member Functions

- constexpr [default_delete](#) () noexcept=default
- template<typename _Up, typename = typename enable_if<!__is_derived_Tp<_Up>::value>::type> [default_delete](#) (const [default_delete](#)< _Up[]> &) noexcept
- void [operator\(\)](#) (_Tp * __ptr) const
- template<typename _Up > enable_if< __is_derived_Tp< _Up >::value >::type [operator\(\)](#) (_Up *) const =delete

4.538.1 Detailed Description

```
template<typename _Tp>
struct std::default_delete<_Tp[]>
```

Specialization for arrays, default_delete.

Definition at line 84 of file unique_ptr.h.

4.538.2 Constructor & Destructor Documentation

4.538.2.1 `template<typename _Tp> constexpr std::default_delete<_Tp[]>::default_delete () [default],
[noexcept]`

Default constructor.

4.538.2.2 `template<typename _Tp> template<typename _Up, typename = typename enable_if<!
_is_derived<_Tp, _Up>::value>::type> std::default_delete<_Tp[]>::default_delete (const default_delete<_Up[]> &)
[inline], [noexcept]`

Converting constructor.

Allows conversion from a deleter for arrays of another type, such as a const-qualified version of _Tp.

Conversions from types derived from _Tp are not allowed because it is unsafe to delete[] an array of derived types through a pointer to the base type.

Definition at line 111 of file unique_ptr.h.

4.538.3 Member Function Documentation

4.538.3.1 `template<typename _Tp> void std::default_delete<_Tp[]>::operator() (_Tp * __ptr) const [inline]`

Calls delete[] __ptr.

Definition at line 115 of file unique_ptr.h.

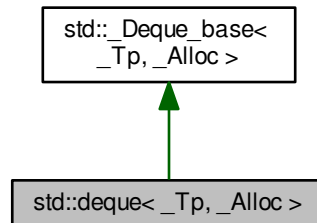
References std::default_delete<_Tp>::operator()().

The documentation for this struct was generated from the following file:

- [unique_ptr.h](#)

4.539 `std::deque<_Tp, _Alloc>` Class Template Reference

Inheritance diagram for `std::deque<_Tp, _Alloc>`:



Public Types

- typedef `_Alloc` **allocator_type**
- typedef `_Base::const_iterator` **const_iterator**
- typedef `_Tp_alloc_type::const_pointer` **const_pointer**
- typedef `_Tp_alloc_type::const_reference` **const_reference**
- typedef `std::reverse_iterator< const_iterator >` **const_reverse_iterator**
- typedef `ptrdiff_t` **difference_type**
- typedef `_Base::iterator` **iterator**
- typedef `_Tp_alloc_type::pointer` **pointer**
- typedef `_Tp_alloc_type::reference` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse_iterator**
- typedef `size_t` **size_type**
- typedef `_Tp` **value_type**

Public Member Functions

- `deque()`
- `deque(const allocator_type &__a)`
- `deque(size_type __n)`
- `deque(size_type __n, const value_type &__value, const allocator_type &__a=allocator_type())`
- `deque(const deque &__x)`
- `deque(deque &&__x)`
- `deque(initializer_list< value_type > __l, const allocator_type &__a=allocator_type())`
- `template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>>`
`deque(_InputIterator __first, _InputIterator __last, const allocator_type &__a=allocator_type())`
- `~deque()` noexcept
- `void assign(size_type __n, const value_type &__val)`
- `template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>>`
`void assign(_InputIterator __first, _InputIterator __last)`

- void [assign](#) (initializer_list< value_type > __l)
- reference [at](#) (size_type __n)
- const_reference [at](#) (size_type __n) const
- reference [back](#) () noexcept
- const_reference [back](#) () const noexcept
- iterator [begin](#) () noexcept
- const_iterator [begin](#) () const noexcept
- const_iterator [cbegin](#) () const noexcept
- const_iterator [cend](#) () const noexcept
- void [clear](#) () noexcept
- const_reverse_iterator [crbegin](#) () const noexcept
- const_reverse_iterator [crend](#) () const noexcept
- template<typename... _Args>
[iterator emplace](#) (const_iterator __position, _Args &&... __args)
- template<typename... _Args>
void [emplace_back](#) (_Args &&... __args)
- template<typename... _Args>
void [emplace_front](#) (_Args &&... __args)
- bool [empty](#) () const noexcept
- iterator [end](#) () noexcept
- const_iterator [end](#) () const noexcept
- iterator [erase](#) (const_iterator __position)
- iterator [erase](#) (const_iterator __first, const_iterator __last)
- reference [front](#) () noexcept
- const_reference [front](#) () const noexcept
- allocator_type [get_allocator](#) () const noexcept
- iterator [insert](#) (const_iterator __position, const value_type & __x)
- iterator [insert](#) (const_iterator __position, value_type && __x)
- iterator [insert](#) (const_iterator __p, initializer_list< value_type > __l)
- iterator [insert](#) (const_iterator __position, size_type __n, const value_type & __x)
- template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>>
[iterator insert](#) (const_iterator __position, _InputIterator __first, _InputIterator __last)
- size_type [max_size](#) () const noexcept
- deque & [operator=](#) (const deque & __x)
- deque & [operator=](#) (deque && __x) noexcept
- deque & [operator=](#) (initializer_list< value_type > __l)
- reference [operator\[\]](#) (size_type __n) noexcept
- const_reference [operator\[\]](#) (size_type __n) const noexcept
- void [pop_back](#) () noexcept
- void [pop_front](#) () noexcept
- void [push_back](#) (const value_type & __x)
- void [push_back](#) (value_type && __x)
- void [push_front](#) (const value_type & __x)
- void [push_front](#) (value_type && __x)
- reverse_iterator [rbegin](#) () noexcept
- const_reverse_iterator [rbegin](#) () const noexcept
- reverse_iterator [rend](#) () noexcept
- const_reverse_iterator [rend](#) () const noexcept
- void [resize](#) (size_type __new_size)
- void [resize](#) (size_type __new_size, const value_type & __x)
- void [shrink_to_fit](#) () noexcept
- size_type [size](#) () const noexcept
- void [swap](#) (deque & __x) noexcept

Protected Types

- enum { **_S_initial_map_size** }
- typedef `_Alloc::template rebind< _Tp * >::other` **_Map_alloc_type**
- typedef pointer * **_Map_pointer**

Protected Member Functions

- `_Tp ** _M_allocate_map (size_t __n)`
- `_Tp * _M_allocate_node ()`
- `template<typename _InputIterator >`
`void _M_assign_aux (_InputIterator __first, _InputIterator __last, std::input_iterator_tag)`
- `template<typename _ForwardIterator >`
`void _M_assign_aux (_ForwardIterator __first, _ForwardIterator __last, std::forward_iterator_tag)`
- `template<typename _Integer >`
`void _M_assign_dispatch (_Integer __n, _Integer __val, __true_type)`
- `template<typename _InputIterator >`
`void _M_assign_dispatch (_InputIterator __first, _InputIterator __last, __false_type)`
- `void _M_create_nodes (_Tp ** __nstart, _Tp ** __nfinish)`
- `void _M_deallocate_map (_Tp ** __p, size_t __n) noexcept`
- `void _M_deallocate_node (_Tp * __p) noexcept`
- `void _M_default_append (size_type __n)`
- `void _M_default_initialize ()`
- `template<typename _Alloc1 >`
`void _M_destroy_data (iterator __first, iterator __last, const _Alloc1 &)`
- `void _M_destroy_data (iterator __first, iterator __last, const std::allocator< _Tp > &)`
- `void _M_destroy_data_aux (iterator __first, iterator __last)`
- `void _M_destroy_nodes (_Tp ** __nstart, _Tp ** __nfinish) noexcept`
- `iterator _M_erase (iterator __pos)`
- `iterator _M_erase (iterator __first, iterator __last)`
- `void _M_erase_at_begin (iterator __pos)`
- `void _M_erase_at_end (iterator __pos)`
- `void _M_fill_assign (size_type __n, const value_type & __val)`
- `void _M_fill_initialize (const value_type & __value)`
- `void _M_fill_insert (iterator __pos, size_type __n, const value_type & __x)`
- `_Map_alloc_type _M_get_map_allocator () const noexcept`
- `_Tp_alloc_type & _M_get_Tp_allocator () noexcept`
- `const _Tp_alloc_type & _M_get_Tp_allocator () const noexcept`
- `template<typename _Integer >`
`void _M_initialize_dispatch (_Integer __n, _Integer __x, __true_type)`
- `template<typename _InputIterator >`
`void _M_initialize_dispatch (_InputIterator __first, _InputIterator __last, __false_type)`
- `void _M_initialize_map (size_t)`
- `template<typename... _Args>`
`iterator _M_insert_aux (iterator __pos, _Args &&... __args)`
- `void _M_insert_aux (iterator __pos, size_type __n, const value_type & __x)`
- `template<typename _ForwardIterator >`
`void _M_insert_aux (iterator __pos, _ForwardIterator __first, _ForwardIterator __last, size_type __n)`
- `template<typename _Integer >`
`void _M_insert_dispatch (iterator __pos, _Integer __n, _Integer __x, __true_type)`

- `template<typename _InputIterator >`
`void _M_insert_dispatch (iterator __pos, _InputIterator __first, _InputIterator __last, __false_type)`
- `void _M_range_check (size_type __n) const`
- `template<typename _InputIterator >`
`void _M_range_insert_aux (iterator __pos, _InputIterator __first, _InputIterator __last, std::input_iterator_tag)`
- `template<typename _ForwardIterator >`
`void _M_range_insert_aux (iterator __pos, _ForwardIterator __first, _ForwardIterator __last, std::forward_iterator_tag)`
- `bool _M_shrink_to_fit ()`

- `template<typename _InputIterator >`
`void _M_range_initialize (_InputIterator __first, _InputIterator __last, std::input_iterator_tag)`
- `template<typename _ForwardIterator >`
`void _M_range_initialize (_ForwardIterator __first, _ForwardIterator __last, std::forward_iterator_tag)`

- `template<typename... _Args>`
`void _M_push_back_aux (_Args &&... __args)`
- `template<typename... _Args>`
`void _M_push_front_aux (_Args &&... __args)`
- `void _M_pop_back_aux ()`
- `void _M_pop_front_aux ()`

- `iterator _M_reserve_elements_at_front (size_type __n)`
- `iterator _M_reserve_elements_at_back (size_type __n)`
- `void _M_new_elements_at_front (size_type __new_elements)`
- `void _M_new_elements_at_back (size_type __new_elements)`

- `void _M_reserve_map_at_back (size_type __nodes_to_add=1)`
- `void _M_reserve_map_at_front (size_type __nodes_to_add=1)`
- `void _M_reallocate_map (size_type __nodes_to_add, bool __add_at_front)`

Static Protected Member Functions

- `static size_t _S_buffer_size () noexcept`

Protected Attributes

- `_Deque_impl _M_impl`

4.539.1 Detailed Description

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
class std::deque<_Tp, _Alloc>
```

A standard container using fixed-size memory allocation and constant-time manipulation of elements at either end.

Template Parameters

<code>_Tp</code>	Type of element.
<code>_Alloc</code>	Allocator type, defaults to <code>allocator<_Tp></code> .

Meets the requirements of a [container](#), a [reversible container](#), and a [sequence](#), including the [optional sequence requirements](#).

In previous HP/SGI versions of deque, there was an extra template parameter so users could control the node size. This extension turned out to violate the C++ standard (it can be detected using template template parameters), and it was removed.

Here's how a `deque<Tp>` manages memory. Each deque has 4 members:

- `Tp** _M_map`
- `size_t _M_map_size`
- `iterator _M_start, _M_finish`

`map_size` is at least 8. `map` is an array of `map_size` pointers-to-*nodes*. (The name `map` has nothing to do with the `std::map` class, and **nodes** should not be confused with `std::list`'s usage of *node*.)

A *node* has no specific type name as such, but it is referred to as *node* in this file. It is a simple array-of-`Tp`. If `Tp` is very large, there will be one `Tp` element per node (i.e., an *array* of one). For non-huge `Tp`'s, node size is inversely related to `Tp` size: the larger the `Tp`, the fewer `Tp`'s will fit in a node. The goal here is to keep the total size of a node relatively small and constant over different `Tp`'s, to improve allocator efficiency.

Not every pointer in the `map` array will point to a node. If the initial number of elements in the deque is small, the `/middle/` `map` pointers will be valid, and the ones at the edges will be unused. This same situation will arise as the `map` grows: available `map` pointers, if any, will be on the ends. As new nodes are created, only a subset of the `map`'s pointers need to be copied *outward*.

Class invariants:

- For any nonsingular iterator `i`:
 - `i.node` points to a member of the `map` array. (Yes, you read that correctly: `i.node` does not actually point to a node.) The member of the `map` array is what actually points to the node.
 - `i.first == *(i.node)` (This points to the node (first `Tp` element).)
 - `i.last == i.first + node_size`
 - `i.cur` is a pointer in the range `[i.first, i.last)`. NOTE: the implication of this is that `i.cur` is always a dereferenceable pointer, even if `i` is a past-the-end iterator.
- `Start` and `Finish` are always nonsingular iterators. NOTE: this means that an empty deque must have one node, a deque with `<N` elements (where `N` is the node buffer size) must have one node, a deque with `N` through `(2N-1)` elements must have two nodes, etc.
- For every node other than `start.node` and `finish.node`, every element in the node is an initialized object. If `start.cur == finish.cur`, then `[start.cur, finish.cur)` are initialized objects, and the elements outside that range are uninitialized storage. Otherwise, `[start.cur, start.last)` and `[finish.first, finish.cur)` are initialized objects, and `[start.first, start.cur)` and `[finish.cur, finish.last)` are uninitialized storage.

- [map, map + map_size) is a valid, non-empty range.
- [start.node, finish.node] is a valid range contained within [map, map + map_size).
- A pointer in the range [map, map + map_size) points to an allocated node if and only if the pointer is in the range [start.node, finish.node].

Here's the magic: nothing in deque is **aware** of the discontinuous storage!

The memory setup and layout occurs in the parent, _Base, and the iterator class is entirely responsible for *leaping* from one node to the next. All the implementation routines for deque itself work only through the start and finish iterators. This keeps the routines simple and sane, and we can use other standard algorithms as well.

Definition at line 735 of file stl_deque.h.

4.539.2 Constructor & Destructor Documentation

4.539.2.1 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::deque<_Tp, _Alloc>::deque ()`
`[inline]`

Creates a deque with no elements.

Definition at line 788 of file stl_deque.h.

4.539.2.2 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::deque<_Tp, _Alloc>::deque (const allocator_type &__a)` `[inline], [explicit]`

Creates a deque with no elements.

Parameters

<code>__a</code>	An allocator object.
------------------	----------------------

Definition at line 795 of file stl_deque.h.

4.539.2.3 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::deque<_Tp, _Alloc>::deque (size_type __n)` `[inline], [explicit]`

Creates a deque with default constructed elements.

Parameters

<code>__n</code>	The number of elements to initially create.
------------------	---

This constructor fills the deque with *n* default constructed elements.

Definition at line 807 of file stl_deque.h.

4.539.2.4 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::deque<_Tp, _Alloc>::deque (size_type __n, const value_type & __value, const allocator_type & __a = allocator_type()) [inline]`

Creates a deque with copies of an exemplar element.

Parameters

<code>__n</code>	The number of elements to initially create.
<code>__value</code>	An element to copy.
<code>__a</code>	An allocator.

This constructor fills the deque with `__n` copies of `__value`.

Definition at line 819 of file `stl_deque.h`.

4.539.2.5 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::deque<_Tp, _Alloc>::deque (const deque<_Tp, _Alloc> & __x) [inline]`

Deque copy constructor.

Parameters

<code>__x</code>	A deque of identical element and allocator types.
------------------	---

The newly-created deque uses a copy of the allocation object used by `__x`.

Definition at line 846 of file `stl_deque.h`.

4.539.2.6 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::deque<_Tp, _Alloc>::deque (deque<_Tp, _Alloc> && __x) [inline]`

Deque move constructor.

Parameters

<code>__x</code>	A deque of identical element and allocator types.
------------------	---

The newly-created deque contains the exact contents of `__x`. The contents of `__x` are a valid, but unspecified deque.

Definition at line 860 of file `stl_deque.h`.

4.539.2.7 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::deque<_Tp, _Alloc>::deque (initializer_list<value_type> __l, const allocator_type & __a = allocator_type()) [inline]`

Builds a deque from an initializer list.

Parameters

<code>__l</code>	An initializer_list.
<code>__a</code>	An allocator object.

Create a deque consisting of copies of the elements in the initializer_list `__l`.

This will call the element type's copy constructor N times (where N is `__l.size()`) and do no memory reallocation.

Definition at line 874 of file `stl_deque.h`.

4.539.2.8 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> template<typename _InputIterator, typename = std::RequireInputIter<_InputIterator>> std::deque<_Tp, _Alloc>::deque (_InputIterator __first, _InputIterator __last, const allocator_type & __a = allocator_type()) [inline]`

Builds a deque from a range.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__a</code>	An allocator object.

Create a deque consisting of copies of the elements from `[__first, __last)`.

If the iterators are forward, bidirectional, or random-access, then this will call the elements' copy constructor N times (where N is `distance(__first, __last)`) and do no memory reallocation. But if only input iterators are used, then this will do at most 2N calls to the copy constructor, and logN memory reallocations.

Definition at line 901 of file `stl_deque.h`.

4.539.2.9 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::deque<_Tp, _Alloc>::~~deque () [inline], [noexcept]`

The dtor only erases the elements, and note that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 922 of file `stl_deque.h`.

4.539.3 Member Function Documentation

4.539.3.1 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::deque<_Tp, _Alloc>::_M_fill_initialize (const value_type & __value) [protected]`

Fills the deque with copies of `value`.

Parameters

<code>__value</code>	Initial value.
----------------------	----------------

Returns

Nothing.

Precondition

`_M_start` and `_M_finish` have already been initialized, but none of the deque's elements have yet been constructed.

This function is called only when the user provides an explicit size (with or without an explicit exemplar value).

4.539.3.2 `template<typename _Tp, typename _Alloc> void std::_Deque_base<_Tp, _Alloc>::_M_initialize_map(size_t __num_elements) [protected], [inherited]`

Layout storage.

Parameters

<code>__num_elements</code>	The count of T's for which to allocate space at first.
-----------------------------	--

Returns

Nothing.

The initial underlying memory layout is a bit complicated...

Definition at line 587 of file `stl_deque.h`.

References `std::max()`.

4.539.3.3 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::deque<_Tp, _Alloc>::_M_new_elements_at_back(size_type __new_elements) [protected]`

Memory-handling helpers for the previous internal insert functions.

4.539.3.4 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::deque<_Tp, _Alloc>::_M_new_elements_at_front(size_type __new_elements) [protected]`

Memory-handling helpers for the previous internal insert functions.

4.539.3.5 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::deque<_Tp, _Alloc>::_M_pop_back_aux() [protected]`

Helper functions for `push_*` and `pop_*`.

4.539.3.6 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::deque< _Tp, _Alloc >::_M_pop_front_aux() [protected]`

Helper functions for push_* and pop_*.

4.539.3.7 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> template<typename... _Args> void std::deque< _Tp, _Alloc >::_M_push_back_aux(_Args &&... __args) [protected]`

Helper functions for push_* and pop_*.

4.539.3.8 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> template<typename... _Args> void std::deque< _Tp, _Alloc >::_M_push_front_aux(_Args &&... __args) [protected]`

Helper functions for push_* and pop_*.

4.539.3.9 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::deque< _Tp, _Alloc >::_M_range_check(size_type __n) const [inline], [protected]`

Safety check used only from at().

Definition at line 1270 of file stl_deque.h.

4.539.3.10 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> template<typename _InputIterator > void std::deque< _Tp, _Alloc >::_M_range_initialize(_InputIterator __first, _InputIterator __last, std::input_iterator_tag) [protected]`

Fills the deque with whatever is in [first,last).

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.

Returns

Nothing.

If the iterators are actually forward iterators (or better), then the memory layout can be done all at once. Else we move forward using push_back on each value from the iterator.

4.539.3.11 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> template<typename _ForwardIterator > void std::deque< _Tp, _Alloc >::_M_range_initialize(_ForwardIterator __first, _ForwardIterator __last, std::forward_iterator_tag) [protected]`

Fills the deque with whatever is in [first,last).

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.

Returns

Nothing.

If the iterators are actually forward iterators (or better), then the memory layout can be done all at once. Else we move forward using `push_back` on each value from the iterator.

```
4.539.3.12 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::deque<_Tp, _Alloc>
           >::_M_reallocate_map ( size_type __nodes_to_add, bool __add_at_front ) [protected]
```

Memory-handling helpers for the major map.

Makes sure the `_M_map` has space for new nodes. Does not actually add the nodes. Can invalidate `_M_map` pointers. (And consequently, deque iterators.)

```
4.539.3.13 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> iterator std::deque<_Tp, _Alloc>
           >::_M_reserve_elements_at_back ( size_type __n ) [inline], [protected]
```

Memory-handling helpers for the previous internal insert functions.

Definition at line 1970 of file `stl_deque.h`.

```
4.539.3.14 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> iterator std::deque<_Tp, _Alloc>
           >::_M_reserve_elements_at_front ( size_type __n ) [inline], [protected]
```

Memory-handling helpers for the previous internal insert functions.

Definition at line 1960 of file `stl_deque.h`.

```
4.539.3.15 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::deque<_Tp, _Alloc>
           >::_M_reserve_map_at_back ( size_type __nodes_to_add = 1 ) [inline], [protected]
```

Memory-handling helpers for the major map.

Makes sure the `_M_map` has space for new nodes. Does not actually add the nodes. Can invalidate `_M_map` pointers. (And consequently, deque iterators.)

Definition at line 1996 of file `stl_deque.h`.

```
4.539.3.16 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::deque<_Tp, _Alloc>
           >::_M_reserve_map_at_front ( size_type __nodes_to_add = 1 ) [inline], [protected]
```

Memory-handling helpers for the major map.

Makes sure the `_M_map` has space for new nodes. Does not actually add the nodes. Can invalidate `_M_map` pointers. (And consequently, deque iterators.)

Definition at line 2004 of file `stl_deque.h`.

```
4.539.3.17 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::deque<_Tp, _Alloc>::assign (
           size_type __n, const value_type & __val ) [inline]
```

Assigns a given value to a deque.

Parameters

<code>__n</code>	Number of elements to be assigned.
<code>__val</code>	Value to be assigned.

This function fills a deque with n copies of the given value. Note that the assignment completely changes the deque and that the resulting deque's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 983 of file `stl_deque.h`.

4.539.3.18 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> template<typename _InputIterator, typename = std::RequireInputIter<_InputIterator>> void std::deque<_Tp, _Alloc>::assign (_InputIterator __first, _InputIterator __last) [inline]`

Assigns a range to a deque.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.

This function fills a deque with copies of the elements in the range `[__first,__last)`.

Note that the assignment completely changes the deque and that the resulting deque's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 1002 of file `stl_deque.h`.

4.539.3.19 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::deque<_Tp, _Alloc>::assign (initializer_list<value_type> __l) [inline]`

Assigns an initializer list to a deque.

Parameters

<code>__l</code>	An initializer_list.
------------------	----------------------

This function fills a deque with copies of the elements in the initializer_list `__l`.

Note that the assignment completely changes the deque and that the resulting deque's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 1027 of file `stl_deque.h`.

4.539.3.20 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> reference std::deque<_Tp, _Alloc>::at (`
`size_type __n) [inline]`

Provides access to the data contained in the deque.

Parameters

<code>__n</code>	The index of the element for which data should be accessed.
------------------	---

Returns

Read/write reference to data.

Exceptions

<code>std::out_of_range</code>	If <code>__n</code> is an invalid index.
--------------------------------	--

This function provides for safer data access. The parameter is first checked that it is in the range of the deque. The function throws `out_of_range` if the check fails.

Definition at line 1292 of file `stl_deque.h`.

4.539.3.21 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reference std::deque<_Tp, _Alloc>::at (`
`size_type __n) const [inline]`

Provides access to the data contained in the deque.

Parameters

<code>__n</code>	The index of the element for which data should be accessed.
------------------	---

Returns

Read-only (constant) reference to data.

Exceptions

<code>std::out_of_range</code>	If <code>__n</code> is an invalid index.
--------------------------------	--

This function provides for safer data access. The parameter is first checked that it is in the range of the deque. The function throws `out_of_range` if the check fails.

Definition at line 1310 of file `stl_deque.h`.

4.539.3.22 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> reference std::deque<_Tp, _Alloc>::back ()`
`[inline], [noexcept]`

Returns a read/write reference to the data at the last element of the deque.

Definition at line 1337 of file `stl_deque.h`.

4.539.3.23 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reference std::deque<_Tp, _Alloc>::back () const` `[inline], [noexcept]`

Returns a read-only (constant) reference to the data at the last element of the deque.

Definition at line 1349 of file `stl_deque.h`.

4.539.3.24 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> iterator std::deque<_Tp, _Alloc>::begin ()`
`[inline], [noexcept]`

Returns a read/write iterator that points to the first element in the deque. Iteration is done in ordinary element order.

Definition at line 1042 of file `stl_deque.h`.

Referenced by `std::deque<_StateSeqT>::deque()`, and `std::operator==()`.

4.539.3.25 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_iterator std::deque<_Tp, _Alloc>::begin () const` `[inline], [noexcept]`

Returns a read-only (constant) iterator that points to the first element in the deque. Iteration is done in ordinary element order.

Definition at line 1050 of file `stl_deque.h`.

4.539.3.26 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_iterator std::deque<_Tp, _Alloc>::cbegin () const` `[inline], [noexcept]`

Returns a read-only (constant) iterator that points to the first element in the deque. Iteration is done in ordinary element order.

Definition at line 1113 of file `stl_deque.h`.

4.539.3.27 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_iterator std::deque<_Tp, _Alloc>::cend () const` `[inline], [noexcept]`

Returns a read-only (constant) iterator that points one past the last element in the deque. Iteration is done in ordinary element order.

Definition at line 1122 of file `stl_deque.h`.

4.539.3.28 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::deque<_Tp, _Alloc>::clear ()`
`[inline], [noexcept]`

Erases all the elements. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1682 of file `stl_deque.h`.

4.539.3.29 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reverse_iterator std::deque<_Tp,`
`_Alloc>::crbegin () const [inline], [noexcept]`

Returns a read-only (constant) reverse iterator that points to the last element in the deque. Iteration is done in reverse element order.

Definition at line 1131 of file `stl_deque.h`.

4.539.3.30 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reverse_iterator std::deque<_Tp,`
`_Alloc>::crend () const [inline], [noexcept]`

Returns a read-only (constant) reverse iterator that points to one before the first element in the deque. Iteration is done in reverse element order.

Definition at line 1140 of file `stl_deque.h`.

4.539.3.31 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> template<typename... _Args> iterator`
`std::deque<_Tp, _Alloc>::emplace (const_iterator __position, _Args &&... __args)`

Inserts an object in deque before specified iterator.

Parameters

<code>__position</code>	A <code>const_iterator</code> into the deque.
<code>__args</code>	Arguments.

Returns

An iterator that points to the inserted data.

This function will insert an object of type `T` constructed with `T(std::forward<Args>(args)...) before the specified location.`

4.539.3.32 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> bool std::deque<_Tp, _Alloc>::empty ()`
`const [inline], [noexcept]`

Returns true if the deque is empty. (Thus `begin()` would equal `end()`.)

Definition at line 1233 of file `stl_deque.h`.

4.539.3.33 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> iterator std::deque<_Tp, _Alloc>::end ()`
`[inline], [noexcept]`

Returns a read/write iterator that points one past the last element in the deque. Iteration is done in ordinary element order.

Definition at line 1059 of file `stl_deque.h`.

Referenced by `std::deque<_StateSeqT>::deque()`, and `std::operator==()`.

4.539.3.34 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_iterator std::deque<_Tp, _Alloc>::end () const` `[inline], [noexcept]`

Returns a read-only (constant) iterator that points one past the last element in the deque. Iteration is done in ordinary element order.

Definition at line 1068 of file `stl_deque.h`.

4.539.3.35 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> iterator std::deque<_Tp, _Alloc>::erase (const_iterator __position)` `[inline]`

Remove element at given position.

Parameters

<code>__position</code>	Iterator pointing to element to be erased.
-------------------------	--

Returns

An iterator pointing to the next element (or `end()`).

This function will erase the element at the given position and thus shorten the deque by one.

The user is cautioned that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1622 of file `stl_deque.h`.

4.539.3.36 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> iterator std::deque<_Tp, _Alloc>::erase (const_iterator __first, const_iterator __last)` `[inline]`

Remove a range of elements.

Parameters

<code>__first</code>	Iterator pointing to the first element to be erased.
<code>__last</code>	Iterator pointing to one past the last element to be erased.

Returns

An iterator pointing to the element pointed to by *last* prior to erasing (or end()).

This function will erase the elements in the range [*__first*,*__last*) and shorten the deque accordingly.

The user is cautioned that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1646 of file `stl_deque.h`.

```
4.539.3.37  template<typename _Tp, typename _Alloc = std::allocator<_Tp>> reference std::deque<_Tp, _Alloc>::front ( )
           [inline], [noexcept]
```

Returns a read/write reference to the data at the first element of the deque.

Definition at line 1321 of file `stl_deque.h`.

```
4.539.3.38  template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reference std::deque<_Tp, _Alloc>::front ( ) const
           [inline], [noexcept]
```

Returns a read-only (constant) reference to the data at the first element of the deque.

Definition at line 1329 of file `stl_deque.h`.

```
4.539.3.39  template<typename _Tp, typename _Alloc = std::allocator<_Tp>> allocator_type std::deque<_Tp, _Alloc>::get_allocator ( ) const
           [inline], [noexcept]
```

Get a copy of the memory allocation object.

Definition at line 1033 of file `stl_deque.h`.

```
4.539.3.40  template<typename _Tp, typename _Alloc = std::allocator<_Tp>> iterator std::deque<_Tp, _Alloc>::insert (
           const_iterator __position, const value_type & __x )
```

Inserts given value into deque before specified iterator.

Parameters

<i>__position</i>	A <code>const_iterator</code> into the deque.
<i>__x</i>	Data to be inserted.

Returns

An iterator that points to the inserted data.

This function will insert a copy of the given value before the specified location.

4.539.3.41 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> iterator std::deque<_Tp, _Alloc>::insert (const_iterator __position, value_type && __x) [inline]`

Inserts given rvalue into deque before specified iterator.

Parameters

<code>__position</code>	A const_iterator into the deque.
<code>__x</code>	Data to be inserted.

Returns

An iterator that points to the inserted data.

This function will insert a copy of the given rvalue before the specified location.

Definition at line 1512 of file `stl_deque.h`.

4.539.3.42 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> iterator std::deque<_Tp, _Alloc>::insert (const_iterator __p, initializer_list<value_type> __l) [inline]`

Inserts an initializer list into the deque.

Parameters

<code>__p</code>	An iterator into the deque.
<code>__l</code>	An initializer_list.

This function will insert copies of the data in the initializer_list `__l` into the deque before the location specified by `__p`. This is known as *list insert*.

Definition at line 1525 of file `stl_deque.h`.

4.539.3.43 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> iterator std::deque<_Tp, _Alloc>::insert (const_iterator __position, size_type __n, const value_type & __x) [inline]`

Inserts a number of copies of given data into the deque.

Parameters

<code>__position</code>	A const_iterator into the deque.
<code>__n</code>	Number of elements to be inserted.
<code>__x</code>	Data to be inserted.

Returns

An iterator that points to the inserted data.

This function will insert a specified number of copies of the given data before the location specified by `__position`.

Definition at line 1541 of file `stl_deque.h`.

```
4.539.3.44  template<typename _Tp, typename _Alloc = std::allocator<_Tp>> template<typename _InputIterator, typename =
std:: _RequireInputIter<_InputIterator>> iterator std::deque<_Tp, _Alloc >::insert ( const_iterator __position,
_InputIterator __first, _InputIterator __last )  [inline]
```

Inserts a range into the deque.

Parameters

<code>__position</code>	A <code>const_iterator</code> into the deque.
<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.

Returns

An iterator that points to the inserted data.

This function will insert copies of the data in the range `[__first,__last)` into the deque before the location specified by `__position`. This is known as *range insert*.

Definition at line 1577 of file `stl_deque.h`.

```
4.539.3.45  template<typename _Tp, typename _Alloc = std::allocator<_Tp>> size_type std::deque<_Tp, _Alloc >::max_size (
) const  [inline], [noexcept]
```

Returns the `size()` of the largest possible deque.

Definition at line 1152 of file `stl_deque.h`.

```
4.539.3.46  template<typename _Tp, typename _Alloc = std::allocator<_Tp>> deque& std::deque<_Tp, _Alloc >::operator= (
const deque<_Tp, _Alloc > &__x )
```

Deque assignment operator.

Parameters

<code>__x</code>	A deque of identical element and allocator types.
------------------	---

All the elements of `x` are copied, but unlike the copy constructor, the allocator object is not copied.

4.539.3.47 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> deque& std::deque<_Tp, _Alloc>::operator= (deque<_Tp, _Alloc> && __x) [inline], [noexcept]`

Deque move assignment operator.

Parameters

<code>__x</code>	A deque of identical element and allocator types.
------------------	---

The contents of `__x` are moved into this deque (without copying). `__x` is a valid, but unspecified deque.

Definition at line 944 of file `stl_deque.h`.

4.539.3.48 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> deque& std::deque<_Tp, _Alloc>::operator= (initializer_list<value_type> __l) [inline]`

Assigns an initializer list to a deque.

Parameters

<code>__l</code>	An <code>initializer_list</code> .
------------------	------------------------------------

This function fills a deque with copies of the elements in the `initializer_list __l`.

Note that the assignment completely changes the deque and that the resulting deque's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 965 of file `stl_deque.h`.

4.539.3.49 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> reference std::deque<_Tp, _Alloc>::operator[] (size_type __n) [inline], [noexcept]`

Subscript access to the data contained in the deque.

Parameters

<code>__n</code>	The index of the element for which data should be accessed.
------------------	---

Returns

Read/write reference to data.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and `out_of_range` lookups are not defined. (For checked lookups see `at()`.)

Definition at line 1249 of file `stl_deque.h`.

```
4.539.3.50  template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reference std::deque< _Tp, _Alloc
              >::operator[]( size_type __n ) const    [inline], [noexcept]
```

Subscript access to the data contained in the deque.

Parameters

<code>__n</code>	The index of the element for which data should be accessed.
------------------	---

Returns

Read-only (constant) reference to data.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and `out_of_range` lookups are not defined. (For checked lookups see `at()`.)

Definition at line 1264 of file `stl_deque.h`.

```
4.539.3.51  template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::deque< _Tp, _Alloc >::pop_back ( )
              [inline], [noexcept]
```

Removes last element.

This is a typical stack operation. It shrinks the deque by one.

Note that no data is returned, and if the last element's data is needed, it should be retrieved before `pop_back()` is called.

Definition at line 1450 of file `stl_deque.h`.

```
4.539.3.52  template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::deque< _Tp, _Alloc >::pop_front ( )
              [inline], [noexcept]
```

Removes first element.

This is a typical stack operation. It shrinks the deque by one.

Note that no data is returned, and if the first element's data is needed, it should be retrieved before `pop_front()` is called.

Definition at line 1429 of file `stl_deque.h`.

```
4.539.3.53  template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::deque< _Tp, _Alloc >::push_back (
              const value_type &__x )    [inline]
```

Add data to the end of the deque.

Parameters

$_x$	Data to be added.
-------	-------------------

This is a typical stack operation. The function creates an element at the end of the deque and assigns the given data to it. Due to the nature of a deque this operation can be done in constant time.

Definition at line 1398 of file stl_deque.h.

```
4.539.3.54 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::deque< _Tp, _Alloc >::push_front (
    const value_type &_x ) [inline]
```

Add data to the front of the deque.

Parameters

$_x$	Data to be added.
-------	-------------------

This is a typical stack operation. The function creates an element at the front of the deque and assigns the given data to it. Due to the nature of a deque this operation can be done in constant time.

Definition at line 1367 of file stl_deque.h.

```
4.539.3.55 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> reverse_iterator std::deque< _Tp, _Alloc
    >::rbegin ( ) [inline], [noexcept]
```

Returns a read/write reverse iterator that points to the last element in the deque. Iteration is done in reverse element order.

Definition at line 1077 of file stl_deque.h.

```
4.539.3.56 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reverse_iterator std::deque< _Tp,
    _Alloc >::rbegin ( ) const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last element in the deque. Iteration is done in reverse element order.

Definition at line 1086 of file stl_deque.h.

```
4.539.3.57 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> reverse_iterator std::deque< _Tp, _Alloc
    >::rend ( ) [inline], [noexcept]
```

Returns a read/write reverse iterator that points to one before the first element in the deque. Iteration is done in reverse element order.

Definition at line 1095 of file stl_deque.h.

4.539.3.58 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reverse_iterator std::deque<_Tp, _Alloc>::rend () const [inline], [noexcept]`

Returns a read-only (constant) reverse iterator that points to one before the first element in the deque. Iteration is done in reverse element order.

Definition at line 1104 of file `stl_deque.h`.

4.539.3.59 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::deque<_Tp, _Alloc>::resize (size_type __new_size) [inline]`

Resizes the deque to the specified number of elements.

Parameters

<code>__new_size</code>	Number of elements the deque should contain.
-------------------------	--

This function will resize the deque to the specified number of elements. If the number is smaller than the deque's current size the deque is truncated, otherwise default constructed elements are appended.

Definition at line 1166 of file `stl_deque.h`.

4.539.3.60 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::deque<_Tp, _Alloc>::resize (size_type __new_size, const value_type &__x) [inline]`

Resizes the deque to the specified number of elements.

Parameters

<code>__new_size</code>	Number of elements the deque should contain.
<code>__x</code>	Data with which new elements should be populated.

This function will resize the deque to the specified number of elements. If the number is smaller than the deque's current size the deque is truncated, otherwise the deque is extended and new elements are populated with given data.

Definition at line 1188 of file `stl_deque.h`.

4.539.3.61 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::deque<_Tp, _Alloc>::shrink_to_fit () [inline], [noexcept]`

A non-binding request to reduce memory use.

Definition at line 1224 of file `stl_deque.h`.

4.539.3.62 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> size_type std::deque<_Tp, _Alloc>::size () const [inline], [noexcept]`

Returns the number of elements in the deque.

Definition at line 1147 of file `stl_deque.h`.

Referenced by `std::operator==()`.

```
4.539.3.63 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::deque<_Tp, _Alloc>::swap (
    deque<_Tp, _Alloc> &__x ) [inline], [noexcept]
```

Swaps data with another deque.

Parameters

<code>__x</code>	A deque of the same element and allocator types.
------------------	--

This exchanges the elements between two deques in constant time. (Four pointers, so it should be quite fast.) Note that the global `std::swap()` function is specialized such that `std::swap(d1,d2)` will feed to this function.

Definition at line 1662 of file `stl_deque.h`.

Referenced by `std::swap()`.

The documentation for this class was generated from the following file:

- [stl_deque.h](#)

4.540 `std::discard_block_engine<_RandomNumberEngine, __p, __r>` Class Template Reference

Public Types

- typedef `_RandomNumberEngine::result_type` [result_type](#)

Public Member Functions

- [discard_block_engine](#) ()
- [discard_block_engine](#) (const `_RandomNumberEngine` &__rng)
- [discard_block_engine](#) (`_RandomNumberEngine` &&__rng)
- [discard_block_engine](#) ([result_type](#) __s)
- template<typename `_Sseq` , typename = typename `std::enable_if<!std::is_same<_Sseq, discard_block_engine>::value && !std::is_<_Sseq, _RandomNumberEngine>::value>::type>`
[discard_block_engine](#) (`_Sseq` &__q)
- const `_RandomNumberEngine` & [base](#) () const noexcept
- void [discard](#) (unsigned long long __z)
- [result_type](#) [operator\(\)](#) ()
- void [seed](#) ()
- void [seed](#) ([result_type](#) __s)
- template<typename `_Sseq` >
void [seed](#) (`_Sseq` &__q)

Static Public Member Functions

- static constexpr [result_type](#) [max](#) ()
- static constexpr [result_type](#) [min](#) ()

Static Public Attributes

- static constexpr size_t **block_size**
- static constexpr size_t **used_block**

Friends

- template<typename _RandomNumberEngine1, size_t __p1, size_t __r1, typename _CharT, typename _Traits >
std::basic_ostream< _CharT, _Traits > & **operator<<** (std::basic_ostream< _CharT, _Traits > &__os, const
std::discard_block_engine< _RandomNumberEngine1, __p1, __r1 > &__x)
- bool **operator==** (const **discard_block_engine** &__lhs, const **discard_block_engine** &__rhs)
- template<typename _RandomNumberEngine1, size_t __p1, size_t __r1, typename _CharT, typename _Traits >
std::basic_istream< _CharT, _Traits > & **operator>>** (std::basic_istream< _CharT, _Traits > &__is, **std::←**
::discard_block_engine< _RandomNumberEngine1, __p1, __r1 > &__x)

4.540.1 Detailed Description

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
class std::discard_block_engine< _RandomNumberEngine, __p, __r >
```

Produces random numbers from some base engine by discarding blocks of data.

0 <= __r <= __p

Definition at line 856 of file random.h.

4.540.2 Member Typedef Documentation

4.540.2.1 template<typename _RandomNumberEngine, size_t __p, size_t __r> typedef _RandomNumberEngine::result_type
std::discard_block_engine< _RandomNumberEngine, __p, __r >::result_type

The type of the generated random value.

Definition at line 859 of file random.h.

4.540.3 Constructor & Destructor Documentation

4.540.3.1 template<typename _RandomNumberEngine, size_t __p, size_t __r> std::discard_block_engine<
_RandomNumberEngine, __p, __r >::discard_block_engine () [inline]

Constructs a default discard_block_engine engine.

The underlying engine is default constructed as well.

Definition at line 874 of file random.h.

4.540.3.2 template<typename _RandomNumberEngine, size_t __p, size_t __r> std::discard_block_engine<
_RandomNumberEngine, __p, __r >::discard_block_engine (const _RandomNumberEngine & __rng)
[inline], [explicit]

Copy constructs a discard_block_engine engine.

Copies an existing base class random number generator.

Parameters

<code>__rng</code>	An existing (base class) engine object.
--------------------	---

Definition at line 884 of file random.h.

```
4.540.3.3 template<typename _RandomNumberEngine , size_t __p, size_t __r> std::discard_block_engine<
    _RandomNumberEngine, __p, __r>::discard_block_engine ( _RandomNumberEngine && __rng ) [inline],
    [explicit]
```

Move constructs a discard_block_engine engine.

Copies an existing base class random number generator.

Parameters

<code>__rng</code>	An existing (base class) engine object.
--------------------	---

Definition at line 894 of file random.h.

```
4.540.3.4 template<typename _RandomNumberEngine , size_t __p, size_t __r> std::discard_block_engine<
    _RandomNumberEngine, __p, __r>::discard_block_engine ( result_type __s ) [inline],[explicit]
```

Seed constructs a discard_block_engine engine.

Constructs the underlying generator engine seeded with __s.

Parameters

<code>__s</code>	A seed value for the base class engine.
------------------	---

Definition at line 904 of file random.h.

```
4.540.3.5 template<typename _RandomNumberEngine , size_t __p, size_t __r> template<typename _Sseq , typename
    = typename std::enable_if<!std::is_same<_Sseq, discard_block_engine>::value && !std::is_same<_Sseq,
    _RandomNumberEngine>::value> ::type> std::discard_block_engine< _RandomNumberEngine, __p, __r
    >::discard_block_engine ( _Sseq & __q ) [inline],[explicit]
```

Generator construct a discard_block_engine engine.

Parameters

<code>__q</code>	A seed sequence.
------------------	------------------

Definition at line 917 of file random.h.

4.540.4 Member Function Documentation

4.540.4.1 `template<typename _RandomNumberEngine, size_t __p, size_t __r> const _RandomNumberEngine& std::discard_block_engine<_RandomNumberEngine, __p, __r>::base() const [inline], [noexcept]`

Gets a const reference to the underlying generator engine object.

Definition at line 961 of file random.h.

4.540.4.2 `template<typename _RandomNumberEngine, size_t __p, size_t __r> void std::discard_block_engine<_RandomNumberEngine, __p, __r>::discard(unsigned long long __z) [inline]`

Discard a sequence of random numbers.

Definition at line 982 of file random.h.

4.540.4.3 `template<typename _RandomNumberEngine, size_t __p, size_t __r> static constexpr result_type std::discard_block_engine<_RandomNumberEngine, __p, __r>::max() [inline], [static]`

Gets the maximum value in the generated random number range.

Definition at line 975 of file random.h.

References `std::max()`.

4.540.4.4 `template<typename _RandomNumberEngine, size_t __p, size_t __r> static constexpr result_type std::discard_block_engine<_RandomNumberEngine, __p, __r>::min() [inline], [static]`

Gets the minimum value in the generated random number range.

Definition at line 968 of file random.h.

References `std::min()`.

4.540.4.5 `template<typename _RandomNumberEngine, size_t __p, size_t __r> result_type std::discard_block_engine<_RandomNumberEngine, __p, __r>::operator()()`

Gets the next value in the generated random number sequence.

4.540.4.6 `template<typename _RandomNumberEngine, size_t __p, size_t __r> void std::discard_block_engine<_RandomNumberEngine, __p, __r>::seed() [inline]`

Reseeds the `discard_block_engine` object with the default seed for the underlying base class generator engine.

Definition at line 926 of file random.h.

4.540.4.7 `template<typename _RandomNumberEngine, size_t __p, size_t __r> void std::discard_block_engine<_RandomNumberEngine, __p, __r>::seed(result_type __s) [inline]`

Reseeds the `discard_block_engine` object with the default seed for the underlying base class generator engine.

Definition at line 937 of file random.h.

4.540.4.8 `template<typename _RandomNumberEngine, size_t __p, size_t __r> template<typename _Sseq> void std::discard_block_engine<_RandomNumberEngine, __p, __r>::seed(_Sseq & __q) [inline]`

Reseeds the `discard_block_engine` object with the given seed sequence.

Parameters

<code>__q</code>	A seed generator function.
------------------	----------------------------

Definition at line 950 of file random.h.

4.540.5 Friends And Related Function Documentation

4.540.5.1 `template<typename _RandomNumberEngine, size_t __p, size_t __r> template<typename _RandomNumberEngine1, size_t __p1, size_t __r1, typename _CharT, typename _Traits> std::basic_ostream<_CharT, _Traits>& operator<< (std::basic_ostream<_CharT, _Traits> & __os, const std::discard_block_engine<_RandomNumberEngine1, __p1, __r1> & __x) [friend]`

Inserts the current state of a discard_block_engine random number generator engine __x into the output stream __os.

Parameters

<code>__os</code>	An output stream.
<code>__x</code>	A discard_block_engine random number generator engine.

Returns

The output stream with the state of __x inserted or in an error state.

4.540.5.2 `template<typename _RandomNumberEngine, size_t __p, size_t __r> bool operator==(const discard_block_engine<_RandomNumberEngine, __p, __r> & __lhs, const discard_block_engine<_RandomNumberEngine, __p, __r> & __rhs) [friend]`

Compares two discard_block_engine random number generator objects of the same type for equality.

Parameters

<code>__lhs</code>	A discard_block_engine random number generator object.
<code>__rhs</code>	Another discard_block_engine random number generator object.

Returns

true if the infinite sequences of generated values would be equal, false otherwise.

Definition at line 1006 of file random.h.

```
4.540.5.3 template<typename _RandomNumberEngine , size_t __p, size_t __r> template<typename _RandomNumberEngine1 ,
size_t __p1, size_t __r1, typename _CharT , typename _Traits > std::basic_istream<_CharT, _Traits>& operator>> (
std::basic_istream<_CharT, _Traits> & __is, std::discard_block_engine<_RandomNumberEngine1, __p1, __r1 >
& __x ) [friend]
```

Extracts the current state of a % subtract_with_carry_engine random number generator engine __x from the input stream __is.

Parameters

<code>__is</code>	An input stream.
<code>__x</code>	A discard_block_engine random number generator engine.

Returns

The input stream with the state of __x extracted or in an error state.

The documentation for this class was generated from the following file:

- [random.h](#)

4.541 std::discrete_distribution<_IntType> Class Template Reference

Classes

- struct [param_type](#)

Public Types

- typedef _IntType [result_type](#)

Public Member Functions

- template<typename _InputIterator >
discrete_distribution (_InputIterator __wbegin, _InputIterator __wend)
- **discrete_distribution** (initializer_list< double > __wl)
- template<typename _Func >
discrete_distribution (size_t __nw, double __xmin, double __xmax, _Func __fw)
- **discrete_distribution** (const [param_type](#) & __p)
- template<typename _ForwardIterator , typename _UniformRandomNumberGenerator >
void **generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator & __urng)
- template<typename _ForwardIterator , typename _UniformRandomNumberGenerator >
void **generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator & __urng, const [param_type](#) & __p)

- template<typename _UniformRandomNumberGenerator >
void **generate** (result_type *__f, result_type *__t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- result_type max () const
- result_type min () const
- template<typename _UniformRandomNumberGenerator >
result_type operator() (_UniformRandomNumberGenerator &__urng)
- template<typename _UniformRandomNumberGenerator >
result_type operator() (_UniformRandomNumberGenerator &__urng, const param_type &__p)
- param_type param () const
- void param (const param_type &__param)
- std::vector< double > probabilities () const
- void reset ()

Friends

- template<typename _IntType1, typename _CharT, typename _Traits >
std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::discrete_distribution< _IntType1 > &__x)
- bool operator== (const discrete_distribution &__d1, const discrete_distribution &__d2)
- template<typename _IntType1, typename _CharT, typename _Traits >
std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::discrete_distribution< _IntType1 > &__x)

4.541.1 Detailed Description

```
template<typename _IntType = int>
class std::discrete_distribution< _IntType >
```

A discrete_distribution random number distribution.

The formula for the discrete probability mass function is

Definition at line 5253 of file random.h.

4.541.2 Member Typedef Documentation

4.541.2.1 template<typename _IntType = int> typedef _IntType std::discrete_distribution< _IntType >::result_type

The type of the range of the distribution.

Definition at line 5256 of file random.h.

4.541.3 Member Function Documentation

4.541.3.1 `template<typename _IntType = int> result_type std::discrete_distribution< _IntType >::max () const`
`[inline]`

Returns the least upper bound value of the distribution.

Definition at line 5373 of file random.h.

4.541.3.2 `template<typename _IntType = int> result_type std::discrete_distribution< _IntType >::min () const`
`[inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 5366 of file random.h.

4.541.3.3 `template<typename _IntType = int> template<typename _UniformRandomNumberGenerator > result_type`
`std::discrete_distribution< _IntType >::operator() (_UniformRandomNumberGenerator & __urng) [inline]`

Generating functions.

Definition at line 5384 of file random.h.

4.541.3.4 `template<typename _IntType = int> param_type std::discrete_distribution< _IntType >::param () const`
`[inline]`

Returns the parameter set of the distribution.

Definition at line 5351 of file random.h.

4.541.3.5 `template<typename _IntType = int> void std::discrete_distribution< _IntType >::param (const param_type &`
`__param) [inline]`

Sets the parameter set of the distribution.

Parameters

<code>__param</code>	The new parameter set of the distribution.
----------------------	--

Definition at line 5359 of file random.h.

4.541.3.6 `template<typename _IntType = int> std::vector<double> std::discrete_distribution< _IntType >::probabilities (`
`) const [inline]`

Returns the probabilities of the distribution.

Definition at line 5341 of file random.h.

4.541.3.7 `template<typename _IntType = int> void std::discrete_distribution<_IntType>::reset () [inline]`

Resets the distribution state.

Definition at line 5334 of file random.h.

4.541.4 Friends And Related Function Documentation

4.541.4.1 `template<typename _IntType = int> template<typename _IntType1, typename _CharT, typename _Traits > std::basic_ostream<_CharT, _Traits>& operator<< (std::basic_ostream<_CharT, _Traits> &__os, const std::discrete_distribution<_IntType1> &__x) [friend]`

Inserts a discrete_distribution random number distribution __x into the output stream __os.

Parameters

<code>__os</code>	An output stream.
<code>__x</code>	A discrete_distribution random number distribution.

Returns

The output stream with the state of __x inserted or in an error state.

4.541.4.2 `template<typename _IntType = int> bool operator== (const discrete_distribution<_IntType> &__d1, const discrete_distribution<_IntType> &__d2) [friend]`

Return true if two discrete distributions have the same parameters.

Definition at line 5419 of file random.h.

4.541.4.3 `template<typename _IntType = int> template<typename _IntType1, typename _CharT, typename _Traits > std::basic_istream<_CharT, _Traits>& operator>> (std::basic_istream<_CharT, _Traits> &__is, std::discrete_distribution<_IntType1> &__x) [friend]`

Extracts a discrete_distribution random number distribution __x from the input stream __is.

Parameters

<code>__is</code>	An input stream.
<code>__x</code>	A discrete_distribution random number generator engine.

Returns

The input stream with __x extracted or in an error state.

The documentation for this class was generated from the following file:

- [random.h](#)

4.542 std::discrete_distribution< _IntType >::param_type Struct Reference

Public Types

- typedef [discrete_distribution](#)< _IntType > **distribution_type**

Public Member Functions

- template<typename _InputIterator >
param_type (_InputIterator __wbegin, _InputIterator __wend)
- **param_type** (initializer_list< double > __wil)
- template<typename _Func >
param_type (size_t __nw, double __xmin, double __xmax, _Func __fw)
- **param_type** (const [param_type](#) &)=default
- [param_type](#) & **operator=** (const [param_type](#) &)=default
- [std::vector](#)< double > **probabilities** () const

Friends

- class **discrete_distribution**< _IntType >
- bool **operator==** (const [param_type](#) &__p1, const [param_type](#) &__p2)

4.542.1 Detailed Description

```
template<typename _IntType = int>
struct std::discrete_distribution< _IntType >::param_type
```

Parameter type.

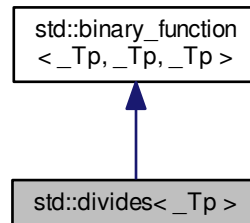
Definition at line 5262 of file random.h.

The documentation for this struct was generated from the following file:

- [random.h](#)

4.543 std::divides< _Tp > Struct Template Reference

Inheritance diagram for std::divides< _Tp >:



Public Types

- typedef _Tp [first_argument_type](#)
- typedef _Tp [result_type](#)
- typedef _Tp [second_argument_type](#)

Public Member Functions

- **_Tp operator()** (const _Tp &__x, const _Tp &__y) const

4.543.1 Detailed Description

```
template<typename _Tp>  
struct std::divides< _Tp >
```

One of the [math functors](#).

Definition at line 194 of file stl_function.h.

4.543.2 Member Typedef Documentation

4.543.2.1 typedef _Tp std::binary_function< _Tp, _Tp, _Tp >::first_argument_type [inherited]

first_argument_type is the type of the first argument

Definition at line 121 of file stl_function.h.

4.543.2.2 `typedef _Tp std::binary_function<_Tp, _Tp, _Tp>::result_type` [inherited]

`result_type` is the return type

Definition at line 127 of file `stl_function.h`.

4.543.2.3 `typedef _Tp std::binary_function<_Tp, _Tp, _Tp>::second_argument_type` [inherited]

`second_argument_type` is the type of the second argument

Definition at line 124 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl_function.h](#)

4.544 `std::enable_shared_from_this<_Tp>` Class Template Reference

Public Member Functions

- [shared_ptr<_Tp>](#) `shared_from_this()`
- [shared_ptr<const _Tp>](#) `shared_from_this() const`

Protected Member Functions

- `enable_shared_from_this` (const [enable_shared_from_this](#) &) noexcept
- `enable_shared_from_this` & `operator=` (const [enable_shared_from_this](#) &) noexcept

Friends

- `template<typename _Tp1, typename _Tp2>`
`void __enable_shared_from_this_helper` (const __shared_count<> &, const [enable_shared_from_this](#)<_Tp1>
 > *, const _Tp2 *) noexcept

4.544.1 Detailed Description

```
template<typename _Tp>
class std::enable_shared_from_this<_Tp>
```

Base class allowing use of member function `shared_from_this`.

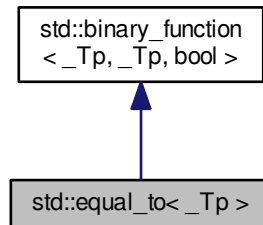
Definition at line 531 of file `shared_ptr.h`.

The documentation for this class was generated from the following file:

- [shared_ptr.h](#)

4.545 std::equal_to< _Tp > Struct Template Reference

Inheritance diagram for std::equal_to< _Tp >:



Public Types

- typedef `_Tp` [first_argument_type](#)
- typedef `bool` [result_type](#)
- typedef `_Tp` [second_argument_type](#)

Public Member Functions

- `bool` **operator()** (`const _Tp &__x`, `const _Tp &__y`) `const`

4.545.1 Detailed Description

```
template<typename _Tp>
struct std::equal_to< _Tp >
```

One of the [comparison functors](#).

Definition at line 340 of file `stl_function.h`.

4.545.2 Member Typedef Documentation

4.545.2.1 typedef `_Tp` `std::binary_function< _Tp, _Tp, bool >::first_argument_type` [\[inherited\]](#)

`first_argument_type` is the type of the first argument

Definition at line 121 of file `stl_function.h`.

4.545.2.2 `typedef bool std::binary_function<_Tp, _Tp, bool>::result_type` [inherited]

`result_type` is the return type

Definition at line 127 of file `stl_function.h`.

4.545.2.3 `typedef _Tp std::binary_function<_Tp, _Tp, bool>::second_argument_type` [inherited]

`second_argument_type` is the type of the second argument

Definition at line 124 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl_function.h](#)

4.546 `std::exponential_distribution<_RealType>` Class Template Reference

Classes

- struct [param_type](#)

Public Types

- typedef `_RealType` [result_type](#)

Public Member Functions

- [exponential_distribution](#) (const [result_type](#) &__lambda=[result_type](#)(1))
- [exponential_distribution](#) (const [param_type](#) &__p)
- template<typename `_ForwardIterator` , typename `_UniformRandomNumberGenerator` >
void [__generate](#) (`_ForwardIterator` __f, `_ForwardIterator` __t, `_UniformRandomNumberGenerator` &__urng)
- template<typename `_ForwardIterator` , typename `_UniformRandomNumberGenerator` >
void [__generate](#) (`_ForwardIterator` __f, `_ForwardIterator` __t, `_UniformRandomNumberGenerator` &__urng, const [param_type](#) &__p)
- template<typename `_UniformRandomNumberGenerator` >
void [__generate](#) ([result_type](#) *__f, [result_type](#) *__t, `_UniformRandomNumberGenerator` &__urng, const [param_type](#) &__p)
- `_RealType` [lambda](#) () const
- [result_type](#) [max](#) () const
- [result_type](#) [min](#) () const
- template<typename `_UniformRandomNumberGenerator` >
[result_type](#) [operator](#)() (`_UniformRandomNumberGenerator` &__urng)
- template<typename `_UniformRandomNumberGenerator` >
[result_type](#) [operator](#)() (`_UniformRandomNumberGenerator` &__urng, const [param_type](#) &__p)
- [param_type](#) [param](#) () const
- void [param](#) (const [param_type](#) &__param)
- void [reset](#) ()

Friends

- bool `operator==` (const `exponential_distribution` &__d1, const `exponential_distribution` &__d2)

4.546.1 Detailed Description

```
template<typename _RealType = double>
class std::exponential_distribution<_RealType>
```

An exponential continuous distribution for random numbers.

The formula for the exponential probability density function is $p(x|\lambda) = \lambda e^{-\lambda x}$.

Table 1085 Distribution Statistics

Mean	$\frac{1}{\lambda}$
Median	$\frac{\ln 2}{\lambda}$
Mode	<i>zero</i>
Range	$[0, \infty]$
Standard Deviation	$\frac{1}{\lambda}$

Definition at line 4648 of file random.h.

4.546.2 Member Typedef Documentation

4.546.2.1 `template<typename _RealType = double> typedef _RealType std::exponential_distribution<_RealType>::result_type`

The type of the range of the distribution.

Definition at line 4651 of file random.h.

4.546.3 Constructor & Destructor Documentation

4.546.3.1 `template<typename _RealType = double> std::exponential_distribution<_RealType>::exponential_distribution (const result_type &__lambda = result_type(1)) [inline], [explicit]`

Constructs an exponential distribution with inverse scale parameter λ .

Definition at line 4686 of file random.h.

4.546.4 Member Function Documentation

4.546.4.1 `template<typename _RealType = double> _RealType std::exponential_distribution< _RealType >::lambda ()`
`const [inline]`

Returns the inverse scale parameter of the distribution.

Definition at line 4707 of file random.h.

4.546.4.2 `template<typename _RealType = double> result_type std::exponential_distribution< _RealType >::max ()`
`const [inline]`

Returns the least upper bound value of the distribution.

Definition at line 4736 of file random.h.

References `std::max()`.

4.546.4.3 `template<typename _RealType = double> result_type std::exponential_distribution< _RealType >::min ()`
`const [inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 4729 of file random.h.

4.546.4.4 `template<typename _RealType = double> template<typename _UniformRandomNumberGenerator > result_type`
`std::exponential_distribution< _RealType >::operator() (_UniformRandomNumberGenerator & __urng)`
`[inline]`

Generating functions.

Definition at line 4744 of file random.h.

4.546.4.5 `template<typename _RealType = double> param_type std::exponential_distribution< _RealType >::param ()`
`const [inline]`

Returns the parameter set of the distribution.

Definition at line 4714 of file random.h.

4.546.4.6 `template<typename _RealType = double> void std::exponential_distribution< _RealType >::param (const`
`param_type & __param) [inline]`

Sets the parameter set of the distribution.

Parameters

<code>__param</code>	The new parameter set of the distribution.
----------------------	--

Definition at line 4722 of file `random.h`.

```
4.546.4.7  template<typename _RealType = double> void std::exponential_distribution<_RealType>::reset ( )  
          [inline]
```

Resets the distribution state.

Has no effect on exponential distributions.

Definition at line 4701 of file `random.h`.

4.546.5 Friends And Related Function Documentation

```
4.546.5.1  template<typename _RealType = double> bool operator==( const exponential_distribution<_RealType> &__d1,  
                        const exponential_distribution<_RealType> &__d2 ) [friend]
```

Return true if two exponential distributions have the same parameters.

Definition at line 4784 of file `random.h`.

The documentation for this class was generated from the following file:

- [random.h](#)

4.547 `std::exponential_distribution<_RealType>::param_type` Struct Reference

Public Types

- typedef [exponential_distribution<_RealType>](#) **distribution_type**

Public Member Functions

- **param_type** (`_RealType` __lambda=`_RealType`(1))
- `_RealType` **lambda** () const

Friends

- bool **operator==** (const [param_type](#) &__p1, const [param_type](#) &__p2)

4.547.1 Detailed Description

```
template<typename _RealType = double>
struct std::exponential_distribution< _RealType >::param_type
```

Parameter type.

Definition at line 4657 of file random.h.

The documentation for this struct was generated from the following file:

- [random.h](#)

4.548 std::extreme_value_distribution< _RealType > Class Template Reference

Classes

- struct [param_type](#)

Public Types

- typedef _RealType [result_type](#)

Public Member Functions

- **extreme_value_distribution** (_RealType __a=_RealType(0), _RealType __b=_RealType(1))
- **extreme_value_distribution** (const [param_type](#) &__p)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator >
void **generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator >
void **generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const [param_type](#) &__p)
- template<typename _UniformRandomNumberGenerator >
void **generate** ([result_type](#) *__f, [result_type](#) *__t, _UniformRandomNumberGenerator &__urng, const [param_type](#) &__p)
- _RealType **a** () const
- _RealType **b** () const
- [result_type](#) **max** () const
- [result_type](#) **min** () const
- template<typename _UniformRandomNumberGenerator >
[result_type](#) **operator()** (_UniformRandomNumberGenerator &__urng)
- template<typename _UniformRandomNumberGenerator >
[result_type](#) **operator()** (_UniformRandomNumberGenerator &__urng, const [param_type](#) &__p)
- [param_type](#) **param** () const
- void **param** (const [param_type](#) &__param)
- void **reset** ()

Friends

- bool `operator==` (const `extreme_value_distribution` &__d1, const `extreme_value_distribution` &__d2)

4.548.1 Detailed Description

```
template<typename _RealType = double>
class std::extreme_value_distribution< _RealType >
```

A `extreme_value_distribution` random number distribution.

The formula for the normal probability mass function is

$$p(x|a, b) = \frac{1}{b} \exp\left(\frac{a-x}{b} - \exp\left(\frac{a-x}{b}\right)\right)$$

Definition at line 5053 of file `random.h`.

4.548.2 Member Typedef Documentation

4.548.2.1 `template<typename _RealType = double> typedef _RealType std::extreme_value_distribution< _RealType >::result_type`

The type of the range of the distribution.

Definition at line 5056 of file `random.h`.

4.548.3 Member Function Documentation

4.548.3.1 `template<typename _RealType = double> _RealType std::extreme_value_distribution< _RealType >::a () const`
`[inline]`

Return the a parameter of the distribution.

Definition at line 5111 of file `random.h`.

4.548.3.2 `template<typename _RealType = double> _RealType std::extreme_value_distribution< _RealType >::b () const`
`[inline]`

Return the b parameter of the distribution.

Definition at line 5118 of file `random.h`.

4.548.3.3 `template<typename _RealType = double> result_type std::extreme_value_distribution< _RealType >::max ()`
`const [inline]`

Returns the least upper bound value of the distribution.

Definition at line 5147 of file random.h.

References `std::max()`.

4.548.3.4 `template<typename _RealType = double> result_type std::extreme_value_distribution< _RealType >::min ()`
`const [inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 5140 of file random.h.

4.548.3.5 `template<typename _RealType = double> template<typename _UniformRandomNumberGenerator > result_type`
`std::extreme_value_distribution< _RealType >::operator() (_UniformRandomNumberGenerator & __urng)`
`[inline]`

Generating functions.

Definition at line 5155 of file random.h.

4.548.3.6 `template<typename _RealType = double> param_type std::extreme_value_distribution< _RealType >::param (`
`) const [inline]`

Returns the parameter set of the distribution.

Definition at line 5125 of file random.h.

4.548.3.7 `template<typename _RealType = double> void std::extreme_value_distribution< _RealType >::param (const`
`param_type & __param) [inline]`

Sets the parameter set of the distribution.

Parameters

<code>__param</code>	The new parameter set of the distribution.
----------------------	--

Definition at line 5133 of file random.h.

4.548.3.8 `template<typename _RealType = double> void std::extreme_value_distribution< _RealType >::reset ()`
`[inline]`

Resets the distribution state.

Definition at line 5104 of file random.h.

4.548.4 Friends And Related Function Documentation

4.548.4.1 `template<typename _RealType = double> bool operator==(const extreme_value_distribution<_RealType> & __d1, const extreme_value_distribution<_RealType> & __d2)` [friend]

Return true if two extreme value distributions have the same parameters.

Definition at line 5190 of file random.h.

The documentation for this class was generated from the following file:

- [random.h](#)

4.549 std::extreme_value_distribution<_RealType>::param_type Struct Reference

Public Types

- typedef [extreme_value_distribution<_RealType>](#) **distribution_type**

Public Member Functions

- **param_type** (`_RealType __a=_RealType(0), _RealType __b=_RealType(1)`)
- `_RealType a` () const
- `_RealType b` () const

Friends

- bool **operator==** (const [param_type](#) &__p1, const [param_type](#) &__p2)

4.549.1 Detailed Description

```
template<typename _RealType = double>
struct std::extreme_value_distribution<_RealType>::param_type
```

Parameter type.

Definition at line 5062 of file random.h.

The documentation for this struct was generated from the following file:

- [random.h](#)

4.550 `std::fisher_f_distribution<_RealType>` Class Template Reference

Classes

- struct [param_type](#)

Public Types

- typedef `_RealType` [result_type](#)

Public Member Functions

- **`fisher_f_distribution`** (`_RealType __m=_RealType(1), _RealType __n=_RealType(1)`)
- **`fisher_f_distribution`** (`const param_type &__p`)
- `template<typename _ForwardIterator, typename _UniformRandomNumberGenerator >`
`void __generate (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng)`
- `template<typename _ForwardIterator, typename _UniformRandomNumberGenerator >`
`void __generate (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param_type &__p)`
- `template<typename _UniformRandomNumberGenerator >`
`void __generate (result_type *__f, result_type *__t, _UniformRandomNumberGenerator &__urng)`
- `template<typename _UniformRandomNumberGenerator >`
`void __generate (result_type *__f, result_type *__t, _UniformRandomNumberGenerator &__urng, const param_type &__p)`
- `_RealType m () const`
- `result_type max () const`
- `result_type min () const`
- `_RealType n () const`
- `template<typename _UniformRandomNumberGenerator >`
`result_type operator() (_UniformRandomNumberGenerator &__urng)`
- `template<typename _UniformRandomNumberGenerator >`
`result_type operator() (_UniformRandomNumberGenerator &__urng, const param_type &__p)`
- `param_type param () const`
- `void param (const param_type &__param)`
- `void reset ()`

Friends

- `template<typename _RealType1, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::fisher_f_distribution<_RealType1 > &__x)`
- `bool operator== (const fisher_f_distribution &__d1, const fisher_f_distribution &__d2)`
- `template<typename _RealType1, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::fisher_f_distribution<_RealType1 > &__x)`

4.550.1 Detailed Description

```
template<typename _RealType = double>
class std::fisher_f_distribution<_RealType>
```

A fisher_f_distribution random number distribution.

The formula for the normal probability mass function is

$$p(x|m, n) = \frac{\Gamma((m+n)/2)}{\Gamma(m/2)\Gamma(n/2)} \left(\frac{m}{n}\right)^{m/2} x^{(m/2)-1} \left(1 + \frac{mx}{n}\right)^{-(m+n)/2}$$

Definition at line 3132 of file random.h.

4.550.2 Member Typedef Documentation

4.550.2.1 `template<typename _RealType = double> typedef _RealType std::fisher_f_distribution<_RealType>::result_type`

The type of the range of the distribution.

Definition at line 3135 of file random.h.

4.550.3 Member Function Documentation

4.550.3.1 `template<typename _RealType = double> result_type std::fisher_f_distribution<_RealType>::max () const [inline]`

Returns the least upper bound value of the distribution.

Definition at line 3226 of file random.h.

References std::max().

4.550.3.2 `template<typename _RealType = double> result_type std::fisher_f_distribution<_RealType>::min () const [inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 3219 of file random.h.

4.550.3.3 `template<typename _RealType = double> template<typename _UniformRandomNumberGenerator> result_type std::fisher_f_distribution<_RealType>::operator() (_UniformRandomNumberGenerator & __urng) [inline]`

Generating functions.

Definition at line 3234 of file random.h.

4.550.3.4 `template<typename _RealType = double> param_type std::fisher_f_distribution<_RealType>::param () const [inline]`

Returns the parameter set of the distribution.

Definition at line 3204 of file random.h.

4.550.3.5 `template<typename _RealType = double> void std::fisher_f_distribution<_RealType>::param (const param_type & __param) [inline]`

Sets the parameter set of the distribution.

Parameters

<code>__param</code>	The new parameter set of the distribution.
----------------------	--

Definition at line 3212 of file random.h.

4.550.3.6 `template<typename _RealType = double> void std::fisher_f_distribution<_RealType>::reset() [inline]`

Resets the distribution state.

Definition at line 3183 of file random.h.

4.550.4 Friends And Related Function Documentation

4.550.4.1 `template<typename _RealType = double> template<typename _RealType1 , typename _CharT , typename _Traits
> std::basic_ostream<_CharT, _Traits>& operator<< (std::basic_ostream<_CharT, _Traits> & __os, const
std::fisher_f_distribution<_RealType1> & __x) [friend]`

Inserts a `fisher_f_distribution` random number distribution `__x` into the output stream `__os`.

Parameters

<code>__os</code>	An output stream.
<code>__x</code>	A <code>fisher_f_distribution</code> random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

4.550.4.2 `template<typename _RealType = double> bool operator==(const fisher_f_distribution<_RealType> & __d1,
const fisher_f_distribution<_RealType> & __d2) [friend]`

Return true if two Fisher f distributions have the same parameters and the sequences that would be generated are equal.

Definition at line 3282 of file random.h.

4.550.4.3 `template<typename _RealType = double> template<typename _RealType1 , typename _CharT , typename
_Traits> std::basic_istream<_CharT, _Traits>& operator>> (std::basic_istream<_CharT, _Traits> & __is,
std::fisher_f_distribution<_RealType1> & __x) [friend]`

Extracts a `fisher_f_distribution` random number distribution `__x` from the input stream `__is`.

Parameters

<code>__is</code>	An input stream.
<code>__x</code>	A <code>fisher_f_distribution</code> random number generator engine.

Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following file:

- [random.h](#)

4.551 `std::fisher_f_distribution<_RealType>::param_type` Struct Reference

Public Types

- typedef [fisher_f_distribution<_RealType>](#) **distribution_type**

Public Member Functions

- **param_type** (`_RealType __m=_RealType(1), _RealType __n=_RealType(1)`)
- `_RealType m` () const
- `_RealType n` () const

Friends

- bool **operator==** (const [param_type](#) &__p1, const [param_type](#) &__p2)

4.551.1 Detailed Description

```
template<typename _RealType = double>
struct std::fisher_f_distribution<_RealType>::param_type
```

Parameter type.

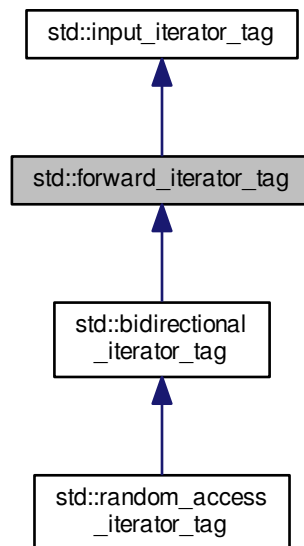
Definition at line 3141 of file `random.h`.

The documentation for this struct was generated from the following file:

- [random.h](#)

4.552 std::forward_iterator_tag Struct Reference

Inheritance diagram for std::forward_iterator_tag:



4.552.1 Detailed Description

Forward iterators support a superset of input iterator operations.

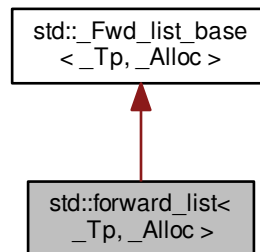
Definition at line 95 of file `stl_iterator_base_types.h`.

The documentation for this struct was generated from the following file:

- [stl_iterator_base_types.h](#)

4.553 std::forward_list< _Tp, _Alloc > Class Template Reference

Inheritance diagram for std::forward_list< _Tp, _Alloc >:



Public Types

- typedef _Alloc **allocator_type**
- typedef [_Fwd_list_const_iterator](#)< _Tp > **const_iterator**
- typedef _Alloc_traits::const_pointer **const_pointer**
- typedef const value_type & **const_reference**
- typedef std::ptrdiff_t **difference_type**
- typedef [_Fwd_list_iterator](#)< _Tp > **iterator**
- typedef _Alloc_traits::pointer **pointer**
- typedef value_type & **reference**
- typedef std::size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- [forward_list](#) (const _Alloc &__al=_Alloc())
- [forward_list](#) (const [forward_list](#) &__list, const _Alloc &__al)
- [forward_list](#) ([forward_list](#) &&__list, const _Alloc &__al) noexcept(_Node_alloc_traits::_S_always_equal())
- [forward_list](#) (size_type __n, const _Alloc &__al=_Alloc())
- [forward_list](#) (size_type __n, const _Tp &__value, const _Alloc &__al=_Alloc())
- template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>>
[forward_list](#) (_InputIterator __first, _InputIterator __last, const _Alloc &__al=_Alloc())
- [forward_list](#) (const [forward_list](#) &__list)
- [forward_list](#) ([forward_list](#) &&__list) noexcept
- [forward_list](#) (std::initializer_list< _Tp > __il, const _Alloc &__al=_Alloc())
- [~forward_list](#) () noexcept
- template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>>
void [assign](#) (_InputIterator __first, _InputIterator __last)
- void [assign](#) (size_type __n, const _Tp &__val)

- void [assign](#) (std::initializer_list< _Tp > __il)
- [iterator before_begin](#) () noexcept
- [const_iterator before_begin](#) () const noexcept
- [iterator begin](#) () noexcept
- [const_iterator begin](#) () const noexcept
- [const_iterator cbefore_begin](#) () const noexcept
- [const_iterator cbegin](#) () const noexcept
- [const_iterator cend](#) () const noexcept
- void [clear](#) () noexcept
- template<typename... _Args>
 [iterator emplace_after](#) (const_iterator __pos, _Args &&... __args)
- template<typename... _Args>
 void [emplace_front](#) (_Args &&... __args)
- bool [empty](#) () const noexcept
- [iterator end](#) () noexcept
- [const_iterator end](#) () const noexcept
- [iterator erase_after](#) (const_iterator __pos)
- [iterator erase_after](#) (const_iterator __pos, const_iterator __last)
- reference [front](#) ()
- const_reference [front](#) () const
- allocator_type [get_allocator](#) () const noexcept
- [iterator insert_after](#) (const_iterator __pos, const _Tp & __val)
- [iterator insert_after](#) (const_iterator __pos, _Tp && __val)
- [iterator insert_after](#) (const_iterator __pos, size_type __n, const _Tp & __val)
- template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>>
 [iterator insert_after](#) (const_iterator __pos, _InputIterator __first, _InputIterator __last)
- [iterator insert_after](#) (const_iterator __pos, std::initializer_list< _Tp > __il)
- size_type [max_size](#) () const noexcept
- void [merge](#) (forward_list && __list)
- void [merge](#) (forward_list & __list)
- template<typename _Comp >
 void [merge](#) (forward_list && __list, _Comp __comp)
- template<typename _Comp >
 void [merge](#) (forward_list & __list, _Comp __comp)
- forward_list & operator= (const forward_list & __list)
- forward_list & operator= (forward_list && __list) noexcept(_Node_alloc_traits::_S_nothrow_move())
- forward_list & operator= (std::initializer_list< _Tp > __il)
- void [pop_front](#) ()
- void [push_front](#) (const _Tp & __val)
- void [push_front](#) (_Tp && __val)
- void [remove](#) (const _Tp & __val)
- template<typename _Pred >
 void [remove_if](#) (_Pred __pred)
- void [resize](#) (size_type __sz)
- void [resize](#) (size_type __sz, const value_type & __val)
- void [reverse](#) () noexcept
- void [sort](#) ()
- template<typename _Comp >
 void [sort](#) (_Comp __comp)
- void [splice_after](#) (const_iterator __pos, forward_list && __list)
- void [splice_after](#) (const_iterator __pos, forward_list & __list)

- void splice_after (const_iterator __pos, forward_list && __list, const_iterator __i)
- void splice_after (const_iterator __pos, forward_list & __list, const_iterator __i)
- void splice_after (const_iterator __pos, forward_list &&, const_iterator __before, const_iterator __last)
- void splice_after (const_iterator __pos, forward_list &, const_iterator __before, const_iterator __last)
- void swap (forward_list & __list) noexcept (Node_alloc_traits::S_nothrow_swap())
- void unique ()
- template<typename _BinPred >
void unique (_BinPred __binary_pred)

Private Member Functions

- template<typename... _Args>
_Node * _M_create_node (_Args &&... __args)
- _Fwd_list_node_base * _M_erase_after (_Fwd_list_node_base * __pos)
- _Fwd_list_node_base * _M_erase_after (_Fwd_list_node_base * __pos, _Fwd_list_node_base * __last)
- _Node * _M_get_node ()
- _Node_alloc_type & _M_get_Node_allocator () noexcept
- const _Node_alloc_type & _M_get_Node_allocator () const noexcept
- template<typename... _Args>
_Fwd_list_node_base * _M_insert_after (const_iterator __pos, _Args &&... __args)
- void _M_put_node (_Node * __p)

Private Attributes

- _Fwd_list_impl _M_impl

4.553.1 Detailed Description

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
class std::forward_list<_Tp, _Alloc>
```

A standard container with linear time access to elements, and fixed time insertion/deletion at any point in the sequence.

Template Parameters

<code>_Tp</code>	Type of element.
<code>_Alloc</code>	Allocator type, defaults to <code>allocator<_Tp></code> .

Meets the requirements of a [container](#), a [sequence](#), including the [optional sequence requirements](#) with the exception of `at` and `operator[]`.

This is a *singly linked* list. Traversal up the list requires linear time, but adding and removing elements (or *nodes*) is done in constant time, regardless of where the change takes place. Unlike `std::vector` and `std::deque`, random-access iterators are not provided, so subscripting (`[]`) access is not allowed. For algorithms which only need sequential access, this lack makes no difference.

Also unlike the other standard containers, `std::forward_list` provides specialized algorithms unique to linked lists, such as splicing, sorting, and in-place reversal.

Definition at line 414 of file `forward_list.h`.

4.553.2 Constructor & Destructor Documentation

4.553.2.1 `template<typename _Tp, typename _Alloc = allocator<_Tp>> std::forward_list<_Tp, _Alloc>::forward_list (const _Alloc & __al = _Alloc()) [inline], [explicit]`

Creates a `forward_list` with no elements.

Parameters

<code>__al</code>	An allocator object.
-------------------	----------------------

Definition at line 446 of file `forward_list.h`.

4.553.2.2 `template<typename _Tp, typename _Alloc = allocator<_Tp>> std::forward_list<_Tp, _Alloc>::forward_list (const forward_list<_Tp, _Alloc> & __list, const _Alloc & __al) [inline]`

Copy constructor with allocator argument.

Parameters

<code>__list</code>	Input list to copy.
<code>__al</code>	An allocator object.

Definition at line 455 of file `forward_list.h`.

References `std::forward_list<_Tp, _Alloc>::begin()`, and `std::forward_list<_Tp, _Alloc>::end()`.

4.553.2.3 `template<typename _Tp, typename _Alloc = allocator<_Tp>> std::forward_list<_Tp, _Alloc>::forward_list (forward_list<_Tp, _Alloc> && __list, const _Alloc & __al) [inline], [noexcept]`

Move constructor with allocator argument.

Parameters

<code>__list</code>	Input list to move.
<code>__al</code>	An allocator object.

Definition at line 464 of file `forward_list.h`.

4.553.2.4 `template<typename _Tp, typename _Alloc = allocator<_Tp>> std::forward_list<_Tp, _Alloc>::forward_list (size_type __n, const _Alloc & __al = _Alloc()) [inline],[explicit]`

Creates a forward_list with default constructed elements.

Parameters

<code>__n</code>	The number of elements to initially create.
------------------	---

This constructor creates the forward_list with `__n` default constructed elements.

Definition at line 477 of file forward_list.h.

4.553.2.5 `template<typename _Tp, typename _Alloc = allocator<_Tp>> std::forward_list<_Tp, _Alloc>::forward_list (size_type __n, const _Tp & __value, const _Alloc & __al = _Alloc()) [inline]`

Creates a forward_list with copies of an exemplar element.

Parameters

<code>__n</code>	The number of elements to initially create.
<code>__value</code>	An element to copy.
<code>__al</code>	An allocator object.

This constructor fills the forward_list with `__n` copies of `__value`.

Definition at line 490 of file forward_list.h.

4.553.2.6 `template<typename _Tp, typename _Alloc = allocator<_Tp>> template<typename _InputIterator, typename = std::RequireInputIter<_InputIterator>>> std::forward_list<_Tp, _Alloc>::forward_list (_InputIterator __first, _InputIterator __last, const _Alloc & __al = _Alloc()) [inline]`

Builds a forward_list from a range.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__al</code>	An allocator object.

Create a forward_list consisting of copies of the elements from [`__first`,`__last`). This is linear in N (where N is distance(`__first`,`__last`)).

Definition at line 507 of file forward_list.h.

4.553.2.7 `template<typename _Tp, typename _Alloc = allocator<_Tp>> std::forward_list<_Tp, _Alloc>::forward_list (const forward_list<_Tp, _Alloc> & __list) [inline]`

The forward_list copy constructor.

Parameters

<code>__list</code>	A forward_list of identical element and allocator types.
---------------------	--

Definition at line 517 of file forward_list.h.

References `std::forward_list<_Tp, _Alloc>::begin()`, and `std::forward_list<_Tp, _Alloc>::end()`.

4.553.2.8 `template<typename _Tp, typename _Alloc = allocator<_Tp>> std::forward_list<_Tp, _Alloc>::forward_list (forward_list<_Tp, _Alloc> && __list) [inline],[noexcept]`

The forward_list move constructor.

Parameters

<code>__list</code>	A forward_list of identical element and allocator types.
---------------------	--

The newly-created forward_list contains the exact contents of `__list`. The contents of `__list` are a valid, but unspecified forward_list.

Definition at line 531 of file forward_list.h.

4.553.2.9 `template<typename _Tp, typename _Alloc = allocator<_Tp>> std::forward_list<_Tp, _Alloc>::forward_list (std::initializer_list<_Tp> __il, const _Alloc & __al = _Alloc()) [inline]`

Builds a forward_list from an initializer_list.

Parameters

<code>__il</code>	An initializer_list of value_type.
<code>__al</code>	An allocator object.

Create a forward_list consisting of copies of the elements in the initializer_list `__il`. This is linear in `__il.size()`.

Definition at line 542 of file forward_list.h.

4.553.2.10 `template<typename _Tp, typename _Alloc = allocator<_Tp>> std::forward_list<_Tp, _Alloc>::~forward_list () [inline],[noexcept]`

The forward_list dtor.

Definition at line 550 of file forward_list.h.

4.553.3 Member Function Documentation

4.553.3.1 `template<typename _Tp, typename _Alloc = allocator<_Tp>> template<typename _InputIterator, typename = std::RequireInputIter<_InputIterator>> void std::forward_list<_Tp, _Alloc>::assign (_InputIterator __first, _InputIterator __last) [inline]`

Assigns a range to a forward_list.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.

This function fills a forward_list with copies of the elements in the range [`__first`,`__last`).

Note that the assignment completely changes the forward_list and that the number of elements of the resulting forward_list is the same as the number of elements assigned. Old data is lost.

Definition at line 615 of file forward_list.h.

4.553.3.2 `template<typename _Tp, typename _Alloc = allocator<_Tp>> void std::forward_list<_Tp, _Alloc>::assign (size_type __n, const _Tp & __val) [inline]`

Assigns a given value to a forward_list.

Parameters

<code>__n</code>	Number of elements to be assigned.
<code>__val</code>	Value to be assigned.

This function fills a forward_list with `__n` copies of the given value. Note that the assignment completely changes the forward_list, and that the resulting forward_list has `__n` elements. Old data is lost.

Definition at line 632 of file forward_list.h.

4.553.3.3 `template<typename _Tp, typename _Alloc = allocator<_Tp>> void std::forward_list<_Tp, _Alloc>::assign (std::initializer_list<_Tp> __il) [inline]`

Assigns an initializer_list to a forward_list.

Parameters

<code>__il</code>	An initializer_list of value_type.
-------------------	------------------------------------

Replace the contents of the forward_list with copies of the elements in the initializer_list `__il`. This is linear in `il.size()`.

Definition at line 644 of file forward_list.h.

4.553.3.4 `template<typename _Tp, typename _Alloc = allocator<_Tp>> iterator std::forward_list<_Tp, _Alloc>::before_begin () [inline], [noexcept]`

Returns a read/write iterator that points before the first element in the `forward_list`. Iteration is done in ordinary element order.

Definition at line 659 of file `forward_list.h`.

Referenced by `std::forward_list<_Tp, _Alloc>::splice_after()`.

4.553.3.5 `template<typename _Tp, typename _Alloc = allocator<_Tp>> const_iterator std::forward_list<_Tp, _Alloc>::before_begin () const [inline], [noexcept]`

Returns a read-only (constant) iterator that points before the first element in the `forward_list`. Iteration is done in ordinary element order.

Definition at line 668 of file `forward_list.h`.

4.553.3.6 `template<typename _Tp, typename _Alloc = allocator<_Tp>> iterator std::forward_list<_Tp, _Alloc>::begin () [inline], [noexcept]`

Returns a read/write iterator that points to the first element in the `forward_list`. Iteration is done in ordinary element order.

Definition at line 676 of file `forward_list.h`.

Referenced by `std::forward_list<_Tp, _Alloc>::forward_list()`, and `std::forward_list<_Tp, _Alloc>::reverse()`.

4.553.3.7 `template<typename _Tp, typename _Alloc = allocator<_Tp>> const_iterator std::forward_list<_Tp, _Alloc>::begin () const [inline], [noexcept]`

Returns a read-only (constant) iterator that points to the first element in the `forward_list`. Iteration is done in ordinary element order.

Definition at line 685 of file `forward_list.h`.

4.553.3.8 `template<typename _Tp, typename _Alloc = allocator<_Tp>> const_iterator std::forward_list<_Tp, _Alloc>::cbefore_begin () const [inline], [noexcept]`

Returns a read-only (constant) iterator that points before the first element in the `forward_list`. Iteration is done in ordinary element order.

Definition at line 721 of file `forward_list.h`.

4.553.3.9 `template<typename _Tp, typename _Alloc = allocator<_Tp>> const_iterator std::forward_list<_Tp, _Alloc>::cbegin () const [inline], [noexcept]`

Returns a read-only (constant) iterator that points to the first element in the `forward_list`. Iteration is done in ordinary element order.

Definition at line 712 of file `forward_list.h`.

Referenced by `std::operator<()`.

4.553.3.10 `template<typename _Tp, typename _Alloc = allocator<_Tp>> const_iterator std::forward_list<_Tp, _Alloc>::cend() const` `[inline], [noexcept]`

Returns a read-only (constant) iterator that points one past the last element in the forward_list. Iteration is done in ordinary element order.

Definition at line 730 of file forward_list.h.

Referenced by std::operator<().

4.553.3.11 `template<typename _Tp, typename _Alloc = allocator<_Tp>> void std::forward_list<_Tp, _Alloc>::clear()` `[inline], [noexcept]`

Erases all the elements.

Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1033 of file forward_list.h.

4.553.3.12 `template<typename _Tp, typename _Alloc = allocator<_Tp>> template<typename... _Args> iterator std::forward_list<_Tp, _Alloc>::emplace_after(const_iterator __pos, _Args &&... __args)` `[inline]`

Constructs object in forward_list after the specified iterator.

Parameters

<code>__pos</code>	A const_iterator into the forward_list.
<code>__args</code>	Arguments.

Returns

An iterator that points to the inserted data.

This function will insert an object of type T constructed with T(std::forward<Args>(args)...) after the specified location. Due to the nature of a forward_list this operation can be done in constant time, and does not invalidate iterators and references.

Definition at line 843 of file forward_list.h.

4.553.3.13 `template<typename _Tp, typename _Alloc = allocator<_Tp>> template<typename... _Args> void std::forward_list<_Tp, _Alloc>::emplace_front(_Args &&... __args)` `[inline]`

Constructs object in forward_list at the front of the list.

Parameters

<code>__args</code>	Arguments.
---------------------	------------

This function will insert an object of type `Tp` constructed with `Tp(std::forward<Args>(args)...) at the front of the list. Due to the nature of a forward_list this operation can be done in constant time, and does not invalidate iterators and references.`

Definition at line 787 of file `forward_list.h`.

4.553.3.14 `template<typename _Tp, typename _Alloc = allocator<_Tp>> bool std::forward_list<_Tp, _Alloc>::empty ()`
`const [inline], [noexcept]`

Returns true if the `forward_list` is empty. (Thus `begin()` would equal `end()`.)

Definition at line 738 of file `forward_list.h`.

Referenced by `std::forward_list<_Tp, _Alloc>::splice_after()`.

4.553.3.15 `template<typename _Tp, typename _Alloc = allocator<_Tp>> iterator std::forward_list<_Tp, _Alloc>::end ()`
`[inline], [noexcept]`

Returns a read/write iterator that points one past the last element in the `forward_list`. Iteration is done in ordinary element order.

Definition at line 694 of file `forward_list.h`.

Referenced by `std::forward_list<_Tp, _Alloc>::forward_list()`, `std::forward_list<_Tp, _Alloc>::reverse()`, and `std::forward_list<_Tp, _Alloc>::splice_after()`.

4.553.3.16 `template<typename _Tp, typename _Alloc = allocator<_Tp>> const_iterator std::forward_list<_Tp, _Alloc>`
`>::end () const [inline], [noexcept]`

Returns a read-only iterator that points one past the last element in the `forward_list`. Iteration is done in ordinary element order.

Definition at line 703 of file `forward_list.h`.

4.553.3.17 `template<typename _Tp, typename _Alloc = allocator<_Tp>> iterator std::forward_list<_Tp, _Alloc>`
`>::erase_after (const_iterator __pos) [inline]`

Removes the element pointed to by the iterator following `pos`.

Parameters

<code>__pos</code>	Iterator pointing before element to be erased.
--------------------	--

Returns

An iterator pointing to the element following the one that was erased, or `end()` if no such element exists.

This function will erase the element at the given position and thus shorten the `forward_list` by one.

Due to the nature of a forward_list this operation can be done in constant time, and only invalidates iterators/references to the element being removed. The user is also cautioned that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 946 of file forward_list.h.

```
4.553.3.18 template<typename _Tp, typename _Alloc = allocator<_Tp>> iterator std::forward_list< _Tp, _Alloc
>::erase_after ( const_iterator __pos, const_iterator __last ) [inline]
```

Remove a range of elements.

Parameters

<code>__pos</code>	Iterator pointing before the first element to be erased.
<code>__last</code>	Iterator pointing to one past the last element to be erased.

Returns

@ `__last`.

This function will erase the elements in the range (`__pos`,`__last`) and shorten the forward_list accordingly.

This operation is linear time in the size of the range and only invalidates iterators/references to the element being removed. The user is also cautioned that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 969 of file forward_list.h.

```
4.553.3.19 template<typename _Tp, typename _Alloc = allocator<_Tp>> reference std::forward_list< _Tp, _Alloc >::front ( )
[inline]
```

Returns a read/write reference to the data at the first element of the forward_list.

Definition at line 755 of file forward_list.h.

```
4.553.3.20 template<typename _Tp, typename _Alloc = allocator<_Tp>> const_reference std::forward_list< _Tp, _Alloc
>::front ( ) const [inline]
```

Returns a read-only (constant) reference to the data at the first element of the forward_list.

Definition at line 766 of file forward_list.h.

```
4.553.3.21 template<typename _Tp, typename _Alloc = allocator<_Tp>> allocator_type std::forward_list< _Tp, _Alloc
>::get_allocator ( ) const [inline], [noexcept]
```

Get a copy of the memory allocation object.

Definition at line 649 of file forward_list.h.

```
4.553.3.22 template<typename _Tp, typename _Alloc = allocator<_Tp>> iterator std::forward_list< _Tp, _Alloc
>::insert_after ( const_iterator __pos, const _Tp & __val ) [inline]
```

Inserts given value into forward_list after specified iterator.

Parameters

<code>__pos</code>	An iterator into the <code>forward_list</code> .
<code>__val</code>	Data to be inserted.

Returns

An iterator that points to the inserted data.

This function will insert a copy of the given value after the specified location. Due to the nature of a `forward_list` this operation can be done in constant time, and does not invalidate iterators and references.

Definition at line 860 of file `forward_list.h`.

References `std::move()`.

```
4.553.3.23  template<typename _Tp, typename _Alloc = allocator<_Tp>> iterator std::forward_list< _Tp, _Alloc
>::insert_after( const_iterator __pos, size_type __n, const _Tp & __val )
```

Inserts a number of copies of given data into the `forward_list`.

Parameters

<code>__pos</code>	An iterator into the <code>forward_list</code> .
<code>__n</code>	Number of elements to be inserted.
<code>__val</code>	Data to be inserted.

Returns

An iterator pointing to the last inserted copy of `val` or `pos` if `n == 0`.

This function will insert a specified number of copies of the given data after the location specified by `pos`.

This operation is linear in the number of elements inserted and does not invalidate iterators and references.

```
4.553.3.24  template<typename _Tp, typename _Alloc = allocator<_Tp>> template<typename _InputIterator, typename =
std::_RequireInputIter<_InputIterator>> iterator std::forward_list< _Tp, _Alloc >::insert_after( const_iterator
__pos, _InputIterator __first, _InputIterator __last )
```

Inserts a range into the `forward_list`.

Parameters

<code>__pos</code>	An iterator into the <code>forward_list</code> .
<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.

Returns

An iterator pointing to the last inserted element or `__pos` if `__first == __last`.

This function will insert copies of the data in the range `[__first, __last)` into the `forward_list` after the location specified by `__pos`.

This operation is linear in the number of elements inserted and does not invalidate iterators and references.

4.553.3.25 `template<typename _Tp, typename _Alloc = allocator<_Tp>> iterator std::forward_list< _Tp, _Alloc >::insert_after (const_iterator __pos, std::initializer_list<_Tp> __il) [inline]`

Inserts the contents of an `initializer_list` into `forward_list` after the specified iterator.

Parameters

<code>__pos</code>	An iterator into the <code>forward_list</code> .
<code>__il</code>	An <code>initializer_list</code> of value_type.

Returns

An iterator pointing to the last inserted element or `__pos` if `__il` is empty.

This function will insert copies of the data in the `initializer_list` `__il` into the `forward_list` before the location specified by `__pos`.

This operation is linear in the number of elements inserted and does not invalidate iterators and references.

Definition at line 925 of file `forward_list.h`.

4.553.3.26 `template<typename _Tp, typename _Alloc = allocator<_Tp>> size_type std::forward_list< _Tp, _Alloc >::max_size () const [inline], [noexcept]`

Returns the largest possible number of elements of `forward_list`.

Definition at line 745 of file `forward_list.h`.

4.553.3.27 `template<typename _Tp, typename _Alloc = allocator<_Tp>> void std::forward_list< _Tp, _Alloc >::merge (forward_list< _Tp, _Alloc > && __list) [inline]`

Merge sorted lists.

Parameters

<code>__list</code>	Sorted list to merge.
---------------------	-----------------------

Assumes that both `list` and this list are sorted according to operator<(). Merges elements of `__list` into this list in sorted order, leaving `__list` empty when complete. Elements in this list precede elements in `__list` that are equal.

Definition at line 1171 of file forward_list.h.

References `std::move()`.

4.553.3.28 `template<typename _Tp, typename _Alloc = allocator<_Tp>> template<typename _Comp > void
std::forward_list<_Tp, _Alloc>::merge (forward_list<_Tp, _Alloc> && __list, _Comp __comp)`

Merge sorted lists according to comparison function.

Parameters

<code>__list</code>	Sorted list to merge.
<code>__comp</code>	Comparison function defining sort order.

Assumes that both `__list` and this list are sorted according to `comp`. Merges elements of `__list` into this list in sorted order, leaving `__list` empty when complete. Elements in this list precede elements in `__list` that are equivalent according to `comp()`.

4.553.3.29 `template<typename _Tp, typename _Alloc = allocator<_Tp>> forward_list& std::forward_list<_Tp, _Alloc>
>::operator= (const forward_list<_Tp, _Alloc> & __list)`

The `forward_list` assignment operator.

Parameters

<code>__list</code>	A <code>forward_list</code> of identical element and allocator types.
---------------------	---

All the elements of `__list` are copied, but unlike the copy constructor, the allocator object is not copied.

4.553.3.30 `template<typename _Tp, typename _Alloc = allocator<_Tp>> forward_list& std::forward_list<_Tp, _Alloc>
>::operator= (forward_list<_Tp, _Alloc> && __list) [inline], [noexcept]`

The `forward_list` move assignment operator.

Parameters

<code>__list</code>	A <code>forward_list</code> of identical element and allocator types.
---------------------	---

The contents of `__list` are moved into this `forward_list` (without copying, if the allocators permit it). `__list` is a valid, but unspecified `forward_list`

Definition at line 574 of file forward_list.h.

References `std::move()`.

4.553.3.31 `template<typename _Tp, typename _Alloc = allocator<_Tp>> forward_list& std::forward_list<_Tp, _Alloc>
>::operator= (std::initializer_list<_Tp> __il) [inline]`

The `forward_list` initializer list assignment operator.

Parameters

<code>__il</code>	An initializer_list of value_type.
-------------------	------------------------------------

Replace the contents of the forward_list with copies of the elements in the initializer_list `__il`. This is linear in `__il.size()`.

Definition at line 594 of file forward_list.h.

4.553.3.32 `template<typename _Tp, typename _Alloc = allocator<_Tp>> void std::forward_list<_Tp, _Alloc>::pop_front ()`
`[inline]`

Removes first element.

This is a typical stack operation. It shrinks the forward_list by one. Due to the nature of a forward_list this operation can be done in constant time, and only invalidates iterators/references to the element being removed.

Note that no data is returned, and if the first element's data is needed, it should be retrieved before `pop_front()` is called.

Definition at line 825 of file forward_list.h.

4.553.3.33 `template<typename _Tp, typename _Alloc = allocator<_Tp>> void std::forward_list<_Tp, _Alloc>::push_front (`
`const _Tp &__val) [inline]`

Add data to the front of the forward_list.

Parameters

<code>__val</code>	Data to be added.
--------------------	-------------------

This is a typical stack operation. The function creates an element at the front of the forward_list and assigns the given data to it. Due to the nature of a forward_list this operation can be done in constant time, and does not invalidate iterators and references.

Definition at line 802 of file forward_list.h.

References `std::move()`.

4.553.3.34 `template<typename _Tp, typename _Alloc = allocator<_Tp>> void std::forward_list<_Tp, _Alloc>::remove (`
`const _Tp &__val)`

Remove all elements equal to value.

Parameters

<code>__val</code>	The value to remove.
--------------------	----------------------

Removes every element in the list equal to `__val`. Remaining elements stay in list order. Note that this function only

erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

4.553.3.35 `template<typename _Tp, typename _Alloc = allocator<_Tp>> template<typename _Pred > void std::forward_list<_Tp, _Alloc>::remove_if (_Pred __pred)`

Remove all elements satisfying a predicate.

Parameters

<code>__pred</code>	Unary predicate function or object.
---------------------	-------------------------------------

Removes every element in the list for which the predicate returns true. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

4.553.3.36 `template<typename _Tp, typename _Alloc = allocator<_Tp>> void std::forward_list<_Tp, _Alloc>::resize (size_type __sz)`

Resizes the `forward_list` to the specified number of elements.

Parameters

<code>__sz</code>	Number of elements the <code>forward_list</code> should contain.
-------------------	--

This function will resize the `forward_list` to the specified number of elements. If the number is smaller than the `forward_list`'s current number of elements the `forward_list` is truncated, otherwise the `forward_list` is extended and the new elements are default constructed.

4.553.3.37 `template<typename _Tp, typename _Alloc = allocator<_Tp>> void std::forward_list<_Tp, _Alloc>::resize (size_type __sz, const value_type & __val)`

Resizes the `forward_list` to the specified number of elements.

Parameters

<code>__sz</code>	Number of elements the <code>forward_list</code> should contain.
<code>__val</code>	Data with which new elements should be populated.

This function will resize the `forward_list` to the specified number of elements. If the number is smaller than the `forward_list`'s current number of elements the `forward_list` is truncated, otherwise the `forward_list` is extended and new elements are populated with given data.

4.553.3.38 `template<typename _Tp, typename _Alloc = allocator<_Tp>> void std::forward_list<_Tp, _Alloc>::reverse () [inline], [noexcept]`

Reverse the elements in list.

Reverse the order of elements in the list in linear time.

Definition at line 1224 of file forward_list.h.

References std::begin(), std::forward_list< _Tp, _Alloc >::begin(), std::end(), std::forward_list< _Tp, _Alloc >::end(), std::move(), and std::swap().

4.553.3.39 `template<typename _Tp, typename _Alloc = allocator<_Tp>> void std::forward_list< _Tp, _Alloc >::sort ()`
`[inline]`

Sort the elements of the list.

Sorts the elements of this list in NlogN time. Equivalent elements remain in list order.

Definition at line 1205 of file forward_list.h.

4.553.3.40 `template<typename _Tp, typename _Alloc = allocator<_Tp>> template<typename _Comp > void`
`std::forward_list< _Tp, _Alloc >::sort (_Comp __comp)`

Sort the forward_list using a comparison function.

Sorts the elements of this list in NlogN time. Equivalent elements remain in list order.

4.553.3.41 `template<typename _Tp, typename _Alloc = allocator<_Tp>> void std::forward_list< _Tp, _Alloc >::splice_after (`
`const_iterator __pos, forward_list< _Tp, _Alloc > && __list) [inline]`

Insert contents of another forward_list.

Parameters

<code>__pos</code>	Iterator referencing the element to insert after.
<code>__list</code>	Source list.

The elements of *list* are inserted in constant time after the element referenced by *pos*. *list* becomes an empty list.

Requires this != x.

Definition at line 1050 of file forward_list.h.

References std::forward_list< _Tp, _Alloc >::before_begin(), std::forward_list< _Tp, _Alloc >::empty(), std::forward_list< _Tp, _Alloc >::end(), and std::move().

4.553.3.42 `template<typename _Tp, typename _Alloc = allocator<_Tp>> void std::forward_list< _Tp, _Alloc >::splice_after (`
`const_iterator __pos, forward_list< _Tp, _Alloc > && __list, const_iterator __i)`

Insert element from another forward_list.

Parameters

<code>__pos</code>	Iterator referencing the element to insert after.
<code>__list</code>	Source list.
<code>__i</code>	Iterator referencing the element before the element to move.

Removes the element in list *list* referenced by *i* and inserts it into the current list after *pos*.

4.553.3.43 `template<typename _Tp, typename _Alloc = allocator<_Tp>> void std::forward_list<_Tp, _Alloc>::splice_after (const_iterator __pos, forward_list<_Tp, _Alloc> &&, const_iterator __before, const_iterator __last) [inline]`

Insert range from another forward_list.

Parameters

<code>__pos</code>	Iterator referencing the element to insert after.
<code>__list</code>	Source list.
<code>__before</code>	Iterator referencing before the start of range in list.
<code>__last</code>	Iterator referencing the end of range in list.

Removes elements in the range (`__before`,`__last`) and inserts them after `__pos` in constant time.

Undefined if `__pos` is in (`__before`,`__last`).

Definition at line 1093 of file forward_list.h.

4.553.3.44 `template<typename _Tp, typename _Alloc = allocator<_Tp>> void std::forward_list<_Tp, _Alloc>::swap (forward_list<_Tp, _Alloc> & __list) [inline], [noexcept]`

Swaps data with another forward_list.

Parameters

<code>__list</code>	A forward_list of the same element and allocator types.
---------------------	---

This exchanges the elements between two lists in constant time. Note that the global `std::swap()` function is specialized such that `std::swap(l1,l2)` will feed to this function.

Definition at line 986 of file forward_list.h.

References `std::swap()`.

Referenced by `std::swap()`.

4.553.3.45 `template<typename _Tp, typename _Alloc = allocator<_Tp>> void std::forward_list<_Tp, _Alloc>::unique ()`
`[inline]`

Remove consecutive duplicate elements.

For each consecutive set of elements with the same value, remove all but the first one. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1142 of file `forward_list.h`.

4.553.3.46 `template<typename _Tp, typename _Alloc = allocator<_Tp>> template<typename _BinPred > void`
`std::forward_list<_Tp, _Alloc>::unique (_BinPred __binary_pred)`

Remove consecutive elements satisfying a predicate.

Parameters

<code>__binary_pred</code>	Binary predicate function or object.
----------------------------	--------------------------------------

For each consecutive set of elements `[first,last)` that satisfy `predicate(first,i)` where `i` is an iterator in `[first,last)`, remove all but the first one. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

The documentation for this class was generated from the following file:

- [forward_list.h](#)

4.554 `std::fpos<_StateT>` Class Template Reference

Public Member Functions

- [fpos](#) ([streamoff](#) __off)
- [operator streamoff](#) () const
- [fpos operator+](#) ([streamoff](#) __off) const
- [fpos & operator+=](#) ([streamoff](#) __off)
- [fpos operator-](#) ([streamoff](#) __off) const
- [streamoff operator-](#) (const [fpos](#) & __other) const
- [fpos & operator-=](#) ([streamoff](#) __off)
- void [state](#) ([_StateT](#) __st)
- [_StateT state](#) () const

4.554.1 Detailed Description

```
template<typename _StateT>  
class std::fpos<_StateT >
```

Class representing stream positions.

The standard places no requirements upon the template parameter `StateT`. In this implementation `StateT` must be `DefaultConstructible`, `CopyConstructible` and `Assignable`. The standard only requires that `fpos` should contain a member of type `StateT`. In this implementation it also contains an offset stored as a signed integer.

Parameters

<i>StateT</i>	Type passed to and returned from state().
---------------	---

Definition at line 112 of file postypes.h.

4.554.2 Constructor & Destructor Documentation

4.554.2.1 `template<typename _StateT> std::fpos<_StateT>::fpos (streamoff __off) [inline]`

Construct position from offset.

Definition at line 133 of file postypes.h.

4.554.3 Member Function Documentation

4.554.3.1 `template<typename _StateT> std::fpos<_StateT>::operator streamoff () const [inline]`

Convert to streamoff.

Definition at line 137 of file postypes.h.

4.554.3.2 `template<typename _StateT> fpos std::fpos<_StateT>::operator+ (streamoff __off) const [inline]`

Add position and offset.

Definition at line 178 of file postypes.h.

4.554.3.3 `template<typename _StateT> fpos& std::fpos<_StateT>::operator+= (streamoff __off) [inline]`

Add offset to this position.

Definition at line 154 of file postypes.h.

4.554.3.4 `template<typename _StateT> fpos std::fpos<_StateT>::operator- (streamoff __off) const [inline]`

Subtract offset from position.

Definition at line 192 of file postypes.h.

4.554.3.5 `template<typename _StateT> streamoff std::fpos<_StateT>::operator- (const fpos<_StateT> & __other) const [inline]`

Subtract position to return offset.

Definition at line 205 of file postypes.h.

4.554.3.6 `template<typename _StateT> fpos& std::fpos<_StateT>::operator=(streamoff __off) [inline]`

Subtract offset from this position.

Definition at line 165 of file postypes.h.

4.554.3.7 `template<typename _StateT> void std::fpos<_StateT>::state (_StateT __st) [inline]`

Remember the value of *st*.

Definition at line 141 of file postypes.h.

4.554.3.8 `template<typename _StateT> _StateT std::fpos<_StateT>::state () const [inline]`

Return the last set value of *st*.

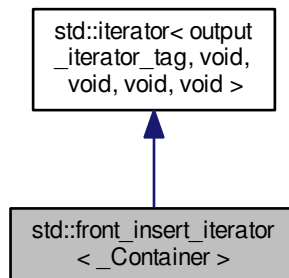
Definition at line 146 of file postypes.h.

The documentation for this class was generated from the following file:

- [postypes.h](#)

4.555 `std::front_insert_iterator<_Container>` Class Template Reference

Inheritance diagram for `std::front_insert_iterator<_Container>`:



Public Types

- typedef `_Container` [container_type](#)
- typedef void [difference_type](#)
- typedef [output_iterator_tag](#) [iterator_category](#)
- typedef void [pointer](#)
- typedef void [reference](#)
- typedef void [value_type](#)

Public Member Functions

- [front_insert_iterator](#) (_Container &__x)
- [front_insert_iterator](#) & [operator*](#) ()
- [front_insert_iterator](#) & [operator++](#) ()
- [front_insert_iterator](#) [operator++](#) (int)
- [front_insert_iterator](#) & [operator=](#) (const typename _Container::value_type &__value)
- [front_insert_iterator](#) & [operator=](#) (typename _Container::value_type &&__value)

Protected Attributes

- _Container * **container**

4.555.1 Detailed Description

```
template<typename _Container>
class std::front_insert_iterator<_Container>
```

Turns assignment into insertion.

These are output iterators, constructed from a container-of-T. Assigning a T to the iterator prepends it to the container using `push_front`.

Tip: Using the `front_inserter` function to create these iterators can save typing.

Definition at line 494 of file `stl_iterator.h`.

4.555.2 Member Typedef Documentation

4.555.2.1 `template<typename _Container> typedef _Container std::front_insert_iterator<_Container>::container_type`

A nested typedef for the type of whatever container you used.

Definition at line 502 of file `stl_iterator.h`.

4.555.2.2 `typedef void std::iterator< output_iterator_tag, void, void, void, void>::difference_type` `[inherited]`

Distance between iterators is represented as this type.

Definition at line 125 of file `stl_iterator_base_types.h`.

4.555.2.3 `typedef output_iterator_tag std::iterator< output_iterator_tag, void, void, void, void>::iterator_category` `[inherited]`

One of the [tag types](#).

Definition at line 121 of file `stl_iterator_base_types.h`.

4.555.2.4 `typedef void std::iterator< output_iterator_tag , void , void , void , void >::pointer` [inherited]

This type represents a pointer-to-value_type.

Definition at line 127 of file `stl_iterator_base_types.h`.

4.555.2.5 `typedef void std::iterator< output_iterator_tag , void , void , void , void >::reference` [inherited]

This type represents a reference-to-value_type.

Definition at line 129 of file `stl_iterator_base_types.h`.

4.555.2.6 `typedef void std::iterator< output_iterator_tag , void , void , void , void >::value_type` [inherited]

The type "pointed to" by the iterator.

Definition at line 123 of file `stl_iterator_base_types.h`.

4.555.3 Constructor & Destructor Documentation

4.555.3.1 `template<typename _Container > std::front_insert_iterator< _Container >::front_insert_iterator (_Container &__x)` [inline],[explicit]

The only way to create this iterator is with a container.

Definition at line 505 of file `stl_iterator.h`.

4.555.4 Member Function Documentation

4.555.4.1 `template<typename _Container > front_insert_iterator& std::front_insert_iterator< _Container >::operator* ()` [inline]

Simply returns `*this`.

Definition at line 543 of file `stl_iterator.h`.

4.555.4.2 `template<typename _Container > front_insert_iterator& std::front_insert_iterator< _Container >::operator++ ()` [inline]

Simply returns `*this`. (This iterator does not *move*.)

Definition at line 548 of file `stl_iterator.h`.

4.555.4.3 `template<typename _Container > front_insert_iterator std::front_insert_iterator< _Container >::operator++ (int)` [inline]

Simply returns `*this`. (This iterator does not *move*.)

Definition at line 553 of file `stl_iterator.h`.

4.555.4.4 `template<typename _Container > front_insert_iterator& std::front_insert_iterator< _Container >::operator= (const typename _Container::value_type &__value)` [inline]

Parameters

<code>__value</code>	An instance of whatever type <code>container_type::const_reference</code> is; presumably a reference-to-const T for <code>container<T></code> .
----------------------	---

Returns

This iterator, for chained operations.

This kind of iterator doesn't really have a *position* in the container (you can think of the position as being permanently at the front, if you like). Assigning a value to the iterator will always prepend the value to the front of the container.

Definition at line 527 of file `stl_iterator.h`.

References `std::move()`.

The documentation for this class was generated from the following file:

- [stl_iterator.h](#)

4.556 std::gamma_distribution<_RealType> Class Template Reference

Classes

- struct [param_type](#)

Public Types

- typedef `_RealType` [result_type](#)

Public Member Functions

- [gamma_distribution](#) (`_RealType __alpha_val=_RealType(1), _RealType __beta_val=_RealType(1)`)
- [gamma_distribution](#) (const [param_type](#) &__p)
- `template<typename _ForwardIterator, typename _UniformRandomNumberGenerator >`
void [__generate](#) (`_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng`)
- `template<typename _ForwardIterator, typename _UniformRandomNumberGenerator >`
void [__generate](#) (`_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param_type &__p`)
- `template<typename _UniformRandomNumberGenerator >`
void [__generate](#) (`result_type *__f, result_type *__t, _UniformRandomNumberGenerator &__urng, const param_type &__p`)
- `_RealType alpha () const`
- `_RealType beta () const`
- `result_type max () const`
- `result_type min () const`
- `template<typename _UniformRandomNumberGenerator >`
`result_type operator\(\) (_UniformRandomNumberGenerator &__urng)`
- `template<typename _UniformRandomNumberGenerator >`
`result_type operator\(\) (_UniformRandomNumberGenerator &__urng, const param_type &__p)`
- `param_type param () const`
- void [param](#) (const [param_type](#) &__param)
- void [reset](#) ()

Friends

- `template<typename _RealType1 , typename _CharT , typename _Traits >
std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const
std::gamma_distribution< _RealType1 > &__x)`
- `bool operator== (const gamma_distribution &__d1, const gamma_distribution &__d2)`
- `template<typename _RealType1 , typename _CharT , typename _Traits >
std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::
::gamma_distribution< _RealType1 > &__x)`

4.556.1 Detailed Description

```
template<typename _RealType = double>
class std::gamma_distribution< _RealType >
```

A gamma continuous distribution for random numbers.

The formula for the gamma probability density function is:

$$p(x|\alpha, \beta) = \frac{1}{\beta\Gamma(\alpha)} (x/\beta)^{\alpha-1} e^{-x/\beta}$$

Definition at line 2504 of file random.h.

4.556.2 Member Typedef Documentation

4.556.2.1 `template<typename _RealType = double> typedef _RealType std::gamma_distribution< _RealType
>::result_type`

The type of the range of the distribution.

Definition at line 2507 of file random.h.

4.556.3 Constructor & Destructor Documentation

4.556.3.1 `template<typename _RealType = double> std::gamma_distribution< _RealType >::gamma_distribution
(_RealType __alpha_val = _RealType(1), _RealType __beta_val = _RealType(1)) [inline],
[explicit]`

Constructs a gamma distribution with parameters α and β .

Definition at line 2556 of file random.h.

4.556.4 Member Function Documentation

4.556.4.1 `template<typename _RealType = double> _RealType std::gamma_distribution<_RealType>::alpha () const`
[inline]

Returns the α of the distribution.

Definition at line 2577 of file random.h.

4.556.4.2 `template<typename _RealType = double> _RealType std::gamma_distribution<_RealType>::beta () const`
[inline]

Returns the β of the distribution.

Definition at line 2584 of file random.h.

4.556.4.3 `template<typename _RealType = double> result_type std::gamma_distribution<_RealType>::max () const`
[inline]

Returns the least upper bound value of the distribution.

Definition at line 2613 of file random.h.

4.556.4.4 `template<typename _RealType = double> result_type std::gamma_distribution<_RealType>::min () const`
[inline]

Returns the greatest lower bound value of the distribution.

Definition at line 2606 of file random.h.

4.556.4.5 `template<typename _RealType = double> template<typename _UniformRandomNumberGenerator> result_type
std::gamma_distribution<_RealType>::operator() (_UniformRandomNumberGenerator &__urng)` [inline]

Generating functions.

Definition at line 2621 of file random.h.

4.556.4.6 `template<typename _RealType = double> param_type std::gamma_distribution<_RealType>::param () const`
[inline]

Returns the parameter set of the distribution.

Definition at line 2591 of file random.h.

4.556.4.7 `template<typename _RealType = double> void std::gamma_distribution<_RealType>::param (const
param_type &__param)` [inline]

Sets the parameter set of the distribution.

Parameters

<code>__param</code>	The new parameter set of the distribution.
----------------------	--

Definition at line 2599 of file random.h.

4.556.4.8 `template<typename _RealType = double> void std::gamma_distribution<_RealType>::reset() [inline]`

Resets the distribution state.

Definition at line 2570 of file random.h.

4.556.5 Friends And Related Function Documentation

4.556.5.1 `template<typename _RealType = double> template<typename _RealType1, typename _CharT, typename _Traits
> std::basic_ostream<_CharT, _Traits>& operator<<(std::basic_ostream<_CharT, _Traits> & __os, const
std::gamma_distribution<_RealType1> & __x) [friend]`

Inserts a gamma_distribution random number distribution `__x` into the output stream `__os`.

Parameters

<code>__os</code>	An output stream.
<code>__x</code>	A gamma_distribution random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

4.556.5.2 `template<typename _RealType = double> bool operator==(const gamma_distribution<_RealType> & __d1,
const gamma_distribution<_RealType> & __d2) [friend]`

Return true if two gamma distributions have the same parameters and the sequences that would be generated are equal.

Definition at line 2657 of file random.h.

4.556.5.3 `template<typename _RealType = double> template<typename _RealType1, typename _CharT, typename
_Traits> std::basic_istream<_CharT, _Traits>& operator>>(std::basic_istream<_CharT, _Traits> & __is,
std::gamma_distribution<_RealType1> & __x) [friend]`

Extracts a gamma_distribution random number distribution `__x` from the input stream `__is`.

Parameters

<code>__is</code>	An input stream.
<code>__x</code>	A <code>gamma_distribution</code> random number generator engine.

Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following file:

- [random.h](#)

4.557 `std::gamma_distribution<_RealType>::param_type` Struct Reference

Public Types

- typedef `gamma_distribution<_RealType>` **distribution_type**

Public Member Functions

- **param_type** (`_RealType __alpha_val=_RealType(1), _RealType __beta_val=_RealType(1)`)
- `_RealType` **alpha** () const
- `_RealType` **beta** () const

Friends

- class **gamma_distribution<_RealType>**
- bool **operator==** (const `param_type` &__p1, const `param_type` &__p2)

4.557.1 Detailed Description

```
template<typename _RealType = double>
struct std::gamma_distribution<_RealType>::param_type
```

Parameter type.

Definition at line 2513 of file `random.h`.

The documentation for this struct was generated from the following file:

- [random.h](#)

4.558 `std::geometric_distribution<_IntType>` Class Template Reference

Classes

- struct [param_type](#)

Public Types

- typedef `_IntType` [result_type](#)

Public Member Functions

- **`geometric_distribution`** (`double __p=0.5`)
- **`geometric_distribution`** (`const param_type &__p`)
- `template<typename _ForwardIterator, typename _UniformRandomNumberGenerator >`
`void __generate (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng)`
- `template<typename _ForwardIterator, typename _UniformRandomNumberGenerator >`
`void __generate (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param_type &__p)`
- `template<typename _UniformRandomNumberGenerator >`
`void __generate (result_type *__f, result_type *__t, _UniformRandomNumberGenerator &__urng, const param_type &__p)`
- [result_type](#) `max () const`
- [result_type](#) `min () const`
- `template<typename _UniformRandomNumberGenerator >`
`result_type operator() (_UniformRandomNumberGenerator &__urng)`
- `template<typename _UniformRandomNumberGenerator >`
`result_type operator() (_UniformRandomNumberGenerator &__urng, const param_type &__p)`
- `double p () const`
- [param_type](#) `param () const`
- `void param (const param_type &__param)`
- `void reset ()`

Friends

- `bool operator== (const geometric_distribution &__d1, const geometric_distribution &__d2)`

4.558.1 Detailed Description

```
template<typename _IntType = int>
class std::geometric_distribution<_IntType>
```

A discrete geometric random number distribution.

The formula for the geometric probability density function is $p(i|p) = p(1 - p)^i$ where p is the parameter of the distribution.

Definition at line 4010 of file `random.h`.

4.558.2 Member Typedef Documentation

4.558.2.1 `template<typename _IntType = int> typedef _IntType std::geometric_distribution<_IntType>::result_type`

The type of the range of the distribution.

Definition at line 4013 of file `random.h`.

4.558.3 Member Function Documentation

4.558.3.1 `template<typename _IntType = int> result_type std::geometric_distribution<_IntType>::max () const` `[inline]`

Returns the least upper bound value of the distribution.

Definition at line 4102 of file `random.h`.

References `std::max()`.

4.558.3.2 `template<typename _IntType = int> result_type std::geometric_distribution<_IntType>::min () const` `[inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 4095 of file `random.h`.

4.558.3.3 `template<typename _IntType = int> template<typename _UniformRandomNumberGenerator> result_type` `std::geometric_distribution<_IntType>::operator() (_UniformRandomNumberGenerator & __urng)` `[inline]`

Generating functions.

Definition at line 4110 of file `random.h`.

4.558.3.4 `template<typename _IntType = int> double std::geometric_distribution<_IntType>::p () const` `[inline]`

Returns the distribution parameter `p`.

Definition at line 4073 of file `random.h`.

4.558.3.5 `template<typename _IntType = int> param_type std::geometric_distribution<_IntType>::param () const` `[inline]`

Returns the parameter set of the distribution.

Definition at line 4080 of file `random.h`.

4.558.3.6 `template<typename _IntType = int> void std::geometric_distribution<_IntType>::param (const param_type &` `__param)` `[inline]`

Sets the parameter set of the distribution.

Parameters

<code>__param</code>	The new parameter set of the distribution.
----------------------	--

Definition at line 4088 of file random.h.

4.558.3.7 `template<typename _IntType = int> void std::geometric_distribution<_IntType>::reset () [inline]`

Resets the distribution state.

Does nothing for the geometric distribution.

Definition at line 4067 of file random.h.

4.558.4 Friends And Related Function Documentation

4.558.4.1 `template<typename _IntType = int> bool operator== (const geometric_distribution<_IntType> &__d1, const geometric_distribution<_IntType> &__d2) [friend]`

Return true if two geometric distributions have the same parameters.

Definition at line 4145 of file random.h.

The documentation for this class was generated from the following file:

- [random.h](#)

4.559 `std::geometric_distribution<_IntType>::param_type` Struct Reference

Public Types

- typedef `geometric_distribution<_IntType>` **distribution_type**

Public Member Functions

- **param_type** (double __p=0.5)
- double **p** () const

Friends

- class `geometric_distribution<_IntType>`
- bool **operator==** (const `param_type` &__p1, const `param_type` &__p2)

4.559.1 Detailed Description

```
template<typename _IntType = int>
struct std::geometric_distribution< _IntType >::param_type
```

Parameter type.

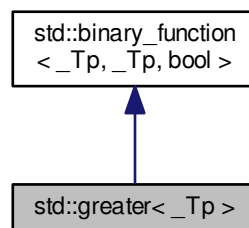
Definition at line 4019 of file `random.h`.

The documentation for this struct was generated from the following file:

- [random.h](#)

4.560 `std::greater<_Tp>` Struct Template Reference

Inheritance diagram for `std::greater<_Tp>`:



Public Types

- typedef `_Tp` [first_argument_type](#)
- typedef `bool` [result_type](#)
- typedef `_Tp` [second_argument_type](#)

Public Member Functions

- `bool` **operator()** (`const _Tp &__x`, `const _Tp &__y`) `const`

4.560.1 Detailed Description

```
template<typename _Tp>
struct std::greater<_Tp>
```

One of the [comparison functors](#).

Definition at line 358 of file `stl_function.h`.

4.560.2 Member Typedef Documentation

4.560.2.1 `typedef _Tp std::binary_function<_Tp, _Tp, bool>::first_argument_type` [inherited]

`first_argument_type` is the type of the first argument

Definition at line 121 of file `stl_function.h`.

4.560.2.2 `typedef bool std::binary_function<_Tp, _Tp, bool>::result_type` [inherited]

`result_type` is the return type

Definition at line 127 of file `stl_function.h`.

4.560.2.3 `typedef _Tp std::binary_function<_Tp, _Tp, bool>::second_argument_type` [inherited]

`second_argument_type` is the type of the second argument

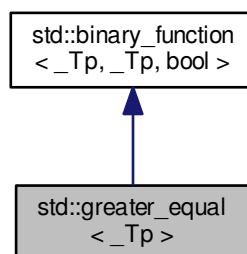
Definition at line 124 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl_function.h](#)

4.561 `std::greater_equal<_Tp>` Struct Template Reference

Inheritance diagram for `std::greater_equal<_Tp>`:



Public Types

- `typedef _Tp` [first_argument_type](#)
- `typedef bool` [result_type](#)
- `typedef _Tp` [second_argument_type](#)

Public Member Functions

- bool **operator()** (const _Tp &__x, const _Tp &__y) const

4.561.1 Detailed Description

```
template<typename _Tp>
struct std::greater_equal< _Tp >
```

One of the [comparison functors](#).

Definition at line 376 of file stl_function.h.

4.561.2 Member Typedef Documentation

4.561.2.1 typedef _Tp std::binary_function< _Tp, _Tp, bool >::first_argument_type [inherited]

first_argument_type is the type of the first argument

Definition at line 121 of file stl_function.h.

4.561.2.2 typedef bool std::binary_function< _Tp, _Tp, bool >::result_type [inherited]

result_type is the return type

Definition at line 127 of file stl_function.h.

4.561.2.3 typedef _Tp std::binary_function< _Tp, _Tp, bool >::second_argument_type [inherited]

second_argument_type is the type of the second argument

Definition at line 124 of file stl_function.h.

The documentation for this struct was generated from the following file:

- [stl_function.h](#)

4.562 std::gslice Class Reference

Public Member Functions

- [gslice](#) ()
- [gslice](#) (size_t __o, const valarray< size_t > &__l, const valarray< size_t > &__s)
- [gslice](#) (const [gslice](#) &)
- [~gslice](#) ()
- [gslice](#) & [operator=](#) (const [gslice](#) &)
- valarray< size_t > [size](#) () const
- size_t [start](#) () const
- valarray< size_t > [stride](#) () const

Friends

- `template<typename _Tp >`
class **valarray**

4.562.1 Detailed Description

Class defining multi-dimensional subset of an array.

The slice class represents a multi-dimensional subset of an array, specified by three parameter sets: start offset, size array, and stride array. The start offset is the index of the first element of the array that is part of the subset. The size and stride array describe each dimension of the slice. Size is the number of elements in that dimension, and stride is the distance in the array between successive elements in that dimension. Each dimension's size and stride is taken to begin at an array element described by the previous dimension. The size array and stride array must be the same size.

For example, if you have `offset==3`, `stride[0]==11`, `size[1]==3`, `stride[1]==3`, then `slice[0,0]==array[3]`, `slice[0,1]==array[6]`, `slice[0,2]==array[9]`, `slice[1,0]==array[14]`, `slice[1,1]==array[17]`, `slice[1,2]==array[20]`.

Definition at line 64 of file `gslice.h`.

The documentation for this class was generated from the following file:

- [gslice.h](#)

4.563 `std::gslice_array<_Tp>` Class Template Reference

Public Types

- typedef `_Tp` **value_type**

Public Member Functions

- [gslice_array](#) (const [gslice_array](#) &)
- void [operator%=>](#) (const valarray<_Tp> &) const
- template<class _Dom>
void **operator%=>** (const _Expr<_Dom, _Tp> &) const
- void [operator&=>](#) (const valarray<_Tp> &) const
- template<class _Dom>
void **operator&=>** (const _Expr<_Dom, _Tp> &) const
- void [operator*=>](#) (const valarray<_Tp> &) const
- template<class _Dom>
void **operator*=>** (const _Expr<_Dom, _Tp> &) const
- void [operator+=>](#) (const valarray<_Tp> &) const
- template<class _Dom>
void **operator+=>** (const _Expr<_Dom, _Tp> &) const
- void [operator-=>](#) (const valarray<_Tp> &) const
- template<class _Dom>
void **operator-=>** (const _Expr<_Dom, _Tp> &) const

- void [operator/](#)= (const valarray< _Tp > &) const
- template<class _Dom >
void **operator/**= (const _Expr< _Dom, _Tp > &) const
- void [operator<<=](#) (const valarray< _Tp > &) const
- template<class _Dom >
void **operator<<=** (const _Expr< _Dom, _Tp > &) const
- [gslice_array](#) & [operator](#)= (const [gslice_array](#) &)
- void [operator](#)= (const valarray< _Tp > &) const
- void [operator](#)= (const _Tp &) const
- template<class _Dom >
void **operator**= (const _Expr< _Dom, _Tp > &) const
- void [operator>>=](#) (const valarray< _Tp > &) const
- template<class _Dom >
void **operator>>=** (const _Expr< _Dom, _Tp > &) const
- void [operator^](#)= (const valarray< _Tp > &) const
- template<class _Dom >
void **operator^**= (const _Expr< _Dom, _Tp > &) const
- void [operator|](#)= (const valarray< _Tp > &) const
- template<class _Dom >
void **operator|**= (const _Expr< _Dom, _Tp > &) const

Friends

- class **valarray**< _Tp >

4.563.1 Detailed Description

```
template<typename _Tp>
class std::gslice_array< _Tp >
```

Reference to multi-dimensional subset of an array.

A `gslice_array` is a reference to the actual elements of an array specified by a `gslice`. The way to get a `gslice_array` is to call `operator[](gslice)` on a `valarray`. The returned `gslice_array` then permits carrying operations out on the referenced subset of elements in the original `valarray`. For example, `operator+=(valarray)` will add values to the subset of elements in the underlying `valarray` this `gslice_array` refers to.

Parameters

<i>Tp</i>	Element type.
-----------	---------------

Definition at line 60 of file `gslice_array.h`.

The documentation for this class was generated from the following file:

- [gslice_array.h](#)

4.564 `std::hash<_Tp>` Struct Template Reference

4.564.1 Detailed Description

```
template<typename _Tp>
struct std::hash<_Tp>
```

Primary class template hash.

Definition at line 58 of file `functional_hash.h`.

The documentation for this struct was generated from the following file:

- [functional_hash.h](#)

4.565 `std::hash<__gnu_cxx::__u16vstring>` Struct Template Reference

Inherits `std::__hash_base<_Result, _Arg>`.

Public Types

- typedef `_Arg` **argument_type**
- typedef `_Result` **result_type**

Public Member Functions

- `size_t operator()` (const [__gnu_cxx::__u16vstring](#) &__s) const noexcept

4.565.1 Detailed Description

```
template<>
struct std::hash<__gnu_cxx::__u16vstring>
```

`std::hash` specialization for `__u16vstring`.

Definition at line 2935 of file `vstring.h`.

The documentation for this struct was generated from the following file:

- [vstring.h](#)

4.566 `std::hash<__gnu_cxx::__u32vstring>` Struct Template Reference

Inherits `std::__hash_base<_Result, _Arg>`.

Public Types

- typedef _Arg **argument_type**
- typedef _Result **result_type**

Public Member Functions

- size_t **operator()** (const [__gnu_cxx::__u32vstring](#) &__s) const noexcept

4.566.1 Detailed Description

```
template<>
struct std::hash< __gnu_cxx::__u32vstring >
```

std::hash specialization for __u32vstring.

Definition at line 2946 of file vstring.h.

The documentation for this struct was generated from the following file:

- [vstring.h](#)

4.567 std::hash< __gnu_cxx::__vstring > Struct Template Reference

Inherits std::__hash_base< _Result, _Arg >.

Public Types

- typedef _Arg **argument_type**
- typedef _Result **result_type**

Public Member Functions

- size_t **operator()** (const [__gnu_cxx::__vstring](#) &__s) const noexcept

4.567.1 Detailed Description

```
template<>
struct std::hash< __gnu_cxx::__vstring >
```

std::hash specialization for __vstring.

Definition at line 2911 of file vstring.h.

The documentation for this struct was generated from the following file:

- [vstring.h](#)

4.568 `std::hash< __gnu_cxx::__wvstring >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

Public Types

- typedef `_Arg` **argument_type**
- typedef `_Result` **result_type**

Public Member Functions

- `size_t operator()` (const `__gnu_cxx::__wvstring` &`_s`) const noexcept

4.568.1 Detailed Description

```
template<>
struct std::hash< __gnu_cxx::__wvstring >
```

`std::hash` specialization for `__wvstring`.

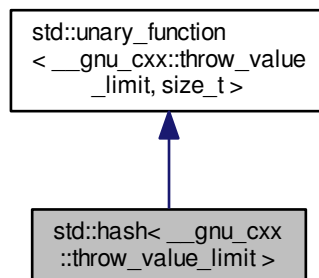
Definition at line 2922 of file `vstring.h`.

The documentation for this struct was generated from the following file:

- [vstring.h](#)

4.569 `std::hash< __gnu_cxx::throw_value_limit >` Struct Template Reference

Inheritance diagram for `std::hash< __gnu_cxx::throw_value_limit >`:



Public Types

- typedef `__gnu_cxx::throw_value_limit` `argument_type`
- typedef `size_t` `result_type`

Public Member Functions

- `size_t operator()` (`const __gnu_cxx::throw_value_limit &__val`) `const`

4.569.1 Detailed Description

```
template<>
struct std::hash<__gnu_cxx::throw_value_limit>
```

Explicit specialization of `std::hash` for `__gnu_cxx::throw_value_limit`.

Definition at line 950 of file `throw_allocator.h`.

4.569.2 Member Typedef Documentation

4.569.2.1 typedef `__gnu_cxx::throw_value_limit` `std::unary_function<__gnu_cxx::throw_value_limit, size_t>::argument_type` `[inherited]`

`argument_type` is the type of the argument

Definition at line 108 of file `stl_function.h`.

4.569.2.2 typedef `size_t` `std::unary_function<__gnu_cxx::throw_value_limit, size_t>::result_type` `[inherited]`

`result_type` is the return type

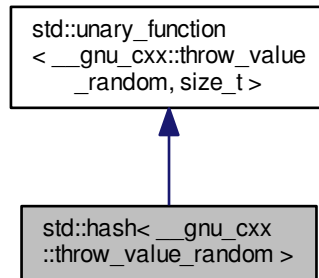
Definition at line 111 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [throw_allocator.h](#)

4.570 `std::hash<__gnu_cxx::throw_value_random>` Struct Template Reference

Inheritance diagram for `std::hash<__gnu_cxx::throw_value_random>`:



Public Types

- typedef `__gnu_cxx::throw_value_random` `argument_type`
- typedef `size_t` `result_type`

Public Member Functions

- `size_t operator()` (const `__gnu_cxx::throw_value_random` &`_val`) const

4.570.1 Detailed Description

```
template<>
struct std::hash<__gnu_cxx::throw_value_random>
```

Explicit specialization of `std::hash` for `__gnu_cxx::throw_value_random`.

Definition at line 965 of file `throw_allocator.h`.

4.570.2 Member Typedef Documentation

4.570.2.1 typedef `__gnu_cxx::throw_value_random` `std::unary_function<__gnu_cxx::throw_value_random, size_t>::argument_type` `[inherited]`

`argument_type` is the type of the argument

Definition at line 108 of file `stl_function.h`.

4.570.2.2 `typedef size_t std::unary_function< __gnu_cxx::throw_value_random, size_t>::result_type`
[inherited]

`result_type` is the return type

Definition at line 111 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [throw_allocator.h](#)

4.571 `std::hash< __shared_ptr< _Tp, _Lp > >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

Public Types

- `typedef _Arg argument_type`
- `typedef _Result result_type`

Public Member Functions

- `size_t operator() (const __shared_ptr< _Tp, _Lp > &__s) const` noexcept

4.571.1 Detailed Description

```
template<typename _Tp, _Lock_policy _Lp>
struct std::hash< __shared_ptr< _Tp, _Lp > >
```

`std::hash` specialization for `__shared_ptr`.

Definition at line 1548 of file `shared_ptr_base.h`.

The documentation for this struct was generated from the following file:

- [shared_ptr_base.h](#)

4.572 `std::hash< _Tp * >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

Public Types

- typedef `_Arg` **argument_type**
- typedef `_Result` **result_type**

Public Member Functions

- `size_t operator() (_Tp *__p)` const noexcept

4.572.1 Detailed Description

```
template<typename _Tp>
struct std::hash< _Tp * >
```

Partial specializations for pointer types.

Definition at line 62 of file `functional_hash.h`.

The documentation for this struct was generated from the following file:

- [functional_hash.h](#)

4.573 `std::hash< bool >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

Public Types

- typedef `_Arg` **argument_type**
- typedef `_Result` **result_type**

Public Member Functions

- `size_t operator() (bool __val)` const noexcept

4.573.1 Detailed Description

```
template<>
struct std::hash< bool >
```

Explicit specialization for `bool`.

Definition at line 80 of file `functional_hash.h`.

The documentation for this struct was generated from the following file:

- [functional_hash.h](#)

4.574 `std::hash< char >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

Public Types

- typedef `_Arg` **argument_type**
- typedef `_Result` **result_type**

Public Member Functions

- `size_t` **operator()** (`char __val`) `const noexcept`

4.574.1 Detailed Description

```
template<>
struct std::hash< char >
```

Explicit specialization for `char`.

Definition at line 83 of file `functional_hash.h`.

The documentation for this struct was generated from the following file:

- [functional_hash.h](#)

4.575 `std::hash< char16_t >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

Public Types

- typedef `_Arg` **argument_type**
- typedef `_Result` **result_type**

Public Member Functions

- `size_t` **operator()** (`char16_t __val`) `const noexcept`

4.575.1 Detailed Description

```
template<>
struct std::hash< char16_t >
```

Explicit specialization for char16_t.

Definition at line 95 of file functional_hash.h.

The documentation for this struct was generated from the following file:

- [functional_hash.h](#)

4.576 std::hash< char32_t > Struct Template Reference

Inherits std::__hash_base< _Result, _Arg >.

Public Types

- typedef _Arg **argument_type**
- typedef _Result **result_type**

Public Member Functions

- size_t **operator()** (char32_t __val) const noexcept

4.576.1 Detailed Description

```
template<>
struct std::hash< char32_t >
```

Explicit specialization for char32_t.

Definition at line 98 of file functional_hash.h.

The documentation for this struct was generated from the following file:

- [functional_hash.h](#)

4.577 std::hash< double > Struct Template Reference

Inherits std::__hash_base< _Result, _Arg >.

Public Types

- typedef `_Arg` **argument_type**
- typedef `_Result` **result_type**

Public Member Functions

- `size_t operator()` (`double __val`) `const noexcept`

4.577.1 Detailed Description

```
template<>
struct std::hash< double >
```

Specialization for double.

Definition at line 176 of file `functional_hash.h`.

The documentation for this struct was generated from the following file:

- [functional_hash.h](#)

4.578 `std::hash< float >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

Public Types

- typedef `_Arg` **argument_type**
- typedef `_Result` **result_type**

Public Member Functions

- `size_t operator()` (`float __val`) `const noexcept`

4.578.1 Detailed Description

```
template<>
struct std::hash< float >
```

Specialization for float.

Definition at line 164 of file `functional_hash.h`.

The documentation for this struct was generated from the following file:

- [functional_hash.h](#)

4.579 `std::hash< int >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

Public Types

- typedef `_Arg` **argument_type**
- typedef `_Result` **result_type**

Public Member Functions

- `size_t operator()` (`int __val`) `const noexcept`

4.579.1 Detailed Description

```
template<>
struct std::hash< int >
```

Explicit specialization for `int`.

Definition at line 104 of file `functional_hash.h`.

The documentation for this struct was generated from the following file:

- [functional_hash.h](#)

4.580 `std::hash< long >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

Public Types

- typedef `_Arg` **argument_type**
- typedef `_Result` **result_type**

Public Member Functions

- `size_t operator()` (`long __val`) `const noexcept`

4.580.1 Detailed Description

```
template<>
struct std::hash< long >
```

Explicit specialization for long.

Definition at line 107 of file `functional_hash.h`.

The documentation for this struct was generated from the following file:

- [functional_hash.h](#)

4.581 `std::hash< long double >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

Public Types

- typedef `_Arg` **argument_type**
- typedef `_Result` **result_type**

Public Member Functions

- `size_t` **operator()** (`long double __val`) const noexcept

4.581.1 Detailed Description

```
template<>
struct std::hash< long double >
```

Specialization for long double.

Definition at line 188 of file `functional_hash.h`.

The documentation for this struct was generated from the following file:

- [functional_hash.h](#)

4.582 `std::hash< long long >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

Public Types

- typedef `_Arg` **argument_type**
- typedef `_Result` **result_type**

Public Member Functions

- `size_t operator()` (long long __val) const noexcept

4.582.1 Detailed Description

```
template<>
struct std::hash< long long >
```

Explicit specialization for long long.

Definition at line 110 of file `functional_hash.h`.

The documentation for this struct was generated from the following file:

- [functional_hash.h](#)

4.583 `std::hash< shared_ptr< _Tp > >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

Public Types

- typedef `_Arg` **argument_type**
- typedef `_Result` **result_type**

Public Member Functions

- `size_t operator()` (const [shared_ptr](#)< `_Tp` > &__s) const noexcept

4.583.1 Detailed Description

```
template<typename _Tp>
struct std::hash< shared_ptr< _Tp > >
```

`std::hash` specialization for `shared_ptr`.

Definition at line 615 of file `shared_ptr.h`.

The documentation for this struct was generated from the following file:

- [shared_ptr.h](#)

4.584 `std::hash< short >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

Public Types

- typedef `_Arg` **argument_type**
- typedef `_Result` **result_type**

Public Member Functions

- `size_t` **operator()** (`short __val`) `const noexcept`

4.584.1 Detailed Description

```
template<>
struct std::hash< short >
```

Explicit specialization for `short`.

Definition at line 101 of file `functional_hash.h`.

The documentation for this struct was generated from the following file:

- [functional_hash.h](#)

4.585 `std::hash< signed char >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

Public Types

- typedef `_Arg` **argument_type**
- typedef `_Result` **result_type**

Public Member Functions

- `size_t` **operator()** (`signed char __val`) `const noexcept`

4.585.1 Detailed Description

```
template<>
struct std::hash< signed char >
```

Explicit specialization for signed char.

Definition at line 86 of file functional_hash.h.

The documentation for this struct was generated from the following file:

- [functional_hash.h](#)

4.586 std::hash< string > Struct Template Reference

Inherits std::__hash_base< _Result, _Arg >.

Public Types

- typedef _Arg **argument_type**
- typedef _Result **result_type**

Public Member Functions

- size_t **operator()** (const [string](#) &__s) const noexcept

4.586.1 Detailed Description

```
template<>
struct std::hash< string >
```

std::hash specialization for string.

Definition at line 3079 of file basic_string.h.

The documentation for this struct was generated from the following file:

- [basic_string.h](#)

4.587 std::hash< u16string > Struct Template Reference

Inherits std::__hash_base< _Result, _Arg >.

Public Types

- typedef _Arg **argument_type**
- typedef _Result **result_type**

Public Member Functions

- size_t **operator()** (const [u16string](#) &__s) const noexcept

4.587.1 Detailed Description

```
template<>
struct std::hash< u16string >
```

std::hash specialization for u16string.

Definition at line 3112 of file basic_string.h.

The documentation for this struct was generated from the following file:

- [basic_string.h](#)

4.588 std::hash< u32string > Struct Template Reference

Inherits std::__hash_base< _Result, _Arg >.

Public Types

- typedef _Arg **argument_type**
- typedef _Result **result_type**

Public Member Functions

- size_t **operator()** (const [u32string](#) &__s) const noexcept

4.588.1 Detailed Description

```
template<>
struct std::hash< u32string >
```

std::hash specialization for u32string.

Definition at line 3127 of file basic_string.h.

The documentation for this struct was generated from the following file:

- [basic_string.h](#)

4.589 `std::hash< unique_ptr< _Tp, _Dp > >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

Public Types

- typedef `_Arg` **argument_type**
- typedef `_Result` **result_type**

Public Member Functions

- `size_t operator()` (const `unique_ptr< _Tp, _Dp > &__u`) const noexcept

4.589.1 Detailed Description

```
template<typename _Tp, typename _Dp>
struct std::hash< unique_ptr< _Tp, _Dp > >
```

`std::hash` specialization for `unique_ptr`.

Definition at line 734 of file `unique_ptr.h`.

The documentation for this struct was generated from the following file:

- [unique_ptr.h](#)

4.590 `std::hash< unsigned char >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

Public Types

- typedef `_Arg` **argument_type**
- typedef `_Result` **result_type**

Public Member Functions

- `size_t operator()` (unsigned char `__val`) const noexcept

4.590.1 Detailed Description

```
template<>
struct std::hash< unsigned char >
```

Explicit specialization for unsigned char.

Definition at line 89 of file `functional_hash.h`.

The documentation for this struct was generated from the following file:

- [functional_hash.h](#)

4.591 `std::hash< unsigned int >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

Public Types

- typedef `_Arg` **argument_type**
- typedef `_Result` **result_type**

Public Member Functions

- `size_t` **operator()** (`unsigned int __val`) `const noexcept`

4.591.1 Detailed Description

```
template<>
struct std::hash< unsigned int >
```

Explicit specialization for unsigned int.

Definition at line 116 of file `functional_hash.h`.

The documentation for this struct was generated from the following file:

- [functional_hash.h](#)

4.592 `std::hash< unsigned long >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

Public Types

- typedef `_Arg` **argument_type**
- typedef `_Result` **result_type**

Public Member Functions

- `size_t operator()` (unsigned long __val) const noexcept

4.592.1 Detailed Description

```
template<>
struct std::hash< unsigned long >
```

Explicit specialization for unsigned long.

Definition at line 119 of file `functional_hash.h`.

The documentation for this struct was generated from the following file:

- [functional_hash.h](#)

4.593 `std::hash< unsigned long long >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

Public Types

- typedef `_Arg` **argument_type**
- typedef `_Result` **result_type**

Public Member Functions

- `size_t operator()` (unsigned long long __val) const noexcept

4.593.1 Detailed Description

```
template<>
struct std::hash< unsigned long long >
```

Explicit specialization for unsigned long long.

Definition at line 122 of file `functional_hash.h`.

The documentation for this struct was generated from the following file:

- [functional_hash.h](#)

4.594 `std::hash< unsigned short >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

Public Types

- typedef `_Arg` **argument_type**
- typedef `_Result` **result_type**

Public Member Functions

- `size_t` **operator()** (`unsigned short __val`) `const noexcept`

4.594.1 Detailed Description

```
template<>
struct std::hash< unsigned short >
```

Explicit specialization for unsigned short.

Definition at line 113 of file `functional_hash.h`.

The documentation for this struct was generated from the following file:

- [functional_hash.h](#)

4.595 `std::hash< wchar_t >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

Public Types

- typedef `_Arg` **argument_type**
- typedef `_Result` **result_type**

Public Member Functions

- `size_t` **operator()** (`wchar_t __val`) `const noexcept`

4.595.1 Detailed Description

```
template<>
struct std::hash< wchar_t >
```

Explicit specialization for wchar_t.

Definition at line 92 of file functional_hash.h.

The documentation for this struct was generated from the following file:

- [functional_hash.h](#)

4.596 std::hash< wstring > Struct Template Reference

Inherits std::__hash_base< _Result, _Arg >.

Public Types

- typedef _Arg **argument_type**
- typedef _Result **result_type**

Public Member Functions

- size_t **operator()** (const [wstring](#) &__s) const noexcept

4.596.1 Detailed Description

```
template<>
struct std::hash< wstring >
```

std::hash specialization for wstring.

Definition at line 3094 of file basic_string.h.

The documentation for this struct was generated from the following file:

- [basic_string.h](#)

4.597 std::hash<::vector< bool, _Alloc > > Struct Template Reference

Inherits std::__hash_base< _Result, _Arg >.

Public Types

- typedef `_Arg` **argument_type**
- typedef `_Result` **result_type**

Public Member Functions

- `size_t operator()` (const `::vector< bool, _Alloc >` &) const noexcept

4.597.1 Detailed Description

```
template<typename _Alloc>
struct std::hash<::vector< bool, _Alloc > >
```

`std::hash` specialization for `vector<bool>`.

Definition at line 1190 of file `stl_bvector.h`.

The documentation for this struct was generated from the following file:

- [stl_bvector.h](#)

4.598 `std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType>` Class Template Reference

Public Types

- typedef `_UIntType` **result_type**

Public Member Functions

- [independent_bits_engine](#) ()
- [independent_bits_engine](#) (const `_RandomNumberEngine` &__rng)
- [independent_bits_engine](#) (`_RandomNumberEngine` &&__rng)
- [independent_bits_engine](#) (**result_type** __s)
- template<typename `_Sseq` , typename = typename `std::enable_if<!std::is_same<_Sseq, independent_bits_engine>::value && !std::is_↵`
same<_Sseq, `_RandomNumberEngine`>::value> ::type>
- [independent_bits_engine](#) (`_Sseq` &__q)
- const `_RandomNumberEngine` & [base](#) () const noexcept
- void [discard](#) (unsigned long long __z)
- **result_type** [operator\(\)](#) ()
- void [seed](#) ()
- void [seed](#) (**result_type** __s)
- template<typename `_Sseq` >
- void [seed](#) (`_Sseq` &__q)

Static Public Member Functions

- static constexpr [result_type](#) max ()
- static constexpr [result_type](#) min ()

Friends

- bool [operator==](#) (const [independent_bits_engine](#) &__lhs, const [independent_bits_engine](#) &__rhs)
- template<typename [_CharT](#), typename [_Traits](#) > [std::basic_istream](#)< [_CharT](#), [_Traits](#) > & [operator>>](#) ([std::basic_istream](#)< [_CharT](#), [_Traits](#) > &__is, [std::independent_bits_engine](#)< [_RandomNumberEngine](#), __w, [UIntType](#) > &__x)

4.598.1 Detailed Description

```
template<typename \_RandomNumberEngine, size_t __w, typename UIntType>
class std::independent\_bits\_engine< \_RandomNumberEngine, __w, UIntType >
```

Produces random numbers by combining random numbers from some base engine to produce random numbers with a specifies number of bits `__w`.

Definition at line 1076 of file random.h.

4.598.2 Member Typedef Documentation

4.598.2.1 `template<typename _RandomNumberEngine , size_t __w, typename UIntType > typedef UIntType std::independent_bits_engine< _RandomNumberEngine, __w, UIntType >::result_type`

The type of the generated random value.

Definition at line 1079 of file random.h.

4.598.3 Constructor & Destructor Documentation

4.598.3.1 `template<typename _RandomNumberEngine , size_t __w, typename UIntType > std::independent_bits_engine< _RandomNumberEngine, __w, UIntType >::independent_bits_engine () [inline]`

Constructs a default `independent_bits_engine` engine.

The underlying engine is default constructed as well.

Definition at line 1092 of file random.h.

4.598.3.2 `template<typename _RandomNumberEngine , size_t __w, typename UIntType > std::independent_bits_engine< _RandomNumberEngine, __w, UIntType >::independent_bits_engine (const _RandomNumberEngine & __rng) [inline], [explicit]`

Copy constructs a `independent_bits_engine` engine.

Copies an existing base class random number generator.

Parameters

<code>__rng</code>	An existing (base class) engine object.
--------------------	---

Definition at line 1102 of file random.h.

```
4.598.3.3 template<typename _RandomNumberEngine, size_t __w, typename _UIntType> std::independent_bits_engine<
    _RandomNumberEngine, __w, _UIntType>::independent_bits_engine ( _RandomNumberEngine && __rng )
    [inline], [explicit]
```

Move constructs a `independent_bits_engine` engine.

Copies an existing base class random number generator.

Parameters

<code>__rng</code>	An existing (base class) engine object.
--------------------	---

Definition at line 1112 of file random.h.

```
4.598.3.4 template<typename _RandomNumberEngine, size_t __w, typename _UIntType> std::independent_bits_engine<
    _RandomNumberEngine, __w, _UIntType>::independent_bits_engine ( result_type __s ) [inline],
    [explicit]
```

Seed constructs a `independent_bits_engine` engine.

Constructs the underlying generator engine seeded with `__s`.

Parameters

<code>__s</code>	A seed value for the base class engine.
------------------	---

Definition at line 1122 of file random.h.

```
4.598.3.5 template<typename _RandomNumberEngine, size_t __w, typename _UIntType> template<typename _Sseq, typename
    = typename std::enable_if<!std::is_same<_Sseq, independent_bits_engine>::value && !std::is_same<_Sseq,
    _RandomNumberEngine>::value> ::type> std::independent_bits_engine< _RandomNumberEngine, __w,
    _UIntType>::independent_bits_engine ( _Sseq & __q ) [inline], [explicit]
```

Generator construct a `independent_bits_engine` engine.

Parameters

<code>__q</code>	A seed sequence.
------------------	------------------

Definition at line 1135 of file random.h.

4.598.4 Member Function Documentation

4.598.4.1 `template<typename _RandomNumberEngine, size_t __w, typename _UIntType> const _RandomNumberEngine& std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType>::base () const [inline], [noexcept]`

Gets a const reference to the underlying generator engine object.

Definition at line 1170 of file random.h.

Referenced by `std::operator<<()`.

4.598.4.2 `template<typename _RandomNumberEngine, size_t __w, typename _UIntType> void std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType>::discard (unsigned long long __z) [inline]`

Discard a sequence of random numbers.

Definition at line 1191 of file random.h.

4.598.4.3 `template<typename _RandomNumberEngine, size_t __w, typename _UIntType> static constexpr result_type std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType>::max () [inline], [static]`

Gets the maximum value in the generated random number range.

Definition at line 1184 of file random.h.

4.598.4.4 `template<typename _RandomNumberEngine, size_t __w, typename _UIntType> static constexpr result_type std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType>::min () [inline], [static]`

Gets the minimum value in the generated random number range.

Definition at line 1177 of file random.h.

4.598.4.5 `template<typename _RandomNumberEngine, size_t __w, typename _UIntType> result_type std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType>::operator() ()`

Gets the next value in the generated random number sequence.

4.598.4.6 `template<typename _RandomNumberEngine, size_t __w, typename _UIntType> void std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType>::seed () [inline]`

Reseeds the `independent_bits_engine` object with the default seed for the underlying base class generator engine.

Definition at line 1144 of file random.h.

4.598.4.7 `template<typename _RandomNumberEngine, size_t __w, typename _UIntType> void std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType>::seed (result_type __s) [inline]`

Reseeds the `independent_bits_engine` object with the default seed for the underlying base class generator engine.

Definition at line 1152 of file random.h.

4.598.4.8 `template<typename _RandomNumberEngine, size_t __w, typename _UIntType> template<typename _Sseq> void std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType>::seed (_Sseq & __q) [inline]`

Reseeds the `independent_bits_engine` object with the given seed sequence.

Parameters

<code>__q</code>	A seed generator function.
------------------	----------------------------

Definition at line 1162 of file random.h.

4.598.5 Friends And Related Function Documentation

4.598.5.1 `template<typename _RandomNumberEngine, size_t __w, typename _UIntType> bool operator==(const independent_bits_engine<_RandomNumberEngine, __w, _UIntType> & __lhs, const independent_bits_engine<_RandomNumberEngine, __w, _UIntType> & __rhs) [friend]`

Compares two `independent_bits_engine` random number generator objects of the same type for equality.

Parameters

<code>__lhs</code>	A <code>independent_bits_engine</code> random number generator object.
<code>__rhs</code>	Another <code>independent_bits_engine</code> random number generator object.

Returns

true if the infinite sequences of generated values would be equal, false otherwise.

Definition at line 1216 of file random.h.

4.598.5.2 `template<typename _RandomNumberEngine, size_t __w, typename _UIntType> template<typename _CharT, typename _Traits> std::basic_istream<_CharT, _Traits>& operator>>(std::basic_istream<_CharT, _Traits> & __is, std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType> & __x) [friend]`

Extracts the current state of a `% subtract_with_carry_engine` random number generator engine `__x` from the input stream `__is`.

Parameters

<code>__is</code>	An input stream.
<code>__x</code>	A <code>independent_bits_engine</code> random number generator engine.

Returns

The input stream with the state of `__x` extracted or in an error state.

Definition at line 1234 of file random.h.

The documentation for this class was generated from the following file:

- [random.h](#)

4.599 `std::indirect_array<_Tp>` Class Template Reference

Public Types

- `typedef _Tp value_type`

Public Member Functions

- `indirect_array` (const `indirect_array` &)
- `void operator%=(const valarray<_Tp> &) const`
- `template<class _Dom>`
`void operator%=(const _Expr<_Dom, _Tp> &) const`
- `void operator&=(const valarray<_Tp> &) const`
- `template<class _Dom>`
`void operator&=(const _Expr<_Dom, _Tp> &) const`
- `void operator*=(const valarray<_Tp> &) const`
- `template<class _Dom>`
`void operator*=(const _Expr<_Dom, _Tp> &) const`
- `void operator+=(const valarray<_Tp> &) const`
- `template<class _Dom>`
`void operator+=(const _Expr<_Dom, _Tp> &) const`
- `void operator-=(const valarray<_Tp> &) const`
- `template<class _Dom>`
`void operator-=(const _Expr<_Dom, _Tp> &) const`
- `void operator/=(const valarray<_Tp> &) const`
- `template<class _Dom>`
`void operator/=(const _Expr<_Dom, _Tp> &) const`
- `void operator<=<= (const valarray<_Tp> &) const`
- `template<class _Dom>`
`void operator<=<= (const _Expr<_Dom, _Tp> &) const`
- `indirect_array` & `operator=` (const `indirect_array` &)
- `void operator=` (const valarray<_Tp> &) const
- `void operator=` (const _Tp &) const
- `template<class _Dom>`
`void operator=` (const _Expr<_Dom, _Tp> &) const
- `void operator>=>= (const valarray<_Tp> &) const`
- `template<class _Dom>`
`void operator>=>= (const _Expr<_Dom, _Tp> &) const`
- `void operator^= (const valarray<_Tp> &) const`
- `template<class _Dom>`
`void operator^= (const _Expr<_Dom, _Tp> &) const`
- `void operator|= (const valarray<_Tp> &) const`
- `template<class _Dom>`
`void operator|= (const _Expr<_Dom, _Tp> &) const`

Friends

- class `gslice_array<_Tp>`
- class `valarray<_Tp>`

4.599.1 Detailed Description

```
template<class _Tp>
class std::indirect_array<_Tp>
```

Reference to arbitrary subset of an array.

An `indirect_array` is a reference to the actual elements of an array specified by an ordered array of indices. The way to get an `indirect_array` is to call `operator[]`(`valarray<size_t>`) on a `valarray`. The returned `indirect_array` then permits carrying operations out on the referenced subset of elements in the original `valarray`.

For example, if an `indirect_array` is obtained using the array (4,2,0) as an argument, and then assigned to an array containing (1,2,3), then the underlying array will have `array[0]==3`, `array[2]==2`, and `array[4]==1`.

Parameters

<i>Tp</i>	Element type.
-----------	---------------

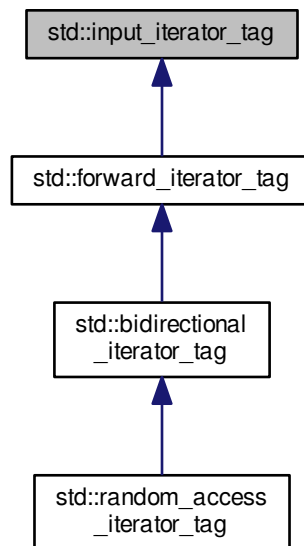
Definition at line 62 of file `indirect_array.h`.

The documentation for this class was generated from the following file:

- [indirect_array.h](#)

4.600 std::input_iterator_tag Struct Reference

Inheritance diagram for std::input_iterator_tag:



4.600.1 Detailed Description

Marking input iterators.

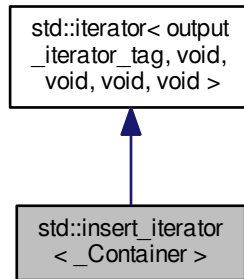
Definition at line 89 of file `stl_iterator_base_types.h`.

The documentation for this struct was generated from the following file:

- [stl_iterator_base_types.h](#)

4.601 `std::insert_iterator<_Container>` Class Template Reference

Inheritance diagram for `std::insert_iterator<_Container>`:



Public Types

- typedef `_Container` `container_type`
- typedef void `difference_type`
- typedef `output_iterator_tag` `iterator_category`
- typedef void `pointer`
- typedef void `reference`
- typedef void `value_type`

Public Member Functions

- `insert_iterator` (`_Container &__x`, `typename _Container::iterator __i`)
- `insert_iterator & operator*` ()
- `insert_iterator & operator++` ()
- `insert_iterator & operator++` (int)
- `insert_iterator & operator=` (const `typename _Container::value_type &__value`)
- `insert_iterator & operator=` (`typename _Container::value_type &&__value`)

Protected Attributes

- `_Container *` **`container`**
- `_Container::iterator` **`iter`**

4.601.1 Detailed Description

```
template<typename _Container>
class std::insert_iterator< _Container >
```

Turns assignment into insertion.

These are output iterators, constructed from a container-of-T. Assigning a T to the iterator inserts it in the container at the iterator's position, rather than overwriting the value at that position.

(Sequences will actually insert a *copy* of the value before the iterator's position.)

Tip: Using the inserter function to create these iterators can save typing.

Definition at line 588 of file stl_iterator.h.

4.601.2 Member Typedef Documentation

4.601.2.1 `template<typename _Container> typedef _Container std::insert_iterator< _Container >::container_type`

A nested typedef for the type of whatever container you used.

Definition at line 597 of file stl_iterator.h.

4.601.2.2 `typedef void std::iterator< output_iterator_tag , void , void , void , void >::difference_type` [inherited]

Distance between iterators is represented as this type.

Definition at line 125 of file stl_iterator_base_types.h.

4.601.2.3 `typedef output_iterator_tag std::iterator< output_iterator_tag , void , void , void , void >::iterator_category` [inherited]

One of the [tag types](#).

Definition at line 121 of file stl_iterator_base_types.h.

4.601.2.4 `typedef void std::iterator< output_iterator_tag , void , void , void , void >::pointer` [inherited]

This type represents a pointer-to-value_type.

Definition at line 127 of file stl_iterator_base_types.h.

4.601.2.5 `typedef void std::iterator< output_iterator_tag , void , void , void , void >::reference` [inherited]

This type represents a reference-to-value_type.

Definition at line 129 of file stl_iterator_base_types.h.

4.601.2.6 `typedef void std::iterator< output_iterator_tag, void, void, void, void >::value_type` `[inherited]`

The type "pointed to" by the iterator.

Definition at line 123 of file `stl_iterator_base_types.h`.

4.601.3 Constructor & Destructor Documentation

4.601.3.1 `template<typename _Container> std::insert_iterator< _Container >::insert_iterator (_Container & __x, typename _Container::iterator __i)` `[inline]`

The only way to create this iterator is with a container and an initial position (a normal iterator into the container).

Definition at line 603 of file `stl_iterator.h`.

4.601.4 Member Function Documentation

4.601.4.1 `template<typename _Container> insert_iterator& std::insert_iterator< _Container >::operator* ()` `[inline]`

Simply returns `*this`.

Definition at line 657 of file `stl_iterator.h`.

4.601.4.2 `template<typename _Container> insert_iterator& std::insert_iterator< _Container >::operator++ ()` `[inline]`

Simply returns `*this`. (This iterator does not *move*.)

Definition at line 662 of file `stl_iterator.h`.

4.601.4.3 `template<typename _Container> insert_iterator& std::insert_iterator< _Container >::operator++ (int)` `[inline]`

Simply returns `*this`. (This iterator does not *move*.)

Definition at line 667 of file `stl_iterator.h`.

4.601.4.4 `template<typename _Container> insert_iterator& std::insert_iterator< _Container >::operator= (const typename _Container::value_type & __value)` `[inline]`

Parameters

<code>__value</code>	An instance of whatever type <code>container_type::const_reference</code> is; presumably a reference-to-const T for <code>container<T></code> .
----------------------	---

Returns

This iterator, for chained operations.

This kind of iterator maintains its own position in the container. Assigning a value to the iterator will insert the value into the container at the place before the iterator.

The position is maintained such that subsequent assignments will insert values immediately after one another. For example,

```
// vector v contains A and Z
insert_iterator i (v, ++v.begin());
i = 1;
i = 2;
i = 3;

// vector v contains A, 1, 2, 3, and Z
```

Definition at line 639 of file `stl_iterator.h`.

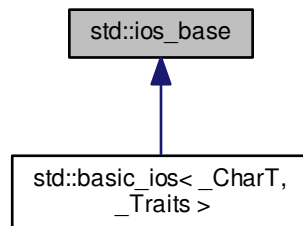
References `std::move()`.

The documentation for this class was generated from the following file:

- [stl_iterator.h](#)

4.602 std::ios_base Class Reference

Inheritance diagram for `std::ios_base`:



Classes

- class [failure](#)

Public Types

- enum [event](#) { [erase_event](#), [imbue_event](#), [copyfmt_event](#) }
- typedef void(* [event_callback](#)) ([event](#) __e, [ios_base](#) &__b, int __i)
- typedef _ios_Fmtflags [fmtflags](#)
- typedef int [io_state](#)
- typedef _ios_istate [iostate](#)
- typedef int [open_mode](#)
- typedef _ios_Openmode [openmode](#)
- typedef int [seek_dir](#)
- typedef _ios_Seekdir [seekdir](#)
- typedef [std::streamoff](#) [streamoff](#)
- typedef [std::streampos](#) [streampos](#)

Public Member Functions

- virtual [~ios_base](#) ()
- const [locale](#) & [_M_getloc](#) () const
- [fmtflags](#) [flags](#) () const
- [fmtflags](#) [flags](#) ([fmtflags](#) __fmtfl)
- [locale](#) [getloc](#) () const
- [locale](#) [imbue](#) (const [locale](#) &__loc) throw ()
- long & [iword](#) (int __ix)
- [streamsize](#) [precision](#) () const
- [streamsize](#) [precision](#) ([streamsize](#) __prec)
- void *& [pword](#) (int __ix)
- void [register_callback](#) ([event_callback](#) __fn, int __index)
- [fmtflags](#) [setf](#) ([fmtflags](#) __fmtfl)
- [fmtflags](#) [setf](#) ([fmtflags](#) __fmtfl, [fmtflags](#) __mask)
- void [unsetf](#) ([fmtflags](#) __mask)
- [streamsize](#) [width](#) () const
- [streamsize](#) [width](#) ([streamsize](#) __wide)

Static Public Member Functions

- static bool [sync_with_stdio](#) (bool __sync=true)
- static int [xalloc](#) () throw ()

Static Public Attributes

- static const [fmtflags](#) [adjustfield](#)
- static const [openmode](#) [app](#)
- static const [openmode](#) [ate](#)
- static const [iostate](#) [badbit](#)
- static const [fmtflags](#) [basefield](#)
- static const [seekdir](#) [beg](#)
- static const [openmode](#) [binary](#)
- static const [fmtflags](#) [boolalpha](#)

- static const [seekdir](#) `cur`
- static const [fmtflags](#) `dec`
- static const [seekdir](#) `end`
- static const [iostate](#) `eofbit`
- static const [iostate](#) `failbit`
- static const [fmtflags](#) `fixed`
- static const [fmtflags](#) `floatfield`
- static const [iostate](#) `goodbit`
- static const [fmtflags](#) `hex`
- static const [openmode](#) `in`
- static const [fmtflags](#) `internal`
- static const [fmtflags](#) `left`
- static const [fmtflags](#) `oct`
- static const [openmode](#) `out`
- static const [fmtflags](#) `right`
- static const [fmtflags](#) `scientific`
- static const [fmtflags](#) `showbase`
- static const [fmtflags](#) `showpoint`
- static const [fmtflags](#) `showpos`
- static const [fmtflags](#) `skipws`
- static const [openmode](#) `trunc`
- static const [fmtflags](#) `unitbuf`
- static const [fmtflags](#) `uppercase`

Protected Types

- enum { `_S_local_word_size` }

Protected Member Functions

- void `_M_call_callbacks` ([event](#) __ev) throw ()
- void `_M_dispose_callbacks` (void) throw ()
- `_Words` & `_M_grow_words` (int __index, bool __iword)
- void `_M_init` () throw ()

Protected Attributes

- `_Callback_list` * `_M_callbacks`
- [iostate](#) `_M_exception`
- [fmtflags](#) `_M_flags`
- [locale](#) `_M_ios_locale`
- `_Words` `_M_local_word` [`_S_local_word_size`]
- [streamsize](#) `_M_precision`
- [iostate](#) `_M_streambuf_state`
- [streamsize](#) `_M_width`
- `_Words` * `_M_word`
- int `_M_word_size`
- `_Words` `_M_word_zero`

4.602.1 Detailed Description

The base of the I/O class hierarchy.

This class defines everything that can be defined about I/O that does not depend on the type of characters being input or output. Most people will only see `ios_base` when they need to specify the full name of the various I/O flags (e.g., the openmodes).

Definition at line 199 of file `ios_base.h`.

4.602.2 Member Typedef Documentation

4.602.2.1 `typedef void(* std::ios_base::event_callback)(event __e, ios_base &__b, int __i)`

The type of an event callback function.

Parameters

<code>__e</code>	One of the members of the event enum.
<code>__b</code>	Reference to the <code>ios_base</code> object.
<code>__i</code>	The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several `ios_base` and `basic_ios` functions, specifically `imbue()`, `copyfmt()`, and `~ios()`.

Definition at line 436 of file `ios_base.h`.

4.602.2.2 `typedef _ios_Fmtflags std::ios_base::fmtflags`

This is a bitmask type.

`_Ios_Fmtflags` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `fmtflags` are:

- `boolalpha`
- `dec`
- `fixed`
- `hex`
- `internal`
- `left`
- `oct`

- right
- scientific
- showbase
- showpoint
- showpos
- skipws
- unitbuf
- uppercase
- adjustfield
- basefield
- floatfield

Definition at line 255 of file ios_base.h.

4.602.2.3 typedef _Ios_Iostate std::ios_base::iostate

This is a bitmask type.

_Ios_Iostate is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type iostate are:

- badbit
- eofbit
- failbit
- goodbit

Definition at line 330 of file ios_base.h.

4.602.2.4 typedef _Ios_Openmode std::ios_base::openmode

This is a bitmask type.

_Ios_Openmode is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type openmode are:

- app
- ate
- binary
- in
- out
- trunc

Definition at line 361 of file ios_base.h.

4.602.2.5 typedef _Ios_Seekdir std::ios_base::seekdir

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- `beg`
- `cur`, equivalent to `SEEK_CUR` in the C standard library.
- `end`, equivalent to `SEEK_END` in the C standard library.

Definition at line 393 of file `ios_base.h`.

4.602.3 Member Enumeration Documentation

4.602.3.1 enum std::ios_base::event

The set of events that may be passed to an event callback.

`erase_event` is used during `~ios()` and `copyfmt()`. `imbue_event` is used during `imbue()`. `copyfmt_event` is used during `copyfmt()`.

Definition at line 419 of file `ios_base.h`.

4.602.4 Constructor & Destructor Documentation

4.602.4.1 virtual std::ios_base::~ios_base () [virtual]

Invokes each callback with `erase_event`. Destroys local storage.

Note that the `ios_base` object for the standard streams never gets destroyed. As a result, any callbacks registered with the standard streams will not get invoked with `erase_event` (unless `copyfmt` is used).

Referenced by `pword()`.

4.602.5 Member Function Documentation

4.602.5.1 const locale& std::ios_base::_M_getloc () const [inline]

Locale access.

Returns

A reference to the current locale.

Like `getloc` above, but returns a reference instead of generating a copy.

Definition at line 706 of file `ios_base.h`.

References `xalloc()`.

4.602.5.2 `fmtflags std::ios_base::flags () const` `[inline]`

Access to format flags.

Returns

The format control flags for both input and output.

Definition at line 551 of file `ios_base.h`.

4.602.5.3 `fmtflags std::ios_base::flags (fmtflags __fmtfl)` `[inline]`

Setting new format flags all at once.

Parameters

<code>__fmtfl</code>	The new flags to set.
----------------------	-----------------------

Returns

The previous format control flags.

This function overwrites all the format flags with `__fmtfl`.

Definition at line 562 of file `ios_base.h`.

4.602.5.4 `locale std::ios_base::getloc () const` `[inline]`

Locale access.

Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Definition at line 695 of file `ios_base.h`.

4.602.5.5 `locale std::ios_base::imbue (const locale & __loc) throw ()`

Setting a new locale.

Parameters

<code>__loc</code>	The new locale.
--------------------	-----------------

Returns

The previous locale.

Sets the new locale for this stream, and then invokes each callback with imbue_event.

Referenced by width().

4.602.5.6 long& std::ios_base::iword (int __ix) [inline]

Access to integer array.

Parameters

<code>__ix</code>	Index into the array.
-------------------	-----------------------

Returns

A reference to an integer associated with the index.

The iword function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 741 of file ios_base.h.

4.602.5.7 streamsize std::ios_base::precision () const [inline]

Flags access.

Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 621 of file ios_base.h.

4.602.5.8 streamsize std::ios_base::precision (streamsize __prec) [inline]

Changing flags.

Parameters

<code>__prec</code>	The new precision value.
---------------------	--------------------------

Returns

The previous value of `precision()`.

Definition at line 630 of file `ios_base.h`.

4.602.5.9 `void*& std::ios_base::pword (int __ix) [inline]`

Access to void pointer array.

Parameters

<code>__ix</code>	Index into the array.
-------------------	-----------------------

Returns

A reference to a `void*` associated with the index.

The `pword` function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 762 of file `ios_base.h`.

References `~ios_base()`.

4.602.5.10 `void std::ios_base::register_callback (event_callback __fn, int __index)`

Add the callback `__fn` with parameter `__index`.

Parameters

<code>__fn</code>	The function to add.
<code>__index</code>	The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

4.602.5.11 `fmtflags std::ios_base::setf (fmtflags __fmtfl) [inline]`

Setting new format flags.

Parameters

<code>__fmtfl</code>	Additional flags to set.
----------------------	--------------------------

Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 578 of file ios_base.h.

Referenced by `std::boolalpha()`, `std::dec()`, `std::fixed()`, `std::hex()`, `std::left()`, `std::oct()`, `std::right()`, `std::scientific()`, `std::showbase()`, `std::showpoint()`, `std::showpos()`, `std::skipws()`, `std::unitbuf()`, and `std::uppercase()`.

4.602.5.12 `fmtflags std::ios_base::setf (fmtflags __fmtfl, fmtflags __mask)` `[inline]`

Setting new format flags.

Parameters

<code>__fmtfl</code>	Additional flags to set.
<code>__mask</code>	The flags mask for <code>fmtfl</code> .

Returns

The previous format control flags.

This function clears `mask` in the format flags, then sets `fmtfl & mask`. An example mask is `ios_base::adjustfield`.

Definition at line 595 of file ios_base.h.

4.602.5.13 `static bool std::ios_base::sync_with_stdio (bool __sync = true)` `[static]`

Interaction with the standard C I/O objects.

Parameters

<code>__sync</code>	Whether to synchronize or not.
---------------------	--------------------------------

Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., `stdout`) and the standard C++ objects (e.g., `cout`). User-declared streams are unaffected. See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt1.html>

Referenced by `width()`.

4.602.5.14 `void std::ios_base::unsetf (fmtflags __mask)` `[inline]`

Clearing format flags.

Parameters

<code>__mask</code>	The flags to unset.
---------------------	---------------------

This function clears `__mask` in the format flags.

Definition at line 610 of file `ios_base.h`.

Referenced by `std::noboolalpha()`, `std::noshowbase()`, `std::noshowpoint()`, `std::noshowpos()`, `std::noskipws()`, `std::nounitbuf()`, and `std::nouppercase()`.

4.602.5.15 `streamsize std::ios_base::width () const` `[inline]`

Flags access.

Returns

The minimum field width to generate on output operations.

Minimum field width refers to the number of characters.

Definition at line 644 of file `ios_base.h`.

4.602.5.16 `streamsize std::ios_base::width (streamsize __wide)` `[inline]`

Changing flags.

Parameters

<code>__wide</code>	The new width value.
---------------------	----------------------

Returns

The previous value of `width()`.

Definition at line 653 of file `ios_base.h`.

References `imbue()`, and `sync_with_stdio()`.

4.602.5.17 `static int std::ios_base::xalloc () throw` `[static]`

Access to unique indices.

Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

Referenced by `_M_getloc()`.

4.602.6 Member Data Documentation

4.602.6.1 const fmtflags std::ios_base::adjustfield [static]

A mask of `left|right|internal`. Useful for the 2-arg form of `setf`.

Definition at line 310 of file `ios_base.h`.

Referenced by `std::internal()`, `std::left()`, and `std::right()`.

4.602.6.2 const openmode std::ios_base::app [static]

Seek to end before each write.

Definition at line 364 of file `ios_base.h`.

4.602.6.3 const openmode std::ios_base::ate [static]

Open and seek to end immediately after opening.

Definition at line 367 of file `ios_base.h`.

4.602.6.4 const iostate std::ios_base::badbit [static]

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Definition at line 334 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::bad()`, and `std::basic_ios<_CharT, _Traits>::fail()`.

4.602.6.5 const fmtflags std::ios_base::basefield [static]

A mask of `dec|oct|hex`. Useful for the 2-arg form of `setf`.

Definition at line 313 of file `ios_base.h`.

Referenced by `std::dec()`, `std::hex()`, and `std::oct()`.

4.602.6.6 `const seekdir std::ios_base::beg` [static]

Request a seek relative to the beginning of the stream.

Definition at line 396 of file `ios_base.h`.

Referenced by `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>::file()`.

4.602.6.7 `const openmode std::ios_base::binary` [static]

Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch27s02.html>.

Definition at line 372 of file `ios_base.h`.

4.602.6.8 `const fmtflags std::ios_base::boolalpha` [static]

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 258 of file `ios_base.h`.

Referenced by `std::boolalpha()`, and `std::noboolalpha()`.

4.602.6.9 `const seekdir std::ios_base::cur` [static]

Request a seek relative to the current position within the sequence.

Definition at line 399 of file `ios_base.h`.

Referenced by `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>::file()`.

4.602.6.10 `const fmtflags std::ios_base::dec` [static]

Converts integer input or generates integer output in decimal base.

Definition at line 261 of file `ios_base.h`.

Referenced by `std::dec()`.

4.602.6.11 `const seekdir std::ios_base::end` [static]

Request a seek relative to the current end of the sequence.

Definition at line 402 of file `ios_base.h`.

4.602.6.12 `const iostate std::ios_base::eofbit` [static]

Indicates that an input operation reached the end of an input sequence.

Definition at line 337 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::eof()`.

4.602.6.13 `const iostate std::ios_base::failbit` `[static]`

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.

Definition at line 342 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::fail()`.

4.602.6.14 `const fmtflags std::ios_base::fixed` `[static]`

Generate floating-point output in fixed-point notation.

Definition at line 264 of file `ios_base.h`.

Referenced by `std::fixed()`.

4.602.6.15 `const fmtflags std::ios_base::floatfield` `[static]`

A mask of `scientific|fixed`. Useful for the 2-arg form of `setf`.

Definition at line 316 of file `ios_base.h`.

Referenced by `std::fixed()`, and `std::scientific()`.

4.602.6.16 `const iostate std::ios_base::goodbit` `[static]`

Indicates all is well.

Definition at line 345 of file `ios_base.h`.

Referenced by `std::basic_ios<_CharT, _Traits>::rdstate()`.

4.602.6.17 `const fmtflags std::ios_base::hex` `[static]`

Converts integer input or generates integer output in hexadecimal base.

Definition at line 267 of file `ios_base.h`.

Referenced by `std::hex()`.

4.602.6.18 `const openmode std::ios_base::in` `[static]`

Open for input. Default for `ifstream` and `fstream`.

Definition at line 375 of file `ios_base.h`.

Referenced by `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>::file()`.

4.602.6.19 `const fmtflags std::ios_base::internal` `[static]`

Adds fill characters at a designated internal point in certain generated output, or identical to `right` if no such point is designated.

Definition at line 272 of file `ios_base.h`.

Referenced by `std::internal()`.

4.602.6.20 `const fmtflags std::ios_base::left` `[static]`

Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.)

Definition at line 276 of file `ios_base.h`.

Referenced by `std::left()`.

4.602.6.21 `const fmtflags std::ios_base::oct` `[static]`

Converts integer input or generates integer output in octal base.

Definition at line 279 of file `ios_base.h`.

Referenced by `std::oct()`.

4.602.6.22 `const openmode std::ios_base::out` `[static]`

Open for output. Default for `ofstream` and `fstream`.

Definition at line 378 of file `ios_base.h`.

Referenced by `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>::file()`.

4.602.6.23 `const fmtflags std::ios_base::right` `[static]`

Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.)

Definition at line 283 of file `ios_base.h`.

Referenced by `std::right()`.

4.602.6.24 `const fmtflags std::ios_base::scientific` `[static]`

Generates floating-point output in scientific notation.

Definition at line 286 of file `ios_base.h`.

Referenced by `std::scientific()`.

4.602.6.25 const fmtflags std::ios_base::showbase [static]

Generates a prefix indicating the numeric base of generated integer output.

Definition at line 290 of file ios_base.h.

Referenced by std::noshowbase(), and std::showbase().

4.602.6.26 const fmtflags std::ios_base::showpoint [static]

Generates a decimal-point character unconditionally in generated floating-point output.

Definition at line 294 of file ios_base.h.

Referenced by std::noshowpoint(), and std::showpoint().

4.602.6.27 const fmtflags std::ios_base::showpos [static]

Generates a + sign in non-negative generated numeric output.

Definition at line 297 of file ios_base.h.

Referenced by std::noshowpos(), and std::showpos().

4.602.6.28 const fmtflags std::ios_base::skipws [static]

Skips leading white space before certain input operations.

Definition at line 300 of file ios_base.h.

Referenced by std::noskipws(), and std::skipws().

4.602.6.29 const openmode std::ios_base::trunc [static]

Open for input. Default for ofstream.

Definition at line 381 of file ios_base.h.

4.602.6.30 const fmtflags std::ios_base::unitbuf [static]

Flushes output after each output operation.

Definition at line 303 of file ios_base.h.

Referenced by std::nounitbuf(), and std::unitbuf().

4.602.6.31 `const fmtflags std::ios_base::uppercase` [static]

Replaces certain lowercase letters with their uppercase equivalents in generated output.

Definition at line 307 of file `ios_base.h`.

Referenced by `std::nouppercase()`, and `std::uppercase()`.

The documentation for this class was generated from the following file:

- [ios_base.h](#)

4.603 `std::ios_base::failure` Class Reference

Inherits exception.

Public Member Functions

- **failure** (const [string](#) &__str) throw ()
- virtual const char * **what** () const throw ()

4.603.1 Detailed Description

These are thrown to indicate problems with io.

27.4.2.1.1 Class `ios_base::failure`.

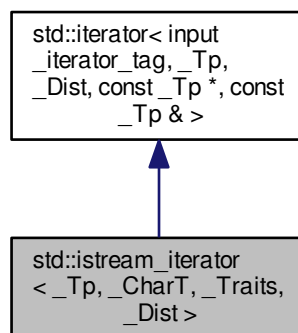
Definition at line 209 of file `ios_base.h`.

The documentation for this class was generated from the following file:

- [ios_base.h](#)

4.604 `std::istream_iterator< _Tp, _CharT, _Traits, _Dist >` Class Template Reference

Inheritance diagram for `std::istream_iterator< _Tp, _CharT, _Traits, _Dist >`:



Public Types

- typedef `_CharT` **char_type**
- typedef `_Dist` [difference_type](#)
- typedef `basic_istream<_CharT, _Traits>` **istream_type**
- typedef [input_iterator_tag](#) **iterator_category**
- typedef `const _Tp *` [pointer](#)
- typedef `const _Tp &` [reference](#)
- typedef `_Traits` **traits_type**
- typedef `_Tp` [value_type](#)

Public Member Functions

- constexpr [istream_iterator](#) ()
- [istream_iterator](#) (istream_type &__s)
- **istream_iterator** (const [istream_iterator](#) &__obj)
- `bool` **_M_equal** (const [istream_iterator](#) &__x) const
- `const _Tp &` **operator*** () const
- [istream_iterator](#) & **operator++** ()
- [istream_iterator](#) **operator++** (int)
- `const _Tp *` **operator->** () const

4.604.1 Detailed Description

```
template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>, typename _Dist = ptrdiff_t>
class std::istream_iterator<_Tp, _CharT, _Traits, _Dist>
```

Provides input iterator semantics for streams.

Definition at line 49 of file `stream_iterator.h`.

4.604.2 Member Typedef Documentation

4.604.2.1 `typedef _Dist std::iterator< input_iterator_tag , _Tp, _Dist , const _Tp *, const _Tp >::difference_type`
[[inherited](#)]

Distance between iterators is represented as this type.

Definition at line 125 of file `stl_iterator_base_types.h`.

4.604.2.2 `typedef input_iterator_tag std::iterator< input_iterator_tag , _Tp, _Dist , const _Tp *, const _Tp & >::iterator_category` [[inherited](#)]

One of the [tag types](#).

Definition at line 121 of file `stl_iterator_base_types.h`.

4.604.2.3 `typedef const _Tp * std::iterator< input_iterator_tag , _Tp, _Dist , const _Tp * , const _Tp & >::pointer`
`[inherited]`

This type represents a pointer-to-value_type.

Definition at line 127 of file `stl_iterator_base_types.h`.

4.604.2.4 `typedef const _Tp & std::iterator< input_iterator_tag , _Tp, _Dist , const _Tp * , const _Tp & >::reference`
`[inherited]`

This type represents a reference-to-value_type.

Definition at line 129 of file `stl_iterator_base_types.h`.

4.604.2.5 `typedef _Tp std::iterator< input_iterator_tag , _Tp, _Dist , const _Tp * , const _Tp & >::value_type`
`[inherited]`

The type "pointed to" by the iterator.

Definition at line 123 of file `stl_iterator_base_types.h`.

4.604.3 Constructor & Destructor Documentation

4.604.3.1 `template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>, typename _Dist = ptrdiff_t>`
`constexpr std::istream_iterator< _Tp, _CharT, _Traits, _Dist >::istream_iterator () [inline]`

Construct end of input stream iterator.

Definition at line 64 of file `stream_iterator.h`.

Referenced by `std::istream_iterator< _Tp, _CharT, _Traits, _Dist >::istream_iterator()`.

4.604.3.2 `template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>, typename _Dist = ptrdiff_t>`
`std::istream_iterator< _Tp, _CharT, _Traits, _Dist >::istream_iterator (istream_type & __s) [inline]`

Construct start of input stream iterator.

Definition at line 68 of file `stream_iterator.h`.

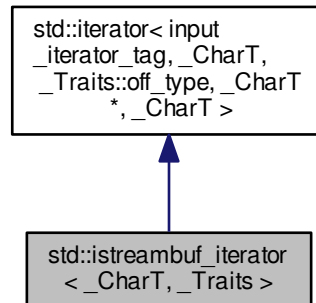
References `std::istream_iterator< _Tp, _CharT, _Traits, _Dist >::istream_iterator()`.

The documentation for this class was generated from the following file:

- [stream_iterator.h](#)

4.605 std::istreambuf_iterator< _CharT, _Traits > Class Template Reference

Inheritance diagram for std::istreambuf_iterator< _CharT, _Traits >:



Public Types

- typedef `_Traits::off_type` [difference_type](#)
 - typedef `input_iterator_tag` [iterator_category](#)
 - typedef `_CharT *` [pointer](#)
 - typedef `_CharT` [reference](#)
 - typedef `_CharT` [value_type](#)
-
- typedef `_CharT` [char_type](#)
 - typedef `_Traits` [traits_type](#)
 - typedef `_Traits::int_type` [int_type](#)
 - typedef `basic_streambuf< _CharT, _Traits >` [streambuf_type](#)
 - typedef `basic_istream< _CharT, _Traits >` [istream_type](#)

Public Member Functions

- constexpr [istreambuf_iterator](#) () noexcept
- **[istreambuf_iterator](#)** (const [istreambuf_iterator](#) &) noexcept=default
- [istreambuf_iterator](#) ([istream_type](#) &__s) noexcept
- [istreambuf_iterator](#) ([streambuf_type](#) *__s) noexcept
- bool [equal](#) (const [istreambuf_iterator](#) &__b) const
- [char_type](#) [operator*](#) () const
- [istreambuf_iterator](#) & [operator++](#) ()
- [istreambuf_iterator](#) [operator++](#) (int)

Friends

- `template<bool _IsMove, typename _CharT2 >`
`__gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, _CharT2 * >::__type __copy_move_a2`
`(istreambuf_iterator< _CharT2 >, istreambuf_iterator< _CharT2 >, _CharT2 *)`
- `template<typename _CharT2 >`
`__gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, ostreambuf_iterator< _CharT2 >::__type copy`
`(istreambuf_iterator< _CharT2 >, istreambuf_iterator< _CharT2 >, ostreambuf_iterator< _CharT2 >)`
- `template<typename _CharT2 >`
`__gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, istreambuf_iterator< _CharT2 >::__type find`
`(istreambuf_iterator< _CharT2 >, istreambuf_iterator< _CharT2 >, const _CharT2 &)`

4.605.1 Detailed Description

```
template<typename _CharT, typename _Traits>
class std::istreambuf_iterator< _CharT, _Traits >
```

Provides input iterator semantics for streambufs.

Definition at line 407 of file `stl_algobase.h`.

4.605.2 Member Typedef Documentation

4.605.2.1 `template<typename _CharT, typename _Traits > typedef _CharT std::istreambuf_iterator< _CharT, _Traits >::char_type`

Public typedefs.

Definition at line 64 of file `streambuf_iterator.h`.

4.605.2.2 `typedef _Traits::off_type std::iterator< input_iterator_tag, _CharT, _Traits::off_type, _CharT *, _CharT >::difference_type` `[inherited]`

Distance between iterators is represented as this type.

Definition at line 125 of file `stl_iterator_base_types.h`.

4.605.2.3 `template<typename _CharT, typename _Traits > typedef _Traits::int_type std::istreambuf_iterator< _CharT, _Traits >::int_type`

Public typedefs.

Definition at line 66 of file `streambuf_iterator.h`.

4.605.2.4 `template<typename _CharT, typename _Traits > typedef basic_istream< _CharT, _Traits> std::istreambuf_iterator< _CharT, _Traits >::istream_type`

Public typedefs.

Definition at line 68 of file `streambuf_iterator.h`.

4.605.2.5 `typedef input_iterator_tag std::iterator< input_iterator_tag , _CharT , _Traits::off_type , _CharT * , _CharT >::iterator_category` [inherited]

One of the [tag types](#).

Definition at line 121 of file `stl_iterator_base_types.h`.

4.605.2.6 `typedef _CharT * std::iterator< input_iterator_tag , _CharT , _Traits::off_type , _CharT * , _CharT >::pointer` [inherited]

This type represents a pointer-to-value_type.

Definition at line 127 of file `stl_iterator_base_types.h`.

4.605.2.7 `typedef _CharT std::iterator< input_iterator_tag , _CharT , _Traits::off_type , _CharT * , _CharT >::reference` [inherited]

This type represents a reference-to-value_type.

Definition at line 129 of file `stl_iterator_base_types.h`.

4.605.2.8 `template<typename _CharT , typename _Traits > typedef basic_streambuf<_CharT, _Traits> std::istreambuf_iterator<_CharT, _Traits>::streambuf_type`

Public typedefs.

Definition at line 67 of file `streambuf_iterator.h`.

4.605.2.9 `template<typename _CharT , typename _Traits > typedef _Traits std::istreambuf_iterator< _CharT, _Traits >::traits_type`

Public typedefs.

Definition at line 65 of file `streambuf_iterator.h`.

4.605.2.10 `typedef _CharT std::iterator< input_iterator_tag , _CharT , _Traits::off_type , _CharT * , _CharT >::value_type` [inherited]

The type "pointed to" by the iterator.

Definition at line 123 of file `stl_iterator_base_types.h`.

4.605.3 Constructor & Destructor Documentation

4.605.3.1 `template<typename _CharT , typename _Traits > constexpr std::istreambuf_iterator< _CharT, _Traits >::istreambuf_iterator()` [inline], [noexcept]

Construct end of input stream iterator.

Definition at line 102 of file `streambuf_iterator.h`.

4.605.3.2 `template<typename _CharT , typename _Traits > std::istreambuf_iterator< _CharT, _Traits >::istreambuf_iterator (istream_type & __s) [inline], [noexcept]`

Construct start of input stream iterator.

Definition at line 112 of file `streambuf_iterator.h`.

4.605.3.3 `template<typename _CharT , typename _Traits > std::istreambuf_iterator< _CharT, _Traits >::istreambuf_iterator (streambuf_type * __s) [inline], [noexcept]`

Construct start of streambuf iterator.

Definition at line 116 of file `streambuf_iterator.h`.

4.605.4 Member Function Documentation

4.605.4.1 `template<typename _CharT , typename _Traits > bool std::istreambuf_iterator< _CharT, _Traits >::equal (const istreambuf_iterator< _CharT, _Traits > & __b) const [inline]`

Return true both iterators are end or both are not end.

Definition at line 172 of file `streambuf_iterator.h`.

4.605.4.2 `template<typename _CharT , typename _Traits > char_type std::istreambuf_iterator< _CharT, _Traits >::operator*() const [inline]`

Return the current character pointed to by iterator. This returns `streambuf.sgetc()`. It cannot be assigned. NB: The result of `operator*()` on an end of stream is undefined.

Definition at line 123 of file `streambuf_iterator.h`.

4.605.4.3 `template<typename _CharT , typename _Traits > istreambuf_iterator& std::istreambuf_iterator< _CharT, _Traits >::operator++ () [inline]`

Advance the iterator. Calls `streambuf.sbumpc()`.

Definition at line 137 of file `streambuf_iterator.h`.

4.605.4.4 `template<typename _CharT , typename _Traits > istreambuf_iterator std::istreambuf_iterator< _CharT, _Traits >::operator++ (int) [inline]`

Advance the iterator. Calls `streambuf.sbumpc()`.

Definition at line 152 of file `streambuf_iterator.h`.

The documentation for this class was generated from the following files:

- [stl_algobase.h](#)
- [streambuf_iterator.h](#)

4.606 `std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >` Struct Template Reference

Public Types

- typedef `_Distance` [difference_type](#)
- typedef `_Category` [iterator_category](#)
- typedef `_Pointer` [pointer](#)
- typedef `_Reference` [reference](#)
- typedef `_Tp` [value_type](#)

4.606.1 Detailed Description

```
template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _Tp*, typename _Reference =  
_Tp&>  
struct std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >
```

Common iterator class.

This class does nothing but define nested typedefs. Iterator classes can inherit from this class to save some work. The typedefs are then used in specializations and overloading.

In particular, there are no default implementations of requirements such as `operator++` and the like. (How could there be?)

Definition at line 118 of file `stl_iterator_base_types.h`.

4.606.2 Member Typedef Documentation

4.606.2.1 `template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _Tp*,
typename _Reference = _Tp&> typedef _Distance std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference
>::difference_type`

Distance between iterators is represented as this type.

Definition at line 125 of file `stl_iterator_base_types.h`.

4.606.2.2 `template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _Tp*,
typename _Reference = _Tp&> typedef _Category std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference
>::iterator_category`

One of the [tag types](#).

Definition at line 121 of file `stl_iterator_base_types.h`.

4.606.2.3 `template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _Tp*, typename
_Reference = _Tp&> typedef _Pointer std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >::pointer`

This type represents a pointer-to-`value_type`.

Definition at line 127 of file `stl_iterator_base_types.h`.

4.606.2.4 `template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _Tp*, typename _Reference = _Tp&> typedef _Reference std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >::reference`

This type represents a reference-to-value_type.

Definition at line 129 of file `stl_iterator_base_types.h`.

4.606.2.5 `template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _Tp*, typename _Reference = _Tp&> typedef _Tp std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >::value_type`

The type "pointed to" by the iterator.

Definition at line 123 of file `stl_iterator_base_types.h`.

The documentation for this struct was generated from the following file:

- [stl_iterator_base_types.h](#)

4.607 `std::iterator_traits< _Tp * >` Struct Template Reference

Public Types

- `typedef ptrdiff_t difference_type`
- `typedef random_access_iterator_tag iterator_category`
- `typedef _Tp * pointer`
- `typedef _Tp & reference`
- `typedef _Tp value_type`

4.607.1 Detailed Description

```
template<typename _Tp>
struct std::iterator_traits< _Tp * >
```

Partial specialization for pointer types.

Definition at line 175 of file `stl_iterator_base_types.h`.

The documentation for this struct was generated from the following file:

- [stl_iterator_base_types.h](#)

4.608 `std::iterator_traits< const _Tp * >` Struct Template Reference

Public Types

- `typedef ptrdiff_t difference_type`
- `typedef random_access_iterator_tag iterator_category`
- `typedef const _Tp * pointer`
- `typedef const _Tp & reference`
- `typedef _Tp value_type`

4.608.1 Detailed Description

```
template<typename _Tp>
struct std::iterator_traits< const _Tp * >
```

Partial specialization for const pointer types.

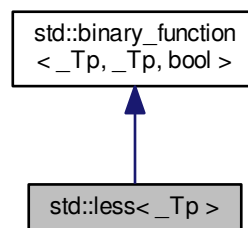
Definition at line 186 of file stl_iterator_base_types.h.

The documentation for this struct was generated from the following file:

- [stl_iterator_base_types.h](#)

4.609 std::less< _Tp > Struct Template Reference

Inheritance diagram for std::less< _Tp >:



Public Types

- typedef _Tp [first_argument_type](#)
- typedef bool [result_type](#)
- typedef _Tp [second_argument_type](#)

Public Member Functions

- bool **operator()** (const _Tp &__x, const _Tp &__y) const

4.609.1 Detailed Description

```
template<typename _Tp>
struct std::less< _Tp >
```

One of the [comparison functors](#).

Definition at line 367 of file stl_function.h.

4.609.2 Member Typedef Documentation

4.609.2.1 `typedef _Tp std::binary_function<_Tp, _Tp, bool>::first_argument_type` [inherited]

`first_argument_type` is the type of the first argument

Definition at line 121 of file `stl_function.h`.

4.609.2.2 `typedef bool std::binary_function<_Tp, _Tp, bool>::result_type` [inherited]

`result_type` is the return type

Definition at line 127 of file `stl_function.h`.

4.609.2.3 `typedef _Tp std::binary_function<_Tp, _Tp, bool>::second_argument_type` [inherited]

`second_argument_type` is the type of the second argument

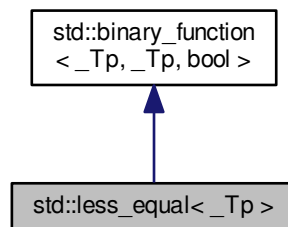
Definition at line 124 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl_function.h](#)

4.610 `std::less_equal<_Tp>` Struct Template Reference

Inheritance diagram for `std::less_equal<_Tp>`:



Public Types

- `typedef _Tp` [first_argument_type](#)
- `typedef bool` [result_type](#)
- `typedef _Tp` [second_argument_type](#)

Public Member Functions

- `bool operator()` (`const _Tp &__x`, `const _Tp &__y`) `const`

4.610.1 Detailed Description

```
template<typename _Tp>
struct std::less_equal<_Tp>
```

One of the [comparison functors](#).

Definition at line 385 of file `stl_function.h`.

4.610.2 Member Typedef Documentation

4.610.2.1 `typedef _Tp std::binary_function<_Tp, _Tp, bool>::first_argument_type` `[inherited]`

`first_argument_type` is the type of the first argument

Definition at line 121 of file `stl_function.h`.

4.610.2.2 `typedef bool std::binary_function<_Tp, _Tp, bool>::result_type` `[inherited]`

`result_type` is the return type

Definition at line 127 of file `stl_function.h`.

4.610.2.3 `typedef _Tp std::binary_function<_Tp, _Tp, bool>::second_argument_type` `[inherited]`

`second_argument_type` is the type of the second argument

Definition at line 124 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl_function.h](#)

4.611 `std::linear_congruential_engine<_UIntType, __a, __c, __m>` Class Template Reference

Public Types

- `typedef _UIntType` [result_type](#)

Public Member Functions

- [linear_congruential_engine](#) ([result_type](#) __s=default_seed)
- template<typename _Sseq, typename = typename std::enable_if<!std::is_same<_Sseq, linear_congruential_engine>::value>::type> [linear_congruential_engine](#) (_Sseq &__q)
- void [discard](#) (unsigned long long __z)
- [result_type](#) operator() ()
- void [seed](#) ([result_type](#) __s=default_seed)
- template<typename _Sseq > std::enable_if< std::is_class< _Sseq >::value >::type [seed](#) (_Sseq &__q)

Static Public Member Functions

- static constexpr [result_type](#) [max](#) ()
- static constexpr [result_type](#) [min](#) ()

Static Public Attributes

- static constexpr [result_type](#) [default_seed](#)
- static constexpr [result_type](#) [increment](#)
- static constexpr [result_type](#) [modulus](#)
- static constexpr [result_type](#) [multiplier](#)

Friends

- template<typename _UIntType1, _UIntType1 __a1, _UIntType1 __c1, _UIntType1 __m1, typename _CharT, typename _Traits > std::basic_ostream< _CharT, _Traits > & [operator<<](#) (std::basic_ostream< _CharT, _Traits > &__os, const [std::linear_congruential_engine](#)< _UIntType1, __a1, __c1, __m1 > &__lcr)
- bool [operator==](#) (const [linear_congruential_engine](#) &__lhs, const [linear_congruential_engine](#) &__rhs)
- template<typename _UIntType1, _UIntType1 __a1, _UIntType1 __c1, _UIntType1 __m1, typename _CharT, typename _Traits > std::basic_istream< _CharT, _Traits > & [operator>>](#) (std::basic_istream< _CharT, _Traits > &__is, [std::linear_congruential_engine](#)< _UIntType1, __a1, __c1, __m1 > &__lcr)

4.611.1 Detailed Description

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
class std::linear_congruential_engine< _UIntType, __a, __c, __m >
```

A model of a linear congruential random number generator.

A random number generator that produces pseudorandom numbers via linear function:

$$x_{i+1} \leftarrow (ax_i + c) \bmod m$$

The template parameter `_UIntType` must be an unsigned integral type large enough to store values up to `(__m-1)`. If the template parameter `__m` is 0, the modulus `__m` used is `std::numeric_limits<_UIntType>::max()` plus 1. Otherwise, the template parameters `__a` and `__c` must be less than `__m`.

The size of the state is 1.

Definition at line 243 of file random.h.

4.611.2 Member Typedef Documentation

4.611.2.1 `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> typedef _UIntType
std::linear_congruential_engine<_UIntType, __a, __c, __m>::result_type`

The type of the generated random value.

Definition at line 246 of file random.h.

4.611.3 Constructor & Destructor Documentation

4.611.3.1 `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> std::linear_congruential_engine<
_UIntType, __a, __c, __m>::linear_congruential_engine (result_type __s = default_seed)
[inline], [explicit]`

Constructs a linear_congruential_engine random number generator engine with seed __s. The default seed value is 1.

Parameters

<code>__s</code>	The initial seed value.
------------------	-------------------------

Definition at line 270 of file random.h.

4.611.3.2 `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> template<typename _Sseq
, typename = typename std::enable_if<!std::is_same<_Sseq, linear_congruential_engine>::value>::type>
std::linear_congruential_engine<_UIntType, __a, __c, __m>::linear_congruential_engine (_Sseq & __q)
[inline], [explicit]`

Constructs a linear_congruential_engine random number generator engine seeded from the seed sequence __q.

Parameters

<code>__q</code>	the seed sequence.
------------------	--------------------

Definition at line 283 of file random.h.

4.611.4 Member Function Documentation

4.611.4.1 `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> void std::linear_↵
_congruential_engine<_UIntType, __a, __c, __m>::discard (unsigned long long __z)
[inline]`

Discard a sequence of random numbers.

Definition at line 327 of file random.h.

4.611.4.2 `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> static constexpr result_type
std::linear_congruential_engine<_UIntType, __a, __c, __m>::max () [inline], [static]`

Gets the largest possible value in the output range.

Definition at line 320 of file random.h.

4.611.4.3 `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> static constexpr result_type
std::linear_congruential_engine<_UIntType, __a, __c, __m>::min () [inline], [static]`

Gets the smallest possible value in the output range.

The minimum depends on the `__c` parameter: if it is zero, the minimum generated must be > 0 , otherwise 0 is allowed.

Definition at line 313 of file random.h.

4.611.4.4 `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> result_type
std::linear_congruential_engine<_UIntType, __a, __c, __m>::operator() () [inline]`

Gets the next random number in the sequence.

Definition at line 337 of file random.h.

4.611.4.5 `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> void std::linear_↵
congruential_engine<_UIntType, __a, __c, __m>::seed (result_type __s = default_seed
)`

Reseeds the `linear_congruential_engine` random number generator engine sequence to the seed `__s`.

Parameters

<code>↵ __s</code>	The new seed.
------------------------	---------------

4.611.4.6 `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> template<typename _Sseq >
std::enable_if<std::is_class<_Sseq>::value>::type std::linear_congruential_engine<_UIntType, __a, __c, __m
>::seed (_Sseq & __q)`

Reseeds the `linear_congruential_engine` random number generator engine sequence using values from the seed sequence `__q`.

Parameters

<code>↵ __q</code>	the seed sequence.
------------------------	--------------------

4.611.5 Friends And Related Function Documentation

4.611.5.1 `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> template<typename _UIntType1, _UIntType1 __a1, _UIntType1 __c1, _UIntType1 __m1, typename _CharT, typename _Traits> std::basic_ostream<_CharT, _Traits>& operator<< (std::basic_ostream<_CharT, _Traits> & __os, const std::linear_congruential_engine<_UIntType1, __a1, __c1, __m1> & __lcr) [friend]`

Writes the textual representation of the state x(i) of x to __os.

Parameters

<code>__os</code>	The output stream.
<code>__lcr</code>	A % linear_congruential_engine random number generator.

Returns

`__os`.

4.611.5.2 `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> bool operator== (const linear_congruential_engine<_UIntType, __a, __c, __m> & __lhs, const linear_congruential_engine<_UIntType, __a, __c, __m> & __rhs) [friend]`

Compares two linear congruential random number generator objects of the same type for equality.

Parameters

<code>__lhs</code>	A linear congruential random number generator object.
<code>__rhs</code>	Another linear congruential random number generator object.

Returns

true if the infinite sequences of generated values would be equal, false otherwise.

Definition at line 355 of file random.h.

4.611.5.3 `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> template<typename _UIntType1, _UIntType1 __a1, _UIntType1 __c1, _UIntType1 __m1, typename _CharT, typename _Traits> std::basic_istream<_CharT, _Traits>& operator>> (std::basic_istream<_CharT, _Traits> & __is, std::linear_congruential_engine<_UIntType1, __a1, __c1, __m1> & __lcr) [friend]`

Sets the state of the engine by reading its textual representation from __is.

The textual representation must have been previously written using an output stream whose imbued locale and whose type's template specialization arguments `_CharT` and `_Traits` were the same as those of `__is`.

Parameters

<code>__is</code>	The input stream.
<code>__lcr</code>	A % linear_congruential_engine random number generator.

Returns

`__is`.

4.611.6 Member Data Documentation

4.611.6.1 `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> constexpr result_type std::linear_congruential_engine<_UIntType, __a, __c, __m>::increment [static]`

An increment.

Definition at line 257 of file random.h.

4.611.6.2 `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> constexpr result_type std::linear_congruential_engine<_UIntType, __a, __c, __m>::modulus [static]`

The modulus.

Definition at line 259 of file random.h.

4.611.6.3 `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> constexpr result_type std::linear_congruential_engine<_UIntType, __a, __c, __m>::multiplier [static]`

The multiplier.

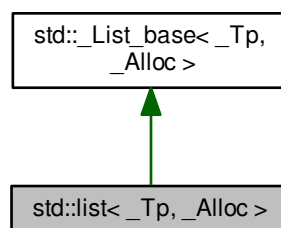
Definition at line 255 of file random.h.

The documentation for this class was generated from the following file:

- [random.h](#)

4.612 `std::list<_Tp, _Alloc>` Class Template Reference

Inheritance diagram for `std::list<_Tp, _Alloc>`:



Public Types

- typedef _Alloc **allocator_type**
- typedef [_List_const_iterator](#)< _Tp > **const_iterator**
- typedef _Tp_alloc_type::const_pointer **const_pointer**
- typedef _Tp_alloc_type::const_reference **const_reference**
- typedef [std::reverse_iterator](#)< [const_iterator](#) > **const_reverse_iterator**
- typedef ptrdiff_t **difference_type**
- typedef [_List_iterator](#)< _Tp > **iterator**
- typedef _Tp_alloc_type::pointer **pointer**
- typedef _Tp_alloc_type::reference **reference**
- typedef [std::reverse_iterator](#)< [iterator](#) > **reverse_iterator**
- typedef size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- [list](#) () noexcept(is_nothrow_default_constructible< _Node_alloc_type >::value)
- [list](#) (const allocator_type &__a) noexcept
- [list](#) (size_type __n)
- [list](#) (size_type __n, const value_type &__value, const allocator_type &__a=allocator_type())
- [list](#) (const [list](#) &__x)
- [list](#) ([list](#) &&__x) noexcept
- [list](#) (initializer_list< value_type > __l, const allocator_type &__a=allocator_type())
- template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>>
 [list](#) (_InputIterator __first, _InputIterator __last, const allocator_type &__a=allocator_type())
- void [assign](#) (size_type __n, const value_type &__val)
- template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>>
 void [assign](#) (_InputIterator __first, _InputIterator __last)
- void [assign](#) (initializer_list< value_type > __l)
- reference [back](#) () noexcept
- const_reference [back](#) () const noexcept
- [iterator begin](#) () noexcept
- [const_iterator begin](#) () const noexcept
- [const_iterator cbegin](#) () const noexcept
- [const_iterator cend](#) () const noexcept
- void [clear](#) () noexcept
- [const_reverse_iterator crbegin](#) () const noexcept
- [const_reverse_iterator crend](#) () const noexcept
- template<typename... _Args>
 [iterator emplace](#) ([const_iterator](#) __position, _Args &&...__args)
- template<typename... _Args>
 void [emplace_back](#) (_Args &&...__args)
- template<typename... _Args>
 void [emplace_front](#) (_Args &&...__args)
- bool [empty](#) () const noexcept
- [iterator end](#) () noexcept
- [const_iterator end](#) () const noexcept
- [iterator erase](#) ([const_iterator](#) __position) noexcept
- [iterator erase](#) ([const_iterator](#) __first, [const_iterator](#) __last) noexcept

- reference [front](#) () noexcept
- const_reference [front](#) () const noexcept
- allocator_type [get_allocator](#) () const noexcept
- [iterator insert](#) (const_iterator __position, const value_type &__x)
- [iterator insert](#) (const_iterator __position, value_type &&__x)
- [iterator insert](#) (const_iterator __p, initializer_list< value_type > __l)
- [iterator insert](#) (const_iterator __position, size_type __n, const value_type &__x)
- template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>>
[iterator insert](#) (const_iterator __position, _InputIterator __first, _InputIterator __last)
- size_type [max_size](#) () const noexcept
- void [merge](#) (list &&__x)
- void [merge](#) (list &__x)
- template<typename _StrictWeakOrdering >
void [merge](#) (list &&__x, _StrictWeakOrdering __comp)
- template<typename _StrictWeakOrdering >
void [merge](#) (list &__x, _StrictWeakOrdering __comp)
- list & [operator=](#) (const list &__x)
- list & [operator=](#) (list &&__x)
- list & [operator=](#) (initializer_list< value_type > __l)
- void [pop_back](#) () noexcept
- void [pop_front](#) () noexcept
- void [push_back](#) (const value_type &__x)
- void [push_back](#) (value_type &&__x)
- void [push_front](#) (const value_type &__x)
- void [push_front](#) (value_type &&__x)
- [reverse_iterator rbegin](#) () noexcept
- [const_reverse_iterator rbegin](#) () const noexcept
- void [remove](#) (const _Tp &__value)
- template<typename _Predicate >
void [remove_if](#) (_Predicate)
- [reverse_iterator rend](#) () noexcept
- [const_reverse_iterator rend](#) () const noexcept
- void [resize](#) (size_type __new_size)
- void [resize](#) (size_type __new_size, const value_type &__x)
- void [reverse](#) () noexcept
- size_type [size](#) () const noexcept
- void [sort](#) ()
- template<typename _StrictWeakOrdering >
void [sort](#) (_StrictWeakOrdering)
- void [splice](#) (const_iterator __position, list &&__x) noexcept
- void [splice](#) (const_iterator __position, list &__x) noexcept
- void [splice](#) (const_iterator __position, list &&__x, const_iterator __i) noexcept
- void [splice](#) (const_iterator __position, list &__x, const_iterator __i) noexcept
- void [splice](#) (const_iterator __position, list &&__x, const_iterator __first, const_iterator __last) noexcept
- void [splice](#) (const_iterator __position, list &__x, const_iterator __first, const_iterator __last) noexcept
- void [swap](#) (list &__x)
- void [unique](#) ()
- template<typename _BinaryPredicate >
void [unique](#) (_BinaryPredicate)

Protected Types

- typedef [_List_node](#)<_Tp> **_Node**

Protected Member Functions

- template<typename _Integer >
void **_M_assign_dispatch** (_Integer __n, _Integer __val, __true_type)
- template<typename _InputIterator >
void **_M_assign_dispatch** (_InputIterator __first, _InputIterator __last, __false_type)
- void **_M_check_equal_allocators** ([list](#) &__x) noexcept
- void **_M_clear** () noexcept
- template<typename... _Args>
[_Node](#) * **_M_create_node** (_Args &&... __args)
- void **_M_default_append** (size_type __n)
- void **_M_default_initialize** (size_type __n)
- void **_M_erase** ([iterator](#) __position) noexcept
- void **_M_fill_assign** (size_type __n, const value_type &__val)
- void **_M_fill_initialize** (size_type __n, const value_type &__x)
- [_List_node](#)<_Tp> * **_M_get_node** ()
- _Node_alloc_type & **_M_get_Node_allocator** () noexcept
- const _Node_alloc_type & **_M_get_Node_allocator** () const noexcept
- _Tp_alloc_type **_M_get_Tp_allocator** () const noexcept
- void **_M_init** () noexcept
- template<typename _Integer >
void **_M_initialize_dispatch** (_Integer __n, _Integer __x, __true_type)
- template<typename _InputIterator >
void **_M_initialize_dispatch** (_InputIterator __first, _InputIterator __last, __false_type)
- template<typename... _Args>
void **_M_insert** ([iterator](#) __position, _Args &&... __args)
- void **_M_put_node** ([_List_node](#)<_Tp> * __p) noexcept
- void **_M_transfer** ([iterator](#) __position, [iterator](#) __first, [iterator](#) __last)

Protected Attributes

- [_List_impl](#) **_M_impl**

4.612.1 Detailed Description

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
class std::list<_Tp, _Alloc>
```

A standard container with linear time access to elements, and fixed time insertion/deletion at any point in the sequence.

Template Parameters

_Tp	Type of element.
_Alloc	Allocator type, defaults to <code>allocator<_Tp></code> .

Meets the requirements of a [container](#), a [reversible container](#), and a [sequence](#), including the [optional sequence requirements](#) with the exception of `at` and `operator[]`.

This is a *doubly linked* list. Traversal up and down the list requires linear time, but adding and removing elements (or *nodes*) is done in constant time, regardless of where the change takes place. Unlike `std::vector` and `std::deque`, random-access iterators are not provided, so subscripting (`[]`) access is not allowed. For algorithms which only need sequential access, this lack makes no difference.

Also unlike the other standard containers, `std::list` provides specialized algorithms unique to linked lists, such as splicing, sorting, and in-place reversal.

A couple points on memory allocation for `list<Tp>`:

First, we never actually allocate a `Tp`, we allocate `List_node<Tp>`'s and trust [20.1.5]/4 to DTRT. This is to ensure that after elements from `list<X,Alloc1>` are spliced into `list<X,Alloc2>`, destroying the memory of the second list is a valid operation, i.e., `Alloc1` giveth and `Alloc2` taketh away.

Second, a list conceptually represented as

```
A <----> B <----> C <----> D
```

is actually circular; a link exists between A and D. The list class holds (as its only data member) a private `list::iterator` pointing to *D*, not to *A*! To get to the head of the list, we start at the tail and move forward by one. When this member iterator's `next/previous` pointers refer to itself, the list is empty.

Definition at line 447 of file `stl_list.h`.

4.612.2 Constructor & Destructor Documentation

4.612.2.1 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::list<_Tp, _Alloc>::list () [inline], [noexcept]`

Creates a list with no elements.

Definition at line 533 of file `stl_list.h`.

4.612.2.2 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::list<_Tp, _Alloc>::list (const allocator_type &__a) [inline], [explicit], [noexcept]`

Creates a list with no elements.

Parameters

<code>__a</code>	An allocator object.
------------------	----------------------

Definition at line 544 of file `stl_list.h`.

4.612.2.3 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::list<_Tp, _Alloc>::list (size_type __n)`
`[inline], [explicit]`

Creates a list with default constructed elements.

Parameters

<code>__n</code>	The number of elements to initially create.
------------------	---

This constructor fills the list with `__n` default constructed elements.

Definition at line 556 of file `stl_list.h`.

4.612.2.4 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::list<_Tp, _Alloc>::list (size_type __n,`
`const value_type & __value, const allocator_type & __a = allocator_type()) [inline]`

Creates a list with copies of an exemplar element.

Parameters

<code>__n</code>	The number of elements to initially create.
<code>__value</code>	An element to copy.
<code>__a</code>	An allocator object.

This constructor fills the list with `__n` copies of `__value`.

Definition at line 568 of file `stl_list.h`.

4.612.2.5 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::list<_Tp, _Alloc>::list (const list<_Tp,`
`_Alloc> & __x) [inline]`

List copy constructor.

Parameters

<code>__x</code>	A list of identical element and allocator types.
------------------	--

The newly-created list uses a copy of the allocation object used by `__x`.

Definition at line 595 of file `stl_list.h`.

4.612.2.6 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::list<_Tp, _Alloc>::list (list<_Tp, _Alloc`
`> && __x) [inline], [noexcept]`

List move constructor.

Parameters

<code>__x</code>	A list of identical element and allocator types.
------------------	--

The newly-created list contains the exact contents of `__x`. The contents of `__x` are a valid, but unspecified list.

Definition at line 607 of file `stl_list.h`.

```
4.612.2.7  template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::list<_Tp, _Alloc>::list ( initializer_list<
value_type > __l, const allocator_type & __a = allocator_type() ) [inline]
```

Builds a list from an `initializer_list`.

Parameters

<code>__l</code>	An <code>initializer_list</code> of <code>value_type</code> .
<code>__a</code>	An allocator object.

Create a list consisting of copies of the elements in the `initializer_list` `__l`. This is linear in `__l.size()`.

Definition at line 618 of file `stl_list.h`.

```
4.612.2.8  template<typename _Tp, typename _Alloc = std::allocator<_Tp>> template<typename _InputIterator, typename =
std::_RequireInputIter<_InputIterator>>> std::list<_Tp, _Alloc>::list ( _InputIterator __first, _InputIterator __last,
const allocator_type & __a = allocator_type() ) [inline]
```

Builds a list from a range.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__a</code>	An allocator object.

Create a list consisting of copies of the elements from `[__first, __last)`. This is linear in `N` (where `N` is `distance(__first, __last)`).

Definition at line 637 of file `stl_list.h`.

4.612.3 Member Function Documentation

```
4.612.3.1  template<typename _Tp, typename _Alloc = std::allocator<_Tp>> template<typename... _Args> _Node* std::list<
_Tp, _Alloc>::_M_create_node ( _Args &&... __args ) [inline], [protected]
```


Parameters

<code>__args</code>	An instance of user data.
---------------------	---------------------------

Allocates space for a new node and constructs a copy of `__args` in it.

Definition at line 509 of file `stl_list.h`.

4.612.3.2 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::list<_Tp, _Alloc>::assign (size_type __n, const value_type & __val) [inline]`

Assigns a given value to a list.

Parameters

<code>__n</code>	Number of elements to be assigned.
<code>__val</code>	Value to be assigned.

This function fills a list with `__n` copies of the given value. Note that the assignment completely changes the list and that the resulting list's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 715 of file `stl_list.h`.

4.612.3.3 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>> void std::list<_Tp, _Alloc>::assign (_InputIterator __first, _InputIterator __last) [inline]`

Assigns a range to a list.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.

This function fills a list with copies of the elements in the range `[__first, __last)`.

Note that the assignment completely changes the list and that the resulting list's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 734 of file `stl_list.h`.

4.612.3.4 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::list<_Tp, _Alloc>::assign (initializer_list<value_type> & __l) [inline]`

Assigns an `initializer_list` to a list.

Parameters

↩	An initializer_list of value_type.
↩	
↩	
↩	
/	

Replace the contents of the list with copies of the elements in the initializer_list `__l`. This is linear in `__l.size()`.

Definition at line 756 of file `stl_list.h`.

4.612.3.5 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> reference std::list<_Tp, _Alloc>::back ()`
`[inline], [noexcept]`

Returns a read/write reference to the data at the last element of the list.

Definition at line 956 of file `stl_list.h`.

4.612.3.6 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reference std::list<_Tp, _Alloc>::back ()`
`const [inline], [noexcept]`

Returns a read-only (constant) reference to the data at the last element of the list.

Definition at line 968 of file `stl_list.h`.

4.612.3.7 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> iterator std::list<_Tp, _Alloc>::begin ()`
`[inline], [noexcept]`

Returns a read/write iterator that points to the first element in the list. Iteration is done in ordinary element order.

Definition at line 771 of file `stl_list.h`.

Referenced by `std::list< __inp, __rebind_inp >::list()`, `std::operator==()`, and `std::list< __inp, __rebind_inp >::splice()`.

4.612.3.8 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_iterator std::list<_Tp, _Alloc>::begin ()`
`const [inline], [noexcept]`

Returns a read-only (constant) iterator that points to the first element in the list. Iteration is done in ordinary element order.

Definition at line 780 of file `stl_list.h`.

4.612.3.9 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_iterator std::list<_Tp, _Alloc>::cbegin ()`
`const [inline], [noexcept]`

Returns a read-only (constant) iterator that points to the first element in the list. Iteration is done in ordinary element order.

Definition at line 844 of file `stl_list.h`.

4.612.3.10 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_iterator std::list<_Tp, _Alloc>::cend () const [inline], [noexcept]`

Returns a read-only (constant) iterator that points one past the last element in the list. Iteration is done in ordinary element order.

Definition at line 853 of file `stl_list.h`.

4.612.3.11 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::list<_Tp, _Alloc>::clear () [inline], [noexcept]`

Erases all the elements. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1302 of file `stl_list.h`.

4.612.3.12 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reverse_iterator std::list<_Tp, _Alloc>::crbegin () const [inline], [noexcept]`

Returns a read-only (constant) reverse iterator that points to the last element in the list. Iteration is done in reverse element order.

Definition at line 862 of file `stl_list.h`.

4.612.3.13 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reverse_iterator std::list<_Tp, _Alloc>::crend () const [inline], [noexcept]`

Returns a read-only (constant) reverse iterator that points to one before the first element in the list. Iteration is done in reverse element order.

Definition at line 871 of file `stl_list.h`.

4.612.3.14 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> template<typename... _Args> iterator std::list<_Tp, _Alloc>::emplace (const_iterator __position, _Args &&... __args)`

Constructs object in list before specified iterator.

Parameters

<code>__position</code>	A <code>const_iterator</code> into the list.
<code>__args</code>	Arguments.

Returns

An iterator that points to the inserted data.

This function will insert an object of type `T` constructed with `T(std::forward<Args>(args)...) before the specified location`. Due to the nature of a list this operation can be done in constant time, and does not invalidate iterators and references.

4.612.3.15 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> bool std::list<_Tp, _Alloc>::empty () const`
`[inline], [noexcept]`

Returns true if the list is empty. (Thus `begin()` would equal `end()`.)

Definition at line 881 of file `stl_list.h`.

Referenced by `std::list< __inp, __rebind_inp >::splice()`.

4.612.3.16 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> iterator std::list<_Tp, _Alloc>::end ()`
`[inline], [noexcept]`

Returns a read/write iterator that points one past the last element in the list. Iteration is done in ordinary element order.

Definition at line 789 of file `stl_list.h`.

Referenced by `std::list< __inp, __rebind_inp >::list()`, `std::operator==()`, and `std::list< __inp, __rebind_inp >::splice()`.

4.612.3.17 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_iterator std::list<_Tp, _Alloc>::end () const`
`[inline], [noexcept]`

Returns a read-only (constant) iterator that points one past the last element in the list. Iteration is done in ordinary element order.

Definition at line 798 of file `stl_list.h`.

4.612.3.18 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> iterator std::list<_Tp, _Alloc>::erase (`
`const_iterator __position) [noexcept]`

Remove element at given position.

Parameters

<code>__position</code>	Iterator pointing to element to be erased.
-------------------------	--

Returns

An iterator pointing to the next element (or `end()`).

This function will erase the element at the given position and thus shorten the list by one.

Due to the nature of a list this operation can be done in constant time, and only invalidates iterators/references to the element being removed. The user is also cautioned that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

4.612.3.19 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> iterator std::list<_Tp, _Alloc>::erase (`
`const_iterator __first, const_iterator __last) [inline], [noexcept]`

Remove a range of elements.

Parameters

<code>__first</code>	Iterator pointing to the first element to be erased.
<code>__last</code>	Iterator pointing to one past the last element to be erased.

Returns

An iterator pointing to the element pointed to by *last* prior to erasing (or end()).

This function will erase the elements in the range [first,last) and shorten the list accordingly.

This operation is linear time in the size of the range and only invalidates iterators/references to the element being removed. The user is also cautioned that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1264 of file stl_list.h.

4.612.3.20 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> reference std::list<_Tp, _Alloc>::front ()`
`[inline], [noexcept]`

Returns a read/write reference to the data at the first element of the list.

Definition at line 940 of file stl_list.h.

4.612.3.21 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reference std::list<_Tp, _Alloc>::front () const`
`[inline], [noexcept]`

Returns a read-only (constant) reference to the data at the first element of the list.

Definition at line 948 of file stl_list.h.

4.612.3.22 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> allocator_type std::list<_Tp, _Alloc>::get_allocator () const`
`[inline], [noexcept]`

Get a copy of the memory allocation object.

Definition at line 762 of file stl_list.h.

4.612.3.23 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> iterator std::list<_Tp, _Alloc>::insert (const_iterator __position, const value_type & __x)`

Inserts given value into list before specified iterator.

Parameters

<code>__position</code>	A const_iterator into the list.
<code>__x</code>	Data to be inserted.

Returns

An iterator that points to the inserted data.

This function will insert a copy of the given value before the specified location. Due to the nature of a list this operation can be done in constant time, and does not invalidate iterators and references.

4.612.3.24 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> iterator std::list<_Tp, _Alloc>::insert (const_iterator __position, value_type && __x) [inline]`

Inserts given rvalue into list before specified iterator.

Parameters

<code>__position</code>	A const_iterator into the list.
<code>__x</code>	Data to be inserted.

Returns

An iterator that points to the inserted data.

This function will insert a copy of the given rvalue before the specified location. Due to the nature of a list this operation can be done in constant time, and does not invalidate iterators and references.

Definition at line 1116 of file `stl_list.h`.

4.612.3.25 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> iterator std::list<_Tp, _Alloc>::insert (const_iterator __p, initializer_list<value_type> __l) [inline]`

Inserts the contents of an initializer_list into list before specified const_iterator.

Parameters

<code>__p</code>	A const_iterator into the list.
<code>__l</code>	An initializer_list of value_type.

Returns

An iterator pointing to the first element inserted (or `__position`).

This function will insert copies of the data in the initializer_list `l` into the list before the location specified by `p`.

This operation is linear in the number of elements inserted and does not invalidate iterators and references.

Definition at line 1135 of file `stl_list.h`.

4.612.3.26 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> iterator std::list< _Tp, _Alloc >::insert (const_iterator __position, size_type __n, const value_type & __x)`

Inserts a number of copies of given data into the list.

Parameters

<code>__position</code>	A <code>const_iterator</code> into the list.
<code>__n</code>	Number of elements to be inserted.
<code>__x</code>	Data to be inserted.

Returns

An iterator pointing to the first element inserted (or `__position`).

This function will insert a specified number of copies of the given data before the location specified by *position*.

This operation is linear in the number of elements inserted and does not invalidate iterators and references.

4.612.3.27 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> template<typename _InputIterator, typename = std::RequireInputIter<_InputIterator>> iterator std::list< _Tp, _Alloc >::insert (const_iterator __position, _InputIterator __first, _InputIterator __last)`

Inserts a range into the list.

Parameters

<code>__position</code>	A <code>const_iterator</code> into the list.
<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.

Returns

An iterator pointing to the first element inserted (or `__position`).

This function will insert copies of the data in the range *[first,last)* into the list before the location specified by *position*.

This operation is linear in the number of elements inserted and does not invalidate iterators and references.

4.612.3.28 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> size_type std::list< _Tp, _Alloc >::max_size () const [inline],[noexcept]`

Returns the `size()` of the largest possible list.

Definition at line 891 of file `stl_list.h`.

4.612.3.29 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::list< _Tp, _Alloc >::merge (list< _Tp, _Alloc > && __x)`

Merge sorted lists.

Parameters

<code>__x</code>	Sorted list to merge.
------------------	-----------------------

Assumes that both `__x` and this list are sorted according to `operator<()`. Merges elements of `__x` into this list in sorted order, leaving `__x` empty when complete. Elements in this list precede elements in `__x` that are equal.

4.612.3.30 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> template<typename _StrictWeakOrdering > void std::list<_Tp, _Alloc>::merge (list<_Tp, _Alloc> && __x, _StrictWeakOrdering __comp)`

Merge sorted lists according to comparison function.

Template Parameters

<code>__StrictWeakOrdering</code>	Comparison function defining sort order.
-----------------------------------	--

Parameters

<code>__x</code>	Sorted list to merge.
<code>__comp</code>	Comparison functor.

Assumes that both `__x` and this list are sorted according to `StrictWeakOrdering`. Merges elements of `__x` into this list in sorted order, leaving `__x` empty when complete. Elements in this list precede elements in `__x` that are equivalent according to `StrictWeakOrdering()`.

4.612.3.31 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> list& std::list<_Tp, _Alloc>::operator= (const list<_Tp, _Alloc> & __x)`

List assignment operator.

No explicit dtor needed as the `_Base` dtor takes care of things. The `_Base` dtor only erases the elements, and note that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Parameters

<code>__x</code>	A list of identical element and allocator types.
------------------	--

All the elements of `__x` are copied, but unlike the copy constructor, the allocator object is not copied.

4.612.3.32 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> list& std::list<_Tp, _Alloc>::operator= (list<_Tp, _Alloc> && __x) [inline]`

List move assignment operator.

Parameters

<code>__x</code>	A list of identical element and allocator types.
<code>__x</code>	

The contents of `__x` are moved into this list (without copying). `__x` is a valid, but unspecified list

Definition at line 680 of file `stl_list.h`.

4.612.3.33 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> list& std::list<_Tp, _Alloc>::operator= (initializer_list<value_type> __l) [inline]`

List initializer list assignment operator.

Parameters

<code>__l</code>	An initializer_list of value_type.
<code>__l</code>	
<code>__l</code>	
<code>__l</code>	
<code>l</code>	

Replace the contents of the list with copies of the elements in the initializer_list `__l`. This is linear in `l.size()`.

Definition at line 697 of file `stl_list.h`.

4.612.3.34 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::list<_Tp, _Alloc>::pop_back () [inline], [noexcept]`

Removes last element.

This is a typical stack operation. It shrinks the list by one. Due to the nature of a list this operation can be done in constant time, and only invalidates iterators/references to the element being removed.

Note that no data is returned, and if the last element's data is needed, it should be retrieved before `pop_back()` is called.

Definition at line 1054 of file `stl_list.h`.

4.612.3.35 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::list<_Tp, _Alloc>::pop_front () [inline], [noexcept]`

Removes first element.

This is a typical stack operation. It shrinks the list by one. Due to the nature of a list this operation can be done in constant time, and only invalidates iterators/references to the element being removed.

Note that no data is returned, and if the first element's data is needed, it should be retrieved before `pop_front()` is called.

Definition at line 1014 of file `stl_list.h`.

4.612.3.36 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::list<_Tp, _Alloc>::push_back (const value_type & __x) [inline]`

Add data to the end of the list.

Parameters

$_ \leftrightarrow$	Data to be added.
$_x$	

This is a typical stack operation. The function creates an element at the end of the list and assigns the given data to it. Due to the nature of a list this operation can be done in constant time, and does not invalidate iterators and references.

Definition at line 1028 of file `stl_list.h`.

```
4.612.3.37  template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::list<_Tp, _Alloc>::push_front ( const
value_type & __x )  [inline]
```

Add data to the front of the list.

Parameters

$_ \leftrightarrow$	Data to be added.
$_x$	

This is a typical stack operation. The function creates an element at the front of the list and assigns the given data to it. Due to the nature of a list this operation can be done in constant time, and does not invalidate iterators and references.

Definition at line 987 of file `stl_list.h`.

```
4.612.3.38  template<typename _Tp, typename _Alloc = std::allocator<_Tp>> reverse_iterator std::list<_Tp, _Alloc
>::rbegin ( )  [inline], [noexcept]
```

Returns a read/write reverse iterator that points to the last element in the list. Iteration is done in reverse element order.

Definition at line 807 of file `stl_list.h`.

```
4.612.3.39  template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reverse_iterator std::list<_Tp, _Alloc
>::rbegin ( ) const  [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last element in the list. Iteration is done in reverse element order.

Definition at line 816 of file `stl_list.h`.

```
4.612.3.40  template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::list<_Tp, _Alloc>::remove ( const _Tp
& __value )
```

Remove all elements equal to value.

Parameters

$__value$	The value to remove.
-------------	----------------------

Removes every element in the list equal to *value*. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

4.612.3.41 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> template<typename _Predicate > void std::list<_Tp, _Alloc>::remove_if (_Predicate)`

Remove all elements satisfying a predicate.

Template Parameters

<code>_Predicate</code>	Unary predicate function or object.
-------------------------	-------------------------------------

Removes every element in the list for which the predicate returns true. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

4.612.3.42 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> reverse_iterator std::list<_Tp, _Alloc>::rend () [inline], [noexcept]`

Returns a read/write reverse iterator that points to one before the first element in the list. Iteration is done in reverse element order.

Definition at line 825 of file `stl_list.h`.

4.612.3.43 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reverse_iterator std::list<_Tp, _Alloc>::rend () const [inline], [noexcept]`

Returns a read-only (constant) reverse iterator that points to one before the first element in the list. Iteration is done in reverse element order.

Definition at line 834 of file `stl_list.h`.

4.612.3.44 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::list<_Tp, _Alloc>::resize (size_type __new_size)`

Resizes the list to the specified number of elements.

Parameters

<code>__new_size</code>	Number of elements the list should contain.
-------------------------	---

This function will resize the list to the specified number of elements. If the number is smaller than the list's current size the list is truncated, otherwise default constructed elements are appended.

4.612.3.45 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::list<_Tp, _Alloc>::resize (size_type __new_size, const value_type & __x)`

Resizes the list to the specified number of elements.

Parameters

<code>__new_size</code>	Number of elements the list should contain.
<code>__x</code>	Data with which new elements should be populated.

This function will resize the list to the specified number of elements. If the number is smaller than the list's current size the list is truncated, otherwise the list is extended and new elements are populated with given data.

4.612.3.46 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::list<_Tp, _Alloc>::reverse ()`
`[inline], [noexcept]`

Reverse the elements in list.

Reverse the order of elements in the list in linear time.

Definition at line 1576 of file `stl_list.h`.

4.612.3.47 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> size_type std::list<_Tp, _Alloc>::size () const`
`[inline], [noexcept]`

Returns the number of elements in the list.

Definition at line 886 of file `stl_list.h`.

4.612.3.48 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::list<_Tp, _Alloc>::sort ()`

Sort the elements.

Sorts the elements of this list in NlogN time. Equivalent elements remain in list order.

4.612.3.49 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> template<typename _StrictWeakOrdering > void`
`std::list<_Tp, _Alloc>::sort (_StrictWeakOrdering)`

Sort the elements according to comparison function.

Sorts the elements of this list in NlogN time. Equivalent elements remain in list order.

4.612.3.50 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::list<_Tp, _Alloc>::splice (`
`const_iterator __position, list<_Tp, _Alloc> && __x) [inline], [noexcept]`

Insert contents of another list.

Parameters

<code>__position</code>	Iterator referencing the element to insert before.
<code>__x</code>	Source list.

The elements of `__x` are inserted in constant time in front of the element referenced by `__position`. `__x` becomes an empty list.

Requires this != `__x`.

Definition at line 1322 of file `stl_list.h`.

```
4.612.3.51 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::list<_Tp, _Alloc>::splice (
    const_iterator __position, list<_Tp, _Alloc> && __x, const_iterator __i ) [inline], [noexcept]
```

Insert element from another list.

Parameters

<code>__position</code>	Const_iterator referencing the element to insert before.
<code>__x</code>	Source list.
<code>__i</code>	Const_iterator referencing the element to move.

Removes the element in list `__x` referenced by `__i` and inserts it into the current list before `__position`.

Definition at line 1354 of file `stl_list.h`.

```
4.612.3.52 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::list<_Tp, _Alloc>::splice (
    const_iterator __position, list<_Tp, _Alloc> & __x, const_iterator __i ) [inline], [noexcept]
```

Insert element from another list.

Parameters

<code>__position</code>	Const_iterator referencing the element to insert before.
<code>__x</code>	Source list.
<code>__i</code>	Const_iterator referencing the element to move.

Removes the element in list `__x` referenced by `__i` and inserts it into the current list before `__position`.

Definition at line 1393 of file `stl_list.h`.

```
4.612.3.53 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::list<_Tp, _Alloc>::splice (
    const_iterator __position, list<_Tp, _Alloc> && __x, const_iterator __first, const_iterator __last )
    [inline], [noexcept]
```

Insert range from another list.

Parameters

<code>__position</code>	Const_iterator referencing the element to insert before.
<code>__x</code>	Source list.
<code>__first</code>	Const_iterator referencing the start of range in x.
<code>__last</code>	Const_iterator referencing the end of range in x.

Removes elements in the range [`__first`,`__last`) and inserts them before `__position` in constant time.

Undefined if `__position` is in [`__first`,`__last`).

Definition at line 1412 of file `stl_list.h`.

4.612.3.54 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::list<_Tp, _Alloc>::splice (`
`const_iterator __position, list<_Tp, _Alloc> &__x, const_iterator __first, const_iterator __last)`
`[inline], [noexcept]`

Insert range from another list.

Parameters

<code>__position</code>	Const_iterator referencing the element to insert before.
<code>__x</code>	Source list.
<code>__first</code>	Const_iterator referencing the start of range in x.
<code>__last</code>	Const_iterator referencing the end of range in x.

Removes elements in the range [`__first`,`__last`) and inserts them before `__position` in constant time.

Undefined if `__position` is in [`__first`,`__last`).

Definition at line 1458 of file `stl_list.h`.

4.612.3.55 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::list<_Tp, _Alloc>::swap (list<_Tp,`
`_Alloc> &__x) [inline]`

Swaps data with another list.

Parameters

<code>__x</code>	A list of the same element and allocator types.
------------------	---

This exchanges the elements between two lists in constant time. Note that the global `std::swap()` function is specialized such that `std::swap(l1,l2)` will feed to this function.

Definition at line 1284 of file `stl_list.h`.

Referenced by `std::swap()`.

4.612.3.56 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::list<_Tp, _Alloc>::unique ()`

Remove consecutive duplicate elements.

For each consecutive set of elements with the same value, remove all but the first one. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

4.612.3.57 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> template<typename _BinaryPredicate > void
std::list<_Tp, _Alloc >::unique (_BinaryPredicate)`

Remove consecutive elements satisfying a predicate.

Template Parameters

<code>_BinaryPredicate</code>	Binary predicate function or object.
-------------------------------	--------------------------------------

For each consecutive set of elements [first,last) that satisfy predicate(first,i) where i is an iterator in [first,last), remove all but the first one. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

The documentation for this class was generated from the following file:

- [stl_list.h](#)

4.613 std::locale Class Reference

Classes

- class [facet](#)
- class [id](#)

Public Types

- typedef int [category](#)

Public Member Functions

- [locale](#) () throw ()
- [locale](#) (const [locale](#) &__other) throw ()
- [locale](#) (const char *__s)
- [locale](#) (const [locale](#) &__base, const char *__s, [category](#) __cat)
- [locale](#) (const [locale](#) &__base, const [locale](#) &__add, [category](#) __cat)
- template<typename _Facet >
[locale](#) (const [locale](#) &__other, _Facet *__f)
- [~locale](#) () throw ()
- template<typename _Facet >
[locale combine](#) (const [locale](#) &__other) const
- [string name](#) () const
- bool [operator!=](#) (const [locale](#) &__other) const throw ()
- template<typename _Char , typename _Traits , typename _Alloc >
bool [operator\(\)](#) (const [basic_string](#)< _Char, _Traits, _Alloc > &__s1, const [basic_string](#)< _Char, _Traits, _Alloc > &__s2) const
- const [locale](#) & [operator=](#) (const [locale](#) &__other) throw ()
- bool [operator==](#) (const [locale](#) &__other) const throw ()

Static Public Member Functions

- static const `locale & classic` ()
- static `locale global` (const `locale` & __loc)

Static Public Attributes

- static const `category none`
- static const `category ctype`
- static const `category numeric`
- static const `category collate`
- static const `category time`
- static const `category monetary`
- static const `category messages`
- static const `category all`

Friends

- template<typename _Cache >
struct **__use_cache**
- class **_Impl**
- class **facet**
- template<typename _Facet >
bool **has_facet** (const `locale` &) throw ()
- template<typename _Facet >
const _Facet & **use_facet** (const `locale` &)

4.613.1 Detailed Description

Container class for localization functionality.

The locale class is first a class wrapper for C library locales. It is also an extensible container for user-defined localization. A locale is a collection of facets that implement various localization features such as money, time, and number printing.

Constructing C++ locales does not change the C library locale.

This library supports efficient construction and copying of locales through a reference counting implementation of the locale class.

Definition at line 62 of file `locale_classes.h`.

4.613.2 Member Typedef Documentation

4.613.2.1 typedef int std::locale::category

Definition of `locale::category`.

Definition at line 67 of file `locale_classes.h`.

4.613.3 Constructor & Destructor Documentation

4.613.3.1 std::locale::locale () throw)

Default constructor.

Constructs a copy of the global locale. If no locale has been explicitly set, this is the C locale.

Referenced by operator!=().

4.613.3.2 std::locale::locale (const locale & __other) throw)

Copy constructor.

Constructs a copy of *other*.

Parameters

<code>__other</code>	The locale to copy.
----------------------	---------------------

4.613.3.3 std::locale::locale (const char * __s) [explicit]

Named locale constructor.

Constructs a copy of the named C library locale.

Parameters

<code>__s</code>	Name of the locale to construct.
------------------	----------------------------------

Exceptions

<code>std::runtime_error</code>	if <code>__s</code> is null or an undefined locale.
---------------------------------	---

4.613.3.4 std::locale::locale (const locale & __base, const char * __s, category __cat)

Construct locale with facets from another locale.

Constructs a copy of the locale *base*. The facets specified by *cat* are replaced with those from the locale named by *s*. If *base* is named, this locale instance will also be named.

Parameters

<code>__base</code>	The locale to copy.
<code>__s</code>	Name of the locale to use facets from.
<code>__cat</code>	Set of categories defining the facets to use from <code>__s</code> .

Exceptions

<code>std::runtime_error</code>	if <code>__s</code> is null or an undefined locale.
---------------------------------	---

4.613.3.5 `std::locale::locale (const locale & __base, const locale & __add, category __cat)`

Construct locale with facets from another locale.

Constructs a copy of the locale *base*. The facets specified by *cat* are replaced with those from the locale *add*. If *base* and *add* are named, this locale instance will also be named.

Parameters

<code>__base</code>	The locale to copy.
<code>__add</code>	The locale to use facets from.
<code>__cat</code>	Set of categories defining the facets to use from add.

4.613.3.6 `template<typename _Facet > std::locale::locale (const locale & __other, _Facet * __f)`

Construct locale with another facet.

Constructs a copy of the locale *__other*. The facet *__f* is added to *__other*, replacing an existing facet of type *Facet* if there is one. If *__f* is null, this locale is a copy of *__other*.

Parameters

<code>__other</code>	The locale to copy.
<code>__f</code>	The facet to add in.

4.613.3.7 `std::locale::~~locale () throw`

Locale destructor.

4.613.4 Member Function Documentation

4.613.4.1 `static const locale& std::locale::classic () [static]`

Return reference to the C locale.

Referenced by operator!=().

4.613.4.2 `template<typename _Facet > locale std::locale::combine (const locale & __other) const`

Construct locale with another facet.

Constructs and returns a new copy of this locale. Adds or replaces an existing facet of type *Facet* from the locale *other* into the new locale.

Template Parameters

<code>_Facet</code>	The facet type to copy from other
---------------------	-----------------------------------

Parameters

<code>__other</code>	The locale to copy from.
----------------------	--------------------------

Returns

Newly constructed locale.

Exceptions

<code>std::runtime_error</code>	if <code>__other</code> has no facet of type <code>_Facet</code> .
---------------------------------	--

4.613.4.3 static locale std::locale::global (const locale & __loc) [static]

Set global locale.

This function sets the global locale to the argument and returns a copy of the previous global locale. If the argument has a name, it will also call `std::setlocale(LC_ALL, loc.name())`.

Parameters

<code>__loc</code>	The new locale to make global.
--------------------	--------------------------------

Returns

Copy of the old global locale.

Referenced by operator!=().

4.613.4.4 string std::locale::name () const

Return locale name.

Returns

Locale name or "*" if unnamed.

4.613.4.5 bool std::locale::operator!=(const locale & __other) const throw) [inline]

Locale inequality.

Parameters

<code>__other</code>	The locale to compare against.
----------------------	--------------------------------

Returns

`! (*this == __other)`

Definition at line 235 of file `locale_classes.h`.

References `__gnu_debug::__base()`, `classic()`, `global()`, `locale()`, `operator()()`, and `operator==()`.

4.613.4.6 `template<typename _Char, typename _Traits, typename _Alloc> bool std::locale::operator() (const basic_string<_Char, _Traits, _Alloc> & __s1, const basic_string<_Char, _Traits, _Alloc> & __s2) const`

Compare two strings according to collate.

Template operator to compare two strings using the compare function of the collate facet in this locale. One use is to provide the locale to the sort function. For example, a vector `v` of strings could be sorted according to locale `loc` by doing:

```
std::sort(v.begin(), v.end(), loc);
```

Parameters

<code>__s1</code>	First string to compare.
<code>__s2</code>	Second string to compare.

Returns

True if `collate<_Char>` facet compares `__s1 < __s2`, else false.

Referenced by `operator!=()`.

4.613.4.7 `const locale& std::locale::operator= (const locale & __other) throw`

Assignment operator.

Set this locale to be a copy of *other*.

Parameters

<code>__other</code>	The locale to copy.
----------------------	---------------------

Returns

A reference to this locale.

Referenced by `std::locale::facet::facet()`, and `std::locale::id::id()`.

4.613.4.8 bool std::locale::operator==(const locale & __other) const throw)

Locale equality.

Parameters

<code>__other</code>	The locale to compare against.
----------------------	--------------------------------

Returns

True if `other` and this refer to the same locale instance, are copies, or have the same name. False otherwise.

Referenced by `operator!=()`.

4.613.5 Member Data Documentation**4.613.5.1 const category std::locale::all [static]**

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match `_S_facet_categories` definition in `locale.cc`

Definition at line 105 of file `locale_classes.h`.

4.613.5.2 const category std::locale::collate [static]

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match `_S_facet_categories` definition in `locale.cc`

Definition at line 101 of file `locale_classes.h`.

4.613.5.3 `const category std::locale::ctype` `[static]`

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match `_S_facet_categories` definition in `locale.cc`

Definition at line 99 of file `locale_classes.h`.

4.613.5.4 `const category std::locale::messages` `[static]`

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match `_S_facet_categories` definition in `locale.cc`

Definition at line 104 of file `locale_classes.h`.

4.613.5.5 `const category std::locale::monetary` `[static]`

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match `_S_facet_categories` definition in `locale.cc`

Definition at line 103 of file `locale_classes.h`.

4.613.5.6 `const category std::locale::none` `[static]`

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match `_S_facet_categories` definition in `locale.cc`

Definition at line 98 of file `locale_classes.h`.

4.613.5.7 `const category std::locale::numeric` `[static]`

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match `_S_facet_categories` definition in `locale.cc`

Definition at line 100 of file `locale_classes.h`.

4.613.5.8 const category std::locale::time [static]

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match `_S_facet_categories` definition in `locale.cc`

Definition at line 102 of file `locale_classes.h`.

The documentation for this class was generated from the following file:

- [locale_classes.h](#)

Static Protected Member Functions

- static `__c_locale _S_clone_c_locale (__c_locale &__cloc) throw ()`
- static void `_S_create_c_locale (__c_locale &__cloc, const char *__s, __c_locale __old=0)`
- static void `_S_destroy_c_locale (__c_locale &__cloc)`
- static `__c_locale _S_get_c_locale ()`
- static const char * `_S_get_c_name () throw ()`
- static `__c_locale _S_lc_ctype_c_locale (__c_locale __cloc, const char *__s)`

Friends

- class `locale`
- class `locale::_Impl`

4.614.1 Detailed Description

Localization functionality base class.

The facet class is the base class for a localization feature, such as money, time, and number printing. It provides common support for facets and reference management.

Facets may not be copied or assigned.

Definition at line 338 of file `locale_classes.h`.

4.614.2 Constructor & Destructor Documentation

4.614.2.1 `std::locale::facet::facet (size_t __refs = 0) throw ()` `[inline]`, `[explicit]`, `[protected]`

Facet constructor.

This is the constructor provided by the standard. If `refs` is 0, the facet is destroyed when the last referencing locale is destroyed. Otherwise the facet will never be destroyed.

Parameters

<code>__refs</code>	The initial value for reference count.
---------------------	--

Definition at line 370 of file `locale_classes.h`.

References `std::locale::operator=()`.

Referenced by `std::__ctype_abstract_base< wchar_t >::narrow()`.

4.614.2.2 `virtual std::locale::facet::~~facet () [protected],[virtual]`

Facet destructor.

The documentation for this class was generated from the following file:

- [locale_classes.h](#)

4.615 `std::locale::id` Class Reference

Public Member Functions

- `id ()`
- `size_t _M_id () const throw ()`

Friends

- `template<typename _Facet >`
`bool has_facet (const locale &) throw ()`
- `class locale`
- `class locale::_Impl`
- `template<typename _Facet >`
`const _Facet & use_facet (const locale &)`

4.615.1 Detailed Description

Facet ID class.

The ID class provides facets with an index used to identify them. Every facet class must define a public static member `locale::id`, or be derived from a facet that provides this member, otherwise the facet cannot be used in a locale. The `locale::id` ensures that each class type gets a unique identifier.

Definition at line 436 of file `locale_classes.h`.

4.615.2 Constructor & Destructor Documentation

4.615.2.1 `std::locale::id::id () [inline]`

Constructor.

Definition at line 467 of file `locale_classes.h`.

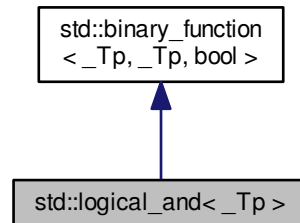
References `std::locale::operator=()`.

The documentation for this class was generated from the following file:

- [locale_classes.h](#)

4.616 std::logical_and< _Tp > Struct Template Reference

Inheritance diagram for std::logical_and< _Tp >:



Public Types

- typedef `_Tp` [first_argument_type](#)
- typedef `bool` [result_type](#)
- typedef `_Tp` [second_argument_type](#)

Public Member Functions

- `bool` **operator()** (`const _Tp &__x`, `const _Tp &__y`) `const`

4.616.1 Detailed Description

```
template<typename _Tp>  
struct std::logical_and< _Tp >
```

One of the [Boolean operations functors](#).

Definition at line 501 of file `stl_function.h`.

4.616.2 Member Typedef Documentation

4.616.2.1 typedef `_Tp` `std::binary_function< _Tp, _Tp, bool >::first_argument_type` [\[inherited\]](#)

`first_argument_type` is the type of the first argument

Definition at line 121 of file `stl_function.h`.

4.616.2.2 `typedef bool std::binary_function<_Tp, _Tp, bool>::result_type` [inherited]

`result_type` is the return type

Definition at line 127 of file `stl_function.h`.

4.616.2.3 `typedef _Tp std::binary_function<_Tp, _Tp, bool>::second_argument_type` [inherited]

`second_argument_type` is the type of the second argument

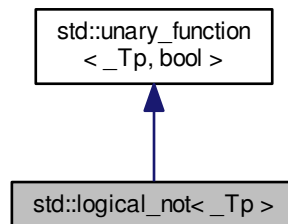
Definition at line 124 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl_function.h](#)

4.617 `std::logical_not<_Tp>` Struct Template Reference

Inheritance diagram for `std::logical_not<_Tp>`:



Public Types

- `typedef _Tp` [argument_type](#)
- `typedef bool` [result_type](#)

Public Member Functions

- `bool` **operator()** (`const _Tp &__x`) `const`

4.617.1 Detailed Description

```
template<typename _Tp>
struct std::logical_not<_Tp>
```

One of the [Boolean operations functors](#).

Definition at line 519 of file `stl_function.h`.

4.617.2 Member Typedef Documentation

4.617.2.1 `typedef _Tp std::unary_function<_Tp, bool>::argument_type` [\[inherited\]](#)

`argument_type` is the type of the argument

Definition at line 108 of file `stl_function.h`.

4.617.2.2 `typedef bool std::unary_function<_Tp, bool>::result_type` [\[inherited\]](#)

`result_type` is the return type

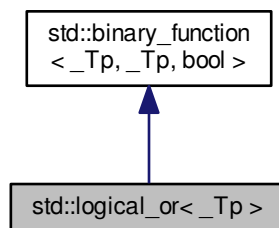
Definition at line 111 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl_function.h](#)

4.618 `std::logical_or<_Tp>` Struct Template Reference

Inheritance diagram for `std::logical_or<_Tp>`:



Public Types

- typedef `_Tp` [first_argument_type](#)
- typedef `bool` [result_type](#)
- typedef `_Tp` [second_argument_type](#)

Public Member Functions

- `bool` **operator()** (`const _Tp &__x`, `const _Tp &__y`) `const`

4.618.1 Detailed Description

```
template<typename _Tp>  
struct std::logical_or<_Tp >
```

One of the [Boolean operations functors](#).

Definition at line 510 of file `stl_function.h`.

4.618.2 Member Typedef Documentation

4.618.2.1 `typedef _Tp std::binary_function<_Tp, _Tp, bool >::first_argument_type` `[inherited]`

`first_argument_type` is the type of the first argument

Definition at line 121 of file `stl_function.h`.

4.618.2.2 `typedef bool std::binary_function<_Tp, _Tp, bool >::result_type` `[inherited]`

`result_type` is the return type

Definition at line 127 of file `stl_function.h`.

4.618.2.3 `typedef _Tp std::binary_function<_Tp, _Tp, bool >::second_argument_type` `[inherited]`

`second_argument_type` is the type of the second argument

Definition at line 124 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl_function.h](#)

4.619 std::lognormal_distribution<_RealType> Class Template Reference

Classes

- struct [param_type](#)

Public Types

- typedef _RealType [result_type](#)

Public Member Functions

- **lognormal_distribution** (_RealType __m=_RealType(0), _RealType __s=_RealType(1))
- **lognormal_distribution** (const [param_type](#) &__p)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator >
void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator >
void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const [param_type](#) &__p)
- template<typename _UniformRandomNumberGenerator >
void **__generate** ([result_type](#) *__f, [result_type](#) *__t, _UniformRandomNumberGenerator &__urng, const [param_type](#) &__p)
- _RealType **m** () const
- [result_type](#) **max** () const
- [result_type](#) **min** () const
- template<typename _UniformRandomNumberGenerator >
[result_type](#) **operator()** (_UniformRandomNumberGenerator &__urng)
- template<typename _UniformRandomNumberGenerator >
[result_type](#) **operator()** (_UniformRandomNumberGenerator &__urng, const [param_type](#) &__p)
- [param_type](#) **param** () const
- void **param** (const [param_type](#) &__param)
- void **reset** ()
- _RealType **s** () const

Friends

- template<typename _RealType1, typename _CharT, typename _Traits >
std::basic_ostream< _CharT, _Traits > & **operator<<** (std::basic_ostream< _CharT, _Traits > &__os, const [std::lognormal_distribution](#)< _RealType1 > &__x)
- bool **operator==** (const [lognormal_distribution](#) &__d1, const [lognormal_distribution](#) &__d2)
- template<typename _RealType1, typename _CharT, typename _Traits >
std::basic_istream< _CharT, _Traits > & **operator>>** (std::basic_istream< _CharT, _Traits > &__is, [std::lognormal_distribution](#)< _RealType1 > &__x)

4.619.1 Detailed Description

```
template<typename _RealType = double>
class std::lognormal_distribution< _RealType >
```

A lognormal_distribution random number distribution.

The formula for the normal probability mass function is

$$p(x|m, s) = \frac{1}{sx\sqrt{2\pi}} \exp - \frac{(\ln x - m)^2}{2s^2}$$

Definition at line 2300 of file random.h.

4.619.2 Member Typedef Documentation

4.619.2.1 `template<typename _RealType = double> typedef _RealType std::lognormal_distribution< _RealType >::result_type`

The type of the range of the distribution.

Definition at line 2303 of file random.h.

4.619.3 Member Function Documentation

4.619.3.1 `template<typename _RealType = double> result_type std::lognormal_distribution< _RealType >::max () const [inline]`

Returns the least upper bound value of the distribution.

Definition at line 2391 of file random.h.

References std::max().

4.619.3.2 `template<typename _RealType = double> result_type std::lognormal_distribution< _RealType >::min () const [inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 2384 of file random.h.

4.619.3.3 `template<typename _RealType = double> template<typename _UniformRandomNumberGenerator > result_type std::lognormal_distribution< _RealType >::operator() (_UniformRandomNumberGenerator & __urng) [inline]`

Generating functions.

Definition at line 2399 of file random.h.

4.619.3.4 `template<typename _RealType = double> param_type std::lognormal_distribution< _RealType >::param () const [inline]`

Returns the parameter set of the distribution.

Definition at line 2369 of file random.h.

4.619.3.5 `template<typename _RealType = double> void std::lognormal_distribution< _RealType >::param (const param_type & __param) [inline]`

Sets the parameter set of the distribution.

Parameters

<code>__param</code>	The new parameter set of the distribution.
----------------------	--

Definition at line 2377 of file random.h.

4.619.3.6 `template<typename _RealType = double> void std::lognormal_distribution<_RealType>::reset ()`
`[inline]`

Resets the distribution state.

Definition at line 2351 of file random.h.

4.619.4 Friends And Related Function Documentation

4.619.4.1 `template<typename _RealType = double> template<typename _RealType1, typename _CharT, typename _Traits`
`> std::basic_ostream<_CharT, _Traits>& operator<< (std::basic_ostream<_CharT, _Traits> & __os, const`
`std::lognormal_distribution<_RealType1> & __x) [friend]`

Inserts a lognormal_distribution random number distribution `__x` into the output stream `__os`.

Parameters

<code>__os</code>	An output stream.
<code>__x</code>	A lognormal_distribution random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

4.619.4.2 `template<typename _RealType = double> bool operator==(const lognormal_distribution<_RealType> & __d1,`
`const lognormal_distribution<_RealType> & __d2) [friend]`

Return true if two lognormal distributions have the same parameters and the sequences that would be generated are equal.

Definition at line 2436 of file random.h.

4.619.4.3 `template<typename _RealType = double> template<typename _RealType1, typename _CharT, typename`
`_Traits> std::basic_istream<_CharT, _Traits>& operator>> (std::basic_istream<_CharT, _Traits> & __is,`
`std::lognormal_distribution<_RealType1> & __x) [friend]`

Extracts a lognormal_distribution random number distribution `__x` from the input stream `__is`.

Parameters

<code>_↔ _is</code>	An input stream.
<code>_↔ _x</code>	A lognormal_distribution random number generator engine.

Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following file:

- [random.h](#)

4.620 `std::lognormal_distribution<_RealType>::param_type` Struct Reference

Public Types

- typedef [lognormal_distribution](#)<_RealType> **distribution_type**

Public Member Functions

- **param_type** (_RealType __m=_RealType(0), _RealType __s=_RealType(1))
- _RealType **m** () const
- _RealType **s** () const

Friends

- bool **operator==** (const [param_type](#) &__p1, const [param_type](#) &__p2)

4.620.1 Detailed Description

```
template<typename _RealType = double>
struct std::lognormal_distribution<_RealType>::param_type
```

Parameter type.

Definition at line 2309 of file random.h.

The documentation for this struct was generated from the following file:

- [random.h](#)

4.621 `std::map<_Key, _Tp, _Compare, _Alloc>` Class Template Reference

Public Types

- typedef `_Alloc` **allocator_type**
- typedef `_Rep_type::const_iterator` **const_iterator**
- typedef `_Alloc_traits::const_pointer` **const_pointer**
- typedef `_Alloc_traits::const_reference` **const_reference**
- typedef `_Rep_type::const_reverse_iterator` **const_reverse_iterator**
- typedef `_Rep_type::difference_type` **difference_type**
- typedef `_Rep_type::iterator` **iterator**
- typedef `_Compare` **key_compare**
- typedef `_Key` **key_type**
- typedef `_Tp` **mapped_type**
- typedef `_Alloc_traits::pointer` **pointer**
- typedef `_Alloc_traits::reference` **reference**
- typedef `_Rep_type::reverse_iterator` **reverse_iterator**
- typedef `_Rep_type::size_type` **size_type**
- typedef `std::pair< const _Key, _Tp >` **value_type**

Public Member Functions

- `map()`
- `map(const _Compare &__comp, const allocator_type &__a=allocator_type())`
- `map(const map &__x)`
- `map(map &&__x) noexcept(is_nothrow_copy_constructible< _Compare >::value)`
- `map(initializer_list< value_type > __l, const _Compare &__comp=_Compare(), const allocator_type &__a=allocator_type())`
- `map(const allocator_type &__a)`
- `map(const map &__m, const allocator_type &__a)`
- `map(map &&__m, const allocator_type &__a) noexcept(is_nothrow_copy_constructible< _Compare >::value && _Alloc_traits::_S_always_equal())`
- `map(initializer_list< value_type > __l, const allocator_type &__a)`
- `template<typename _InputIterator>`
`map(_InputIterator __first, _InputIterator __last, const allocator_type &__a)`
- `template<typename _InputIterator>`
`map(_InputIterator __first, _InputIterator __last)`
- `template<typename _InputIterator>`
`map(_InputIterator __first, _InputIterator __last, const _Compare &__comp, const allocator_type &__a=allocator_type())`
- `mapped_type &at(const key_type &__k)`
- `const mapped_type &at(const key_type &__k) const`
- `iterator begin()` `noexcept`
- `const_iterator begin()` `const` `noexcept`
- `const_iterator cbegin()` `const` `noexcept`
- `const_iterator cend()` `const` `noexcept`
- `void clear()` `noexcept`
- `size_type count(const key_type &__x) const`
- `const_reverse_iterator crbegin()` `const` `noexcept`
- `const_reverse_iterator crend()` `const` `noexcept`

- `template<typename... _Args>`
`std::pair< iterator, bool > emplace (_Args &&... __args)`
- `template<typename... _Args>`
`iterator emplace_hint (const_iterator __pos, _Args &&... __args)`
- `bool empty () const noexcept`
- `iterator end () noexcept`
- `const_iterator end () const noexcept`
- `std::pair< iterator, iterator > equal_range (const key_type &__x)`
- `std::pair< const_iterator, const_iterator > equal_range (const key_type &__x) const`
- `iterator erase (const_iterator __position)`
- `_GLIBCXX_ABI_TAG_CXX11 iterator erase (iterator __position)`
- `size_type erase (const key_type &__x)`
- `iterator erase (const_iterator __first, const_iterator __last)`
- `iterator find (const key_type &__x)`
- `const_iterator find (const key_type &__x) const`
- `allocator_type get_allocator () const noexcept`
- `std::pair< iterator, bool > insert (const value_type &__x)`
- `template<typename _Pair, typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type>`
`std::pair< iterator, bool > insert (_Pair &&__x)`
- `void insert (std::initializer_list< value_type > __list)`
- `iterator insert (const_iterator __position, const value_type &__x)`
- `template<typename _Pair, typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type>`
`iterator insert (const_iterator __position, _Pair &&__x)`
- `template<typename _InputIterator >`
`void insert (_InputIterator __first, _InputIterator __last)`
- `key_compare key_comp () const`
- `iterator lower_bound (const key_type &__x)`
- `const_iterator lower_bound (const key_type &__x) const`
- `size_type max_size () const noexcept`
- `map & operator= (const map &__x)`
- `map & operator= (map &&__x) noexcept(_Alloc_traits::_S_nothrow_move())`
- `map & operator= (initializer_list< value_type > __l)`
- `mapped_type & operator[] (const key_type &__k)`
- `mapped_type & operator[] (key_type &&__k)`
- `reverse_iterator rbegin () noexcept`
- `const_reverse_iterator rbegin () const noexcept`
- `reverse_iterator rend () noexcept`
- `const_reverse_iterator rend () const noexcept`
- `size_type size () const noexcept`
- `void swap (map &__x) noexcept(_Alloc_traits::_S_nothrow_swap())`
- `iterator upper_bound (const key_type &__x)`
- `const_iterator upper_bound (const key_type &__x) const`
- `value_compare value_comp () const`

Friends

- `template<typename _K1, typename _T1, typename _C1, typename _A1 >`
`bool operator< (const map< _K1, _T1, _C1, _A1 > &, const map< _K1, _T1, _C1, _A1 > &)`
- `template<typename _K1, typename _T1, typename _C1, typename _A1 >`
`bool operator== (const map< _K1, _T1, _C1, _A1 > &, const map< _K1, _T1, _C1, _A1 > &)`

4.621.1 Detailed Description

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp> >>
class std::map< _Key, _Tp, _Compare, _Alloc >
```

A standard container made up of (key,value) pairs, which can be retrieved based on a key, in logarithmic time.

Template Parameters

<code>_Key</code>	Type of key objects.
<code>_Tp</code>	Type of mapped objects.
<code>_Compare</code>	Comparison function object type, defaults to <code>less<_Key></code> .
<code>_Alloc</code>	Allocator type, defaults to <code>allocator<pair<const _Key, _Tp></code> .

Meets the requirements of a [container](#), a [reversible container](#), and an [associative container](#) (using unique keys). For a `map<Key, T>` the `key_type` is `Key`, the `mapped_type` is `T`, and the `value_type` is `std::pair<const Key, T>`.

Maps support bidirectional iterators.

The private tree data is declared exactly the same way for `map` and `multimap`; the distinction is made entirely in how the tree functions are called (`*_unique` versus `*_equal`, same as the standard).

Definition at line 96 of file `stl_map.h`.

4.621.2 Constructor & Destructor Documentation

```
4.621.2.1 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >> std::map< _Key, _Tp, _Compare, _Alloc >::map ( ) [inline]
```

Default constructor creates no elements.

Definition at line 162 of file `stl_map.h`.

```
4.621.2.2 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >> std::map< _Key, _Tp, _Compare, _Alloc >::map ( const _Compare &
__comp, const allocator_type & __a = allocator_type() ) [inline], [explicit]
```

Creates a map with no elements.

Parameters

<code>__comp</code>	A comparison object.
<code>__a</code>	An allocator object.

Definition at line 171 of file stl_map.h.

```
4.621.2.3 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> std::map<_Key, _Tp, _Compare, _Alloc>::map ( const map<
_Key, _Tp, _Compare, _Alloc> &__x ) [inline]
```

Map copy constructor.

Parameters

<code>__x</code>	A map of identical element and allocator types.
------------------	---

The newly-created map uses a copy of the allocation object used by `__x`.

Definition at line 182 of file stl_map.h.

```
4.621.2.4 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> std::map<_Key, _Tp, _Compare, _Alloc>::map ( map<_Key, _Tp,
_Compare, _Alloc> &&__x ) [inline], [noexcept]
```

Map move constructor.

Parameters

<code>__x</code>	A map of identical element and allocator types.
------------------	---

The newly-created map contains the exact contents of `__x`. The contents of `__x` are a valid, but unspecified map.

Definition at line 193 of file stl_map.h.

```
4.621.2.5 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> std::map<_Key, _Tp, _Compare, _Alloc>::map (
initializer_list<value_type> __l, const _Compare &__comp = _Compare(), const allocator_type &__a =
allocator_type() ) [inline]
```

Builds a map from an `initializer_list`.

Parameters

<code>__l</code>	An <code>initializer_list</code> .
<code>__comp</code>	A comparison object.
<code>__a</code>	An allocator object.

Create a map consisting of copies of the elements in the `initializer_list` `__l`. This is linear in `N` if the range is already sorted, and `NlogN` otherwise (where `N` is `__l.size()`).

Definition at line 208 of file stl_map.h.

```
4.621.2.6 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> std::map<_Key, _Tp, _Compare, _Alloc>::map ( const
allocator_type & __a ) [inline], [explicit]
```

Allocator-extended default constructor.

Definition at line 216 of file stl_map.h.

```
4.621.2.7 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> std::map<_Key, _Tp, _Compare, _Alloc>::map ( const map<
_Key, _Tp, _Compare, _Alloc> & __m, const allocator_type & __a ) [inline]
```

Allocator-extended copy constructor.

Definition at line 220 of file stl_map.h.

```
4.621.2.8 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> std::map<_Key, _Tp, _Compare, _Alloc>::map ( map<_Key, _Tp,
_Compare, _Alloc> && __m, const allocator_type & __a ) [inline], [noexcept]
```

Allocator-extended move constructor.

Definition at line 224 of file stl_map.h.

```
4.621.2.9 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> std::map<_Key, _Tp, _Compare, _Alloc>::map ( initializer_list<
value_type> & __l, const allocator_type & __a ) [inline]
```

Allocator-extended initializer-list constructor.

Definition at line 230 of file stl_map.h.

```
4.621.2.10 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> template<typename _InputIterator> std::map<_Key, _Tp,
_Compare, _Alloc>::map ( _InputIterator __first, _InputIterator __last, const allocator_type & __a ) [inline]
```

Allocator-extended range constructor.

Definition at line 236 of file stl_map.h.

```
4.621.2.11 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> template<typename _InputIterator> std::map<_Key, _Tp,
_Compare, _Alloc>::map ( _InputIterator __first, _InputIterator __last ) [inline]
```

Builds a map from a range.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.

Create a map consisting of copies of the elements from [`__first`,`__last`). This is linear in N if the range is already sorted, and NlogN otherwise (where N is distance(`__first`,`__last`)).

Definition at line 253 of file `stl_map.h`.

```
4.621.2.12  template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >> template<typename _InputIterator > std::map< _Key, _Tp,
_Compare, _Alloc >::map ( _InputIterator __first, _InputIterator __last, const _Compare & __comp, const
allocator_type & __a = allocator_type() ) [inline]
```

Builds a map from a range.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__comp</code>	A comparison functor.
<code>__a</code>	An allocator object.

Create a map consisting of copies of the elements from [`__first`,`__last`). This is linear in N if the range is already sorted, and NlogN otherwise (where N is distance(`__first`,`__last`)).

Definition at line 270 of file `stl_map.h`.

4.621.3 Member Function Documentation

```
4.621.3.1  template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >> mapped_type& std::map< _Key, _Tp, _Compare, _Alloc >::at ( const
key_type & __k ) [inline]
```

Access to map data.

Parameters

<code>__k</code>	The key for which data should be retrieved.
------------------	---

Returns

A reference to the data whose key is equivalent to `__k`, if such a data is present in the map.

Exceptions

<code>std::out_of_range</code>	If no such data is present.
--------------------------------	-----------------------------

Definition at line 536 of file stl_map.h.

References `std::map< _Key, _Tp, _Compare, _Alloc >::end()`, `std::map< _Key, _Tp, _Compare, _Alloc >::key_comp()`, and `std::map< _Key, _Tp, _Compare, _Alloc >::lower_bound()`.

4.621.3.2 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> iterator std::map< _Key, _Tp, _Compare, _Alloc >::begin ()`
`[inline], [noexcept]`

Returns a read/write iterator that points to the first pair in the map. Iteration is done in ascending order according to the keys.

Definition at line 355 of file stl_map.h.

4.621.3.3 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> const_iterator std::map< _Key, _Tp, _Compare, _Alloc >::begin ()`
`const [inline], [noexcept]`

Returns a read-only (constant) iterator that points to the first pair in the map. Iteration is done in ascending order according to the keys.

Definition at line 364 of file stl_map.h.

4.621.3.4 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> const_iterator std::map< _Key, _Tp, _Compare, _Alloc >::cbegin ()`
`const [inline], [noexcept]`

Returns a read-only (constant) iterator that points to the first pair in the map. Iteration is done in ascending order according to the keys.

Definition at line 428 of file stl_map.h.

4.621.3.5 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> const_iterator std::map< _Key, _Tp, _Compare, _Alloc >::cend ()`
`const [inline], [noexcept]`

Returns a read-only (constant) iterator that points one past the last pair in the map. Iteration is done in ascending order according to the keys.

Definition at line 437 of file stl_map.h.

4.621.3.6 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >> void std::map<_Key, _Tp, _Compare, _Alloc >::clear ()
[inline], [noexcept]`

Erases all elements in a map. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 826 of file `stl_map.h`.

Referenced by `std::map<_Key, _Tp, _Compare, _Alloc >::operator=()`.

4.621.3.7 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >> size_type std::map<_Key, _Tp, _Compare, _Alloc >::count (const
key_type & __x) const [inline]`

Finds the number of elements with given key.

Parameters

<code>_↔</code>	Key of (key, value) pairs to be located.
<code>_X</code>	

Returns

Number of elements with specified key.

This function only makes sense for multimaps; for map the result will either be 0 (not present) or 1 (present).

Definition at line 886 of file stl_map.h.

```
4.621.3.8 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> const_reverse_iterator std::map<_Key, _Tp, _Compare, _Alloc
>::crbegin( ) const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last pair in the map. Iteration is done in descending order according to the keys.

Definition at line 446 of file stl_map.h.

```
4.621.3.9 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> const_reverse_iterator std::map<_Key, _Tp, _Compare, _Alloc
>::crend( ) const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first pair in the map. Iteration is done in descending order according to the keys.

Definition at line 455 of file stl_map.h.

```
4.621.3.10 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> template<typename... _Args> std::pair<iterator, bool>
std::map<_Key, _Tp, _Compare, _Alloc>::emplace( _Args &&... _args ) [inline]
```

Attempts to build and insert a std::pair into the map.

Parameters

<code>__args</code>	Arguments used to generate a new pair instance (see std::piecewise_construct for passing arguments to each part of the pair constructor).
---------------------	---

Returns

A pair, of which the first element is an iterator that points to the possibly inserted pair, and the second is a bool that is true if the pair was actually inserted.

This function attempts to build and insert a (key, value) pair into the map. A map relies on unique keys and thus a pair is only inserted if its first element (the key) is not already present in the map.

Insertion requires logarithmic time.

Definition at line 575 of file `stl_map.h`.

```
4.621.3.11 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> template<typename... _Args> iterator std::map<_Key, _Tp,
_Compare, _Alloc >::emplace_hint ( const_iterator __pos, _Args &&... __args ) [inline]
```

Attempts to build and insert a `std::pair` into the map.

Parameters

<code>__pos</code>	An iterator that serves as a hint as to where the pair should be inserted.
<code>__args</code>	Arguments used to generate a new pair instance (see <code>std::piecewise_construct</code> for passing arguments to each part of the pair constructor).

Returns

An iterator that points to the element with key of the `std::pair` built from `__args` (may or may not be that `std::pair`).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument `emplace()` does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.html> for more on *hinting*.

Insertion requires logarithmic time (if the hint is not taken).

Definition at line 605 of file `stl_map.h`.

```
4.621.3.12 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> bool std::map<_Key, _Tp, _Compare, _Alloc >::empty ( ) const
[inline], [noexcept]
```

Returns true if the map is empty. (Thus `begin()` would equal `end()`.)

Definition at line 464 of file `stl_map.h`.

```
4.621.3.13 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> iterator std::map<_Key, _Tp, _Compare, _Alloc >::end ( )
[inline], [noexcept]
```

Returns a read/write iterator that points one past the last pair in the map. Iteration is done in ascending order according to the keys.

Definition at line 373 of file `stl_map.h`.

Referenced by `std::map<_Key, _Tp, _Compare, _Alloc >::at()`, and `std::map<_Key, _Tp, _Compare, _Alloc >::operator[]()`.

```
4.621.3.14 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> const_iterator std::map<_Key, _Tp, _Compare, _Alloc>::end ( )
const [inline],[noexcept]
```

Returns a read-only (constant) iterator that points one past the last pair in the map. Iteration is done in ascending order according to the keys.

Definition at line 382 of file stl_map.h.

```
4.621.3.15 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> std::pair<iterator, iterator> std::map<_Key, _Tp, _Compare,
_Alloc>::equal_range ( const key_type & __x ) [inline]
```

Finds a subsequence matching given key.

Parameters

<code>__x</code>	Key of (key, value) pairs to be located.
------------------	--

Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multimaps.

Definition at line 955 of file stl_map.h.

```
4.621.3.16 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> std::pair<const_iterator, const_iterator> std::map<_Key, _Tp,
_Compare, _Alloc>::equal_range ( const key_type & __x ) const [inline]
```

Finds a subsequence matching given key.

Parameters

<code>__x</code>	Key of (key, value) pairs to be located.
------------------	--

Returns

Pair of read-only (constant) iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
              c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multimaps.

Definition at line 974 of file stl_map.h.

References std::operator==().

```
4.621.3.17 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> iterator std::map<_Key, _Tp, _Compare, _Alloc>::erase (
const_iterator __position ) [inline]
```

Erases an element from a map.

Parameters

<code>__position</code>	An iterator pointing to the element to be erased.
-------------------------	---

Returns

An iterator pointing to the element immediately following *position* prior to the element being erased. If no such element exists, end() is returned.

This function erases an element, pointed to by the given iterator, from a map. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 725 of file stl_map.h.

Referenced by std::map<_Key, _Tp, _Compare, _Alloc>::erase().

```
4.621.3.18 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> size_type std::map<_Key, _Tp, _Compare, _Alloc>::erase ( const
key_type & __x ) [inline]
```

Erases elements according to the provided key.

Parameters

<code>_↔</code>	Key of element to be erased.
<code>_X</code>	

Returns

The number of elements erased.

This function erases all the elements located by the given key from a map. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 761 of file `stl_map.h`.

```
4.621.3.19 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> iterator std::map< _Key, _Tp, _Compare, _Alloc >::erase (
const_iterator __first, const_iterator __last ) [inline]
```

Erases a `[first,last)` range of elements from a map.

Parameters

<code>__first</code>	Iterator pointing to the start of the range to be erased.
<code>__last</code>	Iterator pointing to the end of the range to be erased.

Returns

The iterator `__last`.

This function erases a sequence of elements from a map. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 781 of file `stl_map.h`.

References `std::map< _Key, _Tp, _Compare, _Alloc >::erase()`.

```
4.621.3.20 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> iterator std::map< _Key, _Tp, _Compare, _Alloc >::find ( const
key_type & __x ) [inline]
```

Tries to locate an element in a map.

Parameters

<code>_↔</code>	Key of (key, value) pair to be located.
<code>_X</code>	

Returns

Iterator pointing to sought-after element, or end() if not found.



This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after pair. If unsuccessful it returns the past-the-end (end()) iterator.

Definition at line 859 of file stl_map.h.

```
4.621.3.21 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> const_iterator std::map<_Key, _Tp, _Compare, _Alloc>::find (
const key_type & __x ) const [inline]
```

Tries to locate an element in a map.

Parameters

	Key of (key, value) pair to be located.
	

Returns

Read-only (constant) iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns a constant iterator pointing to the sought after pair. If unsuccessful it returns the past-the-end (end()) iterator.

Definition at line 874 of file stl_map.h.

```
4.621.3.22 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> allocator_type std::map<_Key, _Tp, _Compare, _Alloc>
>::get_allocator ( ) const [inline], [noexcept]
```



Get a copy of the memory allocation object.

Definition at line 345 of file stl_map.h.

```
4.621.3.23 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> std::pair<iterator, bool> std::map<_Key, _Tp, _Compare, _Alloc>
>::insert ( const value_type & __x ) [inline]
```

Attempts to insert a std::pair into the map.

Parameters

	Pair to be inserted (see std::make_pair for easy creation of pairs).
	

Returns

A pair, of which the first element is an iterator that points to the possibly inserted pair, and the second is a bool that is true if the pair was actually inserted.

This function attempts to insert a (key, value) pair into the map. A map relies on unique keys and thus a pair is only inserted if its first element (the key) is not already present in the map.

Insertion requires logarithmic time.

Definition at line 629 of file stl_map.h.

Referenced by std::map< _Key, _Tp, _Compare, _Alloc >::insert(), std::map< _Key, _Tp, _Compare, _Alloc >::operator=(), and std::map< _Key, _Tp, _Compare, _Alloc >::operator[]().

```
4.621.3.24 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> void std::map< _Key, _Tp, _Compare, _Alloc >::insert (
std::initializer_list< value_type > __list ) [inline]
```

Attempts to insert a list of std::pairs into the map.

Parameters

<code>__list</code>	A std::initializer_list<value_type> of pairs to be inserted.
---------------------	--

Complexity similar to that of the range constructor.

Definition at line 650 of file stl_map.h.

References std::map< _Key, _Tp, _Compare, _Alloc >::insert().

```
4.621.3.25 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> iterator std::map< _Key, _Tp, _Compare, _Alloc >::insert (
const_iterator __position, const value_type & __x ) [inline]
```

Attempts to insert a std::pair into the map.

Parameters

<code>__position</code>	An iterator that serves as a hint as to where the pair should be inserted.
<code>__x</code>	Pair to be inserted (see std::make_pair for easy creation of pairs).

Returns

An iterator that points to the element with key of __x (may or may not be the pair passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument insert() does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.html> for more on *hinting*.

Insertion requires logarithmic time (if the hint is not taken).

Definition at line 679 of file `stl_map.h`.

References `std::map<_Key, _Tp, _Compare, _Alloc>::insert()`.

```
4.621.3.26  template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
            std::allocator<std::pair<const _Key, _Tp>>> template<typename _InputIterator> void std::map<_Key, _Tp,
            _Compare, _Alloc>::insert ( _InputIterator __first, _InputIterator __last ) [inline]
```

Template function that attempts to insert a range of elements.

Parameters

<code>__first</code>	Iterator pointing to the start of the range to be inserted.
<code>__last</code>	Iterator pointing to the end of the range.

Complexity similar to that of the range constructor.

Definition at line 705 of file `stl_map.h`.

```
4.621.3.27  template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
            std::allocator<std::pair<const _Key, _Tp>>> key_compare std::map<_Key, _Tp, _Compare, _Alloc>::key_comp (
            ) const [inline]
```

Returns the key comparison object out of which the map was constructed.

Definition at line 835 of file `stl_map.h`.

Referenced by `std::map<_Key, _Tp, _Compare, _Alloc>::at()`, and `std::map<_Key, _Tp, _Compare, _Alloc>::operator[]()`.

```
4.621.3.28  template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
            std::allocator<std::pair<const _Key, _Tp>>> iterator std::map<_Key, _Tp, _Compare, _Alloc>::lower_bound (
            const key_type & __x ) [inline]
```

Finds the beginning of a subsequence matching given key.

Parameters

<code>__x</code>	Key of (key, value) pair to be located.
------------------	---

Returns

Iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.

Definition at line 901 of file stl_map.h.

Referenced by std::map< _Key, _Tp, _Compare, _Alloc >::at(), and std::map< _Key, _Tp, _Compare, _Alloc >::operator[]().

```
4.621.3.29 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> const_iterator std::map< _Key, _Tp, _Compare, _Alloc
>::lower_bound ( const key_type & __x ) const [inline]
```

Finds the beginning of a subsequence matching given key.

Parameters

<code>__x</code>	Key of (key, value) pair to be located.
------------------	---

Returns

Read-only (constant) iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.

Definition at line 916 of file stl_map.h.

```
4.621.3.30 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> size_type std::map< _Key, _Tp, _Compare, _Alloc >::max_size ( )
const [inline], [noexcept]
```

Returns the maximum size of the map.

Definition at line 474 of file stl_map.h.

```
4.621.3.31 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> map& std::map< _Key, _Tp, _Compare, _Alloc >::operator= ( const
map< _Key, _Tp, _Compare, _Alloc > & __x ) [inline]
```

Map assignment operator.

The dtor only erases the elements, and note that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Parameters

<code>__x</code>	A map of identical element and allocator types.
------------------	---

All the elements of `__x` are copied, but unlike the copy constructor, the allocator object is not copied.

Definition at line 293 of file `stl_map.h`.

```
4.621.3.32 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> map& std::map<_Key, _Tp, _Compare, _Alloc>::operator= (
map<_Key, _Tp, _Compare, _Alloc> && __x ) [inline], [noexcept]
```

Map move assignment operator.

Parameters

<code>__x</code>	A map of identical element and allocator types.
------------------	---

The contents of `__x` are moved into this map (without copying if the allocators compare equal or get moved on assignment). Afterwards `__x` is in a valid, but unspecified state.

Definition at line 309 of file `stl_map.h`.

References `std::map<_Key, _Tp, _Compare, _Alloc>::clear()`, and `std::map<_Key, _Tp, _Compare, _Alloc>::insert()`.

```
4.621.3.33 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> map& std::map<_Key, _Tp, _Compare, _Alloc>::operator= (
initializer_list< value_type > __l ) [inline]
```

Map list assignment operator.

Parameters

<code>__l</code>	An <code>initializer_list</code> .
------------------	------------------------------------

This function fills a map with copies of the elements in the initializer list `__l`.

Note that the assignment completely changes the map and that the resulting map's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 335 of file `stl_map.h`.

References `std::map<_Key, _Tp, _Compare, _Alloc>::clear()`, and `std::map<_Key, _Tp, _Compare, _Alloc>::insert()`.

```
4.621.3.34 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> mapped_type& std::map< _Key, _Tp, _Compare, _Alloc
>::operator[] ( const key_type & __k ) [inline]
```

Subscript ([]) access to map data.

Parameters

\longleftrightarrow _k	The key for which data should be retrieved.
-----------------------------	---

Returns

A reference to the data of the (key,data) pair.

Allows for easy lookup with the subscript ([]) operator. Returns data associated with the key specified in subscript. If the key does not exist, a pair with that key is created using default values, which is then returned.

Lookup requires logarithmic time.

Definition at line 491 of file stl_map.h.

References std::map< _Key, _Tp, _Compare, _Alloc >::end(), std::map< _Key, _Tp, _Compare, _Alloc >::insert(), std::map< _Key, _Tp, _Compare, _Alloc >::key_comp(), std::map< _Key, _Tp, _Compare, _Alloc >::lower_bound(), and std::piecewise_construct.

```
4.621.3.35 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> reverse_iterator std::map< _Key, _Tp, _Compare, _Alloc
>::rbegin ( ) [inline], [noexcept]
```

Returns a read/write reverse iterator that points to the last pair in the map. Iteration is done in descending order according to the keys.

Definition at line 391 of file stl_map.h.

```
4.621.3.36 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> const_reverse_iterator std::map< _Key, _Tp, _Compare, _Alloc
>::rbegin ( ) const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last pair in the map. Iteration is done in descending order according to the keys.

Definition at line 400 of file stl_map.h.

```
4.621.3.37 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> reverse_iterator std::map< _Key, _Tp, _Compare, _Alloc >::rend (
) [inline], [noexcept]
```

Returns a read/write reverse iterator that points to one before the first pair in the map. Iteration is done in descending order according to the keys.

Definition at line 409 of file stl_map.h.

```
4.621.3.38  template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
            std::allocator<std::pair<const _Key, _Tp> >> const_reverse_iterator std::map<_Key, _Tp, _Compare, _Alloc
            >::rend ( ) const    [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first pair in the map. Iteration is done in descending order according to the keys.

Definition at line 418 of file stl_map.h.

```
4.621.3.39  template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
            std::allocator<std::pair<const _Key, _Tp> >> size_type std::map<_Key, _Tp, _Compare, _Alloc >::size ( ) const
            [inline], [noexcept]
```

Returns the size of the map.

Definition at line 469 of file stl_map.h.

```
4.621.3.40  template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
            std::allocator<std::pair<const _Key, _Tp> >> void std::map<_Key, _Tp, _Compare, _Alloc >::swap ( map<
            _Key, _Tp, _Compare, _Alloc > &__x )    [inline], [noexcept]
```

Swaps data with another map.

Parameters

\leftarrow	A map of the same element and allocator types.
$_x$	

This exchanges the elements between two maps in constant time. (It is only swapping a pointer, an integer, and an instance of the `Compare` type (which itself is often stateless and empty), so it should be quite fast.) Note that the global `std::swap()` function is specialized such that `std::swap(m1,m2)` will feed to this function.

Definition at line 813 of file stl_map.h.

Referenced by `std::swap()`.

```
4.621.3.41  template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
            std::allocator<std::pair<const _Key, _Tp> >> iterator std::map<_Key, _Tp, _Compare, _Alloc >::upper_bound (
            const key_type &__x )    [inline]
```

Finds the end of a subsequence matching given key.

Parameters

\leftarrow	Key of (key, value) pair to be located.
$_x$	

Returns

Iterator pointing to the first element greater than key, or end().

Definition at line 926 of file `stl_map.h`.

```
4.621.3.42 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> const_iterator std::map<_Key, _Tp, _Compare, _Alloc>::upper_bound ( const key_type & __x ) const    [inline]
```

Finds the end of a subsequence matching given key.

Parameters

<code>__x</code>	Key of (key, value) pair to be located.
------------------	---

Returns

Read-only (constant) iterator pointing to first iterator greater than key, or end().

Definition at line 936 of file `stl_map.h`.

```
4.621.3.43 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> value_compare std::map<_Key, _Tp, _Compare, _Alloc>::value_comp ( ) const    [inline]
```

Returns a value comparison object, built from the key comparison object out of which the map was constructed.

Definition at line 843 of file `stl_map.h`.

The documentation for this class was generated from the following file:

- [stl_map.h](#)

4.622 `std::mask_array<_Tp>` Class Template Reference

Public Types

- typedef `_Tp` **value_type**

Public Member Functions

- `mask_array` (const `mask_array` &)
- void `operator%=(const valarray<_Tp> &) const`
- template<class `_Dom` >
void `operator%=(const _Expr<_Dom, _Tp> &) const`
- void `operator&=(const valarray<_Tp> &) const`
- template<class `_Dom` >
void `operator&=(const _Expr<_Dom, _Tp> &) const`
- void `operator*=(const valarray<_Tp> &) const`
- template<class `_Dom` >
void `operator*=(const _Expr<_Dom, _Tp> &) const`
- void `operator+=(const valarray<_Tp> &) const`
- template<class `_Dom` >
void `operator+=(const _Expr<_Dom, _Tp> &) const`
- void `operator-= (const valarray<_Tp> &) const`
- template<class `_Dom` >
void `operator-= (const _Expr<_Dom, _Tp> &) const`
- void `operator/= (const valarray<_Tp> &) const`
- template<class `_Dom` >
void `operator/= (const _Expr<_Dom, _Tp> &) const`
- void `operator<=<= (const valarray<_Tp> &) const`
- template<class `_Dom` >
void `operator<=<= (const _Expr<_Dom, _Tp> &) const`
- `mask_array` & `operator=` (const `mask_array` &)
- void `operator=` (const valarray<_Tp> &) const
- void `operator=` (const `_Tp` &) const
- template<class `_Dom` >
void `operator=` (const `_Expr<_Dom, _Tp> &) const`
- template<class `_Ex` >
void `operator=` (const `_Expr<_Ex, _Tp> &__e`) const
- void `operator>>= (const valarray<_Tp> &) const`
- template<class `_Dom` >
void `operator>>= (const _Expr<_Dom, _Tp> &) const`
- void `operator^= (const valarray<_Tp> &) const`
- template<class `_Dom` >
void `operator^= (const _Expr<_Dom, _Tp> &) const`
- void `operator|= (const valarray<_Tp> &) const`
- template<class `_Dom` >
void `operator|= (const _Expr<_Dom, _Tp> &) const`

Friends

- class `valarray<_Tp>`

4.622.1 Detailed Description

```
template<class _Tp>
class std::mask_array<_Tp>
```

Reference to selected subset of an array.

A mask_array is a reference to the actual elements of an array specified by a bitmask in the form of an array of bool. The way to get a mask_array is to call operator[](valarray<bool>) on a valarray. The returned mask_array then permits carrying operations out on the referenced subset of elements in the original valarray.

For example, if a mask_array is obtained using the array (false, true, false, true) as an argument, the mask array has two elements referring to array[1] and array[3] in the underlying array.

Parameters

<i>Tp</i>	Element type.
-----------	---------------

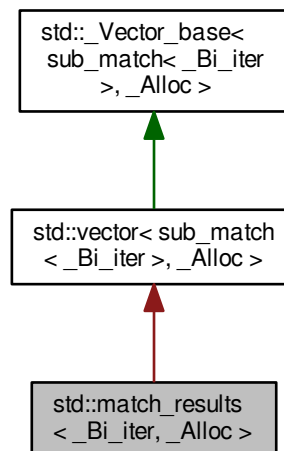
Definition at line 62 of file mask_array.h.

The documentation for this class was generated from the following file:

- [mask_array.h](#)

4.623 std::match_results<_Bi_iter, _Alloc> Class Template Reference

Inheritance diagram for std::match_results<_Bi_iter, _Alloc>:



Public Member Functions

- bool [ready](#) () const

Private Types

- typedef [_Alloc_traits::const_pointer](#) **const_pointer**
- typedef [std::reverse_iterator](#)< const_iterator > **const_reverse_iterator**
- typedef [_Base::pointer](#) **pointer**
- typedef [std::reverse_iterator](#)< iterator > **reverse_iterator**

Private Member Functions

- pointer [_M_allocate](#) (size_t __n)
- pointer [_M_allocate_and_copy](#) (size_type __n, _ForwardIterator __first, _ForwardIterator __last)
- void [_M_assign_aux](#) (_InputIterator __first, _InputIterator __last, [std::input_iterator_tag](#))
- void [_M_assign_aux](#) (_ForwardIterator __first, _ForwardIterator __last, [std::forward_iterator_tag](#))
- void [_M_assign_dispatch](#) (_Integer __n, _Integer __val, __true_type)
- void [_M_assign_dispatch](#) (_InputIterator __first, _InputIterator __last, __false_type)
- size_type [_M_check_len](#) (size_type __n, const char *__s) const
- void [_M_deallocate](#) (pointer __p, size_t __n)
- void [_M_default_append](#) (size_type __n)
- void [_M_default_initialize](#) (size_type __n)
- void [_M_emplace_back_aux](#) (_Args &&...__args)
- iterator [_M_erase](#) (iterator __position)
- iterator [_M_erase](#) (iterator __first, iterator __last)
- void [_M_erase_at_end](#) (pointer __pos) noexcept
- void [_M_fill_assign](#) (size_type __n, const [value_type](#) &__val)
- void [_M_fill_initialize](#) (size_type __n, const [value_type](#) &__value)
- void [_M_fill_insert](#) (iterator __pos, size_type __n, const [value_type](#) &__x)
- _Tp_alloc_type & [_M_get_Tp_allocator](#) () noexcept
- const _Tp_alloc_type & [_M_get_Tp_allocator](#) () const noexcept
- void [_M_initialize_dispatch](#) (_Integer __n, _Integer __value, __true_type)
- void [_M_initialize_dispatch](#) (_InputIterator __first, _InputIterator __last, __false_type)
- void [_M_insert_aux](#) (iterator __position, _Args &&...__args)
- void [_M_insert_dispatch](#) (iterator __pos, _Integer __n, _Integer __val, __true_type)
- void [_M_insert_dispatch](#) (iterator __pos, _InputIterator __first, _InputIterator __last, __false_type)
- void [_M_range_check](#) (size_type __n) const
- void [_M_range_initialize](#) (_InputIterator __first, _InputIterator __last, [std::input_iterator_tag](#))
- void [_M_range_initialize](#) (_ForwardIterator __first, _ForwardIterator __last, [std::forward_iterator_tag](#))
- void [_M_range_insert](#) (iterator __pos, _InputIterator __first, _InputIterator __last, [std::input_iterator_tag](#))
- void [_M_range_insert](#) (iterator __pos, _ForwardIterator __first, _ForwardIterator __last, [std::forward_iterator_tag](#))
- bool [_M_shrink_to_fit](#) ()
- void [assign](#) (size_type __n, const [value_type](#) &__val)
- void [assign](#) (_InputIterator __first, _InputIterator __last)
- void [assign](#) (initializer_list< [value_type](#) > __l)
- [reference](#) at (size_type __n)
- [const_reference](#) at (size_type __n) const

- [reference back](#) () noexcept
- [const_reference back](#) () const noexcept
- iterator [begin](#) () noexcept
- size_type [capacity](#) () const noexcept
- void [clear](#) () noexcept
- [const_reverse_iterator crbegin](#) () const noexcept
- [const_reverse_iterator crend](#) () const noexcept
- [sub_match](#)<_Bi_iter> * [data](#) () noexcept
- const [sub_match](#)<_Bi_iter> * [data](#) () const noexcept
- iterator [emplace](#) (const_iterator __position, _Args &&... __args)
- void [emplace_back](#) (_Args &&... __args)
- iterator [end](#) () noexcept
- iterator [erase](#) (const_iterator __position)
- iterator [erase](#) (const_iterator __first, const_iterator __last)
- [reference front](#) () noexcept
- [const_reference front](#) () const noexcept
- iterator [insert](#) (const_iterator __position, const [value_type](#) &__x)
- iterator [insert](#) (const_iterator __position, [value_type](#) &&__x)
- iterator [insert](#) (const_iterator __position, initializer_list< [value_type](#) > __l)
- iterator [insert](#) (const_iterator __position, size_type __n, const [value_type](#) &__x)
- iterator [insert](#) (const_iterator __position, _InputIterator __first, _InputIterator __last)
- [reference operator\[\]](#) (size_type __n) noexcept
- [const_reference operator\[\]](#) (size_type __n) const noexcept
- void [pop_back](#) () noexcept
- void [push_back](#) (const [value_type](#) &__x)
- void [push_back](#) ([value_type](#) &&__x)
- [reverse_iterator rbegin](#) () noexcept
- [const_reverse_iterator rbegin](#) () const noexcept
- [reverse_iterator rend](#) () noexcept
- [const_reverse_iterator rend](#) () const noexcept
- void [reserve](#) (size_type __n)
- void [resize](#) (size_type __new_size)
- void [resize](#) (size_type __new_size, const [value_type](#) &__x)
- void [shrink_to_fit](#) ()
- void [swap](#) ([vector](#) &__x) noexcept(_Alloc_traits::S_nothrow_swap())

Private Attributes

- [_Vector_impl_M_impl](#)

Friends

- template<typename _Bp, typename _Ap, typename _Cp, typename _Rp, __detail::__RegexExecutorPolicy, bool>
bool [__detail::__regex_algo_impl](#) (_Bp, _Bp, [match_results](#)<_Bp, _Ap> &, const [basic_regex](#)<_Cp, _Rp> &, [regex_constants::match_flag_type](#))
- template<typename, typename, typename, bool>
class [__detail::__Executor](#)
- template<typename, typename, typename>
class [regex_iterator](#)

10.? Public Types

- typedef [sub_match](#)< [_Bi_iter](#) > **value_type**
- typedef const [value_type](#) & **const_reference**
- typedef [const_reference](#) **reference**
- typedef [_Base_type::const_iterator](#) **const_iterator**
- typedef [const_iterator](#) **iterator**
- typedef [__iter_traits::difference_type](#) **difference_type**
- typedef [allocator_traits](#)< [_Alloc](#) >::size_type **size_type**
- typedef [_Alloc](#) **allocator_type**
- typedef [__iter_traits::value_type](#) **char_type**
- typedef [std::basic_string](#)< [char_type](#) > **string_type**

28.10.1 Construction, Copying, and Destruction

- [match_results](#) (const [_Alloc](#) &__a=[_Alloc](#)())
- [match_results](#) (const [match_results](#) &__rhs)=default
- [match_results](#) ([match_results](#) &&__rhs) noexcept=default
- [match_results](#) & [operator=](#) (const [match_results](#) &__rhs)=default
- [match_results](#) & [operator=](#) ([match_results](#) &&__rhs)=default
- [~match_results](#) ()

28.10.2 Size

- size_type [size](#) () const
- size_type [max_size](#) () const
- bool [empty](#) () const

10.3 Element Access

- difference_type [length](#) (size_type __sub=0) const
- difference_type [position](#) (size_type __sub=0) const
- [string_type](#) str (size_type __sub=0) const
- [const_reference](#) [operator\[\]](#) (size_type __sub) const
- [const_reference](#) [prefix](#) () const
- [const_reference](#) [suffix](#) () const
- const_iterator [begin](#) () const
- const_iterator [cbegin](#) () const
- const_iterator [end](#) () const
- const_iterator [cend](#) () const

10.4 Formatting

These functions perform formatted substitution of the matched character sequences into their target. The format specifiers and escape sequences accepted by these functions are determined by their `flags` parameter as documented above.

- `template<typename _Out_iter>`
`_Out_iter format (_Out_iter __out, const char_type * __fmt_first, const char_type * __fmt_last, match_flag_type`
`__flags=regex_constants::format_default) const`
- `template<typename _Out_iter, typename _St, typename _Sa>`
`_Out_iter format (_Out_iter __out, const basic_string< char_type, _St, _Sa > & __fmt, match_flag_type __`
`flags=regex_constants::format_default) const`
- `template<typename _St, typename _Sa>`
`basic_string< char_type, _St, _Sa > format (const basic_string< char_type, _St, _Sa > & __fmt, match_flag_`
`type __flags=regex_constants::format_default) const`
- `string_type format (const char_type * __fmt, match_flag_type __flags=regex_constants::format_default) const`

10.5 Allocator

- `allocator_type get_allocator () const`

10.6 Swap

- `void swap (match_results & __that)`

4.623.1 Detailed Description

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter>>>
class std::match_results< _Bi_iter, _Alloc >
```

The results of a match or search operation.

A collection of character sequences representing the result of a regular expression match. Storage for the collection is allocated and freed as necessary by the member functions of class template `match_results`.

This class satisfies the Sequence requirements, with the exception that only the operations defined for a const-qualified Sequence are supported.

The `sub_match` object stored at index 0 represents sub-expression 0, i.e. the whole match. In this case the `sub_`↵
`match` member `matched` is always true. The `sub_match` object stored at index `n` denotes what matched the marked
sub-expression `n` within the matched expression. If the sub-expression `n` participated in a regular expression match
then the `sub_match` member `matched` evaluates to true, and members `first` and `second` denote the range of characters
[`first`, `second`) which formed that match. Otherwise `matched` is false, and members `first` and `second` point to the end of
the sequence that was searched.

Definition at line 38 of file `regex.h`.

4.623.2 Constructor & Destructor Documentation

4.623.2.1 `template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter>>> std::match_results<_Bi_iter, _Alloc>::match_results (const _Alloc & __a = _Alloc()) [inline], [explicit]`

Constructs a default match_results container.

Postcondition

size() returns 0 and str() returns an empty string.

Definition at line 1592 of file regex.h.

4.623.2.2 `template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter>>> std::match_results<_Bi_iter, _Alloc>::match_results (const match_results<_Bi_iter, _Alloc> & __rhs) [default]`

Copy constructs a match_results.

4.623.2.3 `template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter>>> std::match_results<_Bi_iter, _Alloc>::match_results (match_results<_Bi_iter, _Alloc> && __rhs) [default], [noexcept]`

Move constructs a match_results.

4.623.2.4 `template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter>>> std::match_results<_Bi_iter, _Alloc>::~~match_results () [inline]`

Destroys a match_results object.

Definition at line 1621 of file regex.h.

4.623.3 Member Function Documentation

4.623.3.1 `template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter>>> const_iterator std::match_results<_Bi_iter, _Alloc>::begin () const [inline]`

Gets an iterator to the start of the sub_match collection.

Definition at line 1778 of file regex.h.

Referenced by std::operator==().

4.623.3.2 `template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter>>> const_iterator std::match_results<_Bi_iter, _Alloc>::cbegin () const [inline]`

Gets an iterator to the start of the sub_match collection.

Definition at line 1785 of file regex.h.

4.623.3.3 `template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter>>> const_iterator std::match_results<_Bi_iter, _Alloc>::cend () const [inline]`

Gets an iterator to one-past-the-end of the collection.

Definition at line 1799 of file regex.h.

4.623.3.4 `template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter>>> bool std::match_results<_Bi_iter, _Alloc>::empty () const [inline]`

Indicates if the match_results contains no results.

Return values

<i>true</i>	The match_results object is empty.
<i>false</i>	The match_results object is not empty.

Definition at line 1665 of file regex.h.

Referenced by std::operator==().

4.623.3.5 `template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter>>> const_iterator
std::match_results<_Bi_iter, _Alloc>::end() const [inline]`

Gets an iterator to one-past-the-end of the collection.

Definition at line 1792 of file regex.h.

Referenced by std::operator==().

4.623.3.6 `template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter>>> template<typename _Out_iter >
_Out_iter std::match_results<_Bi_iter, _Alloc>::format(_Out_iter __out, const char_type * __fmt_first, const
char_type * __fmt_last, match_flag_type __flags = regex_constants::format_default) const`

Precondition

ready() == true

4.623.3.7 `template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter>>> template<typename _Out_iter ,
typename _St , typename _Sa > _Out_iter std::match_results<_Bi_iter, _Alloc>::format(_Out_iter __out, const
basic_string<char_type, _St, _Sa> & __fmt, match_flag_type __flags = regex_constants::format_default)
const [inline]`

Precondition

ready() == true

Definition at line 1828 of file regex.h.

4.623.3.8 `template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter>>> template<typename _St ,
typename _Sa > basic_string<char_type, _St, _Sa> std::match_results<_Bi_iter, _Alloc>::format(const
basic_string<char_type, _St, _Sa> & __fmt, match_flag_type __flags = regex_constants::format_default)
const [inline]`

Precondition

ready() == true

Definition at line 1840 of file regex.h.

```
4.623.3.9  template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >> string_type
            std::match_results<_Bi_iter, _Alloc>::format ( const char_type * __fmt, match_flag_type __flags =
            regex_constants::format_default ) const    [inline]
```

Precondition

ready() == true

Definition at line 1852 of file regex.h.

```
4.623.3.10 template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >> allocator_type
            std::match_results<_Bi_iter, _Alloc>::get_allocator ( ) const    [inline]
```

Gets a copy of the allocator.

Definition at line 1874 of file regex.h.

```
4.623.3.11 template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >> difference_type
            std::match_results<_Bi_iter, _Alloc>::length ( size_type __sub = 0 ) const    [inline]
```

Gets the length of the indicated submatch.

Parameters

<code>__sub</code>	indicates the submatch.
--------------------	-------------------------

Precondition

ready() == true

This function returns the length of the indicated submatch, or the length of the entire match if `__sub` is zero (the default).

Definition at line 1684 of file regex.h.

```
4.623.3.12 template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >> size_type
            std::match_results<_Bi_iter, _Alloc>::max_size ( ) const    [inline]
```

Gets the number of matches and submatches.

The number of matches for a given regular expression will be either 0 if there was no match or `mark_count() + 1` if a match was successful. Some matches may be empty.

Returns

the number of matches found.

Definition at line 1656 of file regex.h.


```
4.623.3.13 template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >> match_results&
std::match_results<_Bi_iter, _Alloc >::operator= ( const match_results<_Bi_iter, _Alloc > & __rhs )
[default]
```

Assigns rhs to *this.

```
4.623.3.14 template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >> match_results&
std::match_results<_Bi_iter, _Alloc >::operator= ( match_results<_Bi_iter, _Alloc > && __rhs )
[default]
```

Move-assigns rhs to *this.

```
4.623.3.15 template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >> const_reference
std::match_results<_Bi_iter, _Alloc >::operator[] ( size_type __sub ) const [inline]
```

Gets a sub_match reference for the match or submatch.

Parameters

<code>__sub</code>	indicates the submatch.
--------------------	-------------------------

Precondition

`ready() == true`

This function gets a reference to the indicated submatch, or the entire match if `__sub` is zero.

If `__sub >= size()` then this function returns a sub_match with a special value indicating no submatch.

Definition at line 1732 of file regex.h.

```
4.623.3.16 template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >> difference_type
std::match_results<_Bi_iter, _Alloc >::position ( size_type __sub = 0 ) const [inline]
```

Gets the offset of the beginning of the indicated submatch.

Parameters

<code>__sub</code>	indicates the submatch.
--------------------	-------------------------

Precondition

`ready() == true`

This function returns the offset from the beginning of the target sequence to the beginning of the submatch, unless the value of `__sub` is zero (the default), in which case this function returns the offset from the beginning of the target sequence to the beginning of the match.

Returns -1 if `__sub` is out of range.

Definition at line 1701 of file `regex.h`.

4.623.3.17 `template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter>>> const_reference
std::match_results<_Bi_iter, _Alloc>::prefix () const [inline]`

Gets a `sub_match` representing the match prefix.

Precondition

`ready() == true`

This function gets a reference to a `sub_match` object representing the part of the target range between the start of the target range and the start of the match.

Definition at line 1749 of file `regex.h`.

Referenced by `std::operator==()`.

4.623.3.18 `template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter>>> bool std::match_results<
_Bi_iter, _Alloc>::ready () const [inline]`

Indicates if the `match_results` is ready.

Return values

<i>true</i>	The object has a fully-established result state.
<i>false</i>	The object is not ready.

Definition at line 1632 of file `regex.h`.

Referenced by `std::operator==()`.

4.623.3.19 `template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter>>> size_type
std::match_results<_Bi_iter, _Alloc>::size () const [inline]`

Gets the number of matches and submatches.

The number of matches for a given regular expression will be either 0 if there was no match or `mark_count() + 1` if a match was successful. Some matches may be empty.

Returns

the number of matches found.

Definition at line 1649 of file `regex.h`.

Referenced by `std::operator==()`.

```
4.623.3.20  template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter>>> string_type  
            std::match_results<_Bi_iter, _Alloc>::str ( size_type __sub = 0 ) const  [inline]
```

Gets the match or submatch converted to a string type.

Parameters

<code>__sub</code>	indicates the submatch.
--------------------	-------------------------

Precondition

`ready() == true`

This function gets the submatch (or match, if `__sub` is zero) extracted from the target range and converted to the associated string type.

Definition at line 1717 of file `regex.h`.

```
4.623.3.21  template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >> const_reference
            std::match_results<_Bi_iter, _Alloc>::suffix ( ) const    [inline]
```

Gets a `sub_match` representing the match suffix.

Precondition

`ready() == true`

This function gets a reference to a `sub_match` object representing the part of the target range between the end of the match and the end of the target range.

Definition at line 1766 of file `regex.h`.

Referenced by `std::operator==()`.

```
4.623.3.22  template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >> void std::match_results<
            _Bi_iter, _Alloc>::swap ( match_results<_Bi_iter, _Alloc> &__that )    [inline]
```

Swaps the contents of two `match_results`.

Definition at line 1888 of file `regex.h`.

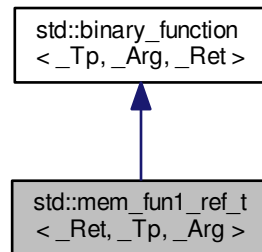
Referenced by `std::swap()`.

The documentation for this class was generated from the following file:

- [regex.h](#)

4.624 std::mem_fun1_ref_t< _Ret, _Tp, _Arg > Class Template Reference

Inheritance diagram for std::mem_fun1_ref_t< _Ret, _Tp, _Arg >:



Public Types

- typedef _Tp [first_argument_type](#)
- typedef _Ret [result_type](#)
- typedef _Arg [second_argument_type](#)

Public Member Functions

- **mem_fun1_ref_t** (_Ret(_Tp::*__pf)(_Arg))
- _Ret **operator()** (_Tp &__r, _Arg __x) const

4.624.1 Detailed Description

```
template<typename _Ret, typename _Tp, typename _Arg>
class std::mem_fun1_ref_t< _Ret, _Tp, _Arg >
```

One of the [adaptors for member pointers](#).

Definition at line 1003 of file stl_function.h.

4.624.2 Member Typedef Documentation

4.624.2.1 typedef _Tp std::binary_function< _Tp, _Arg, _Ret >::first_argument_type [\[inherited\]](#)

first_argument_type is the type of the first argument

Definition at line 121 of file stl_function.h.

4.624.2.2 `typedef _Ret std::binary_function<_Tp, _Arg, _Ret>::result_type` [inherited]

`result_type` is the return type

Definition at line 127 of file `stl_function.h`.

4.624.2.3 `typedef _Arg std::binary_function<_Tp, _Arg, _Ret>::second_argument_type` [inherited]

`second_argument_type` is the type of the second argument

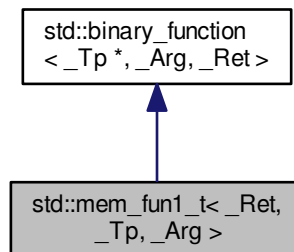
Definition at line 124 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [stl_function.h](#)

4.625 `std::mem_fun1_t<_Ret, _Tp, _Arg>` Class Template Reference

Inheritance diagram for `std::mem_fun1_t<_Ret, _Tp, _Arg>`:



Public Types

- `typedef _Tp *` [first_argument_type](#)
- `typedef _Ret` [result_type](#)
- `typedef _Arg` [second_argument_type](#)

Public Member Functions

- `mem_fun1_t` (`_Ret` (`_Tp::*` `__pf`) (`_Arg`))
- `_Ret operator()` (`_Tp *` `__p`, `_Arg` `__x`) `const`

4.625.1 Detailed Description

```
template<typename _Ret, typename _Tp, typename _Arg>
class std::mem_fun1_t<_Ret, _Tp, _Arg>
```

One of the [adaptors for member pointers](#).

Definition at line 967 of file stl_function.h.

4.625.2 Member Typedef Documentation

4.625.2.1 `typedef _Tp * std::binary_function<_Tp *, _Arg, _Ret>::first_argument_type` [inherited]

`first_argument_type` is the type of the first argument

Definition at line 121 of file stl_function.h.

4.625.2.2 `typedef _Ret std::binary_function<_Tp *, _Arg, _Ret>::result_type` [inherited]

`result_type` is the return type

Definition at line 127 of file stl_function.h.

4.625.2.3 `typedef _Arg std::binary_function<_Tp *, _Arg, _Ret>::second_argument_type` [inherited]

`second_argument_type` is the type of the second argument

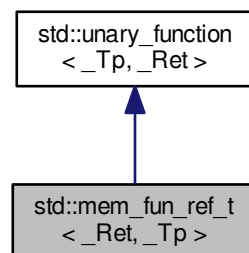
Definition at line 124 of file stl_function.h.

The documentation for this class was generated from the following file:

- [stl_function.h](#)

4.626 std::mem_fun_ref_t<_Ret, _Tp> Class Template Reference

Inheritance diagram for `std::mem_fun_ref_t<_Ret, _Tp>`:



Public Types

- typedef `_Tp` [argument_type](#)
- typedef `_Ret` [result_type](#)

Public Member Functions

- **mem_fun_ref_t** (`_Ret` (`_Tp::*`__pf)())
- `_Ret` **operator()** (`_Tp` &__r) const

4.626.1 Detailed Description

```
template<typename _Ret, typename _Tp>  
class std::mem_fun_ref_t< _Ret, _Tp >
```

One of the [adaptors for member pointers](#).

Definition at line 931 of file `stl_function.h`.

4.626.2 Member Typedef Documentation

4.626.2.1 `typedef _Tp std::unary_function< _Tp, _Ret >::argument_type` `[inherited]`

`argument_type` is the type of the argument

Definition at line 108 of file `stl_function.h`.

4.626.2.2 `typedef _Ret std::unary_function< _Tp, _Ret >::result_type` `[inherited]`

`result_type` is the return type

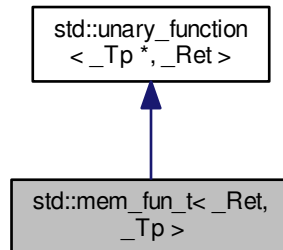
Definition at line 111 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [stl_function.h](#)

4.627 `std::mem_fun_t<_Ret,_Tp>` Class Template Reference

Inheritance diagram for `std::mem_fun_t<_Ret,_Tp>`:



Public Types

- typedef `_Tp *` [argument_type](#)
- typedef `_Ret` [result_type](#)

Public Member Functions

- `mem_fun_t(_Ret(_Tp::*__pf)())`
- `_Ret operator()(_Tp *__p) const`

4.627.1 Detailed Description

```
template<typename _Ret, typename _Tp>
class std::mem_fun_t<_Ret,_Tp>
```

One of the [adaptors for member pointers](#).

Definition at line 895 of file `stl_function.h`.

4.627.2 Member Typedef Documentation

4.627.2.1 `typedef _Tp * std::unary_function<_Tp*,_Ret>::argument_type` [inherited]

`argument_type` is the type of the argument

Definition at line 108 of file `stl_function.h`.

4.627.2.2 `typedef _Ret std::unary_function<_Tp*, _Ret>::result_type` [inherited]

`result_type` is the return type

Definition at line 111 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [stl_function.h](#)

4.628 `std::mersenne_twister_engine<_UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f>`
> Class Template Reference

Public Types

- `typedef _UIntType` [result_type](#)

Public Member Functions

- `mersenne_twister_engine` ([result_type](#) __sd=default_seed)
- `template<typename _Sseq, typename = typename std::enable_if<!std::is_same<_Sseq, mersenne_twister_engine>::value>::type>`
`mersenne_twister_engine` (_Sseq &__q)
- `void` [discard](#) (unsigned long long __z)
- [result_type](#) `operator()` ()
- `void` **seed** ([result_type](#) __sd=default_seed)
- `template<typename _Sseq>`
`std::enable_if< std::is_class< _Sseq >::value >::type` **seed** (_Sseq &__q)

Static Public Member Functions

- `static constexpr` [result_type](#) `max` ()
- `static constexpr` [result_type](#) `min` ()

Static Public Attributes

- `static constexpr` [result_type](#) `default_seed`
- `static constexpr` [result_type](#) `initialization_multiplier`
- `static constexpr` `size_t` `mask_bits`
- `static constexpr` `size_t` `shift_size`
- `static constexpr` `size_t` `state_size`
- `static constexpr` [result_type](#) `tempering_b`
- `static constexpr` [result_type](#) `tempering_c`
- `static constexpr` [result_type](#) `tempering_d`
- `static constexpr` `size_t` `tempering_l`
- `static constexpr` `size_t` `tempering_s`
- `static constexpr` `size_t` `tempering_t`
- `static constexpr` `size_t` `tempering_u`
- `static constexpr` `size_t` `word_size`
- `static constexpr` [result_type](#) `xor_mask`

Friends

- `template<typename _UIntType1, size_t __w1, size_t __n1, size_t __m1, size_t __r1, _UIntType1 __a1, size_t __u1, _UIntType1 __d1, size_t __s1, _UIntType1 __b1, size_t __t1, _UIntType1 __c1, size_t __l1, _UIntType1 __f1, typename _CharT, typename _Traits > std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::mersenne_twister_engine< _UIntType1, __w1, __n1, __m1, __r1, __a1, __u1, __d1, __s1, __b1, __t1, __c1, __l1, __f1 > &__x)`
- `bool operator== (const mersenne_twister_engine &__lhs, const mersenne_twister_engine &__rhs)`
- `template<typename _UIntType1, size_t __w1, size_t __n1, size_t __m1, size_t __r1, _UIntType1 __a1, size_t __u1, _UIntType1 __d1, size_t __s1, _UIntType1 __b1, size_t __t1, _UIntType1 __c1, size_t __l1, _UIntType1 __f1, typename _CharT, typename _Traits > std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::mersenne_twister_engine< _UIntType1, __w1, __n1, __m1, __r1, __a1, __u1, __d1, __s1, __b1, __t1, __c1, __l1, __f1 > &__x)`

4.628.1 Detailed Description

```
template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s,
        _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f>
class std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >
```

A generalized feedback shift register discrete random number generator.

This algorithm avoids multiplication and division and is designed to be friendly to a pipelined architecture. If the parameters are chosen correctly, this generator will produce numbers with a very long period and fairly good apparent entropy, although still not cryptographically strong.

The best way to use this generator is with the predefined `mt19937` class.

This algorithm was originally invented by Makoto Matsumoto and Takuji Nishimura.

Template Parameters

<code>__w</code>	Word size, the number of bits in each element of the state vector.
<code>__n</code>	The degree of recursion.
<code>__m</code>	The period parameter.
<code>__r</code>	The separation point bit index.
<code>__a</code>	The last row of the twist matrix.
<code>__u</code>	The first right-shift tempering matrix parameter.
<code>__d</code>	The first right-shift tempering matrix mask.
<code>__s</code>	The first left-shift tempering matrix parameter.
<code>__b</code>	The first left-shift tempering matrix mask.

Template Parameters

\leftarrow _t	The second left-shift tempering matrix parameter.
\leftarrow _c	The second left-shift tempering matrix mask.
\leftarrow _l	The second right-shift tempering matrix parameter.
\leftarrow _f	Initialization multiplier.

Definition at line 451 of file random.h.

4.628.2 Member Typedef Documentation

4.628.2.1 `template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f> typedef _UIntType std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >::result_type`

The type of the generated random value.

Definition at line 454 of file random.h.

4.628.3 Constructor & Destructor Documentation

4.628.3.1 `template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f> template<typename _Sseq, typename = typename std::enable_if<!std::is_same< _Sseq, mersenne_twister_engine>::value> ::type> std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >::mersenne_twister_engine (_Sseq & __q) [inline], [explicit]`

Constructs a mersenne_twister_engine random number generator engine seeded from the seed sequence __q.

Parameters

\leftarrow _q	the seed sequence.
--------------------	--------------------

Definition at line 515 of file random.h.

4.628.4 Member Function Documentation

4.628.4.1 `template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f> void std::mersenne_twister_engine<_UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f>::discard (unsigned long long __z)`

Discard a sequence of random numbers.

4.628.4.2 `template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f> static constexpr result_type std::mersenne_twister_engine<_UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f>::max () [inline], [static]`

Gets the largest possible value in the output range.

Definition at line 536 of file random.h.

4.628.4.3 `template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f> static constexpr result_type std::mersenne_twister_engine<_UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f>::min () [inline], [static]`

Gets the smallest possible value in the output range.

Definition at line 529 of file random.h.

4.628.5 Friends And Related Function Documentation

4.628.5.1 `template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f> template<typename _UIntType1 , size_t __w1, size_t __n1, size_t __m1, size_t __r1, _UIntType1 __a1, size_t __u1, _UIntType1 __d1, size_t __s1, _UIntType1 __b1, size_t __t1, _UIntType1 __c1, size_t __l1, _UIntType1 __f1, typename _CharT , typename _Traits > std::basic_ostream<_CharT, _Traits>& operator<< (std::basic_ostream<_CharT, _Traits> & __os, const std::mersenne_twister_engine<_UIntType1, __w1, __n1, __m1, __r1, __a1, __u1, __d1, __s1, __b1, __t1, __c1, __l1, __f1> & __x) [friend]`

Inserts the current state of a % mersenne_twister_engine random number generator engine __x into the output stream __os.

Parameters

<code>__os</code>	An output stream.
<code>__x</code>	A % mersenne_twister_engine random number generator engine.

Returns

The output stream with the state of __x inserted or in an error state.

```

4.628.5.2 template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType
    __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f> bool operator== ( const
    mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > & __lhs, const
    mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > & __rhs )
    [friend]

```

Compares two % mersenne_twister_engine random number generator objects of the same type for equality.

Parameters

<code>__lhs</code>	A % mersenne_twister_engine random number generator object.
<code>__rhs</code>	Another % mersenne_twister_engine random number generator object.

Returns

true if the infinite sequences of generated values would be equal, false otherwise.

Definition at line 561 of file random.h.

```

4.628.5.3 template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d,
    size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f> template<typename _UIntType1 ,
    size_t __w1, size_t __n1, size_t __m1, size_t __r1, _UIntType1 __a1, size_t __u1, _UIntType1 __d1, size_t __s1,
    _UIntType1 __b1, size_t __t1, _UIntType1 __c1, size_t __l1, _UIntType1 __f1, typename _CharT , typename
    _Traits > std::basic_istream< _CharT, _Traits>& operator>> ( std::basic_istream< _CharT, _Traits > & __is,
    std::mersenne_twister_engine< _UIntType1, __w1, __n1, __m1, __r1, __a1, __u1, __d1, __s1, __b1, __t1, __c1, __l1,
    __f1 > & __x ) [friend]

```

Extracts the current state of a % mersenne_twister_engine random number generator engine `__x` from the input stream `__is`.

Parameters

<code>__is</code>	An input stream.
<code>__x</code>	A % mersenne_twister_engine random number generator engine.

Returns

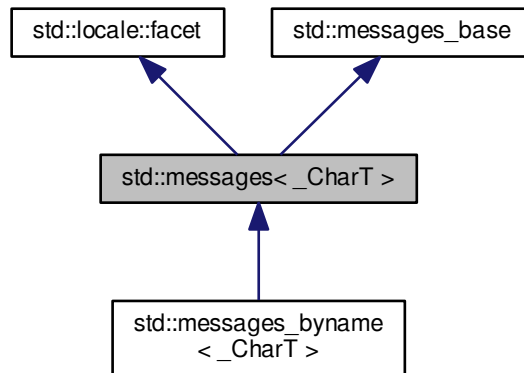
The input stream with the state of `__x` extracted or in an error state.

The documentation for this class was generated from the following file:

- [random.h](#)

4.629 `std::messages<_CharT>` Class Template Reference

Inheritance diagram for `std::messages<_CharT>`:



Public Types

- typedef int **catalog**
- typedef `_CharT` **char_type**
- typedef `basic_string<_CharT>` **string_type**

Public Member Functions

- **messages** (size_t __refs=0)
- **messages** (__c_locale __cloc, const char *__s, size_t __refs=0)
- void **close** (catalog __c) const
- **string_type** **get** (catalog __c, int __set, int __msgid, const **string_type** &__s) const
- catalog **open** (const **basic_string**< char > &__s, const **locale** &__loc) const
- catalog **open** (const **basic_string**< char > &, const **locale** &, const char *) const

Static Public Attributes

- static **locale::id** **id**

Protected Member Functions

- virtual `~messages()`
- `string_type _M_convert_from_char (char *) const`
- `char * _M_convert_to_char (const string_type &__msg) const`
- virtual void `do_close (catalog) const`
- virtual `string_type do_get (catalog, int, int, const string_type &__default) const`
- `template<> string do_get (catalog, int, int, const string &) const`
- `template<> wstring do_get (catalog, int, int, const wstring &) const`
- virtual catalog `do_open (const basic_string<char> &, const locale &) const`

Static Protected Member Functions

- static `__c_locale _S_clone_c_locale (__c_locale &__cloc) throw ()`
- static void `_S_create_c_locale (__c_locale &__cloc, const char *__s, __c_locale __old=0)`
- static void `_S_destroy_c_locale (__c_locale &__cloc)`
- static `__c_locale _S_get_c_locale ()`
- static const char * `_S_get_c_name () throw ()`
- static `__c_locale _S_lc_ctype_c_locale (__c_locale __cloc, const char *__s)`

Protected Attributes

- `__c_locale _M_c_locale_messages`
- `const char * _M_name_messages`

4.629.1 Detailed Description

```
template<typename _CharT>
class std::messages<_CharT>
```

Primary class template `messages`.

This facet encapsulates the code to retrieve messages from message catalogs. The only thing defined by the standard for this facet is the interface. All underlying functionality is implementation-defined.

This library currently implements 3 versions of the message facet. The first version (gnu) is a wrapper around `gettext`, provided by `libintl`. The second version (ieee) is a wrapper around `catgets`. The final version (default) does no actual translation. These implementations are only provided for `char` and `wchar_t` instantiations.

The `messages` template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the `messages` facet.

Definition at line 1695 of file `locale_facets_nonio.h`.

4.629.2 Member Typedef Documentation

4.629.2.1 template<typename _CharT> typedef _CharT std::messages<_CharT>::char_type

Public typedefs.

Definition at line 1701 of file locale_facets_nonio.h.

4.629.2.2 template<typename _CharT> typedef basic_string<_CharT> std::messages<_CharT>::string_type

Public typedefs.

Definition at line 1702 of file locale_facets_nonio.h.

4.629.3 Constructor & Destructor Documentation

4.629.3.1 template<typename _CharT> std::messages<_CharT>::messages (size_t __refs = 0) [explicit]

Constructor performs initialization.

This is the constructor provided by the standard.

Parameters

<code>__refs</code>	Passed to the base facet class.
---------------------	---------------------------------

Definition at line 44 of file messages_members.h.

4.629.3.2 template<typename _CharT> std::messages<_CharT>::messages (__c_locale __cloc, const char * __s, size_t __refs = 0) [explicit]

Internal constructor. Not for general use.

This is a constructor for use by the library itself to set up new locales.

Parameters

<code>__cloc</code>	The C locale.
<code>__s</code>	The name of a locale.
<code>__refs</code>	Refcount to pass to the base class.

Definition at line 50 of file messages_members.h.

References std::basic_string<_CharT, _Traits, _Alloc>::c_str().

4.629.3.3 `template<typename _CharT> std::messages<_CharT>::~~messages ()` [protected],[virtual]

Destructor.

Definition at line 79 of file `messages_members.h`.

References `std::basic_string<_CharT, _Traits, _Alloc>::c_str()`.

4.629.4 Member Function Documentation

4.629.4.1 `template<> string std::messages<char>::do_get (catalog , int , int , const string &) const`
[protected]

Specializations for required instantiations.

4.629.5 Member Data Documentation

4.629.5.1 `template<typename _CharT> locale::id std::messages<_CharT>::id` [static]

Numpunct facet id.

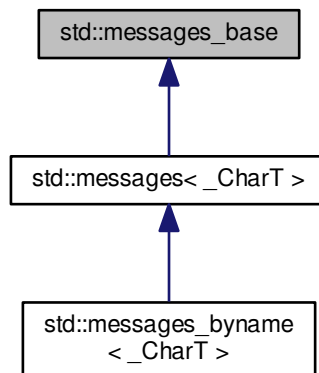
Definition at line 1713 of file `locale_facets_nonio.h`.

The documentation for this class was generated from the following files:

- [locale_facets_nonio.h](#)
- [messages_members.h](#)

4.630 std::messages_base Struct Reference

Inheritance diagram for `std::messages_base`:



Public Types

- typedef int **catalog**

4.630.1 Detailed Description

Messages facet base class providing catalog typedef.

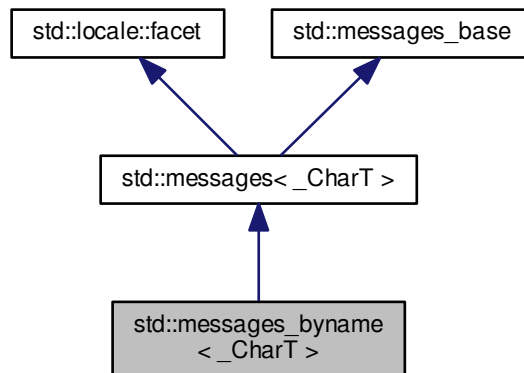
Definition at line 1668 of file locale_facets_nonio.h.

The documentation for this struct was generated from the following file:

- [locale_facets_nonio.h](#)

4.631 std::messages_byname<_CharT> Class Template Reference

Inheritance diagram for std::messages_byname<_CharT>:



Public Types

- typedef int **catalog**
- typedef `_CharT` **char_type**
- typedef [basic_string](#)<`_CharT`> **string_type**

Public Member Functions

- **messages_byname** (const char *__s, size_t __refs=0)
- void **close** (catalog __c) const
- **string_type get** (catalog __c, int __set, int __msgid, const **string_type** &__s) const
- catalog **open** (const **basic_string**< char > &__s, const **locale** &__loc) const
- catalog **open** (const **basic_string**< char > &, const **locale** &, const char *) const

Static Public Attributes

- static **locale::id** id

Protected Member Functions

- **string_type M_convert_from_char** (char *) const
- char * **M_convert_to_char** (const **string_type** &__msg) const
- virtual void **do_close** (catalog) const
- virtual **string_type do_get** (catalog, int, int, const **string_type** &__default) const
- template<>
string do_get (catalog, int, int, const **string** &) const
- template<>
wstring do_get (catalog, int, int, const **wstring** &) const
- virtual catalog **do_open** (const **basic_string**< char > &, const **locale** &) const

Static Protected Member Functions

- static __c_locale **S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **S_get_c_locale** ()
- static const char * **S_get_c_name** () throw ()
- static __c_locale **S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

Protected Attributes

- __c_locale **M_c_locale_messages**
- const char * **M_name_messages**

4.631.1 Detailed Description

```
template<typename _CharT>
class std::messages_byname< _CharT >
```

class messages_byname [22.2.7.2].

Definition at line 1879 of file locale_facets_nonio.h.

4.631.2 Member Function Documentation

4.631.2.1 `template<> string std::messages< char >::do_get (catalog , int , int , const string &) const`
[protected], [inherited]

Specializations for required instantiations.

4.631.3 Member Data Documentation

4.631.3.1 `template<typename _CharT> locale::id std::messages<_CharT>::id` [static], [inherited]

Numpunct facet id.

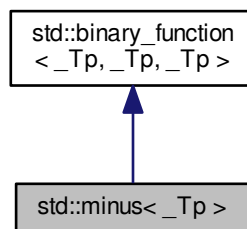
Definition at line 1713 of file locale_facets_nonio.h.

The documentation for this class was generated from the following files:

- [locale_facets_nonio.h](#)
- [messages_members.h](#)

4.632 std::minus<_Tp> Struct Template Reference

Inheritance diagram for std::minus<_Tp>:



Public Types

- typedef `_Tp` [first_argument_type](#)
- typedef `_Tp` [result_type](#)
- typedef `_Tp` [second_argument_type](#)

Public Member Functions

- `_Tp operator() (const _Tp &__x, const _Tp &__y) const`

4.632.1 Detailed Description

```
template<typename _Tp>
struct std::minus< _Tp >
```

One of the [math functors](#).

Definition at line 176 of file `stl_function.h`.

4.632.2 Member Typedef Documentation

4.632.2.1 `typedef _Tp std::binary_function< _Tp, _Tp, _Tp >::first_argument_type` `[inherited]`

`first_argument_type` is the type of the first argument

Definition at line 121 of file `stl_function.h`.

4.632.2.2 `typedef _Tp std::binary_function< _Tp, _Tp, _Tp >::result_type` `[inherited]`

`result_type` is the return type

Definition at line 127 of file `stl_function.h`.

4.632.2.3 `typedef _Tp std::binary_function< _Tp, _Tp, _Tp >::second_argument_type` `[inherited]`

`second_argument_type` is the type of the second argument

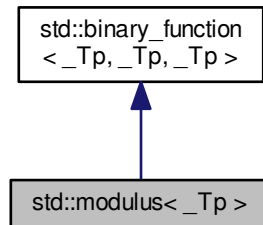
Definition at line 124 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl_function.h](#)

4.633 std::modulus<_Tp> Struct Template Reference

Inheritance diagram for std::modulus<_Tp>:



Public Types

- typedef _Tp [first_argument_type](#)
- typedef _Tp [result_type](#)
- typedef _Tp [second_argument_type](#)

Public Member Functions

- **_Tp operator()** (const _Tp &__x, const _Tp &__y) const

4.633.1 Detailed Description

```
template<typename _Tp>
struct std::modulus<_Tp>
```

One of the [math functors](#).

Definition at line 203 of file stl_function.h.

4.633.2 Member Typedef Documentation

4.633.2.1 typedef _Tp std::binary_function<_Tp, _Tp, _Tp>::first_argument_type [inherited]

first_argument_type is the type of the first argument

Definition at line 121 of file stl_function.h.

4.633.2.2 `typedef _Tp std::binary_function<_Tp, _Tp, _Tp>::result_type` [inherited]

`result_type` is the return type

Definition at line 127 of file `stl_function.h`.

4.633.2.3 `typedef _Tp std::binary_function<_Tp, _Tp, _Tp>::second_argument_type` [inherited]

`second_argument_type` is the type of the second argument

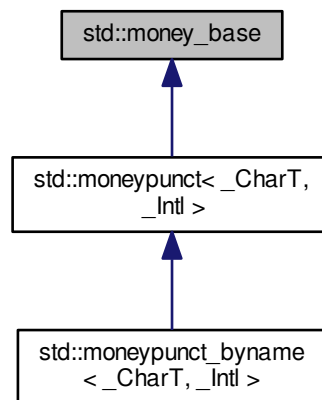
Definition at line 124 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl_function.h](#)

4.634 `std::money_base` Class Reference

Inheritance diagram for `std::money_base`:



Public Types

- enum { **_S_minus**, **_S_zero**, **_S_end** }
- enum **part** {
 none, **space**, **symbol**, **sign**,
 value }

Static Public Member Functions

- static pattern **_S_construct_pattern** (char __precedes, char __space, char __posn) throw ()

Static Public Attributes

- static const char * **_S_atoms**
- static const pattern **_S_default_pattern**

4.634.1 Detailed Description

Money format ordering data.

This class contains an ordered array of 4 fields to represent the pattern for formatting a money amount. Each field may contain one entry from the part enum. symbol, sign, and value must be present and the remaining field must contain either none or space.

See also

money_punct::pos_format() and money_punct::neg_format() for details of how these fields are interpreted.

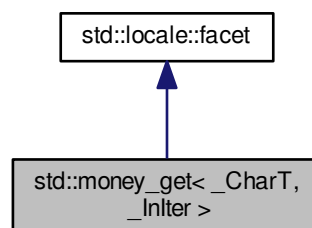
Definition at line 840 of file locale_facets_nonio.h.

The documentation for this class was generated from the following file:

- [locale_facets_nonio.h](#)

4.635 std::money_get< _CharT, _InIter > Class Template Reference

Inheritance diagram for std::money_get< _CharT, _InIter >:



Public Types

- typedef `_CharT` `char_type`
- typedef `_InIter` `iter_type`
- typedef `basic_string<_CharT>` `string_type`

Public Member Functions

- `money_get` (`size_t __refs=0`)
- `iter_type get` (`iter_type __s`, `iter_type __end`, `bool __intl`, `ios_base &__io`, `ios_base::iostate &__err`, `long double &__units`) `const`
- `iter_type get` (`iter_type __s`, `iter_type __end`, `bool __intl`, `ios_base &__io`, `ios_base::iostate &__err`, `string_type &__digits`) `const`

Static Public Attributes

- static `locale::id` `id`

Protected Member Functions

- virtual `~money_get` ()
- `template<bool _Intl>`
`iter_type _M_extract` (`iter_type __s`, `iter_type __end`, `ios_base &__io`, `ios_base::iostate &__err`, `string &__digits`) `const`
- virtual `iter_type do_get` (`iter_type __s`, `iter_type __end`, `bool __intl`, `ios_base &__io`, `ios_base::iostate &__err`, `long double &__units`) `const`
- virtual `iter_type do_get` (`iter_type __s`, `iter_type __end`, `bool __intl`, `ios_base &__io`, `ios_base::iostate &__err`, `string_type &__digits`) `const`

Static Protected Member Functions

- static `__c_locale _S_clone_c_locale` (`__c_locale &__cloc`) `throw ()`
- static void `_S_create_c_locale` (`__c_locale &__cloc`, `const char *__s`, `__c_locale __old=0`)
- static void `_S_destroy_c_locale` (`__c_locale &__cloc`)
- static `__c_locale _S_get_c_locale` ()
- static `const char * _S_get_c_name` () `throw ()`
- static `__c_locale _S_lc_ctype_c_locale` (`__c_locale __cloc`, `const char *__s`)

4.635.1 Detailed Description

```
template<typename _CharT, typename _InIter>
class std::money_get<_CharT, _InIter>
```

Primary class template `money_get`.

This facet encapsulates the code to parse and return a monetary amount from a string.

The `money_get` template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the `money_get` facet.

Definition at line 1370 of file `locale_facets_nonio.h`.

4.635.2 Member Typedef Documentation

4.635.2.1 `template<typename _CharT, typename _InIter> typedef _CharT std::money_get< _CharT, _InIter >::char_type`

Public typedefs.

Definition at line 1376 of file locale_facets_nonio.h.

4.635.2.2 `template<typename _CharT, typename _InIter> typedef _InIter std::money_get< _CharT, _InIter >::iter_type`

Public typedefs.

Definition at line 1377 of file locale_facets_nonio.h.

4.635.2.3 `template<typename _CharT, typename _InIter> typedef basic_string<_CharT> std::money_get< _CharT, _InIter >::string_type`

Public typedefs.

Definition at line 1378 of file locale_facets_nonio.h.

4.635.3 Constructor & Destructor Documentation

4.635.3.1 `template<typename _CharT, typename _InIter> std::money_get< _CharT, _InIter >::money_get (size_t __refs = 0) [inline], [explicit]`

Constructor performs initialization.

This is the constructor provided by the standard.

Parameters

<code>__refs</code>	Passed to the base facet class.
---------------------	---------------------------------

Definition at line 1392 of file locale_facets_nonio.h.

4.635.3.2 `template<typename _CharT, typename _InIter> virtual std::money_get< _CharT, _InIter >::~~money_get () [inline], [protected], [virtual]`

Destructor.

Definition at line 1460 of file locale_facets_nonio.h.

4.635.4 Member Function Documentation

4.635.4.1 `template<typename _CharT, typename _InIter> virtual iter_type std::money_get< _CharT, _InIter >::do_get (iter_type __s, iter_type __end, bool __intl, ios_base & __io, ios_base::iostate & __err, long double & __units) const [protected], [virtual]`

Read and parse a monetary value.

This function reads and parses characters representing a monetary value. This function is a hook for derived classes to change the value returned.

See also

`get()` for details.

4.635.4.2 `template<typename _CharT, typename _InIter> virtual iter_type std::money_get< _CharT, _InIter >::do_get (iter_type __s, iter_type __end, bool __intl, ios_base & __io, ios_base::iostate & __err, string_type & __digits) const [protected], [virtual]`

Read and parse a monetary value.

This function reads and parses characters representing a monetary value. This function is a hook for derived classes to change the value returned.

See also

`get()` for details.

4.635.4.3 `template<typename _CharT, typename _InIter> iter_type std::money_get< _CharT, _InIter >::get (iter_type __s, iter_type __end, bool __intl, ios_base & __io, ios_base::iostate & __err, long double & __units) const [inline]`

Read and parse a monetary value.

This function reads characters from `__s`, interprets them as a monetary value according to `moneypunct` and `ctype` facets retrieved from `io.getloc()`, and returns the result in `units` as an integral value `moneypunct::frac_digits() * the actual amount`. For example, the string \$10.01 in a US locale would store 1001 in `units`.

Any characters not part of a valid money amount are not consumed.

If a money value cannot be parsed from the input stream, sets `err=(err|io.failbit)`. If the stream is consumed before finishing parsing, sets `err=(err|io.failbit|io.eofbit)`. `units` is unchanged if parsing fails.

This function works by returning the result of `do_get()`.

Parameters

<code>__s</code>	Start of characters to parse.
<code>__end</code>	End of characters to parse.
<code>__intl</code>	Parameter to use <code>_facet<moneypunct<CharT,intl> ></code> .
<code>__io</code>	Source of facets and io state.
<code>__err</code>	Error field to set if parsing fails.
<code>__units</code>	Place to store result of parsing.

Returns

Iterator referencing first character beyond valid money amount.

Definition at line 1422 of file locale_facets_nonio.h.

```
4.635.4.4 template<typename _CharT, typename _InIter> iter_type std::money_get< _CharT, _InIter >::get ( iter_type
    __s, iter_type __end, bool __intl, ios_base & __io, ios_base::iostate & __err, string_type & __digits ) const
    [inline]
```

Read and parse a monetary value.

This function reads characters from `__s`, interprets them as a monetary value according to `money_punct` and `ctype` facets retrieved from `io.getloc()`, and returns the result in `digits`. For example, the string `$10.01` in a US locale would store `1001` in `digits`.

Any characters not part of a valid money amount are not consumed.

If a money value cannot be parsed from the input stream, sets `err=(err|io.failbit)`. If the stream is consumed before finishing parsing, sets `err=(err|io.failbit|io.eofbit)`.

This function works by returning the result of `do_get()`.

Parameters

<code>__s</code>	Start of characters to parse.
<code>__end</code>	End of characters to parse.
<code>__intl</code>	Parameter to use <code>_facet<money_punct<CharT,intl>></code> .
<code>__io</code>	Source of facets and io state.
<code>__err</code>	Error field to set if parsing fails.
<code>__digits</code>	Place to store result of parsing.

Returns

Iterator referencing first character beyond valid money amount.

Definition at line 1453 of file locale_facets_nonio.h.

4.635.5 Member Data Documentation

```
4.635.5.1 template<typename _CharT, typename _InIter> locale::id std::money_get< _CharT, _InIter >::id [static]
```

Numpunct facet id.

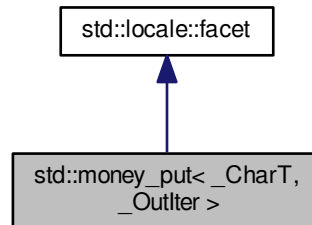
Definition at line 1382 of file locale_facets_nonio.h.

The documentation for this class was generated from the following file:

- [locale_facets_nonio.h](#)

4.636 `std::money_put<_CharT, _OutIter >` Class Template Reference

Inheritance diagram for `std::money_put<_CharT, _OutIter >`:



Public Types

- typedef `_CharT` `char_type`
- typedef `_OutIter` `iter_type`
- typedef `basic_string<_CharT >` `string_type`

Public Member Functions

- `money_put` (`size_t` `__refs=0`)
- `iter_type put` (`iter_type` `__s`, `bool` `__intl`, `ios_base &__io`, `char_type` `__fill`, `long double` `__units`) `const`
- `iter_type put` (`iter_type` `__s`, `bool` `__intl`, `ios_base &__io`, `char_type` `__fill`, `const string_type &__digits`) `const`

Static Public Attributes

- static `locale::id` `id`

Protected Member Functions

- virtual `~money_put` ()
- template<`bool` `_Intl`>
`iter_type _M_insert` (`iter_type` `__s`, `ios_base &__io`, `char_type` `__fill`, `const string_type &__digits`) `const`
- virtual `iter_type do_put` (`iter_type` `__s`, `bool` `__intl`, `ios_base &__io`, `char_type` `__fill`, `long double` `__units`) `const`
- virtual `iter_type do_put` (`iter_type` `__s`, `bool` `__intl`, `ios_base &__io`, `char_type` `__fill`, `const string_type &__digits`) `const`

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_type_c_locale** (__c_locale __cloc, const char *__s)

4.636.1 Detailed Description

```
template<typename _CharT, typename _Outlter>
class std::money_put<_CharT, _Outlter>
```

Primary class template money_put.

This facet encapsulates the code to format and output a monetary amount.

The money_put template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the money_put facet.

Definition at line 1521 of file locale_facets_nonio.h.

4.636.2 Member Typedef Documentation

4.636.2.1 `template<typename _CharT, typename _Outlter> typedef _CharT std::money_put<_CharT, _Outlter>::char_type`

Public typedefs.

Definition at line 1526 of file locale_facets_nonio.h.

4.636.2.2 `template<typename _CharT, typename _Outlter> typedef _Outlter std::money_put<_CharT, _Outlter>::iter_type`

Public typedefs.

Definition at line 1527 of file locale_facets_nonio.h.

4.636.2.3 `template<typename _CharT, typename _Outlter> typedef basic_string<_CharT> std::money_put<_CharT, _Outlter>::string_type`

Public typedefs.

Definition at line 1528 of file locale_facets_nonio.h.

4.636.3 Constructor & Destructor Documentation

4.636.3.1 `template<typename _CharT, typename _Outlter> std::money_put<_CharT, _Outlter>::money_put (size_t __refs = 0) [inline], [explicit]`

Constructor performs initialization.

This is the constructor provided by the standard.

Parameters

<code>__refs</code>	Passed to the base facet class.
---------------------	---------------------------------

Definition at line 1542 of file `locale_facets_nonio.h`.

4.636.3.2 `template<typename _CharT, typename _OutIter> virtual std::money_put<_CharT, _OutIter>::~~money_put ()`
`[inline], [protected], [virtual]`

Destructor.

Definition at line 1592 of file `locale_facets_nonio.h`.

4.636.4 Member Function Documentation

4.636.4.1 `template<typename _CharT, typename _OutIter> virtual iter_type std::money_put<_CharT, _OutIter>::do_put (`
`iter_type __s, bool __intl, ios_base & __io, char_type __fill, long double __units) const` `[protected],`
`[virtual]`

Format and output a monetary value.

This function formats *units* as a monetary value according to `moneypunct` and `ctype` facets retrieved from `io.getloc()`, and writes the resulting characters to `__s`. For example, the value 1001 in a US locale would write \$10.01 to `__s`.

This function is a hook for derived classes to change the value returned.

See also

`put()`.

Parameters

<code>__s</code>	The stream to write to.
<code>__intl</code>	Parameter to use <code>_facet<moneypunct<CharT,intl>></code> .
<code>__io</code>	Source of facets and io state.
<code>__fill</code>	<code>char_type</code> to use for padding.
<code>__units</code>	Place to store result of parsing.

Returns

Iterator after writing.

4.636.4.2 `template<typename _CharT, typename _OutIter> virtual iter_type std::money_put<_CharT, _OutIter>::do_put`
`(iter_type __s, bool __intl, ios_base & __io, char_type __fill, const string_type & __digits) const`
`[protected], [virtual]`

Format and output a monetary value.

This function formats *digits* as a monetary value according to moneypunct and ctype facets retrieved from io.getloc(), and writes the resulting characters to `__s`. For example, the string `1001` in a US locale would write `$10.01` to `__s`.

This function is a hook for derived classes to change the value returned.

See also

`put()`.

Parameters

<code>__s</code>	The stream to write to.
<code>__intl</code>	Parameter to use <code>_facet<moneypunct<CharT,intl>></code> .
<code>__io</code>	Source of facets and io state.
<code>__fill</code>	<code>char_type</code> to use for padding.
<code>__digits</code>	Place to store result of parsing.

Returns

Iterator after writing.

4.636.4.3 `template<typename _CharT, typename _OutIter> iter_type std::money_put<_CharT, _OutIter>::put (iter_type __s, bool __intl, ios_base & __io, char_type __fill, long double __units) const [inline]`

Format and output a monetary value.

This function formats *units* as a monetary value according to moneypunct and ctype facets retrieved from io.getloc(), and writes the resulting characters to `__s`. For example, the value `1001` in a US locale would write `$10.01` to `__s`.

This function works by returning the result of `do_put()`.

Parameters

<code>__s</code>	The stream to write to.
<code>__intl</code>	Parameter to use <code>_facet<moneypunct<CharT,intl>></code> .
<code>__io</code>	Source of facets and io state.
<code>__fill</code>	<code>char_type</code> to use for padding.
<code>__units</code>	Place to store result of parsing.

Returns

Iterator after writing.

Definition at line 1562 of file `locale_facets_nonio.h`.

4.636.4.4 `template<typename _CharT, typename _Outiter> iter_type std::money_put<_CharT, _Outiter>::put (iter_type __s, bool __intl, ios_base & __io, char_type __fill, const string_type & __digits) const [inline]`

Format and output a monetary value.

This function formats *digits* as a monetary value according to `moneypunct` and `ctype` facets retrieved from `io.getloc()`, and writes the resulting characters to `__s`. For example, the string `1001` in a US locale would write `$10.01` to `__s`.

This function works by returning the result of `do_put()`.

Parameters

<code>__s</code>	The stream to write to.
<code>__intl</code>	Parameter to use <code>_facet<moneypunct<CharT,intl>></code> .
<code>__io</code>	Source of facets and io state.
<code>__fill</code>	<code>char_type</code> to use for padding.
<code>__digits</code>	Place to store result of parsing.

Returns

Iterator after writing.

Definition at line 1585 of file `locale_facets_nonio.h`.

4.636.5 Member Data Documentation

4.636.5.1 `template<typename _CharT, typename _Outiter> locale::id std::money_put<_CharT, _Outiter>::id [static]`

Numpunct facet id.

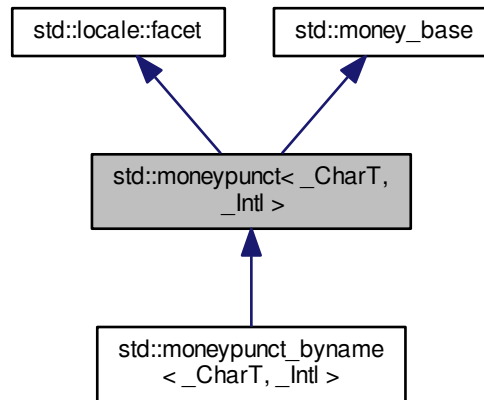
Definition at line 1532 of file `locale_facets_nonio.h`.

The documentation for this class was generated from the following file:

- [locale_facets_nonio.h](#)

4.637 std::moneypunct< _CharT, _Intl > Class Template Reference

Inheritance diagram for std::moneypunct< _CharT, _Intl >:



Public Types

- enum { **`_S_minus`**, **`_S_zero`**, **`_S_end`** }
- typedef `__moneypunct_cache< _CharT, _Intl > __cache_type`
- enum **`part`** {
 `none`, **`space`**, **`symbol`**, **`sign`**,
 `value` }
- typedef `_CharT` `char_type`
- typedef `basic_string< _CharT >` `string_type`

Public Member Functions

- `moneypunct` (`size_t __refs=0`)
- `moneypunct` (`__cache_type * __cache, size_t __refs=0`)
- `moneypunct` (`__c_locale __cloc, const char * __s, size_t __refs=0`)
- `string_type curr_symbol` () const
- `char_type decimal_point` () const
- `int frac_digits` () const
- `string grouping` () const
- `string_type negative_sign` () const
- `string_type positive_sign` () const
- `char_type thousands_sep` () const
- pattern `pos_format` () const
- pattern `neg_format` () const

Static Public Member Functions

- static pattern **_S_construct_pattern** (char __precedes, char __space, char __posn) throw ()

Static Public Attributes

- static const char * **_S_atoms**
- static const pattern **_S_default_pattern**
- static [locale::id](#) **id**
- static const bool **intl**

Protected Member Functions

- virtual [~moneypunct](#) ()
- void **_M_initialize_moneypunct** (__c_locale __cloc=0, const char *__name=0)
- template<>
void **_M_initialize_moneypunct** (__c_locale, const char *)
- template<>
void **_M_initialize_moneypunct** (__c_locale, const char *)
- template<>
void **_M_initialize_moneypunct** (__c_locale, const char *)
- template<>
void **_M_initialize_moneypunct** (__c_locale, const char *)
- virtual [string_type](#) **do_curr_symbol** () const
- virtual [char_type](#) **do_decimal_point** () const
- virtual int **do_frac_digits** () const
- virtual [string](#) **do_grouping** () const
- virtual pattern **do_neg_format** () const
- virtual [string_type](#) **do_negative_sign** () const
- virtual pattern **do_pos_format** () const
- virtual [string_type](#) **do_positive_sign** () const
- virtual [char_type](#) **do_thousands_sep** () const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

4.637.1 Detailed Description

```
template<typename _CharT, bool _Intl>
class std::moneypunct< _CharT, _Intl >
```

Primary class template moneypunct.

This facet encapsulates the punctuation, grouping and other formatting features of money amount string representations.

Definition at line 934 of file `locale_facets_nonio.h`.

4.637.2 Member Typedef Documentation

4.637.2.1 template<typename _CharT, bool _Intl> typedef _CharT std::moneypunct<_CharT, _Intl>::char_type

Public typedefs.

Definition at line 940 of file locale_facets_nonio.h.

4.637.2.2 template<typename _CharT, bool _Intl> typedef basic_string<_CharT> std::moneypunct<_CharT, _Intl>::string_type

Public typedefs.

Definition at line 941 of file locale_facets_nonio.h.

4.637.3 Constructor & Destructor Documentation

4.637.3.1 template<typename _CharT, bool _Intl> std::moneypunct<_CharT, _Intl>::moneypunct (size_t __refs = 0)
[inline], [explicit]

Constructor performs initialization.

This is the constructor provided by the standard.

Parameters

<code>__refs</code>	Passed to the base facet class.
---------------------	---------------------------------

Definition at line 963 of file locale_facets_nonio.h.

4.637.3.2 template<typename _CharT, bool _Intl> std::moneypunct<_CharT, _Intl>::moneypunct (__cache_type * __cache, size_t __refs = 0) [inline], [explicit]

Constructor performs initialization.

This is an internal constructor.

Parameters

<code>__cache</code>	Cache for optimization.
<code>__refs</code>	Passed to the base facet class.

Definition at line 976 of file locale_facets_nonio.h.

4.637.3.3 `template<typename _CharT, bool _Intl> std::moneypunct< _CharT, _Intl >::moneypunct (__c_locale __cloc, const char * __s, size_t __refs = 0) [inline], [explicit]`

Internal constructor. Not for general use.

This is a constructor for use by the library itself to set up new locales.

Parameters

<code>__cloc</code>	The C locale.
<code>__s</code>	The name of a locale.
<code>__refs</code>	Passed to the base facet class.

Definition at line 991 of file `locale_facets_nonio.h`.

4.637.3.4 `template<typename _CharT, bool _Intl> virtual std::moneypunct< _CharT, _Intl >::~~moneypunct () [protected], [virtual]`

Destructor.

Referenced by `std::moneypunct< _CharT, _Intl >::do_neg_format()`.

4.637.4 Member Function Documentation

4.637.4.1 `template<typename _CharT, bool _Intl> string_type std::moneypunct< _CharT, _Intl >::curr_symbol () const [inline]`

Return currency symbol string.

This function returns a `string_type` to use as a currency symbol. It does so by returning `moneypunct<char_↵_type>::do_curr_symbol()`.

Returns

string_type representing a currency symbol.

Definition at line 1061 of file `locale_facets_nonio.h`.

4.637.4.2 `template<typename _CharT, bool _Intl> char_type std::moneypunct< _CharT, _Intl >::decimal_point () const [inline]`

Return decimal point character.

This function returns a `char_type` to use as a decimal point. It does so by returning `moneypunct<char_↵type>::do_decimal_point()`.

Returns

char_type representing a decimal point.

Definition at line 1005 of file `locale_facets_nonio.h`.

4.637.4.3 `template<typename _CharT, bool _Intl> virtual string_type std::moneypunct<_CharT, _Intl>::do_curr_symbol () const` `[inline], [protected], [virtual]`

Return currency symbol string.

This function returns a `string_type` to use as a currency symbol. This function is a hook for derived classes to change the value returned.

See also

`curr_symbol()` for details.

Returns

string_type representing a currency symbol.

Definition at line 1207 of file `locale_facets_nonio.h`.

4.637.4.4 `template<typename _CharT, bool _Intl> virtual char_type std::moneypunct<_CharT, _Intl>::do_decimal_point () const` `[inline], [protected], [virtual]`

Return decimal point character.

Returns a `char_type` to use as a decimal point. This function is a hook for derived classes to change the value returned.

Returns

char_type representing a decimal point.

Definition at line 1169 of file `locale_facets_nonio.h`.

4.637.4.5 `template<typename _CharT, bool _Intl> virtual int std::moneypunct<_CharT, _Intl>::do_frac_digits () const` `[inline], [protected], [virtual]`

Return number of digits in fraction.

This function returns the exact number of digits that make up the fractional part of a money amount. This function is a hook for derived classes to change the value returned.

See also

`frac_digits()` for details.

Returns

Number of digits in amount fraction.

Definition at line 1247 of file `locale_facets_nonio.h`.

4.637.4.6 `template<typename _CharT, bool _Intl> virtual string std::moneypunct<_CharT, _Intl>::do_grouping () const`
`[inline], [protected], [virtual]`

Return grouping specification.

Returns a string representing groupings for the integer part of a number. This function is a hook for derived classes to change the value returned.

See also

`grouping()` for details.

Returns

String representing grouping specification.

Definition at line 1194 of file `locale_facets_nonio.h`.

4.637.4.7 `template<typename _CharT, bool _Intl> virtual pattern std::moneypunct<_CharT, _Intl>::do_neg_format () const`
`[inline], [protected], [virtual]`

Return pattern for money values.

This function returns a pattern describing the formatting of a negative valued money amount. This function is a hook for derived classes to change the value returned.

See also

`neg_format()` for details.

Returns

Pattern for money values.

Definition at line 1275 of file `locale_facets_nonio.h`.

References `std::moneypunct<_CharT, _Intl>::~~moneypunct()`.

4.637.4.8 `template<typename _CharT, bool _Intl> virtual string_type std::moneypunct<_CharT, _Intl>::do_negative_sign () const`
`[inline], [protected], [virtual]`

Return negative sign string.

This function returns a `string_type` to use as a sign for negative amounts. This function is a hook for derived classes to change the value returned.

See also

`negative_sign()` for details.

Returns

string_type representing a negative sign.

Definition at line 1233 of file `locale_facets_nonio.h`.

4.637.4.9 `template<typename _CharT, bool _Intl> virtual pattern std::moneypunct<_CharT, _Intl>::do_pos_format () const`
`[inline], [protected], [virtual]`

Return pattern for money values.

This function returns a pattern describing the formatting of a positive valued money amount. This function is a hook for derived classes to change the value returned.

See also

`pos_format()` for details.

Returns

Pattern for money values.

Definition at line 1261 of file `locale_facets_nonio.h`.

4.637.4.10 `template<typename _CharT, bool _Intl> virtual string_type std::moneypunct<_CharT, _Intl>::do_positive_sign`
`() const [inline], [protected], [virtual]`

Return positive sign string.

This function returns a `string_type` to use as a sign for positive amounts. This function is a hook for derived classes to change the value returned.

See also

`positive_sign()` for details.

Returns

string_type representing a positive sign.

Definition at line 1220 of file `locale_facets_nonio.h`.

4.637.4.11 `template<typename _CharT, bool _Intl> virtual char_type std::moneypunct<_CharT, _Intl>::do_thousands_sep`
`() const [inline], [protected], [virtual]`

Return thousands separator character.

Returns a `char_type` to use as a thousands separator. This function is a hook for derived classes to change the value returned.

Returns

char_type representing a thousands separator.

Definition at line 1181 of file `locale_facets_nonio.h`.

4.637.4.12 `template<typename _CharT, bool _Intl> int std::moneypunct<_CharT, _Intl>::frac_digits () const`
`[inline]`

Return number of digits in fraction.

This function returns the exact number of digits that make up the fractional part of a money amount. It does so by returning `returning moneypunct<char_type>::do_frac_digits()`.

The fractional part of a money amount is optional. But if it is present, there must be `frac_digits()` digits.

Returns

Number of digits in amount fraction.

Definition at line 1111 of file `locale_facets_nonio.h`.

4.637.4.13 `template<typename _CharT, bool _Intl> string std::moneypunct<_CharT, _Intl>::grouping () const`
`[inline]`

Return grouping specification.

This function returns a string representing groupings for the integer part of an amount. Groupings indicate where thousands separators should be inserted.

Each char in the return string is interpreted as an integer rather than a character. These numbers represent the number of digits in a group. The first char in the string represents the number of digits in the least significant group. If a char is negative, it indicates an unlimited number of digits for the group. If more chars from the string are required to group a number, the last char is used repeatedly.

For example, if the `grouping()` returns `\003\002` and is applied to the number 123456789, this corresponds to 12,34,56,789. Note that if the string was `32`, this would put more than 50 digits into the least significant group if the character set is ASCII.

The string is returned by calling `moneypunct<char_type>::do_grouping()`.

Returns

string representing grouping specification.

Definition at line 1048 of file `locale_facets_nonio.h`.

4.637.4.14 `template<typename _CharT, bool _Intl> pattern std::moneypunct<_CharT, _Intl>::neg_format () const`
`[inline]`

Return pattern for money values.

This function returns a pattern describing the formatting of a positive or negative valued money amount. It does so by returning `std::moneypunct<char_type>::do_pos_format()` or `std::moneypunct<char_type>::do_neg_format()`.

The pattern has 4 fields describing the ordering of symbol, sign, value, and none or space. There must be one of each in the pattern. The none and space enums may not appear in the first field and space may not appear in the final field.

The parts of a money string must appear in the order indicated by the fields of the pattern. The symbol field indicates that the value of `curr_symbol()` may be present. The sign field indicates that the value of `positive_sign()` or `negative_sign()` must be present. The value field indicates that the absolute value of the money amount is present. none indicates 0 or more whitespace characters, except at the end, where it permits no whitespace. space indicates that 1 or more whitespace characters must be present.

For example, for the US locale and `pos_format()` pattern {symbol,sign,value,none}, `curr_symbol() == '$'` `positive_sign() == '+'`, and value 10.01, and options set to force the symbol, the corresponding string is `$+10.01`.

Returns

Pattern for money values.

Definition at line 1151 of file `locale_facets_nonio.h`.

4.637.4.15 `template<typename _CharT, bool _Intl> string_type std::moneypunct<_CharT, _Intl>::negative_sign () const`
`[inline]`

Return negative sign string.

This function returns a `string_type` to use as a sign for negative amounts. It does so by returning `std::moneypunct<char_type>::do_negative_sign()`.

If the return value contains more than one character, the first character appears in the position indicated by `neg_format()` and the remainder appear at the end of the formatted string.

Returns

string_type representing a negative sign.

Definition at line 1095 of file `locale_facets_nonio.h`.

4.637.4.16 `template<typename _CharT, bool _Intl> pattern std::moneypunct<_CharT, _Intl>::pos_format () const`
`[inline]`

Return pattern for money values.

This function returns a pattern describing the formatting of a positive or negative valued money amount. It does so by returning `moneypunct<char_type>::do_pos_format()` or `moneypunct<char_type>::do_neg_format()`.

The pattern has 4 fields describing the ordering of symbol, sign, value, and none or space. There must be one of each in the pattern. The none and space enums may not appear in the first field and space may not appear in the final field.

The parts of a money string must appear in the order indicated by the fields of the pattern. The symbol field indicates that the value of `curr_symbol()` may be present. The sign field indicates that the value of `positive_sign()` or `negative_sign()` must be present. The value field indicates that the absolute value of the money amount is present. none indicates 0 or more whitespace characters, except at the end, where it permits no whitespace. space indicates that 1 or more whitespace characters must be present.

For example, for the US locale and `pos_format()` pattern {symbol,sign,value,none}, `curr_symbol() == '$'` `positive_sign() == '+'`, and value 10.01, and options set to force the symbol, the corresponding string is \$+10.01.

Returns

Pattern for money values.

Definition at line 1147 of file `locale_facets_nonio.h`.

4.637.4.17 `template<typename _CharT, bool _Intl> string_type std::moneypunct<_CharT, _Intl>::positive_sign () const`
`[inline]`

Return positive sign string.

This function returns a `string_type` to use as a sign for positive amounts. It does so by returning `moneypunct<char_type>::do_positive_sign()`.

If the return value contains more than one character, the first character appears in the position indicated by `pos_format()` and the remainder appear at the end of the formatted string.

Returns

string_type representing a positive sign.

Definition at line 1078 of file `locale_facets_nonio.h`.

4.637.4.18 `template<typename _CharT, bool _Intl> char_type std::moneypunct<_CharT, _Intl>::thousands_sep () const`
`[inline]`

Return thousands separator character.

This function returns a `char_type` to use as a thousands separator. It does so by returning `moneypunct<char_type>::do_thousands_sep()`.

Returns

`char_type` representing a thousands separator.

Definition at line 1018 of file `locale_facets_nonio.h`.

4.637.5 Member Data Documentation

4.637.5.1 `template<typename _CharT, bool _Intl> locale::id std::moneypunct<_CharT, _Intl>::id` `[static]`

Numpunct facet id.

Definition at line 953 of file locale_facets_nonio.h.

4.637.5.2 `template<typename _CharT, bool _Intl> const bool std::moneypunct<_CharT, _Intl>::intl` `[static]`

This value is provided by the standard, but no reason for its existence.

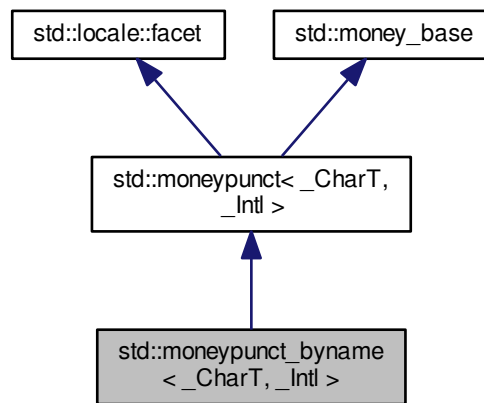
Definition at line 951 of file locale_facets_nonio.h.

The documentation for this class was generated from the following file:

- [locale_facets_nonio.h](#)

4.638 std::moneypunct_byname<_CharT, _Intl> Class Template Reference

Inheritance diagram for std::moneypunct_byname<_CharT, _Intl>:



Public Types

- `enum { _S_minus, _S_zero, _S_end }`
- `typedef __moneypunct_cache<_CharT, _Intl> __cache_type`
- `typedef _CharT char_type`
- `enum part { none, space, symbol, sign, value }`
- `typedef basic_string<_CharT> string_type`

Public Member Functions

- **money_punct_byname** (const char *__s, size_t __refs=0)
- [string_type curr_symbol](#) () const
- [char_type decimal_point](#) () const
- [int frac_digits](#) () const
- [string grouping](#) () const
- [string_type negative_sign](#) () const
- [string_type positive_sign](#) () const
- [char_type thousands_sep](#) () const
- [pattern pos_format](#) () const
- [pattern neg_format](#) () const

Static Public Member Functions

- static pattern **_S_construct_pattern** (char __precedes, char __space, char __posn) throw ()

Static Public Attributes

- static const char * **_S_atoms**
- static const pattern **_S_default_pattern**
- static [locale::id](#) **id**
- static const bool **intl**

Protected Member Functions

- void **_M_initialize_money_punct** (__c_locale __cloc=0, const char *__name=0)
- `template<>`
void **_M_initialize_money_punct** (__c_locale, const char *)
- `template<>`
void **_M_initialize_money_punct** (__c_locale, const char *)
- `template<>`
void **_M_initialize_money_punct** (__c_locale, const char *)
- `template<>`
void **_M_initialize_money_punct** (__c_locale, const char *)
- virtual [string_type do_curr_symbol](#) () const
- virtual [char_type do_decimal_point](#) () const
- virtual [int do_frac_digits](#) () const
- virtual [string do_grouping](#) () const
- virtual [pattern do_neg_format](#) () const
- virtual [string_type do_negative_sign](#) () const
- virtual [pattern do_pos_format](#) () const
- virtual [string_type do_positive_sign](#) () const
- virtual [char_type do_thousands_sep](#) () const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

4.638.1 Detailed Description

```
template<typename _CharT, bool _Intl>
class std::moneypunct_byname<_CharT, _Intl>
```

class moneypunct_byname [22.2.6.4].

Definition at line 1324 of file locale_facets_nonio.h.

4.638.2 Member Function Documentation

4.638.2.1 `template<typename _CharT, bool _Intl> string_type std::moneypunct<_CharT, _Intl>::curr_symbol () const`
`[inline], [inherited]`

Return currency symbol string.

This function returns a `string_type` to use as a currency symbol. It does so by returning `moneypunct<char↵_type>::do_curr_symbol()`.

Returns

string_type representing a currency symbol.

Definition at line 1061 of file locale_facets_nonio.h.

4.638.2.2 `template<typename _CharT, bool _Intl> char_type std::moneypunct<_CharT, _Intl>::decimal_point () const`
`[inline], [inherited]`

Return decimal point character.

This function returns a `char_type` to use as a decimal point. It does so by returning `moneypunct<char↵_type>::do_decimal_point()`.

Returns

char_type representing a decimal point.

Definition at line 1005 of file locale_facets_nonio.h.

```
4.638.2.3  template<typename _CharT, bool _Intl> virtual string_type std::moneypunct<_CharT, _Intl>::do_curr_symbol (
    ) const    [inline], [protected], [virtual], [inherited]
```

Return currency symbol string.

This function returns a `string_type` to use as a currency symbol. This function is a hook for derived classes to change the value returned.

See also

`curr_symbol()` for details.

Returns

string_type representing a currency symbol.

Definition at line 1207 of file `locale_facets_nonio.h`.

```
4.638.2.4  template<typename _CharT, bool _Intl> virtual char_type std::moneypunct<_CharT, _Intl>::do_decimal_point (
    ) const    [inline], [protected], [virtual], [inherited]
```

Return decimal point character.

Returns a `char_type` to use as a decimal point. This function is a hook for derived classes to change the value returned.

Returns

char_type representing a decimal point.

Definition at line 1169 of file `locale_facets_nonio.h`.

```
4.638.2.5  template<typename _CharT, bool _Intl> virtual int std::moneypunct<_CharT, _Intl>::do_frac_digits ( ) const
    [inline], [protected], [virtual], [inherited]
```

Return number of digits in fraction.

This function returns the exact number of digits that make up the fractional part of a money amount. This function is a hook for derived classes to change the value returned.

See also

`frac_digits()` for details.

Returns

Number of digits in amount fraction.

Definition at line 1247 of file `locale_facets_nonio.h`.

4.638.2.6 `template<typename _CharT, bool _Intl> virtual string std::moneypunct<_CharT, _Intl>::do_grouping () const`
`[inline], [protected], [virtual], [inherited]`

Return grouping specification.

Returns a string representing groupings for the integer part of a number. This function is a hook for derived classes to change the value returned.

See also

grouping() for details.

Returns

String representing grouping specification.

Definition at line 1194 of file locale_facets_nonio.h.

4.638.2.7 `template<typename _CharT, bool _Intl> virtual pattern std::moneypunct<_CharT, _Intl>::do_neg_format () const`
`[inline], [protected], [virtual], [inherited]`

Return pattern for money values.

This function returns a pattern describing the formatting of a negative valued money amount. This function is a hook for derived classes to change the value returned.

See also

neg_format() for details.

Returns

Pattern for money values.

Definition at line 1275 of file locale_facets_nonio.h.

References std::moneypunct<_CharT, _Intl>::~~moneypunct().

4.638.2.8 `template<typename _CharT, bool _Intl> virtual string_type std::moneypunct<_CharT, _Intl>::do_negative_sign () const`
`[inline], [protected], [virtual], [inherited]`

Return negative sign string.

This function returns a string_type to use as a sign for negative amounts. This function is a hook for derived classes to change the value returned.

See also

negative_sign() for details.

Returns

string_type representing a negative sign.

Definition at line 1233 of file locale_facets_nonio.h.

4.638.2.9 `template<typename _CharT, bool _Intl> virtual pattern std::moneypunct<_CharT, _Intl>::do_pos_format () const`
`[inline], [protected], [virtual], [inherited]`

Return pattern for money values.

This function returns a pattern describing the formatting of a positive valued money amount. This function is a hook for derived classes to change the value returned.

See also

`pos_format()` for details.

Returns

Pattern for money values.

Definition at line 1261 of file `locale_facets_nonio.h`.

4.638.2.10 `template<typename _CharT, bool _Intl> virtual string_type std::moneypunct<_CharT, _Intl>::do_positive_sign`
`() const [inline], [protected], [virtual], [inherited]`

Return positive sign string.

This function returns a `string_type` to use as a sign for positive amounts. This function is a hook for derived classes to change the value returned.

See also

`positive_sign()` for details.

Returns

string_type representing a positive sign.

Definition at line 1220 of file `locale_facets_nonio.h`.

4.638.2.11 `template<typename _CharT, bool _Intl> virtual char_type std::moneypunct<_CharT, _Intl>::do_thousands_sep`
`() const [inline], [protected], [virtual], [inherited]`

Return thousands separator character.

Returns a `char_type` to use as a thousands separator. This function is a hook for derived classes to change the value returned.

Returns

char_type representing a thousands separator.

Definition at line 1181 of file `locale_facets_nonio.h`.

```
4.638.2.12 template<typename _CharT, bool _Intl> int std::moneypunct<_CharT, _Intl >::frac_digits ( ) const  
[inline], [inherited]
```

Return number of digits in fraction.

This function returns the exact number of digits that make up the fractional part of a money amount. It does so by returning returning moneypunct<char_type>::do_frac_digits().

The fractional part of a money amount is optional. But if it is present, there must be frac_digits() digits.

Returns

Number of digits in amount fraction.

Definition at line 1111 of file locale_facets_nonio.h.

```
4.638.2.13 template<typename _CharT, bool _Intl> string std::moneypunct<_CharT, _Intl >::grouping ( ) const  
[inline], [inherited]
```

Return grouping specification.

This function returns a string representing groupings for the integer part of an amount. Groupings indicate where thousands separators should be inserted.

Each char in the return string is interpret as an integer rather than a character. These numbers represent the number of digits in a group. The first char in the string represents the number of digits in the least significant group. If a char is negative, it indicates an unlimited number of digits for the group. If more chars from the string are required to group a number, the last char is used repeatedly.

For example, if the grouping() returns \003\002 and is applied to the number 123456789, this corresponds to 12,34,56,789. Note that if the string was 32, this would put more than 50 digits into the least significant group if the character set is ASCII.

The string is returned by calling moneypunct<char_type>::do_grouping().

Returns

string representing grouping specification.

Definition at line 1048 of file locale_facets_nonio.h.

4.638.2.14 `template<typename _CharT, bool _Intl> pattern std::moneypunct<_CharT, _Intl>::neg_format () const`
`[inline], [inherited]`

Return pattern for money values.

This function returns a pattern describing the formatting of a positive or negative valued money amount. It does so by returning `moneypunct<char_type>::do_pos_format()` or `moneypunct<char_type>::do_neg_format()`.

The pattern has 4 fields describing the ordering of symbol, sign, value, and none or space. There must be one of each in the pattern. The none and space enums may not appear in the first field and space may not appear in the final field.

The parts of a money string must appear in the order indicated by the fields of the pattern. The symbol field indicates that the value of `curr_symbol()` may be present. The sign field indicates that the value of `positive_sign()` or `negative_sign()` must be present. The value field indicates that the absolute value of the money amount is present. none indicates 0 or more whitespace characters, except at the end, where it permits no whitespace. space indicates that 1 or more whitespace characters must be present.

For example, for the US locale and `pos_format()` pattern {symbol,sign,value,none}, `curr_symbol() == '$'` `positive_sign() == '+'`, and value 10.01, and options set to force the symbol, the corresponding string is `$+10.01`.

Returns

Pattern for money values.

Definition at line 1151 of file `locale_facets_nonio.h`.

4.638.2.15 `template<typename _CharT, bool _Intl> string_type std::moneypunct<_CharT, _Intl>::negative_sign () const`
`[inline], [inherited]`

Return negative sign string.

This function returns a `string_type` to use as a sign for negative amounts. It does so by returning `moneypunct<char_type>::do_negative_sign()`.

If the return value contains more than one character, the first character appears in the position indicated by `neg_format()` and the remainder appear at the end of the formatted string.

Returns

string_type representing a negative sign.

Definition at line 1095 of file `locale_facets_nonio.h`.

4.638.2.16 `template<typename _CharT, bool _Intl> pattern std::moneypunct<_CharT, _Intl>::pos_format () const`
`[inline], [inherited]`

Return pattern for money values.

This function returns a pattern describing the formatting of a positive or negative valued money amount. It does so by returning `moneypunct<char_type>::do_pos_format()` or `moneypunct<char_type>::do_neg_format()`.

The pattern has 4 fields describing the ordering of symbol, sign, value, and none or space. There must be one of each in the pattern. The none and space enums may not appear in the first field and space may not appear in the final field.

The parts of a money string must appear in the order indicated by the fields of the pattern. The symbol field indicates that the value of `curr_symbol()` may be present. The sign field indicates that the value of `positive_sign()` or `negative_sign()` must be present. The value field indicates that the absolute value of the money amount is present. none indicates 0 or more whitespace characters, except at the end, where it permits no whitespace. space indicates that 1 or more whitespace characters must be present.

For example, for the US locale and `pos_format()` pattern {symbol,sign,value,none}, `curr_symbol() == '$'` `positive_sign() == '+'`, and value 10.01, and options set to force the symbol, the corresponding string is \$+10.01.

Returns

Pattern for money values.

Definition at line 1147 of file `locale_facets_nonio.h`.

4.638.2.17 `template<typename _CharT, bool _Intl> string_type std::moneypunct<_CharT, _Intl>::positive_sign () const`
`[inline], [inherited]`

Return positive sign string.

This function returns a `string_type` to use as a sign for positive amounts. It does so by returning `moneypunct<char_type>::do_positive_sign()`.

If the return value contains more than one character, the first character appears in the position indicated by `pos_format()` and the remainder appear at the end of the formatted string.

Returns

string_type representing a positive sign.

Definition at line 1078 of file `locale_facets_nonio.h`.

4.638.2.18 `template<typename _CharT, bool _Intl> char_type std::moneypunct<_CharT, _Intl>::thousands_sep () const`
`[inline], [inherited]`

Return thousands separator character.

This function returns a `char_type` to use as a thousands separator. It does so by returning `moneypunct<char_type>::do_thousands_sep()`.

Returns

`char_type` representing a thousands separator.

Definition at line 1018 of file `locale_facets_nonio.h`.

4.638.3 Member Data Documentation

4.638.3.1 `template<typename _CharT, bool _Intl> locale::id std::moneypunct<_CharT, _Intl>::id` `[static]`,
`[inherited]`

Numpunct facet id.

Definition at line 953 of file `locale_facets_nonio.h`.

The documentation for this class was generated from the following file:

- [locale_facets_nonio.h](#)

4.639 `std::move_iterator<_Iterator>` Class Template Reference

Public Types

- `typedef __traits_type::difference_type` **difference_type**
- `typedef __traits_type::iterator_category` **iterator_category**
- `typedef _Iterator` **iterator_type**
- `typedef _Iterator` **pointer**
- `typedef value_type &&` **reference**
- `typedef __traits_type::value_type` **value_type**

Public Member Functions

- **move_iterator** (iterator_type __i)
- `template<typename _Iter>`
move_iterator (const [move_iterator](#)<_Iter> &__i)
- iterator_type **base** () const
- reference **operator*** () const
- [move_iterator](#) **operator+** (difference_type __n) const
- [move_iterator](#) & **operator++** ()
- [move_iterator](#) **operator++** (int)
- [move_iterator](#) & **operator+=** (difference_type __n)
- [move_iterator](#) **operator-** (difference_type __n) const
- [move_iterator](#) & **operator--** ()
- [move_iterator](#) **operator--** (int)
- [move_iterator](#) & **operator-=** (difference_type __n)
- pointer **operator->** () const
- reference **operator[]** (difference_type __n) const

Protected Types

- `typedef iterator_traits<_Iterator>` **__traits_type**

Protected Attributes

- `_Iterator _M_current`

4.639.1 Detailed Description

```
template<typename _Iterator>
class std::move_iterator<_Iterator>
```

Class template `move_iterator` is an iterator adapter with the same behavior as the underlying iterator except that its dereference operator implicitly converts the value returned by the underlying iterator's dereference operator to an rvalue reference. Some generic algorithms can be called with move iterators to replace copying with moving.

Definition at line 403 of file `cpp_type_traits.h`.

The documentation for this class was generated from the following files:

- [cpp_type_traits.h](#)
- [stl_iterator.h](#)

4.640 `std::multimap<_Key, _Tp, _Compare, _Alloc>` Class Template Reference

Public Types

- typedef `_Alloc` **allocator_type**
- typedef `_Rep_type::const_iterator` **const_iterator**
- typedef `_Alloc_traits::const_pointer` **const_pointer**
- typedef `_Alloc_traits::const_reference` **const_reference**
- typedef `_Rep_type::const_reverse_iterator` **const_reverse_iterator**
- typedef `_Rep_type::difference_type` **difference_type**
- typedef `_Rep_type::iterator` **iterator**
- typedef `_Compare` **key_compare**
- typedef `_Key` **key_type**
- typedef `_Tp` **mapped_type**
- typedef `_Alloc_traits::pointer` **pointer**
- typedef `_Alloc_traits::reference` **reference**
- typedef `_Rep_type::reverse_iterator` **reverse_iterator**
- typedef `_Rep_type::size_type` **size_type**
- typedef `std::pair< const _Key, _Tp >` **value_type**

Public Member Functions

- [multimap](#) ()
- [multimap](#) (const [_Compare](#) &__comp, const [allocator_type](#) &__a=allocator_type())
- [multimap](#) (const [multimap](#) &__x)
- [multimap](#) ([multimap](#) &&__x) noexcept(is_nothrow_copy_constructible< [_Compare](#) >::value)
- [multimap](#) (initializer_list< [value_type](#) > __l, const [_Compare](#) &__comp=_Compare(), const [allocator_type](#) &__a=allocator_type())
- [multimap](#) (const [allocator_type](#) &__a)
- [multimap](#) (const [multimap](#) &__m, const [allocator_type](#) &__a)
- [multimap](#) ([multimap](#) &&__m, const [allocator_type](#) &__a) noexcept(is_nothrow_copy_constructible< [_Compare](#) >::value && [_Alloc_traits::S_always_equal](#)())
- [multimap](#) (initializer_list< [value_type](#) > __l, const [allocator_type](#) &__a)
- template<typename [_InputIterator](#) >
[multimap](#) ([_InputIterator](#) __first, [_InputIterator](#) __last, const [allocator_type](#) &__a)
- template<typename [_InputIterator](#) >
[multimap](#) ([_InputIterator](#) __first, [_InputIterator](#) __last)
- template<typename [_InputIterator](#) >
[multimap](#) ([_InputIterator](#) __first, [_InputIterator](#) __last, const [_Compare](#) &__comp, const [allocator_type](#) &__a=allocator_type())
- iterator [begin](#) () noexcept
- const_iterator [begin](#) () const noexcept
- const_iterator [cbegin](#) () const noexcept
- const_iterator [cend](#) () const noexcept
- void [clear](#) () noexcept
- size_type [count](#) (const [key_type](#) &__x) const
- const_reverse_iterator [crbegin](#) () const noexcept
- const_reverse_iterator [crend](#) () const noexcept
- template<typename... [_Args](#)>
iterator [emplace](#) ([_Args](#) &&...__args)
- template<typename... [_Args](#)>
iterator [emplace_hint](#) (const_iterator __pos, [_Args](#) &&...__args)
- bool [empty](#) () const noexcept
- iterator [end](#) () noexcept
- const_iterator [end](#) () const noexcept
- [std::pair](#)< iterator, iterator > [equal_range](#) (const [key_type](#) &__x)
- [std::pair](#)< const_iterator, const_iterator > [equal_range](#) (const [key_type](#) &__x) const
- iterator [erase](#) (const_iterator __position)
- [_GLIBCXX_ABI_TAG_CXX11](#) iterator [erase](#) (iterator __position)
- size_type [erase](#) (const [key_type](#) &__x)
- iterator [erase](#) (const_iterator __first, const_iterator __last)
- iterator [find](#) (const [key_type](#) &__x)
- const_iterator [find](#) (const [key_type](#) &__x) const
- [allocator_type](#) [get_allocator](#) () const noexcept
- iterator [insert](#) (const [value_type](#) &__x)
- template<typename [_Pair](#) , typename = typename std::enable_if<std::is_constructible<[value_type](#), [_Pair](#)&&::value>::type>
iterator [insert](#) ([_Pair](#) &&__x)
- iterator [insert](#) (const_iterator __position, const [value_type](#) &__x)
- template<typename [_Pair](#) , typename = typename std::enable_if<std::is_constructible<[value_type](#), [_Pair](#)&&::value>::type>
iterator [insert](#) (const_iterator __position, [_Pair](#) &&__x)
- template<typename [_InputIterator](#) >
void [insert](#) ([_InputIterator](#) __first, [_InputIterator](#) __last)

- void [insert](#) (initializer_list< [value_type](#) > __l)
- key_compare [key_comp](#) () const
- iterator [lower_bound](#) (const key_type &__x)
- const_iterator [lower_bound](#) (const key_type &__x) const
- size_type [max_size](#) () const noexcept
- [multimap](#) & [operator=](#) (const [multimap](#) &__x)
- [multimap](#) & [operator=](#) ([multimap](#) &&__x) noexcept(_Alloc_traits::_S_nothrow_move())
- [multimap](#) & [operator=](#) (initializer_list< [value_type](#) > __l)
- [reverse_iterator](#) [rbegin](#) () noexcept
- [const_reverse_iterator](#) [rbegin](#) () const noexcept
- [reverse_iterator](#) [rend](#) () noexcept
- [const_reverse_iterator](#) [rend](#) () const noexcept
- size_type [size](#) () const noexcept
- void [swap](#) ([multimap](#) &__x) noexcept(_Alloc_traits::_S_nothrow_swap())
- iterator [upper_bound](#) (const key_type &__x)
- const_iterator [upper_bound](#) (const key_type &__x) const
- value_compare [value_comp](#) () const

Friends

- template<typename _K1, typename _T1, typename _C1, typename _A1 >
bool [operator](#)< (const [multimap](#)< _K1, _T1, _C1, _A1 > &, const [multimap](#)< _K1, _T1, _C1, _A1 > &)
- template<typename _K1, typename _T1, typename _C1, typename _A1 >
bool [operator==](#) (const [multimap](#)< _K1, _T1, _C1, _A1 > &, const [multimap](#)< _K1, _T1, _C1, _A1 > &)

4.640.1 Detailed Description

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>>>
class std::multimap<_Key, _Tp, _Compare, _Alloc>
```

A standard container made up of (key,value) pairs, which can be retrieved based on a key, in logarithmic time.

Template Parameters

_Key	Type of key objects.
_Tp	Type of mapped objects.
_Compare	Comparison function object type, defaults to less<_Key>.
_Alloc	Allocator type, defaults to allocator<pair<const _Key, _Tp>>.

Meets the requirements of a [container](#), a [reversible container](#), and an [associative container](#) (using equivalent keys). For a `multimap<Key, T>` the `key_type` is `Key`, the `mapped_type` is `T`, and the `value_type` is `std::pair<const Key, T>`.

Multimaps support bidirectional iterators.

The private tree data is declared exactly the same way for `map` and `multimap`; the distinction is made entirely in how the tree functions are called (`*_unique` versus `*_equal`, same as the standard).

Definition at line 95 of file stl_multimap.h.

4.640.2 Constructor & Destructor Documentation

4.640.2.1 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> std::multimap<_Key, _Tp, _Compare, _Alloc>::multimap ()`
`[inline]`

Default constructor creates no elements.

Definition at line 160 of file stl_multimap.h.

4.640.2.2 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> std::multimap<_Key, _Tp, _Compare, _Alloc>::multimap (const _Compare & __comp, const allocator_type & __a = allocator_type())` `[inline], [explicit]`

Creates a multimap with no elements.

Parameters

<code>__comp</code>	A comparison object.
<code>__a</code>	An allocator object.

Definition at line 169 of file stl_multimap.h.

4.640.2.3 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> std::multimap<_Key, _Tp, _Compare, _Alloc>::multimap (const multimap<_Key, _Tp, _Compare, _Alloc> & __x)` `[inline]`

Multimap copy constructor.

Parameters

<code>__x</code>	A multimap of identical element and allocator types.
------------------	--

The newly-created multimap uses a copy of the allocation object used by `__x`.

Definition at line 180 of file stl_multimap.h.

4.640.2.4 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> std::multimap<_Key, _Tp, _Compare, _Alloc>::multimap (multimap<_Key, _Tp, _Compare, _Alloc> && __x)` `[inline], [noexcept]`

Multimap move constructor.

Parameters

<code>__l</code>	A multimap of identical element and allocator types.
<code>__x</code>	

The newly-created multimap contains the exact contents of `__x`. The contents of `__x` are a valid, but unspecified multimap.

Definition at line 191 of file `stl_multimap.h`.

```
4.640.2.5 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> std::multimap<_Key, _Tp, _Compare, _Alloc>::multimap
( initializer_list<value_type> &__l, const _Compare &__comp = _Compare(), const allocator_type &__a =
allocator_type() ) [inline]
```

Builds a multimap from an `initializer_list`.

Parameters

<code>__l</code>	An <code>initializer_list</code> .
<code>__comp</code>	A comparison functor.
<code>__a</code>	An allocator object.

Create a multimap consisting of copies of the elements from the `initializer_list`. This is linear in N if the list is already sorted, and $N\log N$ otherwise (where N is `__l.size()`).

Definition at line 205 of file `stl_multimap.h`.

```
4.640.2.6 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> std::multimap<_Key, _Tp, _Compare, _Alloc>::multimap ( const
allocator_type &__a ) [inline],[explicit]
```

Allocator-extended default constructor.

Definition at line 213 of file `stl_multimap.h`.

```
4.640.2.7 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> std::multimap<_Key, _Tp, _Compare, _Alloc>::multimap ( const
multimap<_Key, _Tp, _Compare, _Alloc> &__m, const allocator_type &__a ) [inline]
```

Allocator-extended copy constructor.

Definition at line 217 of file `stl_multimap.h`.

```
4.640.2.8 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> std::multimap<_Key, _Tp, _Compare, _Alloc>::multimap (
multimap<_Key, _Tp, _Compare, _Alloc> &&__m, const allocator_type &__a ) [inline],[noexcept]
```

Allocator-extended move constructor.

Definition at line 221 of file `stl_multimap.h`.

```
4.640.2.9  template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >> std::multimap<_Key, _Tp, _Compare, _Alloc >::multimap (
initializer_list< value_type > __l, const allocator_type & __a ) [inline]
```

Allocator-extended initialier-list constructor.

Definition at line 227 of file stl_multimap.h.

```
4.640.2.10 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >> template<typename _InputIterator > std::multimap<_Key,
_Tp, _Compare, _Alloc >::multimap ( _InputIterator __first, _InputIterator __last, const allocator_type & __a )
[inline]
```

Allocator-extended range constructor.

Definition at line 233 of file stl_multimap.h.

```
4.640.2.11 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >> template<typename _InputIterator > std::multimap<_Key, _Tp,
_Compare, _Alloc >::multimap ( _InputIterator __first, _InputIterator __last ) [inline]
```

Builds a multimap from a range.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.

Create a multimap consisting of copies of the elements from [`__first`,`__last`). This is linear in N if the range is already sorted, and NlogN otherwise (where N is distance(`__first`,`__last`)).

Definition at line 249 of file stl_multimap.h.

```
4.640.2.12 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >> template<typename _InputIterator > std::multimap<_Key, _Tp,
_Compare, _Alloc >::multimap ( _InputIterator __first, _InputIterator __last, const _Compare & __comp, const
allocator_type & __a = allocator_type() ) [inline]
```

Builds a multimap from a range.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__comp</code>	A comparison functor.
<code>__a</code>	An allocator object.

Create a multimap consisting of copies of the elements from [`__first`,`__last`). This is linear in N if the range is already

sorted, and NlogN otherwise (where N is distance(__first,__last)).

Definition at line 265 of file stl_multimap.h.

4.640.3 Member Function Documentation

4.640.3.1 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> iterator std::multimap<_Key, _Tp, _Compare, _Alloc>::begin () [inline], [noexcept]`

Returns a read/write iterator that points to the first pair in the multimap. Iteration is done in ascending order according to the keys.

Definition at line 350 of file stl_multimap.h.

4.640.3.2 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> const_iterator std::multimap<_Key, _Tp, _Compare, _Alloc>::begin () const [inline], [noexcept]`

Returns a read-only (constant) iterator that points to the first pair in the multimap. Iteration is done in ascending order according to the keys.

Definition at line 359 of file stl_multimap.h.

4.640.3.3 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> const_iterator std::multimap<_Key, _Tp, _Compare, _Alloc>::cbegin () const [inline], [noexcept]`

Returns a read-only (constant) iterator that points to the first pair in the multimap. Iteration is done in ascending order according to the keys.

Definition at line 423 of file stl_multimap.h.

4.640.3.4 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> const_iterator std::multimap<_Key, _Tp, _Compare, _Alloc>::cend () const [inline], [noexcept]`

Returns a read-only (constant) iterator that points one past the last pair in the multimap. Iteration is done in ascending order according to the keys.

Definition at line 432 of file stl_multimap.h.

4.640.3.5 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> void std::multimap<_Key, _Tp, _Compare, _Alloc>::clear () [inline], [noexcept]`

Erases all elements in a multimap. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 736 of file stl_multimap.h.

Referenced by `std::multimap<_Key, _Tp, _Compare, _Alloc>::operator=()`.

4.640.3.6 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> size_type std::multimap<_Key, _Tp, _Compare, _Alloc>::count (const key_type & __x) const [inline]`

Finds the number of elements with given key.

Parameters

<code>_↔</code>	Key of (key, value) pairs to be located.
<code>_X</code>	

Returns

Number of elements with specified key.

Definition at line 793 of file `stl_multimap.h`.

```
4.640.3.7  template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
            std::allocator<std::pair<const _Key, _Tp> >> const_reverse_iterator std::multimap< _Key, _Tp, _Compare,
            _Alloc >::crbegin ( ) const    [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last pair in the multimap. Iteration is done in descending order according to the keys.

Definition at line 441 of file `stl_multimap.h`.

```
4.640.3.8  template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
            std::allocator<std::pair<const _Key, _Tp> >> const_reverse_iterator std::multimap< _Key, _Tp, _Compare,
            _Alloc >::crend ( ) const    [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first pair in the multimap. Iteration is done in descending order according to the keys.

Definition at line 450 of file `stl_multimap.h`.

```
4.640.3.9  template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
            std::allocator<std::pair<const _Key, _Tp> >> template<typename... _Args> iterator std::multimap< _Key, _Tp,
            _Compare, _Alloc >::emplace ( _Args &&... __args )    [inline]
```

Build and insert a `std::pair` into the multimap.

Parameters

<code>__args</code>	Arguments used to generate a new pair instance (see <code>std::piecewise_construct</code> for passing arguments to each part of the pair constructor).
---------------------	--

Returns

An iterator that points to the inserted (key,value) pair.

This function builds and inserts a (key, value) pair into the multimap. Contrary to a `std::map` the multimap does not rely on unique keys and thus multiple pairs with the same key can be inserted.

Insertion requires logarithmic time.

Definition at line 490 of file `stl_multimap.h`.

```
4.640.3.10 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> template<typename... _Args> iterator std::multimap<_Key, _Tp,
_Compare, _Alloc>::emplace_hint( const_iterator __pos, _Args&&... __args ) [inline]
```

Builds and inserts a std::pair into the multimap.

Parameters

<code>__pos</code>	An iterator that serves as a hint as to where the pair should be inserted.
<code>__args</code>	Arguments used to generate a new pair instance (see std::piecewise_construct for passing arguments to each part of the pair constructor).

Returns

An iterator that points to the inserted (key,value) pair.

This function inserts a (key, value) pair into the multimap. Contrary to a std::map the multimap does not rely on unique keys and thus multiple pairs with the same key can be inserted. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.html>.

Insertion requires logarithmic time (if the hint is not taken).

Definition at line 517 of file stl_multimap.h.

```
4.640.3.11 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> bool std::multimap<_Key, _Tp, _Compare, _Alloc>::empty( )
const [inline], [noexcept]
```

Returns true if the multimap is empty.

Definition at line 457 of file stl_multimap.h.

```
4.640.3.12 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> iterator std::multimap<_Key, _Tp, _Compare, _Alloc>::end( )
[inline], [noexcept]
```

Returns a read/write iterator that points one past the last pair in the multimap. Iteration is done in ascending order according to the keys.

Definition at line 368 of file stl_multimap.h.

```
4.640.3.13 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> const_iterator std::multimap<_Key, _Tp, _Compare, _Alloc>::end(
) const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last pair in the multimap. Iteration is done in ascending order according to the keys.

Definition at line 377 of file stl_multimap.h.

```
4.640.3.14 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> std::pair<iterator, iterator> std::multimap<_Key, _Tp, _Compare,
_Alloc>::equal_range( const key_type &__x ) [inline]
```

Finds a subsequence matching given key.

Parameters

$_↔$	Key of (key, value) pairs to be located.
$_X$	

Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
               c.upper_bound(val))
```

(but is faster than making the calls separately).

Definition at line 860 of file stl_multimap.h.

```
4.640.3.15 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >> std::pair<const_iterator, const_iterator> std::multimap<_Key,
_Tp, _Compare, _Alloc>::equal_range ( const key_type & __x ) const [inline]
```

Finds a subsequence matching given key.

Parameters

$_↔$	Key of (key, value) pairs to be located.
$_X$	

Returns

Pair of read-only (constant) iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
               c.upper_bound(val))
```

(but is faster than making the calls separately).

Definition at line 877 of file stl_multimap.h.

References std::operator==().

```
4.640.3.16 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >> iterator std::multimap<_Key, _Tp, _Compare, _Alloc>::erase (
const_iterator __position ) [inline]
```

Erases an element from a multimap.

Parameters

<code>__position</code>	An iterator pointing to the element to be erased.
-------------------------	---

Returns

An iterator pointing to the element immediately following *position* prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from a multimap. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 631 of file `stl_multimap.h`.

Referenced by `std::multimap< _Key, _Tp, _Compare, _Alloc >::erase()`.

```
4.640.3.17  template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
            std::allocator<std::pair<const _Key, _Tp> >> size_type std::multimap< _Key, _Tp, _Compare, _Alloc >::erase (
            const key_type & __x )  [inline]
```

Erases elements according to the provided key.

Parameters

<code>__key</code>	Key of element to be erased.
--------------------	------------------------------

Returns

The number of elements erased.

This function erases all elements located by the given key from a multimap. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 667 of file `stl_multimap.h`.

```
4.640.3.18  template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
            std::allocator<std::pair<const _Key, _Tp> >> iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::erase (
            const_iterator __first, const_iterator __last )  [inline]
```

Erases a `[first,last)` range of elements from a multimap.

Parameters

<code>__first</code>	Iterator pointing to the start of the range to be erased.
<code>__last</code>	Iterator pointing to the end of the range to be erased .

Returns

The iterator `__last`.

This function erases a sequence of elements from a multimap. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 688 of file `stl_multimap.h`.

References `std::multimap<_Key, _Tp, _Compare, _Alloc>::erase()`.

```
4.640.3.19 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> iterator std::multimap<_Key, _Tp, _Compare, _Alloc>::find ( const
key_type & __x ) [inline]
```

Tries to locate an element in a multimap.

Parameters

<code>__x</code>	Key of (key, value) pair to be located.
------------------	---

Returns

Iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after pair. If unsuccessful it returns the past-the-end (`end()`) iterator.

Definition at line 769 of file `stl_multimap.h`.

```
4.640.3.20 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> const_iterator std::multimap<_Key, _Tp, _Compare, _Alloc>::find (
const key_type & __x ) const [inline]
```

Tries to locate an element in a multimap.

Parameters

<code>__x</code>	Key of (key, value) pair to be located.
------------------	---

Returns

Read-only (constant) iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns a constant iterator pointing to the sought after pair. If unsuccessful it returns the past-the-end (`end()`) iterator.

Definition at line 784 of file `stl_multimap.h`.

```
4.640.3.21 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> allocator_type std::multimap<_Key, _Tp, _Compare, _Alloc
>::get_allocator( ) const [inline], [noexcept]
```

Get a copy of the memory allocation object.

Definition at line 340 of file stl_multimap.h.

```
4.640.3.22 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> iterator std::multimap<_Key, _Tp, _Compare, _Alloc >::insert (
const value_type & __x ) [inline]
```

Inserts a std::pair into the multimap.

Parameters

<code>__x</code>	Pair to be inserted (see std::make_pair for easy creation of pairs).
------------------	--

Returns

An iterator that points to the inserted (key,value) pair.

This function inserts a (key, value) pair into the multimap. Contrary to a std::map the multimap does not rely on unique keys and thus multiple pairs with the same key can be inserted.

Insertion requires logarithmic time.

Definition at line 537 of file stl_multimap.h.

Referenced by std::multimap< _Key, _Tp, _Compare, _Alloc >::insert(), and std::multimap< _Key, _Tp, _Compare, _Alloc >::operator=().

```
4.640.3.23 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> iterator std::multimap<_Key, _Tp, _Compare, _Alloc >::insert (
const_iterator __position, const value_type & __x ) [inline]
```

Inserts a std::pair into the multimap.

Parameters

<code>__position</code>	An iterator that serves as a hint as to where the pair should be inserted.
<code>__x</code>	Pair to be inserted (see std::make_pair for easy creation of pairs).

Returns

An iterator that points to the inserted (key,value) pair.

This function inserts a (key, value) pair into the multimap. Contrary to a `std::map` the multimap does not rely on unique keys and thus multiple pairs with the same key can be inserted. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17>.↩
html

Insertion requires logarithmic time (if the hint is not taken).

Definition at line 571 of file `stl_multimap.h`.

References `std::multimap<_Key, _Tp, _Compare, _Alloc>::insert()`.

```
4.640.3.24  template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
            std::allocator<std::pair<const _Key, _Tp>>> template<typename _InputIterator > void std::multimap<_Key,
            _Tp, _Compare, _Alloc>::insert ( _InputIterator __first, _InputIterator __last ) [inline]
```

A template function that attempts to insert a range of elements.

Parameters

<code>__first</code>	Iterator pointing to the start of the range to be inserted.
<code>__last</code>	Iterator pointing to the end of the range.

Complexity similar to that of the range constructor.

Definition at line 598 of file `stl_multimap.h`.

```
4.640.3.25  template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
            std::allocator<std::pair<const _Key, _Tp>>> void std::multimap<_Key, _Tp, _Compare, _Alloc>::insert (
            initializer_list<value_type> __l ) [inline]
```

Attempts to insert a list of `std::pairs` into the multimap.

Parameters

↩ _↩ ↩ _↩ /	A <code>std::initializer_list<value_type></code> of pairs to be inserted.
-------------------------	---

Complexity similar to that of the range constructor.

Definition at line 610 of file `stl_multimap.h`.

References `std::multimap<_Key, _Tp, _Compare, _Alloc>::insert()`.

```
4.640.3.26 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> key_compare std::multimap< _Key, _Tp, _Compare, _Alloc
>::key_comp ( ) const [inline]
```

Returns the key comparison object out of which the multimap was constructed.

Definition at line 745 of file stl_multimap.h.

```
4.640.3.27 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> iterator std::multimap< _Key, _Tp, _Compare, _Alloc
>::lower_bound ( const key_type & __x ) [inline]
```

Finds the beginning of a subsequence matching given key.

Parameters

<code>__x</code>	Key of (key, value) pair to be located.
-------------------------	---

Returns

Iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.

Definition at line 808 of file stl_multimap.h.

```
4.640.3.28 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> const_iterator std::multimap< _Key, _Tp, _Compare, _Alloc
>::lower_bound ( const key_type & __x ) const [inline]
```

Finds the beginning of a subsequence matching given key.

Parameters

<code>__x</code>	Key of (key, value) pair to be located.
-------------------------	---

Returns

Read-only (constant) iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful the iterator will point to the next greatest element or, if no such greater element exists, to end().

Definition at line 823 of file stl_multimap.h.

```
4.640.3.29  template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
            std::allocator<std::pair<const _Key, _Tp>>> size_type std::multimap<_Key, _Tp, _Compare, _Alloc>::max_size
            ( ) const    [inline], [noexcept]
```

Returns the maximum size of the multimap.

Definition at line 467 of file `stl_multimap.h`.

```
4.640.3.30  template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
            std::allocator<std::pair<const _Key, _Tp>>> multimap& std::multimap<_Key, _Tp, _Compare, _Alloc>
            >::operator= ( const multimap<_Key, _Tp, _Compare, _Alloc> &__x )    [inline]
```

Multimap assignment operator.

The dtor only erases the elements, and note that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Parameters

<code>__x</code>	A multimap of identical element and allocator types.
------------------	--

All the elements of `__x` are copied, but unlike the copy constructor, the allocator object is not copied.

Definition at line 288 of file `stl_multimap.h`.

```
4.640.3.31  template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
            std::allocator<std::pair<const _Key, _Tp>>> multimap& std::multimap<_Key, _Tp, _Compare, _Alloc>
            >::operator= ( multimap<_Key, _Tp, _Compare, _Alloc> &&__x )    [inline], [noexcept]
```

Multimap move assignment operator.

Parameters

<code>__x</code>	A multimap of identical element and allocator types.
------------------	--

The contents of `__x` are moved into this multimap (without copying if the allocators compare equal or get moved on assignment). Afterwards `__x` is in a valid, but unspecified state.

Definition at line 304 of file `stl_multimap.h`.

References `std::multimap<_Key, _Tp, _Compare, _Alloc>::clear()`, and `std::multimap<_Key, _Tp, _Compare, _Alloc>::insert()`.

```
4.640.3.32  template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
            std::allocator<std::pair<const _Key, _Tp>>> multimap& std::multimap<_Key, _Tp, _Compare, _Alloc>
            >::operator= ( initializer_list<value_type> __l )    [inline]
```

Multimap list assignment operator.

Parameters

↩	An initializer_list.
_↩	
↩	
_↩	
/	

This function fills a multimap with copies of the elements in the initializer list __l.

Note that the assignment completely changes the multimap and that the resulting multimap's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 330 of file stl_multimap.h.

References std::multimap< _Key, _Tp, _Compare, _Alloc >::clear(), and std::multimap< _Key, _Tp, _Compare, _Alloc >::insert().

4.640.3.33 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> reverse_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::rbegin () [inline], [noexcept]`

Returns a read/write reverse iterator that points to the last pair in the multimap. Iteration is done in descending order according to the keys.

Definition at line 386 of file stl_multimap.h.

4.640.3.34 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> const_reverse_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::rbegin () const [inline], [noexcept]`

Returns a read-only (constant) reverse iterator that points to the last pair in the multimap. Iteration is done in descending order according to the keys.

Definition at line 395 of file stl_multimap.h.

4.640.3.35 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> reverse_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::rend () [inline], [noexcept]`

Returns a read/write reverse iterator that points to one before the first pair in the multimap. Iteration is done in descending order according to the keys.

Definition at line 404 of file stl_multimap.h.

4.640.3.36 `template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>> const_reverse_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::rend () const [inline], [noexcept]`

Returns a read-only (constant) reverse iterator that points to one before the first pair in the multimap. Iteration is done in descending order according to the keys.

Definition at line 413 of file stl_multimap.h.

```
4.640.3.37  template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
            std::allocator<std::pair<const _Key, _Tp>>> size_type std::multimap<_Key, _Tp, _Compare, _Alloc >::size ( )
            const [inline],[noexcept]
```

Returns the size of the multimap.

Definition at line 462 of file stl_multimap.h.

```
4.640.3.38  template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
            std::allocator<std::pair<const _Key, _Tp>>> void std::multimap<_Key, _Tp, _Compare, _Alloc >::swap (
            multimap<_Key, _Tp, _Compare, _Alloc > &__x ) [inline],[noexcept]
```

Swaps data with another multimap.

Parameters

\leftrightarrow	A multimap of the same element and allocator types.
x	

This exchanges the elements between two multimaps in constant time. (It is only swapping a pointer, an integer, and an instance of the `Compare` type (which itself is often stateless and empty), so it should be quite fast.) Note that the global `std::swap()` function is specialized such that `std::swap(m1,m2)` will feed to this function.

Definition at line 723 of file stl_multimap.h.

Referenced by `std::swap()`.

```
4.640.3.39  template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
            std::allocator<std::pair<const _Key, _Tp>>> iterator std::multimap<_Key, _Tp, _Compare, _Alloc
            >::upper_bound ( const key_type &__x ) [inline]
```

Finds the end of a subsequence matching given key.

Parameters

\leftrightarrow	Key of (key, value) pair to be located.
x	

Returns

Iterator pointing to the first element greater than key, or `end()`.

Definition at line 833 of file stl_multimap.h.

```
4.640.3.40  template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
            std::allocator<std::pair<const _Key, _Tp>>> const_iterator std::multimap<_Key, _Tp, _Compare, _Alloc
            >::upper_bound ( const key_type &__x ) const [inline]
```

Finds the end of a subsequence matching given key.

Parameters

<code>_↔</code>	Key of (key, value) pair to be located.
<code>_X</code>	

Returns

Read-only (constant) iterator pointing to first iterator greater than key, or end().

Definition at line 843 of file `stl_multimap.h`.

```
4.640.3.41 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> value_compare std::multimap<_Key, _Tp, _Compare, _Alloc
>::value_comp ( ) const [inline]
```

Returns a value comparison object, built from the key comparison object out of which the multimap was constructed.

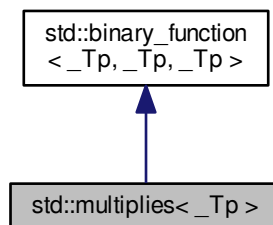
Definition at line 753 of file `stl_multimap.h`.

The documentation for this class was generated from the following file:

- [stl_multimap.h](#)

4.641 `std::multiplies<_Tp>` Struct Template Reference

Inheritance diagram for `std::multiplies<_Tp>`:



Public Types

- typedef `_Tp` [first_argument_type](#)
- typedef `_Tp` [result_type](#)
- typedef `_Tp` [second_argument_type](#)

Public Member Functions

- `_Tp operator()` (`const _Tp &__x`, `const _Tp &__y`) `const`

4.641.1 Detailed Description

```
template<typename _Tp>
struct std::multiplies<_Tp>
```

One of the [math functors](#).

Definition at line 185 of file `stl_function.h`.

4.641.2 Member Typedef Documentation

4.641.2.1 `typedef _Tp std::binary_function<_Tp, _Tp, _Tp>::first_argument_type` `[inherited]`

`first_argument_type` is the type of the first argument

Definition at line 121 of file `stl_function.h`.

4.641.2.2 `typedef _Tp std::binary_function<_Tp, _Tp, _Tp>::result_type` `[inherited]`

`result_type` is the return type

Definition at line 127 of file `stl_function.h`.

4.641.2.3 `typedef _Tp std::binary_function<_Tp, _Tp, _Tp>::second_argument_type` `[inherited]`

`second_argument_type` is the type of the second argument

Definition at line 124 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl_function.h](#)

4.642 std::multiset< _Key, _Compare, _Alloc > Class Template Reference

Public Types

- typedef _Alloc **allocator_type**
- typedef _Rep_type::const_iterator **const_iterator**
- typedef _Alloc_traits::const_pointer **const_pointer**
- typedef _Alloc_traits::const_reference **const_reference**
- typedef [_Rep_type::const_reverse_iterator](#) **const_reverse_iterator**
- typedef _Rep_type::difference_type **difference_type**
- typedef _Rep_type::const_iterator **iterator**
- typedef _Compare **key_compare**
- typedef _Key **key_type**
- typedef _Alloc_traits::pointer **pointer**
- typedef _Alloc_traits::reference **reference**
- typedef [_Rep_type::const_reverse_iterator](#) **reverse_iterator**
- typedef _Rep_type::size_type **size_type**
- typedef _Compare **value_compare**
- typedef _Key **value_type**

Public Member Functions

- [multiset](#) ()
- [multiset](#) (const _Compare &__comp, const allocator_type &__a=allocator_type())
- template<typename _InputIterator >
[multiset](#) (_InputIterator __first, _InputIterator __last)
- template<typename _InputIterator >
[multiset](#) (_InputIterator __first, _InputIterator __last, const _Compare &__comp, const allocator_type &__a=allocator_type())
- [multiset](#) (const [multiset](#) &__x)
- [multiset](#) ([multiset](#) &&__x) noexcept(is_nothrow_copy_constructible< _Compare >::value)
- [multiset](#) (initializer_list< value_type > __l, const _Compare &__comp=_Compare(), const allocator_type &__a=allocator_type())
- [multiset](#) (const allocator_type &__a)
- [multiset](#) (const [multiset](#) &__m, const allocator_type &__a)
- [multiset](#) ([multiset](#) &&__m, const allocator_type &__a) noexcept(is_nothrow_copy_constructible< _Compare >::value && _Alloc_traits::S_always_equal())
- [multiset](#) (initializer_list< value_type > __l, const allocator_type &__a)
- template<typename _InputIterator >
[multiset](#) (_InputIterator __first, _InputIterator __last, const allocator_type &__a)
- iterator [begin](#) () const noexcept
- iterator [cbegin](#) () const noexcept
- iterator [cend](#) () const noexcept
- void [clear](#) () noexcept
- size_type [count](#) (const key_type &__x) const
- [reverse_iterator](#) [crbegin](#) () const noexcept
- [reverse_iterator](#) [crend](#) () const noexcept
- template<typename... _Args>
iterator [emplace](#) (_Args &&...__args)

- `template<typename... _Args>`
`iterator emplace_hint (const_iterator __pos, _Args &&... __args)`
- `bool empty () const noexcept`
- `iterator end () const noexcept`
- `_GLIBCXX_ABI_TAG_CXX11 iterator erase (const_iterator __position)`
- `size_type erase (const key_type &__x)`
- `_GLIBCXX_ABI_TAG_CXX11 iterator erase (const_iterator __first, const_iterator __last)`
- `allocator_type get_allocator () const noexcept`
- `iterator insert (const value_type &__x)`
- `iterator insert (value_type && __x)`
- `iterator insert (const_iterator __position, const value_type &__x)`
- `iterator insert (const_iterator __position, value_type && __x)`
- `template<typename _InputIterator >`
`void insert (_InputIterator __first, _InputIterator __last)`
- `void insert (initializer_list< value_type > __l)`
- `key_compare key_comp () const`
- `size_type max_size () const noexcept`
- `multiset & operator= (const multiset &__x)`
- `multiset & operator= (multiset && __x) noexcept(_Alloc_traits::_S_nothrow_move())`
- `multiset & operator= (initializer_list< value_type > __l)`
- `reverse_iterator rbegin () const noexcept`
- `reverse_iterator rend () const noexcept`
- `size_type size () const noexcept`
- `void swap (multiset &__x) noexcept(_Alloc_traits::_S_nothrow_swap())`
- `value_compare value_comp () const`

- `iterator find (const key_type &__x)`
- `const_iterator find (const key_type &__x) const`

- `iterator lower_bound (const key_type &__x)`
- `const_iterator lower_bound (const key_type &__x) const`

- `iterator upper_bound (const key_type &__x)`
- `const_iterator upper_bound (const key_type &__x) const`

- `std::pair< iterator, iterator > equal_range (const key_type &__x)`
- `std::pair< const_iterator, const_iterator > equal_range (const key_type &__x) const`

Friends

- `template<typename _K1, typename _C1, typename _A1 >`
`bool operator< (const multiset< _K1, _C1, _A1 > &, const multiset< _K1, _C1, _A1 > &)`
- `template<typename _K1, typename _C1, typename _A1 >`
`bool operator== (const multiset< _K1, _C1, _A1 > &, const multiset< _K1, _C1, _A1 > &)`

4.642.1 Detailed Description

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
class std::multiset< _Key, _Compare, _Alloc >
```

A standard container made up of elements, which can be retrieved in logarithmic time.

Template Parameters

<code>_Key</code>	Type of key objects.
<code>_Compare</code>	Comparison function object type, defaults to <code>less<_Key></code> .
<code>_Alloc</code>	Allocator type, defaults to <code>allocator<_Key></code> .

Meets the requirements of a [container](#), a [reversible container](#), and an [associative container](#) (using equivalent keys). For a `multiset<Key>` the `key_type` and `value_type` are `Key`.

Multisets support bidirectional iterators.

The private tree data is declared exactly the same way for set and multiset; the distinction is made entirely in how the tree functions are called (`*_unique` versus `*_equal`, same as the standard).

Definition at line 92 of file `stl_multiset.h`.

4.642.2 Constructor & Destructor Documentation

4.642.2.1 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>>
std::multiset< _Key, _Compare, _Alloc >::multiset () [inline]`

Default constructor creates no elements.

Definition at line 140 of file `stl_multiset.h`.

4.642.2.2 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>>
std::multiset< _Key, _Compare, _Alloc >::multiset (const _Compare & __comp, const allocator_type & __a =
allocator_type()) [inline], [explicit]`

Creates a multiset with no elements.

Parameters

<code>__comp</code>	Comparator to use.
<code>__a</code>	An allocator object.

Definition at line 149 of file `stl_multiset.h`.

4.642.2.3 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>>
template<typename _InputIterator > std::multiset< _Key, _Compare, _Alloc >::multiset (_InputIterator __first,
_InputIterator __last) [inline]`

Builds a multiset from a range.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.

Create a multiset consisting of copies of the elements from [first,last). This is linear in N if the range is already sorted, and NlogN otherwise (where N is distance(__first,__last)).

Definition at line 163 of file stl_multiset.h.

```
4.642.2.4  template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
            template<typename _InputIterator > std::multiset<_Key, _Compare, _Alloc>::multiset ( _InputIterator __first,
            _InputIterator __last, const _Compare & __comp, const allocator_type & __a = allocator_type() )
            [inline]
```

Builds a multiset from a range.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__comp</code>	A comparison functor.
<code>__a</code>	An allocator object.

Create a multiset consisting of copies of the elements from [__first,__last). This is linear in N if the range is already sorted, and NlogN otherwise (where N is distance(__first,__last)).

Definition at line 179 of file stl_multiset.h.

```
4.642.2.5  template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
            std::multiset<_Key, _Compare, _Alloc>::multiset ( const multiset<_Key, _Compare, _Alloc> & __x )
            [inline]
```

Multiset copy constructor.

Parameters

<code>__x</code>	A multiset of identical element and allocator types.
------------------	--

The newly-created multiset uses a copy of the allocation object used by __x.

Definition at line 192 of file stl_multiset.h.

```
4.642.2.6  template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
            std::multiset<_Key, _Compare, _Alloc>::multiset ( multiset<_Key, _Compare, _Alloc> && __x ) [inline],
            [noexcept]
```

Multiset move constructor.

Parameters

<code>__x</code>	A multiset of identical element and allocator types.
------------------	--

The newly-created multiset contains the exact contents of `__x`. The contents of `__x` are a valid, but unspecified multiset.

Definition at line 203 of file `stl_multiset.h`.

```
4.642.2.7 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
    std::multiset< _Key, _Compare, _Alloc >::multiset ( initializer_list< value_type > __l, const _Compare & __comp =
        _Compare(), const allocator_type & __a = allocator_type() ) [inline]
```

Builds a multiset from an `initializer_list`.

Parameters

<code>__l</code>	An <code>initializer_list</code> .
<code>__comp</code>	A comparison functor.
<code>__a</code>	An allocator object.

Create a multiset consisting of copies of the elements from the list. This is linear in `N` if the list is already sorted, and `NlogN` otherwise (where `N` is `__l.size()`).

Definition at line 217 of file `stl_multiset.h`.

```
4.642.2.8 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
    std::multiset< _Key, _Compare, _Alloc >::multiset ( const allocator_type & __a ) [inline], [explicit]
```

Allocator-extended default constructor.

Definition at line 225 of file `stl_multiset.h`.

```
4.642.2.9 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
    std::multiset< _Key, _Compare, _Alloc >::multiset ( const multiset< _Key, _Compare, _Alloc > & __m, const
    allocator_type & __a ) [inline]
```

Allocator-extended copy constructor.

Definition at line 229 of file `stl_multiset.h`.

```
4.642.2.10 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
    std::multiset< _Key, _Compare, _Alloc >::multiset ( multiset< _Key, _Compare, _Alloc > && __m, const
    allocator_type & __a ) [inline], [noexcept]
```

Allocator-extended move constructor.

Definition at line 233 of file `stl_multiset.h`.

```
4.642.2.11 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
    std::multiset< _Key, _Compare, _Alloc >::multiset ( initializer_list< value_type > __l, const allocator_type & __a )
    [inline]
```

Allocator-extended initialier-list constructor.

Definition at line 239 of file `stl_multiset.h`.

4.642.2.12 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
 template<typename _InputIterator > std::multiset<_Key, _Compare, _Alloc>::multiset (_InputIterator __first,
 _InputIterator __last, const allocator_type & __a) [inline]`

Allocator-extended range constructor.

Definition at line 245 of file `stl_multiset.h`.

4.642.3 Member Function Documentation

4.642.3.1 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> iterator
 std::multiset<_Key, _Compare, _Alloc>::begin () const [inline], [noexcept]`

Returns a read-only (constant) iterator that points to the first element in the multiset. Iteration is done in ascending order according to the keys.

Definition at line 330 of file `stl_multiset.h`.

4.642.3.2 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> iterator
 std::multiset<_Key, _Compare, _Alloc>::cbegin () const [inline], [noexcept]`

Returns a read-only (constant) iterator that points to the first element in the multiset. Iteration is done in ascending order according to the keys.

Definition at line 367 of file `stl_multiset.h`.

4.642.3.3 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> iterator
 std::multiset<_Key, _Compare, _Alloc>::cend () const [inline], [noexcept]`

Returns a read-only (constant) iterator that points one past the last element in the multiset. Iteration is done in ascending order according to the keys.

Definition at line 376 of file `stl_multiset.h`.

4.642.3.4 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> void
 std::multiset<_Key, _Compare, _Alloc>::clear () [inline], [noexcept]`

Erases all elements in a multiset. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 654 of file `stl_multiset.h`.

Referenced by `std::multiset<_Key, _Compare, _Alloc>::operator=()`.

4.642.3.5 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
 size_type std::multiset<_Key, _Compare, _Alloc>::count (const key_type & __x) const [inline]`

Finds the number of elements with given key.

Parameters

<code>_↔</code>	Key of elements to be located.
<code>_X</code>	

Returns

Number of elements with specified key.

Definition at line 665 of file stl_multiset.h.

```
4.642.3.6 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
reverse_iterator std::multiset<_Key, _Compare, _Alloc>::crbegin ( ) const [inline],[noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last element in the multiset. Iteration is done in descending order according to the keys.

Definition at line 385 of file stl_multiset.h.

```
4.642.3.7 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
reverse_iterator std::multiset<_Key, _Compare, _Alloc>::crend ( ) const [inline],[noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last element in the multiset. Iteration is done in descending order according to the keys.

Definition at line 394 of file stl_multiset.h.

```
4.642.3.8 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
template<typename... _Args> iterator std::multiset<_Key, _Compare, _Alloc>::emplace ( _Args &&... _args )
[inline]
```

Builds and inserts an element into the multiset.

Parameters

<code>__args</code>	Arguments used to generate the element instance to be inserted.
---------------------	---

Returns

An iterator that points to the inserted element.

This function inserts an element into the multiset. Contrary to a `std::set` the multiset does not rely on unique keys and thus multiple copies of the same element can be inserted.

Insertion requires logarithmic time.

Definition at line 447 of file stl_multiset.h.

```
4.642.3.9  template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>>
            template<typename... _Args> iterator std::multiset<_Key, _Compare, _Alloc>::emplace_hint( const_iterator __pos,
            _Args &&... __args ) [inline]
```

Builds and inserts an element into the multiset.

Parameters

<code>__pos</code>	An iterator that serves as a hint as to where the element should be inserted.
<code>__args</code>	Arguments used to generate the element instance to be inserted.

Returns

An iterator that points to the inserted element.

This function inserts an element into the multiset. Contrary to a `std::set` the multiset does not rely on unique keys and thus multiple copies of the same element can be inserted.

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.html> for more on *hinting*.

Insertion requires logarithmic time (if the hint is not taken).

Definition at line 473 of file `stl_multiset.h`.

```
4.642.3.10  template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>> bool
            std::multiset<_Key, _Compare, _Alloc>::empty( ) const [inline], [noexcept]
```

Returns true if the set is empty.

Definition at line 400 of file `stl_multiset.h`.

```
4.642.3.11  template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>> iterator
            std::multiset<_Key, _Compare, _Alloc>::end( ) const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the multiset. Iteration is done in ascending order according to the keys.

Definition at line 339 of file `stl_multiset.h`.

```
4.642.3.12  template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>>
            std::pair<iterator, iterator> std::multiset<_Key, _Compare, _Alloc>::equal_range( const key_type & __x )
            [inline]
```

Finds a subsequence matching given key.

Parameters

<code>_↔</code>	Key to be located.
<code>_X</code>	

Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
               c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multisets.

Definition at line 745 of file stl_multiset.h.

```
4.642.3.13 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
            std::pair<const_iterator, const_iterator> std::multiset<_Key, _Compare, _Alloc >::equal_range ( const key_type
            & __x ) const    [inline]
```

Finds a subsequence matching given key.

Parameters

<code>_↔</code>	Key to be located.
<code>_X</code>	

Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
               c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multisets.

Definition at line 749 of file stl_multiset.h.

```
4.642.3.14 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
            _GLIBCXX_ABI_TAG_CXX11 iterator std::multiset<_Key, _Compare, _Alloc >::erase ( const_iterator __position )
            [inline]
```

Erases an element from a multiset.

Parameters

<code>__position</code>	An iterator pointing to the element to be erased.
-------------------------	---

Returns

An iterator pointing to the element immediately following *position* prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from a multiset. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 575 of file `stl_multiset.h`.

Referenced by `std::multiset<_Key, _Compare, _Alloc>::erase()`.

```
4.642.3.15  template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
            size_type std::multiset<_Key, _Compare, _Alloc>::erase ( const key_type & __x )  [inline]
```

Erases elements according to the provided key.

Parameters

<code>__x</code>	Key of element to be erased.
------------------	------------------------------

Returns

The number of elements erased.

This function erases all elements located by the given key from a multiset. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 605 of file `stl_multiset.h`.

```
4.642.3.16  template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
            _GLIBCXX_ABI_TAG_CXX11 iterator std::multiset<_Key, _Compare, _Alloc>::erase ( const_iterator __first,
            const_iterator __last )  [inline]
```

Erases a `[first,last)` range of elements from a multiset.

Parameters

<code>__first</code>	Iterator pointing to the start of the range to be erased.
<code>__last</code>	Iterator pointing to the end of the range to be erased.

Returns

The iterator *last*.

This function erases a sequence of elements from a multiset. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 627 of file stl_multiset.h.

References std::multiset< _Key, _Compare, _Alloc >::erase().

4.642.3.17 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> iterator
std::multiset< _Key, _Compare, _Alloc >::find (const key_type & __x) [inline]`

Tries to locate an element in a set.

Parameters

<code>__x</code>	Element to be located.
------------------	------------------------

Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end (`end()`) iterator.

Definition at line 683 of file stl_multiset.h.

4.642.3.18 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
const_iterator std::multiset< _Key, _Compare, _Alloc >::find (const key_type & __x) const [inline]`

Tries to locate an element in a set.

Parameters

<code>__x</code>	Element to be located.
------------------	------------------------

Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end (`end()`) iterator.

Definition at line 687 of file stl_multiset.h.

4.642.3.19 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
allocator_type std::multiset< _Key, _Compare, _Alloc >::get_allocator () const [inline], [noexcept]`

Returns the memory allocation object.

Definition at line 321 of file `stl_multiset.h`.

4.642.3.20 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> iterator
std::multiset< _Key, _Compare, _Alloc >::insert (const value_type & __x) [inline]`

Inserts an element into the multiset.

Parameters

<code>__x</code>	Element to be inserted.
------------------	-------------------------

Returns

An iterator that points to the inserted element.

This function inserts an element into the multiset. Contrary to a `std::set` the multiset does not rely on unique keys and thus multiple copies of the same element can be inserted.

Insertion requires logarithmic time.

Definition at line 492 of file `stl_multiset.h`.

References `std::move()`.

Referenced by `std::multiset< _Key, _Compare, _Alloc >::insert()`, and `std::multiset< _Key, _Compare, _Alloc >::operator=()`.

4.642.3.21 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> iterator
std::multiset< _Key, _Compare, _Alloc >::insert (const_iterator __position, const value_type & __x) [inline]`

Inserts an element into the multiset.

Parameters

<code>__position</code>	An iterator that serves as a hint as to where the element should be inserted.
<code>__x</code>	Element to be inserted.

Returns

An iterator that points to the inserted element.

This function inserts an element into the multiset. Contrary to a `std::set` the multiset does not rely on unique keys and thus multiple copies of the same element can be inserted.

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.html> for more on *hinting*.

Insertion requires logarithmic time (if the hint is not taken).

Definition at line 522 of file stl_multiset.h.

References std::multiset< _Key, _Compare, _Alloc >::insert(), and std::move().

```
4.642.3.22 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
            template<typename _InputIterator > void std::multiset< _Key, _Compare, _Alloc >::insert ( _InputIterator __first,
            _InputIterator __last ) [inline]
```

A template function that tries to insert a range of elements.

Parameters

<code>__first</code>	Iterator pointing to the start of the range to be inserted.
<code>__last</code>	Iterator pointing to the end of the range.

Complexity similar to that of the range constructor.

Definition at line 541 of file stl_multiset.h.

```
4.642.3.23 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> void
            std::multiset< _Key, _Compare, _Alloc >::insert ( initializer_list< value_type > __l ) [inline]
```

Attempts to insert a list of elements into the multiset.

Parameters

<code>←</code> <code>__l</code> <code>←</code> <code>←</code> <code>__l</code> <code>l</code>	A std::initializer_list<value_type> of elements to be inserted.
--	---

Complexity similar to that of the range constructor.

Definition at line 553 of file stl_multiset.h.

References std::multiset< _Key, _Compare, _Alloc >::insert().

```
4.642.3.24 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
            key_compare std::multiset< _Key, _Compare, _Alloc >::key_comp ( ) const [inline]
```

Returns the comparison object.

Definition at line 313 of file stl_multiset.h.

4.642.3.25 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> iterator
std::multiset<_Key, _Compare, _Alloc>::lower_bound (const key_type & __x) [inline]`

Finds the beginning of a subsequence matching given key.

Parameters

<code>__x</code>	Key to be located.
------------------	--------------------

Returns

Iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.

Definition at line 704 of file `stl_multiset.h`.

4.642.3.26 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
const_iterator std::multiset<_Key, _Compare, _Alloc>::lower_bound (const key_type & __x) const [inline]`

Finds the beginning of a subsequence matching given key.

Parameters

<code>__x</code>	Key to be located.
------------------	--------------------

Returns

Iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.

Definition at line 708 of file `stl_multiset.h`.

4.642.3.27 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
size_type std::multiset<_Key, _Compare, _Alloc>::max_size () const [inline], [noexcept]`

Returns the maximum size of the set.

Definition at line 410 of file `stl_multiset.h`.

4.642.3.28 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
multiset& std::multiset<_Key, _Compare, _Alloc>::operator= (const multiset<_Key, _Compare, _Alloc> & __x
) [inline]`

Multiset assignment operator.

Parameters

<code>__x</code>	A multiset of identical element and allocator types.
------------------	--

All the elements of `__x` are copied, but unlike the copy constructor, the allocator object is not copied.

Definition at line 259 of file `stl_multiset.h`.

```
4.642.3.29 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
multiset& std::multiset< _Key, _Compare, _Alloc >::operator= ( multiset< _Key, _Compare, _Alloc > && __x )
[inline], [noexcept]
```

Multiset move assignment operator.

Parameters

<code>__x</code>	A multiset of identical element and allocator types.
------------------	--

The contents of `__x` are moved into this multiset (without copying if the allocators compare equal or get moved on assignment). Afterwards `__x` is in a valid, but unspecified state.

Definition at line 275 of file `stl_multiset.h`.

References `std::multiset< _Key, _Compare, _Alloc >::clear()`, and `std::multiset< _Key, _Compare, _Alloc >::insert()`.

```
4.642.3.30 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
multiset& std::multiset< _Key, _Compare, _Alloc >::operator= ( initializer_list< value_type > __l ) [inline]
```

Multiset list assignment operator.

Parameters

<code>__l</code>	An <code>initializer_list</code> .
------------------	------------------------------------

This function fills a multiset with copies of the elements in the initializer list `__l`.

Note that the assignment completely changes the multiset and that the resulting multiset's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 301 of file `stl_multiset.h`.

References `std::multiset< _Key, _Compare, _Alloc >::clear()`, and `std::multiset< _Key, _Compare, _Alloc >::insert()`.

```
4.642.3.31  template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
            reverse_iterator std::multiset<_Key, _Compare, _Alloc>::rbegin( ) const  [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last element in the multiset. Iteration is done in descending order according to the keys.

Definition at line 348 of file stl_multiset.h.

```
4.642.3.32  template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
            reverse_iterator std::multiset<_Key, _Compare, _Alloc>::rend( ) const  [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last element in the multiset. Iteration is done in descending order according to the keys.

Definition at line 357 of file stl_multiset.h.

```
4.642.3.33  template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
            size_type std::multiset<_Key, _Compare, _Alloc>::size( ) const  [inline], [noexcept]
```

Returns the size of the set.

Definition at line 405 of file stl_multiset.h.

```
4.642.3.34  template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> void
            std::multiset<_Key, _Compare, _Alloc>::swap( multiset<_Key, _Compare, _Alloc> & __x ) [inline],
            [noexcept]
```

Swaps data with another multiset.

Parameters

<code>__x</code>	A multiset of the same element and allocator types.
------------------	---

This exchanges the elements between two multisets in constant time. (It is only swapping a pointer, an integer, and an instance of the `Compare` type (which itself is often stateless and empty), so it should be quite fast.) Note that the global `std::swap()` function is specialized such that `std::swap(s1,s2)` will feed to this function.

Definition at line 425 of file stl_multiset.h.

Referenced by `std::swap()`.

```
4.642.3.35  template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> iterator
            std::multiset<_Key, _Compare, _Alloc>::upper_bound( const key_type & __x ) [inline]
```

Finds the end of a subsequence matching given key.

Parameters

$_x$	Key to be located.
-------	--------------------

Returns

Iterator pointing to the first element greater than key, or end().

Definition at line 720 of file stl_multiset.h.

```
4.642.3.36 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
const_iterator std::multiset<_Key, _Compare, _Alloc>::upper_bound ( const key_type & __x ) const [inline]
```

Finds the end of a subsequence matching given key.

Parameters

$_x$	Key to be located.
-------	--------------------

Returns

Iterator pointing to the first element greater than key, or end().

Definition at line 724 of file stl_multiset.h.

```
4.642.3.37 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
value_compare std::multiset<_Key, _Compare, _Alloc>::value_comp ( ) const [inline]
```

Returns the comparison object.

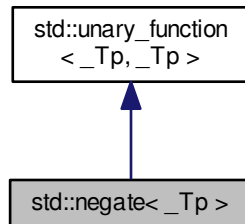
Definition at line 317 of file stl_multiset.h.

The documentation for this class was generated from the following file:

- [stl_multiset.h](#)

4.643 `std::negate<_Tp>` Struct Template Reference

Inheritance diagram for `std::negate<_Tp>`:



Public Types

- typedef `_Tp` [argument_type](#)
- typedef `_Tp` [result_type](#)

Public Member Functions

- `_Tp` **operator()** (const `_Tp` &__x) const

4.643.1 Detailed Description

```
template<typename _Tp>
struct std::negate<_Tp>
```

One of the [math functors](#).

Definition at line 212 of file `stl_function.h`.

4.643.2 Member Typedef Documentation

4.643.2.1 typedef `_Tp` `std::unary_function<_Tp, _Tp>::argument_type` [inherited]

`argument_type` is the type of the argument

Definition at line 108 of file `stl_function.h`.

4.643.2.2 `typedef _Tp std::unary_function<_Tp, _Tp>::result_type` [inherited]

`result_type` is the return type

Definition at line 111 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl_function.h](#)

4.644 `std::negative_binomial_distribution<_IntType>` Class Template Reference

Classes

- struct [param_type](#)

Public Types

- `typedef _IntType` [result_type](#)

Public Member Functions

- **`negative_binomial_distribution`** (`_IntType` __k=1, `double` __p=0.5)
- **`negative_binomial_distribution`** (const [param_type](#) &__p)
- `template<typename _ForwardIterator, typename _UniformRandomNumberGenerator >`
`void` **`__generate`** (`_ForwardIterator` __f, `_ForwardIterator` __t, `_UniformRandomNumberGenerator` &__urng)
- `template<typename _ForwardIterator, typename _UniformRandomNumberGenerator >`
`void` **`__generate`** (`_ForwardIterator` __f, `_ForwardIterator` __t, `_UniformRandomNumberGenerator` &__urng, const [param_type](#) &__p)
- `template<typename _UniformRandomNumberGenerator >`
`void` **`__generate`** ([result_type](#) *__f, [result_type](#) *__t, `_UniformRandomNumberGenerator` &__urng)
- `template<typename _UniformRandomNumberGenerator >`
`void` **`__generate`** ([result_type](#) *__f, [result_type](#) *__t, `_UniformRandomNumberGenerator` &__urng, const [param_type](#) &__p)
- `_IntType` **`k`** () const
- [result_type](#) **`max`** () const
- [result_type](#) **`min`** () const
- `template<typename _UniformRandomNumberGenerator >`
[result_type](#) **`operator()`** (`_UniformRandomNumberGenerator` &__urng)
- `template<typename _UniformRandomNumberGenerator >`
[result_type](#) **`operator()`** (`_UniformRandomNumberGenerator` &__urng, const [param_type](#) &__p)
- `double` **`p`** () const
- [param_type](#) **`param`** () const
- `void` **`param`** (const [param_type](#) &__param)
- `void` **`reset`** ()

Friends

- `template<typename _IntType1, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`
`std::negative_binomial_distribution< _IntType1 > &__x)`
- `bool operator== (const negative_binomial_distribution &__d1, const negative_binomial_distribution &__d2)`
- `template<typename _IntType1, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::`
`::negative_binomial_distribution< _IntType1 > &__x)`

4.644.1 Detailed Description

```
template<typename _IntType = int>
class std::negative_binomial_distribution< _IntType >
```

A negative_binomial_distribution random number distribution.

The formula for the negative binomial probability mass function is $p(i) = \binom{n}{i} p^i (1-p)^{t-i}$ where t and p are the parameters of the distribution.

Definition at line 4210 of file random.h.

4.644.2 Member Typedef Documentation

4.644.2.1 `template<typename _IntType = int> typedef _IntType std::negative_binomial_distribution< _IntType >::result_type`

The type of the range of the distribution.

Definition at line 4213 of file random.h.

4.644.3 Member Function Documentation

4.644.3.1 `template<typename _IntType = int> _IntType std::negative_binomial_distribution< _IntType >::k () const`
`[inline]`

Return the k parameter of the distribution.

Definition at line 4268 of file random.h.

4.644.3.2 `template<typename _IntType = int> result_type std::negative_binomial_distribution< _IntType >::max ()`
`const [inline]`

Returns the least upper bound value of the distribution.

Definition at line 4304 of file random.h.

References `std::max()`.

4.644.3.3 `template<typename _IntType = int> result_type std::negative_binomial_distribution<_IntType>::min ()
const [inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 4297 of file random.h.

4.644.3.4 `template<typename _IntType = int> template<typename _UniformRandomNumberGenerator> result_type
std::negative_binomial_distribution<_IntType>::operator() (_UniformRandomNumberGenerator & __urng)`

Generating functions.

4.644.3.5 `template<typename _IntType = int> double std::negative_binomial_distribution<_IntType>::p () const
[inline]`

Return the p parameter of the distribution.

Definition at line 4275 of file random.h.

4.644.3.6 `template<typename _IntType = int> param_type std::negative_binomial_distribution<_IntType>::param ()
const [inline]`

Returns the parameter set of the distribution.

Definition at line 4282 of file random.h.

4.644.3.7 `template<typename _IntType = int> void std::negative_binomial_distribution<_IntType>::param (const
param_type & __param) [inline]`

Sets the parameter set of the distribution.

Parameters

<code>__param</code>	The new parameter set of the distribution.
----------------------	--

Definition at line 4290 of file random.h.

4.644.3.8 `template<typename _IntType = int> void std::negative_binomial_distribution<_IntType>::reset ()
[inline]`

Resets the distribution state.

Definition at line 4261 of file random.h.

4.644.4 Friends And Related Function Documentation

```
4.644.4.1  template<typename _IntType = int> template<typename _IntType1 , typename _CharT , typename _Traits >
            std::basic_ostream<_CharT, _Traits>& operator<< ( std::basic_ostream<_CharT, _Traits > &__os, const
            std::negative_binomial_distribution<_IntType1 > &__x ) [friend]
```

Inserts a `negative_binomial_distribution` random number distribution `__x` into the output stream `__os`.

Parameters

<code>__os</code>	An output stream.
<code>__x</code>	A <code>negative_binomial_distribution</code> random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

```
4.644.4.2  template<typename _IntType = int> bool operator==( const negative_binomial_distribution<_IntType > &__d1,
                  const negative_binomial_distribution<_IntType > &__d2 ) [friend]
```

Return true if two negative binomial distributions have the same parameters and the sequences that would be generated are equal.

Definition at line 4353 of file `random.h`.

```
4.644.4.3  template<typename _IntType = int> template<typename _IntType1 , typename _CharT , typename _Traits
            > std::basic_istream<_CharT, _Traits>& operator>> ( std::basic_istream<_CharT, _Traits > &__is,
            std::negative_binomial_distribution<_IntType1 > &__x ) [friend]
```

Extracts a `negative_binomial_distribution` random number distribution `__x` from the input stream `__is`.

Parameters

<code>__is</code>	An input stream.
<code>__x</code>	A <code>negative_binomial_distribution</code> random number generator engine.

Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following file:

- [random.h](#)

4.645 `std::negative_binomial_distribution<_IntType>::param_type` Struct Reference

Public Types

- typedef `negative_binomial_distribution<_IntType>` `distribution_type`

Public Member Functions

- **param_type** (`_IntType __k=1`, `double __p=0.5`)
- `_IntType k` () const
- `double p` () const

Friends

- `bool operator==` (const [param_type](#) &__p1, const [param_type](#) &__p2)

4.645.1 Detailed Description

```
template<typename _IntType = int>
struct std::negative_binomial_distribution< _IntType >::param_type
```

Parameter type.

Definition at line 4219 of file `random.h`.

The documentation for this struct was generated from the following file:

- [random.h](#)

4.646 `std::nested_exception` Class Reference

Inherited by `std::_Nested_exception< _Except >`.

Public Member Functions

- **nested_exception** (const [nested_exception](#) &)=default
- `exception_ptr nested_ptr` () const
- [nested_exception](#) & **operator=** (const [nested_exception](#) &)=default
- `void rethrow_nested` () const `__attribute__((__noreturn__))`

4.646.1 Detailed Description

Exception class with `exception_ptr` data member.

Definition at line 55 of file `nested_exception.h`.

The documentation for this class was generated from the following file:

- [nested_exception.h](#)

4.647 `std::normal_distribution<_RealType>` Class Template Reference

Classes

- struct [param_type](#)

Public Types

- typedef `_RealType` [result_type](#)

Public Member Functions

- [normal_distribution](#) ([result_type](#) __mean=[result_type](#)(0), [result_type](#) __stddev=[result_type](#)(1))
- **normal_distribution** (const [param_type](#) &__p)
- template<typename `_ForwardIterator`, typename `_UniformRandomNumberGenerator` >
void **generate** (`_ForwardIterator` __f, `_ForwardIterator` __t, `_UniformRandomNumberGenerator` &__urng)
- template<typename `_ForwardIterator`, typename `_UniformRandomNumberGenerator` >
void **generate** (`_ForwardIterator` __f, `_ForwardIterator` __t, `_UniformRandomNumberGenerator` &__urng, const [param_type](#) &__p)
- template<typename `_UniformRandomNumberGenerator` >
void **generate** ([result_type](#) *__f, [result_type](#) *__t, `_UniformRandomNumberGenerator` &__urng, const [param_type](#) &__p)
- [result_type](#) max () const
- `_RealType` mean () const
- [result_type](#) min () const
- template<typename `_UniformRandomNumberGenerator` >
[result_type](#) operator() (`_UniformRandomNumberGenerator` &__urng)
- template<typename `_UniformRandomNumberGenerator` >
[result_type](#) operator() (`_UniformRandomNumberGenerator` &__urng, const [param_type](#) &__p)
- [param_type](#) param () const
- void [param](#) (const [param_type](#) &__param)
- void [reset](#) ()
- `_RealType` stddev () const

Friends

- template<typename `_RealType1`, typename `_CharT`, typename `_Traits` >
std::basic_ostream< `_CharT`, `_Traits` > & operator<< (std::basic_ostream< `_CharT`, `_Traits` > &__os, const [std::normal_distribution](#)< `_RealType1` > &__x)
- template<typename `_RealType1` >
bool operator== (const [std::normal_distribution](#)< `_RealType1` > &__d1, const [std::normal_distribution](#)< `_RealType1` > &__d2)
- template<typename `_RealType1`, typename `_CharT`, typename `_Traits` >
std::basic_istream< `_CharT`, `_Traits` > & operator>> (std::basic_istream< `_CharT`, `_Traits` > &__is, [std::normal_distribution](#)< `_RealType1` > &__x)

4.647.1 Detailed Description

```
template<typename _RealType = double>
class std::normal_distribution<_RealType>
```

A normal continuous distribution for random numbers.

The formula for the normal probability density function is

$$p(x|\mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x-\mu}{2\sigma^2}}$$

Definition at line 2087 of file random.h.

4.647.2 Member Typedef Documentation

4.647.2.1 `template<typename _RealType = double> typedef _RealType std::normal_distribution<_RealType>::result_type`

The type of the range of the distribution.

Definition at line 2090 of file random.h.

4.647.3 Constructor & Destructor Documentation

4.647.3.1 `template<typename _RealType = double> std::normal_distribution<_RealType>::normal_distribution (result_type __mean = result_type(0), result_type __stddev = result_type(1)) [inline], [explicit]`

Constructs a normal distribution with parameters *mean* and standard deviation.

Definition at line 2132 of file random.h.

4.647.4 Member Function Documentation

4.647.4.1 `template<typename _RealType = double> result_type std::normal_distribution<_RealType>::max () const [inline]`

Returns the least upper bound value of the distribution.

Definition at line 2189 of file random.h.

4.647.4.2 `template<typename _RealType = double> _RealType std::normal_distribution<_RealType>::mean () const [inline]`

Returns the mean of the distribution.

Definition at line 2153 of file random.h.

4.647.4.3 `template<typename _RealType = double> result_type std::normal_distribution<_RealType>::min () const`
`[inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 2182 of file random.h.

4.647.4.4 `template<typename _RealType = double> template<typename _UniformRandomNumberGenerator> result_type`
`std::normal_distribution<_RealType>::operator() (_UniformRandomNumberGenerator & __urng) [inline]`

Generating functions.

Definition at line 2197 of file random.h.

4.647.4.5 `template<typename _RealType = double> param_type std::normal_distribution<_RealType>::param () const`
`[inline]`

Returns the parameter set of the distribution.

Definition at line 2167 of file random.h.

4.647.4.6 `template<typename _RealType = double> void std::normal_distribution<_RealType>::param (const`
`param_type & __param) [inline]`

Sets the parameter set of the distribution.

Parameters

<code>__param</code>	The new parameter set of the distribution.
----------------------	--

Definition at line 2175 of file random.h.

4.647.4.7 `template<typename _RealType = double> void std::normal_distribution<_RealType>::reset () [inline]`

Resets the distribution state.

Definition at line 2146 of file random.h.

4.647.4.8 `template<typename _RealType = double> _RealType std::normal_distribution<_RealType>::stddev () const`
`[inline]`

Returns the standard deviation of the distribution.

Definition at line 2160 of file random.h.

4.647.5 Friends And Related Function Documentation

4.647.5.1 `template<typename _RealType = double> template<typename _RealType1, typename _CharT, typename _Traits>
> std::basic_ostream<_CharT, _Traits>& operator<< (std::basic_ostream<_CharT, _Traits> & __os, const
std::normal_distribution<_RealType1> & __x) [friend]`

Inserts a `normal_distribution` random number distribution `__x` into the output stream `__os`.

Parameters

<code>__os</code>	An output stream.
<code>__x</code>	A <code>normal_distribution</code> random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

4.647.5.2 `template<typename _RealType = double> template<typename _RealType1 > bool operator==(const
std::normal_distribution< _RealType1 > & __d1, const std::normal_distribution< _RealType1 > & __d2)
[friend]`

Return true if two normal distributions have the same parameters and the sequences that would be generated are equal.

4.647.5.3 `template<typename _RealType = double> template<typename _RealType1 , typename _CharT , typename
_Traits > std::basic_istream< _CharT, _Traits>& operator>> (std::basic_istream< _CharT, _Traits > & __is,
std::normal_distribution< _RealType1 > & __x) [friend]`

Extracts a `normal_distribution` random number distribution `__x` from the input stream `__is`.

Parameters

<code>__is</code>	An input stream.
<code>__x</code>	A <code>normal_distribution</code> random number generator engine.

Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following file:

- [random.h](#)

4.648 `std::normal_distribution< _RealType >::param_type` Struct Reference

Public Types

- typedef `normal_distribution< _RealType >` **distribution_type**

Public Member Functions

- **param_type** (`_RealType __mean=_RealType(0), _RealType __stddev=_RealType(1)`)
- `_RealType` **mean** () const
- `_RealType` **stddev** () const

Friends

- bool **operator==** (const [param_type](#) &__p1, const [param_type](#) &__p2)

4.648.1 Detailed Description

```
template<typename _RealType = double>
struct std::normal_distribution< _RealType >::param_type
```

Parameter type.

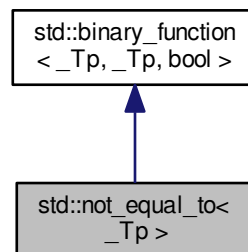
Definition at line 2096 of file random.h.

The documentation for this struct was generated from the following file:

- [random.h](#)

4.649 std::not_equal_to<_Tp> Struct Template Reference

Inheritance diagram for std::not_equal_to<_Tp>:



Public Types

- typedef `_Tp` [first_argument_type](#)
- typedef bool [result_type](#)
- typedef `_Tp` [second_argument_type](#)

Public Member Functions

- bool **operator()** (const `_Tp` &__x, const `_Tp` &__y) const

4.649.1 Detailed Description

```
template<typename _Tp>
struct std::not_equal_to<_Tp>
```

One of the [comparison functors](#).

Definition at line 349 of file stl_function.h.

4.649.2 Member Typedef Documentation

4.649.2.1 `typedef _Tp std::binary_function<_Tp, _Tp, bool>::first_argument_type` [\[inherited\]](#)

`first_argument_type` is the type of the first argument

Definition at line 121 of file stl_function.h.

4.649.2.2 `typedef bool std::binary_function<_Tp, _Tp, bool>::result_type` [\[inherited\]](#)

`result_type` is the return type

Definition at line 127 of file stl_function.h.

4.649.2.3 `typedef _Tp std::binary_function<_Tp, _Tp, bool>::second_argument_type` [\[inherited\]](#)

`second_argument_type` is the type of the second argument

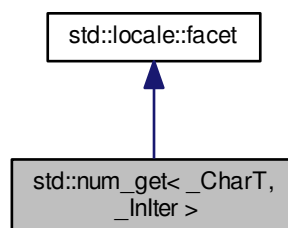
Definition at line 124 of file stl_function.h.

The documentation for this struct was generated from the following file:

- [stl_function.h](#)

4.650 `std::num_get<_CharT, _InIter>` Class Template Reference

Inheritance diagram for `std::num_get<_CharT, _InIter>`:



Public Types

- typedef `_CharT` `char_type`
- typedef `_InIter` `iter_type`

Public Member Functions

- `num_get` (`size_t` __refs=0)
- `iter_type get` (`iter_type` __in, `iter_type` __end, `ios_base` &__io, `ios_base::iostate` &__err, `bool` &__v) const
- `iter_type get` (`iter_type` __in, `iter_type` __end, `ios_base` &__io, `ios_base::iostate` &__err, `void *&`__v) const
- `iter_type get` (`iter_type` __in, `iter_type` __end, `ios_base` &__io, `ios_base::iostate` &__err, `long` &__v) const
- `iter_type get` (`iter_type` __in, `iter_type` __end, `ios_base` &__io, `ios_base::iostate` &__err, `unsigned short` &__v) const
- `iter_type get` (`iter_type` __in, `iter_type` __end, `ios_base` &__io, `ios_base::iostate` &__err, `unsigned int` &__v) const
- `iter_type get` (`iter_type` __in, `iter_type` __end, `ios_base` &__io, `ios_base::iostate` &__err, `unsigned long` &__v) const
- `iter_type get` (`iter_type` __in, `iter_type` __end, `ios_base` &__io, `ios_base::iostate` &__err, `long long` &__v) const
- `iter_type get` (`iter_type` __in, `iter_type` __end, `ios_base` &__io, `ios_base::iostate` &__err, `unsigned long long` &__v) const
- `iter_type get` (`iter_type` __in, `iter_type` __end, `ios_base` &__io, `ios_base::iostate` &__err, `float` &__v) const
- `iter_type get` (`iter_type` __in, `iter_type` __end, `ios_base` &__io, `ios_base::iostate` &__err, `double` &__v) const
- `iter_type get` (`iter_type` __in, `iter_type` __end, `ios_base` &__io, `ios_base::iostate` &__err, `long double` &__v) const

Static Public Attributes

- static `locale::id` `id`

Protected Member Functions

- virtual `~num_get` ()
- `iter_type _M_extract_float` (`iter_type`, `iter_type`, `ios_base` &, `ios_base::iostate` &, `string` &) const
- `template<typename _ValueT>`
`iter_type _M_extract_int` (`iter_type`, `iter_type`, `ios_base` &, `ios_base::iostate` &, `_ValueT` &) const
- `template<typename _CharT2>`
`__gnu_cxx::__enable_if<__is_char<_CharT2>::__value, int>::__type _M_find` (`const _CharT2 *`, `size_t` __len, `_CharT2` __c) const
- `template<typename _CharT2>`
`__gnu_cxx::__enable_if<!__is_char<_CharT2>::__value, int>::__type _M_find` (`const _CharT2 *` __zero, `size_t` __len, `_CharT2` __c) const
- virtual `iter_type do_get` (`iter_type`, `iter_type`, `ios_base` &, `ios_base::iostate` &, `bool` &) const
- virtual `iter_type do_get` (`iter_type` __beg, `iter_type` __end, `ios_base` &__io, `ios_base::iostate` &__err, `long` &__v) const
- virtual `iter_type do_get` (`iter_type` __beg, `iter_type` __end, `ios_base` &__io, `ios_base::iostate` &__err, `unsigned short` &__v) const

- virtual `iter_type do_get (iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, unsigned int & __v) const`
- virtual `iter_type do_get (iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, unsigned long & __v) const`
- virtual `iter_type do_get (iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, long long & __v) const`
- virtual `iter_type do_get (iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, unsigned long long & __v) const`
- virtual `iter_type do_get (iter_type, iter_type, ios_base &, ios_base::iostate &, float &) const`
- virtual `iter_type do_get (iter_type, iter_type, ios_base &, ios_base::iostate &, double &) const`
- virtual `iter_type do_get (iter_type, iter_type, ios_base &, ios_base::iostate &, long double &) const`
- virtual `iter_type do_get (iter_type, iter_type, ios_base &, ios_base::iostate &, void *&) const`

Static Protected Member Functions

- static `__c_locale _S_clone_c_locale (__c_locale & __cloc) throw ()`
- static void `_S_create_c_locale (__c_locale & __cloc, const char * __s, __c_locale __old=0)`
- static void `_S_destroy_c_locale (__c_locale & __cloc)`
- static `__c_locale _S_get_c_locale ()`
- static const char * `_S_get_c_name () throw ()`
- static `__c_locale _S_lc_ctype_c_locale (__c_locale __cloc, const char * __s)`

4.650.1 Detailed Description

```
template<typename _CharT, typename _InIter>
class std::num_get< _CharT, _InIter >
```

Primary class template `num_get`.

This facet encapsulates the code to parse and return a number from a string. It is used by the `istream` numeric extraction operators.

The `num_get` template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the `num_get` facet.

Definition at line 1915 of file `locale_facets.h`.

4.650.2 Member Typedef Documentation

4.650.2.1 `template<typename _CharT, typename _InIter> typedef _CharT std::num_get< _CharT, _InIter >::char_type`

Public typedefs.

Definition at line 1921 of file `locale_facets.h`.

4.650.2.2 `template<typename _CharT, typename _InIter > typedef _InIter std::num_get<_CharT, _InIter >::iter_type`

Public typedefs.

Definition at line 1922 of file locale_facets.h.

4.650.3 Constructor & Destructor Documentation

4.650.3.1 `template<typename _CharT, typename _InIter > std::num_get<_CharT, _InIter >::num_get (size_t __refs = 0)`
`[inline], [explicit]`

Constructor performs initialization.

This is the constructor provided by the standard.

Parameters

<code>__refs</code>	Passed to the base facet class.
---------------------	---------------------------------

Definition at line 1936 of file locale_facets.h.

4.650.3.2 `template<typename _CharT, typename _InIter > virtual std::num_get<_CharT, _InIter >::~~num_get ()`
`[inline], [protected], [virtual]`

Destructor.

Definition at line 2108 of file locale_facets.h.

4.650.4 Member Function Documentation

4.650.4.1 `template<typename _CharT, typename _InIter > virtual iter_type std::num_get<_CharT, _InIter >::do_get (`
`iter_type, iter_type, ios_base &, ios_base::iostate &, bool &) const` `[protected], [virtual]`

Numeric parsing.

Parses the input stream into the variable `v`. This function is a hook for derived classes to change the value returned.

See also

`get()` for more details.

Parameters

<code>__beg</code>	Start of input stream.
<code>__end</code>	End of input stream.
<code>__io</code>	Source of locale and flags.
<code>__err</code>	Error flags to set.
<code>__v</code>	Value to format and insert.

Returns

Iterator after reading.

```
4.650.4.2  template<typename _CharT, typename _InIter > virtual iter_type std::num_get<_CharT, _InIter >::do_get (
            iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, long & __v ) const    [inline],
            [protected], [virtual]
```

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

See also

`get()` for more details.

Parameters

<code>__beg</code>	Start of input stream.
<code>__end</code>	End of input stream.
<code>__io</code>	Source of locale and flags.
<code>__err</code>	Error flags to set.
<code>__v</code>	Value to format and insert.

Returns

Iterator after reading.

Definition at line 2176 of file `locale_facets.h`.

```
4.650.4.3  template<typename _CharT, typename _InIter > virtual iter_type std::num_get<_CharT, _InIter >::do_get (
            iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, unsigned short & __v ) const
            [inline], [protected], [virtual]
```

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

See also

`get()` for more details.

Parameters

<code>__beg</code>	Start of input stream.
<code>__end</code>	End of input stream.
<code>__io</code>	Source of locale and flags.
<code>__err</code>	Error flags to set.
<code>__v</code>	Value to format and insert.

Returns

Iterator after reading.

Definition at line 2181 of file locale_facets.h.

```
4.650.4.4 template<typename _CharT, typename _InIter > virtual iter_type std::num_get<_CharT, _InIter >::do_get (
    iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, unsigned int & __v ) const
    [inline], [protected], [virtual]
```

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

See also

get() for more details.

Parameters

<code>__beg</code>	Start of input stream.
<code>__end</code>	End of input stream.
<code>__io</code>	Source of locale and flags.
<code>__err</code>	Error flags to set.
<code>__v</code>	Value to format and insert.

Returns

Iterator after reading.

Definition at line 2186 of file locale_facets.h.

```
4.650.4.5 template<typename _CharT, typename _InIter > virtual iter_type std::num_get<_CharT, _InIter >::do_get (
    iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, unsigned long & __v ) const
    [inline], [protected], [virtual]
```

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

See also

get() for more details.

Parameters

<code>__beg</code>	Start of input stream.
<code>__end</code>	End of input stream.
<code>__io</code>	Source of locale and flags.
<code>__err</code>	Error flags to set.
<code>__v</code>	Value to format and insert.

Returns

Iterator after reading.

Definition at line 2191 of file locale_facets.h.

```
4.650.4.6  template<typename _CharT, typename _InIter > virtual iter_type std::num_get< _CharT, _InIter >::do_get (
            iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, long long & __v ) const
            [inline], [protected], [virtual]
```

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

See also

get() for more details.

Parameters

<code>__beg</code>	Start of input stream.
<code>__end</code>	End of input stream.
<code>__io</code>	Source of locale and flags.
<code>__err</code>	Error flags to set.
<code>__v</code>	Value to format and insert.

Returns

Iterator after reading.

Definition at line 2197 of file locale_facets.h.

```
4.650.4.7  template<typename _CharT, typename _InIter > virtual iter_type std::num_get< _CharT, _InIter >::do_get (
            iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, unsigned long long & __v ) const
            [inline], [protected], [virtual]
```

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

See also

get() for more details.

Parameters

<code>__beg</code>	Start of input stream.
<code>__end</code>	End of input stream.
<code>__io</code>	Source of locale and flags.
<code>__err</code>	Error flags to set.
<code>__v</code>	Value to format and insert.

Returns

Iterator after reading.

Definition at line 2202 of file locale_facets.h.

```
4.650.4.8 template<typename _CharT, typename _InIter > virtual iter_type std::num_get<_CharT, _InIter >::do_get (
    iter_type, iter_type, ios_base &, ios_base::iostate &, float & ) const [protected], [virtual]
```

Numeric parsing.

Parses the input stream into the variable v. This function is a hook for derived classes to change the value returned.

See also

get() for more details.

Parameters

<code>__beg</code>	Start of input stream.
<code>__end</code>	End of input stream.
<code>__io</code>	Source of locale and flags.
<code>__err</code>	Error flags to set.
<code>__v</code>	Value to format and insert.

Returns

Iterator after reading.

```
4.650.4.9 template<typename _CharT, typename _InIter > virtual iter_type std::num_get<_CharT, _InIter >::do_get (
    iter_type, iter_type, ios_base &, ios_base::iostate &, double & ) const [protected], [virtual]
```

Numeric parsing.

Parses the input stream into the variable v. This function is a hook for derived classes to change the value returned.

See also

get() for more details.

Parameters

<code>__beg</code>	Start of input stream.
<code>__end</code>	End of input stream.
<code>__io</code>	Source of locale and flags.
<code>__err</code>	Error flags to set.
<code>__v</code>	Value to format and insert.

Returns

Iterator after reading.

```
4.650.4.10  template<typename _CharT, typename _InIter > virtual iter_type std::num_get< _CharT, _InIter >::do_get (
            iter_type, iter_type, ios_base &, ios_base::iostate &, long double & ) const  [protected],
            [virtual]
```

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

See also

`get()` for more details.

Parameters

<code>__beg</code>	Start of input stream.
<code>__end</code>	End of input stream.
<code>__io</code>	Source of locale and flags.
<code>__err</code>	Error flags to set.
<code>__v</code>	Value to format and insert.

Returns

Iterator after reading.

```
4.650.4.11  template<typename _CharT, typename _InIter > virtual iter_type std::num_get< _CharT, _InIter >::do_get (
            iter_type, iter_type, ios_base &, ios_base::iostate &, void *& ) const  [protected], [virtual]
```

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

See also

`get()` for more details.

Parameters

<code>__beg</code>	Start of input stream.
<code>__end</code>	End of input stream.
<code>__io</code>	Source of locale and flags.
<code>__err</code>	Error flags to set.
<code>__v</code>	Value to format and insert.

Returns

Iterator after reading.

4.650.4.12 `template<typename _CharT, typename _InIter> iter_type std::num_get<_CharT, _InIter>::get(iter_type __in, iter_type __end, ios_base & __io, ios_base::iostate & __err, bool & __v) const [inline]`

Numeric parsing.

Parses the input stream into the bool *v*. It does so by calling `num_get::do_get()`.

If `ios_base::boolalpha` is set, attempts to read `ctype<CharT>::truenamename()` or `ctype<CharT>::falsename()`. Sets *v* to true or false if successful. Sets *err* to `ios_base::failbit` if reading the string fails. Sets *err* to `ios_base::eofbit` if the stream is emptied.

If `ios_base::boolalpha` is not set, proceeds as with reading a long, except if the value is 1, sets *v* to true, if the value is 0, sets *v* to false, and otherwise set *err* to `ios_base::failbit`.

Parameters

<code>__in</code>	Start of input stream.
<code>__end</code>	End of input stream.
<code>__io</code>	Source of locale and flags.
<code>__err</code>	Error flags to set.
<code>__v</code>	Value to format and insert.

Returns

Iterator after reading.

Definition at line 1962 of file `locale_facets.h`.

4.650.4.13 `template<typename _CharT, typename _InIter> iter_type std::num_get<_CharT, _InIter>::get(iter_type __in, iter_type __end, ios_base & __io, ios_base::iostate & __err, long & __v) const [inline]`

Numeric parsing.

Parses the input stream into the integral variable *v*. It does so by calling `num_get::do_get()`.

Parsing is affected by the flag settings in *io*.

The basic parse is affected by the value of `io.flags()` & `ios_base::basefield`. If equal to `ios_base::oct`, parses like the `scanf` `o` specifier. Else if equal to `ios_base::hex`, parses like `X` specifier. Else if `basefield` equal to 0, parses like the `i` specifier. Otherwise, parses like `d` for signed and `u` for unsigned types. The matching type length modifier is also used.

Digit grouping is interpreted according to `num_punct::grouping()` and `num_punct::thousands_sep()`. If the pattern of digit groups isn't consistent, sets *err* to `ios_base::failbit`.

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets *err* to `ios_base::failbit` and leaves *v* unaltered. Sets *err* to `ios_base::eofbit` if the stream is emptied.

Parameters

<code>__in</code>	Start of input stream.
<code>__end</code>	End of input stream.
<code>__io</code>	Source of locale and flags.
<code>__err</code>	Error flags to set.
<code>__v</code>	Value to format and insert.

Returns

Iterator after reading.

Definition at line 1999 of file `locale_facets.h`.

```
4.650.4.14  template<typename _CharT, typename _InIter > iter_type std::num_get< _CharT, _InIter >::get ( iter_type __in,
            iter_type __end, ios_base & __io, ios_base::iostate & __err, unsigned short & __v ) const  [inline]
```

Numeric parsing.

Parses the input stream into the integral variable `v`. It does so by calling `num_get::do_get()`.

Parsing is affected by the flag settings in `io`.

The basic parse is affected by the value of `io.flags()` & `ios_base::basefield`. If equal to `ios_base::oct`, parses like the `scanf o` specifier. Else if equal to `ios_base::hex`, parses like `X` specifier. Else if `basefield` equal to 0, parses like the `i` specifier. Otherwise, parses like `d` for signed and `u` for unsigned types. The matching type length modifier is also used.

Digit grouping is interpreted according to `numpunct::grouping()` and `numpunct::thousands_sep()`. If the pattern of digit groups isn't consistent, sets `err` to `ios_base::failbit`.

If parsing the string yields a valid value for `v`, `v` is set. Otherwise, sets `err` to `ios_base::failbit` and leaves `v` unaltered. Sets `err` to `ios_base::eofbit` if the stream is emptied.

Parameters

<code>__in</code>	Start of input stream.
<code>__end</code>	End of input stream.
<code>__io</code>	Source of locale and flags.
<code>__err</code>	Error flags to set.
<code>__v</code>	Value to format and insert.

Returns

Iterator after reading.

Definition at line 2004 of file `locale_facets.h`.

4.650.4.15 `template<typename _CharT, typename _InIter> iter_type std::num_get<_CharT, _InIter>::get(iter_type __in, iter_type __end, ios_base & __io, ios_base::iostate & __err, unsigned int & __v) const [inline]`

Numeric parsing.

Parses the input stream into the integral variable *v*. It does so by calling `num_get::do_get()`.

Parsing is affected by the flag settings in *io*.

The basic parse is affected by the value of `io.flags()` & `ios_base::basefield`. If equal to `ios_base::oct`, parses like the `scanf o` specifier. Else if equal to `ios_base::hex`, parses like `X` specifier. Else if `basefield` equal to 0, parses like the `i` specifier. Otherwise, parses like `d` for signed and `u` for unsigned types. The matching type length modifier is also used.

Digit grouping is interpreted according to `numpunct::grouping()` and `numpunct::thousands_sep()`. If the pattern of digit groups isn't consistent, sets *err* to `ios_base::failbit`.

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets *err* to `ios_base::failbit` and leaves *v* unaltered. Sets *err* to `ios_base::eofbit` if the stream is emptied.

Parameters

<code>__in</code>	Start of input stream.
<code>__end</code>	End of input stream.
<code>__io</code>	Source of locale and flags.
<code>__err</code>	Error flags to set.
<code>__v</code>	Value to format and insert.

Returns

Iterator after reading.

Definition at line 2009 of file `locale_facets.h`.

4.650.4.16 `template<typename _CharT, typename _InIter> iter_type std::num_get<_CharT, _InIter>::get(iter_type __in, iter_type __end, ios_base & __io, ios_base::iostate & __err, unsigned long & __v) const [inline]`

Numeric parsing.

Parses the input stream into the integral variable *v*. It does so by calling `num_get::do_get()`.

Parsing is affected by the flag settings in *io*.

The basic parse is affected by the value of `io.flags()` & `ios_base::basefield`. If equal to `ios_base::oct`, parses like the `scanf o` specifier. Else if equal to `ios_base::hex`, parses like `X` specifier. Else if `basefield` equal to 0, parses like the `i` specifier. Otherwise, parses like `d` for signed and `u` for unsigned types. The matching type length modifier is also used.

Digit grouping is interpreted according to `numpunct::grouping()` and `numpunct::thousands_sep()`. If the pattern of digit groups isn't consistent, sets *err* to `ios_base::failbit`.

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets *err* to `ios_base::failbit` and leaves *v* unaltered. Sets *err* to `ios_base::eofbit` if the stream is emptied.

Parameters

<code>__in</code>	Start of input stream.
<code>__end</code>	End of input stream.
<code>__io</code>	Source of locale and flags.
<code>__err</code>	Error flags to set.
<code>__v</code>	Value to format and insert.

Returns

Iterator after reading.

Definition at line 2014 of file `locale_facets.h`.

```
4.650.4.17 template<typename _CharT, typename _InIter > iter_type std::num_get<_CharT, _InIter >::get ( iter_type __in,
iter_type __end, ios_base & __io, ios_base::iostate & __err, long long & __v ) const [inline]
```

Numeric parsing.

Parses the input stream into the integral variable `v`. It does so by calling `num_get::do_get()`.

Parsing is affected by the flag settings in `io`.

The basic parse is affected by the value of `io.flags()` & `ios_base::basefield`. If equal to `ios_base::oct`, parses like the `scanf o` specifier. Else if equal to `ios_base::hex`, parses like `X` specifier. Else if `basefield` equal to 0, parses like the `i` specifier. Otherwise, parses like `d` for signed and `u` for unsigned types. The matching type length modifier is also used.

Digit grouping is interpreted according to `numpunct::grouping()` and `numpunct::thousands_sep()`. If the pattern of digit groups isn't consistent, sets `err` to `ios_base::failbit`.

If parsing the string yields a valid value for `v`, `v` is set. Otherwise, sets `err` to `ios_base::failbit` and leaves `v` unaltered. Sets `err` to `ios_base::eofbit` if the stream is emptied.

Parameters

<code>__in</code>	Start of input stream.
<code>__end</code>	End of input stream.
<code>__io</code>	Source of locale and flags.
<code>__err</code>	Error flags to set.
<code>__v</code>	Value to format and insert.

Returns

Iterator after reading.

Definition at line 2020 of file `locale_facets.h`.

4.650.4.18 `template<typename _CharT, typename _InIter> iter_type std::num_get<_CharT, _InIter>::get (iter_type __in, iter_type __end, ios_base & __io, ios_base::iostate & __err, unsigned long long & __v) const [inline]`

Numeric parsing.

Parses the input stream into the integral variable *v*. It does so by calling `num_get::do_get()`.

Parsing is affected by the flag settings in *io*.

The basic parse is affected by the value of `io.flags()` & `ios_base::basefield`. If equal to `ios_base::oct`, parses like the `scanf` `o` specifier. Else if equal to `ios_base::hex`, parses like `X` specifier. Else if `basefield` equal to 0, parses like the `i` specifier. Otherwise, parses like `d` for signed and `u` for unsigned types. The matching type length modifier is also used.

Digit grouping is interpreted according to `num_punct::grouping()` and `num_punct::thousands_sep()`. If the pattern of digit groups isn't consistent, sets `err` to `ios_base::failbit`.

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets `err` to `ios_base::failbit` and leaves *v* unaltered. Sets `err` to `ios_base::eofbit` if the stream is emptied.

Parameters

<code>__in</code>	Start of input stream.
<code>__end</code>	End of input stream.
<code>__io</code>	Source of locale and flags.
<code>__err</code>	Error flags to set.
<code>__v</code>	Value to format and insert.

Returns

Iterator after reading.

Definition at line 2025 of file `locale_facets.h`.

4.650.4.19 `template<typename _CharT, typename _InIter> iter_type std::num_get<_CharT, _InIter>::get (iter_type __in, iter_type __end, ios_base & __io, ios_base::iostate & __err, float & __v) const [inline]`

Numeric parsing.

Parses the input stream into the integral variable *v*. It does so by calling `num_get::do_get()`.

The input characters are parsed like the `scanf` `g` specifier. The matching type length modifier is also used.

The decimal point character used is `num_punct::decimal_point()`. Digit grouping is interpreted according to `num_punct::grouping()` and `num_punct::thousands_sep()`. If the pattern of digit groups isn't consistent, sets `err` to `ios_base::failbit`.

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets `err` to `ios_base::failbit` and leaves *v* unaltered. Sets `err` to `ios_base::eofbit` if the stream is emptied.

Parameters

<code>__in</code>	Start of input stream.
<code>__end</code>	End of input stream.
<code>__io</code>	Source of locale and flags.
<code>__err</code>	Error flags to set.
<code>__v</code>	Value to format and insert.

Returns

Iterator after reading.

Definition at line 2059 of file `locale_facets.h`.

```
4.650.4.20 template<typename _CharT, typename _InIter > iter_type std::num_get<_CharT, _InIter >::get( iter_type __in,
iter_type __end, ios_base & __io, ios_base::iostate & __err, double & __v ) const [inline]
```

Numeric parsing.

Parses the input stream into the integral variable `v`. It does so by calling `num_get::do_get()`.

The input characters are parsed like the `scanf g` specifier. The matching type length modifier is also used.

The decimal point character used is `num_punct::decimal_point()`. Digit grouping is interpreted according to `num_punct::grouping()` and `num_punct::thousands_sep()`. If the pattern of digit groups isn't consistent, sets `err` to `ios_base::failbit`.

If parsing the string yields a valid value for `v`, `v` is set. Otherwise, sets `err` to `ios_base::failbit` and leaves `v` unaltered. Sets `err` to `ios_base::eofbit` if the stream is emptied.

Parameters

<code>__in</code>	Start of input stream.
<code>__end</code>	End of input stream.
<code>__io</code>	Source of locale and flags.
<code>__err</code>	Error flags to set.
<code>__v</code>	Value to format and insert.

Returns

Iterator after reading.

Definition at line 2064 of file `locale_facets.h`.

```
4.650.4.21 template<typename _CharT, typename _InIter > iter_type std::num_get<_CharT, _InIter >::get( iter_type __in,
iter_type __end, ios_base & __io, ios_base::iostate & __err, long double & __v ) const [inline]
```

Numeric parsing.

Parses the input stream into the integral variable *v*. It does so by calling num_get::do_get().

The input characters are parsed like the scanf g specifier. The matching type length modifier is also used.

The decimal point character used is numpunct::decimal_point(). Digit grouping is interpreted according to numpunct::grouping() and numpunct::thousands_sep(). If the pattern of digit groups isn't consistent, sets err to ios_base::failbit.

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets err to ios_base::failbit and leaves *v* unaltered. Sets err to ios_base::eofbit if the stream is emptied.

Parameters

<code>__in</code>	Start of input stream.
<code>__end</code>	End of input stream.
<code>__io</code>	Source of locale and flags.
<code>__err</code>	Error flags to set.
<code>__v</code>	Value to format and insert.

Returns

Iterator after reading.

Definition at line 2069 of file locale_facets.h.

```
4.650.4.22 template<typename _CharT, typename _InIter > iter_type std::num_get<_CharT, _InIter >::get( iter_type __in,
iter_type __end, ios_base & __io, ios_base::iostate & __err, void *& __v ) const [inline]
```

Numeric parsing.

Parses the input stream into the pointer variable *v*. It does so by calling num_get::do_get().

The input characters are parsed like the scanf p specifier.

Digit grouping is interpreted according to numpunct::grouping() and numpunct::thousands_sep(). If the pattern of digit groups isn't consistent, sets err to ios_base::failbit.

Note that the digit grouping effect for pointers is a bit ambiguous in the standard and shouldn't be relied on. See DR 344.

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets err to ios_base::failbit and leaves *v* unaltered. Sets err to ios_base::eofbit if the stream is emptied.

Parameters

<code>__in</code>	Start of input stream.
<code>__end</code>	End of input stream.
<code>__io</code>	Source of locale and flags.
<code>__err</code>	Error flags to set.
<code>__v</code>	Value to format and insert.

Returns

Iterator after reading.

Definition at line 2102 of file locale_facets.h.

4.650.5 Member Data Documentation

4.650.5.1 `template<typename _CharT, typename _InIter> locale::id std::num_get<_CharT, _InIter>::id` `[static]`

Numpunct facet id.

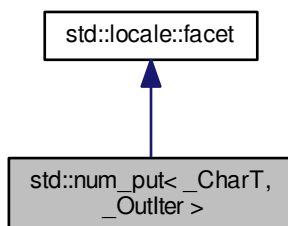
Definition at line 1926 of file locale_facets.h.

The documentation for this class was generated from the following file:

- [locale_facets.h](#)

4.651 std::num_put<_CharT, _OutIter> Class Template Reference

Inheritance diagram for `std::num_put<_CharT, _OutIter>`:

**Public Types**

- `typedef _CharT` [char_type](#)
- `typedef _OutIter` [iter_type](#)

Public Member Functions

- `num_put` (size_t __refs=0)
- `iter_type put` (iter_type __s, ios_base & __io, char_type __fill, bool __v) const
- `iter_type put` (iter_type __s, ios_base & __io, char_type __fill, const void *__v) const
- `iter_type put` (iter_type __s, ios_base & __io, char_type __fill, long __v) const
- `iter_type put` (iter_type __s, ios_base & __io, char_type __fill, unsigned long __v) const
- `iter_type put` (iter_type __s, ios_base & __io, char_type __fill, long long __v) const
- `iter_type put` (iter_type __s, ios_base & __io, char_type __fill, unsigned long long __v) const
- `iter_type put` (iter_type __s, ios_base & __io, char_type __fill, double __v) const
- `iter_type put` (iter_type __s, ios_base & __io, char_type __fill, long double __v) const

Static Public Attributes

- static `locale::id` id

Protected Member Functions

- virtual `~num_put` ()
- void `_M_group_float` (const char *__grouping, size_t __grouping_size, char_type __sep, const char_type *__p, char_type *__new, char_type *__cs, int & __len) const
- void `_M_group_int` (const char *__grouping, size_t __grouping_size, char_type __sep, ios_base & __io, char_type *__new, char_type *__cs, int & __len) const
- template<typename _ValueT>
`iter_type _M_insert_float` (iter_type, ios_base & __io, char_type __fill, char __mod, _ValueT __v) const
- template<typename _ValueT>
`iter_type _M_insert_int` (iter_type, ios_base & __io, char_type __fill, _ValueT __v) const
- void `_M_pad` (char_type __fill, streamsize __w, ios_base & __io, char_type *__new, const char_type *__cs, int & __len) const
- virtual `iter_type do_put` (iter_type __s, ios_base & __io, char_type __fill, bool __v) const
- virtual `iter_type do_put` (iter_type __s, ios_base & __io, char_type __fill, long __v) const
- virtual `iter_type do_put` (iter_type __s, ios_base & __io, char_type __fill, unsigned long __v) const
- virtual `iter_type do_put` (iter_type __s, ios_base & __io, char_type __fill, long long __v) const
- virtual `iter_type do_put` (iter_type __s, ios_base & __io, char_type __fill, unsigned long long __v) const
- virtual `iter_type do_put` (iter_type, ios_base & __io, char_type __fill, double) const
- virtual `iter_type do_put` (iter_type, ios_base & __io, char_type __fill, long double) const
- virtual `iter_type do_put` (iter_type, ios_base & __io, char_type __fill, const void *) const

Static Protected Member Functions

- static `_c_locale _S_clone_c_locale` (_c_locale & __cloc) throw ()
- static void `_S_create_c_locale` (_c_locale & __cloc, const char *__s, _c_locale __old=0)
- static void `_S_destroy_c_locale` (_c_locale & __cloc)
- static `_c_locale _S_get_c_locale` ()
- static const char * `_S_get_c_name` () throw ()
- static `_c_locale _S_lc_ctype_c_locale` (_c_locale __cloc, const char *__s)

4.651.1 Detailed Description

```
template<typename _CharT, typename _Outiter>
class std::num_put< _CharT, _Outiter >
```

Primary class template num_put.

This facet encapsulates the code to convert a number to a string. It is used by the ostream numeric insertion operators.

The num_put template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the num_put facet.

Definition at line 2254 of file locale_facets.h.

4.651.2 Member Typedef Documentation

4.651.2.1 `template<typename _CharT, typename _Outiter > typedef _CharT std::num_put< _CharT, _Outiter >::char_type`

Public typedefs.

Definition at line 2260 of file locale_facets.h.

4.651.2.2 `template<typename _CharT, typename _Outiter > typedef _Outiter std::num_put< _CharT, _Outiter >::iter_type`

Public typedefs.

Definition at line 2261 of file locale_facets.h.

4.651.3 Constructor & Destructor Documentation

4.651.3.1 `template<typename _CharT, typename _Outiter > std::num_put< _CharT, _Outiter >::num_put (size_t __refs = 0) [inline], [explicit]`

Constructor performs initialization.

This is the constructor provided by the standard.

Parameters

<code>__refs</code>	Passed to the base facet class.
---------------------	---------------------------------

Definition at line 2275 of file locale_facets.h.

4.651.3.2 `template<typename _CharT, typename _Outiter> virtual std::num_put<_CharT, _Outiter>::~~num_put ()`
`[inline], [protected], [virtual]`

Destructor.

Definition at line 2454 of file locale_facets.h.

4.651.4 Member Function Documentation

4.651.4.1 `template<typename _CharT, typename _Outiter> virtual iter_type std::num_put<_CharT, _Outiter>::do_put (`
`iter_type __s, ios_base & __io, char_type __fill, bool __v) const` `[protected], [virtual]`

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

Parameters

<code>__s</code>	Stream to write to.
<code>__io</code>	Source of locale and flags.
<code>__fill</code>	Char_type to use for filling.
<code>__v</code>	Value to format and insert.

Returns

Iterator after writing.

4.651.4.2 `template<typename _CharT, typename _Outiter> virtual iter_type std::num_put<_CharT, _Outiter>::do_put`
`(iter_type __s, ios_base & __io, char_type __fill, long __v) const` `[inline], [protected],`
`[virtual]`

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

Parameters

<code>__s</code>	Stream to write to.
<code>__io</code>	Source of locale and flags.
<code>__fill</code>	Char_type to use for filling.
<code>__v</code>	Value to format and insert.

Returns

Iterator after writing.

Definition at line 2474 of file locale_facets.h.

```
4.651.4.3  template<typename _CharT, typename _Outiter > virtual iter_type std::num_put<_CharT, _Outiter >::do_put (
            iter_type __s, ios_base & __io, char_type __fill, unsigned long __v ) const    [inline], [protected],
            [virtual]
```

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

Parameters

<code>__s</code>	Stream to write to.
<code>__io</code>	Source of locale and flags.
<code>__fill</code>	Char_type to use for filling.
<code>__v</code>	Value to format and insert.

Returns

Iterator after writing.

Definition at line 2478 of file locale_facets.h.

```
4.651.4.4  template<typename _CharT, typename _Outiter > virtual iter_type std::num_put<_CharT, _Outiter >::do_put
            ( iter_type __s, ios_base & __io, char_type __fill, long long __v ) const    [inline], [protected],
            [virtual]
```

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

Parameters

<code>__s</code>	Stream to write to.
<code>__io</code>	Source of locale and flags.
<code>__fill</code>	Char_type to use for filling.
<code>__v</code>	Value to format and insert.

Returns

Iterator after writing.

Definition at line 2484 of file locale_facets.h.

```
4.651.4.5 template<typename _CharT, typename _Outiter > virtual iter_type std::num_put<_CharT, _Outiter >::do_put (
    iter_type __s, ios_base & __io, char_type __fill, unsigned long long __v ) const    [inline], [protected],
    [virtual]
```

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

Parameters

<code>__s</code>	Stream to write to.
<code>__io</code>	Source of locale and flags.
<code>__fill</code>	Char_type to use for filling.
<code>__v</code>	Value to format and insert.

Returns

Iterator after writing.

Definition at line 2489 of file locale_facets.h.

```
4.651.4.6 template<typename _CharT, typename _Outiter > virtual iter_type std::num_put<_CharT, _Outiter >::do_put (
    iter_type, ios_base &, char_type, double ) const    [protected], [virtual]
```

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

Parameters

<code>__s</code>	Stream to write to.
<code>__io</code>	Source of locale and flags.
<code>__fill</code>	Char_type to use for filling.
<code>__v</code>	Value to format and insert.

Returns

Iterator after writing.

4.651.4.7 `template<typename _CharT, typename _Outiter > virtual iter_type std::num_put<_CharT, _Outiter >::do_put (iter_type, ios_base &, char_type, long double) const` `[protected], [virtual]`

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

Parameters

<code>_↔ _s</code>	Stream to write to.
<code>_↔ _io</code>	Source of locale and flags.
<code>_↔ _fill</code>	Char_type to use for filling.
<code>_↔ _v</code>	Value to format and insert.

Returns

Iterator after writing.

4.651.4.8 `template<typename _CharT, typename _Outiter > virtual iter_type std::num_put<_CharT, _Outiter >::do_put (iter_type, ios_base &, char_type, const void *) const` `[protected], [virtual]`

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

Parameters

<code>_↔ _s</code>	Stream to write to.
<code>_↔ _io</code>	Source of locale and flags.
<code>_↔ _fill</code>	Char_type to use for filling.
<code>_↔ _v</code>	Value to format and insert.

Returns

Iterator after writing.

4.651.4.9 `template<typename _CharT, typename _Outlter > iter_type std::num_put<_CharT, _Outlter >::put (iter_type __s, ios_base & __io, char_type __fill, bool __v) const [inline]`

Numeric formatting.

Formats the boolean *v* and inserts it into a stream. It does so by calling `num_put::do_put()`.

If `ios_base::boolalpha` is set, writes `ctype<CharT>::truenamename()` or `ctype<CharT>::falsename()`. Otherwise formats *v* as an int.

Parameters

<code>↔ __s</code>	Stream to write to.
<code>↔ __io</code>	Source of locale and flags.
<code>↔ __fill</code>	Char_type to use for filling.
<code>↔ __v</code>	Value to format and insert.

Returns

Iterator after writing.

Definition at line 2293 of file `locale_facets.h`.

4.651.4.10 `template<typename _CharT, typename _Outlter > iter_type std::num_put<_CharT, _Outlter >::put (iter_type __s, ios_base & __io, char_type __fill, long __v) const [inline]`

Numeric formatting.

Formats the integral value *v* and inserts it into a stream. It does so by calling `num_put::do_put()`.

Formatting is affected by the flag settings in *io*.

The basic format is affected by the value of `io.flags()` & `ios_base::basefield`. If equal to `ios_base::oct`, formats like the `printf o` specifier. Else if equal to `ios_base::hex`, formats like `x` or `X` with `ios_base::uppercase` unset or set respectively. Otherwise, formats like `d`, `ld`, `lld` for signed and `u`, `lu`, `llu` for unsigned values. Note that if both `oct` and `hex` are set, neither will take effect.

If `ios_base::showpos` is set, '+' is output before positive values. If `ios_base::showbase` is set, '0' precedes octal values (except 0) and '0[xX]' precedes hex values.

The decimal point character used is `num_punct::decimal_point()`. Thousands separators are inserted according to `num_punct::grouping()` and `num_punct::thousands_sep()`.

If `io.width()` is non-zero, enough *fill* characters are inserted to make the result at least that wide. If `(io.flags() & ios_base::adjustfield) == ios_base::left`, result is padded at the end. If `ios_base::internal`, then padding occurs immediately after either a '+' or '-' or after '0x' or '0X'. Otherwise, padding occurs at the beginning.

Parameters

<code>__s</code>	Stream to write to.
<code>__io</code>	Source of locale and flags.
<code>__fill</code>	Char_type to use for filling.
<code>__v</code>	Value to format and insert.

Returns

Iterator after writing.

Definition at line 2335 of file locale_facets.h.

```
4.651.4.11 template<typename _CharT, typename _Outiter > iter_type std::num_put<_CharT, _Outiter >::put ( iter_type
__s, ios_base & __io, char_type __fill, unsigned long __v ) const [inline]
```

Numeric formatting.

Formats the integral value `v` and inserts it into a stream. It does so by calling `num_put::do_put()`.

Formatting is affected by the flag settings in `io`.

The basic format is affected by the value of `io.flags()` & `ios_base::basefield`. If equal to `ios_base::oct`, formats like the `printf o` specifier. Else if equal to `ios_base::hex`, formats like `x` or `X` with `ios_base::uppercase` unset or set respectively. Otherwise, formats like `d`, `ld`, `lld` for signed and `u`, `lu`, `llu` for unsigned values. Note that if both `oct` and `hex` are set, neither will take effect.

If `ios_base::showpos` is set, `+` is output before positive values. If `ios_base::showbase` is set, `'0'` precedes octal values (except 0) and `'0[xX]'` precedes hex values.

The decimal point character used is `numput::decimal_point()`. Thousands separators are inserted according to `numput::grouping()` and `numput::thousands_sep()`.

If `io.width()` is non-zero, enough `fill` characters are inserted to make the result at least that wide. If `(io.flags() & ios_base::adjustfield) == ios_base::left`, result is padded at the end. If `ios_base::internal`, then padding occurs immediately after either a `+` or `-` or after `'0x'` or `'0X'`. Otherwise, padding occurs at the beginning.

Parameters

<code>__s</code>	Stream to write to.
<code>__io</code>	Source of locale and flags.
<code>__fill</code>	Char_type to use for filling.
<code>__v</code>	Value to format and insert.

Returns

Iterator after writing.

Definition at line 2339 of file locale_facets.h.

```
4.651.4.12 template<typename _CharT, typename _Outiter > iter_type std::num_put<_CharT, _Outiter >::put ( iter_type
    __s, ios_base & __io, char_type __fill, long long __v ) const [inline]
```

Numeric formatting.

Formats the integral value *v* and inserts it into a stream. It does so by calling num_put::do_put().

Formatting is affected by the flag settings in *io*.

The basic format is affected by the value of io.flags() & ios_base::basefield. If equal to ios_base::oct, formats like the printf o specifier. Else if equal to ios_base::hex, formats like x or X with ios_base::uppercase unset or set respectively. Otherwise, formats like d, ld, lld for signed and u, lu, llu for unsigned values. Note that if both oct and hex are set, neither will take effect.

If ios_base::showpos is set, '+' is output before positive values. If ios_base::showbase is set, '0' precedes octal values (except 0) and '0[xX]' precedes hex values.

The decimal point character used is numpunct::decimal_point(). Thousands separators are inserted according to numpunct::grouping() and numpunct::thousands_sep().

If io.width() is non-zero, enough *fill* characters are inserted to make the result at least that wide. If (io.flags() & ios_base::adjustfield) == ios_base::left, result is padded at the end. If ios_base::internal, then padding occurs immediately after either a '+' or '-' or after '0x' or '0X'. Otherwise, padding occurs at the beginning.

Parameters

<code>__s</code>	Stream to write to.
<code>__io</code>	Source of locale and flags.
<code>__fill</code>	Char_type to use for filling.
<code>__v</code>	Value to format and insert.

Returns

Iterator after writing.

Definition at line 2345 of file locale_facets.h.

```
4.651.4.13 template<typename _CharT, typename _Outiter > iter_type std::num_put<_CharT, _Outiter >::put ( iter_type
    __s, ios_base & __io, char_type __fill, unsigned long long __v ) const [inline]
```

Numeric formatting.

Formats the integral value *v* and inserts it into a stream. It does so by calling `num_put::do_put()`.

Formatting is affected by the flag settings in *io*.

The basic format is affected by the value of `io.flags()` & `ios_base::basefield`. If equal to `ios_base::oct`, formats like the `printf o` specifier. Else if equal to `ios_base::hex`, formats like `x` or `X` with `ios_base::uppercase` unset or set respectively. Otherwise, formats like `d`, `ld`, `lld` for signed and `u`, `lu`, `llu` for unsigned values. Note that if both `oct` and `hex` are set, neither will take effect.

If `ios_base::showpos` is set, `'+'` is output before positive values. If `ios_base::showbase` is set, `'0'` precedes octal values (except 0) and `'0[xX]'` precedes hex values.

The decimal point character used is `numput::decimal_point()`. Thousands separators are inserted according to `numput::grouping()` and `numput::thousands_sep()`.

If `io.width()` is non-zero, enough *fill* characters are inserted to make the result at least that wide. If `(io.flags() & ios_base::adjustfield) == ios_base::left`, result is padded at the end. If `ios_base::internal`, then padding occurs immediately after either a `'+'` or `'-'` or after `'0x'` or `'0X'`. Otherwise, padding occurs at the beginning.

Parameters

<code>__s</code>	Stream to write to.
<code>__io</code>	Source of locale and flags.
<code>__fill</code>	Char_type to use for filling.
<code>__v</code>	Value to format and insert.

Returns

Iterator after writing.

Definition at line 2349 of file `locale_facets.h`.

```
4.651.4.14 template<typename _CharT, typename _OutIter> iter_type std::num_put<_CharT, _OutIter>::put ( iter_type
    __s, ios_base & __io, char_type __fill, double __v ) const [inline]
```

Numeric formatting.

Formats the floating point value *v* and inserts it into a stream. It does so by calling `num_put::do_put()`.

Formatting is affected by the flag settings in *io*.

The basic format is affected by the value of `io.flags()` & `ios_base::floatfield`. If equal to `ios_base::fixed`, formats like the `printf f` specifier. Else if equal to `ios_base::scientific`, formats like `e` or `E` with `ios_base::uppercase` unset or set respectively. Otherwise, formats like `g` or `G` depending on uppercase. Note that if both `fixed` and `scientific` are set, the effect will also be like `g` or `G`.

The output precision is given by `io.precision()`. This precision is capped at `numeric_limits::digits10 + 2` (different for double and long double). The default precision is 6.

If `ios_base::showpos` is set, '+' is output before positive values. If `ios_base::showpoint` is set, a decimal point will always be output.

The decimal point character used is `numpunct::decimal_point()`. Thousands separators are inserted according to `numpunct::grouping()` and `numpunct::thousands_sep()`.

If `io.width()` is non-zero, enough *fill* characters are inserted to make the result at least that wide. If `(io.flags() & ios_base::adjustfield) == ios_base::left`, result is padded at the end. If `ios_base::internal`, then padding occurs immediately after either a '+' or '-' or after '0x' or '0X'. Otherwise, padding occurs at the beginning.

Parameters

<code>__s</code>	Stream to write to.
<code>__io</code>	Source of locale and flags.
<code>__fill</code>	Char_type to use for filling.
<code>__v</code>	Value to format and insert.

Returns

Iterator after writing.

Definition at line 2398 of file `locale_facets.h`.

4.651.4.15 `template<typename _CharT, typename _OutIter> iter_type std::num_put<_CharT, _OutIter>::put (iter_type __s, ios_base & __io, char_type __fill, long double __v) const [inline]`

Numeric formatting.

Formats the floating point value *v* and inserts it into a stream. It does so by calling `num_put::do_put()`.

Formatting is affected by the flag settings in *io*.

The basic format is affected by the value of `io.flags() & ios_base::floatfield`. If equal to `ios_base::fixed`, formats like the printf *f* specifier. Else if equal to `ios_base::scientific`, formats like *e* or *E* with `ios_base::uppercase` unset or set respectively. Otherwise, formats like *g* or *G* depending on uppercase. Note that if both fixed and scientific are set, the effect will also be like *g* or *G*.

The output precision is given by `io.precision()`. This precision is capped at `numeric_limits::digits10 + 2` (different for double and long double). The default precision is 6.

If `ios_base::showpos` is set, '+' is output before positive values. If `ios_base::showpoint` is set, a decimal point will always be output.

The decimal point character used is `numpunct::decimal_point()`. Thousands separators are inserted according to `numpunct::grouping()` and `numpunct::thousands_sep()`.

If `io.width()` is non-zero, enough *fill* characters are inserted to make the result at least that wide. If `(io.flags() & ios_base::adjustfield) == ios_base::left`, result is padded at the end. If `ios_base::internal`, then padding occurs immediately after either a '+' or '-' or after '0x' or '0X'. Otherwise, padding occurs at the beginning.

Parameters

<code>__s</code>	Stream to write to.
<code>__io</code>	Source of locale and flags.
<code>__fill</code>	Char_type to use for filling.
<code>__v</code>	Value to format and insert.

Returns

Iterator after writing.

Definition at line 2402 of file locale_facets.h.

```
4.651.4.16 template<typename _CharT, typename _Outiter > iter_type std::num_put<_CharT, _Outiter >::put ( iter_type
__s, ios_base & __io, char_type __fill, const void *__v ) const [inline]
```

Numeric formatting.

Formats the pointer value *v* and inserts it into a stream. It does so by calling `num_put::do_put()`.

This function formats *v* as an unsigned long with `ios_base::hex` and `ios_base::showbase` set.

Parameters

<code>__s</code>	Stream to write to.
<code>__io</code>	Source of locale and flags.
<code>__fill</code>	Char_type to use for filling.
<code>__v</code>	Value to format and insert.

Returns

Iterator after writing.

Definition at line 2423 of file locale_facets.h.

4.651.5 Member Data Documentation

```
4.651.5.1 template<typename _CharT, typename _Outiter > locale::id std::num_put<_CharT, _Outiter >::id [static]
```

Numpunct facet id.

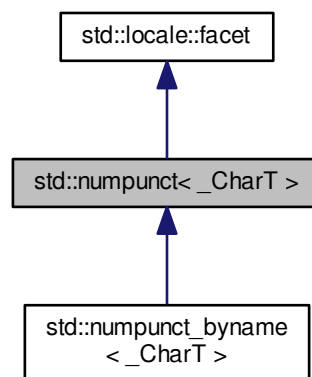
Definition at line 2265 of file locale_facets.h.

The documentation for this class was generated from the following file:

- [locale_facets.h](#)

4.652 std::num_punct<_CharT> Class Template Reference

Inheritance diagram for std::num_punct<_CharT>:



Public Types

- typedef __num_punct_cache<_CharT> **__cache_type**
- typedef _CharT [char_type](#)
- typedef [basic_string](#)<_CharT> [string_type](#)

Public Member Functions

- [num_punct](#) (size_t __refs=0)
- [num_punct](#) (__cache_type *__cache, size_t __refs=0)
- [num_punct](#) (__c_locale __cloc, size_t __refs=0)
- [char_type](#) [decimal_point](#) () const
- [string_type](#) [falsename](#) () const
- [string_type](#) [grouping](#) () const
- [char_type](#) [thousands_sep](#) () const
- [string_type](#) [truenam](#)e () const

Static Public Attributes

- static `locale::id` `id`

Protected Member Functions

- virtual `~numpunct` ()
- void `_M_initialize_numpunct` (__c_locale __cloc=0)
- template<>
void `_M_initialize_numpunct` (__c_locale __cloc)
- template<>
void `_M_initialize_numpunct` (__c_locale __cloc)
- virtual `char_type do_decimal_point` () const
- virtual `string_type do_falsename` () const
- virtual `string do_grouping` () const
- virtual `char_type do_thousands_sep` () const
- virtual `string_type do_truename` () const

Static Protected Member Functions

- static __c_locale `_S_clone_c_locale` (__c_locale &__cloc) throw ()
- static void `_S_create_c_locale` (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void `_S_destroy_c_locale` (__c_locale &__cloc)
- static __c_locale `_S_get_c_locale` ()
- static const char * `_S_get_c_name` () throw ()
- static __c_locale `_S_lc_ctype_c_locale` (__c_locale __cloc, const char *__s)

Protected Attributes

- __cache_type * `_M_data`

4.652.1 Detailed Description

```
template<typename _CharT>
class std::numpunct<_CharT>
```

Primary class template `numpunct`.

This facet stores several pieces of information related to printing and scanning numbers, such as the decimal point character. It takes a template parameter specifying the char type. The `numpunct` facet is used by streams for many I/O operations involving numbers.

The `numpunct` template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from a `numpunct` facet.

Definition at line 1641 of file `locale_facets.h`.

4.652.2 Member Typedef Documentation

4.652.2.1 template<typename _CharT> typedef _CharT std::numpunct<_CharT>::char_type

Public typedefs.

Definition at line 1647 of file locale_facets.h.

4.652.2.2 template<typename _CharT> typedef basic_string<_CharT> std::numpunct<_CharT>::string_type

Public typedefs.

Definition at line 1648 of file locale_facets.h.

4.652.3 Constructor & Destructor Documentation

4.652.3.1 template<typename _CharT> std::numpunct<_CharT>::numpunct (size_t __refs = 0) [inline],
[explicit]

Numpunct constructor.

Parameters

<code>__refs</code>	Refcount to pass to the base class.
---------------------	-------------------------------------

Definition at line 1665 of file locale_facets.h.

4.652.3.2 template<typename _CharT> std::numpunct<_CharT>::numpunct (__cache_type * __cache, size_t __refs = 0) [inline], [explicit]

Internal constructor. Not for general use.

This is a constructor for use by the library itself to set up the predefined locale facets.

Parameters

<code>__cache</code>	<code>__numpunct_cache</code> object.
<code>__refs</code>	Refcount to pass to the base class.

Definition at line 1679 of file locale_facets.h.

4.652.3.3 template<typename _CharT> std::numpunct<_CharT>::numpunct (__c_locale __cloc, size_t __refs = 0) [inline], [explicit]

Internal constructor. Not for general use.

This is a constructor for use by the library itself to set up new locales.

Parameters

<code>__cloc</code>	The C locale.
<code>__refs</code>	Refcount to pass to the base class.

Definition at line 1693 of file `locale_facets.h`.

4.652.3.4 `template<typename _CharT> virtual std::numpunct<_CharT>::~~numpunct ()` `[protected]`,
`[virtual]`

Destructor.

Referenced by `std::numpunct<_CharT>::do_falsename()`.

4.652.4 Member Function Documentation

4.652.4.1 `template<typename _CharT> char_type std::numpunct<_CharT>::decimal_point () const` `[inline]`

Return decimal point character.

This function returns a `char_type` to use as a decimal point. It does so by returning `returning numpunct<char_type>::do_decimal_point()`.

Returns

char_type representing a decimal point.

Definition at line 1707 of file `locale_facets.h`.

4.652.4.2 `template<typename _CharT> virtual char_type std::numpunct<_CharT>::do_decimal_point () const`
`[inline]`, `[protected]`, `[virtual]`

Return decimal point character.

Returns a `char_type` to use as a decimal point. This function is a hook for derived classes to change the value returned.

Returns

char_type representing a decimal point.

Definition at line 1794 of file `locale_facets.h`.

4.652.4.3 `template<typename _CharT> virtual string_type std::num_punct<_CharT>::do_falsename () const`
[inline], [protected], [virtual]

Return string representation of bool false.

Returns a `string_type` containing the text representation for false bool variables. This function is a hook for derived classes to change the value returned.

Returns

`string_type` representing printed form of false.

Definition at line 1845 of file `locale_facets.h`.

References `std::num_punct<_CharT>::~~num_punct()`.

4.652.4.4 `template<typename _CharT> virtual string std::num_punct<_CharT>::do_grouping () const` [inline],
[protected], [virtual]

Return grouping specification.

Returns a string representing groupings for the integer part of a number. This function is a hook for derived classes to change the value returned.

See also

`grouping()` for details.

Returns

String representing grouping specification.

Definition at line 1819 of file `locale_facets.h`.

4.652.4.5 `template<typename _CharT> virtual char_type std::num_punct<_CharT>::do_thousands_sep () const`
[inline], [protected], [virtual]

Return thousands separator character.

Returns a `char_type` to use as a thousands separator. This function is a hook for derived classes to change the value returned.

Returns

`char_type` representing a thousands separator.

Definition at line 1806 of file `locale_facets.h`.

4.652.4.6 `template<typename _CharT > virtual string_type std::num_punct<_CharT>::do_truename () const`
`[inline], [protected], [virtual]`

Return string representation of bool true.

Returns a `string_type` containing the text representation for true bool variables. This function is a hook for derived classes to change the value returned.

Returns

`string_type` representing printed form of true.

Definition at line 1832 of file `locale_facets.h`.

4.652.4.7 `template<typename _CharT > string_type std::num_punct<_CharT>::falsename () const` `[inline]`

Return string representation of bool false.

This function returns a `string_type` containing the text representation for false bool variables. It does so by calling `num_punct<char_type>::do_falsename()`.

Returns

`string_type` representing printed form of false.

Definition at line 1777 of file `locale_facets.h`.

4.652.4.8 `template<typename _CharT > string std::num_punct<_CharT>::grouping () const` `[inline]`

Return grouping specification.

This function returns a string representing groupings for the integer part of a number. Groupings indicate where thousands separators should be inserted in the integer part of a number.

Each char in the return string is interpreted as an integer rather than a character. These numbers represent the number of digits in a group. The first char in the string represents the number of digits in the least significant group. If a char is negative, it indicates an unlimited number of digits for the group. If more chars from the string are required to group a number, the last char is used repeatedly.

For example, if the `grouping()` returns `"\003\002"` and is applied to the number 123456789, this corresponds to 12,34,56,789. Note that if the string was `"32"`, this would put more than 50 digits into the least significant group if the character set is ASCII.

The string is returned by calling `num_punct<char_type>::do_grouping()`.

Returns

string representing grouping specification.

Definition at line 1751 of file `locale_facets.h`.

4.652.4.9 `template<typename _CharT> char_type std::numpunct<_CharT>::thousands_sep() const [inline]`

Return thousands separator character.

This function returns a `char_type` to use as a thousands separator. It does so by returning `numpunct<char_type>::do_thousands_sep()`.

Returns

`char_type` representing a thousands separator.

Definition at line 1720 of file `locale_facets.h`.

4.652.4.10 `template<typename _CharT> string_type std::numpunct<_CharT>::truename() const [inline]`

Return string representation of bool true.

This function returns a `string_type` containing the text representation for true bool variables. It does so by calling `numpunct<char_type>::do_truename()`.

Returns

`string_type` representing printed form of true.

Definition at line 1764 of file `locale_facets.h`.

4.652.5 Member Data Documentation

4.652.5.1 `template<typename _CharT> locale::id std::numpunct<_CharT>::id [static]`

Numpunct facet id.

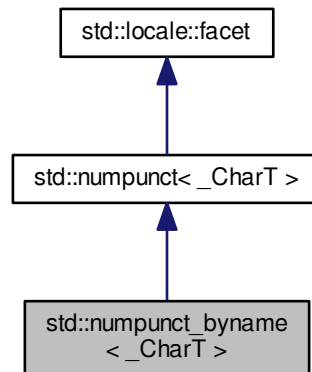
Definition at line 1657 of file `locale_facets.h`.

The documentation for this class was generated from the following file:

- [locale_facets.h](#)

4.653 `std::numpunct_byname<_CharT>` Class Template Reference

Inheritance diagram for `std::numpunct_byname<_CharT>`:



Public Types

- typedef `__numpunct_cache<_CharT>` **`__cache_type`**
- typedef `_CharT` **`char_type`**
- typedef `basic_string<_CharT>` **`string_type`**

Public Member Functions

- **`numpunct_byname`** (`const char *__s, size_t __refs=0`)
- `char_type decimal_point` () const
- `string_type falsename` () const
- `string_type grouping` () const
- `char_type thousands_sep` () const
- `string_type truename` () const

Static Public Attributes

- static `locale::id` `id`

Protected Member Functions

- void **_M_initialize_numpunct** (__c_locale __cloc=0)
- template<>
void **_M_initialize_numpunct** (__c_locale __cloc)
- template<>
void **_M_initialize_numpunct** (__c_locale __cloc)
- virtual [char_type](#) **do_decimal_point** () const
- virtual [string_type](#) **do_falsename** () const
- virtual [string](#) **do_grouping** () const
- virtual [char_type](#) **do_thousands_sep** () const
- virtual [string_type](#) **do_truename** () const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

Protected Attributes

- __cache_type * **_M_data**

4.653.1 Detailed Description

```
template<typename _CharT>
class std::num_punct_byname<_CharT>
```

class num_punct_byname [22.2.3.2].

Definition at line 1874 of file locale_facets.h.

4.653.2 Member Function Documentation

4.653.2.1 `template<typename _CharT> char_type std::num_punct<_CharT>::decimal_point () const` `[inline]`,
`[inherited]`

Return decimal point character.

This function returns a `char_type` to use as a decimal point. It does so by returning `num_punct<char_type>::do_decimal_point()`.

Returns

char_type representing a decimal point.

Definition at line 1707 of file locale_facets.h.

4.653.2.2 `template<typename _CharT> virtual char_type std::num_punct<_CharT>::do_decimal_point () const`
`[inline], [protected], [virtual], [inherited]`

Return decimal point character.

Returns a `char_type` to use as a decimal point. This function is a hook for derived classes to change the value returned.

Returns

char_type representing a decimal point.

Definition at line 1794 of file `locale_facets.h`.

4.653.2.3 `template<typename _CharT> virtual string_type std::num_punct<_CharT>::do_falsename () const`
`[inline], [protected], [virtual], [inherited]`

Return string representation of bool false.

Returns a `string_type` containing the text representation for false bool variables. This function is a hook for derived classes to change the value returned.

Returns

string_type representing printed form of false.

Definition at line 1845 of file `locale_facets.h`.

References `std::num_punct<_CharT>::~num_punct()`.

4.653.2.4 `template<typename _CharT> virtual string std::num_punct<_CharT>::do_grouping () const` `[inline],`
`[protected], [virtual], [inherited]`

Return grouping specification.

Returns a string representing groupings for the integer part of a number. This function is a hook for derived classes to change the value returned.

See also

`grouping()` for details.

Returns

String representing grouping specification.

Definition at line 1819 of file `locale_facets.h`.

4.653.2.5 `template<typename _CharT> virtual char_type std::num_punct<_CharT>::do_thousands_sep () const`
[inline], [protected], [virtual], [inherited]

Return thousands separator character.

Returns a `char_type` to use as a thousands separator. This function is a hook for derived classes to change the value returned.

Returns

char_type representing a thousands separator.

Definition at line 1806 of file `locale_facets.h`.

4.653.2.6 `template<typename _CharT> virtual string_type std::num_punct<_CharT>::do_truename () const`
[inline], [protected], [virtual], [inherited]

Return string representation of bool true.

Returns a `string_type` containing the text representation for true bool variables. This function is a hook for derived classes to change the value returned.

Returns

`string_type` representing printed form of true.

Definition at line 1832 of file `locale_facets.h`.

4.653.2.7 `template<typename _CharT> string_type std::num_punct<_CharT>::falsename () const` [inline],
[inherited]

Return string representation of bool false.

This function returns a `string_type` containing the text representation for false bool variables. It does so by calling `num_punct<char_type>::do_falsename()`.

Returns

`string_type` representing printed form of false.

Definition at line 1777 of file `locale_facets.h`.

4.653.2.8 `template<typename _CharT> string std::numpunct<_CharT>::grouping () const [inline],
[inherited]`

Return grouping specification.

This function returns a string representing groupings for the integer part of a number. Groupings indicate where thousands separators should be inserted in the integer part of a number.

Each char in the return string is interpreted as an integer rather than a character. These numbers represent the number of digits in a group. The first char in the string represents the number of digits in the least significant group. If a char is negative, it indicates an unlimited number of digits for the group. If more chars from the string are required to group a number, the last char is used repeatedly.

For example, if the grouping() returns "\003\002" and is applied to the number 123456789, this corresponds to 12,34,56,789. Note that if the string was "32", this would put more than 50 digits into the least significant group if the character set is ASCII.

The string is returned by calling `numpunct<char_type>::do_grouping()`.

Returns

string representing grouping specification.

Definition at line 1751 of file `locale_facets.h`.

4.653.2.9 `template<typename _CharT> char_type std::numpunct<_CharT>::thousands_sep () const [inline],
[inherited]`

Return thousands separator character.

This function returns a `char_type` to use as a thousands separator. It does so by returning `numpunct<char_type>::do_thousands_sep()`.

Returns

`char_type` representing a thousands separator.

Definition at line 1720 of file `locale_facets.h`.

4.653.2.10 `template<typename _CharT> string_type std::numpunct<_CharT>::truename () const [inline],
[inherited]`

Return string representation of bool true.

This function returns a `string_type` containing the text representation for true bool variables. It does so by calling `numpunct<char_type>::do_truename()`.

Returns

`string_type` representing printed form of true.

Definition at line 1764 of file `locale_facets.h`.

4.653.3 Member Data Documentation

4.653.3.1 template<typename _CharT> locale::id std::numpunct<_CharT>::id [static], [inherited]

Numpunct facet id.

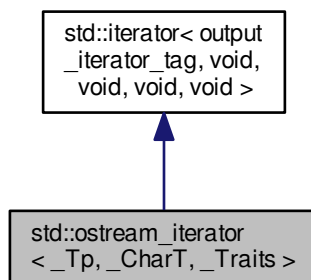
Definition at line 1657 of file locale_facets.h.

The documentation for this class was generated from the following file:

- [locale_facets.h](#)

4.654 std::ostream_iterator< _Tp, _CharT, _Traits > Class Template Reference

Inheritance diagram for std::ostream_iterator< _Tp, _CharT, _Traits >:



Public Types

- typedef void [difference_type](#)
- typedef [output_iterator_tag](#) [iterator_category](#)
- typedef void [pointer](#)
- typedef void [reference](#)
- typedef void [value_type](#)

- typedef _CharT [char_type](#)
- typedef _Traits [traits_type](#)
- typedef basic_ostream< _CharT, _Traits > [ostream_type](#)

Public Member Functions

- [ostream_iterator](#) ([ostream_type](#) &__s)
- [ostream_iterator](#) ([ostream_type](#) &__s, const [_CharT](#) *__c)
- [ostream_iterator](#) (const [ostream_iterator](#) &__obj)
- [ostream_iterator](#) & **operator*** ()
- [ostream_iterator](#) & **operator++** ()
- [ostream_iterator](#) & **operator++** (int)
- [ostream_iterator](#) & **operator=** (const [_Tp](#) &__value)

4.654.1 Detailed Description

```
template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>>
class std::ostream_iterator< _Tp, _CharT, _Traits >
```

Provides output iterator semantics for streams.

This class provides an iterator to write to an ostream. The type `Tp` is the only type written by this iterator and there must be an `operator<<(Tp)` defined.

Template Parameters

<code>_Tp</code>	The type to write to the ostream.
<code>_CharT</code>	The ostream char_type.
<code>_Traits</code>	The ostream char_traits.

Definition at line 154 of file `stream_iterator.h`.

4.654.2 Member Typedef Documentation

4.654.2.1 `template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>> typedef _CharT std::ostream_iterator< _Tp, _CharT, _Traits >::char_type`

Public typedef.

Definition at line 160 of file `stream_iterator.h`.

4.654.2.2 `typedef void std::iterator< output_iterator_tag, void, void, void, void >::difference_type` `[inherited]`

Distance between iterators is represented as this type.

Definition at line 125 of file `stl_iterator_base_types.h`.

4.654.2.3 `typedef output_iterator_tag std::iterator< output_iterator_tag, void, void, void, void >::iterator_category` `[inherited]`

One of the [tag types](#).

Definition at line 121 of file `stl_iterator_base_types.h`.

4.654.2.4 `template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>> typedef
basic_ostream<_CharT, _Traits> std::ostream_iterator< _Tp, _CharT, _Traits >::ostream_type`

Public typedef.

Definition at line 162 of file `stream_iterator.h`.

4.654.2.5 `typedef void std::iterator< output_iterator_tag, void, void, void, void >::pointer [inherited]`

This type represents a pointer-to-value_type.

Definition at line 127 of file `stl_iterator_base_types.h`.

4.654.2.6 `typedef void std::iterator< output_iterator_tag, void, void, void, void >::reference [inherited]`

This type represents a reference-to-value_type.

Definition at line 129 of file `stl_iterator_base_types.h`.

4.654.2.7 `template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>> typedef _Traits
std::ostream_iterator< _Tp, _CharT, _Traits >::traits_type`

Public typedef.

Definition at line 161 of file `stream_iterator.h`.

4.654.2.8 `typedef void std::iterator< output_iterator_tag, void, void, void, void >::value_type [inherited]`

The type "pointed to" by the iterator.

Definition at line 123 of file `stl_iterator_base_types.h`.

4.654.3 Constructor & Destructor Documentation

4.654.3.1 `template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>>
std::ostream_iterator< _Tp, _CharT, _Traits >::ostream_iterator (ostream_type &__s) [inline]`

Construct from an ostream.

Definition at line 171 of file `stream_iterator.h`.

4.654.3.2 `template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>>
std::ostream_iterator< _Tp, _CharT, _Traits >::ostream_iterator (ostream_type &__s, const _CharT * __c)
[inline]`

Construct from an ostream.

The delimiter string `c` is written to the stream after every `Tp` written to the stream. The delimiter is not copied, and thus must not be destroyed while this iterator is in use.

Parameters

<code>_↔ _s</code>	Underlying ostream to write to.
<code>_↔ _c</code>	CharT delimiter string to insert.

Definition at line 183 of file stream_iterator.h.

```
4.654.3.3  template<typename _Tp , typename _CharT = char, typename _Traits = char_traits<_CharT>>
            std::ostream_iterator< _Tp, _CharT, _Traits >::ostream_iterator ( const ostream_iterator< _Tp, _CharT,
            _Traits > &__obj )  [inline]
```

Copy constructor.

Definition at line 187 of file stream_iterator.h.

4.654.4 Member Function Documentation

```
4.654.4.1  template<typename _Tp , typename _CharT = char, typename _Traits = char_traits<_CharT>> ostream_iterator&
            std::ostream_iterator< _Tp, _CharT, _Traits >::operator= ( const _Tp &__value )  [inline]
```

Writes *value* to underlying ostream using operator<<. If constructed with delimiter string, writes delimiter to ostream.

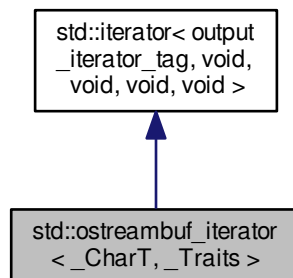
Definition at line 193 of file stream_iterator.h.

The documentation for this class was generated from the following file:

- [stream_iterator.h](#)

4.655 std::ostreambuf_iterator< _CharT, _Traits > Class Template Reference

Inheritance diagram for std::ostreambuf_iterator< _CharT, _Traits >:



Public Types

- typedef void [difference_type](#)
- typedef [output_iterator_tag](#) [iterator_category](#)
- typedef void [pointer](#)
- typedef void [reference](#)
- typedef void [value_type](#)
- typedef [_CharT](#) [char_type](#)
- typedef [_Traits](#) [traits_type](#)
- typedef basic_streambuf< [_CharT](#), [_Traits](#) > [streambuf_type](#)
- typedef basic_ostream< [_CharT](#), [_Traits](#) > [ostream_type](#)

Public Member Functions

- [ostreambuf_iterator](#) ([ostream_type](#) &__s) noexcept
- [ostreambuf_iterator](#) ([streambuf_type](#) *__s) noexcept
- [ostreambuf_iterator](#) & [M_put](#) (const [_CharT](#) *__ws, [streamsize](#) __len)
- bool [failed](#) () const noexcept
- [ostreambuf_iterator](#) & [operator*](#) ()
- [ostreambuf_iterator](#) & [operator++](#) (int)
- [ostreambuf_iterator](#) & [operator++](#) ()
- [ostreambuf_iterator](#) & [operator=](#) ([_CharT](#) __c)

Friends

- template<typename [_CharT2](#) >
[__gnu_cxx::__enable_if< __is_char< \[_CharT2\]\(#\) >::__value, \[ostreambuf_iterator\]\(#\)< \[_CharT2\]\(#\) >::__type](#) **copy**
([istreambuf_iterator](#)< [_CharT2](#) >, [istreambuf_iterator](#)< [_CharT2](#) >, [ostreambuf_iterator](#)< [_CharT2](#) >)

4.655.1 Detailed Description

```
template<typename \_CharT, typename \_Traits>
class std::ostreambuf_iterator< \_CharT, \_Traits >
```

Provides output iterator semantics for streambufs.

Definition at line 410 of file `stl_algobase.h`.

4.655.2 Member Typedef Documentation

4.655.2.1 template<typename [_CharT](#), typename [_Traits](#)> typedef [_CharT](#) std::ostreambuf_iterator< [_CharT](#), [_Traits](#) >::char_type

Public typedefs.

Definition at line 223 of file `streambuf_iterator.h`.

4.655.2.2 **typedef void std::iterator< output_iterator_tag , void , void , void , void >::difference_type** [inherited]

Distance between iterators is represented as this type.

Definition at line 125 of file stl_iterator_base_types.h.

4.655.2.3 **typedef output_iterator_tag std::iterator< output_iterator_tag , void , void , void , void >::iterator_category** [inherited]

One of the [tag types](#).

Definition at line 121 of file stl_iterator_base_types.h.

4.655.2.4 **template<typename _CharT, typename _Traits> typedef basic_ostream<_CharT, _Traits> std::ostreambuf_iterator< _CharT, _Traits >::ostream_type**

Public typedefs.

Definition at line 226 of file streambuf_iterator.h.

4.655.2.5 **typedef void std::iterator< output_iterator_tag , void , void , void , void >::pointer** [inherited]

This type represents a pointer-to-value_type.

Definition at line 127 of file stl_iterator_base_types.h.

4.655.2.6 **typedef void std::iterator< output_iterator_tag , void , void , void , void >::reference** [inherited]

This type represents a reference-to-value_type.

Definition at line 129 of file stl_iterator_base_types.h.

4.655.2.7 **template<typename _CharT, typename _Traits> typedef basic_streambuf<_CharT, _Traits> std::ostreambuf_iterator< _CharT, _Traits >::streambuf_type**

Public typedefs.

Definition at line 225 of file streambuf_iterator.h.

4.655.2.8 **template<typename _CharT, typename _Traits> typedef _Traits std::ostreambuf_iterator< _CharT, _Traits >::traits_type**

Public typedefs.

Definition at line 224 of file streambuf_iterator.h.

4.655.2.9 **typedef void std::iterator< output_iterator_tag , void , void , void , void >::value_type** [inherited]

The type "pointed to" by the iterator.

Definition at line 123 of file stl_iterator_base_types.h.

4.655.3 Constructor & Destructor Documentation

4.655.3.1 `template<typename _CharT, typename _Traits> std::ostreambuf_iterator<_CharT, _Traits>::ostreambuf_iterator(ostream_type &__s) [inline], [noexcept]`

Construct output iterator from ostream.

Definition at line 241 of file ostreambuf_iterator.h.

4.655.3.2 `template<typename _CharT, typename _Traits> std::ostreambuf_iterator<_CharT, _Traits>::ostreambuf_iterator(streambuf_type *__s) [inline], [noexcept]`

Construct output iterator from streambuf.

Definition at line 245 of file ostreambuf_iterator.h.

4.655.4 Member Function Documentation

4.655.4.1 `template<typename _CharT, typename _Traits> bool std::ostreambuf_iterator<_CharT, _Traits>::failed() const [inline], [noexcept]`

Return true if previous operator=() failed.

Definition at line 275 of file ostreambuf_iterator.h.

4.655.4.2 `template<typename _CharT, typename _Traits> ostreambuf_iterator& std::ostreambuf_iterator<_CharT, _Traits>::operator*() [inline]`

Return *this.

Definition at line 260 of file ostreambuf_iterator.h.

4.655.4.3 `template<typename _CharT, typename _Traits> ostreambuf_iterator& std::ostreambuf_iterator<_CharT, _Traits>::operator++(int) [inline]`

Return *this.

Definition at line 265 of file ostreambuf_iterator.h.

4.655.4.4 `template<typename _CharT, typename _Traits> ostreambuf_iterator& std::ostreambuf_iterator<_CharT, _Traits>::operator++() [inline]`

Return *this.

Definition at line 270 of file ostreambuf_iterator.h.

4.655.4.5 `template<typename _CharT, typename _Traits> ostreambuf_iterator& std::ostreambuf_iterator< _CharT, _Traits >::operator= (_CharT __c) [inline]`

Write character to streambuf. Calls streambuf.sputc().

Definition at line 250 of file streambuf_iterator.h.

The documentation for this class was generated from the following files:

- [stl_algobase.h](#)
- [streambuf_iterator.h](#)

4.656 `std::output_iterator_tag` Struct Reference

4.656.1 Detailed Description

Marking output iterators.

Definition at line 92 of file stl_iterator_base_types.h.

The documentation for this struct was generated from the following file:

- [stl_iterator_base_types.h](#)

4.657 `std::owner_less< _Tp >` Struct Template Reference

4.657.1 Detailed Description

```
template<typename _Tp>
struct std::owner_less< _Tp >
```

Primary template owner_less.

Definition at line 513 of file shared_ptr.h.

The documentation for this struct was generated from the following file:

- [shared_ptr.h](#)

4.658 `std::owner_less< shared_ptr< _Tp > >` Struct Template Reference

Inherits `std::Sp_owner_less< _Tp, _Tp1 >`.

Public Types

- `typedef _Tp` [first_argument_type](#)
- `typedef bool` [result_type](#)
- `typedef _Tp` [second_argument_type](#)

Public Member Functions

- `bool operator()` (`const _Tp &__lhs, const _Tp &__rhs`) `const`
- `bool operator()` (`const _Tp &__lhs, const _Tp1 &__rhs`) `const`
- `bool operator()` (`const _Tp1 &__lhs, const _Tp &__rhs`) `const`

4.658.1 Detailed Description

```
template<typename _Tp>
struct std::owner_less< shared_ptr< _Tp > >
```

Partial specialization of `owner_less` for `shared_ptr`.

Definition at line 517 of file `shared_ptr.h`.

4.658.2 Member Typedef Documentation

4.658.2.1 `typedef _Tp std::binary_function< _Tp, _Tp, bool >::first_argument_type` [inherited]

`first_argument_type` is the type of the first argument

Definition at line 121 of file `stl_function.h`.

4.658.2.2 `typedef bool std::binary_function< _Tp, _Tp, bool >::result_type` [inherited]

`result_type` is the return type

Definition at line 127 of file `stl_function.h`.

4.658.2.3 `typedef _Tp std::binary_function< _Tp, _Tp, bool >::second_argument_type` [inherited]

`second_argument_type` is the type of the second argument

Definition at line 124 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [shared_ptr.h](#)

4.659 `std::owner_less< weak_ptr< _Tp > >` Struct Template Reference

Inherits `std::_Sp_owner_less< _Tp, _Tp1 >`.

Public Types

- typedef `_Tp` [first_argument_type](#)
- typedef `bool` [result_type](#)
- typedef `_Tp` [second_argument_type](#)

Public Member Functions

- `bool` **operator()** (`const _Tp &__lhs, const _Tp &__rhs`) `const`
- `bool` **operator()** (`const _Tp &__lhs, const _Tp1 &__rhs`) `const`
- `bool` **operator()** (`const _Tp1 &__lhs, const _Tp &__rhs`) `const`

4.659.1 Detailed Description

```
template<typename _Tp>
struct std::owner_less< weak_ptr< _Tp > >
```

Partial specialization of `owner_less` for `weak_ptr`.

Definition at line 523 of file `shared_ptr.h`.

4.659.2 Member Typedef Documentation

4.659.2.1 `typedef _Tp std::binary_function< _Tp, _Tp, bool >::first_argument_type` [inherited]

`first_argument_type` is the type of the first argument

Definition at line 121 of file `stl_function.h`.

4.659.2.2 `typedef bool std::binary_function< _Tp, _Tp, bool >::result_type` [inherited]

`result_type` is the return type

Definition at line 127 of file `stl_function.h`.

4.659.2.3 `typedef _Tp std::binary_function< _Tp, _Tp, bool >::second_argument_type` [inherited]

`second_argument_type` is the type of the second argument

Definition at line 124 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [shared_ptr.h](#)

4.660 std::pair<_T1, _T2> Struct Template Reference

Public Types

- typedef `_T1` **first_type**
- typedef `_T2` **second_type**

Public Member Functions

- constexpr `pair` ()
- constexpr `pair` (const `_T1` &__a, const `_T2` &__b)
- template<class `_U1` , class `_U2` , class = typename enable_if<__and<is_convertible<const `_U1`&, `_T1`>, is_convertible<const `_U2`&, `_T2`>>::value>::type>
constexpr `pair` (const `pair`< `_U1`, `_U2` > &__p)
- constexpr `pair` (const `pair` &)=default
- constexpr `pair` (`pair` &&)=default
- template<class `_U1` , class = typename enable_if<is_convertible< `_U1`, `_T1`>::value>::type>
constexpr `pair` (`_U1` &&__x, const `_T2` &__y)
- template<class `_U2` , class = typename enable_if<is_convertible< `_U2`, `_T2`>::value>::type>
constexpr `pair` (const `_T1` &__x, `_U2` &&__y)
- template<class `_U1` , class `_U2` , class = typename enable_if<__and<is_convertible< `_U1`, `_T1`>, is_convertible< `_U2`, `_T2`>>::value>::type>
constexpr `pair` (`_U1` &&__x, `_U2` &&__y)
- template<class `_U1` , class `_U2` , class = typename enable_if<__and<is_convertible< `_U1`, `_T1`>, is_convertible< `_U2`, `_T2`>>::value>::type>
constexpr `pair` (`pair`< `_U1`, `_U2` > &&__p)
- template<typename... `_Args1`, typename... `_Args2`>
`pair` (`piecewise_construct_t`, tuple< `_Args1`... >, tuple< `_Args2`... >)
- `pair` & operator= (const `pair` &__p)
- `pair` & operator= (`pair` &&__p) noexcept(__and< is_nothrow_move_assignable< `_T1` >, is_nothrow_move_assignable< `_T2` >>::value)
- template<class `_U1` , class `_U2` >
`pair` & operator= (const `pair`< `_U1`, `_U2` > &__p)
- template<class `_U1` , class `_U2` >
`pair` & operator= (`pair`< `_U1`, `_U2` > &&__p)
- void **swap** (`pair` &__p) noexcept(noexcept(swap(`first`, __p.first)) &&noexcept(swap(`second`, __p.second)))

Public Attributes

- `_T1` **first**
- `_T2` **second**

4.660.1 Detailed Description

```
template<class _T1, class _T2>
struct std::pair<_T1, _T2>
```

Struct holding two objects of arbitrary type.

Template Parameters

<code>_T1</code>	Type of first object.
<code>_T2</code>	Type of second object.

Definition at line 96 of file `stl_pair.h`.

4.660.2 Member Typedef Documentation

4.660.2.1 `template<class _T1, class _T2> typedef _T2 std::pair<_T1, _T2>::second_type`

`first_type` is the first bound type

Definition at line 99 of file `stl_pair.h`.

4.660.3 Constructor & Destructor Documentation

4.660.3.1 `template<class _T1, class _T2> constexpr std::pair<_T1, _T2>::pair () [inline]`

`second` is a copy of the second object

The default constructor creates `first` and `second` using their respective default constructors.

Definition at line 108 of file `stl_pair.h`.

4.660.3.2 `template<class _T1, class _T2> constexpr std::pair<_T1, _T2>::pair (const _T1 & __a, const _T2 & __b) [inline]`

Two objects may be passed to a `pair` constructor to be copied.

Definition at line 112 of file `stl_pair.h`.

4.660.3.3 `template<class _T1, class _T2> template<class _U1, class _U2, class = typename enable_if<__and_<is_↵ convertible<const _U1&, _T1>, is_convertible<const _U2&, _T2>>::value>::type> constexpr std::pair<_T1, _T2>::pair (const pair<_U1, _U2> & __p) [inline]`

There is also a templated copy ctor for the `pair` class itself.

Definition at line 124 of file `stl_pair.h`.

4.660.4 Member Data Documentation

4.660.4.1 `template<class _T1, class _T2> _T1 std::pair<_T1, _T2>::first`

`second_type` is the second bound type

Definition at line 101 of file `stl_pair.h`.

Referenced by `__gnu_parallel::__qsb_conquer()`, `std::Temporary_buffer<_ForwardIterator, _Tp>::Temporary_buffer()`, `std::set<_Key, _Compare, _Alloc>::insert()`, `std::minmax_element()`, `__gnu_pbds::detail::pat_trie_map<Key, Mapped, Node_And_It_Traits, _Alloc>::node_end()`, `std::operator<()`, `std::operator==()`, and `std::pair<_Biliter, _Biliter>::pair()`.

4.660.4.2 `template<class _T1, class _T2> _T2 std::pair<_T1, _T2>::second`

`first` is a copy of the first object

Definition at line 102 of file `stl_pair.h`.

Referenced by `__gnu_parallel::__qsb_conquer()`, `std::Temporary_buffer<_ForwardIterator, _Tp>::Temporary_buffer()`, `std::set<_Key, _Compare, _Alloc>::insert()`, `std::minmax_element()`, `std::operator==()`, and `std::pair<_Biliter, _Biliter>::pair()`.

The documentation for this struct was generated from the following file:

- [stl_pair.h](#)

4.661 `std::piecewise_constant_distribution<_RealType>` Class Template Reference

Classes

- struct [param_type](#)

Public Types

- typedef `_RealType` [result_type](#)

Public Member Functions

- `template<typename _InputIteratorB, typename _InputIteratorW >`
piecewise_constant_distribution (`_InputIteratorB __bfirst, _InputIteratorB __bend, _InputIteratorW __wbegin`)
- `template<typename _Func >`
piecewise_constant_distribution (`initializer_list< _RealType > __bl, _Func __fw`)
- `template<typename _Func >`
piecewise_constant_distribution (`size_t __nw, _RealType __xmin, _RealType __xmax, _Func __fw`)
- **piecewise_constant_distribution** (`const param_type &__p`)
- `template<typename _ForwardIterator, typename _UniformRandomNumberGenerator >`
void __generate (`_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng`)
- `template<typename _ForwardIterator, typename _UniformRandomNumberGenerator >`
void __generate (`_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param_type &__p`)
- `template<typename _UniformRandomNumberGenerator >`
void __generate (`result_type *__f, result_type *__t, _UniformRandomNumberGenerator &__urng, const param_type &__p`)
- `std::vector< double > densities` (`() const`)
- `std::vector< _RealType > intervals` (`() const`)
- `result_type max` (`() const`)
- `result_type min` (`() const`)
- `template<typename _UniformRandomNumberGenerator >`
result_type operator() (`_UniformRandomNumberGenerator &__urng`)
- `template<typename _UniformRandomNumberGenerator >`
result_type operator() (`_UniformRandomNumberGenerator &__urng, const param_type &__p`)
- `param_type param` (`() const`)
- `void param` (`const param_type &__param`)
- `void reset` (`()`)

Friends

- `template<typename _RealType1, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & operator<<` (`(std::basic_ostream< _CharT, _Traits > &__os, const std::piecewise_constant_distribution< _RealType1 > &__x)`)
- `bool operator==` (`(const piecewise_constant_distribution &__d1, const piecewise_constant_distribution &__d2)`)
- `template<typename _RealType1, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & operator>>` (`(std::basic_istream< _CharT, _Traits > &__is, std::piecewise_constant_distribution< _RealType1 > &__x)`)

4.661.1 Detailed Description

```
template<typename _RealType = double>
class std::piecewise_constant_distribution< _RealType >
```

A `piecewise_constant_distribution` random number distribution.

The formula for the piecewise constant probability mass function is

Definition at line 5483 of file `random.h`.

4.661.2 Member Typedef Documentation

4.661.2.1 `template<typename _RealType = double> typedef _RealType std::piecewise_constant_distribution<_RealType>::result_type`

The type of the range of the distribution.

Definition at line 5486 of file random.h.

4.661.3 Member Function Documentation

4.661.3.1 `template<typename _RealType = double> std::vector<double> std::piecewise_constant_distribution<_RealType>::densities () const [inline]`

Returns a vector of the probability densities.

Definition at line 5604 of file random.h.

4.661.3.2 `template<typename _RealType = double> std::vector<_RealType> std::piecewise_constant_distribution<_RealType>::intervals () const [inline]`

Returns a vector of the intervals.

Definition at line 5588 of file random.h.

4.661.3.3 `template<typename _RealType = double> result_type std::piecewise_constant_distribution<_RealType>::max () const [inline]`

Returns the least upper bound value of the distribution.

Definition at line 5639 of file random.h.

4.661.3.4 `template<typename _RealType = double> result_type std::piecewise_constant_distribution<_RealType>::min () const [inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 5629 of file random.h.

4.661.3.5 `template<typename _RealType = double> template<typename _UniformRandomNumberGenerator> result_type std::piecewise_constant_distribution<_RealType>::operator() (_UniformRandomNumberGenerator & __urng) [inline]`

Generating functions.

Definition at line 5650 of file random.h.

4.661.3.6 `template<typename _RealType = double> param_type std::piecewise_constant_distribution<_RealType>::param () const [inline]`

Returns the parameter set of the distribution.

Definition at line 5614 of file random.h.

4.661.3.7 `template<typename _RealType = double> void std::piecewise_constant_distribution<_RealType>::param (const param_type & __param) [inline]`

Sets the parameter set of the distribution.

Parameters

<code>__param</code>	The new parameter set of the distribution.
----------------------	--

Definition at line 5622 of file random.h.

4.661.3.8 `template<typename _RealType = double> void std::piecewise_constant_distribution<_RealType>::reset ()`
`[inline]`

Resets the distribution state.

Definition at line 5581 of file random.h.

4.661.4 Friends And Related Function Documentation

4.661.4.1 `template<typename _RealType = double> template<typename _RealType1 , typename _CharT , typename _Traits`
`> std::basic_ostream<_CharT, _Traits>& operator<< (std::basic_ostream<_CharT, _Traits> & __os, const`
`std::piecewise_constant_distribution<_RealType1> & __x) [friend]`

Inserts a `piecewise_constant_distribution` random number distribution `__x` into the output stream `__os`.

Parameters

<code>__os</code>	An output stream.
<code>__x</code>	A <code>piecewise_constant_distribution</code> random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

4.661.4.2 `template<typename _RealType = double> bool operator==(const piecewise_constant_distribution<_RealType`
`> & __d1, const piecewise_constant_distribution<_RealType> & __d2) [friend]`

Return true if two `piecewise constant distributions` have the same parameters.

Definition at line 5685 of file random.h.

4.661.4.3 `template<typename _RealType = double> template<typename _RealType1 , typename _CharT , typename`
`_Traits> std::basic_istream<_CharT, _Traits>& operator>> (std::basic_istream<_CharT, _Traits> & __is,`
`std::piecewise_constant_distribution<_RealType1> & __x) [friend]`

Extracts a `piecewise_constant_distribution` random number distribution `__x` from the input stream `__is`.

Parameters

<code>__is</code>	An input stream.
<code>__x</code>	A piecewise_constant_distribution random number generator engine.

Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following file:

- [random.h](#)

4.662 std::piecewise_constant_distribution< _RealType >::param_type Struct Reference

Public Types

- typedef [piecewise_constant_distribution](#)< _RealType > **distribution_type**

Public Member Functions

- template<typename _InputIteratorB , typename _InputIteratorW >
param_type (_InputIteratorB __bfirst, _InputIteratorB __bend, _InputIteratorW __wbegin)
- template<typename _Func >
param_type (initializer_list< _RealType > __bi, _Func __fw)
- template<typename _Func >
param_type (size_t __nw, _RealType __xmin, _RealType __xmax, _Func __fw)
- **param_type** (const [param_type](#) &)=default
- [std::vector](#)< double > **densities** () const
- [std::vector](#)< _RealType > **intervals** () const
- [param_type](#) & **operator=** (const [param_type](#) &)=default

Friends

- bool **operator==** (const [param_type](#) &__p1, const [param_type](#) &__p2)
- class [piecewise_constant_distribution](#)< _RealType >

4.662.1 Detailed Description

```
template<typename _RealType = double>
struct std::piecewise_constant_distribution< _RealType >::param_type
```

Parameter type.

Definition at line 5492 of file random.h.

The documentation for this struct was generated from the following file:

- [random.h](#)

4.663 `std::piecewise_construct_t` Struct Reference

4.663.1 Detailed Description

`piecewise_construct_t`

Definition at line 76 of file `stl_pair.h`.

The documentation for this struct was generated from the following file:

- [stl_pair.h](#)

4.664 `std::piecewise_linear_distribution<_RealType>` Class Template Reference

Classes

- struct [param_type](#)

Public Types

- typedef `_RealType` [result_type](#)

Public Member Functions

- template<typename `_InputIteratorB` , typename `_InputIteratorW` >
`piecewise_linear_distribution` (`_InputIteratorB` `__bfirst`, `_InputIteratorB` `__bend`, `_InputIteratorW` `__wbegin`)
- template<typename `_Func` >
`piecewise_linear_distribution` (initializer_list< `_RealType` > `__bl`, `_Func` `__fw`)
- template<typename `_Func` >
`piecewise_linear_distribution` (size_t `__nw`, `_RealType` `__xmin`, `_RealType` `__xmax`, `_Func` `__fw`)
- **`piecewise_linear_distribution`** (const [param_type](#) &`__p`)
- template<typename `_ForwardIterator` , typename `_UniformRandomNumberGenerator` >
void **`__generate`** (`_ForwardIterator` `__f`, `_ForwardIterator` `__t`, `_UniformRandomNumberGenerator` &`__urng`)
- template<typename `_ForwardIterator` , typename `_UniformRandomNumberGenerator` >
void **`__generate`** (`_ForwardIterator` `__f`, `_ForwardIterator` `__t`, `_UniformRandomNumberGenerator` &`__urng`, const [param_type](#) &`__p`)
- template<typename `_UniformRandomNumberGenerator` >
void **`__generate`** ([result_type](#) *`__f`, [result_type](#) *`__t`, `_UniformRandomNumberGenerator` &`__urng`, const [param_type](#) &`__p`)
- `std::vector`< double > [densities](#) () const
- `std::vector`< `_RealType` > [intervals](#) () const
- [result_type](#) [max](#) () const
- [result_type](#) [min](#) () const
- template<typename `_UniformRandomNumberGenerator` >
[result_type](#) [operator\(\)](#) (`_UniformRandomNumberGenerator` &`__urng`)
- template<typename `_UniformRandomNumberGenerator` >
[result_type](#) [operator\(\)](#) (`_UniformRandomNumberGenerator` &`__urng`, const [param_type](#) &`__p`)
- [param_type](#) [param](#) () const
- void [param](#) (const [param_type](#) &`__param`)
- void [reset](#) ()

Friends

- template<typename _RealType1, typename _CharT, typename _Traits >
std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::piecewise_linear_distribution< _RealType1 > &__x)
- bool operator== (const piecewise_linear_distribution &__d1, const piecewise_linear_distribution &__d2)
- template<typename _RealType1, typename _CharT, typename _Traits >
std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::piecewise_linear_distribution< _RealType1 > &__x)

4.664.1 Detailed Description

```
template<typename _RealType = double>
class std::piecewise_linear_distribution< _RealType >
```

A piecewise_linear_distribution random number distribution.

The formula for the piecewise linear probability mass function is

Definition at line 5750 of file random.h.

4.664.2 Member Typedef Documentation

4.664.2.1 template<typename _RealType = double> typedef _RealType std::piecewise_linear_distribution< _RealType >::result_type

The type of the range of the distribution.

Definition at line 5753 of file random.h.

4.664.3 Member Function Documentation

4.664.3.1 template<typename _RealType = double> std::vector<double> std::piecewise_linear_distribution< _RealType >::densities () const [inline]

Return a vector of the probability densities of the distribution.

Definition at line 5874 of file random.h.

References std::vector<_Tp, _Alloc>::empty().

4.664.3.2 template<typename _RealType = double> std::vector<_RealType> std::piecewise_linear_distribution< _RealType >::intervals () const [inline]

Return the intervals of the distribution.

Definition at line 5857 of file random.h.

References std::vector<_Tp, _Alloc>::empty().

4.664.3.3 `template<typename _RealType = double> result_type std::piecewise_linear_distribution< _RealType >::max () const [inline]`

Returns the least upper bound value of the distribution.

Definition at line 5909 of file random.h.

References `std::vector< _Tp, _Alloc >::back()`, and `std::vector< _Tp, _Alloc >::empty()`.

4.664.3.4 `template<typename _RealType = double> result_type std::piecewise_linear_distribution< _RealType >::min () const [inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 5899 of file random.h.

References `std::vector< _Tp, _Alloc >::empty()`, and `std::vector< _Tp, _Alloc >::front()`.

4.664.3.5 `template<typename _RealType = double> template<typename _UniformRandomNumberGenerator > result_type std::piecewise_linear_distribution< _RealType >::operator() (_UniformRandomNumberGenerator & __urng) [inline]`

Generating functions.

Definition at line 5920 of file random.h.

4.664.3.6 `template<typename _RealType = double> param_type std::piecewise_linear_distribution< _RealType >::param () const [inline]`

Returns the parameter set of the distribution.

Definition at line 5884 of file random.h.

4.664.3.7 `template<typename _RealType = double> void std::piecewise_linear_distribution< _RealType >::param (const param_type & __param) [inline]`

Sets the parameter set of the distribution.

Parameters

<code>__param</code>	The new parameter set of the distribution.
----------------------	--

Definition at line 5892 of file random.h.

4.664.3.8 `template<typename _RealType = double> void std::piecewise_linear_distribution< _RealType >::reset () [inline]`

Resets the distribution state.

Definition at line 5850 of file random.h.

4.664.4 Friends And Related Function Documentation

4.664.4.1 `template<typename _RealType = double> template<typename _RealType1, typename _CharT, typename _Traits> std::basic_ostream<_CharT, _Traits>& operator<< (std::basic_ostream<_CharT, _Traits> & __os, const std::piecewise_linear_distribution<_RealType1> & __x) [friend]`

Inserts a `piecewise_linear_distribution` random number distribution `__x` into the output stream `__os`.

Parameters

<code>__os</code>	An output stream.
<code>__x</code>	A <code>piecewise_linear_distribution</code> random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

4.664.4.2 `template<typename _RealType = double> bool operator==(const piecewise_linear_distribution<_RealType> & __d1, const piecewise_linear_distribution<_RealType> & __d2) [friend]`

Return true if two `piecewise_linear_distribution` have the same parameters.

Definition at line 5955 of file `random.h`.

4.664.4.3 `template<typename _RealType = double> template<typename _RealType1, typename _CharT, typename _Traits> std::basic_istream<_CharT, _Traits>& operator>> (std::basic_istream<_CharT, _Traits> & __is, std::piecewise_linear_distribution<_RealType1> & __x) [friend]`

Extracts a `piecewise_linear_distribution` random number distribution `__x` from the input stream `__is`.

Parameters

<code>__is</code>	An input stream.
<code>__x</code>	A <code>piecewise_linear_distribution</code> random number generator engine.

Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following file:

- [random.h](#)

4.665 `std::piecewise_linear_distribution<_RealType>::param_type` Struct Reference

Public Types

- typedef `piecewise_linear_distribution<_RealType>` **distribution_type**

Public Member Functions

- template<typename _InputIteratorB, typename _InputIteratorW >
param_type (_InputIteratorB __bfirst, _InputIteratorB __bend, _InputIteratorW __wbegin)
- template<typename _Func >
param_type (initializer_list<_RealType> __bl, _Func __fw)
- template<typename _Func >
param_type (size_t __nw, _RealType __xmin, _RealType __xmax, _Func __fw)
- **param_type** (const `param_type` &)=default
- `std::vector<double>` **densities** () const
- `std::vector<_RealType>` **intervals** () const
- `param_type` & **operator=** (const `param_type` &)=default

Friends

- bool **operator==** (const `param_type` &__p1, const `param_type` &__p2)
- class `piecewise_linear_distribution<_RealType>`

4.665.1 Detailed Description

```
template<typename _RealType = double>
struct std::piecewise_linear_distribution<_RealType>::param_type
```

Parameter type.

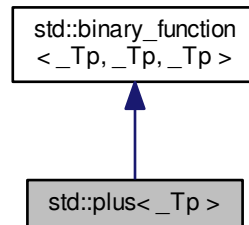
Definition at line 5759 of file random.h.

The documentation for this struct was generated from the following file:

- [random.h](#)

4.666 std::plus< _Tp > Struct Template Reference

Inheritance diagram for std::plus< _Tp >:



Public Types

- typedef `_Tp` [first_argument_type](#)
- typedef `_Tp` [result_type](#)
- typedef `_Tp` [second_argument_type](#)

Public Member Functions

- `_Tp operator()` (`const _Tp &__x`, `const _Tp &__y`) `const`

4.666.1 Detailed Description

```
template<typename _Tp>
struct std::plus< _Tp >
```

One of the [math functors](#).

Definition at line 167 of file `stl_function.h`.

4.666.2 Member Typedef Documentation

4.666.2.1 typedef `_Tp` `std::binary_function< _Tp, _Tp, _Tp >::first_argument_type` [\[inherited\]](#)

`first_argument_type` is the type of the first argument

Definition at line 121 of file `stl_function.h`.

4.666.2.2 `typedef _Tp std::binary_function<_Tp, _Tp, _Tp>::result_type` [inherited]

`result_type` is the return type

Definition at line 127 of file `stl_function.h`.

4.666.2.3 `typedef _Tp std::binary_function<_Tp, _Tp, _Tp>::second_argument_type` [inherited]

`second_argument_type` is the type of the second argument

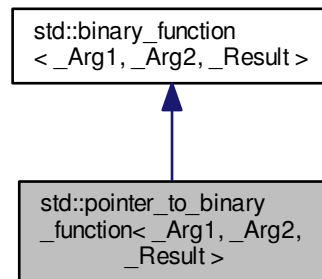
Definition at line 124 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl_function.h](#)

4.667 `std::pointer_to_binary_function<_Arg1, _Arg2, _Result>` Class Template Reference

Inheritance diagram for `std::pointer_to_binary_function<_Arg1, _Arg2, _Result>`:



Public Types

- `typedef _Arg1` [first_argument_type](#)
- `typedef _Result` [result_type](#)
- `typedef _Arg2` [second_argument_type](#)

Public Member Functions

- **`pointer_to_binary_function`** (`_Result(*__x)(_Arg1, _Arg2)`)
- `_Result` **`operator()`** (`_Arg1 __x, _Arg2 __y`) `const`

Protected Attributes

- `_Result(* _M_ptr)(_Arg1, _Arg2)`

4.667.1 Detailed Description

```
template<typename _Arg1, typename _Arg2, typename _Result>
class std::pointer_to_binary_function< _Arg1, _Arg2, _Result >
```

One of the [adaptors for function pointers](#).

Definition at line 800 of file `stl_function.h`.

4.667.2 Member Typedef Documentation

4.667.2.1 `template<typename _Arg1, typename _Arg2, typename _Result> typedef _Arg1 std::binary_function< _Arg1, _Arg2, _Result >::first_argument_type` [inherited]

`first_argument_type` is the type of the first argument

Definition at line 121 of file `stl_function.h`.

4.667.2.2 `template<typename _Arg1, typename _Arg2, typename _Result> typedef _Result std::binary_function< _Arg1, _Arg2, _Result >::result_type` [inherited]

`result_type` is the return type

Definition at line 127 of file `stl_function.h`.

4.667.2.3 `template<typename _Arg1, typename _Arg2, typename _Result> typedef _Arg2 std::binary_function< _Arg1, _Arg2, _Result >::second_argument_type` [inherited]

`second_argument_type` is the type of the second argument

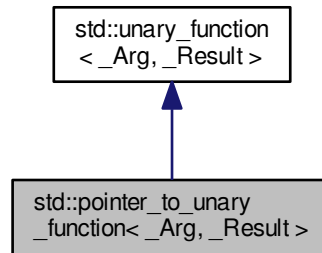
Definition at line 124 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [stl_function.h](#)

4.668 `std::pointer_to_unary_function< _Arg, _Result >` Class Template Reference

Inheritance diagram for `std::pointer_to_unary_function< _Arg, _Result >`:



Public Types

- typedef `_Arg` [argument_type](#)
- typedef `_Result` [result_type](#)

Public Member Functions

- **`pointer_to_unary_function`** (`_Result(*__x)(_Arg)`)
- `_Result` **`operator()`** (`_Arg __x`) const

Protected Attributes

- `_Result(* _M_ptr)(_Arg)`

4.668.1 Detailed Description

```
template<typename _Arg, typename _Result>
class std::pointer_to_unary_function< _Arg, _Result >
```

One of the [adaptors for function pointers](#).

Definition at line 775 of file `stl_function.h`.

4.668.2 Member Typedef Documentation

4.668.2.1 `template<typename _Arg, typename _Result> typedef _Arg std::unary_function< _Arg, _Result>::argument_type` [inherited]

`argument_type` is the type of the argument

Definition at line 108 of file `stl_function.h`.

4.668.2.2 `template<typename _Arg, typename _Result> typedef _Result std::unary_function< _Arg, _Result>::result_type` [inherited]

`result_type` is the return type

Definition at line 111 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [stl_function.h](#)

4.669 std::pointer_traits<_Ptr> Struct Template Reference

Inherits `std::__ptrtr_pointer_to<_Ptr>`.

Public Types

- `typedef __ptrtr_diff_type<_Ptr>::__type` [difference_type](#)
- `typedef __ptrtr_elt_type<_Ptr>::__type` [element_type](#)
- `typedef _Ptr` [pointer](#)
- `template<typename _Up>`
using **rebind** = `typename __ptrtr_rebind<_Ptr, _Up>::__type`

Static Public Member Functions

- `static _Ptr` **pointer_to** (`__element_type &__e`)

4.669.1 Detailed Description

```
template<typename _Ptr>
struct std::pointer_traits<_Ptr>
```

Uniform interface to all pointer-like types.

Definition at line 132 of file `ptr_traits.h`.

4.669.2 Member Typedef Documentation

4.669.2.1 `template<typename _Ptr> typedef __ptrtr_diff_type<_Ptr>::__type std::pointer_traits<_Ptr>::difference_type`

Type used to represent the difference between two pointers.

Definition at line 139 of file `ptr_traits.h`.

4.669.2.2 `template<typename _Ptr> typedef __ptrtr_elt_type<_Ptr>::__type std::pointer_traits<_Ptr>::element_type`

The type pointed to.

Definition at line 137 of file `ptr_traits.h`.

4.669.2.3 `template<typename _Ptr> typedef _Ptr std::pointer_traits<_Ptr>::pointer`

The pointer type.

Definition at line 135 of file `ptr_traits.h`.

The documentation for this struct was generated from the following file:

- [ptr_traits.h](#)

4.670 `std::pointer_traits<_Tp * >` Struct Template Reference

Public Types

- `typedef ptrdiff_t difference_type`
- `typedef _Tp element_type`
- `typedef _Tp * pointer`
- `template<typename _Up > using rebind = _Up *`

Static Public Member Functions

- static `pointer pointer_to` (typename `__ptrtr_not_void< element_type >::__type &__r`) noexcept

4.670.1 Detailed Description

```
template<typename _Tp>
struct std::pointer_traits<_Tp * >
```

Partial specialization for built-in pointers.

Definition at line 150 of file `ptr_traits.h`.

4.670.2 Member Typedef Documentation

4.670.2.1 `template<typename _Tp> typedef ptrdiff_t std::pointer_traits<_Tp*>::difference_type`

Type used to represent the difference between two pointers.

Definition at line 157 of file ptr_traits.h.

4.670.2.2 `template<typename _Tp> typedef _Tp std::pointer_traits<_Tp*>::element_type`

The type pointed to.

Definition at line 155 of file ptr_traits.h.

4.670.2.3 `template<typename _Tp> typedef _Tp* std::pointer_traits<_Tp*>::pointer`

The pointer type.





Definition at line 153 of file ptr_traits.h.

4.670.3 Member Function Documentation

4.670.3.1 `template<typename _Tp> static pointer std::pointer_traits<_Tp*>::pointer_to (typename __ptrtr_not_void<element_type>::__type &__r) [inline],[static],[noexcept]`

Obtain a pointer to an object.

Parameters

	A reference to an object of type <code>element_type</code>
<code>__</code> 	
	
<code>__</code> 	
<code>r</code>	

Returns

`addressof(__r)`

Definition at line 168 of file ptr_traits.h.

References `std::addressof()`.

The documentation for this struct was generated from the following file:

- [ptr_traits.h](#)

4.671 `std::poisson_distribution< _IntType >` Class Template Reference

Classes

- struct `param_type`

Public Types

- typedef `_IntType` `result_type`

Public Member Functions

- **`poisson_distribution`** (double `__mean=1.0`)
- **`poisson_distribution`** (const `param_type` &`__p`)
- template<typename `_ForwardIterator` , typename `_UniformRandomNumberGenerator` >
void **`__generate`** (`_ForwardIterator` `__f`, `_ForwardIterator` `__t`, `_UniformRandomNumberGenerator` &`__urng`)
- template<typename `_ForwardIterator` , typename `_UniformRandomNumberGenerator` >
void **`__generate`** (`_ForwardIterator` `__f`, `_ForwardIterator` `__t`, `_UniformRandomNumberGenerator` &`__urng`, const `param_type` &`__p`)
- template<typename `_UniformRandomNumberGenerator` >
void **`__generate`** (`result_type` *`__f`, `result_type` *`__t`, `_UniformRandomNumberGenerator` &`__urng`, const `param_type` &`__p`)
- `result_type` **`max`** () const
- double **`mean`** () const
- `result_type` **`min`** () const
- template<typename `_UniformRandomNumberGenerator` >
`result_type` **`operator()`** (`_UniformRandomNumberGenerator` &`__urng`)
- template<typename `_UniformRandomNumberGenerator` >
`result_type` **`operator()`** (`_UniformRandomNumberGenerator` &`__urng`, const `param_type` &`__p`)
- `param_type` **`param`** () const
- void **`param`** (const `param_type` &`__param`)
- void **`reset`** ()

Friends

- template<typename `_IntType1` , typename `_CharT` , typename `_Traits` >
std::basic_ostream< `_CharT`, `_Traits` > & **`operator<<`** (std::basic_ostream< `_CharT`, `_Traits` > &`__os`, const `std::poisson_distribution`< `_IntType1` > &`__x`)
- bool **`operator==`** (const `poisson_distribution` &`__d1`, const `poisson_distribution` &`__d2`)
- template<typename `_IntType1` , typename `_CharT` , typename `_Traits` >
std::basic_istream< `_CharT`, `_Traits` > & **`operator>>`** (std::basic_istream< `_CharT`, `_Traits` > &`__is`, `std::poisson_distribution`< `_IntType1` > &`__x`)

4.671.1 Detailed Description

```
template<typename _IntType = int>
class std::poisson_distribution<_IntType>
```

A discrete Poisson random number distribution.

The formula for the Poisson probability density function is $p(i|\mu) = \frac{\mu^i}{i!} e^{-\mu}$ where μ is the parameter of the distribution.

Definition at line 4432 of file random.h.

4.671.2 Member Typedef Documentation

4.671.2.1 `template<typename _IntType = int> typedef _IntType std::poisson_distribution<_IntType>::result_type`

The type of the range of the distribution.

Definition at line 4435 of file random.h.

4.671.3 Member Function Documentation

4.671.3.1 `template<typename _IntType = int> result_type std::poisson_distribution<_IntType>::max () const`
[inline]

Returns the least upper bound value of the distribution.

Definition at line 4526 of file random.h.

References std::max().

4.671.3.2 `template<typename _IntType = int> double std::poisson_distribution<_IntType>::mean () const` [inline]

Returns the distribution parameter `mean`.

Definition at line 4497 of file random.h.

4.671.3.3 `template<typename _IntType = int> result_type std::poisson_distribution<_IntType>::min () const`
[inline]

Returns the greatest lower bound value of the distribution.

Definition at line 4519 of file random.h.

4.671.3.4 `template<typename _IntType = int> template<typename _UniformRandomNumberGenerator> result_type`
`std::poisson_distribution<_IntType>::operator() (_UniformRandomNumberGenerator & __urng)` [inline]

Generating functions.

Definition at line 4534 of file random.h.

4.671.3.5 `template<typename _IntType = int> param_type std::poisson_distribution<_IntType>::param () const`
[inline]

Returns the parameter set of the distribution.

Definition at line 4504 of file random.h.

4.671.3.6 `template<typename _IntType = int> void std::poisson_distribution<_IntType>::param (const param_type &`
`__param)` [inline]

Sets the parameter set of the distribution.

Parameters

<code>__param</code>	The new parameter set of the distribution.
----------------------	--

Definition at line 4512 of file random.h.

4.671.3.7 `template<typename _IntType = int> void std::poisson_distribution<_IntType>::reset () [inline]`

Resets the distribution state.

Definition at line 4490 of file random.h.

4.671.4 Friends And Related Function Documentation

4.671.4.1 `template<typename _IntType = int> template<typename _IntType1, typename _CharT, typename _Traits > std::basic_ostream<_CharT, _Traits>& operator<< (std::basic_ostream<_CharT, _Traits> & __os, const std::poisson_distribution<_IntType1> & __x) [friend]`

Inserts a poisson_distribution random number distribution `__x` into the output stream `__os`.

Parameters

<code>__os</code>	An output stream.
<code>__x</code>	A poisson_distribution random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

4.671.4.2 `template<typename _IntType = int> bool operator== (const poisson_distribution<_IntType> & __d1, const poisson_distribution<_IntType> & __d2) [friend]`

Return true if two Poisson distributions have the same parameters and the sequences that would be generated are equal.

Definition at line 4570 of file random.h.

4.671.4.3 `template<typename _IntType = int> template<typename _IntType1, typename _CharT, typename _Traits > std::basic_istream<_CharT, _Traits>& operator>> (std::basic_istream<_CharT, _Traits> & __is, std::poisson_distribution<_IntType1> & __x) [friend]`

Extracts a poisson_distribution random number distribution `__x` from the input stream `__is`.

Parameters

<code>__is</code>	An input stream.
<code>__x</code>	A poisson_distribution random number generator engine.

Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following file:

- [random.h](#)

4.672 std::poisson_distribution< _IntType >::param_type Struct Reference

Public Types

- typedef [poisson_distribution](#)< _IntType > **distribution_type**

Public Member Functions

- **param_type** (double __mean=1.0)
- double **mean** () const

Friends

- bool **operator==** (const [param_type](#) &__p1, const [param_type](#) &__p2)
- class **poisson_distribution**< _IntType >

4.672.1 Detailed Description

```
template<typename _IntType = int>
struct std::poisson_distribution< _IntType >::param_type
```

Parameter type.

Definition at line 4441 of file random.h.

The documentation for this struct was generated from the following file:

- [random.h](#)

4.673 `std::priority_queue<_Tp, _Sequence, _Compare>` Class Template Reference

Public Types

- typedef `_Sequence::const_reference` **const_reference**
- typedef `_Sequence` **container_type**
- typedef `_Sequence::reference` **reference**
- typedef `_Sequence::size_type` **size_type**
- typedef `_Sequence::value_type` **value_type**

Public Member Functions

- [priority_queue](#) (const `_Compare` &__x, const `_Sequence` &__s)
- **priority_queue** (const `_Compare` &__x=`_Compare`(), `_Sequence` &&__s=`_Sequence`())
- template<typename `_InputIterator` >
[priority_queue](#) (`_InputIterator` __first, `_InputIterator` __last, const `_Compare` &__x, const `_Sequence` &__s)
- template<typename `_InputIterator` >
priority_queue (`_InputIterator` __first, `_InputIterator` __last, const `_Compare` &__x=`_Compare`(), `_Sequence` &&__s=`_Sequence`())
- template<typename... `_Args`>
void **emplace** (`_Args` &&...__args)
- bool [empty](#) () const
- void [pop](#) ()
- void [push](#) (const `value_type` &__x)
- void **push** (`value_type` &&__x)
- `size_type` [size](#) () const
- void **swap** ([priority_queue](#) &__pq) noexcept(noexcept(swap(c, __pq.c))&&noexcept(swap(comp, __pq.comp)))
- const_reference [top](#) () const

Protected Attributes

- `_Sequence` **c**
- `_Compare` **comp**

4.673.1 Detailed Description

```
template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename _Sequence::value_type>>
class std::priority_queue<_Tp, _Sequence, _Compare>
```

A standard container automatically sorting its contents.

Template Parameters

<code>_Tp</code>	Type of element.
<code>_Sequence</code>	Type of underlying sequence, defaults to <code>vector<_Tp></code> .
<code>_Compare</code>	Comparison function object type, defaults to <code>less<_Sequence::value_type></code> .

This is not a true container, but an *adaptor*. It holds another container, and provides a wrapper interface to that container. The wrapper is what enforces priority-based sorting and queue behavior. Very few of the standard container/sequence interface requirements are met (e.g., iterators).

The second template parameter defines the type of the underlying sequence/container. It defaults to `std::vector`, but it can be any type that supports `front()`, `push_back`, `pop_back`, and random-access iterators, such as `std::deque` or an appropriate user-defined type.

The third template parameter supplies the means of making priority comparisons. It defaults to `less<value_type>` but can be anything defining a strict weak ordering.

Members not found in *normal* containers are `container_type`, which is a typedef for the second Sequence parameter, and `push`, `pop`, and `top`, which are standard queue operations.

Note

No equality/comparison operators are provided for `priority_queue`.

Sorting of the elements takes place as they are added to, and removed from, the `priority_queue` using the `priority_queue`'s member functions. If you access the elements by other means, and change their data such that the sorting order would be different, the `priority_queue` will not re-sort the elements for you. (How could it know to do so?)

Definition at line 370 of file `stl_queue.h`.

4.673.2 Constructor & Destructor Documentation

4.673.2.1 `template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename _Sequence::value_type>> std::priority_queue<_Tp, _Sequence, _Compare>::priority_queue (const _Compare &__x, const _Sequence &__s) [inline], [explicit]`

Default constructor creates no elements.

Definition at line 405 of file `stl_queue.h`.

References `std::queue<_Tp, _Sequence>::c`, `std::make_heap()`, and `std::move()`.

4.673.2.2 `template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename _Sequence::value_type>> template<typename _InputIterator> std::priority_queue<_Tp, _Sequence, _Compare>::priority_queue (_InputIterator __first, _InputIterator __last, const _Compare &__x, const _Sequence &__s) [inline]`

Builds a queue from a range.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__x</code>	A comparison functor describing a strict weak ordering.
<code>__s</code>	An initial sequence with which to start.

Begins by copying `__s`, inserting a copy of the elements from `[first,last)` into the copy of `__s`, then ordering the copy according to `__x`.

For more information on function objects, see the documentation on [functor base classes](#).

Definition at line 445 of file `stl_queue.h`.

References `std::queue<_Tp, _Sequence >::c`, `std::make_heap()`, and `std::move()`.

4.673.3 Member Function Documentation

4.673.3.1 `template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename
_Sequence::value_type>> bool std::priority_queue<_Tp, _Sequence, _Compare >::empty () const
[inline]`

Returns true if the queue is empty.

Definition at line 471 of file `stl_queue.h`.

Referenced by `__gnu_parallel::multiseq_partition()`, and `__gnu_parallel::multiseq_selection()`.

4.673.3.2 `template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename
_Sequence::value_type>> void std::priority_queue<_Tp, _Sequence, _Compare >::pop () [inline]`

Removes first element.

This is a typical queue operation. It shrinks the queue by one. The time complexity of the operation depends on the underlying sequence.

Note that no data is returned, and if the first element's data is needed, it should be retrieved before `pop()` is called.

Definition at line 534 of file `stl_queue.h`.

References `std::pop_heap()`, and `std::swap()`.

Referenced by `__gnu_parallel::multiseq_partition()`, and `__gnu_parallel::multiseq_selection()`.

4.673.3.3 `template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename
_Sequence::value_type>> void std::priority_queue<_Tp, _Sequence, _Compare >::push (const value_type & __x
) [inline]`

Add data to the queue.

Parameters

<code>__x</code>	Data to be added.
------------------	-------------------

This is a typical queue operation. The time complexity of the operation depends on the underlying sequence.

Definition at line 499 of file `stl_queue.h`.

References `std::move()`, `std::queue<_Tp, _Sequence>::push()`, and `std::push_heap()`.

Referenced by `__gnu_parallel::multiseq_partition()`, and `__gnu_parallel::multiseq_selection()`.

4.673.3.4 `template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename _Sequence::value_type>> size_type std::priority_queue<_Tp, _Sequence, _Compare>::size () const`
`[inline]`

Returns the number of elements in the queue.

Definition at line 476 of file `stl_queue.h`.

4.673.3.5 `template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename _Sequence::value_type>> const_reference std::priority_queue<_Tp, _Sequence, _Compare>::top () const`
`[inline]`

Returns a read-only (constant) reference to the data at the first element of the queue.

Definition at line 484 of file `stl_queue.h`.

Referenced by `__gnu_parallel::multiseq_partition()`, and `__gnu_parallel::multiseq_selection()`.

The documentation for this class was generated from the following file:

- [stl_queue.h](#)

4.674 `std::queue<_Tp, _Sequence>` Class Template Reference

Public Types

- `typedef _Sequence::const_reference` **const_reference**
- `typedef _Sequence` **container_type**
- `typedef _Sequence::reference` **reference**
- `typedef _Sequence::size_type` **size_type**
- `typedef _Sequence::value_type` **value_type**

Public Member Functions

- [queue](#) (`const _Sequence &__c`)
- **queue** (`_Sequence &&__c=_Sequence()`)
- reference [back](#) ()
- `const_reference` [back](#) () `const`
- `template<typename... _Args>`
`void` **emplace** (`_Args &&...__args`)
- `bool` [empty](#) () `const`
- reference [front](#) ()
- `const_reference` [front](#) () `const`
- `void` [pop](#) ()
- `void` [push](#) (`const value_type &__x`)
- `void` **push** (`value_type &&__x`)
- `size_type` [size](#) () `const`
- `void` **swap** ([queue](#) &__q) `noexcept(noexcept(swap(c, __q.c)))`

Protected Attributes

- `_Sequence` [c](#)

Friends

- `template<typename _Tp1, typename _Seq1 >`
`bool operator< (const queue< _Tp1, _Seq1 > &, const queue< _Tp1, _Seq1 > &)`
- `template<typename _Tp1, typename _Seq1 >`
`bool operator== (const queue< _Tp1, _Seq1 > &, const queue< _Tp1, _Seq1 > &)`

4.674.1 Detailed Description

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
class std::queue< _Tp, _Sequence >
```

A standard container giving FIFO behavior.

Template Parameters

<code>_Tp</code>	Type of element.
<code>_Sequence</code>	Type of underlying sequence, defaults to <code>deque<_Tp></code> .

Meets many of the requirements of a [container](#), but does not define anything to do with iterators. Very few of the other standard container interfaces are defined.

This is not a true container, but an *adaptor*. It holds another container, and provides a wrapper interface to that container. The wrapper is what enforces strict first-in-first-out queue behavior.

The second template parameter defines the type of the underlying sequence/container. It defaults to `std::deque`, but it can be any type that supports `front`, `back`, `push_back`, and `pop_front`, such as `std::list` or an appropriate user-defined type.

Members not found in *normal* containers are `container_type`, which is a typedef for the second `Sequence` parameter, and `push` and `pop`, which are standard queue/FIFO operations.

Definition at line 96 of file `stl_queue.h`.

4.674.2 Constructor & Destructor Documentation

4.674.2.1 `template<typename _Tp, typename _Sequence = deque<_Tp>> std::queue< _Tp, _Sequence >::queue (const _Sequence &_c) [inline], [explicit]`

Default constructor creates no elements.

Definition at line 141 of file `stl_queue.h`.

References `std::queue< _Tp, _Sequence >::c`, and `std::move()`.

4.674.3 Member Function Documentation

4.674.3.1 `template<typename _Tp, typename _Sequence = deque<_Tp>> reference std::queue<_Tp, _Sequence>::back ()`
`[inline]`

Returns a read/write reference to the data at the last element of the queue.

Definition at line 188 of file `stl_queue.h`.

4.674.3.2 `template<typename _Tp, typename _Sequence = deque<_Tp>> const_reference std::queue<_Tp, _Sequence>::back () const` `[inline]`

Returns a read-only (constant) reference to the data at the last element of the queue.

Definition at line 199 of file `stl_queue.h`.

4.674.3.3 `template<typename _Tp, typename _Sequence = deque<_Tp>> bool std::queue<_Tp, _Sequence>::empty ()`
`const` `[inline]`

Returns true if the queue is empty.

Definition at line 153 of file `stl_queue.h`.

4.674.3.4 `template<typename _Tp, typename _Sequence = deque<_Tp>> reference std::queue<_Tp, _Sequence>::front ()`
`[inline]`

Returns a read/write reference to the data at the first element of the queue.

Definition at line 166 of file `stl_queue.h`.

4.674.3.5 `template<typename _Tp, typename _Sequence = deque<_Tp>> const_reference std::queue<_Tp, _Sequence>::front () const` `[inline]`

Returns a read-only (constant) reference to the data at the first element of the queue.

Definition at line 177 of file `stl_queue.h`.

4.674.3.6 `template<typename _Tp, typename _Sequence = deque<_Tp>> void std::queue<_Tp, _Sequence>::pop ()`
`[inline]`

Removes first element.

This is a typical queue operation. It shrinks the queue by one. The time complexity of the operation depends on the underlying sequence.

Note that no data is returned, and if the first element's data is needed, it should be retrieved before `pop()` is called.

Definition at line 241 of file `stl_queue.h`.

References `std::queue<_Tp, _Sequence>::c`, and `std::swap()`.

4.674.3.7 `template<typename _Tp, typename _Sequence = deque<_Tp>> void std::queue<_Tp, _Sequence>::push (const value_type &__x)` `[inline]`

Add data to the end of the queue.

Parameters

\leftrightarrow	Data to be added.
\times	

This is a typical queue operation. The function creates an element at the end of the queue and assigns the given data to it. The time complexity of the operation depends on the underlying sequence.

Definition at line 215 of file `stl_queue.h`.

References `std::move()`.

Referenced by `std::priority_queue<_Tp, _Sequence, _Compare>::push()`.

4.674.3.8 `template<typename _Tp, typename _Sequence = deque<_Tp>> size_type std::queue<_Tp, _Sequence>::size ()`
`const [inline]`

Returns the number of elements in the queue.

Definition at line 158 of file `stl_queue.h`.

4.674.4 Member Data Documentation

4.674.4.1 `template<typename _Tp, typename _Sequence = deque<_Tp>> _Sequence std::queue<_Tp, _Sequence>::c`
`[protected]`

'c' is the underlying container. Maintainers wondering why this isn't uglified as per style guidelines should note that this name is specified in the standard, [23.2.3.1]. (Why? Presumably for the same reason that it's protected instead of private: to allow derivation. But none of the other containers allow for derivation. Odd.)

Definition at line 129 of file `stl_queue.h`.

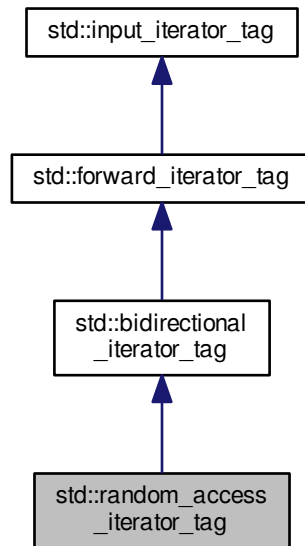
Referenced by `std::operator<()`, `std::operator==()`, `std::queue<_Tp, _Sequence>::pop()`, `std::priority_queue<_Tp, _Sequence, _Compare>::priority_queue()`, and `std::queue<_Tp, _Sequence>::queue()`.

The documentation for this class was generated from the following file:

- [stl_queue.h](#)

4.675 `std::random_access_iterator_tag` Struct Reference

Inheritance diagram for `std::random_access_iterator_tag`:



4.675.1 Detailed Description

Random-access iterators support a superset of bidirectional iterator operations.

Definition at line 103 of file `stl_iterator_base_types.h`.

The documentation for this struct was generated from the following file:

- [stl_iterator_base_types.h](#)

4.676 `std::random_device` Class Reference

Public Types

- typedef unsigned int [result_type](#)

Public Member Functions

- **random_device** (const [std::string](#) & __token="mt19937")
- **random_device** (const [random_device](#) &)=delete
- double **entropy** () const noexcept
- [result_type](#) **operator**() ()
- void **operator=** (const [random_device](#) &)=delete

Static Public Member Functions

- static constexpr [result_type](#) **max** ()
- static constexpr [result_type](#) **min** ()

4.676.1 Detailed Description

A standard interface to a platform-specific non-deterministic random number generator (if any are available).

Definition at line 1577 of file random.h.

4.676.2 Member Typedef Documentation

4.676.2.1 typedef unsigned int [std::random_device::result_type](#)

The type of the generated random value.

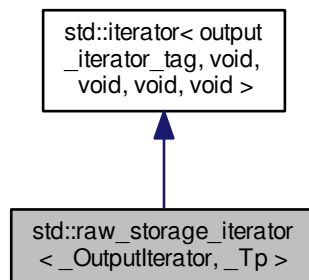
Definition at line 1581 of file random.h.

The documentation for this class was generated from the following file:

- [random.h](#)

4.677 [std::raw_storage_iterator< _OutputIterator, _Tp >](#) Class Template Reference

Inheritance diagram for [std::raw_storage_iterator< _OutputIterator, _Tp >](#):



Public Types

- typedef void [difference_type](#)
- typedef [output_iterator_tag](#) [iterator_category](#)
- typedef void [pointer](#)
- typedef void [reference](#)
- typedef void [value_type](#)

Public Member Functions

- **`raw_storage_iterator`** (`_OutputIterator __x`)
- [raw_storage_iterator](#) & **`operator*`** ()
- [raw_storage_iterator](#)< `_OutputIterator, _Tp` > & **`operator++`** ()
- [raw_storage_iterator](#)< `_OutputIterator, _Tp` > **`operator++`** (int)
- [raw_storage_iterator](#) & **`operator=`** (const `_Tp` & `__element`)

Protected Attributes

- `_OutputIterator` **`_M_iter`**

4.677.1 Detailed Description

```
template<class _OutputIterator, class _Tp>
class std::raw_storage_iterator< _OutputIterator, _Tp >
```

This iterator class lets algorithms store their results into uninitialized memory.

Definition at line 68 of file `stl_raw_storage_iter.h`.

4.677.2 Member Typedef Documentation

4.677.2.1 typedef void `std::iterator< output_iterator_tag , void , void , void , void >::difference_type` `[inherited]`

Distance between iterators is represented as this type.

Definition at line 125 of file `stl_iterator_base_types.h`.

4.677.2.2 typedef [output_iterator_tag](#) `std::iterator< output_iterator_tag , void , void , void , void >::iterator_category` `[inherited]`

One of the [tag types](#).

Definition at line 121 of file `stl_iterator_base_types.h`.

4.677.2.3 `typedef void std::iterator< output_iterator_tag, void, void, void, void >::pointer` [inherited]

This type represents a pointer-to-value_type.

Definition at line 127 of file `stl_iterator_base_types.h`.

4.677.2.4 `typedef void std::iterator< output_iterator_tag, void, void, void, void >::reference` [inherited]

This type represents a reference-to-value_type.

Definition at line 129 of file `stl_iterator_base_types.h`.

4.677.2.5 `typedef void std::iterator< output_iterator_tag, void, void, void, void >::value_type` [inherited]

The type "pointed to" by the iterator.

Definition at line 123 of file `stl_iterator_base_types.h`.

The documentation for this class was generated from the following file:

- [stl_raw_storage_iter.h](#)

4.678 `std::regex_error` Class Reference

Inherits `runtime_error`.

Public Member Functions

- [regex_error](#) ([regex_constants::error_type](#) __ecode)
- [regex_constants::error_type](#) code () const

4.678.1 Detailed Description

A regular expression exception class.

The regular expression library throws objects of this class on error.

Definition at line 135 of file `regex_error.h`.

4.678.2 Constructor & Destructor Documentation

4.678.2.1 `std::regex_error::regex_error(regex_constants::error_type __ecode)` [explicit]

Constructs a `regex_error` object.

Parameters

<code>__ecode</code>	the regex error code.
----------------------	-----------------------

4.678.3 Member Function Documentation

4.678.3.1 `regex_constants::error_type` `std::regex_error::code () const` `[inline]`

Gets the regex error code.

Returns

the regex error code.

Definition at line 156 of file `regex_error.h`.

The documentation for this class was generated from the following file:

- [regex_error.h](#)

4.679 `std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >` Class Template Reference

Public Types

- typedef `std::ptrdiff_t` **difference_type**
- typedef [std::forward_iterator_tag](#) **iterator_category**
- typedef const [value_type](#) * **pointer**
- typedef const [value_type](#) & **reference**
- typedef [basic_regex< _Ch_type, _Rx_traits >](#) **regex_type**
- typedef [match_results< _Bi_iter >](#) **value_type**

Public Member Functions

- [regex_iterator](#) ()
- [regex_iterator](#) ([_Bi_iter](#) __a, [_Bi_iter](#) __b, const [regex_type](#) &__re, [regex_constants::match_flag_type](#) __m↵
m=[regex_constants::match_default](#))
- [regex_iterator](#) (const [regex_iterator](#) &__rhs)=default
- bool [operator!=](#) (const [regex_iterator](#) &__rhs) const
- const [value_type](#) & [operator*](#) () const
- [regex_iterator](#) & [operator++](#) ()
- [regex_iterator](#) [operator++](#) (int)
- const [value_type](#) * [operator->](#) () const
- [regex_iterator](#) & [operator=](#) (const [regex_iterator](#) &__rhs)=default
- bool [operator==](#) (const [regex_iterator](#) &__rhs) const

4.679.1 Detailed Description

```
template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>>
class std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >
```

An iterator adaptor that will provide repeated calls of `regex_search` over a range until no more matches remain.

Definition at line 2420 of file `regex.h`.

4.679.2 Constructor & Destructor Documentation

```
4.679.2.1 template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename
_Rx_traits = regex_traits<_Ch_type>> std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_iterator ( )
[inline]
```

Provides a singular iterator, useful for indicating one-past-the-end of a range.

Definition at line 2434 of file `regex.h`.

```
4.679.2.2 template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename
_Rx_traits = regex_traits<_Ch_type>> std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_iterator
( _Bi_iter __a, _Bi_iter __b, const regex_type & __re, regex_constants::match_flag_type __m =
regex_constants::match_default ) [inline]
```

Constructs a `regex_iterator`...

Parameters

<code>__a</code>	[IN] The start of a text range to search.
<code>__b</code>	[IN] One-past-the-end of the text range to search.
<code>__re</code>	[IN] The regular expression to match.
<code>__m</code>	[IN] Policy flags for match rules.

Definition at line 2445 of file `regex.h`.

References `std::regex_search()`.

```
4.679.2.3 template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits
= regex_traits<_Ch_type>> std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_iterator ( const
regex_iterator< _Bi_iter, _Ch_type, _Rx_traits > & __rhs ) [default]
```

Copy constructs a `regex_iterator`.

4.679.3 Member Function Documentation

4.679.3.1 `template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>> bool std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator!= (const regex_iterator< _Bi_iter, _Ch_type, _Rx_traits > & __rhs) const [inline]`

Tests the inequivalence of two regex iterators.

Definition at line 2475 of file regex.h.

4.679.3.2 `template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>> const value_type& std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator* () const [inline]`

Dereferences a regex_iterator.

Definition at line 2482 of file regex.h.

4.679.3.3 `template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>> regex_iterator& std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator++ ()`

Increments a regex_iterator.

4.679.3.4 `template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>> regex_iterator std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator++ (int) [inline]`

Postincrements a regex_iterator.

Definition at line 2502 of file regex.h.

4.679.3.5 `template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>> const value_type* std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator-> () const [inline]`

Selects a regex_iterator member.

Definition at line 2489 of file regex.h.

4.679.3.6 `template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>> regex_iterator& std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator= (const regex_iterator< _Bi_iter, _Ch_type, _Rx_traits > & __rhs) [default]`

Assigns one regex_iterator to another.

```
4.679.3.7 template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits
= regex_traits<_Ch_type>> bool std::regex_iterator<_Bi_iter, _Ch_type, _Rx_traits >::operator==( const
regex_iterator<_Bi_iter, _Ch_type, _Rx_traits > & __rhs ) const
```

Tests the equivalence of two regex iterators.

The documentation for this class was generated from the following file:

- [regex.h](#)

4.680 std::regex_token_iterator<_Bi_iter, _Ch_type, _Rx_traits > Class Template Reference

Public Types

- typedef std::ptrdiff_t **difference_type**
- typedef [std::forward_iterator_tag](#) **iterator_category**
- typedef const [value_type](#) * **pointer**
- typedef const [value_type](#) & **reference**
- typedef [basic_regex](#)<_Ch_type, _Rx_traits > **regex_type**
- typedef [sub_match](#)<_Bi_iter > **value_type**

Public Member Functions

- [regex_token_iterator](#) ()
- [regex_token_iterator](#) (_Bi_iter __a, _Bi_iter __b, const [regex_type](#) &__re, int __submatch=0, [regex_constants::match_flag_type](#) __m=[regex_constants::match_default](#))
- [regex_token_iterator](#) (_Bi_iter __a, _Bi_iter __b, const [regex_type](#) &__re, const [std::vector](#)< int > &__submatches, [regex_constants::match_flag_type](#) __m=[regex_constants::match_default](#))
- [regex_token_iterator](#) (_Bi_iter __a, _Bi_iter __b, const [regex_type](#) &__re, initializer_list< int > __submatches, [regex_constants::match_flag_type](#) __m=[regex_constants::match_default](#))
- template<std::size_t _Nm>
[regex_token_iterator](#) (_Bi_iter __a, _Bi_iter __b, const [regex_type](#) &__re, const int(&__submatches)[_Nm], [regex_constants::match_flag_type](#) __m=[regex_constants::match_default](#))
- [regex_token_iterator](#) (const [regex_token_iterator](#) &__rhs)
- bool [operator!=](#) (const [regex_token_iterator](#) &__rhs) const
- const [value_type](#) & [operator*](#) () const
- [regex_token_iterator](#) & [operator++](#) ()
- [regex_token_iterator](#) [operator++](#) (int)
- const [value_type](#) * [operator->](#) () const
- [regex_token_iterator](#) & [operator=](#) (const [regex_token_iterator](#) &__rhs)
- bool [operator==](#) (const [regex_token_iterator](#) &__rhs) const

4.680.1 Detailed Description

```
template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>>
class std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >
```

Iterates over submatches in a range (or *splits* a text string).

The purpose of this iterator is to enumerate all, or all specified, matches of a regular expression within a text range. The dereferenced value of an iterator of this class is a std::sub_match object.

Definition at line 2535 of file regex.h.

4.680.2 Constructor & Destructor Documentation

```
4.680.2.1 template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits
= regex_traits<_Ch_type>> std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_token_iterator
( ) [inline]
```

Default constructs a regex_token_iterator.

A default-constructed regex_token_iterator is a singular iterator that will compare equal to the one-past-the-end value for any iterator of the same type.

Definition at line 2553 of file regex.h.

```
4.680.2.2 template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits
= regex_traits<_Ch_type>> std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_token_iterator
( _Bi_iter __a, _Bi_iter __b, const regex_type & __re, int __submatch = 0, regex_constants::match_flag_type
__m = regex_constants::match_default ) [inline]
```

Constructs a regex_token_iterator...

Parameters

<code>__a</code>	[IN] The start of the text to search.
<code>__b</code>	[IN] One-past-the-end of the text to search.
<code>__re</code>	[IN] The regular expression to search for.
<code>__submatch</code>	[IN] Which submatch to return. There are some special values for this parameter: <ul style="list-style-type: none"> • -1 each enumerated subexpression does NOT match the regular expression (aka field splitting) • 0 the entire string matching the subexpression is returned for each match within the text. • >0 enumerates only the indicated subexpression from a match within the text.
<code>__m</code>	[IN] Policy flags for match rules.

Definition at line 2575 of file regex.h.

```
4.680.2.3  template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits
           = regex_traits<_Ch_type>> std::regex_token_iterator<_Bi_iter, _Ch_type, _Rx_traits>::regex_token_iterator
           ( _Bi_iter __a, _Bi_iter __b, const regex_type & __re, const std::vector< int > & __submatches,
             regex_constants::match_flag_type __m = regex_constants::match_default ) [inline]
```

Constructs a `regex_token_iterator`...

Parameters

<code>__a</code>	[IN] The start of the text to search.
<code>__b</code>	[IN] One-past-the-end of the text to search.
<code>__re</code>	[IN] The regular expression to search for.
<code>__submatches</code>	[IN] A list of subexpressions to return for each regular expression match within the text.
<code>__m</code>	[IN] Policy flags for match rules.

Definition at line 2591 of file `regex.h`.

```
4.680.2.4  template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename
           _Rx_traits = regex_traits<_Ch_type>> std::regex_token_iterator<_Bi_iter, _Ch_type, _Rx_traits
           >::regex_token_iterator ( _Bi_iter __a, _Bi_iter __b, const regex_type & __re, initializer_list< int >
           __submatches, regex_constants::match_flag_type __m = regex_constants::match_default ) [inline]
```

Constructs a `regex_token_iterator`...

Parameters

<code>__a</code>	[IN] The start of the text to search.
<code>__b</code>	[IN] One-past-the-end of the text to search.
<code>__re</code>	[IN] The regular expression to search for.
<code>__submatches</code>	[IN] A list of subexpressions to return for each regular expression match within the text.
<code>__m</code>	[IN] Policy flags for match rules.

Definition at line 2608 of file `regex.h`.

```
4.680.2.5  template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits
           = regex_traits<_Ch_type>> template<std::size_t _Nm> std::regex_token_iterator<_Bi_iter, _Ch_type, _Rx_traits
           >::regex_token_iterator ( _Bi_iter __a, _Bi_iter __b, const regex_type & __re, const int(&) __submatches[_Nm],
           regex_constants::match_flag_type __m = regex_constants::match_default ) [inline]
```

Constructs a `regex_token_iterator`...

Parameters

<code>__a</code>	[IN] The start of the text to search.
<code>__b</code>	[IN] One-past-the-end of the text to search.
<code>__re</code>	[IN] The regular expression to search for.
<code>__submatches</code>	[IN] A list of subexpressions to return for each regular expression match within the text.
<code>__m</code>	[IN] Policy flags for match rules.

Definition at line 2626 of file regex.h.

```
4.680.2.6 template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits
= regex_traits<_Ch_type>> std::regex_token_iterator<_Bi_iter, _Ch_type, _Rx_traits>::regex_token_iterator
( const regex_token_iterator<_Bi_iter, _Ch_type, _Rx_traits> & __rhs ) [inline]
```

Copy constructs a regex_token_iterator.

Parameters

<code>__rhs</code>	[IN] A regex_token_iterator to copy.
--------------------	--------------------------------------

Definition at line 2639 of file regex.h.

4.680.3 Member Function Documentation

```
4.680.3.1 template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits
= regex_traits<_Ch_type>> bool std::regex_token_iterator<_Bi_iter, _Ch_type, _Rx_traits>::operator!=( const
regex_token_iterator<_Bi_iter, _Ch_type, _Rx_traits> & __rhs ) const [inline]
```

Compares a regex_token_iterator to another for inequality.

Definition at line 2661 of file regex.h.

```
4.680.3.2 template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits
= regex_traits<_Ch_type>> const value_type& std::regex_token_iterator<_Bi_iter, _Ch_type, _Rx_traits
>::operator*( ) const [inline]
```

Dereferences a regex_token_iterator.

Definition at line 2668 of file regex.h.

```
4.680.3.3 template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits
= regex_traits<_Ch_type>> regex_token_iterator& std::regex_token_iterator<_Bi_iter, _Ch_type, _Rx_traits
>::operator++( )
```

Increments a regex_token_iterator.

```
4.680.3.4 template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits
= regex_traits<_Ch_type>> regex_token_iterator std::regex_token_iterator<_Bi_iter, _Ch_type, _Rx_traits
>::operator++( int ) [inline]
```

Postincrements a regex_token_iterator.

Definition at line 2688 of file regex.h.

```
4.680.3.5  template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits
           = regex_traits<_Ch_type>> const value_type* std::regex_token_iterator<_Bi_iter, _Ch_type, _Rx_traits
           >::operator-> ( ) const    [inline]
```

Selects a `regex_token_iterator` member.

Definition at line 2675 of file `regex.h`.

```
4.680.3.6  template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits
           = regex_traits<_Ch_type>> regex_token_iterator& std::regex_token_iterator<_Bi_iter, _Ch_type, _Rx_traits
           >::operator= ( const regex_token_iterator<_Bi_iter, _Ch_type, _Rx_traits > & __rhs )
```

Assigns a `regex_token_iterator` to another.

Parameters

<code>__rhs</code>	[IN] A <code>regex_token_iterator</code> to copy.
--------------------	---

```
4.680.3.7  template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits
           = regex_traits<_Ch_type>> bool std::regex_token_iterator<_Bi_iter, _Ch_type, _Rx_traits >::operator== ( const
           regex_token_iterator<_Bi_iter, _Ch_type, _Rx_traits > & __rhs ) const
```

Compares a `regex_token_iterator` to another for equality.

The documentation for this class was generated from the following file:

- [regex.h](#)

4.681 std::regex_traits<_Ch_type> Struct Template Reference

Public Types

- typedef `_RegexMask` **char_class_type**
- typedef `_Ch_type` **char_type**
- typedef `std::locale` **locale_type**
- typedef `std::basic_string< char_type >` **string_type**

Public Member Functions

- [regex_traits](#) ()
- [locale_type](#) `getloc` () const
- [locale_type](#) `imbue` ([locale_type](#) __loc)
- bool [isctype](#) (`_Ch_type` __c, `char_class_type` __f) const
- template<typename `_Fwd_iter` >
`char_class_type` [lookup_classname](#) (`_Fwd_iter` __first, `_Fwd_iter` __last, bool __icase=false) const

- template<typename _Fwd_iter >
string_type lookup_collatename (_Fwd_iter __first, _Fwd_iter __last) const
- template<typename _Fwd_iter >
string_type transform (_Fwd_iter __first, _Fwd_iter __last) const
- template<typename _Fwd_iter >
string_type transform_primary (_Fwd_iter __first, _Fwd_iter __last) const
- char_type translate (char_type __c) const
- char_type translate_nocase (char_type __c) const
- int value (_Ch_type __ch, int __radix) const

Static Public Member Functions

- static std::size_t length (const char_type *__p)

Protected Attributes

- locale_type _M_locale

4.681.1 Detailed Description

```
template<typename _Ch_type>
struct std::regex_traits<_Ch_type>
```

Describes aspects of a regular expression.

A regular expression traits class that satisfies the requirements of section [28.7].

The class regex is parameterized around a set of related types and functions used to complete the definition of its semantics. This class satisfies the requirements of such a traits class.

Definition at line 91 of file regex.h.

4.681.2 Constructor & Destructor Documentation

4.681.2.1 template<typename _Ch_type> std::regex_traits<_Ch_type>::regex_traits () [inline]

Constructs a default traits object.

Definition at line 171 of file regex.h.

4.681.3 Member Function Documentation

4.681.3.1 template<typename _Ch_type> locale_type std::regex_traits<_Ch_type>::getloc () const [inline]

Gets a copy of the current locale in use by the regex_traits object.

Definition at line 384 of file regex.h.

References std::regex_constants::awk, std::regex_constants::basic, std::regex_constants::collate, std::regex_constants::ECMAScript, std::regex_constants::egrep, std::regex_constants::extended, std::regex_constants::grep, std::regex_constants::icase, std::regex_constants::nosubs, and std::regex_constants::optimize.

4.681.3.2 template<typename _Ch_type> locale_type std::regex_traits<_Ch_type>::imbue (locale_type __loc) [inline]

Imbues the regex_traits object with a copy of a new locale.

Parameters

<code>__loc</code>	A locale.
--------------------	-----------

Returns

a copy of the previous locale in use by the `regex_traits` object.

Note

Calling `imbue` with a different locale than the one currently in use invalidates all cached data held by `*this`.

Definition at line 373 of file `regex.h`.

References `std::swap()`.

4.681.3.3 `template<typename _Ch_type > bool std::regex_traits<_Ch_type>::isctype (_Ch_type __c, char_class_type __f) const`

Determines if `c` is a member of an identified class.

Parameters

<code>__c</code>	a character.
<code>__f</code>	a class type (as returned from <code>lookup_classname</code>).

Returns

true if the character `__c` is a member of the classification represented by `__f`, false otherwise.

Exceptions

<code>std::bad_cast</code>	if the current locale does not have a ctype facet.
----------------------------	--

4.681.3.4 `template<typename _Ch_type > static std::size_t std::regex_traits<_Ch_type>::length (const char_type * __p) [inline], [static]`

Gives the length of a C-style string starting at `__p`.

Parameters

<code>__p</code>	a pointer to the start of a character sequence.
------------------	---

Returns

the number of characters between *__p and the first default-initialized value of type `char_type`. In other words, uses the C-string algorithm for determining the length of a sequence of characters.

Definition at line 184 of file `regex.h`.

4.681.3.5 `template<typename _Ch_type > template<typename _Fwd_iter > char_class_type std::regex_traits<_Ch_type >::lookup_classname (_Fwd_iter __first, _Fwd_iter __last, bool __icase = false) const`

Maps one or more characters to a named character classification.

Parameters

<code>__first</code>	beginning of the character sequence.
<code>__last</code>	one-past-the-end of the character sequence.
<code>__icase</code>	ignores the case of the classification name.

Returns

an unspecified value that represents the character classification named by the character sequence designated by the iterator range `[__first, __last)`. If `icase` is true, the returned mask identifies the classification regardless of the case of the characters to be matched (for example, `[[:lower:]]` is the same as `[[:alpha:]]`), otherwise a case-dependent classification is returned. The value returned shall be independent of the case of the characters in the character sequence. If the name is not recognized then returns a value that compares equal to 0.

At least the following names (or their wide-character equivalent) are supported.

- `d`
- `w`
- `s`
- `alnum`
- `alpha`
- `blank`
- `cntrl`
- `digit`
- `graph`
- `lower`
- `print`
- `punct`
- `space`
- `upper`
- `xdigit`

4.681.3.6 `template<typename _Ch_type > template<typename _Fwd_iter > string_type std::regex_traits< _Ch_type >::lookup_collatename (_Fwd_iter __first, _Fwd_iter __last) const`

Gets a collation element by name.

Parameters

<code>__first</code>	beginning of the collation element name.
<code>__last</code>	one-past-the-end of the collation element name.

Returns

a sequence of one or more characters that represents the collating element consisting of the character sequence designated by the iterator range [`__first`, `__last`). Returns an empty string if the character sequence is not a valid collating element.

4.681.3.7 `template<typename _Ch_type > template<typename _Fwd_iter > string_type std::regex_traits< _Ch_type >::transform (_Fwd_iter __first, _Fwd_iter __last) const [inline]`

Gets a sort key for a character sequence.

Parameters

<code>__first</code>	beginning of the character sequence.
<code>__last</code>	one-past-the-end of the character sequence.

Returns a sort key for the character sequence designated by the iterator range [`F1`, `F2`) such that if the character sequence [`G1`, `G2`) sorts before the character sequence [`H1`, `H2`) then `v.transform(G1, G2) < v.transform(H1, H2)`.

What this really does is provide a more efficient way to compare a string to multiple other strings in locales with fancy collation rules and equivalence classes.

Returns

a locale-specific sort key equivalent to the input range.

Exceptions

<code>std::bad_cast</code>	if the current locale does not have a collate facet.
----------------------------	--

Definition at line 237 of file `regex.h`.

References `std::basic_string< _CharT, _Traits, _Alloc >::data()`, and `std::basic_string< _CharT, _Traits, _Alloc >::size()`.

4.681.3.8 `template<typename _Ch_type> template<typename _Fwd_iter> string_type std::regex_traits<_Ch_type>::transform_primary (_Fwd_iter __first, _Fwd_iter __last) const [inline]`

Gets a sort key for a character sequence, independent of case.

Parameters

<code>__first</code>	beginning of the character sequence.
<code>__last</code>	one-past-the-end of the character sequence.

Effects: if `typeid(use_facet<collate<_Ch_type>>) == typeid(collate_byname<_Ch_type>)` and the form of the sort key returned by `collate_byname<_Ch_type>::transform(__first, __last)` is known and can be converted into a primary sort key then returns that key, otherwise returns an empty string.

Todo Implement this function correctly.

Definition at line 261 of file `regex.h`.

References `std::vector<_Tp, _Alloc>::data()`, and `std::vector<_Tp, _Alloc>::size()`.

4.681.3.9 `template<typename _Ch_type> char_type std::regex_traits<_Ch_type>::translate (char_type __c) const [inline]`

Performs the identity translation.

Parameters

<code>__c</code>	A character to the locale-specific character set.
------------------	---

Returns

`__c`.

Definition at line 195 of file `regex.h`.

4.681.3.10 `template<typename _Ch_type> char_type std::regex_traits<_Ch_type>::translate_nocase (char_type __c) const [inline]`

Translates a character into a case-insensitive equivalent.

Parameters

<code>__c</code>	A character to the locale-specific character set.
------------------	---

Returns

the locale-specific lower-case equivalent of `__c`.

Exceptions

<code>std::bad_cast</code>	if the imbued locale does not support the ctype facet.
----------------------------	--

Definition at line 208 of file `regex.h`.

4.681.3.11 `template<typename _Ch_type> int std::regex_traits<_Ch_type>::value (_Ch_type __ch, int __radix) const`

Converts a digit to an int.

Parameters

<code>__ch</code>	a character representing a digit.
<code>__radix</code>	the radix if the numeric conversion (limited to 8, 10, or 16).

Returns

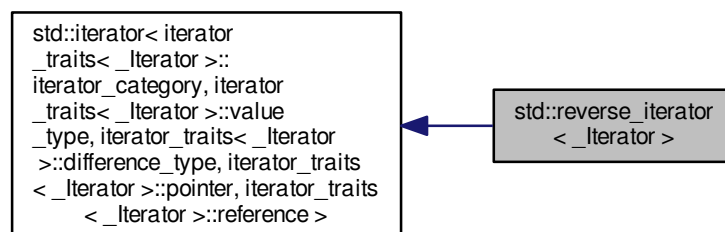
the value represented by the digit `__ch` in base `radix` if the character `__ch` is a valid digit in base `radix`; otherwise returns -1.

The documentation for this struct was generated from the following file:

- [regex.h](#)

4.682 `std::reverse_iterator<_Iterator>` Class Template Reference

Inheritance diagram for `std::reverse_iterator<_Iterator>`:



Public Types

- typedef __traits_type::difference_type **difference_type**
- typedef iterator_traits<_Iterator>::iterator_category iterator_category
- typedef _Iterator **iterator_type**
- typedef __traits_type::pointer **pointer**
- typedef __traits_type::reference **reference**
- typedef iterator_traits<_Iterator>::value_type value_type

Public Member Functions

- [reverse_iterator](#) ()
- [reverse_iterator](#) (iterator_type __x)
- [reverse_iterator](#) (const [reverse_iterator](#) &__x)
- template<typename _Iter >
 [reverse_iterator](#) (const [reverse_iterator](#)<_Iter> &__x)
- iterator_type [base](#) () const
- reference [operator*](#) () const
- [reverse_iterator](#) [operator+](#) (difference_type __n) const
- [reverse_iterator](#) & [operator++](#) ()
- [reverse_iterator](#) [operator++](#) (int)
- [reverse_iterator](#) & [operator+=](#) (difference_type __n)
- [reverse_iterator](#) [operator-](#) (difference_type __n) const
- [reverse_iterator](#) & [operator--](#) ()
- [reverse_iterator](#) [operator--](#) (int)
- [reverse_iterator](#) & [operator-=](#) (difference_type __n)
- pointer [operator->](#) () const
- reference [operator\[\]](#) (difference_type __n) const

Protected Types

- typedef iterator_traits<_Iterator> __traits_type

Protected Attributes

- _Iterator **current**

4.682.1 Detailed Description

```
template<typename _Iterator>
class std::reverse_iterator<_Iterator>
```

Bidirectional and random access iterators have corresponding reverse iterator adaptors that iterate through the data structure in the opposite direction. They have the same signatures as the corresponding iterators. The fundamental relation between a reverse iterator and its corresponding iterator *i* is established by the identity:

```
&*(reverse_iterator(i)) == &(i - 1)
```

This mapping is dictated by the fact that while there is always a pointer past the end of an array, there might not be a valid pointer before the beginning of an array. [24.4.1]/1,2

Reverse iterators can be tricky and surprising at first. Their semantics make sense, however, and the trickiness is a side effect of the requirement that the iterators must be safe.

Definition at line 97 of file stl_iterator.h.

4.682.2 Member Typedef Documentation

4.682.2.1 `typedef iterator_traits<_Iterator>::iterator_category std::iterator< iterator_traits<_Iterator>::iterator_category, iterator_traits<_Iterator>::value_type, iterator_traits<_Iterator>::difference_type, iterator_traits<_Iterator>::pointer, iterator_traits<_Iterator>::reference>::iterator_category` `[inherited]`

One of the [tag types](#).

Definition at line 121 of file `stl_iterator_base_types.h`.

4.682.2.2 `typedef iterator_traits<_Iterator>::value_type std::iterator< iterator_traits<_Iterator>::iterator_category, iterator_traits<_Iterator>::value_type, iterator_traits<_Iterator>::difference_type, iterator_traits<_Iterator>::pointer, iterator_traits<_Iterator>::reference>::value_type` `[inherited]`

The type "pointed to" by the iterator.

Definition at line 123 of file `stl_iterator_base_types.h`.

4.682.3 Constructor & Destructor Documentation

4.682.3.1 `template<typename _Iterator> std::reverse_iterator<_Iterator>::reverse_iterator ()` `[inline]`

The default constructor value-initializes member `current`. If it is a pointer, that means it is zero-initialized.

Definition at line 121 of file `stl_iterator.h`.

Referenced by `std::reverse_iterator<_Iterator>::operator+()`, and `std::reverse_iterator<_Iterator>::operator-()`.

4.682.3.2 `template<typename _Iterator> std::reverse_iterator<_Iterator>::reverse_iterator (iterator_type __x)` `[inline], [explicit]`

This iterator will move in the opposite direction that `x` does.

Definition at line 127 of file `stl_iterator.h`.

4.682.3.3 `template<typename _Iterator> std::reverse_iterator<_Iterator>::reverse_iterator (const reverse_iterator<_Iterator> &__x)` `[inline]`

The copy constructor is normal.

Definition at line 132 of file `stl_iterator.h`.

4.682.3.4 `template<typename _Iterator> template<typename _Iter> std::reverse_iterator<_Iterator>::reverse_iterator (const reverse_iterator<_Iter> &__x)` `[inline]`

A `reverse_iterator` across other types can be copied if the underlying iterator can be converted to the type of `current`.

Definition at line 140 of file `stl_iterator.h`.

4.682.4 Member Function Documentation

4.682.4.1 `template<typename _Iterator> iterator_type std::reverse_iterator<_Iterator>::base () const [inline]`

Returns

`current`, the iterator used for underlying work.

Definition at line 147 of file `stl_iterator.h`.

Referenced by `std::inserter()`, and `std::operator==()`.

4.682.4.2 `template<typename _Iterator> reference std::reverse_iterator<_Iterator>::operator* () const [inline]`

Returns

A reference to the value at `-current`

This requires that `-current` is dereferenceable.

Warning

This implementation requires that for an iterator of the underlying iterator type, `x`, a reference obtained by `*x` remains valid after `x` has been modified or destroyed. This is a bug: <http://gcc.gnu.org/PR51823>

Definition at line 161 of file `stl_iterator.h`.

Referenced by `std::inserter()`, and `std::reverse_iterator<_Iterator>::operator->()`.

4.682.4.3 `template<typename _Iterator> reverse_iterator std::reverse_iterator<_Iterator>::operator+ (difference_type __n) const [inline]`

Returns

A `reverse_iterator` that refers to `current - __n`

The underlying iterator must be a Random Access Iterator.

Definition at line 232 of file `stl_iterator.h`.

References `std::reverse_iterator<_Iterator>::reverse_iterator()`.

Referenced by `std::inserter()`, and `std::operator==()`.

4.682.4.4 `template<typename _Iterator> reverse_iterator& std::reverse_iterator<_Iterator>::operator++ () [inline]`

Returns

`*this`

Decrements the underlying iterator.

Definition at line 182 of file `stl_iterator.h`.

Referenced by `std::inserter()`.

4.682.4.5 `template<typename _Iterator> reverse_iterator std::reverse_iterator< _Iterator >::operator++ (int)`
`[inline]`

Returns

The original value of `*this`

Decrements the underlying iterator.

Definition at line 194 of file `stl_iterator.h`.

4.682.4.6 `template<typename _Iterator> reverse_iterator& std::reverse_iterator< _Iterator >::operator+= (difference_type __n)` `[inline]`

Returns

`*this`

Moves the underlying iterator backwards `__n` steps. The underlying iterator must be a Random Access Iterator.

Definition at line 242 of file `stl_iterator.h`.

Referenced by `std::inserter()`.

4.682.4.7 `template<typename _Iterator> reverse_iterator std::reverse_iterator< _Iterator >::operator- (difference_type __n) const` `[inline]`

Returns

A `reverse_iterator` that refers to `current - __n`

The underlying iterator must be a Random Access Iterator.

Definition at line 254 of file `stl_iterator.h`.

References `std::reverse_iterator< _Iterator >::reverse_iterator()`.

Referenced by `std::inserter()`, and `std::operator==()`.

4.682.4.8 `template<typename _Iterator> reverse_iterator& std::reverse_iterator< _Iterator >::operator-- ()`
`[inline]`

Returns

`*this`

Increments the underlying iterator.

Definition at line 207 of file `stl_iterator.h`.

Referenced by `std::inserter()`.

4.682.4.9 `template<typename _Iterator> reverse_iterator std::reverse_iterator<_Iterator>::operator-- (int)`
`[inline]`

Returns

A `reverse_iterator` with the previous value of `*this`

Increments the underlying iterator.

Definition at line 219 of file `stl_iterator.h`.

4.682.4.10 `template<typename _Iterator> reverse_iterator& std::reverse_iterator<_Iterator>::operator= (`
`difference_type __n) [inline]`

Returns

`*this`

Moves the underlying iterator forwards `__n` steps. The underlying iterator must be a Random Access Iterator.

Definition at line 264 of file `stl_iterator.h`.

Referenced by `std::inserter()`.

4.682.4.11 `template<typename _Iterator> pointer std::reverse_iterator<_Iterator>::operator-> () const [inline]`

Returns

A pointer to the value at `-current`

This requires that `-current` is dereferenceable.

Definition at line 173 of file `stl_iterator.h`.

References `std::reverse_iterator<_Iterator>::operator*()`.

Referenced by `std::inserter()`.

4.682.4.12 `template<typename _Iterator> reference std::reverse_iterator<_Iterator>::operator[] (difference_type __n)`
`const [inline]`

Returns

The value at `current - __n - 1`

The underlying iterator must be a Random Access Iterator.

Definition at line 276 of file `stl_iterator.h`.

Referenced by `std::inserter()`.

The documentation for this class was generated from the following file:

- [stl_iterator.h](#)

4.683 std::seed_seq Class Reference

Public Types

- typedef uint_least32_t [result_type](#)

Public Member Functions

- [seed_seq](#) ()
- template<typename _IntType >
seed_seq (std::initializer_list< _IntType > il)
- template<typename _InputIterator >
seed_seq (_InputIterator __begin, _InputIterator __end)
- template<typename _RandomAccessIterator >
void **generate** (_RandomAccessIterator __begin, _RandomAccessIterator __end)
- template<typename OutputIterator >
void **param** (OutputIterator __dest) const
- size_t **size** () const

4.683.1 Detailed Description

The seed_seq class generates sequences of seeds for random number generators.

Definition at line 6027 of file random.h.

4.683.2 Member Typedef Documentation

4.683.2.1 typedef uint_least32_t std::seed_seq::result_type

The type of the seed vales.

Definition at line 6032 of file random.h.

4.683.3 Constructor & Destructor Documentation

4.683.3.1 std::seed_seq::seed_seq () [inline]

Default constructor.

Definition at line 6035 of file random.h.

The documentation for this class was generated from the following file:

- [random.h](#)

4.684 std::set< _Key, _Compare, _Alloc > Class Template Reference

Public Types

- typedef _Key [key_type](#)
 - typedef _Key [value_type](#)
 - typedef _Compare [key_compare](#)
 - typedef _Compare [value_compare](#)
 - typedef _Alloc [allocator_type](#)
-
- typedef _Alloc_traits::pointer [pointer](#)
 - typedef _Alloc_traits::const_pointer [const_pointer](#)
 - typedef _Alloc_traits::reference [reference](#)
 - typedef _Alloc_traits::const_reference [const_reference](#)
 - typedef _Rep_type::const_iterator [iterator](#)
 - typedef _Rep_type::const_iterator [const_iterator](#)
 - typedef _Rep_type::const_reverse_iterator [reverse_iterator](#)
 - typedef _Rep_type::const_reverse_iterator [const_reverse_iterator](#)
 - typedef _Rep_type::size_type [size_type](#)
 - typedef _Rep_type::difference_type [difference_type](#)

Public Member Functions

- [set](#) ()
- [set](#) (const _Compare &__comp, const [allocator_type](#) &__a=[allocator_type](#)())
- template<typename _InputIterator >
[set](#) (_InputIterator __first, _InputIterator __last)
- template<typename _InputIterator >
[set](#) (_InputIterator __first, _InputIterator __last, const _Compare &__comp, const [allocator_type](#) &__a=[allocator_type](#)())
- [set](#) (const [set](#) &__x)
- [set](#) ([set](#) &&__x) noexcept(is_nothrow_copy_constructible< _Compare >::value)
- [set](#) (initializer_list< [value_type](#) > __l, const _Compare &__comp=_Compare(), const [allocator_type](#) &__a=[allocator_type](#)())
- [set](#) (const [allocator_type](#) &__a)
- [set](#) (const [set](#) &__x, const [allocator_type](#) &__a)
- [set](#) ([set](#) &&__x, const [allocator_type](#) &__a) noexcept(is_nothrow_copy_constructible< _Compare >::value && __a._Alloc_traits::_S_always_equal())
- [set](#) (initializer_list< [value_type](#) > __l, const [allocator_type](#) &__a)
- template<typename _InputIterator >
[set](#) (_InputIterator __first, _InputIterator __last, const [allocator_type](#) &__a)
- [iterator begin](#) () const noexcept
- [iterator cbegin](#) () const noexcept
- [iterator cend](#) () const noexcept
- void [clear](#) () noexcept
- [size_type count](#) (const [key_type](#) &__x) const
- [reverse_iterator crbegin](#) () const noexcept
- [reverse_iterator crend](#) () const noexcept

- `template<typename... _Args>`
`std::pair< iterator, bool > emplace (_Args &&...__args)`
- `template<typename... _Args>`
`iterator emplace_hint (const_iterator __pos, _Args &&...__args)`
- `bool empty ()` const noexcept
- `iterator end ()` const noexcept
- `_GLIBCXX_ABI_TAG_CXX11 iterator erase (const_iterator __position)`
- `size_type erase (const key_type &__x)`
- `_GLIBCXX_ABI_TAG_CXX11 iterator erase (const_iterator __first, const_iterator __last)`
- `allocator_type get_allocator ()` const noexcept
- `std::pair< iterator, bool > insert (const value_type &__x)`
- `std::pair< iterator, bool > insert (value_type &&__x)`
- `iterator insert (const_iterator __position, const value_type &__x)`
- `iterator insert (const_iterator __position, value_type &&__x)`
- `template<typename _InputIterator >`
`void insert (_InputIterator __first, _InputIterator __last)`
- `void insert (initializer_list< value_type > __l)`
- `key_compare key_comp ()` const
- `size_type max_size ()` const noexcept
- `set & operator= (const set &__x)`
- `set & operator= (set &&__x) noexcept(_Alloc_traits::_S_nothrow_move())`
- `set & operator= (initializer_list< value_type > __l)`
- `reverse_iterator rbegin ()` const noexcept
- `reverse_iterator rend ()` const noexcept
- `size_type size ()` const noexcept
- `void swap (set &__x) noexcept(_Alloc_traits::_S_nothrow_swap())`
- `value_compare value_comp ()` const
- `iterator find (const key_type &__x)`
- `const_iterator find (const key_type &__x) const`
- `iterator lower_bound (const key_type &__x)`
- `const_iterator lower_bound (const key_type &__x) const`
- `iterator upper_bound (const key_type &__x)`
- `const_iterator upper_bound (const key_type &__x) const`
- `std::pair< iterator, iterator > equal_range (const key_type &__x)`
- `std::pair< const_iterator, const_iterator > equal_range (const key_type &__x) const`

Friends

- `template<typename _K1, typename _C1, typename _A1 >`
`bool operator< (const set< _K1, _C1, _A1 > &, const set< _K1, _C1, _A1 > &)`
- `template<typename _K1, typename _C1, typename _A1 >`
`bool operator== (const set< _K1, _C1, _A1 > &, const set< _K1, _C1, _A1 > &)`

4.684.1 Detailed Description

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
class std::set< _Key, _Compare, _Alloc >
```

A standard container made up of unique keys, which can be retrieved in logarithmic time.

Template Parameters

<code>_Key</code>	Type of key objects.
<code>_Compare</code>	Comparison function object type, defaults to <code>less<_Key></code> .
<code>_Alloc</code>	Allocator type, defaults to <code>allocator<_Key></code> .

Meets the requirements of a [container](#), a [reversible container](#), and an [associative container](#) (using unique keys).

Sets support bidirectional iterators.

The private tree data is declared exactly the same way for set and multiset; the distinction is made entirely in how the tree functions are called (`*_unique` versus `*_equal`, same as the standard).

Definition at line 90 of file `stl_set.h`.

4.684.2 Member Typedef Documentation

4.684.2.1 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> typedef _Alloc std::set<_Key, _Compare, _Alloc>::allocator_type`

Public typedefs.

Definition at line 107 of file `stl_set.h`.

4.684.2.2 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> typedef _Rep_type::const_iterator std::set<_Key, _Compare, _Alloc>::const_iterator`

Iterator-related typedefs.

Definition at line 131 of file `stl_set.h`.

4.684.2.3 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> typedef _Alloc_traits::const_pointer std::set<_Key, _Compare, _Alloc>::const_pointer`

Iterator-related typedefs.

Definition at line 124 of file `stl_set.h`.

4.684.2.4 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> typedef _Alloc_traits::const_reference std::set<_Key, _Compare, _Alloc>::const_reference`

Iterator-related typedefs.

Definition at line 126 of file `stl_set.h`.

4.684.2.5 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> typedef
_Rep_type::const_reverse_iterator std::set<_Key, _Compare, _Alloc>::const_reverse_iterator`

Iterator-related typedefs.

Definition at line 133 of file `stl_set.h`.

4.684.2.6 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> typedef
_Rep_type::difference_type std::set<_Key, _Compare, _Alloc>::difference_type`

Iterator-related typedefs.

Definition at line 135 of file `stl_set.h`.

4.684.2.7 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> typedef
_Rep_type::const_iterator std::set<_Key, _Compare, _Alloc>::iterator`

Iterator-related typedefs.

Definition at line 130 of file `stl_set.h`.

4.684.2.8 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> typedef
_Compare std::set<_Key, _Compare, _Alloc>::key_compare`

Public typedefs.

Definition at line 105 of file `stl_set.h`.

4.684.2.9 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> typedef
_Key std::set<_Key, _Compare, _Alloc>::key_type`

Public typedefs.

Definition at line 103 of file `stl_set.h`.

4.684.2.10 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> typedef
_Alloc_traits::pointer std::set<_Key, _Compare, _Alloc>::pointer`

Iterator-related typedefs.

Definition at line 123 of file `stl_set.h`.

4.684.2.11 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> typedef
_Alloc_traits::reference std::set<_Key, _Compare, _Alloc>::reference`

Iterator-related typedefs.

Definition at line 125 of file `stl_set.h`.

4.684.2.12 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> typedef
_Rep_type::const_reverse_iterator std::set<_Key, _Compare, _Alloc>::reverse_iterator`

Iterator-related typedefs.

Definition at line 132 of file `stl_set.h`.

4.684.2.13 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> typedef
_Rep_type::size_type std::set<_Key, _Compare, _Alloc>::size_type`

Iterator-related typedefs.

Definition at line 134 of file `stl_set.h`.

4.684.2.14 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> typedef
_Compare std::set<_Key, _Compare, _Alloc>::value_compare`

Public typedefs.

Definition at line 106 of file `stl_set.h`.

4.684.2.15 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> typedef
_Key std::set<_Key, _Compare, _Alloc>::value_type`

Public typedefs.

Definition at line 104 of file `stl_set.h`.

4.684.3 Constructor & Destructor Documentation

4.684.3.1 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::set<_Key, _Compare, _Alloc>::set () [inline]`

Default constructor creates no elements.

Definition at line 142 of file `stl_set.h`.

4.684.3.2 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::set<_Key, _Compare, _Alloc>::set (const _Compare & __comp, const allocator_type & __a =
allocator_type()) [inline], [explicit]`

Creates a set with no elements.

Parameters

<code>__comp</code>	Comparator to use.
<code>__a</code>	An allocator object.

Definition at line 151 of file `stl_set.h`.

```
4.684.3.3  template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
            template<typename _InputIterator > std::set<_Key, _Compare, _Alloc>::set ( _InputIterator __first, _InputIterator
            __last ) [inline]
```

Builds a set from a range.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.

Create a set consisting of copies of the elements from `[__first,__last)`. This is linear in N if the range is already sorted, and $N\log N$ otherwise (where N is `distance(__first,__last)`).

Definition at line 166 of file `stl_set.h`.

```
4.684.3.4  template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
            template<typename _InputIterator > std::set<_Key, _Compare, _Alloc>::set ( _InputIterator __first, _InputIterator
            __last, const _Compare & __comp, const allocator_type & __a = allocator_type() ) [inline]
```

Builds a set from a range.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__comp</code>	A comparison functor.
<code>__a</code>	An allocator object.

Create a set consisting of copies of the elements from `[__first,__last)`. This is linear in N if the range is already sorted, and $N\log N$ otherwise (where N is `distance(__first,__last)`).

Definition at line 183 of file `stl_set.h`.

```
4.684.3.5  template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
            std::set<_Key, _Compare, _Alloc>::set ( const set<_Key, _Compare, _Alloc> & __x ) [inline]
```

Set copy constructor.

Parameters

<code>__x</code>	A set of identical element and allocator types.
------------------	---

The newly-created set uses a copy of the allocation object used by `__x`.

Definition at line 196 of file stl_set.h.

4.684.3.6 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::set<_Key, _Compare, _Alloc>::set (set<_Key, _Compare, _Alloc> && __x) [inline], [noexcept]`

Set move constructor

Parameters

<code>__x</code>	A set of identical element and allocator types.
------------------	---

The newly-created set contains the exact contents of x. The contents of x are a valid, but unspecified set.

Definition at line 207 of file stl_set.h.

4.684.3.7 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::set<_Key, _Compare, _Alloc>::set (initializer_list<value_type> __l, const _Compare & __comp =
_Compare(), const allocator_type & __a = allocator_type()) [inline]`

Builds a set from an initializer_list.

Parameters

<code>__l</code>	An initializer_list.
<code>__comp</code>	A comparison functor.
<code>__a</code>	An allocator object.

Create a set consisting of copies of the elements in the list. This is linear in N if the list is already sorted, and NlogN otherwise (where N is `__l.size()`).

Definition at line 221 of file stl_set.h.

4.684.3.8 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::set<_Key, _Compare, _Alloc>::set (const allocator_type & __a) [inline], [explicit]`

Allocator-extended default constructor.

Definition at line 229 of file stl_set.h.

4.684.3.9 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::set<_Key, _Compare, _Alloc>::set (const set<_Key, _Compare, _Alloc> & __x, const allocator_type & __a
) [inline]`

Allocator-extended copy constructor.

Definition at line 233 of file stl_set.h.

```
4.684.3.10 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::set<_Key, _Compare, _Alloc>::set ( set<_Key, _Compare, _Alloc> && __x, const allocator_type & __a )
[inline], [noexcept]
```

Allocator-extended move constructor.

Definition at line 237 of file stl_set.h.

```
4.684.3.11 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::set<_Key, _Compare, _Alloc>::set ( initializer_list<value_type> __l, const allocator_type & __a )
[inline]
```

Allocator-extended initializer-list constructor.

Definition at line 243 of file stl_set.h.

```
4.684.3.12 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
template<typename _InputIterator> std::set<_Key, _Compare, _Alloc>::set ( _InputIterator __first, _InputIterator
__last, const allocator_type & __a ) [inline]
```

Allocator-extended range constructor.

Definition at line 249 of file stl_set.h.

4.684.4 Member Function Documentation

```
4.684.4.1 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> iterator
std::set<_Key, _Compare, _Alloc>::begin ( ) const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the set. Iteration is done in ascending order according to the keys.

Definition at line 334 of file stl_set.h.

```
4.684.4.2 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> iterator
std::set<_Key, _Compare, _Alloc>::cbegin ( ) const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the set. Iteration is done in ascending order according to the keys.

Definition at line 371 of file stl_set.h.

```
4.684.4.3 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> iterator
std::set<_Key, _Compare, _Alloc>::cend ( ) const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the set. Iteration is done in ascending order according to the keys.

Definition at line 380 of file stl_set.h.

4.684.4.4 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> void
std::set<_Key, _Compare, _Alloc>::clear () [inline], [noexcept]`

Erases all elements in a set. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 669 of file `stl_set.h`.

Referenced by `std::set< _Key, _Compare, _Alloc >::operator=()`.

4.684.4.5 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
size_type std::set<_Key, _Compare, _Alloc>::count (const key_type & __x) const [inline]`

Finds the number of elements.

Parameters

<code>__x</code>	Element to located.
------------------	---------------------

Returns

Number of elements with specified key.

This function only makes sense for multisets; for set the result will either be 0 (not present) or 1 (present).

Definition at line 683 of file `stl_set.h`.

4.684.4.6 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
reverse_iterator std::set<_Key, _Compare, _Alloc>::crbegin () const [inline], [noexcept]`

Returns a read-only (constant) iterator that points to the last element in the set. Iteration is done in descending order according to the keys.

Definition at line 389 of file `stl_set.h`.

4.684.4.7 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
reverse_iterator std::set<_Key, _Compare, _Alloc>::crend () const [inline], [noexcept]`

Returns a read-only (constant) reverse iterator that points to the last pair in the set. Iteration is done in descending order according to the keys.

Definition at line 398 of file `stl_set.h`.

4.684.4.8 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
template<typename... _Args> std::pair<iterator, bool> std::set<_Key, _Compare, _Alloc>::emplace (_Args
&&... __args) [inline]`

Attempts to build and insert an element into the set.

Parameters

<code>__args</code>	Arguments used to generate an element.
---------------------	--

Returns

A pair, of which the first element is an iterator that points to the possibly inserted element, and the second is a bool that is true if the element was actually inserted.

This function attempts to build and insert an element into the set. A set relies on unique keys and thus an element is only inserted if it is not already present in the set.

Insertion requires logarithmic time.

Definition at line 452 of file `stl_set.h`.

```
4.684.4.9  template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
           template<typename... _Args> iterator std::set<_Key, _Compare, _Alloc>::emplace_hint ( const_iterator __pos,
           _Args &&... __args ) [inline]
```

Attempts to insert an element into the set.

Parameters

<code>__pos</code>	An iterator that serves as a hint as to where the element should be inserted.
<code>__args</code>	Arguments used to generate the element to be inserted.

Returns

An iterator that points to the element with key equivalent to the one generated from `__args` (may or may not be the element itself).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument `emplace()` does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.html>.

Insertion requires logarithmic time (if the hint is not taken).

Definition at line 478 of file `stl_set.h`.

```
4.684.4.10 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> bool
           std::set<_Key, _Compare, _Alloc>::empty ( ) const [inline], [noexcept]
```

Returns true if the set is empty.

Definition at line 404 of file `stl_set.h`.

4.684.4.11 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
 iterator std::set<_Key, _Compare, _Alloc>::end () const [inline], [noexcept]`

Returns a read-only (constant) iterator that points one past the last element in the set. Iteration is done in ascending order according to the keys.

Definition at line 343 of file stl_set.h.

4.684.4.12 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
 std::pair<iterator, iterator> std::set<_Key, _Compare, _Alloc>::equal_range (const key_type & __x)
 [inline]`

Finds a subsequence matching given key.

Parameters

<code>__x</code>	Key to be located.
------------------	--------------------

Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
              c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multisets.

Definition at line 763 of file stl_set.h.

4.684.4.13 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
 std::pair<const_iterator, const_iterator> std::set<_Key, _Compare, _Alloc>::equal_range (const
 key_type & __x) const [inline]`

Finds a subsequence matching given key.

Parameters

<code>__x</code>	Key to be located.
------------------	--------------------

Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
              c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multisets.

Definition at line 767 of file stl_set.h.

```
4.684.4.14  template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
             _GLIBCXX_ABI_TAG_CXX11 iterator std::set< _Key, _Compare, _Alloc >::erase ( const_iterator __position )
             [inline]
```

Erases an element from a set.

Parameters

<code>__position</code>	An iterator pointing to the element to be erased.
-------------------------	---

Returns

An iterator pointing to the element immediately following `__position` prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from a set. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 590 of file stl_set.h.

Referenced by `std::set< _Key, _Compare, _Alloc >::erase()`.

```
4.684.4.15  template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
             size_type std::set< _Key, _Compare, _Alloc >::erase ( const key_type & __x ) [inline]
```

Erases elements according to the provided key.

Parameters

<code>__x</code>	Key of element to be erased.
------------------	------------------------------

Returns

The number of elements erased.

This function erases all the elements located by the given key from a set. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 620 of file stl_set.h.

```
4.684.4.16  template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
            _GLIBCXX_ABI_TAG_CXX11 iterator std::set< _Key, _Compare, _Alloc >::erase ( const_iterator __first,
            const_iterator __last ) [inline]
```

Erases a [`__first`,`__last`) range of elements from a set.

Parameters

<code>__first</code>	Iterator pointing to the start of the range to be erased.
<code>__last</code>	Iterator pointing to the end of the range to be erased.

Returns

The iterator `__last`.

This function erases a sequence of elements from a set. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 642 of file stl_set.h.

References `std::set< _Key, _Compare, _Alloc >::erase()`.

```
4.684.4.17  template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
            iterator std::set< _Key, _Compare, _Alloc >::find ( const key_type & __x ) [inline]
```

Tries to locate an element in a set.

Parameters

<code>__x</code>	Element to be located.
------------------	------------------------

Returns

Iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end (`end()`) iterator.

Definition at line 701 of file `stl_set.h`.

```
4.684.4.18  template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
            const_iterator std::set<_Key, _Compare, _Alloc>::find ( const key_type & __x ) const  [inline]
```

Tries to locate an element in a set.

Parameters

<code>__x</code>	Element to be located.
------------------	------------------------

Returns

Iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end (`end()`) iterator.

Definition at line 705 of file `stl_set.h`.

```
4.684.4.19  template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
            allocator_type std::set<_Key, _Compare, _Alloc>::get_allocator ( ) const  [inline], [noexcept]
```

Returns the allocator object with which the set was constructed.

Definition at line 325 of file `stl_set.h`.

```
4.684.4.20  template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
            std::pair<iterator, bool> std::set<_Key, _Compare, _Alloc>::insert ( const value_type & __x )  [inline]
```

Attempts to insert an element into the set.

Parameters

<code>__x</code>	Element to be inserted.
------------------	-------------------------

Returns

A pair, of which the first element is an iterator that points to the possibly inserted element, and the second is a `bool` that is true if the element was actually inserted.

This function attempts to insert an element into the set. A set relies on unique keys and thus an element is only inserted if it is not already present in the set.

Insertion requires logarithmic time.

Definition at line 499 of file stl_set.h.

References std::pair< _T1, _T2 >::first, std::move(), and std::pair< _T1, _T2 >::second.

Referenced by std::set< _Key, _Compare, _Alloc >::insert(), and std::set< _Key, _Compare, _Alloc >::operator=().

```
4.684.4.21 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
            iterator std::set< _Key, _Compare, _Alloc >::insert ( const_iterator __position, const value_type & __x )
            [inline]
```

Attempts to insert an element into the set.

Parameters

<code>__position</code>	An iterator that serves as a hint as to where the element should be inserted.
<code>__x</code>	Element to be inserted.

Returns

An iterator that points to the element with key of `__x` (may or may not be the element passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument `insert()` does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.html>.↵

Insertion requires logarithmic time (if the hint is not taken).

Definition at line 536 of file stl_set.h.

References std::set< _Key, _Compare, _Alloc >::insert(), and std::move().

```
4.684.4.22 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
            template<typename _InputIterator > void std::set< _Key, _Compare, _Alloc >::insert ( _InputIterator __first,
            _InputIterator __last ) [inline]
```

A template function that attempts to insert a range of elements.

Parameters

<code>__first</code>	Iterator pointing to the start of the range to be inserted.
<code>__last</code>	Iterator pointing to the end of the range.

Complexity similar to that of the range constructor.

Definition at line 556 of file stl_set.h.

4.684.4.23 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> void
std::set<_Key, _Compare, _Alloc>::insert (initializer_list< value_type > __l) [inline]`

Attempts to insert a list of elements into the set.

Parameters

↔	A std::initializer_list<value_type> of elements to be inserted.
__↔	
↔	
__↔	
l	

Complexity similar to that of the range constructor.

Definition at line 568 of file stl_set.h.

References std::set<_Key, _Compare, _Alloc>::insert().

4.684.4.24 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
key_compare std::set<_Key, _Compare, _Alloc>::key_comp () const [inline]`

Returns the comparison object with which the set was constructed.

Definition at line 317 of file stl_set.h.

4.684.4.25 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
iterator std::set<_Key, _Compare, _Alloc>::lower_bound (const key_type & __x) [inline]`

Finds the beginning of a subsequence matching given key.

Parameters

__↔	Key to be located.
__x	

Returns

Iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.

Definition at line 722 of file stl_set.h.

4.684.4.26 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
const_iterator std::set<_Key, _Compare, _Alloc>::lower_bound (const key_type & __x) const [inline]`

Finds the beginning of a subsequence matching given key.

Parameters

<code>_↔</code>	Key to be located.
<code>_x</code>	

Returns

Iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.

Definition at line 726 of file stl_set.h.

```
4.684.4.27 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
    size_type std::set< _Key, _Compare, _Alloc >::max_size( ) const    [inline], [noexcept]
```

Returns the maximum size of the set.

Definition at line 414 of file stl_set.h.

```
4.684.4.28 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> set&
    std::set< _Key, _Compare, _Alloc >::operator=( const set< _Key, _Compare, _Alloc > &_x )    [inline]
```

Set assignment operator.

Parameters

<code>_↔</code>	A set of identical element and allocator types.
<code>_x</code>	

All the elements of `__x` are copied, but unlike the copy constructor, the allocator object is not copied.

Definition at line 263 of file stl_set.h.

```
4.684.4.29 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> set&
    std::set< _Key, _Compare, _Alloc >::operator=( set< _Key, _Compare, _Alloc > &&_x )    [inline],
    [noexcept]
```

Set move assignment operator.

Parameters

<code>_↔</code>	A set of identical element and allocator types.
<code>_x</code>	

The contents of `__x` are moved into this set (without copying if the allocators compare equal or get moved on assign-

ment). Afterwards `__x` is in a valid, but unspecified state.

Definition at line 279 of file `stl_set.h`.

References `std::set<_Key, _Compare, _Alloc>::clear()`, and `std::set<_Key, _Compare, _Alloc>::insert()`.

4.684.4.30 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> set& std::set<_Key, _Compare, _Alloc>::operator= (initializer_list<value_type> __l) [inline]`

Set list assignment operator.

Parameters

↩	An initializer_list.
↩	
↩	
↩	
↩	
↩	

This function fills a set with copies of the elements in the initializer list `__l`.

Note that the assignment completely changes the set and that the resulting set's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 305 of file `stl_set.h`.

References `std::set<_Key, _Compare, _Alloc>::clear()`, and `std::set<_Key, _Compare, _Alloc>::insert()`.

4.684.4.31 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> reverse_iterator std::set<_Key, _Compare, _Alloc>::rbegin () const [inline], [noexcept]`

Returns a read-only (constant) iterator that points to the last element in the set. Iteration is done in descending order according to the keys.

Definition at line 352 of file `stl_set.h`.

4.684.4.32 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> reverse_iterator std::set<_Key, _Compare, _Alloc>::rend () const [inline], [noexcept]`

Returns a read-only (constant) reverse iterator that points to the last pair in the set. Iteration is done in descending order according to the keys.

Definition at line 361 of file `stl_set.h`.

4.684.4.33 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> size_type std::set<_Key, _Compare, _Alloc>::size () const [inline], [noexcept]`

Returns the size of the set.

Definition at line 409 of file `stl_set.h`.

4.684.4.34 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> void std::set<_Key, _Compare, _Alloc>::swap (set<_Key, _Compare, _Alloc> & __x) [inline], [noexcept]`

Swaps data with another set.

Parameters

<code>_↔</code>	A set of the same element and allocator types.
<code>_X</code>	

This exchanges the elements between two sets in constant time. (It is only swapping a pointer, an integer, and an instance of the `Compare` type (which itself is often stateless and empty), so it should be quite fast.) Note that the global `std::swap()` function is specialized such that `std::swap(s1,s2)` will feed to this function.

Definition at line 429 of file `stl_set.h`.

Referenced by `std::swap()`.

4.684.4.35 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
iterator std::set<_Key, _Compare, _Alloc >::upper_bound (const key_type & __x) [inline]`

Finds the end of a subsequence matching given key.

Parameters

<code>_↔</code>	Key to be located.
<code>_X</code>	

Returns

Iterator pointing to the first element greater than key, or `end()`.

Definition at line 738 of file `stl_set.h`.

4.684.4.36 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
const_iterator std::set<_Key, _Compare, _Alloc >::upper_bound (const key_type & __x) const [inline]`

Finds the end of a subsequence matching given key.

Parameters

<code>_↔</code>	Key to be located.
<code>_X</code>	

Returns

Iterator pointing to the first element greater than key, or `end()`.

Definition at line 742 of file `stl_set.h`.

4.684.4.37 `template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
value_compare std::set<_Key, _Compare, _Alloc>::value_comp () const [inline]`

Returns the comparison object with which the set was constructed.

Definition at line 321 of file `stl_set.h`.

The documentation for this class was generated from the following file:

- [stl_set.h](#)

4.685 `std::shared_ptr<_Tp>` Class Template Reference

Inherits `std::__shared_ptr<_Tp, _Lp>`.

Public Types

- `typedef _Tp element_type`

Public Member Functions

- constexpr `shared_ptr()` noexcept
- `shared_ptr` (const `shared_ptr` &) noexcept=default
- `template<typename _Tp1 >`
`shared_ptr` (`_Tp1 *`__p)
- `template<typename _Tp1, typename _Deleter >`
`shared_ptr` (`_Tp1 *`__p, `_Deleter` __d)
- `template<typename _Deleter >`
`shared_ptr` (nullptr_t __p, `_Deleter` __d)
- `template<typename _Tp1, typename _Deleter, typename _Alloc >`
`shared_ptr` (`_Tp1 *`__p, `_Deleter` __d, `_Alloc` __a)
- `template<typename _Deleter, typename _Alloc >`
`shared_ptr` (nullptr_t __p, `_Deleter` __d, `_Alloc` __a)
- `template<typename _Tp1 >`
`shared_ptr` (const `shared_ptr`<_Tp1> &__r, `_Tp *`__p) noexcept
- `template<typename _Tp1, typename = typename std::enable_if<std::is_convertible<_Tp1*, _Tp*>::value>::type>`
`shared_ptr` (const `shared_ptr`<_Tp1> &__r) noexcept
- `shared_ptr` (`shared_ptr` &&__r) noexcept
- `template<typename _Tp1, typename = typename std::enable_if<std::is_convertible<_Tp1*, _Tp*>::value>::type>`
`shared_ptr` (`shared_ptr`<_Tp1> &&__r) noexcept
- `template<typename _Tp1 >`
`shared_ptr` (const `weak_ptr`<_Tp1> &__r)
- `template<typename _Tp1, typename _Del >`
`shared_ptr` (`std::unique_ptr`<_Tp1, `_Del`> &&__r)
- constexpr `shared_ptr` (nullptr_t __p) noexcept
- `template<typename _Tp1 >`
`shared_ptr` (`std::auto_ptr`<_Tp1> &&__r)
- `_Tp *` `get` () const noexcept

- **operator bool** () const
- std::add_lvalue_reference< _Tp >::type **operator*** () const noexcept
- _Tp * **operator->** () const noexcept
- **shared_ptr** & **operator=** (const **shared_ptr** &) noexcept=default
- template<typename _Tp1 >
shared_ptr & **operator=** (const **shared_ptr**< _Tp1 > &__r) noexcept
- **shared_ptr** & **operator=** (**shared_ptr** &&__r) noexcept
- template<class _Tp1 >
shared_ptr & **operator=** (**shared_ptr**< _Tp1 > &&__r) noexcept
- template<typename _Tp1 , typename _Del >
shared_ptr & **operator=** (std::unique_ptr< _Tp1, _Del > &&__r)
- template<typename _Tp1 >
bool **owner_before** (__shared_ptr< _Tp1, _Lp > const &__rhs) const
- template<typename _Tp1 >
bool **owner_before** (__weak_ptr< _Tp1, _Lp > const &__rhs) const
- void **reset** () noexcept
- template<typename _Tp1 >
void **reset** (_Tp1 *__p)
- template<typename _Tp1 , typename _Deleter >
void **reset** (_Tp1 *__p, _Deleter __d)
- template<typename _Tp1 , typename _Deleter , typename _Alloc >
void **reset** (_Tp1 *__p, _Deleter __d, _Alloc __a)
- void **swap** (__shared_ptr< _Tp, _Lp > &__other) noexcept
- bool **unique** () const noexcept
- long **use_count** () const noexcept

Friends

- template<typename _Tp1 , typename _Alloc , typename... _Args>
shared_ptr< _Tp1 > **allocate_shared** (const _Alloc &__a, _Args &&...__args)
- class **weak_ptr**< _Tp >

4.685.1 Detailed Description

```
template<typename _Tp>
class std::shared_ptr< _Tp >
```

A smart pointer with reference-counted copy semantics.

The object pointed to is deleted when the last **shared_ptr** pointing to it is destroyed or reset.

Definition at line 93 of file **shared_ptr.h**.

4.685.2 Constructor & Destructor Documentation

4.685.2.1 `template<typename _Tp> constexpr std::shared_ptr<_Tp>::shared_ptr () [inline], [noexcept]`

Construct an empty `shared_ptr`.

Postcondition

`use_count()==0 && get()==0`

Definition at line 100 of file `shared_ptr.h`.

Referenced by `std::auto_ptr<_Tp>::auto_ptr()`, and `std::shared_ptr<__detail::_NFA<_Rx_traits>>::shared_ptr()`.

4.685.2.2 `template<typename _Tp> template<typename _Tp1> std::shared_ptr<_Tp>::shared_ptr (_Tp1 * __p) [inline], [explicit]`

Construct a `shared_ptr` that owns the pointer `__p`.

Parameters

<code>__p</code>	A pointer that is convertible to <code>element_type*</code> .
------------------	---

Postcondition

`use_count() == 1 && get() == __p`

Exceptions

<code>std::bad_alloc</code> , in	which case <code>delete __p</code> is called.
----------------------------------	---

Definition at line 112 of file `shared_ptr.h`.

4.685.2.3 `template<typename _Tp> template<typename _Tp1, typename _Deleter> std::shared_ptr<_Tp>::shared_ptr (_Tp1 * __p, _Deleter __d) [inline]`

Construct a `shared_ptr` that owns the pointer `__p` and the deleter `__d`.

Parameters

<code>__p</code>	A pointer.
<code>__d</code>	A deleter.

Postcondition

use_count() == 1 && get() == __p

Exceptions

<i>std::bad_alloc</i> , in	which case __d(__p) is called.
----------------------------	--------------------------------

Requirements: _Deleter's copy constructor and destructor must not throw

__shared_ptr will release __p by calling __d(__p)

Definition at line 129 of file shared_ptr.h.

4.685.2.4 `template<typename _Tp> template<typename _Deleter > std::shared_ptr< _Tp >::shared_ptr (nullptr_t __p, _Deleter __d) [inline]`

Construct a shared_ptr that owns a null pointer and the deleter __d.

Parameters

<code>__p</code>	A null pointer constant.
<code>__d</code>	A deleter.

Postcondition

use_count() == 1 && get() == __p

Exceptions

<i>std::bad_alloc</i> , in	which case __d(__p) is called.
----------------------------	--------------------------------

Requirements: _Deleter's copy constructor and destructor must not throw

The last owner will call __d(__p)

Definition at line 146 of file shared_ptr.h.

4.685.2.5 `template<typename _Tp> template<typename _Tp1, typename _Deleter, typename _Alloc > std::shared_ptr< _Tp >::shared_ptr (_Tp1 * __p, _Deleter __d, _Alloc __a) [inline]`

Construct a shared_ptr that owns the pointer __p and the deleter __d.

Parameters

$_p$	A pointer.
$_d$	A deleter.
$_a$	An allocator.

Postcondition

`use_count() == 1 && get() == $_p$`

Exceptions

<code><i>std::bad_alloc</i></code> , in	which case <code>$_d(_p)$</code> is called.
---	--

Requirements: `_Deleter`'s copy constructor and destructor must not throw `_Alloc`'s copy constructor and destructor must not throw.

`__shared_ptr` will release `__p` by calling `__d(__p)`

Definition at line 165 of file `shared_ptr.h`.

4.685.2.6 `template<typename _Tp> template<typename _Deleter, typename _Alloc> std::shared_ptr<_Tp>::shared_ptr (nullptr_t __p, _Deleter __d, _Alloc __a) [inline]`

Construct a `shared_ptr` that owns a null pointer and the deleter `__d`.

Parameters

$_p$	A null pointer constant.
$_d$	A deleter.
$_a$	An allocator.

Postcondition

`use_count() == 1 && get() == $_p$`

Exceptions

<code><i>std::bad_alloc</i></code> , in	which case <code>$_d(_p)$</code> is called.
---	--

Requirements: _Deleter's copy constructor and destructor must not throw _Alloc's copy constructor and destructor must not throw.

The last owner will call __d(__p)

Definition at line 184 of file shared_ptr.h.

4.685.2.7 `template<typename _Tp> template<typename _Tp1 > std::shared_ptr< _Tp >::shared_ptr (const shared_ptr< _Tp1 > & __r, _Tp* __p) [inline], [noexcept]`

Constructs a shared_ptr instance that stores __p and shares ownership with __r.

Parameters

<code>__r</code>	A shared_ptr.
<code>__p</code>	A pointer that will remain valid while *__r is valid.

Postcondition

`get() == __p && use_count() == __r.use_count()`

This can be used to construct a shared_ptr to a sub-object of an object managed by an existing shared_ptr.

```
shared_ptr< pair<int,int> > pii(new pair<int,int>());
shared_ptr<int> pi(pii, &pii->first);
assert(pii.use_count() == 2);
```

Definition at line 206 of file shared_ptr.h.

4.685.2.8 `template<typename _Tp> template<typename _Tp1, typename = typename std::enable_if<std::is_convertible<_Tp1*, _Tp*>::value>::type> std::shared_ptr< _Tp >::shared_ptr (const shared_ptr< _Tp1 > & __r) [inline], [noexcept]`

If __r is empty, constructs an empty shared_ptr; otherwise construct a shared_ptr that shares ownership with __r.

Parameters

<code>__r</code>	A shared_ptr.
------------------	---------------

Postcondition

`get() == __r.get() && use_count() == __r.use_count()`

Definition at line 218 of file shared_ptr.h.

4.685.2.9 `template<typename _Tp> std::shared_ptr<_Tp>::shared_ptr(shared_ptr<_Tp> && __r) [inline],
[noexcept]`

Move-constructs a shared_ptr instance from __r.

Parameters

↩	A shared_ptr rvalue.
__↩	
↩	
__↩	
<i>r</i>	

Postcondition

*this contains the old value of __r, __r is empty.

Definition at line 226 of file shared_ptr.h.

4.685.2.10 `template<typename _Tp> template<typename _Tp1, typename = typename std::enable_if<std::is_convertible<_Tp1*,
_Tp*>::value>::type> std::shared_ptr<_Tp>::shared_ptr(shared_ptr<_Tp1> && __r) [inline],
[noexcept]`

Move-constructs a shared_ptr instance from __r.

Parameters

↩	A shared_ptr rvalue.
__↩	
↩	
__↩	
<i>r</i>	

Postcondition

*this contains the old value of __r, __r is empty.

Definition at line 236 of file shared_ptr.h.

4.685.2.11 `template<typename _Tp> template<typename _Tp1> std::shared_ptr<_Tp>::shared_ptr(const weak_ptr<
_Tp1> & __r) [inline], [explicit]`

Constructs a shared_ptr that shares ownership with __r and stores a copy of the pointer stored in __r.

Parameters

\leftrightarrow	A weak_ptr.
$_ \leftrightarrow$	
\leftarrow	
$_ \leftarrow$	
<i>r</i>	

Postcondition

use_count() == $_r$.use_count()

Exceptions

<i>bad_weak_ptr</i>	when $_r$.expired(), in which case the constructor has no effect.
---------------------	---

Definition at line 248 of file shared_ptr.h.

4.685.2.12 `template<typename _Tp> constexpr std::shared_ptr<_Tp>::shared_ptr(nullptr_t __p) [inline],
[noexcept]`

Construct an empty shared_ptr.

Parameters

$_ \leftrightarrow$	A null pointer constant.
$_p$	

Postcondition

use_count() == 0 && get() == nullptr

Definition at line 265 of file shared_ptr.h.

4.685.3 Friends And Related Function Documentation

4.685.3.1 `template<typename _Tp> template<typename _Tp1, typename _Alloc, typename... _Args> shared_ptr<_Tp1>
allocate_shared(const _Alloc & __a, _Args &&... __args) [friend]`

Create an object that is owned by a shared_ptr.

Parameters

$_a$	An allocator.
$_args$	Arguments for the $_Tp$ object's constructor.

Returns

A `shared_ptr` that owns the newly created object.

Exceptions

<i>An</i>	exception thrown from <code>_Alloc::allocate</code> or from the constructor of <code>_Tp</code> .
-----------	---

A copy of `__a` will be used to allocate memory for the `shared_ptr` and the new object.

Definition at line 591 of file `shared_ptr.h`.

Referenced by `std::shared_ptr< __detail::_NFA< _Rx_traits > >::shared_ptr()`.

The documentation for this class was generated from the following files:

- [shared_ptr.h](#)
- [auto_ptr.h](#)

4.686 std::shuffle_order_engine< _RandomNumberEngine, __k > Class Template Reference**Public Types**

- typedef `_RandomNumberEngine::result_type` [result_type](#)

Public Member Functions

- [shuffle_order_engine](#) ()
- [shuffle_order_engine](#) (const `_RandomNumberEngine` & __rng)
- [shuffle_order_engine](#) (`_RandomNumberEngine` && __rng)
- [shuffle_order_engine](#) ([result_type](#) __s)
- template<typename `_Sseq` , typename = typename `std::enable_if<!std::is_same<_Sseq, shuffle_order_engine>::value && !std::is_↔ same<_Sseq, _RandomNumberEngine>::value> ::type>`
[shuffle_order_engine](#) (`_Sseq` & __q)
- const `_RandomNumberEngine` & [base](#) () const noexcept
- void [discard](#) (unsigned long long __z)
- [result_type](#) [operator\(\)](#) ()
- void [seed](#) ()
- void [seed](#) ([result_type](#) __s)
- template<typename `_Sseq` >
void [seed](#) (`_Sseq` & __q)

Static Public Member Functions

- static constexpr [result_type](#) [max](#) ()
- static constexpr [result_type](#) [min](#) ()

Static Public Attributes

- static constexpr size_t **table_size**

Friends

- template<typename _RandomNumberEngine1, size_t __k1, typename _CharT, typename _Traits >
std::basic_ostream< _CharT, _Traits > & **operator<<** (std::basic_ostream< _CharT, _Traits > &__os, const
std::shuffle_order_engine< _RandomNumberEngine1, __k1 > &__x)
- bool **operator==** (const shuffle_order_engine &__lhs, const shuffle_order_engine &__rhs)
- template<typename _RandomNumberEngine1, size_t __k1, typename _CharT, typename _Traits >
std::basic_istream< _CharT, _Traits > & **operator>>** (std::basic_istream< _CharT, _Traits > &__is, std::←
::shuffle_order_engine< _RandomNumberEngine1, __k1 > &__x)

4.686.1 Detailed Description

```
template<typename _RandomNumberEngine, size_t __k>
class std::shuffle_order_engine< _RandomNumberEngine, __k >
```

Produces random numbers by combining random numbers from some base engine to produce random numbers with a specifies number of bits `__w`.

Definition at line 1294 of file random.h.

4.686.2 Member Typedef Documentation

4.686.2.1 `template<typename _RandomNumberEngine, size_t __k> typedef _RandomNumberEngine::result_type
std::shuffle_order_engine< _RandomNumberEngine, __k >::result_type`

The type of the generated random value.

Definition at line 1297 of file random.h.

4.686.3 Constructor & Destructor Documentation

4.686.3.1 `template<typename _RandomNumberEngine, size_t __k> std::shuffle_order_engine< _RandomNumberEngine,
__k >::shuffle_order_engine () [inline]`

Constructs a default shuffle_order_engine engine.

The underlying engine is default constructed as well.

Definition at line 1310 of file random.h.

4.686.3.2 `template<typename _RandomNumberEngine, size_t __k> std::shuffle_order_engine< _RandomNumberEngine,
__k >::shuffle_order_engine (const _RandomNumberEngine & __rng) [inline],[explicit]`

Copy constructs a shuffle_order_engine engine.

Copies an existing base class random number generator.

Parameters

<code>__rng</code>	An existing (base class) engine object.
--------------------	---

Definition at line 1321 of file random.h.

```
4.686.3.3  template<typename _RandomNumberEngine, size_t __k> std::shuffle_order_engine< _RandomNumberEngine,
__k >::shuffle_order_engine ( _RandomNumberEngine && __rng ) [inline],[explicit]
```

Move constructs a shuffle_order_engine engine.

Copies an existing base class random number generator.

Parameters

<code>__rng</code>	An existing (base class) engine object.
--------------------	---

Definition at line 1332 of file random.h.

```
4.686.3.4  template<typename _RandomNumberEngine, size_t __k> std::shuffle_order_engine< _RandomNumberEngine,
__k >::shuffle_order_engine ( result_type __s ) [inline],[explicit]
```

Seed constructs a shuffle_order_engine engine.

Constructs the underlying generator engine seeded with __s.

Parameters

<code>__s</code>	A seed value for the base class engine.
------------------	---

Definition at line 1343 of file random.h.

```
4.686.3.5  template<typename _RandomNumberEngine, size_t __k> template<typename _Sseq , typename =
typename std::enable_if<!std::is_same< _Sseq, shuffle_order_engine>::value && !std::is_same< _Sseq,
_RandomNumberEngine>::value> ::type> std::shuffle_order_engine< _RandomNumberEngine, __k
>::shuffle_order_engine ( _Sseq & __q ) [inline],[explicit]
```

Generator construct a shuffle_order_engine engine.

Parameters

<code>__q</code>	A seed sequence.
------------------	------------------

Definition at line 1357 of file random.h.

4.686.4 Member Function Documentation

4.686.4.1 `template<typename _RandomNumberEngine, size_t __k> const _RandomNumberEngine& std::shuffle_order_engine<_RandomNumberEngine, __k>::base() const [inline], [noexcept]`

Gets a const reference to the underlying generator engine object.

Definition at line 1400 of file random.h.

4.686.4.2 `template<typename _RandomNumberEngine, size_t __k> void std::shuffle_order_engine<_RandomNumberEngine, __k>::discard(unsigned long long __z) [inline]`

Discard a sequence of random numbers.

Definition at line 1421 of file random.h.

4.686.4.3 `template<typename _RandomNumberEngine, size_t __k> static constexpr result_type std::shuffle_order_engine<_RandomNumberEngine, __k>::max() [inline], [static]`

Gets the maximum value in the generated random number range.

Definition at line 1414 of file random.h.

References std::max().

4.686.4.4 `template<typename _RandomNumberEngine, size_t __k> static constexpr result_type std::shuffle_order_engine<_RandomNumberEngine, __k>::min() [inline], [static]`

Gets the minimum value in the generated random number range.

Definition at line 1407 of file random.h.

References std::min().

4.686.4.5 `template<typename _RandomNumberEngine, size_t __k> result_type std::shuffle_order_engine<_RandomNumberEngine, __k>::operator()()`

Gets the next value in the generated random number sequence.

4.686.4.6 `template<typename _RandomNumberEngine, size_t __k> void std::shuffle_order_engine<_RandomNumberEngine, __k>::seed() [inline]`

Reseeds the shuffle_order_engine object with the default seed for the underlying base class generator engine.

Definition at line 1366 of file random.h.

4.686.4.7 `template<typename _RandomNumberEngine, size_t __k> void std::shuffle_order_engine<_RandomNumberEngine, __k>::seed(result_type __s) [inline]`

Reseeds the shuffle_order_engine object with the default seed for the underlying base class generator engine.

Definition at line 1377 of file random.h.

4.686.4.8 `template<typename _RandomNumberEngine, size_t __k> template<typename _Sseq> void std::shuffle_order_engine<_RandomNumberEngine, __k>::seed(_Sseq & __q) [inline]`

Reseeds the shuffle_order_engine object with the given seed sequence.

Parameters

<code>__q</code>	A seed generator function.
------------------	----------------------------

Definition at line 1390 of file random.h.

4.686.5 Friends And Related Function Documentation

4.686.5.1 `template<typename _RandomNumberEngine, size_t __k> template<typename _RandomNumberEngine1, size_t __k1, typename _CharT, typename _Traits > std::basic_ostream<_CharT, _Traits>& operator<< (std::basic_ostream<_CharT, _Traits> & __os, const std::shuffle_order_engine<_RandomNumberEngine1, __k1> & __x)`
`[friend]`

Inserts the current state of a shuffle_order_engine random number generator engine `__x` into the output stream `__os`.

Parameters

<code>__os</code>	An output stream.
<code>__x</code>	A shuffle_order_engine random number generator engine.

Returns

The output stream with the state of `__x` inserted or in an error state.

4.686.5.2 `template<typename _RandomNumberEngine, size_t __k> bool operator==(const shuffle_order_engine<_RandomNumberEngine, __k> & __lhs, const shuffle_order_engine<_RandomNumberEngine, __k> & __rhs)`
`[friend]`

Compares two shuffle_order_engine random number generator objects of the same type for equality.

Parameters

<code>__lhs</code>	A shuffle_order_engine random number generator object.
<code>__rhs</code>	Another shuffle_order_engine random number generator object.

Returns

true if the infinite sequences of generated values would be equal, false otherwise.

Definition at line 1445 of file random.h.


```
4.686.5.3 template<typename _RandomNumberEngine, size_t __k> template<typename _RandomNumberEngine1, size_t __k1,
typename _CharT, typename _Traits > std::basic_istream<_CharT, _Traits>& operator>> ( std::basic_istream<
_CharT, _Traits > & __is, std::shuffle_order_engine<_RandomNumberEngine1, __k1> & __x ) [friend]
```

Extracts the current state of a % subtract_with_carry_engine random number generator engine __x from the input stream __is.

Parameters

<code>__is</code>	An input stream.
<code>__x</code>	A shuffle_order_engine random number generator engine.

Returns

The input stream with the state of __x extracted or in an error state.

The documentation for this class was generated from the following file:

- [random.h](#)

4.687 std::slice Class Reference

Public Member Functions

- [slice](#) ()
- [slice](#) (size_t __o, size_t __d, size_t __s)
- size_t [size](#) () const
- size_t [start](#) () const
- size_t [stride](#) () const

4.687.1 Detailed Description

Class defining one-dimensional subset of an array.

The slice class represents a one-dimensional subset of an array, specified by three parameters: start offset, size, and stride. The start offset is the index of the first element of the array that is part of the subset. The size is the total number of elements in the subset. Stride is the distance between each successive array element to include in the subset.

For example, with an array of size 10, and a slice with offset 1, size 3 and stride 2, the subset consists of array elements 1, 3, and 5.

Definition at line 59 of file slice_array.h.

The documentation for this class was generated from the following file:

- [slice_array.h](#)

4.688 `std::slice_array<_Tp>` Class Template Reference

Public Types

- typedef `_Tp` **value_type**

Public Member Functions

- `slice_array` (const `slice_array` &)
- void `operator%=>` (const `valarray`< `_Tp` > &) const
- template<class `_Dom` >
void `operator%=>` (const `_Expr`< `_Dom`, `_Tp` > &) const
- void `operator&=>` (const `valarray`< `_Tp` > &) const
- template<class `_Dom` >
void `operator&=>` (const `_Expr`< `_Dom`, `_Tp` > &) const
- void `operator*=>` (const `valarray`< `_Tp` > &) const
- template<class `_Dom` >
void `operator*=>` (const `_Expr`< `_Dom`, `_Tp` > &) const
- void `operator+=>` (const `valarray`< `_Tp` > &) const
- template<class `_Dom` >
void `operator+=>` (const `_Expr`< `_Dom`, `_Tp` > &) const
- void `operator-=>` (const `valarray`< `_Tp` > &) const
- template<class `_Dom` >
void `operator-=>` (const `_Expr`< `_Dom`, `_Tp` > &) const
- void `operator/=` (const `valarray`< `_Tp` > &) const
- template<class `_Dom` >
void `operator/=` (const `_Expr`< `_Dom`, `_Tp` > &) const
- void `operator<=>` (const `valarray`< `_Tp` > &) const
- template<class `_Dom` >
void `operator<=>` (const `_Expr`< `_Dom`, `_Tp` > &) const
- `slice_array` & `operator=` (const `slice_array` &)
- void `operator=` (const `valarray`< `_Tp` > &) const
- void `operator=` (const `_Tp` &) const
- template<class `_Dom` >
void `operator=` (const `_Expr`< `_Dom`, `_Tp` > &) const
- void `operator>=>` (const `valarray`< `_Tp` > &) const
- template<class `_Dom` >
void `operator>=>` (const `_Expr`< `_Dom`, `_Tp` > &) const
- void `operator^=>` (const `valarray`< `_Tp` > &) const
- template<class `_Dom` >
void `operator^=>` (const `_Expr`< `_Dom`, `_Tp` > &) const
- void `operator|=` (const `valarray`< `_Tp` > &) const
- template<class `_Dom` >
void `operator|=` (const `_Expr`< `_Dom`, `_Tp` > &) const

Friends

- class `valarray`< `_Tp` >

4.688.1 Detailed Description

```
template<typename _Tp>
class std::slice_array<_Tp>
```

Reference to one-dimensional subset of an array.

A `slice_array` is a reference to the actual elements of an array specified by a slice. The way to get a `slice_array` is to call `operator[]`(slice) on a `valarray`. The returned `slice_array` then permits carrying operations out on the referenced subset of elements in the original `valarray`. For example, `operator+=(valarray)` will add values to the subset of elements in the underlying `valarray` this `slice_array` refers to.

Parameters

<i>Tp</i>	Element type.
-----------	---------------

Definition at line 123 of file `slice_array.h`.

The documentation for this class was generated from the following file:

- [slice_array.h](#)

4.689 `std::stack<_Tp, _Sequence>` Class Template Reference

Public Types

- typedef `_Sequence::const_reference` **const_reference**
- typedef `_Sequence` **container_type**
- typedef `_Sequence::reference` **reference**
- typedef `_Sequence::size_type` **size_type**
- typedef `_Sequence::value_type` **value_type**

Public Member Functions

- [stack](#) (const `_Sequence` &__c)
- **stack** (`_Sequence` &&__c=`_Sequence`())
- template<typename... `_Args`>
void **emplace** (`_Args` &&... __args)
- bool [empty](#) () const
- void [pop](#) ()
- void [push](#) (const `value_type` &__x)
- void **push** (`value_type` &&__x)
- `size_type` [size](#) () const
- void **swap** ([stack](#) &__s) noexcept(noexcept(`swap`(c, __s.c)))
- reference [top](#) ()
- `const_reference` [top](#) () const

Protected Attributes

- `_Sequence c`

Friends

- `template<typename _Tp1, typename _Seq1 >`
`bool operator< (const stack< _Tp1, _Seq1 > &, const stack< _Tp1, _Seq1 > &)`
- `template<typename _Tp1, typename _Seq1 >`
`bool operator== (const stack< _Tp1, _Seq1 > &, const stack< _Tp1, _Seq1 > &)`

4.689.1 Detailed Description

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
class std::stack< _Tp, _Sequence >
```

A standard container giving FILO behavior.

Template Parameters

<code>_Tp</code>	Type of element.
<code>_Sequence</code>	Type of underlying sequence, defaults to <code>deque<_Tp></code> .

Meets many of the requirements of a [container](#), but does not define anything to do with iterators. Very few of the other standard container interfaces are defined.

This is not a true container, but an *adaptor*. It holds another container, and provides a wrapper interface to that container. The wrapper is what enforces strict first-in-last-out stack behavior.

The second template parameter defines the type of the underlying sequence/container. It defaults to `std::deque`, but it can be any type that supports `back`, `push_back`, and `pop_front`, such as `std::list`, `std::vector`, or an appropriate user-defined type.

Members not found in *normal* containers are `container_type`, which is a typedef for the second `Sequence` parameter, and `push`, `pop`, and `top`, which are standard stack/FILO operations.

Definition at line 99 of file `stl_stack.h`.

4.689.2 Constructor & Destructor Documentation

4.689.2.1 `template<typename _Tp, typename _Sequence = deque<_Tp>> std::stack< _Tp, _Sequence >::stack (const _Sequence &_c) [inline], [explicit]`

Default constructor creates no elements.

Definition at line 137 of file `stl_stack.h`.

Referenced by `std::stack< _StateSeqT >::stack()`.

4.689.3 Member Function Documentation

4.689.3.1 `template<typename _Tp, typename _Sequence = deque<_Tp>> bool std::stack< _Tp, _Sequence >::empty ()`
`const [inline]`

Returns true if the stack is empty.

Definition at line 149 of file `stl_stack.h`.

4.689.3.2 `template<typename _Tp, typename _Sequence = deque<_Tp>> void std::stack< _Tp, _Sequence >::pop ()`
`[inline]`

Removes first element.

This is a typical stack operation. It shrinks the stack by one. The time complexity of the operation depends on the underlying sequence.

Note that no data is returned, and if the first element's data is needed, it should be retrieved before `pop()` is called.

Definition at line 215 of file `stl_stack.h`.

4.689.3.3 `template<typename _Tp, typename _Sequence = deque<_Tp>> void std::stack< _Tp, _Sequence >::push (const`
`value_type &__x) [inline]`

Add data to the top of the stack.

Parameters

<code>__x</code>	Data to be added.
------------------	-------------------

This is a typical stack operation. The function creates an element at the top of the stack and assigns the given data to it. The time complexity of the operation depends on the underlying sequence.

Definition at line 189 of file `stl_stack.h`.

Referenced by `std::stack< _StateSeqT >::push()`.

4.689.3.4 `template<typename _Tp, typename _Sequence = deque<_Tp>> size_type std::stack< _Tp, _Sequence >::size ()`
`const [inline]`

Returns the number of elements in the stack.

Definition at line 154 of file `stl_stack.h`.

4.689.3.5 `template<typename _Tp, typename _Sequence = deque<_Tp>> reference std::stack< _Tp, _Sequence >::top ()`
`[inline]`

Returns a read/write reference to the data at the first element of the stack.

Definition at line 162 of file `stl_stack.h`.

4.689.3.6 `template<typename _Tp, typename _Sequence = deque<_Tp>> const_reference std::stack<_Tp, _Sequence>::top
() const [inline]`

Returns a read-only (constant) reference to the data at the first element of the stack.

Definition at line 173 of file `stl_stack.h`.

The documentation for this class was generated from the following file:

- [stl_stack.h](#)

4.690 `std::student_t_distribution<_RealType>` Class Template Reference

Classes

- struct [param_type](#)

Public Types

- typedef `_RealType` [result_type](#)

Public Member Functions

- **`student_t_distribution`** (`_RealType __n=_RealType(1)`)
- **`student_t_distribution`** (const [param_type](#) &__p)
- `template<typename _ForwardIterator, typename _UniformRandomNumberGenerator >`
`void __generate` (`_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng`)
- `template<typename _ForwardIterator, typename _UniformRandomNumberGenerator >`
`void __generate` (`_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const`
`param_type &__p`)
- `template<typename _UniformRandomNumberGenerator >`
`void __generate` (`result_type *__f, result_type *__t, _UniformRandomNumberGenerator &__urng`)
- `template<typename _UniformRandomNumberGenerator >`
`void __generate` (`result_type *__f, result_type *__t, _UniformRandomNumberGenerator &__urng, const`
`param_type &__p`)
- `result_type max` () const
- `result_type min` () const
- `_RealType n` () const
- `template<typename _UniformRandomNumberGenerator >`
`result_type operator()` (`_UniformRandomNumberGenerator &__urng`)
- `template<typename _UniformRandomNumberGenerator >`
`result_type operator()` (`_UniformRandomNumberGenerator &__urng, const param_type &__p`)
- `param_type param` () const
- `void param` (const [param_type](#) &__param)
- `void reset` ()

Friends

- template<typename _RealType1, typename _CharT, typename _Traits >
std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::student_t_distribution< _RealType1 > &__x)
- bool operator== (const student_t_distribution &__d1, const student_t_distribution &__d2)
- template<typename _RealType1, typename _CharT, typename _Traits >
std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::student_t_distribution< _RealType1 > &__x)

4.690.1 Detailed Description

```
template<typename _RealType = double>
class std::student_t_distribution< _RealType >
```

A student_t_distribution random number distribution.

The formula for the normal probability mass function is:

$$p(x|n) = \frac{1}{\sqrt{(n\pi)}} \frac{\Gamma((n+1)/2)}{\Gamma(n/2)} \left(1 + \frac{x^2}{n}\right)^{-(n+1)/2}$$

Definition at line 3356 of file random.h.

4.690.2 Member Typedef Documentation

4.690.2.1 template<typename _RealType = double> typedef _RealType std::student_t_distribution< _RealType >::result_type

The type of the range of the distribution.

Definition at line 3359 of file random.h.

4.690.3 Member Function Documentation

4.690.3.1 template<typename _RealType = double> result_type std::student_t_distribution< _RealType >::max () const [inline]

Returns the least upper bound value of the distribution.

Definition at line 3439 of file random.h.

References std::max().

4.690.3.2 `template<typename _RealType = double> result_type std::student_t_distribution<_RealType>::min () const`
`[inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 3432 of file random.h.

4.690.3.3 `template<typename _RealType = double> template<typename _UniformRandomNumberGenerator> result_type`
`std::student_t_distribution<_RealType>::operator() (_UniformRandomNumberGenerator & __urng)`
`[inline]`

Generating functions.

Definition at line 3447 of file random.h.

4.690.3.4 `template<typename _RealType = double> param_type std::student_t_distribution<_RealType>::param ()`
`const [inline]`

Returns the parameter set of the distribution.

Definition at line 3417 of file random.h.

4.690.3.5 `template<typename _RealType = double> void std::student_t_distribution<_RealType>::param (const`
`param_type & __param) [inline]`

Sets the parameter set of the distribution.

Parameters

<code>__param</code>	The new parameter set of the distribution.
----------------------	--

Definition at line 3425 of file random.h.

4.690.3.6 `template<typename _RealType = double> void std::student_t_distribution<_RealType>::reset ()`
`[inline]`

Resets the distribution state.

Definition at line 3400 of file random.h.

4.690.4 Friends And Related Function Documentation

4.690.4.1 `template<typename _RealType = double> template<typename _RealType1 , typename _CharT , typename _Traits`
`> std::basic_ostream<_CharT, _Traits>& operator<< (std::basic_ostream<_CharT, _Traits> & __os, const`
`std::student_t_distribution<_RealType1> & __x) [friend]`

Inserts a `student_t_distribution` random number distribution `__x` into the output stream `__os`.

Parameters

<code>__os</code>	An output stream.
<code>__x</code>	A student_t_distribution random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

4.690.4.2 `template<typename _RealType = double> bool operator==(const student_t_distribution<_RealType> &__d1, const student_t_distribution<_RealType> &__d2) [friend]`

Return true if two Student t distributions have the same parameters and the sequences that would be generated are equal.

Definition at line 3496 of file random.h.

4.690.4.3 `template<typename _RealType = double> template<typename _RealType1, typename _CharT, typename _Traits> std::basic_istream<_CharT, _Traits>& operator>> (std::basic_istream<_CharT, _Traits> &__is, std::student_t_distribution<_RealType1> &__x) [friend]`

Extracts a student_t_distribution random number distribution `__x` from the input stream `__is`.

Parameters

<code>__is</code>	An input stream.
<code>__x</code>	A student_t_distribution random number generator engine.

Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following file:

- [random.h](#)

4.691 std::student_t_distribution<_RealType>::param_type Struct Reference

Public Types

- typedef [student_t_distribution](#)<_RealType> **distribution_type**

Public Member Functions

- **param_type** (_RealType __n=_RealType(1))
- _RealType **n** () const

Friends

- bool **operator==** (const [param_type](#) &__p1, const [param_type](#) &__p2)

4.691.1 Detailed Description

```
template<typename _RealType = double>
struct std::student_t_distribution< _RealType >::param_type
```

Parameter type.

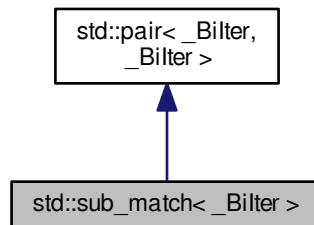
Definition at line 3365 of file random.h.

The documentation for this struct was generated from the following file:

- [random.h](#)

4.692 std::sub_match<_Bilter > Class Template Reference

Inheritance diagram for std::sub_match<_Bilter >:



Public Types

- typedef __iter_traits::difference_type **difference_type**
- typedef _Bilter **first_type**
- typedef _Bilter **iterator**
- typedef _Bilter [second_type](#)
- typedef [std::basic_string](#)< value_type > **string_type**
- typedef __iter_traits::value_type **value_type**

Public Member Functions

- int [compare](#) (const [sub_match](#) &__s) const
- int [compare](#) (const [string_type](#) &__s) const
- int [compare](#) (const value_type *__s) const
- difference_type [length](#) () const
- [operator string_type](#) () const
- [string_type](#) [str](#) () const
- void [swap](#) ([pair](#) &__p) noexcept(noexcept(swap([first](#), __p.first)) &&noexcept(swap([second](#), __p.second)))

Public Attributes

- [_Bilter](#) [first](#)
- bool [matched](#)
- [_Bilter](#) [second](#)

4.692.1 Detailed Description

```
template<typename _Bilter>
class std::sub_match<_Bilter >
```

A sequence of characters matched by a particular marked sub-expression.

An object of this class is essentially a pair of iterators marking a matched subexpression within a regular expression pattern match. Such objects can be converted to and compared with std::basic_string objects of a similar base character type as the pattern matched by the regular expression.

The iterators that make up the pair are the usual half-open interval referencing the actual original pattern matched.

Definition at line 857 of file regex.h.

4.692.2 Member Typedef Documentation

4.692.2.1 typedef [_Bilter](#) [std::pair](#)<[_Bilter](#), [_Bilter](#)>::[second_type](#) [inherited]

[first_type](#) is the first bound type

Definition at line 99 of file stl_pair.h.

4.692.3 Member Function Documentation

4.692.3.1 template<typename [_Bilter](#)> int [std::sub_match](#)<[_Bilter](#)>::[compare](#) (const [sub_match](#)<[_Bilter](#)> &__s)
const [inline]

Compares this and another matched sequence.

Parameters

<code>__s</code>	Another matched sequence to compare to this one.
------------------	--

Return values

<code>< 0</code>	this matched sequence will collate before <code>__s</code> .
<code>= 0</code>	this matched sequence is equivalent to <code>__s</code> .
<code>> 0</code>	this matched sequence will collate after <code>__s</code> .

Definition at line 918 of file regex.h.

Referenced by `std::operator!=()`, `std::operator<()`, `std::operator<=()`, `std::operator==()`, `std::operator>()`, and `std::operator>=()`.

4.692.3.2 `template<typename _Bilter> int std::sub_match<_Bilter>::compare (const string_type & __s) const`
`[inline]`

Compares this `sub_match` to a string.

Parameters

<code>__s</code>	A string to compare to this <code>sub_match</code> .
------------------	--

Return values

<code>< 0</code>	this matched sequence will collate before <code>__s</code> .
<code>= 0</code>	this matched sequence is equivalent to <code>__s</code> .
<code>> 0</code>	this matched sequence will collate after <code>__s</code> .

Definition at line 931 of file regex.h.

4.692.3.3 `template<typename _Bilter> int std::sub_match<_Bilter>::compare (const value_type * __s) const`
`[inline]`

Compares this `sub_match` to a C-style string.

Parameters

<code>__s</code>	A C-style string to compare to this <code>sub_match</code> .
------------------	--

Return values

<code><0</code>	this matched sequence will collate before <code>__s</code> .
<code>=0</code>	this matched sequence is equivalent to <code>__s</code> .
<code>>0</code>	this matched sequence will collate after <code>__s</code> .

Definition at line 944 of file regex.h.

4.692.3.4 `template<typename _Bilter> difference_type std::sub_match<_Bilter>::length () const [inline]`

Gets the length of the matching sequence.

Definition at line 875 of file regex.h.

4.692.3.5 `template<typename _Bilter> std::sub_match<_Bilter>::operator string_type () const [inline]`

Gets the matching sequence as a string.

Returns

the matching sequence as a string.

This is the implicit conversion operator. It is identical to the `str()` member function except that it will want to pop up in unexpected places and cause a great deal of confusion and cursing from the unwary.

Definition at line 888 of file regex.h.

4.692.3.6 `template<typename _Bilter> string_type std::sub_match<_Bilter>::str () const [inline]`

Gets the matching sequence as a string.

Returns

the matching sequence as a string.

Definition at line 901 of file regex.h.

Referenced by `std::sub_match<_Bi_iter>::compare()`, and `std::operator<<()`.

4.692.4 Member Data Documentation

4.692.4.1 `_Bilter std::pair<_Bilter, _Bilter>::first [inherited]`

`second_type` is the second bound type

Definition at line 101 of file stl_pair.h.

4.692.4.2 `_Bilter std::pair<_Bilter, _Bilter>::second` [inherited]

`first` is a copy of the first object

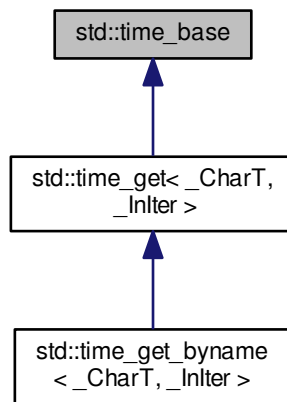
Definition at line 102 of file `stl_pair.h`.

The documentation for this class was generated from the following file:

- [regex.h](#)

4.693 `std::time_base` Class Reference

Inheritance diagram for `std::time_base`:



Public Types

- enum **dateorder** {
no_order, **dmy**, **mdy**, **ydm**,
ydm }

4.693.1 Detailed Description

Time format ordering data.

This class provides an enum representing different orderings of time: day, month, and year.

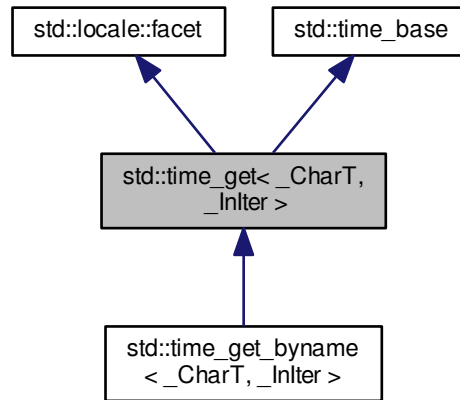
Definition at line 52 of file `locale_facets_nonio.h`.

The documentation for this class was generated from the following file:

- [locale_facets_nonio.h](#)

4.694 std::time_get<_CharT, _InIter> Class Template Reference

Inheritance diagram for std::time_get<_CharT, _InIter>:



Public Types

- typedef `basic_string<_CharT>` `__string_type`
- enum `dateorder` {
 `no_order`, `dmy`, `mdy`, `ymd`,
 `ydm` }
- typedef `_CharT` `char_type`
- typedef `_InIter` `iter_type`

Public Member Functions

- `time_get` (`size_t` __refs=0)
- `dateorder` `date_order` () const
- `iter_type` `get_date` (`iter_type` __beg, `iter_type` __end, `ios_base` & __io, `ios_base::iostate` & __err, `tm` * __tm) const
- `iter_type` `get_monthname` (`iter_type` __beg, `iter_type` __end, `ios_base` & __io, `ios_base::iostate` & __err, `tm` * __tm) const
- `iter_type` `get_time` (`iter_type` __beg, `iter_type` __end, `ios_base` & __io, `ios_base::iostate` & __err, `tm` * __tm) const
- `iter_type` `get_weekday` (`iter_type` __beg, `iter_type` __end, `ios_base` & __io, `ios_base::iostate` & __err, `tm` * __tm) const
- `iter_type` `get_year` (`iter_type` __beg, `iter_type` __end, `ios_base` & __io, `ios_base::iostate` & __err, `tm` * __tm) const

Static Public Attributes

- static `locale::id` `id`

Protected Member Functions

- virtual `~time_get()`
- `iter_type M_extract_name(iter_type __beg, iter_type __end, int &__member, const _CharT **__names, size_t __indexlen, ios_base &__io, ios_base::iostate &__err) const`
- `iter_type M_extract_num(iter_type __beg, iter_type __end, int &__member, int __min, int __max, size_t __len, ios_base &__io, ios_base::iostate &__err) const`
- `iter_type M_extract_via_format(iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm, const _CharT *__format) const`
- `iter_type M_extract_wday_or_month(iter_type __beg, iter_type __end, int &__member, const _CharT **__names, size_t __indexlen, ios_base &__io, ios_base::iostate &__err) const`
- virtual dateorder `do_date_order()` const
- virtual `iter_type do_get_date(iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm) const`
- virtual `iter_type do_get_monthname(iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm) const`
- virtual `iter_type do_get_time(iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm) const`
- virtual `iter_type do_get_weekday(iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm) const`
- virtual `iter_type do_get_year(iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm) const`

Static Protected Member Functions

- static `_c_locale _S_clone_c_locale(_c_locale &__cloc) throw()`
- static void `_S_create_c_locale(_c_locale &__cloc, const char *__s, _c_locale __old=0)`
- static void `_S_destroy_c_locale(_c_locale &__cloc)`
- static `_c_locale _S_get_c_locale()`
- static const char * `_S_get_c_name()` throw()
- static `_c_locale _S_lc_type_c_locale(_c_locale __cloc, const char *__s)`

4.694.1 Detailed Description

```
template<typename _CharT, typename _InIter>
class std::time_get<_CharT, _InIter>
```

Primary class template `time_get`.

This facet encapsulates the code to parse and return a date or time from a string. It is used by the istream numeric extraction operators.

The `time_get` template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the `time_get` facet.

Definition at line 368 of file `locale_facets_nonio.h`.

4.694.2 Member Typedef Documentation

4.694.2.1 `template<typename _CharT, typename _InIter> typedef _CharT std::time_get<_CharT, _InIter>::char_type`

Public typedefs.

Definition at line 374 of file locale_facets_nonio.h.

4.694.2.2 `template<typename _CharT, typename _InIter> typedef _InIter std::time_get<_CharT, _InIter>::iter_type`

Public typedefs.

Definition at line 375 of file locale_facets_nonio.h.

4.694.3 Constructor & Destructor Documentation

4.694.3.1 `template<typename _CharT, typename _InIter> std::time_get<_CharT, _InIter>::time_get (size_t __refs = 0)
[inline], [explicit]`

Constructor performs initialization.

This is the constructor provided by the standard.

Parameters

<code>__refs</code>	Passed to the base facet class.
---------------------	---------------------------------

Definition at line 390 of file locale_facets_nonio.h.

4.694.3.2 `template<typename _CharT, typename _InIter> virtual std::time_get<_CharT, _InIter>::~time_get ()
[inline], [protected], [virtual]`

Destructor.

Definition at line 546 of file locale_facets_nonio.h.

4.694.4 Member Function Documentation

4.694.4.1 `template<typename _CharT, typename _InIter> dateorder std::time_get<_CharT, _InIter>::date_order () const
[inline]`

Return preferred order of month, day, and year.

This function returns an enum from `timebase::dateorder` giving the preferred ordering if the format `x` given to `time_put<_CharT, _InIter>::put()` only uses month, day, and year. If the format `x` for the associated locale uses other fields, this function returns `timebase::dateorder::noorder`.

NOTE: The library always returns `noorder` at the moment.

Returns

A member of `timebase::dateorder`.

Definition at line 407 of file `locale_facets_nonio.h`.

```
4.694.4.2  template<typename _CharT, typename _InIter > virtual dateorder std::time_get<_CharT, _InIter >::do_date_order (
            ) const    [protected], [virtual]
```

Return preferred order of month, day, and year.

This function returns an enum from `timebase::dateorder` giving the preferred ordering if the format `x` given to `time_put<_CharT, _InIter >::put()` only uses month, day, and year. This function is a hook for derived classes to change the value returned.

Returns

A member of `timebase::dateorder`.

```
4.694.4.3  template<typename _CharT, typename _InIter > virtual iter_type std::time_get<_CharT, _InIter >::do_get_date
            ( iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, tm * __tm ) const
            [protected], [virtual]
```

Parse input date string.

This function parses a date according to the format `X` and puts the results into a user-supplied struct `tm`. This function is a hook for derived classes to change the value returned.

See also

`get_date()` for details.

Parameters

<code>__beg</code>	Start of string to parse.
<code>__end</code>	End of string to parse.
<code>__io</code>	Source of the locale.
<code>__err</code>	Error flags to set.
<code>__tm</code>	Pointer to struct <code>tm</code> to fill in.

Returns

Iterator to first char beyond date string.

```
4.694.4.4  template<typename _CharT, typename _InIter > virtual iter_type std::time_get<_CharT, _InIter
            >::do_get_monthname ( iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, tm * __tm )
            const    [protected], [virtual]
```

Parse input month string.

This function parses a month name and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

See also

get_monthname() for details.

Parameters

<code>__beg</code>	Start of string to parse.
<code>__end</code>	End of string to parse.
<code>__io</code>	Source of the locale.
<code>__err</code>	Error flags to set.
<code>__tm</code>	Pointer to struct tm to fill in.

Returns

Iterator to first char beyond month name.

```
4.694.4.5 template<typename _CharT, typename _InIter > virtual iter_type std::time_get< _CharT, _InIter >::do_get_time
( iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, tm * __tm ) const
[protected], [virtual]
```

Parse input time string.

This function parses a time according to the format x and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

See also

get_time() for details.

Parameters

<code>__beg</code>	Start of string to parse.
<code>__end</code>	End of string to parse.
<code>__io</code>	Source of the locale.
<code>__err</code>	Error flags to set.
<code>__tm</code>	Pointer to struct tm to fill in.

Returns

Iterator to first char beyond time string.

```
4.694.4.6  template<typename _CharT, typename _InIter > virtual iter_type std::time_get< _CharT, _InIter >::do_get_weekday
           ( iter_type __beg, iter_type __end, ios_base & , ios_base::iostate & __err, tm * __tm ) const
           [protected], [virtual]
```

Parse input weekday string.

This function parses a weekday name and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

See also

get_weekday() for details.

Parameters

<code>__beg</code>	Start of string to parse.
<code>__end</code>	End of string to parse.
<code>__io</code>	Source of the locale.
<code>__err</code>	Error flags to set.
<code>__tm</code>	Pointer to struct tm to fill in.

Returns

Iterator to first char beyond weekday name.

```
4.694.4.7  template<typename _CharT, typename _InIter > virtual iter_type std::time_get< _CharT, _InIter >::do_get_year
           ( iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, tm * __tm ) const
           [protected], [virtual]
```

Parse input year string.

This function reads up to 4 characters to parse a year string and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

See also

get_year() for details.

Parameters

<code>__beg</code>	Start of string to parse.
<code>__end</code>	End of string to parse.
<code>__io</code>	Source of the locale.
<code>__err</code>	Error flags to set.
<code>__tm</code>	Pointer to struct tm to fill in.

Returns

Iterator to first char beyond year.

4.694.4.8 `template<typename _CharT, typename _InIter> iter_type std::time_get<_CharT, _InIter>::get_date (iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, tm * __tm) const [inline]`

Parse input date string.

This function parses a date according to the format *x* and puts the results into a user-supplied struct *tm*. The result is returned by calling `time_get::do_get_date()`.

If there is a valid date string according to format *x*, *tm* will be filled in accordingly and the returned iterator will point to the first character beyond the date string. If an error occurs before the end, `err |= ios_base::failbit`. If parsing reads all the characters, `err |= ios_base::eofbit`.

Parameters

<code>__beg</code>	Start of string to parse.
<code>__end</code>	End of string to parse.
<code>__io</code>	Source of the locale.
<code>__err</code>	Error flags to set.
<code>__tm</code>	Pointer to struct <i>tm</i> to fill in.

Returns

Iterator to first char beyond date string.

Definition at line 456 of file `locale_facets_nonio.h`.

4.694.4.9 `template<typename _CharT, typename _InIter> iter_type std::time_get<_CharT, _InIter>::get_monthname (iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, tm * __tm) const [inline]`

Parse input month string.

This function parses a month name and puts the results into a user-supplied struct *tm*. The result is returned by calling `time_get::do_get_monthname()`.

Parsing starts by parsing an abbreviated month name. If a valid abbreviation is followed by a character that would lead to the full month name, parsing continues until the full name is found or an error occurs. Otherwise parsing finishes at the end of the abbreviated name.

If an error occurs before the end, `err |= ios_base::failbit`. If parsing reads all the characters, `err |= ios_base::eofbit`.

Parameters

<code>__beg</code>	Start of string to parse.
<code>__end</code>	End of string to parse.
<code>__io</code>	Source of the locale.
<code>__err</code>	Error flags to set.
<code>tm</code>	Pointer to struct <i>tm</i> to fill in.

Returns

Iterator to first char beyond month name.

Definition at line 513 of file locale_facets_nonio.h.

4.694.4.10 `template<typename _CharT, typename _InIter > iter_type std::time_get< _CharT, _InIter >::get_time (iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, tm * __tm) const [inline]`

Parse input time string.

This function parses a time according to the format *X* and puts the results into a user-supplied struct tm. The result is returned by calling `time_get::do_get_time()`.

If there is a valid time string according to format *X*, *tm* will be filled in accordingly and the returned iterator will point to the first character beyond the time string. If an error occurs before the end, `err |= ios_base::failbit`. If parsing reads all the characters, `err |= ios_base::eofbit`.

Parameters

<code>__beg</code>	Start of string to parse.
<code>__end</code>	End of string to parse.
<code>__io</code>	Source of the locale.
<code>__err</code>	Error flags to set.
<code>__tm</code>	Pointer to struct tm to fill in.

Returns

Iterator to first char beyond time string.

Definition at line 431 of file locale_facets_nonio.h.

4.694.4.11 `template<typename _CharT, typename _InIter > iter_type std::time_get< _CharT, _InIter >::get_weekday (iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, tm * __tm) const [inline]`

Parse input weekday string.

This function parses a weekday name and puts the results into a user-supplied struct tm. The result is returned by calling `time_get::do_get_weekday()`.

Parsing starts by parsing an abbreviated weekday name. If a valid abbreviation is followed by a character that would lead to the full weekday name, parsing continues until the full name is found or an error occurs. Otherwise parsing finishes at the end of the abbreviated name.

If an error occurs before the end, `err |= ios_base::failbit`. If parsing reads all the characters, `err |= ios_base::eofbit`.

Parameters

<code>__beg</code>	Start of string to parse.
<code>__end</code>	End of string to parse.
<code>__io</code>	Source of the locale.
<code>__err</code>	Error flags to set.
<code>__tm</code>	Pointer to struct tm to fill in.

Returns

Iterator to first char beyond weekday name.

Definition at line 484 of file locale_facets_nonio.h.

4.694.4.12 `template<typename _CharT, typename _InIter> iter_type std::time_get<_CharT, _InIter>::get_year (iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, tm * __tm) const [inline]`

Parse input year string.

This function reads up to 4 characters to parse a year string and puts the results into a user-supplied struct tm. The result is returned by calling time_get::do_get_year().

4 consecutive digits are interpreted as a full year. If there are exactly 2 consecutive digits, the library interprets this as the number of years since 1900.

If an error occurs before the end, err |= ios_base::failbit. If parsing reads all the characters, err |= ios_base::eofbit.

Parameters

<code>__beg</code>	Start of string to parse.
<code>__end</code>	End of string to parse.
<code>__io</code>	Source of the locale.
<code>__err</code>	Error flags to set.
<code>__tm</code>	Pointer to struct tm to fill in.

Returns

Iterator to first char beyond year.

Definition at line 539 of file locale_facets_nonio.h.

4.694.5 Member Data Documentation

4.694.5.1 `template<typename _CharT, typename _InIter> locale::id std::time_get<_CharT, _InIter>::id [static]`

Numpunct facet id.

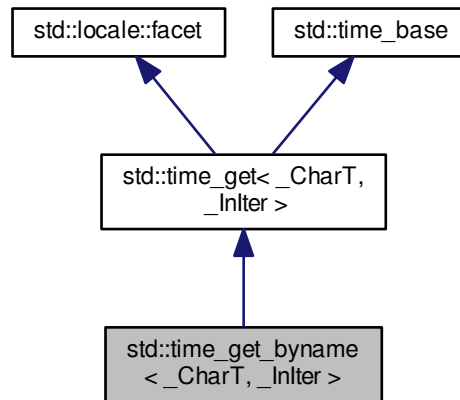
Definition at line 380 of file locale_facets_nonio.h.

The documentation for this class was generated from the following file:

- [locale_facets_nonio.h](#)

4.695 `std::time_get_byname<_CharT, _Inlter >` Class Template Reference

Inheritance diagram for `std::time_get_byname<_CharT, _Inlter >`:



Public Types

- typedef `basic_string<_CharT>` `__string_type`
- typedef `_CharT` `char_type`
- enum `dateorder` {
 `no_order`, `dmy`, `mdy`, `ymd`,
 `ydm` }
- typedef `_Inlter` `iter_type`

Public Member Functions

- `time_get_byname` (`const char *`, `size_t __refs=0`)
- `dateorder date_order` () `const`
- `iter_type get_date` (`iter_type __beg`, `iter_type __end`, `ios_base &__io`, `ios_base::iostate &__err`, `tm *__tm`) `const`
- `iter_type get_monthname` (`iter_type __beg`, `iter_type __end`, `ios_base &__io`, `ios_base::iostate &__err`, `tm *__tm`) `const`
- `iter_type get_time` (`iter_type __beg`, `iter_type __end`, `ios_base &__io`, `ios_base::iostate &__err`, `tm *__tm`) `const`
- `iter_type get_weekday` (`iter_type __beg`, `iter_type __end`, `ios_base &__io`, `ios_base::iostate &__err`, `tm *__tm`) `const`
- `iter_type get_year` (`iter_type __beg`, `iter_type __end`, `ios_base &__io`, `ios_base::iostate &__err`, `tm *__tm`) `const`

Static Public Attributes

- static `locale::id` `id`

Protected Member Functions

- `iter_type _M_extract_name` (`iter_type` __beg, `iter_type` __end, `int` &__member, `const _CharT**` __names, `size_t` __indexlen, `ios_base` &__io, `ios_base::iostate` &__err) `const`
- `iter_type _M_extract_num` (`iter_type` __beg, `iter_type` __end, `int` &__member, `int` __min, `int` __max, `size_t` __len, `ios_base` &__io, `ios_base::iostate` &__err) `const`
- `iter_type _M_extract_via_format` (`iter_type` __beg, `iter_type` __end, `ios_base` &__io, `ios_base::iostate` &__err, `tm` *__tm, `const _CharT*` __format) `const`
- `iter_type _M_extract_wday_or_month` (`iter_type` __beg, `iter_type` __end, `int` &__member, `const _CharT**` __names, `size_t` __indexlen, `ios_base` &__io, `ios_base::iostate` &__err) `const`
- virtual `dateorder` `do_date_order` () `const`
- virtual `iter_type` `do_get_date` (`iter_type` __beg, `iter_type` __end, `ios_base` &__io, `ios_base::iostate` &__err, `tm` *__tm) `const`
- virtual `iter_type` `do_get_monthname` (`iter_type` __beg, `iter_type` __end, `ios_base` &, `ios_base::iostate` &__err, `tm` *__tm) `const`
- virtual `iter_type` `do_get_time` (`iter_type` __beg, `iter_type` __end, `ios_base` &__io, `ios_base::iostate` &__err, `tm` *__tm) `const`
- virtual `iter_type` `do_get_weekday` (`iter_type` __beg, `iter_type` __end, `ios_base` &, `ios_base::iostate` &__err, `tm` *__tm) `const`
- virtual `iter_type` `do_get_year` (`iter_type` __beg, `iter_type` __end, `ios_base` &__io, `ios_base::iostate` &__err, `tm` *__tm) `const`

Static Protected Member Functions

- static `_c_locale` `_S_clone_c_locale` (`_c_locale` &__cloc) `throw ()`
- static `void` `_S_create_c_locale` (`_c_locale` &__cloc, `const char*` __s, `_c_locale` __old=0)
- static `void` `_S_destroy_c_locale` (`_c_locale` &__cloc)
- static `_c_locale` `_S_get_c_locale` ()
- static `const char*` `_S_get_c_name` () `throw ()`
- static `_c_locale` `_S_lc_ctype_c_locale` (`_c_locale` __cloc, `const char*` __s)

4.695.1 Detailed Description

```
template<typename _CharT, typename _InIter>
class std::time_get_byname<_CharT, _InIter>
```

class `time_get_byname` [22.2.5.2].

Definition at line 686 of file `locale_facets_nonio.h`.

4.695.2 Member Function Documentation

4.695.2.1 `template<typename _CharT, typename _InIter > dateorder std::time_get< _CharT, _InIter >::date_order () const`
`[inline], [inherited]`

Return preferred order of month, day, and year.

This function returns an enum from `timebase::dateorder` giving the preferred ordering if the format `x` given to `time_put::put()` only uses month, day, and year. If the format `x` for the associated locale uses other fields, this function returns `timebase::dateorder::noorder`.

NOTE: The library always returns `noorder` at the moment.

Returns

A member of `timebase::dateorder`.

Definition at line 407 of file `locale_facets_nonio.h`.

4.695.2.2 `template<typename _CharT, typename _InIter > virtual dateorder std::time_get< _CharT, _InIter >::do_date_order () const`
`[protected], [virtual], [inherited]`

Return preferred order of month, day, and year.

This function returns an enum from `timebase::dateorder` giving the preferred ordering if the format `x` given to `time_put::put()` only uses month, day, and year. This function is a hook for derived classes to change the value returned.

Returns

A member of `timebase::dateorder`.

4.695.2.3 `template<typename _CharT, typename _InIter > virtual iter_type std::time_get< _CharT, _InIter >::do_get_date (iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, tm * __tm) const`
`[protected], [virtual], [inherited]`

Parse input date string.

This function parses a date according to the format `X` and puts the results into a user-supplied struct `tm`. This function is a hook for derived classes to change the value returned.

See also

`get_date()` for details.

Parameters

<code>__beg</code>	Start of string to parse.
<code>__end</code>	End of string to parse.
<code>__io</code>	Source of the locale.
<code>__err</code>	Error flags to set.
<code>__tm</code>	Pointer to struct <code>tm</code> to fill in.

Returns

Iterator to first char beyond date string.

```
4.695.2.4 template<typename _CharT, typename _InIter> virtual iter_type std::time_get<_CharT, _InIter>::do_get_monthname( iter_type __beg, iter_type __end, ios_base &, ios_base::iostate & __err, tm * __tm ) const [protected], [virtual], [inherited]
```

Parse input month string.

This function parses a month name and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

See also

get_monthname() for details.

Parameters

<code>__beg</code>	Start of string to parse.
<code>__end</code>	End of string to parse.
<code>__io</code>	Source of the locale.
<code>__err</code>	Error flags to set.
<code>__tm</code>	Pointer to struct tm to fill in.

Returns

Iterator to first char beyond month name.

```
4.695.2.5 template<typename _CharT, typename _InIter> virtual iter_type std::time_get<_CharT, _InIter>::do_get_time( iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, tm * __tm ) const [protected], [virtual], [inherited]
```

Parse input time string.

This function parses a time according to the format x and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

See also

get_time() for details.

Parameters

<code>__beg</code>	Start of string to parse.
<code>__end</code>	End of string to parse.
<code>__io</code>	Source of the locale.
<code>__err</code>	Error flags to set.
<code>__tm</code>	Pointer to struct tm to fill in.

Returns

Iterator to first char beyond time string.

```
4.695.2.6  template<typename _CharT, typename _InIter > virtual iter_type std::time_get< _CharT, _InIter >::do_get_weekday
           ( iter_type __beg, iter_type __end, ios_base & , ios_base::iostate & __err, tm * __tm ) const
           [protected],[virtual],[inherited]
```

Parse input weekday string.

This function parses a weekday name and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

See also

get_weekday() for details.

Parameters

<code>__beg</code>	Start of string to parse.
<code>__end</code>	End of string to parse.
<code>__io</code>	Source of the locale.
<code>__err</code>	Error flags to set.
<code>__tm</code>	Pointer to struct tm to fill in.

Returns

Iterator to first char beyond weekday name.

```
4.695.2.7  template<typename _CharT, typename _InIter > virtual iter_type std::time_get< _CharT, _InIter >::do_get_year
           ( iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, tm * __tm ) const
           [protected],[virtual],[inherited]
```

Parse input year string.

This function reads up to 4 characters to parse a year string and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

See also

get_year() for details.

Parameters

<code>__beg</code>	Start of string to parse.
<code>__end</code>	End of string to parse.
<code>__io</code>	Source of the locale.
<code>__err</code>	Error flags to set.
<code>__tm</code>	Pointer to struct tm to fill in.

Returns

Iterator to first char beyond year.

4.695.2.8 `template<typename _CharT, typename _InIter> iter_type std::time_get<_CharT, _InIter>::get_date (iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, tm * __tm) const [inline], [inherited]`

Parse input date string.

This function parses a date according to the format *x* and puts the results into a user-supplied struct *tm*. The result is returned by calling `time_get::do_get_date()`.

If there is a valid date string according to format *x*, *tm* will be filled in accordingly and the returned iterator will point to the first character beyond the date string. If an error occurs before the end, `err |= ios_base::failbit`. If parsing reads all the characters, `err |= ios_base::eofbit`.

Parameters

<code>__beg</code>	Start of string to parse.
<code>__end</code>	End of string to parse.
<code>__io</code>	Source of the locale.
<code>__err</code>	Error flags to set.
<code>__tm</code>	Pointer to struct <i>tm</i> to fill in.

Returns

Iterator to first char beyond date string.

Definition at line 456 of file `locale_facets_nonio.h`.

4.695.2.9 `template<typename _CharT, typename _InIter> iter_type std::time_get<_CharT, _InIter>::get_monthname (iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, tm * __tm) const [inline], [inherited]`

Parse input month string.

This function parses a month name and puts the results into a user-supplied struct *tm*. The result is returned by calling `time_get::do_get_monthname()`.

Parsing starts by parsing an abbreviated month name. If a valid abbreviation is followed by a character that would lead to the full month name, parsing continues until the full name is found or an error occurs. Otherwise parsing finishes at the end of the abbreviated name.

If an error occurs before the end, `err |= ios_base::failbit`. If parsing reads all the characters, `err |= ios_base::eofbit`.

Parameters

<code>__beg</code>	Start of string to parse.
<code>__end</code>	End of string to parse.
<code>__io</code>	Source of the locale.
<code>__err</code>	Error flags to set.
<code>__tm</code>	Pointer to struct <i>tm</i> to fill in.

Returns

Iterator to first char beyond month name.

Definition at line 513 of file locale_facets_nonio.h.

```
4.695.2.10 template<typename _CharT, typename _InIter > iter_type std::time_get< _CharT, _InIter >::get_time ( iter_type
    __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, tm * __tm ) const [inline],
    [inherited]
```

Parse input time string.

This function parses a time according to the format *X* and puts the results into a user-supplied struct tm. The result is returned by calling time_get::do_get_time().

If there is a valid time string according to format *X*, *tm* will be filled in accordingly and the returned iterator will point to the first character beyond the time string. If an error occurs before the end, err |= ios_base::failbit. If parsing reads all the characters, err |= ios_base::eofbit.

Parameters

<code>__beg</code>	Start of string to parse.
<code>__end</code>	End of string to parse.
<code>__io</code>	Source of the locale.
<code>__err</code>	Error flags to set.
<code>__tm</code>	Pointer to struct tm to fill in.

Returns

Iterator to first char beyond time string.

Definition at line 431 of file locale_facets_nonio.h.

```
4.695.2.11 template<typename _CharT, typename _InIter > iter_type std::time_get< _CharT, _InIter >::get_weekday
    ( iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, tm * __tm ) const
    [inline],[inherited]
```

Parse input weekday string.

This function parses a weekday name and puts the results into a user-supplied struct tm. The result is returned by calling time_get::do_get_weekday().

Parsing starts by parsing an abbreviated weekday name. If a valid abbreviation is followed by a character that would lead to the full weekday name, parsing continues until the full name is found or an error occurs. Otherwise parsing finishes at the end of the abbreviated name.

If an error occurs before the end, err |= ios_base::failbit. If parsing reads all the characters, err |= ios_base::eofbit.

Parameters

<code>__beg</code>	Start of string to parse.
<code>__end</code>	End of string to parse.
<code>__io</code>	Source of the locale.
<code>__err</code>	Error flags to set.
<code>__tm</code>	Pointer to struct tm to fill in.

Returns

Iterator to first char beyond weekday name.

Definition at line 484 of file locale_facets_nonio.h.

```
4.695.2.12 template<typename _CharT, typename _InIter > iter_type std::time_get<_CharT, _InIter >::get_year ( iter_type
    __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, tm * __tm ) const [inline],
    [inherited]
```

Parse input year string.

This function reads up to 4 characters to parse a year string and puts the results into a user-supplied struct tm. The result is returned by calling time_get::do_get_year().

4 consecutive digits are interpreted as a full year. If there are exactly 2 consecutive digits, the library interprets this as the number of years since 1900.

If an error occurs before the end, err |= ios_base::failbit. If parsing reads all the characters, err |= ios_base::eofbit.

Parameters

<code>__beg</code>	Start of string to parse.
<code>__end</code>	End of string to parse.
<code>__io</code>	Source of the locale.
<code>__err</code>	Error flags to set.
<code>__tm</code>	Pointer to struct tm to fill in.

Returns

Iterator to first char beyond year.

Definition at line 539 of file locale_facets_nonio.h.

4.695.3 Member Data Documentation

4.695.3.1 `template<typename _CharT, typename _Inlter > locale::id std::time_get<_CharT, _Inlter >::id` `[static],`
`[inherited]`

Numpunct facet id.

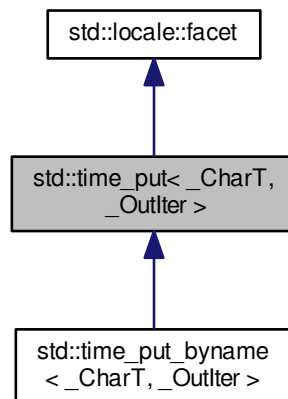
Definition at line 380 of file `locale_facets_nonio.h`.

The documentation for this class was generated from the following file:

- [locale_facets_nonio.h](#)

4.696 `std::time_put<_CharT, _Outlter >` Class Template Reference

Inheritance diagram for `std::time_put<_CharT, _Outlter >`:



Public Types

- typedef `_CharT` [char_type](#)
- typedef `_Outlter` [iter_type](#)

Public Member Functions

- [time_put](#) (`size_t __refs=0`)
- [iter_type put](#) ([iter_type](#) __s, [ios_base](#) &__io, [char_type](#) __fill, const tm *__tm, const `_CharT` *__beg, const `_CharT` *__end) const
- [iter_type put](#) ([iter_type](#) __s, [ios_base](#) &__io, [char_type](#) __fill, const tm *__tm, char __format, char __mod=0) const

Static Public Attributes

- static [locale::id](#) id

Protected Member Functions

- virtual [~time_put](#) ()
- virtual [iter_type do_put](#) ([iter_type](#) __s, [ios_base](#) &__io, [char_type](#) __fill, const tm *__tm, char __format, char __mod) const

Static Protected Member Functions

- static [__c_locale _S_clone_c_locale](#) ([__c_locale](#) &__cloc) throw ()
- static void [_S_create_c_locale](#) ([__c_locale](#) &__cloc, const char *__s, [__c_locale](#) __old=0)
- static void [_S_destroy_c_locale](#) ([__c_locale](#) &__cloc)
- static [__c_locale _S_get_c_locale](#) ()
- static const char * [_S_get_c_name](#) () throw ()
- static [__c_locale _S_lc_ctype_c_locale](#) ([__c_locale](#) __cloc, const char *__s)

4.696.1 Detailed Description

```
template<typename _CharT, typename _Outiter>
class std::time_put<_CharT, _Outiter>
```

Primary class template `time_put`.

This facet encapsulates the code to format and output dates and times according to formats used by `strftime()`.

The `time_put` template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the `time_put` facet.

Definition at line 715 of file `locale_facets_nonio.h`.

4.696.2 Member Typedef Documentation

4.696.2.1 `template<typename _CharT, typename _Outiter> typedef _CharT std::time_put<_CharT, _Outiter>::char_type`

Public typedefs.

Definition at line 721 of file `locale_facets_nonio.h`.

4.696.2.2 `template<typename _CharT, typename _Outiter> typedef _Outiter std::time_put<_CharT, _Outiter>::iter_type`

Public typedefs.

Definition at line 722 of file `locale_facets_nonio.h`.

4.696.3 Constructor & Destructor Documentation

4.696.3.1 `template<typename _CharT, typename _Outiter> std::time_put<_CharT, _Outiter>::time_put (size_t __refs = 0)`
`[inline], [explicit]`

Constructor performs initialization.

This is the constructor provided by the standard.

Parameters

<code>__refs</code>	Passed to the base facet class.
---------------------	---------------------------------

Definition at line 736 of file `locale_facets_nonio.h`.

4.696.3.2 `template<typename _CharT, typename _Outiter > virtual std::time_put< _CharT, _Outiter >::~time_put ()`
`[inline], [protected], [virtual]`

Destructor.

Definition at line 782 of file `locale_facets_nonio.h`.

4.696.4 Member Function Documentation

4.696.4.1 `template<typename _CharT, typename _Outiter > virtual iter_type std::time_put< _CharT, _Outiter >::do_put (`
`iter_type __s, ios_base & __io, char_type __fill, const tm * __tm, char __format, char __mod) const`
`[protected], [virtual]`

Format and output a time or date.

This function formats the data in struct `tm` according to the provided format char and optional modifier. This function is a hook for derived classes to change the value returned.

See also

`put()` for more details.

Parameters

<code>__s</code>	The stream to write to.
<code>__io</code>	Source of locale.
<code>__fill</code>	<code>char_type</code> to use for padding.
<code>__tm</code>	Struct <code>tm</code> with date and time info to format.
<code>__format</code>	Format char.
<code>__mod</code>	Optional modifier char.

Returns

Iterator after writing.

4.696.4.2 `template<typename _CharT, typename _Outiter > iter_type std::time_put< _CharT, _Outiter >::put (iter_type`
`__s, ios_base & __io, char_type __fill, const tm * __tm, const _CharT * __beg, const _CharT * __end) const`

Format and output a time or date.

This function formats the data in struct tm according to the provided format string. The format string is interpreted as by `strftime()`.

Parameters

<code>__s</code>	The stream to write to.
<code>__io</code>	Source of locale.
<code>__fill</code>	<code>char_type</code> to use for padding.
<code>__tm</code>	Struct <code>tm</code> with date and time info to format.
<code>__beg</code>	Start of format string.
<code>__end</code>	End of format string.

Returns

Iterator after writing.

4.696.4.3 `template<typename _CharT, typename _Outiter> iter_type std::time_put<_CharT, _Outiter>::put (iter_type __s, ios_base & __io, char_type __fill, const tm * __tm, char __format, char __mod = 0) const [inline]`

Format and output a time or date.

This function formats the data in struct `tm` according to the provided format char and optional modifier. The format and modifier are interpreted as by `strftime()`. It does so by returning `time_put::do_put()`.

Parameters

<code>__s</code>	The stream to write to.
<code>__io</code>	Source of locale.
<code>__fill</code>	<code>char_type</code> to use for padding.
<code>__tm</code>	Struct <code>tm</code> with date and time info to format.
<code>__format</code>	Format char.
<code>__mod</code>	Optional modifier char.

Returns

Iterator after writing.

Definition at line 775 of file `locale_facets_nonio.h`.

4.696.5 Member Data Documentation

4.696.5.1 `template<typename _CharT, typename _Outiter> locale::id std::time_put<_CharT, _Outiter>::id [static]`

Numpunct facet id.

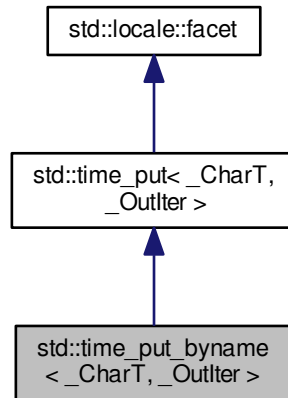
Definition at line 726 of file `locale_facets_nonio.h`.

The documentation for this class was generated from the following file:

- [locale_facets_nonio.h](#)

4.697 std::time_put_byname< _CharT, _Outlter > Class Template Reference

Inheritance diagram for std::time_put_byname< _CharT, _Outlter >:



Public Types

- typedef `_CharT` **char_type**
- typedef `_Outlter` **iter_type**

Public Member Functions

- **time_put_byname** (const char *, size_t __refs=0)
- **iter_type put** (iter_type __s, ios_base &__io, char_type __fill, const tm *__tm, const _CharT *__beg, const _CharT *__end) const
- **iter_type put** (iter_type __s, ios_base &__io, char_type __fill, const tm *__tm, char __format, char __mod=0) const

Static Public Attributes

- static `locale::id` **id**

Protected Member Functions

- virtual **iter_type do_put** (iter_type __s, ios_base &__io, char_type __fill, const tm *__tm, char __format, char __mod) const

Static Protected Member Functions

- static `__c_locale _S_clone_c_locale (__c_locale &__cloc) throw ()`
- static void `_S_create_c_locale (__c_locale &__cloc, const char *__s, __c_locale __old=0)`
- static void `_S_destroy_c_locale (__c_locale &__cloc)`
- static `__c_locale _S_get_c_locale ()`
- static const char * `_S_get_c_name () throw ()`
- static `__c_locale _S_lc_ctype_c_locale (__c_locale __cloc, const char *__s)`

4.697.1 Detailed Description

```
template<typename _CharT, typename _Outiter>
class std::time_put_byname< _CharT, _Outiter >
```

class `time_put_byname` [22.2.5.4].

Definition at line 811 of file `locale_facets_nonio.h`.

4.697.2 Member Function Documentation

4.697.2.1 `template<typename _CharT, typename _Outiter> virtual iter_type std::time_put< _CharT, _Outiter >::do_put (iter_type __s, ios_base & __io, char_type __fill, const tm * __tm, char __format, char __mod) const`
 [protected], [virtual], [inherited]

Format and output a time or date.

This function formats the data in struct `tm` according to the provided format char and optional modifier. This function is a hook for derived classes to change the value returned.

See also

`put()` for more details.

Parameters

<code>__s</code>	The stream to write to.
<code>__io</code>	Source of locale.
<code>__fill</code>	<code>char_type</code> to use for padding.
<code>__tm</code>	Struct <code>tm</code> with date and time info to format.
<code>__format</code>	Format char.
<code>__mod</code>	Optional modifier char.

Returns

Iterator after writing.

4.697.2.2 `template<typename _CharT, typename _Outiter > iter_type std::time_put<_CharT, _Outiter >::put (iter_type __s, ios_base & __io, char_type __fill, const tm * __tm, const _CharT * __beg, const _CharT * __end) const`
`[inherited]`

Format and output a time or date.

This function formats the data in struct tm according to the provided format string. The format string is interpreted as by `strftime()`.

Parameters

<code>__s</code>	The stream to write to.
<code>__io</code>	Source of locale.
<code>__fill</code>	char_type to use for padding.
<code>__tm</code>	Struct tm with date and time info to format.
<code>__beg</code>	Start of format string.
<code>__end</code>	End of format string.

Returns

Iterator after writing.

4.697.2.3 `template<typename _CharT, typename _Outiter > iter_type std::time_put<_CharT, _Outiter >::put (iter_type __s, ios_base & __io, char_type __fill, const tm * __tm, char __format, char __mod = 0) const` `[inline]`,
`[inherited]`

Format and output a time or date.

This function formats the data in struct tm according to the provided format char and optional modifier. The format and modifier are interpreted as by `strftime()`. It does so by returning `time_put::do_put()`.

Parameters

<code>__s</code>	The stream to write to.
<code>__io</code>	Source of locale.
<code>__fill</code>	char_type to use for padding.
<code>__tm</code>	Struct tm with date and time info to format.
<code>__format</code>	Format char.
<code>__mod</code>	Optional modifier char.

Returns

Iterator after writing.

Definition at line 775 of file `locale_facets_nonio.h`.

4.697.3 Member Data Documentation

4.697.3.1 `template<typename _CharT, typename _Outiter> locale::id std::time_put<_CharT, _Outiter>::id` `[static]`, `[inherited]`

Numpunct facet id.

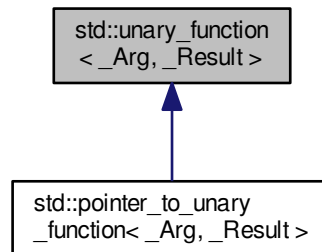
Definition at line 726 of file `locale_facets_nonio.h`.

The documentation for this class was generated from the following file:

- [locale_facets_nonio.h](#)

4.698 `std::unary_function<_Arg, _Result>` Struct Template Reference

Inheritance diagram for `std::unary_function<_Arg, _Result>`:



Public Types

- typedef `_Arg` [argument_type](#)
- typedef `_Result` [result_type](#)

4.698.1 Detailed Description

```
template<typename _Arg, typename _Result>
struct std::unary_function<_Arg, _Result>
```

This is one of the [functor base classes](#).

Definition at line 105 of file `stl_function.h`.

4.698.2 Member Typedef Documentation

4.698.2.1 template<typename _Arg, typename _Result> typedef _Arg std::unary_function< _Arg, _Result >::argument_type

argument_type is the type of the argument

Definition at line 108 of file stl_function.h.

4.698.2.2 template<typename _Arg, typename _Result> typedef _Result std::unary_function< _Arg, _Result >::result_type

result_type is the return type

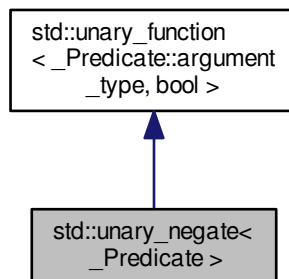
Definition at line 111 of file stl_function.h.

The documentation for this struct was generated from the following file:

- [stl_function.h](#)

4.699 std::unary_negate< _Predicate > Class Template Reference

Inheritance diagram for std::unary_negate< _Predicate >:



Public Types

- typedef `_Predicate::argument_type` [argument_type](#)
- typedef `bool` [result_type](#)

Public Member Functions

- **unary_negate** (`const _Predicate &__x`)
- **bool operator()** (`const typename _Predicate::argument_type &__x`) `const`

Protected Attributes

- `_Predicate` `_M_pred`

4.699.1 Detailed Description

```
template<typename _Predicate>
class std::unary_negate< _Predicate >
```

One of the [negation functors](#).

Definition at line 704 of file `stl_function.h`.

4.699.2 Member Typedef Documentation

4.699.2.1 `typedef _Predicate::argument_type std::unary_function< _Predicate::argument_type, bool >::argument_type` `[inherited]`

`argument_type` is the type of the argument

Definition at line 108 of file `stl_function.h`.

4.699.2.2 `typedef bool std::unary_function< _Predicate::argument_type, bool >::result_type` `[inherited]`

`result_type` is the return type

Definition at line 111 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [stl_function.h](#)

4.700 `std::uniform_int_distribution< _IntType >` Class Template Reference

Classes

- struct [param_type](#)

Public Types

- `typedef _IntType` [result_type](#)

Public Member Functions

- [uniform_int_distribution](#) (_IntType __a=0, _IntType __b=std::numeric_limits< _IntType >::max())
- [uniform_int_distribution](#) (const [param_type](#) &__p)
- [template](#)<typename _ForwardIterator, typename _UniformRandomNumberGenerator >
void [__generate](#) (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng)
- [template](#)<typename _ForwardIterator, typename _UniformRandomNumberGenerator >
void [__generate](#) (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const [param_type](#) &__p)
- [template](#)<typename _UniformRandomNumberGenerator >
void [__generate](#) ([result_type](#) *__f, [result_type](#) *__t, _UniformRandomNumberGenerator &__urng, const [param_type](#) &__p)
- [result_type](#) [a](#) () const
- [result_type](#) [b](#) () const
- [result_type](#) [max](#) () const
- [result_type](#) [min](#) () const
- [template](#)<typename _UniformRandomNumberGenerator >
[result_type](#) [operator\(\)](#) (_UniformRandomNumberGenerator &__urng)
- [template](#)<typename _UniformRandomNumberGenerator >
[result_type](#) [operator\(\)](#) (_UniformRandomNumberGenerator &__urng, const [param_type](#) &__p)
- [param_type](#) [param](#) () const
- void [param](#) (const [param_type](#) &__param)
- void [reset](#) ()

Friends

- bool [operator==](#) (const [uniform_int_distribution](#) &__d1, const [uniform_int_distribution](#) &__d2)

4.700.1 Detailed Description

```
template<typename _IntType = int>
class std::uniform_int_distribution< _IntType >
```

Uniform discrete distribution for random numbers. A discrete random distribution on the range $[min, max]$ with equal probability throughout the range.

Definition at line 1668 of file random.h.

4.700.2 Member Typedef Documentation

4.700.2.1 [template](#)<typename _IntType = int> typedef _IntType [std::uniform_int_distribution](#)< _IntType >::[result_type](#)

The type of the range of the distribution.

Definition at line 1671 of file random.h.

4.700.3 Constructor & Destructor Documentation

4.700.3.1 `template<typename _IntType = int> std::uniform_int_distribution< _IntType >::uniform_int_distribution (_IntType __a = 0, _IntType __b = std::numeric_limits<_IntType>::max()) [inline], [explicit]`

Constructs a uniform distribution object.

Definition at line 1711 of file random.h.

4.700.4 Member Function Documentation

4.700.4.1 `template<typename _IntType = int> result_type std::uniform_int_distribution< _IntType >::max () const [inline]`

Returns the inclusive upper bound of the distribution range.

Definition at line 1763 of file random.h.

4.700.4.2 `template<typename _IntType = int> result_type std::uniform_int_distribution< _IntType >::min () const [inline]`

Returns the inclusive lower bound of the distribution range.

Definition at line 1756 of file random.h.

4.700.4.3 `template<typename _IntType = int> template<typename _UniformRandomNumberGenerator > result_type std::uniform_int_distribution< _IntType >::operator() (_UniformRandomNumberGenerator & __urng) [inline]`

Generating functions.

Definition at line 1771 of file random.h.

4.700.4.4 `template<typename _IntType = int> param_type std::uniform_int_distribution< _IntType >::param () const [inline]`

Returns the parameter set of the distribution.

Definition at line 1741 of file random.h.

4.700.4.5 `template<typename _IntType = int> void std::uniform_int_distribution< _IntType >::param (const param_type & __param) [inline]`

Sets the parameter set of the distribution.

Parameters

<code>__param</code>	The new parameter set of the distribution.
----------------------	--

Definition at line 1749 of file `random.h`.

4.700.4.6 `template<typename _IntType = int> void std::uniform_int_distribution<_IntType>::reset () [inline]`

Resets the distribution state.

Does nothing for the uniform integer distribution.

Definition at line 1727 of file `random.h`.

4.700.5 Friends And Related Function Documentation

4.700.5.1 `template<typename _IntType = int> bool operator== (const uniform_int_distribution<_IntType> &__d1, const uniform_int_distribution<_IntType> &__d2) [friend]`

Return true if two uniform integer distributions have the same parameters.

Definition at line 1806 of file `random.h`.

The documentation for this class was generated from the following file:

- [random.h](#)

4.701 `std::uniform_int_distribution<_IntType>::param_type` Struct Reference

Public Types

- typedef `uniform_int_distribution<_IntType>` **distribution_type**

Public Member Functions

- **param_type** (`_IntType __a=0, _IntType __b=std::numeric_limits<_IntType>::max()`)
- **result_type a** () const
- **result_type b** () const

Friends

- bool **operator==** (const `param_type` &__p1, const `param_type` &__p2)

4.701.1 Detailed Description

```
template<typename _IntType = int>
struct std::uniform_int_distribution< _IntType >::param_type
```

Parameter type.

Definition at line 1677 of file random.h.

The documentation for this struct was generated from the following file:

- [random.h](#)

4.702 std::uniform_real_distribution< _RealType > Class Template Reference

Classes

- struct [param_type](#)

Public Types

- typedef _RealType [result_type](#)

Public Member Functions

- [uniform_real_distribution](#) (_RealType __a=_RealType(0), _RealType __b=_RealType(1))
- **uniform_real_distribution** (const [param_type](#) &__p)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator >
void **generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator >
void **generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const [param_type](#) &__p)
- template<typename _UniformRandomNumberGenerator >
void **generate** ([result_type](#) *__f, [result_type](#) *__t, _UniformRandomNumberGenerator &__urng, const [param_type](#) &__p)
- [result_type](#) **a** () const
- [result_type](#) **b** () const
- [result_type](#) **max** () const
- [result_type](#) **min** () const
- template<typename _UniformRandomNumberGenerator >
[result_type](#) **operator()** (_UniformRandomNumberGenerator &__urng)
- template<typename _UniformRandomNumberGenerator >
[result_type](#) **operator()** (_UniformRandomNumberGenerator &__urng, const [param_type](#) &__p)
- [param_type](#) **param** () const
- void **param** (const [param_type](#) &__param)
- void **reset** ()

Friends

- bool `operator==` (const `uniform_real_distribution` &__d1, const `uniform_real_distribution` &__d2)

4.702.1 Detailed Description

```
template<typename _RealType = double>
class std::uniform_real_distribution<_RealType>
```

Uniform continuous distribution for random numbers.

A continuous random distribution on the range [min, max) with equal probability throughout the range. The URNG should be real-valued and deliver number in the range [0, 1).

Definition at line 1869 of file random.h.

4.702.2 Member Typedef Documentation

4.702.2.1 `template<typename _RealType = double> typedef _RealType std::uniform_real_distribution<_RealType>::result_type`

The type of the range of the distribution.

Definition at line 1872 of file random.h.

4.702.3 Constructor & Destructor Documentation

4.702.3.1 `template<typename _RealType = double> std::uniform_real_distribution<_RealType>::uniform_real_distribution (_RealType __a = _RealType(0), _RealType __b = _RealType(1))`
`[inline], [explicit]`

Constructs a `uniform_real_distribution` object.

Parameters

<code>↔</code> __a	[IN] The lower bound of the distribution.
<code>↔</code> __b	[IN] The upper bound of the distribution.

Definition at line 1915 of file random.h.

4.702.4 Member Function Documentation

4.702.4.1 `template<typename _RealType = double> result_type std::uniform_real_distribution< _RealType >::max ()`
`const [inline]`

Returns the inclusive upper bound of the distribution range.

Definition at line 1967 of file random.h.

4.702.4.2 `template<typename _RealType = double> result_type std::uniform_real_distribution< _RealType >::min ()`
`const [inline]`

Returns the inclusive lower bound of the distribution range.

Definition at line 1960 of file random.h.

4.702.4.3 `template<typename _RealType = double> template<typename _UniformRandomNumberGenerator > result_type`
`std::uniform_real_distribution< _RealType >::operator() (_UniformRandomNumberGenerator & __urng)`
`[inline]`

Generating functions.

Definition at line 1975 of file random.h.

4.702.4.4 `template<typename _RealType = double> param_type std::uniform_real_distribution< _RealType >::param ()`
`const [inline]`

Returns the parameter set of the distribution.

Definition at line 1945 of file random.h.

4.702.4.5 `template<typename _RealType = double> void std::uniform_real_distribution< _RealType >::param (const`
`param_type & __param) [inline]`

Sets the parameter set of the distribution.

Parameters

<code>__param</code>	The new parameter set of the distribution.
----------------------	--

Definition at line 1953 of file random.h.

4.702.4.6 `template<typename _RealType = double> void std::uniform_real_distribution< _RealType >::reset ()`
`[inline]`

Resets the distribution state.

Does nothing for the uniform real distribution.

Definition at line 1931 of file random.h.

4.702.5 Friends And Related Function Documentation

4.702.5.1 `template<typename _RealType = double> bool operator==(const uniform_real_distribution<_RealType> &__d1, const uniform_real_distribution<_RealType> &__d2) [friend]`

Return true if two uniform real distributions have the same parameters.

Definition at line 2015 of file random.h.

The documentation for this class was generated from the following file:

- [random.h](#)

4.703 std::uniform_real_distribution<_RealType>::param_type Struct Reference

Public Types

- typedef [uniform_real_distribution](#)<_RealType> **distribution_type**

Public Member Functions

- **param_type** (_RealType __a=_RealType(0), _RealType __b=_RealType(1))
- [result_type](#) **a** () const
- [result_type](#) **b** () const

Friends

- bool **operator==** (const [param_type](#) &__p1, const [param_type](#) &__p2)

4.703.1 Detailed Description

```
template<typename _RealType = double>
struct std::uniform_real_distribution<_RealType>::param_type
```

Parameter type.

Definition at line 1878 of file random.h.

The documentation for this struct was generated from the following file:

- [random.h](#)

4.704 std::unique_ptr<_Tp, _Dp> Class Template Reference

Public Types

- typedef _Dp **deleter_type**
- typedef _Tp **element_type**
- typedef _Pointer::type **pointer**

Public Member Functions

- constexpr [unique_ptr](#) () noexcept
- [unique_ptr](#) (pointer __p) noexcept
- [unique_ptr](#) (pointer __p, typename conditional< is_reference< deleter_type >::value, deleter_type, const deleter_type & >::type __d) noexcept
- [unique_ptr](#) (pointer __p, typename remove_reference< deleter_type >::type &&__d) noexcept
- constexpr [unique_ptr](#) (nullptr_t) noexcept
- [unique_ptr](#) ([unique_ptr](#) &&__u) noexcept
- template<typename _Up, typename _Ep, typename = _Require< is_convertible<typename [unique_ptr](#)<_Up, _Ep>::pointer, pointer>, __not_<is_array<_Up>>, typename conditional<is_reference<_Dp>::value, is_same<_Ep, _Dp>, is_convertible<_Ep, _Dp>>>::type>>
[unique_ptr](#) ([unique_ptr](#)< _Up, _Ep > &&__u) noexcept
- template<typename _Up, typename >
[unique_ptr](#) ([auto_ptr](#)< _Up > &&__u) noexcept
- **[unique_ptr](#)** (const [unique_ptr](#) &)=delete
- ~[unique_ptr](#) () noexcept
- pointer [get](#) () const noexcept
- deleter_type & [get_deleter](#) () noexcept
- const deleter_type & [get_deleter](#) () const noexcept
- [operator bool](#) () const noexcept
- add_lvalue_reference< element_type >::type [operator*](#) () const
- pointer [operator->](#) () const noexcept
- [unique_ptr](#) & [operator=](#) ([unique_ptr](#) &&__u) noexcept
- template<typename _Up, typename _Ep >
enable_if< __and< is_convertible< typename [unique_ptr](#)< _Up, _Ep >::pointer, pointer >, __not_< is_array< _Up > > >::value, [unique_ptr](#) & >::type [operator=](#) ([unique_ptr](#)< _Up, _Ep > &&__u) noexcept
- [unique_ptr](#) & [operator=](#) (nullptr_t) noexcept
- [unique_ptr](#) & **[operator=](#)** (const [unique_ptr](#) &)=delete
- pointer [release](#) () noexcept
- void [reset](#) (pointer __p=pointer()) noexcept
- void [swap](#) ([unique_ptr](#) &__u) noexcept

4.704.1 Detailed Description

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
class std::unique_ptr<_Tp, _Dp>
```

20.7.1.2 unique_ptr for single objects.

Definition at line 129 of file unique_ptr.h.

4.704.2 Constructor & Destructor Documentation

4.704.2.1 `template<typename _Tp, typename _Dp = default_delete<_Tp>> constexpr std::unique_ptr< _Tp, _Dp >::unique_ptr () [inline], [noexcept]`

Default constructor, creates a unique_ptr that owns nothing.

Definition at line 157 of file unique_ptr.h.

Referenced by std::auto_ptr< _Tp >::auto_ptr().

4.704.2.2 `template<typename _Tp, typename _Dp = default_delete<_Tp>> std::unique_ptr< _Tp, _Dp >::unique_ptr (pointer __p) [inline], [explicit], [noexcept]`

Takes ownership of a pointer.

Parameters

<code>__p</code>	A pointer to an object of <code>element_type</code>
------------------	---

The deleter will be value-initialized.

Definition at line 169 of file unique_ptr.h.

4.704.2.3 `template<typename _Tp, typename _Dp = default_delete<_Tp>> std::unique_ptr< _Tp, _Dp >::unique_ptr (pointer __p, typename conditional< is_reference< deleter_type >::value, deleter_type, const deleter_type & >::type __d) [inline], [noexcept]`

Takes ownership of a pointer.

Parameters

<code>__p</code>	A pointer to an object of <code>element_type</code>
<code>__d</code>	A reference to a deleter.

The deleter will be initialized with `__d`

Definition at line 181 of file unique_ptr.h.

4.704.2.4 `template<typename _Tp, typename _Dp = default_delete<_Tp>> std::unique_ptr< _Tp, _Dp >::unique_ptr (pointer __p, typename remove_reference< deleter_type >::type && __d) [inline], [noexcept]`

Takes ownership of a pointer.

Parameters

\leftrightarrow _p	A pointer to an object of <code>element_type</code>
\leftrightarrow _d	An rvalue reference to a deleter.

The deleter will be initialized with `std::move(__d)`

Definition at line 193 of file `unique_ptr.h`.

4.704.2.5 `template<typename _Tp, typename _Dp = default_delete<_Tp>> constexpr std::unique_ptr<_Tp, _Dp>::unique_ptr(nullptr_t) [inline], [noexcept]`

Creates a `unique_ptr` that owns nothing.

Definition at line 200 of file `unique_ptr.h`.

4.704.2.6 `template<typename _Tp, typename _Dp = default_delete<_Tp>> std::unique_ptr<_Tp, _Dp>::unique_ptr(unique_ptr<_Tp, _Dp> && __u) [inline], [noexcept]`

Move constructor.

Definition at line 205 of file `unique_ptr.h`.

4.704.2.7 `template<typename _Tp, typename _Dp = default_delete<_Tp>> template<typename _Up, typename _Ep, typename = Require< is_convertible<typename unique_ptr<_Up, _Ep>::pointer, pointer>, __not__<is_array<_Up>>, typename conditional<is_reference<_Dp>::value, is_same<_Ep, _Dp>, is_convertible<_Ep, _Dp>>::type>> std::unique_ptr<_Tp, _Dp>::unique_ptr(unique_ptr<_Up, _Ep> && __u) [inline], [noexcept]`

Converting constructor from another type.

Requires that the pointer owned by `__u` is convertible to the type of pointer owned by this object, `__u` does not own an array, and `__u` has a compatible deleter type.

Definition at line 220 of file `unique_ptr.h`.

4.704.2.8 `template<typename _Tp, typename _Dp = default_delete<_Tp>> std::unique_ptr<_Tp, _Dp>::~~unique_ptr() [inline], [noexcept]`

Destructor, invokes the deleter if the stored pointer is not null.

Definition at line 232 of file `unique_ptr.h`.

4.704.3 Member Function Documentation

4.704.3.1 `template<typename _Tp, typename _Dp = default_delete<_Tp>> pointer std::unique_ptr< _Tp, _Dp >::get ()
const [inline], [noexcept]`

Return the stored pointer.

Definition at line 304 of file `unique_ptr.h`.

Referenced by `std::unique_ptr< _Tp[], _Dp >::swap()`.

4.704.3.2 `template<typename _Tp, typename _Dp = default_delete<_Tp>> deleter_type& std::unique_ptr< _Tp, _Dp
>::get_deleter () [inline], [noexcept]`

Return a reference to the stored deleter.

Definition at line 309 of file `unique_ptr.h`.

4.704.3.3 `template<typename _Tp, typename _Dp = default_delete<_Tp>> const deleter_type& std::unique_ptr< _Tp, _Dp
>::get_deleter () const [inline], [noexcept]`

Return a reference to the stored deleter.

Definition at line 314 of file `unique_ptr.h`.

4.704.3.4 `template<typename _Tp, typename _Dp = default_delete<_Tp>> std::unique_ptr< _Tp, _Dp >::operator bool ()
const [inline], [explicit], [noexcept]`

Return `true` if the stored pointer is not null.

Definition at line 318 of file `unique_ptr.h`.

4.704.3.5 `template<typename _Tp, typename _Dp = default_delete<_Tp>> add_lvalue_reference<element_type>::type
std::unique_ptr< _Tp, _Dp >::operator*() const [inline]`

Dereference the stored pointer.

Definition at line 288 of file `unique_ptr.h`.

4.704.3.6 `template<typename _Tp, typename _Dp = default_delete<_Tp>> pointer std::unique_ptr< _Tp, _Dp >::operator-> () const [inline], [noexcept]`

Return the stored pointer.

Definition at line 296 of file `unique_ptr.h`.

4.704.3.7 `template<typename _Tp, typename _Dp = default_delete<_Tp>> unique_ptr& std::unique_ptr< _Tp, _Dp
>::operator=(unique_ptr< _Tp, _Dp > && __u) [inline], [noexcept]`

Move assignment operator.

Parameters

<code>_↔</code>	The object to transfer ownership from.
<code>_u</code>	

Invokes the deleter first if this object owns a pointer.

Definition at line 249 of file `unique_ptr.h`.

```
4.704.3.8 template<typename _Tp, typename _Dp = default_delete<_Tp>> template<typename _Up, typename _Ep >
enable_if< __and_< is_convertible<typename unique_ptr<_Up, _Ep>::pointer, pointer>, __not_<is_array<_Up>
> >::value, unique_ptr<>::type std::unique_ptr<_Tp, _Dp >::operator= ( unique_ptr<_Up, _Ep > && _u )
[inline], [noexcept]
```

Assignment from another type.

Parameters

<code>_↔</code>	The object to transfer ownership from, which owns a convertible pointer to a non-array object.
<code>_u</code>	

Invokes the deleter first if this object owns a pointer.

Definition at line 269 of file `unique_ptr.h`.

```
4.704.3.9 template<typename _Tp, typename _Dp = default_delete<_Tp>> unique_ptr& std::unique_ptr<_Tp, _Dp
>::operator= ( nullptr_t ) [inline], [noexcept]
```

Reset the `unique_ptr` to empty, invoking the deleter if necessary.

Definition at line 278 of file `unique_ptr.h`.

```
4.704.3.10 template<typename _Tp, typename _Dp = default_delete<_Tp>> pointer std::unique_ptr<_Tp, _Dp >::release ( )
[inline], [noexcept]
```

Release ownership of any stored pointer.

Definition at line 325 of file `unique_ptr.h`.

```
4.704.3.11 template<typename _Tp, typename _Dp = default_delete<_Tp>> void std::unique_ptr<_Tp, _Dp >::reset ( pointer
_p=pointer() ) [inline], [noexcept]
```

Replace the stored pointer.

Parameters

<code>_↔</code>	The new pointer to store.
<code>_p</code>	

The deleter will be invoked if a pointer is already owned.

Definition at line 339 of file unique_ptr.h.

4.704.3.12 `template<typename _Tp, typename _Dp = default_delete<_Tp>> void std::unique_ptr<_Tp, _Dp>::swap (unique_ptr<_Tp, _Dp> &__u) [inline], [noexcept]`

Exchange the pointer and deleter with another object.

Definition at line 349 of file unique_ptr.h.

Referenced by `std::unique_ptr<_Tp[], _Dp>::swap()`.

The documentation for this class was generated from the following files:

- [unique_ptr.h](#)
- [auto_ptr.h](#)

4.705 std::unique_ptr< _Tp[], _Dp > Class Template Reference

Public Types

- typedef `_Dp` **deleter_type**
- typedef `_Tp` **element_type**
- typedef `_Pointer::type` **pointer**

Public Member Functions

- constexpr [unique_ptr](#) () noexcept
- [unique_ptr](#) (pointer __p) noexcept
- template<typename _Up, typename = _Require<is_pointer<pointer>, is_convertible<_Up*, pointer>, __is_derived_Tp<_Up>>> [unique_ptr](#) (_Up *__p)=delete
- [unique_ptr](#) (pointer __p, typename conditional< is_reference< deleter_type >::value, deleter_type, const deleter_type & >::type __d) noexcept
- [unique_ptr](#) (pointer __p, typename remove_reference< deleter_type >::type &&__d) noexcept
- [unique_ptr](#) ([unique_ptr](#) &&__u) noexcept
- constexpr [unique_ptr](#) (nullptr_t) noexcept
- template<typename _Up, typename _Ep, typename = _Require<__safe_conversion<_Up, _Ep>, typename conditional<is_reference<__Dp>::value, is_same<_Ep, _Dp>, is_convertible<_Ep, _Dp>>::type >> [unique_ptr](#) ([unique_ptr](#)<_Up, _Ep> &&__u) noexcept
- [unique_ptr](#) (const [unique_ptr](#) &)=delete
- template<typename _Up, typename = _Require<is_pointer<pointer>, is_convertible<_Up*, pointer>, __is_derived_Tp<_Up>>> [unique_ptr](#) (_Up *, typename conditional< is_reference< deleter_type >::value, deleter_type, const deleter_type & >::type)=delete
- template<typename _Up, typename = _Require<is_pointer<pointer>, is_convertible<_Up*, pointer>, __is_derived_Tp<_Up>>> [unique_ptr](#) (_Up *, typename remove_reference< deleter_type >::type &&)=delete
- [~unique_ptr](#) ()
- pointer [get](#) () const noexcept
- deleter_type & [get_deleter](#) () noexcept

- `const deleter_type & get_deleter () const noexcept`
- `operator bool () const noexcept`
- `unique_ptr & operator= (unique_ptr && __u) noexcept`
- `template<typename _Up, typename _Ep >
enable_if< __safe_conversion< _Up, _Ep >::value, unique_ptr & >::type operator= (unique_ptr< _Up, _Ep >
&& __u) noexcept`
- `unique_ptr & operator= (nullptr_t) noexcept`
- `unique_ptr & operator= (const unique_ptr &)=delete`
- `std::add_lvalue_reference< element_type >::type operator[] (size_t __i) const`
- `pointer release () noexcept`
- `void reset (pointer __p=pointer()) noexcept`
- `template<typename _Up, typename = _Require<is_pointer<pointer>, is_convertible<_Up*, pointer>, __is_derived_Tp<_Up>>>
void reset (_Up *)=delete`
- `void swap (unique_ptr & __u) noexcept`

4.705.1 Detailed Description

```
template<typename _Tp, typename _Dp>
class std::unique_ptr< _Tp[], _Dp >
```

20.7.1.3 unique_ptr for array objects with a runtime length

Definition at line 365 of file `unique_ptr.h`.

4.705.2 Constructor & Destructor Documentation

4.705.2.1 `template<typename _Tp, typename _Dp > constexpr std::unique_ptr< _Tp[], _Dp >::unique_ptr ()
[inline], [noexcept]`

Default constructor, creates a `unique_ptr` that owns nothing.

Definition at line 414 of file `unique_ptr.h`.

4.705.2.2 `template<typename _Tp, typename _Dp > std::unique_ptr< _Tp[], _Dp >::unique_ptr (pointer __p)
[inline], [explicit], [noexcept]`

Takes ownership of a pointer.

Parameters

<code>__p</code>	A pointer to an array of <code>element_type</code>
------------------	--

The deleter will be value-initialized.

Definition at line 426 of file `unique_ptr.h`.

4.705.2.3 `template<typename _Tp, typename _Dp > std::unique_ptr< _Tp[], _Dp >::unique_ptr (pointer __p, typename conditional< is_reference< deleter_type >::value, deleter_type, const deleter_type & >::type __d) [inline], [noexcept]`

Takes ownership of a pointer.

Parameters

<code>__p</code>	A pointer to an array of <code>element_type</code>
<code>__d</code>	A reference to a deleter.

The deleter will be initialized with `__d`

Definition at line 444 of file `unique_ptr.h`.

4.705.2.4 `template<typename _Tp, typename _Dp > std::unique_ptr< _Tp[], _Dp >::unique_ptr (pointer __p, typename remove_reference< deleter_type >::type && __d) [inline], [noexcept]`

Takes ownership of a pointer.

Parameters

<code>__p</code>	A pointer to an array of <code>element_type</code>
<code>__d</code>	A reference to a deleter.

The deleter will be initialized with `std::move(__d)`

Definition at line 456 of file `unique_ptr.h`.

4.705.2.5 `template<typename _Tp, typename _Dp > std::unique_ptr< _Tp[], _Dp >::unique_ptr (unique_ptr< _Tp[], _Dp > && __u) [inline], [noexcept]`

Move constructor.

Definition at line 463 of file `unique_ptr.h`.

4.705.2.6 `template<typename _Tp, typename _Dp > constexpr std::unique_ptr< _Tp[], _Dp >::unique_ptr (nullptr_t) [inline], [noexcept]`

Creates a `unique_ptr` that owns nothing.

Definition at line 467 of file `unique_ptr.h`.

4.705.2.7 `template<typename _Tp, typename _Dp> std::unique_ptr<_Tp[], _Dp>::~~unique_ptr () [inline]`

Destructor, invokes the deleter if the stored pointer is not null.

Definition at line 480 of file `unique_ptr.h`.

References `std::get_deleter()`.

4.705.3 Member Function Documentation

4.705.3.1 `template<typename _Tp, typename _Dp> pointer std::unique_ptr<_Tp[], _Dp>::get () const [inline], [noexcept]`

Return the stored pointer.

Definition at line 541 of file `unique_ptr.h`.

4.705.3.2 `template<typename _Tp, typename _Dp> deleter_type& std::unique_ptr<_Tp[], _Dp>::get_deleter () [inline], [noexcept]`

Return a reference to the stored deleter.

Definition at line 546 of file `unique_ptr.h`.

4.705.3.3 `template<typename _Tp, typename _Dp> const deleter_type& std::unique_ptr<_Tp[], _Dp>::get_deleter () const [inline], [noexcept]`

Return a reference to the stored deleter.

Definition at line 551 of file `unique_ptr.h`.

4.705.3.4 `template<typename _Tp, typename _Dp> std::unique_ptr<_Tp[], _Dp>::operator bool () const [inline], [explicit], [noexcept]`

Return `true` if the stored pointer is not null.

Definition at line 555 of file `unique_ptr.h`.

4.705.3.5 `template<typename _Tp, typename _Dp> unique_ptr& std::unique_ptr<_Tp[], _Dp>::operator= (unique_ptr<_Tp[], _Dp> && __u) [inline], [noexcept]`

Move assignment operator.

Parameters

<code>__u</code>	The object to transfer ownership from.
------------------	--

Invokes the deleter first if this object owns a pointer.

Definition at line 497 of file unique_ptr.h.

References std::get_deleter().

```
4.705.3.6 template<typename _Tp, typename _Dp > template<typename _Up, typename _Ep > enable_if<__safe_↔
conversion<_Up, _Ep>::value, unique_ptr<>::type std::unique_ptr< _Tp[], _Dp >::operator= ( unique_ptr<
_Up, _Ep > && __u ) [inline], [noexcept]
```

Assignment from another type.

Parameters

<code>↔</code> <code>_u</code>	The object to transfer ownership from, which owns a convertible pointer to an array object.
-----------------------------------	---

Invokes the deleter first if this object owns a pointer.

Definition at line 514 of file unique_ptr.h.

References std::get_deleter().

```
4.705.3.7 template<typename _Tp, typename _Dp > unique_ptr& std::unique_ptr< _Tp[], _Dp >::operator= ( nullptr_t )
[inline], [noexcept]
```

Reset the unique_ptr to empty, invoking the deleter if necessary.

Definition at line 523 of file unique_ptr.h.

```
4.705.3.8 template<typename _Tp, typename _Dp > std::add_lvalue_reference<element_type>::type std::unique_ptr< _Tp[],
_Dp >::operator[]( size_t __i ) const [inline]
```

Access an element of owned array.

Definition at line 533 of file unique_ptr.h.

```
4.705.3.9 template<typename _Tp, typename _Dp > pointer std::unique_ptr< _Tp[], _Dp >::release ( ) [inline],
[noexcept]
```

Release ownership of any stored pointer.

Definition at line 562 of file unique_ptr.h.

```
4.705.3.10 template<typename _Tp, typename _Dp > void std::unique_ptr< _Tp[], _Dp >::reset ( pointer __p = pointer ( )
) [inline], [noexcept]
```

Replace the stored pointer.

Parameters

<code>_↔</code>	The new pointer to store.
<code>_p</code>	

The deleter will be invoked if a pointer is already owned.

Definition at line 576 of file `unique_ptr.h`.

References `std::get_deleter()`, and `std::swap()`.

4.705.3.11 `template<typename _Tp, typename _Dp> void std::unique_ptr<_Tp[], _Dp>::swap(unique_ptr<_Tp[], _Dp> &_u) [inline], [noexcept]`

Exchange the pointer and deleter with another object.

Definition at line 591 of file `unique_ptr.h`.

References `std::unique_ptr<_Tp, _Dp>::get()`, `std::operator>()`, `std::operator>=()`, `std::unique_ptr<_Tp, _Dp> <↔ ::swap()`, and `std::swap()`.

The documentation for this class was generated from the following file:

- [unique_ptr.h](#)

4.706 `std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>` Class Template Reference

Public Types

- `typedef _Hashtable::key_type` [key_type](#)
- `typedef _Hashtable::value_type` [value_type](#)
- `typedef _Hashtable::mapped_type` [mapped_type](#)
- `typedef _Hashtable::hasher` [hasher](#)
- `typedef _Hashtable::key_equal` [key_equal](#)
- `typedef _Hashtable::allocator_type` [allocator_type](#)
- `typedef _Hashtable::pointer` [pointer](#)
- `typedef _Hashtable::const_pointer` [const_pointer](#)
- `typedef _Hashtable::reference` [reference](#)
- `typedef _Hashtable::const_reference` [const_reference](#)
- `typedef _Hashtable::iterator` [iterator](#)
- `typedef _Hashtable::const_iterator` [const_iterator](#)
- `typedef _Hashtable::local_iterator` [local_iterator](#)
- `typedef _Hashtable::const_local_iterator` [const_local_iterator](#)
- `typedef _Hashtable::size_type` [size_type](#)
- `typedef _Hashtable::difference_type` [difference_type](#)

Public Member Functions

- [unordered_map](#) ([size_type](#) __n=10, const [hasher](#) &__hf=[hasher](#)(), const [key_equal](#) &__eq=[key_equal](#)(), const [allocator_type](#) &__a=[allocator_type](#)())
- [template](#)<[typename](#) _InputIterator >
[unordered_map](#) (_InputIterator __f, _InputIterator __l, [size_type](#) __n=0, const [hasher](#) &__hf=[hasher](#)(), const [key_equal](#) &__eq=[key_equal](#)(), const [allocator_type](#) &__a=[allocator_type](#)())
- [unordered_map](#) (const [unordered_map](#) &)=default
- [unordered_map](#) ([unordered_map](#) &&)=default
- [unordered_map](#) (const [allocator_type](#) &__a)
- [unordered_map](#) (const [unordered_map](#) &__umap, const [allocator_type](#) &__a)
- [unordered_map](#) ([unordered_map](#) &&__umap, const [allocator_type](#) &__a)
- [unordered_map](#) (initializer_list< [value_type](#) > __l, [size_type](#) __n=0, const [hasher](#) &__hf=[hasher](#)(), const [key_equal](#) &__eq=[key_equal](#)(), const [allocator_type](#) &__a=[allocator_type](#)())
- [iterator](#) [begin](#) () noexcept
- [local_iterator](#) [begin](#) ([size_type](#) __n)
- [size_type](#) [bucket](#) (const [key_type](#) &__key) const
- [size_type](#) [bucket_count](#) () const noexcept
- [size_type](#) [bucket_size](#) ([size_type](#) __n) const
- void [clear](#) () noexcept
- [size_type](#) [count](#) (const [key_type](#) &__x) const
- [template](#)<[typename](#)... _Args>
[std::pair](#)< [iterator](#), bool > [emplace](#) (_Args &&...__args)
- [template](#)<[typename](#)... _Args>
[iterator](#) [emplace_hint](#) (const [iterator](#) __pos, _Args &&...__args)
- bool [empty](#) () const noexcept
- [iterator](#) [end](#) () noexcept
- [local_iterator](#) [end](#) ([size_type](#) __n)
- [size_type](#) [erase](#) (const [key_type](#) &__x)
- [iterator](#) [erase](#) (const [iterator](#) __first, const [iterator](#) __last)
- [allocator_type](#) [get_allocator](#) () const noexcept
- [hasher](#) [hash_function](#) () const
- [template](#)<[typename](#) _InputIterator >
void [insert](#) (_InputIterator __first, _InputIterator __last)
- void [insert](#) (initializer_list< [value_type](#) > __l)
- [key_equal](#) [key_eq](#) () const
- float [load_factor](#) () const noexcept
- [size_type](#) [max_bucket_count](#) () const noexcept
- float [max_load_factor](#) () const noexcept
- void [max_load_factor](#) (float __z)
- [size_type](#) [max_size](#) () const noexcept
- [unordered_map](#) & [operator=](#) (const [unordered_map](#) &)=default
- [unordered_map](#) & [operator=](#) ([unordered_map](#) &&)=default
- [unordered_map](#) & [operator=](#) (initializer_list< [value_type](#) > __l)
- void [rehash](#) ([size_type](#) __n)
- void [reserve](#) ([size_type](#) __n)
- [size_type](#) [size](#) () const noexcept
- void [swap](#) ([unordered_map](#) &__x) noexcept(noexcept(__M_h.swap(__x._M_h)))

- [const_iterator](#) [begin](#) () const noexcept
- [const_iterator](#) [cbegin](#) () const noexcept

- `const_iterator end ()` `const noexcept`
- `const_iterator cend ()` `const noexcept`
- `std::pair< iterator, bool > insert (const value_type &__x)`
- `template<typename _Pair, typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type>
std::pair< iterator, bool > insert (_Pair &&__x)`
- `iterator insert (const_iterator __hint, const value_type &__x)`
- `template<typename _Pair, typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type>
iterator insert (const_iterator __hint, _Pair &&__x)`
- `iterator erase (const_iterator __position)`
- `iterator erase (iterator __it)`
- `iterator find (const key_type &__x)`
- `const_iterator find (const key_type &__x)` `const`
- `std::pair< iterator, iterator > equal_range (const key_type &__x)`
- `std::pair< const_iterator, const_iterator > equal_range (const key_type &__x)` `const`
- `mapped_type & operator[] (const key_type &__k)`
- `mapped_type & operator[] (key_type &&__k)`
- `mapped_type & at (const key_type &__k)`
- `const mapped_type & at (const key_type &__k)` `const`
- `const_local_iterator begin (size_type __n)` `const`
- `const_local_iterator cbegin (size_type __n)` `const`
- `const_local_iterator end (size_type __n)` `const`
- `const_local_iterator cend (size_type __n)` `const`

Friends

- `template<typename _Key1, typename _Tp1, typename _Hash1, typename _Pred1, typename _Alloc1 >
bool operator== (const unordered_map< _Key1, _Tp1, _Hash1, _Pred1, _Alloc1 > &, const unordered_map<
_Key1, _Tp1, _Hash1, _Pred1, _Alloc1 > &)`

4.706.1 Detailed Description

```
template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::
::allocator<std::pair<const _Key, _Tp> >>
class std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >
```

A standard container composed of unique keys (containing at most one of each key value) that associates values of another type with the keys.

Template Parameters

<code>_Key</code>	Type of key objects.
<code>_Tp</code>	Type of mapped objects.
<code>_Hash</code>	Hashing function object type, defaults to <code>hash<_Value></code> .
<code>_Pred</code>	Predicate function object type, defaults to <code>equal_to<_Value></code> .
<code>_Alloc</code>	Allocator type, defaults to <code>std::allocator<std::pair<const _Key, _Tp>></code> .

Meets the requirements of a [container](#), and [unordered associative container](#)

The resulting value type of the container is `std::pair<const _Key, _Tp>`.

Base is `_Hashtable`, dispatched at compile time via template alias `__umap_hashtable`.

Definition at line 98 of file `unordered_map.h`.

4.706.2 Member Typedef Documentation

4.706.2.1 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> typedef _Hashtable::allocator_type std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::allocator_type`

Public typedefs.

Definition at line 112 of file `unordered_map.h`.

4.706.2.2 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> typedef _Hashtable::const_iterator std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::const_iterator`

Iterator-related typedefs.

Definition at line 122 of file `unordered_map.h`.

4.706.2.3 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> typedef _Hashtable::const_local_iterator std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::const_local_iterator`

Iterator-related typedefs.

Definition at line 124 of file `unordered_map.h`.

4.706.2.4 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> typedef _Hashtable::const_pointer std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::const_pointer`

Iterator-related typedefs.

Definition at line 118 of file `unordered_map.h`.

4.706.2.5 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> >> typedef _Hashtable::const_reference std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::const_reference`

Iterator-related typedefs.

Definition at line 120 of file unordered_map.h.

4.706.2.6 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> >> typedef _Hashtable::difference_type std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::difference_type`

Iterator-related typedefs.

Definition at line 126 of file unordered_map.h.

4.706.2.7 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> >> typedef _Hashtable::hasher std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::hasher`

Public typedefs.

Definition at line 110 of file unordered_map.h.

4.706.2.8 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> >> typedef _Hashtable::iterator std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::iterator`

Iterator-related typedefs.

Definition at line 121 of file unordered_map.h.

4.706.2.9 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> >> typedef _Hashtable::key_equal std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::key_equal`

Public typedefs.

Definition at line 111 of file unordered_map.h.

4.706.2.10 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> >> typedef _Hashtable::key_type std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::key_type`

Public typedefs.

Definition at line 107 of file unordered_map.h.

4.706.2.11 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> typedef _Hashtable::local_iterator std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::local_iterator`

Iterator-related typedefs.

Definition at line 123 of file unordered_map.h.

4.706.2.12 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> typedef _Hashtable::mapped_type std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::mapped_type`

Public typedefs.

Definition at line 109 of file unordered_map.h.

4.706.2.13 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> typedef _Hashtable::pointer std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::pointer`

Iterator-related typedefs.

Definition at line 117 of file unordered_map.h.

4.706.2.14 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> typedef _Hashtable::reference std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::reference`

Iterator-related typedefs.

Definition at line 119 of file unordered_map.h.

4.706.2.15 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> typedef _Hashtable::size_type std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::size_type`

Iterator-related typedefs.

Definition at line 125 of file unordered_map.h.

4.706.2.16 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> typedef _Hashtable::value_type std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::value_type`

Public typedefs.

Definition at line 108 of file unordered_map.h.

4.706.3 Constructor & Destructor Documentation

4.706.3.1 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::unordered_map (size_type __n = 10, const hasher & __hf = hasher(), const key_equal & __eq = key_equal(), const allocator_type & __a = allocator_type()) [inline], [explicit]`

Default constructor creates no elements.

Parameters

<code>__n</code>	Initial number of buckets.
<code>__hf</code>	A hash functor.
<code>__eqf</code>	A key equality functor.
<code>__a</code>	An allocator object.

Definition at line 139 of file `unordered_map.h`.

Referenced by `std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::unordered_map()`.

```
4.706.3.2 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc
= std::allocator<std::pair<const _Key, _Tp>>> template<typename _InputIterator> std::unordered_map<
_Key, _Tp, _Hash, _Pred, _Alloc>::unordered_map ( _InputIterator __f, _InputIterator __l, size_type __n = 0,
const hasher & __hf = hasher(), const key_equal & __eqf = key_equal(), const allocator_type & __a =
allocator_type() ) [inline]
```

Builds an `unordered_map` from a range.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__n</code>	Minimal initial number of buckets.
<code>__hf</code>	A hash functor.
<code>__eqf</code>	A key equality functor.
<code>__a</code>	An allocator object.

Create an `unordered_map` consisting of copies of the elements from `[__first, __last)`. This is linear in `N` (where `N` is `distance(__first, __last)`).

Definition at line 160 of file `unordered_map.h`.

References `std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::unordered_map()`.

```
4.706.3.3 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc
= std::allocator<std::pair<const _Key, _Tp>>> std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc
>::unordered_map ( const unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc> & ) [default]
```

Copy constructor.

```
4.706.3.4 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc
= std::allocator<std::pair<const _Key, _Tp>>> std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc
>::unordered_map ( unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc> && ) [default]
```

Move constructor.

```
4.706.3.5 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc
= std::allocator<std::pair<const _Key, _Tp>>> std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc
>::unordered_map ( const allocator_type & __a ) [inline], [explicit]
```

Creates an `unordered_map` with no elements.

Parameters

\longleftrightarrow _a	An allocator object.
-----------------------------	----------------------

Definition at line 179 of file unordered_map.h.

References std::move(), and std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::unordered_map().

4.706.3.6 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::unordered_map (initializer_list< value_type > __l, size_type __n = 0, const hasher & __hf = hasher(), const key_equal & __eqf = key_equal(), const allocator_type & __a = allocator_type()) [inline]

Builds an unordered_map from an initializer_list.

Parameters

__l	An initializer_list.
__n	Minimal initial number of buckets.
__hf	A hash functor.
__eqf	A key equality functor.
__a	An allocator object.

Create an unordered_map consisting of copies of the elements in the list. This is linear in N (where N is __l.size()).

Definition at line 214 of file unordered_map.h.

References std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::operator=().

4.706.4 Member Function Documentation

4.706.4.1 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> mapped_type& std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::at (const key_type & __k) [inline]

Access to unordered_map data.

Parameters

\longleftrightarrow _k	The key for which data should be retrieved.
-----------------------------	---

Returns

A reference to the data whose key is equal to __k, if such a data is present in the unordered_map.

Exceptions

<code>std::out_of_range</code>	If no such data is present.
--------------------------------	-----------------------------

Definition at line 643 of file `unordered_map.h`.

```
4.706.4.2  template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> const mapped_type& std::unordered_map<_Key, _Tp, _Hash,
_Pred, _Alloc>::at ( const key_type & __k ) const    [inline]
```

Access to `unordered_map` data.

Parameters

<code>__k</code>	The key for which data should be retrieved.
------------------	---

Returns

A reference to the data whose key is equal to `__k`, if such a data is present in the `unordered_map`.

Exceptions

<code>std::out_of_range</code>	If no such data is present.
--------------------------------	-----------------------------

Definition at line 647 of file `unordered_map.h`.

```
4.706.4.3  template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> iterator std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::begin ( ) [inline], [noexcept]
```

Returns a read/write iterator that points to the first element in the `unordered_map`.

Definition at line 278 of file `unordered_map.h`.

```
4.706.4.4  template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> const_iterator std::unordered_map<_Key, _Tp, _Hash, _Pred,
_Alloc>::begin ( ) const    [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the `unordered_map`.

Definition at line 287 of file `unordered_map.h`.

```
4.706.4.5  template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> local_iterator std::unordered_map<_Key, _Tp, _Hash, _Pred,
_Alloc>::begin ( size_type __n )    [inline]
```

Returns a read/write iterator pointing to the first bucket element.

Parameters

$_n$	The bucket index.
-------	-------------------

Returns

A read/write local iterator.

Definition at line 688 of file unordered_map.h.

```
4.706.4.6  template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> const_local_iterator std::unordered_map<_Key, _Tp, _Hash,
_Pred, _Alloc>::begin ( size_type __n ) const    [inline]
```

Returns a read-only (constant) iterator pointing to the first bucket element.

Parameters

$_n$	The bucket index.
-------	-------------------

Returns

A read-only local iterator.

Definition at line 699 of file unordered_map.h.

```
4.706.4.7  template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> size_type std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc
>::bucket_count ( ) const    [inline], [noexcept]
```

Returns the number of buckets of the unordered_map.

Definition at line 655 of file unordered_map.h.

```
4.706.4.8  template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> const_iterator std::unordered_map<_Key, _Tp, _Hash, _Pred,
_Alloc>::cbegin ( ) const    [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the unordered_map.

Definition at line 291 of file unordered_map.h.

```
4.706.4.9  template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> const_local_iterator std::unordered_map<_Key, _Tp, _Hash,
_Pred, _Alloc>::cbegin ( size_type __n ) const    [inline]
```

Returns a read-only (constant) iterator pointing to the first bucket element.

Parameters

\leftarrow _n	The bucket index.
--------------------	-------------------

Returns

A read-only local iterator.

Definition at line 703 of file unordered_map.h.

```
4.706.4.10 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> const_iterator std::unordered_map<_Key, _Tp, _Hash, _Pred,
_Alloc>::cend( ) const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the unordered_map.

Definition at line 313 of file unordered_map.h.

```
4.706.4.11 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> const_local_iterator std::unordered_map<_Key, _Tp, _Hash,
_Pred, _Alloc>::cend( size_type __n ) const [inline]
```

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

Parameters

\leftarrow _n	The bucket index.
--------------------	-------------------

Returns

A read-only local iterator.

Definition at line 729 of file unordered_map.h.

```
4.706.4.12 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> void std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc
>::clear( ) [inline], [noexcept]
```

Erases all elements in an unordered_map. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 527 of file unordered_map.h.

```
4.706.4.13 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> size_type std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc
>::count( const key_type &__x ) const [inline]
```

Finds the number of elements.

Parameters

<code>_↔</code>	Key to count.
<code>_X</code>	

Returns

Number of elements with specified key.

This function only makes sense for `unordered_multimap`; for `unordered_map` the result will either be 0 (not present) or 1 (present).

Definition at line 591 of file `unordered_map.h`.

```
4.706.4.14 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc
= std::allocator<std::pair<const _Key, _Tp> >> template<typename... _Args> std::pair<iterator, bool>
std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc >::emplace ( _Args &&... _args ) [inline]
```

Attempts to build and insert a `std::pair` into the `unordered_map`.

Parameters

<code>__args</code>	Arguments used to generate a new pair instance (see <code>std::piecewise_construct</code> for passing arguments to each part of the pair constructor).
---------------------	--

Returns

A pair, of which the first element is an iterator that points to the possibly inserted pair, and the second is a bool that is true if the pair was actually inserted.

This function attempts to build and insert a (key, value) pair into the `unordered_map`. An `unordered_map` relies on unique keys and thus a pair is only inserted if its first element (the key) is not already present in the `unordered_map`.

Insertion requires amortized constant time.

Definition at line 340 of file `unordered_map.h`.

```
4.706.4.15 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp> >> template<typename... _Args> iterator std::unordered_map<
_Key, _Tp, _Hash, _Pred, _Alloc >::emplace_hint ( const_iterator __pos, _Args &&... _args ) [inline]
```

Attempts to build and insert a `std::pair` into the `unordered_map`.

Parameters

<code>__pos</code>	An iterator that serves as a hint as to where the pair should be inserted.
<code>__args</code>	Arguments used to generate a new pair instance (see <code>std::piecewise_construct</code> for passing arguments to each part of the pair constructor).

Returns

An iterator that points to the element with key of the `std::pair` built from `__args` (may or may not be that `std::pair`).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument `emplace()` does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.html> for more on *hinting*.

Insertion requires amortized constant time.

Definition at line 370 of file `unordered_map.h`.

```
4.706.4.16  template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
            std::allocator<std::pair<const _Key, _Tp>>> bool std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc
            >::empty( ) const    [inline], [noexcept]
```

Returns true if the `unordered_map` is empty.

Definition at line 258 of file `unordered_map.h`.

```
4.706.4.17  template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
            std::allocator<std::pair<const _Key, _Tp>>> iterator std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc
            >::end( )    [inline], [noexcept]
```

Returns a read/write iterator that points one past the last element in the `unordered_map`.

Definition at line 300 of file `unordered_map.h`.

```
4.706.4.18  template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
            std::allocator<std::pair<const _Key, _Tp>>> const_iterator std::unordered_map<_Key, _Tp, _Hash, _Pred,
            _Alloc >::end( ) const    [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the `unordered_map`.

Definition at line 309 of file `unordered_map.h`.

```
4.706.4.19  template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
            std::allocator<std::pair<const _Key, _Tp>>> local_iterator std::unordered_map<_Key, _Tp, _Hash, _Pred,
            _Alloc >::end( size_type __n )    [inline]
```

Returns a read/write iterator pointing to one past the last bucket elements.

Parameters

<code>__n</code>	The bucket index.
------------------	-------------------

Returns

A read/write local iterator.

Definition at line 714 of file unordered_map.h.

```
4.706.4.20 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> const_local_iterator std::unordered_map<_Key, _Tp, _Hash,
_Pred, _Alloc>::end ( size_type __n ) const [inline]
```

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

Parameters

$_↔$	The bucket index.
$_n$	

Returns

A read-only local iterator.

Definition at line 725 of file unordered_map.h.

```
4.706.4.21 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> std::pair<iterator, iterator> std::unordered_map<_Key, _Tp,
_Hash, _Pred, _Alloc>::equal_range ( const key_type & __x ) [inline]
```

Finds a subsequence matching given key.

Parameters

$_↔$	Key to be located.
$_x$	

Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function probably only makes sense for unordered_multimap.

Definition at line 604 of file unordered_map.h.

```
4.706.4.22 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class
_Alloc = std::allocator<std::pair<const _Key, _Tp>>> std::pair<const_iterator, const_iterator>
std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::equal_range ( const key_type & __x ) const
[inline]
```

Finds a subsequence matching given key.

Parameters

<code>_↔</code>	Key to be located.
<code>_x</code>	

Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function probably only makes sense for `unordered_multimap`.

Definition at line 608 of file `unordered_map.h`.

```
4.706.4.23  template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
            std::allocator<std::pair<const _Key, _Tp>>> iterator std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc
            >::erase( const_iterator __position ) [inline]
```

Erases an element from an `unordered_map`.

Parameters

<code>__position</code>	An iterator pointing to the element to be erased.
-------------------------	---

Returns

An iterator pointing to the element immediately following `__position` prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from an `unordered_map`. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 477 of file `unordered_map.h`.

```
4.706.4.24  template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
            std::allocator<std::pair<const _Key, _Tp>>> iterator std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc
            >::erase( iterator __it ) [inline]
```

Erases an element from an `unordered_map`.

Parameters

<code>__position</code>	An iterator pointing to the element to be erased.
-------------------------	---

Returns

An iterator pointing to the element immediately following `__position` prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from an `unordered_map`. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 482 of file `unordered_map.h`.

```
4.706.4.25  template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
            std::allocator<std::pair<const _Key, _Tp>>> size_type std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc
            >::erase( const key_type & __x ) [inline]
```

Erases elements according to the provided key.

Parameters

<code>__x</code>	Key of element to be erased.
------------------	------------------------------

Returns

The number of elements erased.

This function erases all the elements located by the given key from an `unordered_map`. For an `unordered_map` the result of this function can only be 0 (not present) or 1 (present). Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 499 of file `unordered_map.h`.

```
4.706.4.26  template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
            std::allocator<std::pair<const _Key, _Tp>>> iterator std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc
            >::erase( const_iterator __first, const_iterator __last ) [inline]
```

Erases a `[__first, __last)` range of elements from an `unordered_map`.

Parameters

<code>__first</code>	Iterator pointing to the start of the range to be erased.
<code>__last</code>	Iterator pointing to the end of the range to be erased.

Returns

The iterator `__last`.

This function erases a sequence of elements from an `unordered_map`. Note that this function only erases the elements, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 517 of file `unordered_map.h`.

```
4.706.4.27 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> iterator std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc
>::find ( const key_type & __x ) [inline]
```

Tries to locate an element in an `unordered_map`.

Parameters

<code>__x</code>	Key to be located.
------------------	--------------------

Returns

Iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end (`end()`) iterator.

Definition at line 573 of file `unordered_map.h`.

```
4.706.4.28 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> const_iterator std::unordered_map<_Key, _Tp, _Hash, _Pred,
_Alloc>::find ( const key_type & __x ) const [inline]
```

Tries to locate an element in an `unordered_map`.

Parameters

<code>__x</code>	Key to be located.
------------------	--------------------

Returns

Iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end (`end()`) iterator.

Definition at line 577 of file `unordered_map.h`.

```
4.706.4.29 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> allocator_type std::unordered_map<_Key, _Tp, _Hash, _Pred,
_Alloc>::get_allocator ( ) const [inline], [noexcept]
```

Returns the allocator object with which the unordered_map was constructed.

Definition at line 251 of file unordered_map.h.

```
4.706.4.30 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> hasher std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc
>::hash_function ( ) const [inline]
```



Returns the hash functor object with which the unordered_map was constructed.

Definition at line 549 of file unordered_map.h.

```
4.706.4.31 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> std::pair<iterator, bool> std::unordered_map<_Key, _Tp,
_Hash, _Pred, _Alloc>::insert ( const value_type & __x ) [inline]
```

Attempts to insert a std::pair into the unordered_map.

Parameters

	Pair to be inserted (see std::make_pair for easy creation of pairs).
	

Returns

A pair, of which the first element is an iterator that points to the possibly inserted pair, and the second is a bool that is true if the pair was actually inserted.

This function attempts to insert a (key, value) pair into the unordered_map. An unordered_map relies on unique keys and thus a pair is only inserted if its first element (the key) is not already present in the unordered_map.

Insertion requires amortized constant time.

Definition at line 392 of file unordered_map.h.

```
4.706.4.32 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class
_Alloc = std::allocator<std::pair<const _Key, _Tp>>> template<typename _Pair, typename = typename
std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type> std::pair<iterator, bool>
std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::insert ( _Pair && __x ) [inline]
```

Attempts to insert a std::pair into the unordered_map.

Parameters

<code>__x</code>	Pair to be inserted (see <code>std::make_pair</code> for easy creation of pairs).
------------------	---

Returns

A pair, of which the first element is an iterator that points to the possibly inserted pair, and the second is a bool that is true if the pair was actually inserted.

This function attempts to insert a (key, value) pair into the `unordered_map`. An `unordered_map` relies on unique keys and thus a pair is only inserted if its first element (the key) is not already present in the `unordered_map`.

Insertion requires amortized constant time.

Definition at line 399 of file `unordered_map.h`.

```
4.706.4.33 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> iterator std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc
>::insert ( const_iterator __hint, const value_type & __x ) [inline]
```

Attempts to insert a `std::pair` into the `unordered_map`.

Parameters

<code>__hint</code>	An iterator that serves as a hint as to where the pair should be inserted.
<code>__x</code>	Pair to be inserted (see <code>std::make_pair</code> for easy creation of pairs).

Returns

An iterator that points to the element with key of `__x` (may or may not be the pair passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument `insert()` does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.html> for more on *hinting*.

Insertion requires amortized constant time.

Definition at line 426 of file `unordered_map.h`.

```
4.706.4.34 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class
_Alloc = std::allocator<std::pair<const _Key, _Tp>>> template<typename _Pair, typename = typename
std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type> iterator std::unordered_map<_Key,
_Tp, _Hash, _Pred, _Alloc>::insert ( const_iterator __hint, _Pair && __x ) [inline]
```

Attempts to insert a `std::pair` into the `unordered_map`.

Parameters

<code>__hint</code>	An iterator that serves as a hint as to where the pair should be inserted.
<code>__x</code>	Pair to be inserted (see <code>std::make_pair</code> for easy creation of pairs).

Returns

An iterator that points to the element with key of `__x` (may or may not be the pair passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument `insert()` does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.html> for more on *hinting*.

Insertion requires amortized constant time.

Definition at line 433 of file `unordered_map.h`.

```
4.706.4.35  template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
            std::allocator<std::pair<const _Key, _Tp> >> template<typename _InputIterator > void std::unordered_map<
            _Key, _Tp, _Hash, _Pred, _Alloc >::insert ( _InputIterator __first, _InputIterator __last ) [inline]
```

A template function that attempts to insert a range of elements.

Parameters

<code>__first</code>	Iterator pointing to the start of the range to be inserted.
<code>__last</code>	Iterator pointing to the end of the range.

Complexity similar to that of the range constructor.

Definition at line 448 of file `unordered_map.h`.

```
4.706.4.36  template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
            std::allocator<std::pair<const _Key, _Tp> >> void std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc
            >::insert ( initializer_list<value_type> __l ) [inline]
```

Attempts to insert a list of elements into the `unordered_map`.

Parameters

<code>__l</code>	A <code>std::initializer_list<value_type></code> of elements to be inserted.
------------------	--

Complexity similar to that of the range constructor.

Definition at line 459 of file unordered_map.h.

```
4.706.4.37 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> key_equal std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc
>::key_eq( ) const [inline]
```

Returns the key comparison object with which the unordered_map was constructed.

Definition at line 555 of file unordered_map.h.

```
4.706.4.38 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> float std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc
>::load_factor( ) const [inline], [noexcept]
```

Returns the average number of elements per bucket.

Definition at line 737 of file unordered_map.h.

```
4.706.4.39 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> size_type std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc
>::max_bucket_count( ) const [inline], [noexcept]
```

Returns the maximum number of buckets of the unordered_map.

Definition at line 660 of file unordered_map.h.

```
4.706.4.40 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> float std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc
>::max_load_factor( ) const [inline], [noexcept]
```


Returns a positive number that the unordered_map tries to keep the load factor less than or equal to.

Definition at line 743 of file unordered_map.h.

```
4.706.4.41 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> void std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc
>::max_load_factor( float __z ) [inline]
```

Change the unordered_map maximum load factor.

Parameters

	The new maximum load factor.
<code>__z</code>	

Definition at line 751 of file unordered_map.h.


```
4.706.4.42 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> size_type std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc
>::max_size ( ) const [inline], [noexcept]
```

Returns the maximum size of the unordered_map.

Definition at line 268 of file unordered_map.h.

```
4.706.4.43 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> unordered_map& std::unordered_map<_Key, _Tp, _Hash,
_Pred, _Alloc>::operator= ( const unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc> & ) [default]
```

Copy assignment operator.

Referenced by std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::unordered_map(), and std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>::unordered_multimap().

```
4.706.4.44 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> unordered_map& std::unordered_map<_Key, _Tp, _Hash,
_Pred, _Alloc>::operator= ( unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc> && ) [default]
```

Move assignment operator.

```
4.706.4.45 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> unordered_map& std::unordered_map<_Key, _Tp, _Hash,
_Pred, _Alloc>::operator= ( initializer_list<value_type> __l ) [inline]
```

Unordered_map list assignment operator.

Parameters

↵	An initializer_list.
↵	
↵	
↵	
l	

This function fills an unordered_map with copies of the elements in the initializer list __l.

Note that the assignment completely changes the unordered_map and that the resulting unordered_map's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 242 of file unordered_map.h.

```
4.706.4.46 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> mapped_type& std::unordered_map<_Key, _Tp, _Hash, _Pred,
_Alloc>::operator[] ( const key_type & __k ) [inline]
```

Subscript ([]) access to unordered_map data.

Parameters

\leftrightarrow _k	The key for which data should be retrieved.
-------------------------	---

Returns

A reference to the data of the (key,data) pair.

Allows for easy lookup with the subscript ([]) operator. Returns data associated with the key specified in subscript. If the key does not exist, a pair with that key is created using default values, which is then returned.

Lookup requires constant time.

Definition at line 626 of file unordered_map.h.

```
4.706.4.47  template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
            std::allocator<std::pair<const _Key, _Tp>>> mapped_type& std::unordered_map<_Key, _Tp, _Hash, _Pred,
            _Alloc>::operator[]( key_type &&_k ) [inline]
```

Subscript ([]) access to unordered_map data.

Parameters

\leftrightarrow _k	The key for which data should be retrieved.
-------------------------	---

Returns

A reference to the data of the (key,data) pair.

Allows for easy lookup with the subscript ([]) operator. Returns data associated with the key specified in subscript. If the key does not exist, a pair with that key is created using default values, which is then returned.

Lookup requires constant time.

Definition at line 630 of file unordered_map.h.

References std::move().

```
4.706.4.48  template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
            std::allocator<std::pair<const _Key, _Tp>>> void std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc
            >::rehash( size_type __n ) [inline]
```

May rehash the unordered_map.

Parameters

<code>_↔</code>	The new number of buckets.
<code>_n</code>	

Rehash will occur only if the new number of buckets respect the unordered_map maximum load factor.

Definition at line 762 of file unordered_map.h.

```
4.706.4.49 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> void std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc
>::reserve ( size_type __n ) [inline]
```

Prepare the unordered_map for a specified number of elements.

Parameters

<code>_↔</code>	Number of elements required.
<code>_n</code>	

Same as rehash(ceil(n / max_load_factor())).

Definition at line 773 of file unordered_map.h.

```
4.706.4.50 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> size_type std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc
>::size ( ) const [inline], [noexcept]
```

Returns the size of the unordered_map.

Definition at line 263 of file unordered_map.h.

```
4.706.4.51 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> void std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc
>::swap ( unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc> &__x ) [inline], [noexcept]
```

Swaps data with another unordered_map.

Parameters

<code>_↔</code>	An unordered_map of the same element and allocator types.
<code>_x</code>	

This exchanges the elements between two unordered_map in constant time. Note that the global std::swap() function is specialized such that std::swap(m1,m2) will feed to this function.

Definition at line 540 of file unordered_map.h.

Referenced by `std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc >::reserve()`.

The documentation for this class was generated from the following file:

- [unordered_map.h](#)

4.707 `std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc >` Class Template Reference

Public Types

- `typedef _Hashtable::key_type` [key_type](#)
- `typedef _Hashtable::value_type` [value_type](#)
- `typedef _Hashtable::mapped_type` [mapped_type](#)
- `typedef _Hashtable::hasher` [hasher](#)
- `typedef _Hashtable::key_equal` [key_equal](#)
- `typedef _Hashtable::allocator_type` [allocator_type](#)
- `typedef _Hashtable::pointer` [pointer](#)
- `typedef _Hashtable::const_pointer` [const_pointer](#)
- `typedef _Hashtable::reference` [reference](#)
- `typedef _Hashtable::const_reference` [const_reference](#)
- `typedef _Hashtable::iterator` [iterator](#)
- `typedef _Hashtable::const_iterator` [const_iterator](#)
- `typedef _Hashtable::local_iterator` [local_iterator](#)
- `typedef _Hashtable::const_local_iterator` [const_local_iterator](#)
- `typedef _Hashtable::size_type` [size_type](#)
- `typedef _Hashtable::difference_type` [difference_type](#)

Public Member Functions

- `unordered_multimap` ([size_type](#) __n=10, const [hasher](#) &__hf=[hasher](#)(), const [key_equal](#) &__eq=[key_equal](#)(), const [allocator_type](#) &__a=[allocator_type](#)())
- `template<typename _InputIterator >`
`unordered_multimap` (_InputIterator __f, _InputIterator __l, [size_type](#) __n=0, const [hasher](#) &__hf=[hasher](#)(), const [key_equal](#) &__eq=[key_equal](#)(), const [allocator_type](#) &__a=[allocator_type](#)())
- `unordered_multimap` (const `unordered_multimap` &)=default
- `unordered_multimap` (`unordered_multimap` &&)=default
- `unordered_multimap` (const [allocator_type](#) &__a)
- `unordered_multimap` (const `unordered_multimap` &__ummap, const [allocator_type](#) &__a)
- `unordered_multimap` (`unordered_multimap` &&__ummap, const [allocator_type](#) &__a)
- `unordered_multimap` (initializer_list< [value_type](#) > __l, [size_type](#) __n=0, const [hasher](#) &__hf=[hasher](#)(), const [key_equal](#) &__eq=[key_equal](#)(), const [allocator_type](#) &__a=[allocator_type](#)())
- `iterator begin` () noexcept
- `local_iterator begin` ([size_type](#) __n)
- [size_type bucket](#) (const [key_type](#) &__key) const
- [size_type bucket_count](#) () const noexcept
- [size_type bucket_size](#) ([size_type](#) __n) const
- `void clear` () noexcept

- [size_type count](#) (const [key_type](#) &__x) const
 - [template<typename... _Args>](#)
[iterator emplace](#) (_Args &&...__args)
 - [template<typename... _Args>](#)
[iterator emplace_hint](#) (const [iterator](#) __pos, _Args &&...__args)
 - [bool empty](#) () const noexcept
 - [iterator end](#) () noexcept
 - [local_iterator end](#) (size_type __n)
 - [size_type erase](#) (const [key_type](#) &__x)
 - [iterator erase](#) (const [iterator](#) __first, const [iterator](#) __last)
 - [allocator_type get_allocator](#) () const noexcept
 - [hasher hash_function](#) () const
 - [template<typename _InputIterator>](#)
void [insert](#) (_InputIterator __first, _InputIterator __last)
 - void [insert](#) (initializer_list< [value_type](#) > __l)
 - [key_equal key_eq](#) () const
 - [float load_factor](#) () const noexcept
 - [size_type max_bucket_count](#) () const noexcept
 - [float max_load_factor](#) () const noexcept
 - void [max_load_factor](#) (float __z)
 - [size_type max_size](#) () const noexcept
 - [unordered_multimap & operator=](#) (const [unordered_multimap](#) &)=default
 - [unordered_multimap & operator=](#) ([unordered_multimap](#) &&)=default
 - [unordered_multimap & operator=](#) (initializer_list< [value_type](#) > __l)
 - void [rehash](#) (size_type __n)
 - void [reserve](#) (size_type __n)
 - [size_type size](#) () const noexcept
 - void [swap](#) ([unordered_multimap](#) &__x) noexcept(noexcept(_M_h.swap(__x._M_h)))
-
- [const_iterator begin](#) () const noexcept
 - [const_iterator cbegin](#) () const noexcept
-
- [const_iterator end](#) () const noexcept
 - [const_iterator cend](#) () const noexcept
-
- [iterator insert](#) (const [value_type](#) &__x)
 - [template<typename _Pair , typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type>](#)
[iterator insert](#) (_Pair &&__x)
-
- [iterator insert](#) (const [iterator](#) __hint, const [value_type](#) &__x)
 - [template<typename _Pair , typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type>](#)
[iterator insert](#) (const [iterator](#) __hint, _Pair &&__x)
-
- [iterator erase](#) (const [iterator](#) __position)
 - [iterator erase](#) ([iterator](#) __it)
-
- [iterator find](#) (const [key_type](#) &__x)
 - [const_iterator find](#) (const [key_type](#) &__x) const

- `std::pair< iterator, iterator > equal_range` (const `key_type` &__x)
- `std::pair< const_iterator, const_iterator > equal_range` (const `key_type` &__x) const
- `const_local_iterator begin` (size_type __n) const
- `const_local_iterator cbegin` (size_type __n) const
- `const_local_iterator end` (size_type __n) const
- `const_local_iterator cend` (size_type __n) const

Friends

- `template<typename _Key1, typename _Tp1, typename _Hash1, typename _Pred1, typename _Alloc1 > bool operator==` (const `unordered_multimap< _Key1, _Tp1, _Hash1, _Pred1, _Alloc1 >` &, const `unordered_multimap< _Key1, _Tp1, _Hash1, _Pred1, _Alloc1 >` &)

4.707.1 Detailed Description

```
template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >>
class std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >
```

A standard container composed of equivalent keys (possibly containing multiple of each key value) that associates values of another type with the keys.

Template Parameters

<code>_Key</code>	Type of key objects.
<code>_Tp</code>	Type of mapped objects.
<code>_Hash</code>	Hashing function object type, defaults to <code>hash<_Value></code> .
<code>_Pred</code>	Predicate function object type, defaults to <code>equal_to<_Value></code> .
<code>_Alloc</code>	Allocator type, defaults to <code>std::allocator<std::pair<const _Key, _Tp>></code> .

Meets the requirements of a `container`, and `unordered associative container`

The resulting value type of the container is `std::pair<const _Key, _Tp>`.

Base is `_Hashtable`, dispatched at compile time via template alias `__ummap_hashtable`.

Definition at line 810 of file `unordered_map.h`.

4.707.2 Member Typedef Documentation

4.707.2.1 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> typedef _Hashtable::allocator_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::allocator_type`

Public typedefs.

Definition at line 824 of file `unordered_map.h`.

4.707.2.2 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> typedef _Hashtable::const_iterator std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>::const_iterator`

Iterator-related typedefs.

Definition at line 834 of file unordered_map.h.

4.707.2.3 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> typedef _Hashtable::const_local_iterator std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>::const_local_iterator`

Iterator-related typedefs.

Definition at line 836 of file unordered_map.h.

4.707.2.4 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> typedef _Hashtable::const_pointer std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>::const_pointer`

Iterator-related typedefs.

Definition at line 830 of file unordered_map.h.

4.707.2.5 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> typedef _Hashtable::const_reference std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>::const_reference`

Iterator-related typedefs.

Definition at line 832 of file unordered_map.h.

4.707.2.6 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> typedef _Hashtable::difference_type std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>::difference_type`

Iterator-related typedefs.

Definition at line 838 of file unordered_map.h.

4.707.2.7 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> typedef _Hashtable::hasher std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>::hasher`

Public typedefs.

Definition at line 822 of file unordered_map.h.

4.707.2.8 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> typedef _Hashtable::iterator std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>::iterator`

Iterator-related typedefs.

Definition at line 833 of file `unordered_map.h`.

4.707.2.9 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> typedef _Hashtable::key_equal std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>::key_equal`

Public typedefs.

Definition at line 823 of file `unordered_map.h`.

4.707.2.10 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> typedef _Hashtable::key_type std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>::key_type`

Public typedefs.

Definition at line 819 of file `unordered_map.h`.

4.707.2.11 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> typedef _Hashtable::local_iterator std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>::local_iterator`

Iterator-related typedefs.

Definition at line 835 of file `unordered_map.h`.

4.707.2.12 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> typedef _Hashtable::mapped_type std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>::mapped_type`

Public typedefs.

Definition at line 821 of file `unordered_map.h`.

4.707.2.13 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> typedef _Hashtable::pointer std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>::pointer`

Iterator-related typedefs.

Definition at line 829 of file `unordered_map.h`.

4.707.2.14 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> typedef _Hashtable::reference std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::reference`

Iterator-related typedefs.

Definition at line 831 of file unordered_map.h.

4.707.2.15 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> typedef _Hashtable::size_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::size_type`

Iterator-related typedefs.

Definition at line 837 of file unordered_map.h.

4.707.2.16 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> typedef _Hashtable::value_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::value_type`

Public typedefs.

Definition at line 820 of file unordered_map.h.

4.707.3 Constructor & Destructor Documentation

4.707.3.1 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::unordered_multimap (size_type __n = 10, const hasher & __hf = hasher(), const key_equal & __eqf = key_equal(), const allocator_type & __a = allocator_type()) [inline], [explicit]`

Default constructor creates no elements.

Parameters

<code>__n</code>	Initial number of buckets.
<code>__hf</code>	A hash functor.
<code>__eqf</code>	A key equality functor.
<code>__a</code>	An allocator object.

Definition at line 851 of file unordered_map.h.

4.707.3.2 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> template<typename _InputIterator> std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::unordered_multimap (_InputIterator __f, _InputIterator __l, size_type __n = 0, const hasher & __hf = hasher(), const key_equal & __eqf = key_equal(), const allocator_type & __a = allocator_type()) [inline]`

Builds an unordered_multimap from a range.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__n</code>	Minimal initial number of buckets.
<code>__hf</code>	A hash functor.
<code>__eqf</code>	A key equality functor.
<code>__a</code>	An allocator object.

Create an `unordered_multimap` consisting of copies of the elements from `[__first,__last)`. This is linear in `N` (where `N` is `distance(__first,__last)`).

Definition at line 872 of file `unordered_map.h`.

```
4.707.3.3  template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc
           = std::allocator<std::pair<const _Key, _Tp>>> std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc
           >::unordered_multimap ( const unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc> & ) [default]
```

Copy constructor.

```
4.707.3.4  template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc
           = std::allocator<std::pair<const _Key, _Tp>>> std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc
           >::unordered_multimap ( unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc> && ) [default]
```

Move constructor.

```
4.707.3.5  template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc
           = std::allocator<std::pair<const _Key, _Tp>>> std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc
           >::unordered_multimap ( const allocator_type & __a ) [inline],[explicit]
```

Creates an `unordered_multimap` with no elements.

Parameters

<code>__a</code>	An allocator object.
------------------	----------------------

Definition at line 891 of file `unordered_map.h`.

References `std::move()`.

```
4.707.3.6  template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc
           = std::allocator<std::pair<const _Key, _Tp>>> std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc
           >::unordered_multimap ( initializer_list<value_type> __l, size_type __n = 0, const hasher & __hf
           = hasher(), const key_equal & __eqf = key_equal(), const allocator_type & __a = allocator_type() )
           [inline]
```

Builds an `unordered_multimap` from an `initializer_list`.

Parameters

<code>__l</code>	An initializer_list.
<code>__n</code>	Minimal initial number of buckets.
<code>__hf</code>	A hash functor.
<code>__eqf</code>	A key equality functor.
<code>__a</code>	An allocator object.

Create an unordered_multimap consisting of copies of the elements in the list. This is linear in N (where N is `__l.size()`).

Definition at line 926 of file unordered_map.h.

References `std::unordered_map<_Key,_Tp,_Hash,_Pred,_Alloc>::operator=()`.

4.707.4 Member Function Documentation

4.707.4.1 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> iterator std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>::begin () [inline], [noexcept]`

Returns a read/write iterator that points to the first element in the unordered_multimap.

Definition at line 990 of file unordered_map.h.

4.707.4.2 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> const_iterator std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>::begin () const [inline], [noexcept]`

Returns a read-only (constant) iterator that points to the first element in the unordered_multimap.

Definition at line 999 of file unordered_map.h.

4.707.4.3 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> local_iterator std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>::begin (size_type __n) [inline]`

Returns a read/write iterator pointing to the first bucket element.

Parameters

<code>__↔</code>	The bucket index.
<code>__n</code>	

Returns

A read/write local iterator.

Definition at line 1339 of file unordered_map.h.

```
4.707.4.4  template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>>> const_local_iterator std::unordered_multimap<_Key, _Tp,
_Hash, _Pred, _Alloc >::begin ( size_type __n ) const    [inline]
```

Returns a read-only (constant) iterator pointing to the first bucket element.

Parameters

$_n$	The bucket index.
-------	-------------------

Returns

A read-only local iterator.

Definition at line 1350 of file unordered_map.h.

```
4.707.4.5  template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>>> size_type std::unordered_multimap<_Key, _Tp, _Hash, _Pred,
_Alloc >::bucket_count ( ) const    [inline], [noexcept]
```

Returns the number of buckets of the unordered_multimap.

Definition at line 1306 of file unordered_map.h.

```
4.707.4.6  template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>>> const_iterator std::unordered_multimap<_Key, _Tp, _Hash,
_Pred, _Alloc >::cbegin ( ) const    [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the unordered_multimap.

Definition at line 1003 of file unordered_map.h.

```
4.707.4.7  template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>>> const_local_iterator std::unordered_multimap<_Key, _Tp,
_Hash, _Pred, _Alloc >::cbegin ( size_type __n ) const    [inline]
```

Returns a read-only (constant) iterator pointing to the first bucket element.

Parameters

$_n$	The bucket index.
-------	-------------------

Returns

A read-only local iterator.

Definition at line 1354 of file unordered_map.h.

4.707.4.8 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> const_iterator std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>::cend() const [inline], [noexcept]`

Returns a read-only (constant) iterator that points one past the last element in the unordered_multimap.

Definition at line 1025 of file unordered_map.h.

4.707.4.9 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> const_local_iterator std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>::cend(size_type __n) const [inline]`

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

Parameters

<code>__n</code>	The bucket index.
------------------	-------------------

Returns

A read-only local iterator.

Definition at line 1380 of file unordered_map.h.

4.707.4.10 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> void std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>::clear() [inline], [noexcept]`

Erases all elements in an unordered_multimap. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1222 of file unordered_map.h.

4.707.4.11 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> size_type std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>::count(const key_type &__x) const [inline]`

Finds the number of elements.

Parameters

<code>__x</code>	Key to count.
------------------	---------------

Returns

Number of elements with specified key.

Definition at line 1283 of file unordered_map.h.

```
4.707.4.12 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>,
class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> template<typename... _Args> iterator
std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc >::emplace ( _Args &&... __args ) [inline]
```

Attempts to build and insert a std::pair into the unordered_multimap.

Parameters

<code>__args</code>	Arguments used to generate a new pair instance (see std::piecewise_construct for passing arguments to each part of the pair constructor).
---------------------	---

Returns

An iterator that points to the inserted pair.

This function attempts to build and insert a (key, value) pair into the unordered_multimap.

Insertion requires amortized constant time.

Definition at line 1048 of file unordered_map.h.

```
4.707.4.13 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>,
class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> template<typename... _Args> iterator
std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc >::emplace_hint ( const_iterator __pos, _Args
&&... __args ) [inline]
```

Attempts to build and insert a std::pair into the unordered_multimap.

Parameters

<code>__pos</code>	An iterator that serves as a hint as to where the pair should be inserted.
<code>__args</code>	Arguments used to generate a new pair instance (see std::piecewise_construct for passing arguments to each part of the pair constructor).

Returns

An iterator that points to the element with key of the std::pair built from `__args`.

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.html> for more on *hinting*.

Insertion requires amortized constant time.

Definition at line 1074 of file unordered_map.h.

```
4.707.4.14 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> bool std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc
>::empty( ) const [inline], [noexcept]
```

Returns true if the unordered_multimap is empty.

Definition at line 970 of file unordered_map.h.

```
4.707.4.15 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred,
_Alloc >::end( ) [inline], [noexcept]
```

Returns a read/write iterator that points one past the last element in the unordered_multimap.

Definition at line 1012 of file unordered_map.h.

```
4.707.4.16 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> const_iterator std::unordered_multimap< _Key, _Tp, _Hash,
_Pred, _Alloc >::end( ) const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the unordered_multimap.

Definition at line 1021 of file unordered_map.h.

```
4.707.4.17 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> local_iterator std::unordered_multimap< _Key, _Tp, _Hash,
_Pred, _Alloc >::end( size_type __n ) [inline]
```

Returns a read/write iterator pointing to one past the last bucket elements.

Parameters

$_n$	The bucket index.
-------	-------------------

Returns

A read/write local iterator.

Definition at line 1365 of file unordered_map.h.

```
4.707.4.18 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> const_local_iterator std::unordered_multimap< _Key, _Tp,
_Hash, _Pred, _Alloc >::end( size_type __n ) const [inline]
```

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

Parameters

<code>_↔</code>	The bucket index.
<code>_n</code>	

Returns

A read-only local iterator.

Definition at line 1376 of file `unordered_map.h`.

```
4.707.4.19 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp> >> std::pair<iterator, iterator> std::unordered_multimap<
_Key, _Tp, _Hash, _Pred, _Alloc >::equal_range ( const key_type & __x ) [inline]
```

Finds a subsequence matching given key.

Parameters

<code>_↔</code>	Key to be located.
<code>_x</code>	

Returns

Pair of iterators that possibly points to the subsequence matching given key.

Definition at line 1294 of file `unordered_map.h`.

```
4.707.4.20 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class
_Alloc = std::allocator<std::pair<const _Key, _Tp> >> std::pair<const_iterator, const_iterator>
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::equal_range ( const key_type & __x ) const
[inline]
```

Finds a subsequence matching given key.

Parameters

<code>_↔</code>	Key to be located.
<code>_x</code>	

Returns

Pair of iterators that possibly points to the subsequence matching given key.

Definition at line 1298 of file `unordered_map.h`.

4.707.4.21 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> iterator std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>::erase (const_iterator __position) [inline]`

Erases an element from an unordered_multimap.

Parameters

<code>__position</code>	An iterator pointing to the element to be erased.
-------------------------	---

Returns

An iterator pointing to the element immediately following `__position` prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from an unordered_multimap. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1172 of file unordered_map.h.

4.707.4.22 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> iterator std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>::erase (iterator __it) [inline]`

Erases an element from an unordered_multimap.

Parameters

<code>__position</code>	An iterator pointing to the element to be erased.
-------------------------	---

Returns

An iterator pointing to the element immediately following `__position` prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from an unordered_multimap. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1177 of file unordered_map.h.

4.707.4.23 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> size_type std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>::erase (const key_type & __x) [inline]`

Erases elements according to the provided key.

Parameters

<code>_↔</code>	Key of elements to be erased.
<code>_x</code>	

Returns

The number of elements erased.

This function erases all the elements located by the given key from an `unordered_multimap`. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1193 of file `unordered_map.h`.

```
4.707.4.24  template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
             std::allocator<std::pair<const _Key, _Tp>>> iterator std::unordered_multimap<_Key, _Tp, _Hash, _Pred,
             _Alloc>::erase ( const_iterator __first, const_iterator __last )  [inline]
```

Erases a [`__first`,`__last`) range of elements from an `unordered_multimap`.

Parameters

<code>__first</code>	Iterator pointing to the start of the range to be erased.
<code>__last</code>	Iterator pointing to the end of the range to be erased.

Returns

The iterator `__last`.

This function erases a sequence of elements from an `unordered_multimap`. Note that this function only erases the elements, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1212 of file `unordered_map.h`.

```
4.707.4.25  template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
             std::allocator<std::pair<const _Key, _Tp>>> iterator std::unordered_multimap<_Key, _Tp, _Hash, _Pred,
             _Alloc>::find ( const key_type & __x )  [inline]
```

Tries to locate an element in an `unordered_multimap`.

Parameters

<code>_↔</code>	Key to be located.
<code>_x</code>	

Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end (end()) iterator.

Definition at line 1269 of file unordered_map.h.

```
4.707.4.26 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> const_iterator std::unordered_multimap<_Key, _Tp, _Hash,
_Pred, _Alloc>::find ( const key_type & __x ) const [inline]
```

Tries to locate an element in an unordered_multimap.

Parameters

\leftrightarrow	Key to be located.
$_x$	

Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end (end()) iterator.

Definition at line 1273 of file unordered_map.h.

```
4.707.4.27 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> allocator_type std::unordered_multimap<_Key, _Tp, _Hash,
_Pred, _Alloc>::get_allocator ( ) const [inline], [noexcept]
```

Returns the allocator object with which the unordered_multimap was constructed.

Definition at line 963 of file unordered_map.h.

```
4.707.4.28 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> hasher std::unordered_multimap<_Key, _Tp, _Hash, _Pred,
_Alloc>::hash_function ( ) const [inline]
```

Returns the hash functor object with which the unordered_multimap was constructed.

Definition at line 1245 of file unordered_map.h.

```
4.707.4.29 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> iterator std::unordered_multimap<_Key, _Tp, _Hash, _Pred,
_Alloc>::insert ( const value_type & __x ) [inline]
```

Inserts a std::pair into the unordered_multimap.

Parameters

<code>__x</code>	Pair to be inserted (see <code>std::make_pair</code> for easy creation of pairs).
------------------	---

Returns

An iterator that points to the inserted pair.

Insertion requires amortized constant time.

Definition at line 1088 of file `unordered_map.h`.

```
4.707.4.30  template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class
              _Alloc = std::allocator<std::pair<const _Key, _Tp> >> template<typename _Pair, typename = typename
              std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type> iterator std::unordered_multimap<
              _Key, _Tp, _Hash, _Pred, _Alloc >::insert( _Pair && __x ) [inline]
```

Inserts a `std::pair` into the `unordered_multimap`.

Parameters

<code>__x</code>	Pair to be inserted (see <code>std::make_pair</code> for easy creation of pairs).
------------------	---

Returns

An iterator that points to the inserted pair.

Insertion requires amortized constant time.

Definition at line 1095 of file `unordered_map.h`.

```
4.707.4.31  template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
              std::allocator<std::pair<const _Key, _Tp> >> iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred,
              _Alloc >::insert( const_iterator __hint, const value_type & __x ) [inline]
```

Inserts a `std::pair` into the `unordered_multimap`.

Parameters

<code>__hint</code>	An iterator that serves as a hint as to where the pair should be inserted.
<code>__x</code>	Pair to be inserted (see <code>std::make_pair</code> for easy creation of pairs).

Returns

An iterator that points to the element with key of `__x` (may or may not be the pair passed in).

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.html> for more on *hinting*.

Insertion requires amortized constant time.

Definition at line 1120 of file unordered_map.h.

```
4.707.4.32 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class
        _Alloc = std::allocator<std::pair<const _Key, _Tp> >> template<typename _Pair, typename = typename
        std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type> iterator std::unordered_multimap<
        _Key, _Tp, _Hash, _Pred, _Alloc >::insert( const_iterator __hint, _Pair && __x ) [inline]
```

Inserts a std::pair into the unordered_multimap.

Parameters

<code>__hint</code>	An iterator that serves as a hint as to where the pair should be inserted.
<code>__x</code>	Pair to be inserted (see std::make_pair for easy creation of pairs).

Returns

An iterator that points to the element with key of `__x` (may or may not be the pair passed in).

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.html> for more on *hinting*.

Insertion requires amortized constant time.

Definition at line 1127 of file unordered_map.h.

```
4.707.4.33 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class
        _Alloc = std::allocator<std::pair<const _Key, _Tp> >> template<typename _InputIterator > void
        std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::insert( _InputIterator __first, _InputIterator __last )
        [inline]
```

A template function that attempts to insert a range of elements.

Parameters

<code>__first</code>	Iterator pointing to the start of the range to be inserted.
<code>__last</code>	Iterator pointing to the end of the range.

Complexity similar to that of the range constructor.

Definition at line 1142 of file unordered_map.h.

```
4.707.4.34 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> void std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc
>::insert( initializer_list<value_type> __l ) [inline]
```

Attempts to insert a list of elements into the unordered_multimap.

Parameters

↔	A std::initializer_list<value_type> of elements to be inserted.
↔	
↔	
↔	
/	

Complexity similar to that of the range constructor.

Definition at line 1154 of file unordered_map.h.

```
4.707.4.35 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> key_equal std::unordered_multimap<_Key, _Tp, _Hash, _Pred,
_Alloc>::key_eq( ) const [inline]
```

Returns the key comparison object with which the unordered_multimap was constructed.

Definition at line 1251 of file unordered_map.h.

```
4.707.4.36 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> float std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc
>::load_factor( ) const [inline], [noexcept]
```

Returns the average number of elements per bucket.

Definition at line 1388 of file unordered_map.h.

```
4.707.4.37 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> size_type std::unordered_multimap<_Key, _Tp, _Hash, _Pred,
_Alloc>::max_bucket_count( ) const [inline], [noexcept]
```

Returns the maximum number of buckets of the unordered_multimap.

Definition at line 1311 of file unordered_map.h.

```
4.707.4.38 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
std::allocator<std::pair<const _Key, _Tp>>> float std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc
>::max_load_factor( ) const [inline], [noexcept]
```

Returns a positive number that the unordered_multimap tries to keep the load factor less than or equal to.

Definition at line 1394 of file unordered_map.h.

```
4.707.4.39 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =  
std::allocator<std::pair<const _Key, _Tp>>> void std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc  
>::max_load_factor( float __z ) [inline]
```

Change the unordered_multimap maximum load factor.

Parameters

<code>_↔</code>	The new maximum load factor.
<code>_Z</code>	

Definition at line 1402 of file unordered_map.h.

```
4.707.4.40  template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
             std::allocator<std::pair<const _Key, _Tp>>> size_type std::unordered_multimap<_Key, _Tp, _Hash, _Pred,
             _Alloc>::max_size( ) const    [inline], [noexcept]
```

Returns the maximum size of the unordered_multimap.

Definition at line 980 of file unordered_map.h.

```
4.707.4.41  template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
             std::allocator<std::pair<const _Key, _Tp>>> unordered_multimap& std::unordered_multimap<_Key,
             _Tp, _Hash, _Pred, _Alloc>::operator=( const unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc> & )
             [default]
```

Copy assignment operator.

```
4.707.4.42  template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
             std::allocator<std::pair<const _Key, _Tp>>> unordered_multimap& std::unordered_multimap<_Key,
             _Tp, _Hash, _Pred, _Alloc>::operator=( unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc> && )
             [default]
```

Move assignment operator.

```
4.707.4.43  template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc =
             std::allocator<std::pair<const _Key, _Tp>>> unordered_multimap& std::unordered_multimap<_Key, _Tp,
             _Hash, _Pred, _Alloc>::operator=( initializer_list<value_type> __l )    [inline]
```

Unordered_multimap list assignment operator.

Parameters

<code>↔</code>	An initializer_list.
<code>_↔</code>	
<code>↔</code>	
<code>_↔</code>	
<code>/</code>	

This function fills an unordered_multimap with copies of the elements in the initializer list `__l`.

Note that the assignment completely changes the unordered_multimap and that the resulting unordered_multimap's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 954 of file unordered_map.h.

4.707.4.44 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> void std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>::rehash(size_type __n) [inline]`

May rehash the unordered_multimap.

Parameters

<code>__n</code>	The new number of buckets.
------------------	----------------------------

Rehash will occur only if the new number of buckets respect the unordered_multimap maximum load factor.

Definition at line 1413 of file unordered_map.h.

4.707.4.45 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> void std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>::reserve(size_type __n) [inline]`

Prepare the unordered_multimap for a specified number of elements.

Parameters

<code>__n</code>	Number of elements required.
------------------	------------------------------

Same as `rehash(ceil(n / max_load_factor()))`.

Definition at line 1424 of file unordered_map.h.

References `std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::swap()`, and `std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>::swap()`.

4.707.4.46 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> size_type std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>::size() const [inline], [noexcept]`

Returns the size of the unordered_multimap.

Definition at line 975 of file unordered_map.h.

4.707.4.47 `template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp>>> void std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>::swap(unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc> &__x) [inline], [noexcept]`

Swaps data with another unordered_multimap.

Parameters

<code>_↔</code>	An unordered_multimap of the same element and allocator types.
<code>_X</code>	

This exchanges the elements between two unordered_multimap in constant time. Note that the global `std::swap()` function is specialized such that `std::swap(m1,m2)` will feed to this function.

Definition at line 1236 of file unordered_map.h.

Referenced by `std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc >::reserve()`.

The documentation for this class was generated from the following file:

- [unordered_map.h](#)

4.708 `std::unordered_multiset<_Value, _Hash, _Pred, _Alloc >` Class Template Reference

Public Types

- `typedef _Hashtable::key_type` [key_type](#)
- `typedef _Hashtable::value_type` [value_type](#)
- `typedef _Hashtable::hasher` [hasher](#)
- `typedef _Hashtable::key_equal` [key_equal](#)
- `typedef _Hashtable::allocator_type` [allocator_type](#)
- `typedef _Hashtable::pointer` [pointer](#)
- `typedef _Hashtable::const_pointer` [const_pointer](#)
- `typedef _Hashtable::reference` [reference](#)
- `typedef _Hashtable::const_reference` [const_reference](#)
- `typedef _Hashtable::iterator` [iterator](#)
- `typedef _Hashtable::const_iterator` [const_iterator](#)
- `typedef _Hashtable::local_iterator` [local_iterator](#)
- `typedef _Hashtable::const_local_iterator` [const_local_iterator](#)
- `typedef _Hashtable::size_type` [size_type](#)
- `typedef _Hashtable::difference_type` [difference_type](#)

Public Member Functions

- `unordered_multiset` (`size_type` __n=10, const `hasher` &__hf=`hasher`(), const `key_equal` &__eq=`key_equal`(), const `allocator_type` &__a=`allocator_type`())
- `template<typename _InputIterator >`
`unordered_multiset` (`_InputIterator` __f, `_InputIterator` __l, `size_type` __n=0, const `hasher` &__hf=`hasher`(), const `key_equal` &__eq=`key_equal`(), const `allocator_type` &__a=`allocator_type`())
- `unordered_multiset` (const `unordered_multiset` &)=default
- `unordered_multiset` (`unordered_multiset` &&)=default
- `unordered_multiset` (initializer_list< `value_type` > __l, `size_type` __n=0, const `hasher` &__hf=`hasher`(), const `key_equal` &__eq=`key_equal`(), const `allocator_type` &__a=`allocator_type`())
- `unordered_multiset` (const `allocator_type` &__a)
- `unordered_multiset` (const `unordered_multiset` &__umset, const `allocator_type` &__a)
- `unordered_multiset` (`unordered_multiset` &&__umset, const `allocator_type` &__a)
- `size_type` `bucket` (const `key_type` &__key) const
- `size_type` `bucket_count` () const noexcept
- `size_type` `bucket_size` (`size_type` __n) const
- `const_iterator` `cbegin` () const noexcept
- `const_iterator` `cend` () const noexcept
- void `clear` () noexcept
- `size_type` `count` (const `key_type` &__x) const
- `template<typename... _Args>`
`iterator` `emplace` (`_Args` &&...__args)
- `template<typename... _Args>`
`iterator` `emplace_hint` (const `iterator` __pos, `_Args` &&...__args)
- bool `empty` () const noexcept
- `size_type` `erase` (const `key_type` &__x)
- `iterator` `erase` (const `iterator` __first, const `iterator` __last)
- `allocator_type` `get_allocator` () const noexcept
- `hasher` `hash_function` () const
- `template<typename _InputIterator >`
void `insert` (`_InputIterator` __first, `_InputIterator` __last)
- void `insert` (initializer_list< `value_type` > __l)
- `key_equal` `key_eq` () const
- float `load_factor` () const noexcept
- `size_type` `max_bucket_count` () const noexcept
- float `max_load_factor` () const noexcept
- void `max_load_factor` (float __z)
- `size_type` `max_size` () const noexcept
- `unordered_multiset` & `operator=` (const `unordered_multiset` &)=default
- `unordered_multiset` & `operator=` (`unordered_multiset` &&)=default
- `unordered_multiset` & `operator=` (initializer_list< `value_type` > __l)
- void `rehash` (`size_type` __n)
- void `reserve` (`size_type` __n)
- `size_type` `size` () const noexcept
- void `swap` (`unordered_multiset` &__x) noexcept(noexcept(__M_h.swap(__x._M_h)))
- `iterator` `begin` () noexcept
- `const_iterator` `begin` () const noexcept
- `iterator` `end` () noexcept

- `const_iterator end` () const noexcept
- `iterator insert` (const `value_type` &__x)
- `iterator insert` (`value_type` &&__x)
- `iterator insert` (const_iterator __hint, const `value_type` &__x)
- `iterator insert` (const_iterator __hint, `value_type` &&__x)
- `iterator erase` (const_iterator __position)
- `iterator erase` (iterator __it)
- `iterator find` (const `key_type` &__x)
- `const_iterator find` (const `key_type` &__x) const
- `std::pair< iterator, iterator > equal_range` (const `key_type` &__x)
- `std::pair< const_iterator, const_iterator > equal_range` (const `key_type` &__x) const
- `local_iterator begin` (size_type __n)
- `const_local_iterator begin` (size_type __n) const
- `const_local_iterator cbegin` (size_type __n) const
- `local_iterator end` (size_type __n)
- `const_local_iterator end` (size_type __n) const
- `const_local_iterator cend` (size_type __n) const

Friends

- `template<typename _Value1, typename _Hash1, typename _Pred1, typename _Alloc1 > bool operator==` (const `unordered_multiset< _Value1, _Hash1, _Pred1, _Alloc1 >` &, const `unordered_multiset< _Value1, _Hash1, _Pred1, _Alloc1 >` &)

4.708.1 Detailed Description

```
template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>>
class std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >
```

A standard container composed of equivalent keys (possibly containing multiple of each key value) in which the elements' keys are the elements themselves.

Template Parameters

<code>_Value</code>	Type of key objects.
<code>_Hash</code>	Hashing function object type, defaults to <code>hash<_Value></code> .
<code>_Pred</code>	Predicate function object type, defaults to <code>equal_to<_Value></code> .
<code>_Alloc</code>	Allocator type, defaults to <code>allocator<_Key></code> .

Meets the requirements of a [container](#), and [unordered associative container](#)

Base is _Hashtable, dispatched at compile time via template alias __umset_hashtable.

Definition at line 728 of file unordered_set.h.

4.708.2 Member Typedef Documentation

4.708.2.1 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> typedef _Hashtable::allocator_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::allocator_type`

Public typedefs.

Definition at line 741 of file unordered_set.h.

4.708.2.2 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> typedef _Hashtable::const_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::const_iterator`

Iterator-related typedefs.

Definition at line 751 of file unordered_set.h.

4.708.2.3 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> typedef _Hashtable::const_local_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::const_local_iterator`

Iterator-related typedefs.

Definition at line 753 of file unordered_set.h.

4.708.2.4 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> typedef _Hashtable::const_pointer std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::const_pointer`

Iterator-related typedefs.

Definition at line 747 of file unordered_set.h.

4.708.2.5 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> typedef _Hashtable::const_reference std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::const_reference`

Iterator-related typedefs.

Definition at line 749 of file unordered_set.h.

4.708.2.6 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> typedef _Hashtable::difference_type std::unordered_multiset<_Value, _Hash, _Pred, _Alloc>::difference_type`

Iterator-related typedefs.

Definition at line 755 of file unordered_set.h.

4.708.2.7 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> typedef _Hashtable::hasher std::unordered_multiset<_Value, _Hash, _Pred, _Alloc>::hasher`

Public typedefs.

Definition at line 739 of file unordered_set.h.

4.708.2.8 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> typedef _Hashtable::iterator std::unordered_multiset<_Value, _Hash, _Pred, _Alloc>::iterator`

Iterator-related typedefs.

Definition at line 750 of file unordered_set.h.

4.708.2.9 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> typedef _Hashtable::key_equal std::unordered_multiset<_Value, _Hash, _Pred, _Alloc>::key_equal`

Public typedefs.

Definition at line 740 of file unordered_set.h.

4.708.2.10 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> typedef _Hashtable::key_type std::unordered_multiset<_Value, _Hash, _Pred, _Alloc>::key_type`

Public typedefs.

Definition at line 737 of file unordered_set.h.

4.708.2.11 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> typedef _Hashtable::local_iterator std::unordered_multiset<_Value, _Hash, _Pred, _Alloc>::local_iterator`

Iterator-related typedefs.

Definition at line 752 of file unordered_set.h.

```
4.708.2.12 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> typedef _Hashtable::pointer std::unordered_multiset< _Value, _Hash, _Pred, _Alloc
>::pointer
```

Iterator-related typedefs.

Definition at line 746 of file unordered_set.h.

```
4.708.2.13 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> typedef _Hashtable::reference std::unordered_multiset< _Value, _Hash, _Pred, _Alloc
>::reference
```

Iterator-related typedefs.

Definition at line 748 of file unordered_set.h.

```
4.708.2.14 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> typedef _Hashtable::size_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc
>::size_type
```

Iterator-related typedefs.

Definition at line 754 of file unordered_set.h.

```
4.708.2.15 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> typedef _Hashtable::value_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc
>::value_type
```

Public typedefs.

Definition at line 738 of file unordered_set.h.

4.708.3 Constructor & Destructor Documentation

```
4.708.3.1 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::unordered_multiset
( size_type __n = 10, const hasher & __hf = hasher(), const key_equal & __eqf = key_equal(), const
allocator_type & __a = allocator_type() ) [inline],[explicit]
```

Default constructor creates no elements.

Parameters

<code>__n</code>	Initial number of buckets.
<code>__hf</code>	A hash functor.
<code>__eqf</code>	A key equality functor.
<code>__a</code>	An allocator object.

Definition at line 767 of file unordered_set.h.

```
4.708.3.2 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> template<typename _InputIterator > std::unordered_multiset< _Value, _Hash, _Pred,
_Alloc >::unordered_multiset ( _InputIterator __f, _InputIterator __l, size_type __n = 0, const hasher & __hf
= hasher(), const key_equal & __eqf = key_equal(), const allocator_type & __a = allocator_type() )
[inline]
```

Builds an unordered_multiset from a range.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__n</code>	Minimal initial number of buckets.
<code>__hf</code>	A hash functor.
<code>__eqf</code>	A key equality functor.
<code>__a</code>	An allocator object.

Create an unordered_multiset consisting of copies of the elements from `[__first,__last)`. This is linear in N (where N is `distance(__first,__last)`).

Definition at line 788 of file unordered_set.h.

```
4.708.3.3 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::unordered_multiset (
const unordered_multiset< _Value, _Hash, _Pred, _Alloc > & ) [default]
```

Copy constructor.

```
4.708.3.4 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::unordered_multiset (
unordered_multiset< _Value, _Hash, _Pred, _Alloc > && ) [default]
```

Move constructor.

```
4.708.3.5 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::unordered_multiset (
initializer_list< value_type > __l, size_type __n = 0, const hasher & __hf = hasher(), const key_equal & __eqf
= key_equal(), const allocator_type & __a = allocator_type() ) [inline]
```

Builds an unordered_multiset from an initializer_list.

Parameters

<code>__l</code>	An initializer_list.
<code>__n</code>	Minimal initial number of buckets.
<code>__hf</code>	A hash functor.
<code>__eqf</code>	A key equality functor.
<code>__a</code>	An allocator object.

Create an unordered_multiset consisting of copies of the elements in the list. This is linear in N (where N is `__l.size()`).

Definition at line 813 of file unordered_set.h.

References `std::unordered_set< _Value, _Hash, _Pred, _Alloc >::operator=()`.

```
4.708.3.6 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::unordered_multiset (
const allocator_type & __a ) [inline], [explicit]
```

Creates an unordered_multiset with no elements.

Parameters

<code>__a</code>	An allocator object.
------------------	----------------------

Definition at line 834 of file unordered_set.h.

References `std::move()`.

4.708.4 Member Function Documentation

```
4.708.4.1 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::begin ( )
[inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the unordered_multiset.

Definition at line 907 of file unordered_set.h.

```
4.708.4.2 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> const_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::begin ( )
const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the unordered_multiset.

Definition at line 911 of file unordered_set.h.

```
4.708.4.3 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> local_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::begin (
size_type __n ) [inline]
```

Returns a read-only (constant) iterator pointing to the first bucket element.

Parameters

$_ \leftrightarrow$	The bucket index.
$_n$	

Returns

A read-only local iterator.

Definition at line 1235 of file unordered_set.h.

```
4.708.4.4  template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> const_local_iterator std::unordered_multiset<_Value, _Hash, _Pred, _Alloc>::begin
( size_type __n ) const    [inline]
```

Returns a read-only (constant) iterator pointing to the first bucket element.

Parameters

$_ \leftrightarrow$	The bucket index.
$_n$	

Returns

A read-only local iterator.

Definition at line 1239 of file unordered_set.h.

```
4.708.4.5  template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> size_type std::unordered_multiset<_Value, _Hash, _Pred, _Alloc>::bucket_count ( )
const    [inline], [noexcept]
```

Returns the number of buckets of the unordered_multiset.

Definition at line 1201 of file unordered_set.h.

```
4.708.4.6  template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> const_iterator std::unordered_multiset<_Value, _Hash, _Pred, _Alloc>::cbegin ( )
const    [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the unordered_multiset.

Definition at line 934 of file unordered_set.h.

```
4.708.4.7  template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> const_local_iterator std::unordered_multiset<_Value, _Hash, _Pred, _Alloc
>::cbegin ( size_type __n ) const    [inline]
```

Returns a read-only (constant) iterator pointing to the first bucket element.

Parameters

$_n$	The bucket index.
-------	-------------------

Returns

A read-only local iterator.

Definition at line 1243 of file unordered_set.h.

```
4.708.4.8  template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> const_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::cend ( )
const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the unordered_multiset.

Definition at line 942 of file unordered_set.h.

```
4.708.4.9  template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> const_local_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::cend
( size_type _n ) const [inline]
```

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

Parameters

$_n$	The bucket index.
-------	-------------------

Returns

A read-only local iterator.

Definition at line 1263 of file unordered_set.h.

```
4.708.4.10 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> void std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::clear ( )
[inline], [noexcept]
```

Erases all elements in an unordered_multiset.

Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1118 of file unordered_set.h.

```
4.708.4.11 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> size_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::count( const
key_type & __x ) const [inline]
```

Finds the number of elements.

Parameters

<code>_↔</code>	Element to located.
<code>_X</code>	

Returns

Number of elements with specified key.

Definition at line 1178 of file unordered_set.h.

```
4.708.4.12  template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
            std::allocator<_Value>> template<typename... _Args> iterator std::unordered_multiset< _Value, _Hash,
            _Pred, _Alloc >::emplace ( _Args &&... __args )  [inline]
```

Builds and insert an element into the unordered_multiset.

Parameters

<code>__args</code>	Arguments used to generate an element.
---------------------	--

Returns

An iterator that points to the inserted element.

Insertion requires amortized constant time.

Definition at line 956 of file unordered_set.h.

```
4.708.4.13  template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
            std::allocator<_Value>> template<typename... _Args> iterator std::unordered_multiset< _Value, _Hash,
            _Pred, _Alloc >::emplace_hint ( const_iterator __pos, _Args &&... __args )  [inline]
```

Inserts an element into the unordered_multiset.

Parameters

<code>__pos</code>	An iterator that serves as a hint as to where the element should be inserted.
<code>__args</code>	Arguments used to generate the element to be inserted.

Returns

An iterator that points to the inserted element.

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.html>.↵

Insertion requires amortized constant time.

Definition at line 978 of file unordered_set.h.

```
4.708.4.14 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> bool std::unordered_multiset<_Value, _Hash, _Pred, _Alloc>::empty ( ) const
[inline], [noexcept]
```

Returns true if the unordered_multiset is empty.

Definition at line 886 of file unordered_set.h.

```
4.708.4.15 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> iterator std::unordered_multiset<_Value, _Hash, _Pred, _Alloc>::end ( )
[inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the unordered_multiset.

Definition at line 921 of file unordered_set.h.

```
4.708.4.16 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> const_iterator std::unordered_multiset<_Value, _Hash, _Pred, _Alloc>::end ( )
const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the unordered_multiset.

Definition at line 925 of file unordered_set.h.

```
4.708.4.17 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> local_iterator std::unordered_multiset<_Value, _Hash, _Pred, _Alloc>::end (
size_type __n ) [inline]
```

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

Parameters

<u>↵</u>	The bucket index.
<u><i>n</i></u>	

Returns

A read-only local iterator.

Definition at line 1255 of file unordered_set.h.

```
4.708.4.18  template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
            std::allocator<_Value>> const_local_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::end
            ( size_type __n ) const    [inline]
```

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

Parameters

<code>__n</code>	The bucket index.
------------------	-------------------

Returns

A read-only local iterator.

Definition at line 1259 of file unordered_set.h.

```
4.708.4.19  template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
            std::allocator<_Value>> std::pair<iterator, iterator> std::unordered_multiset< _Value, _Hash, _Pred,
            _Alloc >::equal_range ( const key_type & __x )    [inline]
```

Finds a subsequence matching given key.

Parameters

<code>__x</code>	Key to be located.
------------------	--------------------

Returns

Pair of iterators that possibly points to the subsequence matching given key.

Definition at line 1189 of file unordered_set.h.

```
4.708.4.20  template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
            std::allocator<_Value>> std::pair<const_iterator, const_iterator> std::unordered_multiset< _Value,
            _Hash, _Pred, _Alloc >::equal_range ( const key_type & __x ) const    [inline]
```

Finds a subsequence matching given key.

Parameters

<code>__x</code>	Key to be located.
------------------	--------------------

Returns

Pair of iterators that possibly points to the subsequence matching given key.

Definition at line 1193 of file unordered_set.h.

```
4.708.4.21 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::erase (
const_iterator __position ) [inline]
```

Erases an element from an unordered_multiset.

Parameters

<code>__position</code>	An iterator pointing to the element to be erased.
-------------------------	---

Returns

An iterator pointing to the element immediately following `__position` prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from an unordered_multiset.

Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1064 of file unordered_set.h.

```
4.708.4.22 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::erase ( iterator
__it ) [inline]
```

Erases an element from an unordered_multiset.

Parameters

<code>__position</code>	An iterator pointing to the element to be erased.
-------------------------	---

Returns

An iterator pointing to the element immediately following `__position` prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from an unordered_multiset.

Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1069 of file unordered_set.h.

```
4.708.4.23  template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
            std::allocator<_Value>> size_type std::unordered_multiset<_Value, _Hash, _Pred, _Alloc >::erase( const
            key_type & __x ) [inline]
```

Erases elements according to the provided key.

Parameters

<code>__x</code>	Key of element to be erased.
------------------	------------------------------

Returns

The number of elements erased.

This function erases all the elements located by the given key from an `unordered_multiset`.

Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1087 of file `unordered_set.h`.

```
4.708.4.24  template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
            std::allocator<_Value>> iterator std::unordered_multiset<_Value, _Hash, _Pred, _Alloc >::erase (
            const_iterator __first, const_iterator __last ) [inline]
```

Erases a [`__first`,`__last`) range of elements from an `unordered_multiset`.

Parameters

<code>__first</code>	Iterator pointing to the start of the range to be erased.
<code>__last</code>	Iterator pointing to the end of the range to be erased.

Returns

The iterator `__last`.

This function erases a sequence of elements from an `unordered_multiset`.

Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1107 of file `unordered_set.h`.

```
4.708.4.25  template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
            std::allocator<_Value>> iterator std::unordered_multiset<_Value, _Hash, _Pred, _Alloc >::find ( const
            key_type & __x ) [inline]
```

Tries to locate an element in an `unordered_multiset`.

Parameters

$_x$	Element to be located.
-------	------------------------

Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end (end()) iterator.

Definition at line 1164 of file unordered_set.h.

```
4.708.4.26 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> const_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::find ( const
key_type & _x ) const [inline]
```

Tries to locate an element in an unordered_multiset.

Parameters

$_x$	Element to be located.
-------	------------------------

Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end (end()) iterator.

Definition at line 1168 of file unordered_set.h.

```
4.708.4.27 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> allocator_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc
>::get_allocator ( ) const [inline], [noexcept]
```

Returns the allocator object with which the unordered_multiset was constructed.

Definition at line 879 of file unordered_set.h.

```
4.708.4.28 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> hasher std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::hash_function ( )
const [inline]
```

Returns the hash functor object with which the unordered_multiset was constructed.

Definition at line 1140 of file unordered_set.h.

4.708.4.29 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::insert (const
value_type & __x) [inline]`

Inserts an element into the unordered_multiset.

Parameters

<code>_↔</code>	Element to be inserted.
<code>_x</code>	

Returns

An iterator that points to the inserted element.

Insertion requires amortized constant time.

Definition at line 990 of file unordered_set.h.

```
4.708.4.30 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::insert (
value_type && _x ) [inline]
```

Inserts an element into the unordered_multiset.

Parameters

<code>_↔</code>	Element to be inserted.
<code>_x</code>	

Returns

An iterator that points to the inserted element.

Insertion requires amortized constant time.

Definition at line 994 of file unordered_set.h.

References std::move().

```
4.708.4.31 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::insert (
const_iterator __hint, const value_type & _x ) [inline]
```

Inserts an element into the unordered_multiset.

Parameters

<code>__hint</code>	An iterator that serves as a hint as to where the element should be inserted.
<code>__x</code>	Element to be inserted.

Returns

An iterator that points to the inserted element.

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17>.↵
html

Insertion requires amortized constant.

Definition at line 1016 of file unordered_set.h.

```
4.708.4.32  template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
            std::allocator<_Value>> iterator std::unordered_multiset<_Value, _Hash, _Pred, _Alloc >::insert (
            const_iterator __hint, value_type && __x ) [inline]
```

Inserts an element into the unordered_multiset.

Parameters

<code>__hint</code>	An iterator that serves as a hint as to where the element should be inserted.
<code>__x</code>	Element to be inserted.

Returns

An iterator that points to the inserted element.

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17>.↵
html

Insertion requires amortized constant.

Definition at line 1020 of file unordered_set.h.

References std::move().

```
4.708.4.33  template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
            std::allocator<_Value>> template<typename _InputIterator > void std::unordered_multiset<_Value, _Hash,
            _Pred, _Alloc >::insert ( _InputIterator __first, _InputIterator __last ) [inline]
```

A template function that inserts a range of elements.

Parameters

<code>__first</code>	Iterator pointing to the start of the range to be inserted.
<code>__last</code>	Iterator pointing to the end of the range.

Complexity similar to that of the range constructor.

Definition at line 1034 of file unordered_set.h.

```
4.708.4.34 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> void std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::insert ( initializer_list<
value_type> &_l ) [inline]
```

Inserts a list of elements into the unordered_multiset.

Parameters

↩	A std::initializer_list<value_type> of elements to be inserted.
↩	
↩	
↩	
↩	
↩	

Complexity similar to that of the range constructor.

Definition at line 1045 of file unordered_set.h.

```
4.708.4.35 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> key_equal std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::key_eq ( )
const [inline]
```

Returns the key comparison object with which the unordered_multiset was constructed.

Definition at line 1146 of file unordered_set.h.

```
4.708.4.36 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> float std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::load_factor ( ) const
[inline], [noexcept]
```

Returns the average number of elements per bucket.

Definition at line 1271 of file unordered_set.h.

```
4.708.4.37 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> size_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc
>::max_bucket_count ( ) const [inline], [noexcept]
```

Returns the maximum number of buckets of the unordered_multiset.

Definition at line 1206 of file unordered_set.h.

```
4.708.4.38 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> float std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::max_load_factor ( )
const [inline], [noexcept]
```

Returns a positive number that the unordered_multiset tries to keep the load factor less than or equal to.

Definition at line 1277 of file unordered_set.h.

4.708.4.39 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> void std::unordered_multiset<_Value, _Hash, _Pred, _Alloc>::max_load_factor (float
__z) [inline]`

Change the unordered_multiset maximum load factor.

Parameters

<code>_Z</code>	The new maximum load factor.
-----------------	------------------------------

Definition at line 1285 of file unordered_set.h.

```
4.708.4.40 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> size_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::max_size ( )
const [inline], [noexcept]
```

Returns the maximum size of the unordered_multiset.

Definition at line 896 of file unordered_set.h.

```
4.708.4.41 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> unordered_multiset& std::unordered_multiset< _Value, _Hash, _Pred, _Alloc
>::operator= ( const unordered_multiset< _Value, _Hash, _Pred, _Alloc > & ) [default]
```

Copy assignment operator.

```
4.708.4.42 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> unordered_multiset& std::unordered_multiset< _Value, _Hash, _Pred, _Alloc
>::operator= ( unordered_multiset< _Value, _Hash, _Pred, _Alloc > && ) [default]
```

Move assignment operator.

```
4.708.4.43 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> unordered_multiset& std::unordered_multiset< _Value, _Hash, _Pred, _Alloc
>::operator= ( initializer_list< value_type > __l ) [inline]
```

Unordered_multiset list assignment operator.

Parameters

<code>__l</code>	An initializer_list.
------------------	----------------------

This function fills an unordered_multiset with copies of the elements in the initializer list `__l`.

Note that the assignment completely changes the unordered_multiset and that the resulting unordered_set's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 870 of file unordered_set.h.

4.708.4.44 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> void std::unordered_multiset<_Value, _Hash, _Pred, _Alloc>::rehash (size_type __n) [inline]`

May rehash the unordered_multiset.

Parameters

<code>__n</code>	The new number of buckets.
------------------	----------------------------

Rehash will occur only if the new number of buckets respect the unordered_multiset maximum load factor.

Definition at line 1296 of file unordered_set.h.

4.708.4.45 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> void std::unordered_multiset<_Value, _Hash, _Pred, _Alloc>::reserve (size_type __n) [inline]`

Prepare the unordered_multiset for a specified number of elements.

Parameters

<code>__n</code>	Number of elements required.
------------------	------------------------------

Same as `rehash(ceil(n / max_load_factor()))`.

Definition at line 1307 of file unordered_set.h.

References `std::unordered_set<_Value, _Hash, _Pred, _Alloc>::swap()`, and `std::unordered_multiset<_Value, __Hash, _Pred, _Alloc>::swap()`.

4.708.4.46 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> size_type std::unordered_multiset<_Value, _Hash, _Pred, _Alloc>::size () const [inline], [noexcept]`

Returns the size of the unordered_multiset.

Definition at line 891 of file unordered_set.h.

4.708.4.47 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> void std::unordered_multiset<_Value, _Hash, _Pred, _Alloc>::swap (unordered_multiset<_Value, _Hash, _Pred, _Alloc> &__x) [inline], [noexcept]`

Swaps data with another unordered_multiset.

Parameters

<code>_↔</code>	An unordered_multiset of the same element and allocator types.
<code>_X</code>	

This exchanges the elements between two sets in constant time. Note that the global `std::swap()` function is specialized such that `std::swap(s1,s2)` will feed to this function.

Definition at line 1131 of file `unordered_set.h`.

Referenced by `std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::reserve()`.

The documentation for this class was generated from the following file:

- [unordered_set.h](#)

4.709 `std::unordered_set< _Value, _Hash, _Pred, _Alloc >` Class Template Reference

Public Types

- `typedef _Hashtable::key_type` [key_type](#)
- `typedef _Hashtable::value_type` [value_type](#)
- `typedef _Hashtable::hasher` [hasher](#)
- `typedef _Hashtable::key_equal` [key_equal](#)
- `typedef _Hashtable::allocator_type` [allocator_type](#)
- `typedef _Hashtable::pointer` [pointer](#)
- `typedef _Hashtable::const_pointer` [const_pointer](#)
- `typedef _Hashtable::reference` [reference](#)
- `typedef _Hashtable::const_reference` [const_reference](#)
- `typedef _Hashtable::iterator` [iterator](#)
- `typedef _Hashtable::const_iterator` [const_iterator](#)
- `typedef _Hashtable::local_iterator` [local_iterator](#)
- `typedef _Hashtable::const_local_iterator` [const_local_iterator](#)
- `typedef _Hashtable::size_type` [size_type](#)
- `typedef _Hashtable::difference_type` [difference_type](#)

Public Member Functions

- `unordered_set` (`size_type` __n=10, const `hasher` &__hf=`hasher`(), const `key_equal` &__eq=`key_equal`(), const `allocator_type` &__a=`allocator_type`())
- `template<typename _InputIterator >`
`unordered_set` (`_InputIterator` __f, `_InputIterator` __l, `size_type` __n=0, const `hasher` &__hf=`hasher`(), const `key_↵`
`_equal` &__eq=`key_equal`(), const `allocator_type` &__a=`allocator_type`())
- `unordered_set` (const `unordered_set` &)=default
- `unordered_set` (`unordered_set` &&)=default
- `unordered_set` (const `allocator_type` &__a)
- `unordered_set` (const `unordered_set` &__uset, const `allocator_type` &__a)
- `unordered_set` (`unordered_set` &&__uset, const `allocator_type` &__a)
- `unordered_set` (initializer_list< `value_type` > __l, `size_type` __n=0, const `hasher` &__hf=`hasher`(), const `key_↵`
`equal` &__eq=`key_equal`(), const `allocator_type` &__a=`allocator_type`())
- `size_type bucket` (const `key_type` &__key) const
- `size_type bucket_count` () const noexcept
- `size_type bucket_size` (`size_type` __n) const
- `const_iterator cbegin` () const noexcept
- `const_iterator cend` () const noexcept
- `void clear` () noexcept
- `size_type count` (const `key_type` &__x) const
- `template<typename... _Args>`
`std::pair< iterator, bool > emplace` (`_Args` &&... __args)
- `template<typename... _Args>`
`iterator emplace_hint` (const `iterator` __pos, `_Args` &&... __args)
- `bool empty` () const noexcept
- `size_type erase` (const `key_type` &__x)
- `iterator erase` (const `iterator` __first, const `iterator` __last)
- `allocator_type get_allocator` () const noexcept
- `hasher hash_function` () const
- `template<typename _InputIterator >`
`void insert` (`_InputIterator` __first, `_InputIterator` __last)
- `void insert` (initializer_list< `value_type` > __l)
- `key_equal key_eq` () const
- `float load_factor` () const noexcept
- `size_type max_bucket_count` () const noexcept
- `float max_load_factor` () const noexcept
- `void max_load_factor` (float __z)
- `size_type max_size` () const noexcept
- `unordered_set & operator=` (const `unordered_set` &)=default
- `unordered_set & operator=` (`unordered_set` &&)=default
- `unordered_set & operator=` (initializer_list< `value_type` > __l)
- `void rehash` (`size_type` __n)
- `void reserve` (`size_type` __n)
- `size_type size` () const noexcept
- `void swap` (`unordered_set` &__x) noexcept(noexcept(_M_h.swap(__x._M_h)))

- `iterator begin` () noexcept
- `const_iterator begin` () const noexcept

- `iterator end` () noexcept

- `const_iterator end` () const noexcept
- `std::pair< iterator, bool > insert` (const `value_type` &__x)
- `std::pair< iterator, bool > insert` (`value_type` &&__x)
- `iterator insert` (const `iterator` __hint, const `value_type` &__x)
- `iterator insert` (const `iterator` __hint, `value_type` &&__x)
- `iterator erase` (const `iterator` __position)
- `iterator erase` (`iterator` __it)
- `iterator find` (const `key_type` &__x)
- `const_iterator find` (const `key_type` &__x) const
- `std::pair< iterator, iterator > equal_range` (const `key_type` &__x)
- `std::pair< const_iterator, const_iterator > equal_range` (const `key_type` &__x) const
- `local_iterator begin` (size_type __n)
- `const_local_iterator begin` (size_type __n) const
- `const_local_iterator cbegin` (size_type __n) const
- `local_iterator end` (size_type __n)
- `const_local_iterator end` (size_type __n) const
- `const_local_iterator cend` (size_type __n) const

Friends

- `template<typename _Value1, typename _Hash1, typename _Pred1, typename _Alloc1 > bool operator==` (const `unordered_set< _Value1, _Hash1, _Pred1, _Alloc1 >` &, const `unordered_set< _Value1, _Hash1, _Pred1, _Alloc1 >` &)

4.709.1 Detailed Description

```
template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>>
class std::unordered_set< _Value, _Hash, _Pred, _Alloc >
```

A standard container composed of unique keys (containing at most one of each key value) in which the elements' keys are the elements themselves.

Template Parameters

<code>_Value</code>	Type of key objects.
<code>_Hash</code>	Hashing function object type, defaults to <code>hash<_Value></code> .
<code>_Pred</code>	Predicate function object type, defaults to <code>equal_to<_Value></code> .
<code>_Alloc</code>	Allocator type, defaults to <code>allocator<_Key></code> .

Meets the requirements of a `container`, and `unordered associative container`

Base is `_Hashtable`, dispatched at compile time via template alias `__uset_hashtable`.

Definition at line 93 of file `unordered_set.h`.

4.709.2 Member Typedef Documentation

4.709.2.1 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> typedef _Hashtable::allocator_type std::unordered_set<_Value, _Hash, _Pred, _Alloc>::allocator_type`

Public typedefs.

Definition at line 106 of file `unordered_set.h`.

4.709.2.2 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> typedef _Hashtable::const_iterator std::unordered_set<_Value, _Hash, _Pred, _Alloc>::const_iterator`

Iterator-related typedefs.

Definition at line 116 of file `unordered_set.h`.

4.709.2.3 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> typedef _Hashtable::const_local_iterator std::unordered_set<_Value, _Hash, _Pred, _Alloc>::const_local_iterator`

Iterator-related typedefs.

Definition at line 118 of file `unordered_set.h`.

4.709.2.4 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> typedef _Hashtable::const_pointer std::unordered_set<_Value, _Hash, _Pred, _Alloc>::const_pointer`

Iterator-related typedefs.

Definition at line 112 of file `unordered_set.h`.

4.709.2.5 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> typedef _Hashtable::const_reference std::unordered_set<_Value, _Hash, _Pred, _Alloc>::const_reference`

Iterator-related typedefs.

Definition at line 114 of file `unordered_set.h`.

```
4.709.2.6 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> typedef _Hashtable::difference_type std::unordered_set< _Value, _Hash, _Pred, _Alloc
>::difference_type
```

Iterator-related typedefs.

Definition at line 120 of file unordered_set.h.

```
4.709.2.7 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> typedef _Hashtable::hasher std::unordered_set< _Value, _Hash, _Pred, _Alloc >::hasher
```

Public typedefs.

Definition at line 104 of file unordered_set.h.

```
4.709.2.8 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> typedef _Hashtable::iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::iterator
```

Iterator-related typedefs.

Definition at line 115 of file unordered_set.h.

```
4.709.2.9 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> typedef _Hashtable::key_equal std::unordered_set< _Value, _Hash, _Pred, _Alloc
>::key_equal
```

Public typedefs.

Definition at line 105 of file unordered_set.h.

```
4.709.2.10 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> typedef _Hashtable::key_type std::unordered_set< _Value, _Hash, _Pred, _Alloc
>::key_type
```

Public typedefs.

Definition at line 102 of file unordered_set.h.

```
4.709.2.11 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> typedef _Hashtable::local_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc
>::local_iterator
```

Iterator-related typedefs.

Definition at line 117 of file unordered_set.h.

4.709.2.12 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> typedef _Hashtable::pointer std::unordered_set< _Value, _Hash, _Pred, _Alloc >::pointer`

Iterator-related typedefs.

Definition at line 111 of file `unordered_set.h`.

4.709.2.13 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> typedef _Hashtable::reference std::unordered_set< _Value, _Hash, _Pred, _Alloc >::reference`

Iterator-related typedefs.

Definition at line 113 of file `unordered_set.h`.

4.709.2.14 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> typedef _Hashtable::size_type std::unordered_set< _Value, _Hash, _Pred, _Alloc >::size_type`

Iterator-related typedefs.

Definition at line 119 of file `unordered_set.h`.

4.709.2.15 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> typedef _Hashtable::value_type std::unordered_set< _Value, _Hash, _Pred, _Alloc >::value_type`

Public typedefs.

Definition at line 103 of file `unordered_set.h`.

4.709.3 Constructor & Destructor Documentation

4.709.3.1 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> std::unordered_set< _Value, _Hash, _Pred, _Alloc >::unordered_set(size_type __n = 10, const hasher & __hf = hasher(), const key_equal & __eqf = key_equal(), const allocator_type & __a = allocator_type()) [inline], [explicit]`

Default constructor creates no elements.

Parameters

<code>__n</code>	Initial number of buckets.
<code>__hf</code>	A hash functor.
<code>__eqf</code>	A key equality functor.
<code>__a</code>	An allocator object.

Definition at line 132 of file unordered_set.h.

Referenced by std::unordered_set< _Value, _Hash, _Pred, _Alloc >::unordered_set().

4.709.3.2 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> template<typename _InputIterator > std::unordered_set< _Value, _Hash, _Pred, _Alloc >::unordered_set (_InputIterator __f, _InputIterator __l, size_type __n = 0, const hasher & __hf = hasher(), const key_equal & __eqf = key_equal(), const allocator_type & __a = allocator_type()) [inline]`

Builds an unordered_set from a range.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__n</code>	Minimal initial number of buckets.
<code>__hf</code>	A hash functor.
<code>__eqf</code>	A key equality functor.
<code>__a</code>	An allocator object.

Create an unordered_set consisting of copies of the elements from [`__first`,`__last`). This is linear in N (where N is `distance(__first,__last)`).

Definition at line 153 of file unordered_set.h.

References std::unordered_set< _Value, _Hash, _Pred, _Alloc >::unordered_set().

4.709.3.3 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> std::unordered_set< _Value, _Hash, _Pred, _Alloc >::unordered_set (const unordered_set< _Value, _Hash, _Pred, _Alloc > &) [default]`

Copy constructor.

4.709.3.4 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> std::unordered_set< _Value, _Hash, _Pred, _Alloc >::unordered_set (unordered_set< _Value, _Hash, _Pred, _Alloc > &&) [default]`

Move constructor.

4.709.3.5 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> std::unordered_set< _Value, _Hash, _Pred, _Alloc >::unordered_set (const allocator_type & __a) [inline],[explicit]`

Creates an unordered_set with no elements.

Parameters

<code>__a</code>	An allocator object.
------------------	----------------------

Definition at line 172 of file unordered_set.h.

References `std::move()`, and `std::unordered_set<_Value, _Hash, _Pred, _Alloc>::unordered_set()`.

4.709.3.6 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>>> std::unordered_set<_Value, _Hash, _Pred, _Alloc>::unordered_set(initializer_list<value_type> __l, size_type __n = 0, const hasher & __hf = hasher(), const key_equal & __eqf = key_equal(), const allocator_type & __a = allocator_type()) [inline]`

Builds an `unordered_set` from an `initializer_list`.

Parameters

<code>__l</code>	An <code>initializer_list</code> .
<code>__n</code>	Minimal initial number of buckets.
<code>__hf</code>	A hash functor.
<code>__eqf</code>	A key equality functor.
<code>__a</code>	An allocator object.

Create an `unordered_set` consisting of copies of the elements in the list. This is linear in N (where N is `__l.size()`).

Definition at line 207 of file unordered_set.h.

References `std::unordered_set<_Value, _Hash, _Pred, _Alloc>::operator=()`.

4.709.4 Member Function Documentation

4.709.4.1 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>>> iterator std::unordered_set<_Value, _Hash, _Pred, _Alloc>::begin() [inline], [noexcept]`

Returns a read-only (constant) iterator that points to the first element in the `unordered_set`.

Definition at line 272 of file unordered_set.h.

4.709.4.2 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>>> const_iterator std::unordered_set<_Value, _Hash, _Pred, _Alloc>::begin() const [inline], [noexcept]`

Returns a read-only (constant) iterator that points to the first element in the `unordered_set`.

Definition at line 276 of file unordered_set.h.

4.709.4.3 `template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>>> local_iterator std::unordered_set<_Value, _Hash, _Pred, _Alloc>::begin(size_type __n) [inline]`

Returns a read-only (constant) iterator pointing to the first bucket element.

Parameters

$_n$	The bucket index.
-------	-------------------

Returns

A read-only local iterator.

Definition at line 623 of file unordered_set.h.

```
4.709.4.4  template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> const_local_iterator std::unordered_set<_Value, _Hash, _Pred, _Alloc>::begin (
size_type __n ) const  [inline]
```

Returns a read-only (constant) iterator pointing to the first bucket element.

Parameters

$_n$	The bucket index.
-------	-------------------

Returns

A read-only local iterator.

Definition at line 627 of file unordered_set.h.

```
4.709.4.5  template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> size_type std::unordered_set<_Value, _Hash, _Pred, _Alloc>::bucket_count ( ) const
[inline], [noexcept]
```

Returns the number of buckets of the unordered_set.

Definition at line 589 of file unordered_set.h.

```
4.709.4.6  template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> const_iterator std::unordered_set<_Value, _Hash, _Pred, _Alloc>::cbegin ( ) const
[inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the unordered_set.

Definition at line 299 of file unordered_set.h.

```
4.709.4.7  template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> const_local_iterator std::unordered_set<_Value, _Hash, _Pred, _Alloc>::cbegin (
size_type __n ) const  [inline]
```

Returns a read-only (constant) iterator pointing to the first bucket element.

Parameters

$_ \leftrightarrow$ $_n$	The bucket index.
-------------------------------	-------------------

Returns

A read-only local iterator.

Definition at line 631 of file unordered_set.h.

```
4.709.4.8  template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> const_iterator std::unordered_set<_Value, _Hash, _Pred, _Alloc>::end ( ) const
[inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the unordered_set.

Definition at line 307 of file unordered_set.h.

```
4.709.4.9  template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> const_local_iterator std::unordered_set<_Value, _Hash, _Pred, _Alloc>::end (
size_type __n ) const [inline]
```

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

Parameters

$_ \leftrightarrow$ $_n$	The bucket index.
-------------------------------	-------------------

Returns

A read-only local iterator.

Definition at line 651 of file unordered_set.h.

```
4.709.4.10 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> void std::unordered_set<_Value, _Hash, _Pred, _Alloc>::clear ( ) [inline],
[noexcept]
```

Erases all elements in an unordered_set. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 500 of file unordered_set.h.

```
4.709.4.11 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> size_type std::unordered_set<_Value, _Hash, _Pred, _Alloc>::count ( const
key_type & __x ) const [inline]
```

Finds the number of elements.

Parameters

<code>_↔</code>	Element to located.
<code>_X</code>	

Returns

Number of elements with specified key.

This function only makes sense for unordered_multisets; for unordered_set the result will either be 0 (not present) or 1 (present).

Definition at line 564 of file unordered_set.h.

```
4.709.4.12 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> template<typename... _Args> std::pair<iterator, bool> std::unordered_set<
_Value, _Hash, _Pred, _Alloc >::emplace ( _Args &&... _args ) [inline]
```

Attempts to build and insert an element into the unordered_set.

Parameters

<code>__args</code>	Arguments used to generate an element.
---------------------	--

Returns

A pair, of which the first element is an iterator that points to the possibly inserted element, and the second is a bool that is true if the element was actually inserted.

This function attempts to build and insert an element into the unordered_set. An unordered_set relies on unique keys and thus an element is only inserted if it is not already present in the unordered_set.

Insertion requires amortized constant time.

Definition at line 329 of file unordered_set.h.

```
4.709.4.13 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> template<typename... _Args> iterator std::unordered_set< _Value, _Hash, _Pred,
_Alloc >::emplace_hint ( const_iterator __pos, _Args &&... _args ) [inline]
```

Attempts to insert an element into the unordered_set.

Parameters

<code>__pos</code>	An iterator that serves as a hint as to where the element should be inserted.
<code>__args</code>	Arguments used to generate the element to be inserted.

Returns

An iterator that points to the element with key equivalent to the one generated from `__args` (may or may not be the element itself).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument `emplace()` does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.html>.

Insertion requires amortized constant time.

Definition at line 355 of file `unordered_set.h`.

```
4.709.4.14  template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
            std::allocator<_Value>> bool std::unordered_set<_Value, _Hash, _Pred, _Alloc >::empty ( ) const
            [inline], [noexcept]
```

Returns true if the `unordered_set` is empty.

Definition at line 251 of file `unordered_set.h`.

```
4.709.4.15  template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
            std::allocator<_Value>> iterator std::unordered_set<_Value, _Hash, _Pred, _Alloc >::end ( ) [inline],
            [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the `unordered_set`.

Definition at line 286 of file `unordered_set.h`.

```
4.709.4.16  template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
            std::allocator<_Value>> const_iterator std::unordered_set<_Value, _Hash, _Pred, _Alloc >::end ( ) const
            [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the `unordered_set`.

Definition at line 290 of file `unordered_set.h`.

```
4.709.4.17  template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
            std::allocator<_Value>> local_iterator std::unordered_set<_Value, _Hash, _Pred, _Alloc >::end ( size_type
            __n ) [inline]
```

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

Parameters

<code>__n</code>	The bucket index.
------------------	-------------------

Returns

A read-only local iterator.

Definition at line 643 of file unordered_set.h.

```
4.709.4.18 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> const_local_iterator std::unordered_set<_Value, _Hash, _Pred, _Alloc>::end (
size_type __n ) const [inline]
```

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

Parameters

$_n$	The bucket index.
-------	-------------------

Returns

A read-only local iterator.

Definition at line 647 of file unordered_set.h.

```
4.709.4.19 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> std::pair<iterator, iterator> std::unordered_set<_Value, _Hash, _Pred, _Alloc>::
equal_range ( const key_type & __x ) [inline]
```

Finds a subsequence matching given key.

Parameters

$_x$	Key to be located.
-------	--------------------

Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function probably only makes sense for multisets.

Definition at line 577 of file unordered_set.h.

```
4.709.4.20 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> std::pair<const_iterator, const_iterator> std::unordered_set<_Value, _Hash,
_Pred, _Alloc>::equal_range ( const key_type & __x ) const [inline]
```

Finds a subsequence matching given key.

Parameters

<code>__key</code>	Key to be located.
--------------------	--------------------

Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function probably only makes sense for multisets.

Definition at line 581 of file `unordered_set.h`.

```
4.709.4.21  template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
             std::allocator<_Value>> iterator std::unordered_set<_Value, _Hash, _Pred, _Alloc>::erase ( const_iterator
             __position ) [inline]
```

Erases an element from an `unordered_set`.

Parameters

<code>__position</code>	An iterator pointing to the element to be erased.
-------------------------	---

Returns

An iterator pointing to the element immediately following `__position` prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from an `unordered_set`. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 450 of file `unordered_set.h`.

```
4.709.4.22  template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
             std::allocator<_Value>> iterator std::unordered_set<_Value, _Hash, _Pred, _Alloc>::erase ( iterator __it )
             [inline]
```

Erases an element from an `unordered_set`.

Parameters

<code>__position</code>	An iterator pointing to the element to be erased.
-------------------------	---

Returns

An iterator pointing to the element immediately following `__position` prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from an `unordered_set`. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 455 of file `unordered_set.h`.

```
4.709.4.23  template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
            std::allocator<_Value>> size_type std::unordered_set<_Value, _Hash, _Pred, _Alloc >::erase ( const
            key_type & __x ) [inline]
```

Erases elements according to the provided key.

Parameters

<code>__x</code>	Key of element to be erased.
------------------	------------------------------

Returns

The number of elements erased.

This function erases all the elements located by the given key from an `unordered_set`. For an `unordered_set` the result of this function can only be 0 (not present) or 1 (present). Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 472 of file `unordered_set.h`.

```
4.709.4.24  template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
            std::allocator<_Value>> iterator std::unordered_set<_Value, _Hash, _Pred, _Alloc >::erase ( const_iterator
            __first, const_iterator __last ) [inline]
```

Erases a [`__first`,`__last`) range of elements from an `unordered_set`.

Parameters

<code>__first</code>	Iterator pointing to the start of the range to be erased.
<code>__last</code>	Iterator pointing to the end of the range to be erased.

Returns

The iterator `__last`.

This function erases a sequence of elements from an `unordered_set`. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 490 of file `unordered_set.h`.

```
4.709.4.25  template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
            std::allocator<_Value>> iterator std::unordered_set<_Value, _Hash, _Pred, _Alloc>::find ( const key_type &
            __x ) [inline]
```

Tries to locate an element in an `unordered_set`.

Parameters

<code>__x</code>	Element to be located.
------------------	------------------------

Returns

Iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end (`end()`) iterator.

Definition at line 546 of file `unordered_set.h`.

```
4.709.4.26  template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
            std::allocator<_Value>> const_iterator std::unordered_set<_Value, _Hash, _Pred, _Alloc>::find ( const
            key_type & __x ) const [inline]
```

Tries to locate an element in an `unordered_set`.

Parameters

<code>__x</code>	Element to be located.
------------------	------------------------

Returns

Iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end (`end()`) iterator.

Definition at line 550 of file `unordered_set.h`.


```
4.709.4.27 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> allocator_type std::unordered_set<_Value, _Hash, _Pred, _Alloc >::get_allocator ( )
const [inline],[noexcept]
```

Returns the allocator object with which the unordered_set was constructed.

Definition at line 244 of file unordered_set.h.

```
4.709.4.28 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> hasher std::unordered_set<_Value, _Hash, _Pred, _Alloc >::hash_function ( ) const
[inline]
```

Returns the hash functor object with which the unordered_set was constructed.

Definition at line 522 of file unordered_set.h.

```
4.709.4.29 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> std::pair<iterator, bool> std::unordered_set<_Value, _Hash, _Pred, _Alloc >::insert
( const value_type & __x ) [inline]
```

Attempts to insert an element into the unordered_set.

Parameters

<code>__x</code>	Element to be inserted.
------------------	-------------------------

Returns

A pair, of which the first element is an iterator that points to the possibly inserted element, and the second is a bool that is true if the element was actually inserted.

This function attempts to insert an element into the unordered_set. An unordered_set relies on unique keys and thus an element is only inserted if it is not already present in the unordered_set.

Insertion requires amortized constant time.

Definition at line 373 of file unordered_set.h.

```
4.709.4.30 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> std::pair<iterator, bool> std::unordered_set<_Value, _Hash, _Pred, _Alloc >::insert
( value_type && __x ) [inline]
```

Attempts to insert an element into the unordered_set.

Parameters

<code>__x</code>	Element to be inserted.
------------------	-------------------------

Returns

A pair, of which the first element is an iterator that points to the possibly inserted element, and the second is a bool that is true if the element was actually inserted.

This function attempts to insert an element into the `unordered_set`. An `unordered_set` relies on unique keys and thus an element is only inserted if it is not already present in the `unordered_set`.

Insertion requires amortized constant time.

Definition at line 377 of file `unordered_set.h`.

References `std::move()`.

```
4.709.4.31  template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
            std::allocator<_Value>> iterator std::unordered_set<_Value, _Hash, _Pred, _Alloc>::insert ( const_iterator
            __hint, const value_type & __x ) [inline]
```

Attempts to insert an element into the `unordered_set`.

Parameters

<code>__hint</code>	An iterator that serves as a hint as to where the element should be inserted.
<code>__x</code>	Element to be inserted.

Returns

An iterator that points to the element with key of `__x` (may or may not be the element passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument `insert()` does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.html>.↔

Insertion requires amortized constant.

Definition at line 402 of file `unordered_set.h`.

```
4.709.4.32  template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
            std::allocator<_Value>> iterator std::unordered_set<_Value, _Hash, _Pred, _Alloc>::insert ( const_iterator
            __hint, value_type && __x ) [inline]
```

Attempts to insert an element into the `unordered_set`.

Parameters

<code>__hint</code>	An iterator that serves as a hint as to where the element should be inserted.
<code>__x</code>	Element to be inserted.

Returns

An iterator that points to the element with key of `__x` (may or may not be the element passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument `insert()` does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.html>.

Insertion requires amortized constant.

Definition at line 406 of file `unordered_set.h`.

References `std::move()`.

```
4.709.4.33 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> template<typename _InputIterator> void std::unordered_set<_Value, _Hash, _Pred,
_Alloc>::insert ( _InputIterator __first, _InputIterator __last ) [inline]
```

A template function that attempts to insert a range of elements.

Parameters

<code>__first</code>	Iterator pointing to the start of the range to be inserted.
<code>__last</code>	Iterator pointing to the end of the range.

Complexity similar to that of the range constructor.

Definition at line 421 of file `unordered_set.h`.

```
4.709.4.34 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> void std::unordered_set<_Value, _Hash, _Pred, _Alloc>::insert ( initializer_list<
value_type> __l ) [inline]
```

Attempts to insert a list of elements into the `unordered_set`.

Parameters

<code>__l</code>	A <code>std::initializer_list<value_type></code> of elements to be inserted.
------------------	--

Complexity similar to that of the range constructor.

Definition at line 432 of file `unordered_set.h`.

```
4.709.4.35 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> key_equal std::unordered_set<_Value, _Hash, _Pred, _Alloc>::key_eq ( ) const
[inline]
```

Returns the key comparison object with which the unordered_set was constructed.

Definition at line 528 of file unordered_set.h.

```
4.709.4.36 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> float std::unordered_set<_Value, _Hash, _Pred, _Alloc>::load_factor ( ) const
[inline], [noexcept]
```

Returns the average number of elements per bucket.

Definition at line 659 of file unordered_set.h.

```
4.709.4.37 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> size_type std::unordered_set<_Value, _Hash, _Pred, _Alloc>::max_bucket_count ( )
const [inline], [noexcept]
```

Returns the maximum number of buckets of the unordered_set.

Definition at line 594 of file unordered_set.h.

```
4.709.4.38 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> float std::unordered_set<_Value, _Hash, _Pred, _Alloc>::max_load_factor ( ) const
[inline], [noexcept]
```


Returns a positive number that the unordered_set tries to keep the load factor less than or equal to.

Definition at line 665 of file unordered_set.h.

```
4.709.4.39 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> void std::unordered_set<_Value, _Hash, _Pred, _Alloc>::max_load_factor ( float __z )
[inline]
```

Change the unordered_set maximum load factor.

Parameters

	The new maximum load factor.
__z	

Definition at line 673 of file unordered_set.h.

```
4.709.4.40 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
std::allocator<_Value>> size_type std::unordered_set<_Value, _Hash, _Pred, _Alloc>::max_size ( ) const
[inline], [noexcept]
```

Returns the maximum size of the unordered_set.


Rehash will occur only if the new number of buckets respect the unordered_set maximum load factor.

Definition at line 684 of file unordered_set.h.

```
4.709.4.45  template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
            std::allocator<_Value>> void std::unordered_set< _Value, _Hash, _Pred, _Alloc >::reserve ( size_type __n )
            [inline]
```

Prepare the unordered_set for a specified number of elements.

Parameters

 <code>__n</code>	Number of elements required.
--	------------------------------

Same as `rehash(ceil(n / max_load_factor()))`.

Definition at line 695 of file unordered_set.h.

```
4.709.4.46  template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
            std::allocator<_Value>> size_type std::unordered_set< _Value, _Hash, _Pred, _Alloc >::size ( ) const
            [inline], [noexcept]
```


Returns the size of the unordered_set.

Definition at line 256 of file unordered_set.h.

```
4.709.4.47  template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc =
            std::allocator<_Value>> void std::unordered_set< _Value, _Hash, _Pred, _Alloc >::swap ( unordered_set<
            _Value, _Hash, _Pred, _Alloc > & __x ) [inline], [noexcept]
```

Swaps data with another unordered_set.

Parameters

 <code>__x</code>	An unordered_set of the same element and allocator types.
--	---

This exchanges the elements between two sets in constant time. Note that the global `std::swap()` function is specialized such that `std::swap(s1,s2)` will feed to this function.

Definition at line 513 of file unordered_set.h.

Referenced by `std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::reserve()`.

The documentation for this class was generated from the following file:

- [unordered_set.h](#)

4.710 std::uses_allocator< _Tp, _Alloc > Struct Template Reference

Inherits integral_constant< bool, __uses_allocator_helper< _Tp, _Alloc >::value >.

4.710.1 Detailed Description

```
template<typename _Tp, typename _Alloc>
struct std::uses_allocator< _Tp, _Alloc >
```

Declare uses_allocator so it can be specialized in <queue> etc.

[allocator.uses.trait]

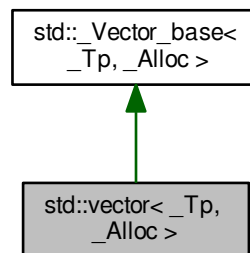
Definition at line 71 of file memoryfwd.h.

The documentation for this struct was generated from the following file:

- [memoryfwd.h](#)

4.711 std::vector< _Tp, _Alloc > Class Template Reference

Inheritance diagram for std::vector< _Tp, _Alloc >:



Public Types

- typedef _Alloc **allocator_type**
- typedef __gnu_cxx::__normal_iterator< const_pointer, [vector](#) > **const_iterator**
- typedef _Alloc_traits::const_pointer **const_pointer**
- typedef _Alloc_traits::const_reference **const_reference**
- typedef [std::reverse_iterator](#)< const_iterator > **const_reverse_iterator**
- typedef ptrdiff_t **difference_type**
- typedef __gnu_cxx::__normal_iterator< pointer, [vector](#) > **iterator**
- typedef _Base::pointer **pointer**
- typedef _Alloc_traits::reference **reference**
- typedef [std::reverse_iterator](#)< iterator > **reverse_iterator**
- typedef size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- [vector](#) () noexcept(is_nothrow_default_constructible< __Alloc >::value)
- [vector](#) (const allocator_type &__a) noexcept
- [vector](#) (size_type __n, const allocator_type &__a=allocator_type())
- [vector](#) (size_type __n, const value_type &__value, const allocator_type &__a=allocator_type())
- [vector](#) (const [vector](#) &__x)
- [vector](#) ([vector](#) &&__x) noexcept
- [vector](#) (const [vector](#) &__x, const allocator_type &__a)
- [vector](#) ([vector](#) &&__rv, const allocator_type &__m) noexcept(_Alloc_traits::S_always_equal())
- [vector](#) (initializer_list< value_type > __l, const allocator_type &__a=allocator_type())
- template<typename __InputIterator, typename = std::_RequireInputIter<__InputIterator>>>
 [vector](#) (__InputIterator __first, __InputIterator __last, const allocator_type &__a=allocator_type())
- ~[vector](#) () noexcept
- void [assign](#) (size_type __n, const value_type &__val)
- template<typename __InputIterator, typename = std::_RequireInputIter<__InputIterator>>>
 void [assign](#) (__InputIterator __first, __InputIterator __last)
- void [assign](#) (initializer_list< value_type > __l)
- reference [at](#) (size_type __n)
- const_reference [at](#) (size_type __n) const
- reference [back](#) () noexcept
- const_reference [back](#) () const noexcept
- iterator [begin](#) () noexcept
- const_iterator [begin](#) () const noexcept
- size_type [capacity](#) () const noexcept
- const_iterator [cbegin](#) () const noexcept
- const_iterator [cend](#) () const noexcept
- void [clear](#) () noexcept
- const_reverse_iterator [crbegin](#) () const noexcept
- const_reverse_iterator [crend](#) () const noexcept
- __Tp * [data](#) () noexcept
- const __Tp * [data](#) () const noexcept
- template<typename... __Args>
 iterator [emplace](#) (const_iterator __position, __Args &&... __args)
- template<typename... __Args>
 void [emplace_back](#) (__Args &&... __args)
- bool [empty](#) () const noexcept
- iterator [end](#) () noexcept
- const_iterator [end](#) () const noexcept
- iterator [erase](#) (const_iterator __position)
- iterator [erase](#) (const_iterator __first, const_iterator __last)
- reference [front](#) () noexcept
- const_reference [front](#) () const noexcept
- iterator [insert](#) (const_iterator __position, const value_type &__x)
- iterator [insert](#) (const_iterator __position, value_type &&__x)
- iterator [insert](#) (const_iterator __position, initializer_list< value_type > __l)
- iterator [insert](#) (const_iterator __position, size_type __n, const value_type &__x)
- template<typename __InputIterator, typename = std::_RequireInputIter<__InputIterator>>>
 iterator [insert](#) (const_iterator __position, __InputIterator __first, __InputIterator __last)
- size_type [max_size](#) () const noexcept
- [vector](#) & [operator=](#) (const [vector](#) &__x)

- [vector](#) & [operator=](#) ([vector](#) &&__x) noexcept([_Alloc_traits::S_nothrow_move](#)())
- [vector](#) & [operator=](#) ([initializer_list](#)< [value_type](#) > __l)
- reference [operator\[\]](#) ([size_type](#) __n) noexcept
- const_reference [operator\[\]](#) ([size_type](#) __n) const noexcept
- void [pop_back](#) () noexcept
- void [push_back](#) (const [value_type](#) &__x)
- void [push_back](#) ([value_type](#) &&__x)
- [reverse_iterator](#) [rbegin](#) () noexcept
- const [reverse_iterator](#) [rbegin](#) () const noexcept
- [reverse_iterator](#) [rend](#) () noexcept
- const [reverse_iterator](#) [rend](#) () const noexcept
- void [reserve](#) ([size_type](#) __n)
- void [resize](#) ([size_type](#) __new_size)
- void [resize](#) ([size_type](#) __new_size, const [value_type](#) &__x)
- void [shrink_to_fit](#) ()
- [size_type](#) [size](#) () const noexcept
- void [swap](#) ([vector](#) &__x) noexcept([_Alloc_traits::S_nothrow_swap](#)())

Protected Member Functions

- pointer [_M_allocate](#) ([size_t](#) __n)
- template<typename [_ForwardIterator](#) >
pointer [_M_allocate_and_copy](#) ([size_type](#) __n, [_ForwardIterator](#) __first, [_ForwardIterator](#) __last)
- template<typename [_InputIterator](#) >
void [_M_assign_aux](#) ([_InputIterator](#) __first, [_InputIterator](#) __last, [std::input_iterator_tag](#))
- template<typename [_ForwardIterator](#) >
void [_M_assign_aux](#) ([_ForwardIterator](#) __first, [_ForwardIterator](#) __last, [std::forward_iterator_tag](#))
- template<typename [_Integer](#) >
void [_M_assign_dispatch](#) ([_Integer](#) __n, [_Integer](#) __val, __true_type)
- template<typename [_InputIterator](#) >
void [_M_assign_dispatch](#) ([_InputIterator](#) __first, [_InputIterator](#) __last, __false_type)
- [size_type](#) [_M_check_len](#) ([size_type](#) __n, const char *__s) const
- void [_M_deallocate](#) (pointer __p, [size_t](#) __n)
- void [_M_default_append](#) ([size_type](#) __n)
- void [_M_default_initialize](#) ([size_type](#) __n)
- template<typename... [_Args](#)>
void [_M_emplace_back_aux](#) ([_Args](#) &&...__args)
- iterator [_M_erase](#) (iterator __position)
- iterator [_M_erase](#) (iterator __first, iterator __last)
- void [_M_erase_at_end](#) (pointer __pos) noexcept
- void [_M_fill_assign](#) ([size_type](#) __n, const [value_type](#) &__val)
- void [_M_fill_initialize](#) ([size_type](#) __n, const [value_type](#) &__value)
- void [_M_fill_insert](#) (iterator __pos, [size_type](#) __n, const [value_type](#) &__x)
- [_Tp_alloc_type](#) & [_M_get_Tp_allocator](#) () noexcept
- const [_Tp_alloc_type](#) & [_M_get_Tp_allocator](#) () const noexcept
- template<typename [_Integer](#) >
void [_M_initialize_dispatch](#) ([_Integer](#) __n, [_Integer](#) __value, __true_type)
- template<typename [_InputIterator](#) >
void [_M_initialize_dispatch](#) ([_InputIterator](#) __first, [_InputIterator](#) __last, __false_type)

- `template<typename... _Args>`
`void _M_insert_aux (iterator __position, _Args &&... __args)`
- `template<typename _Integer >`
`void _M_insert_dispatch (iterator __pos, _Integer __n, _Integer __val, __true_type)`
- `template<typename _InputIterator >`
`void _M_insert_dispatch (iterator __pos, _InputIterator __first, _InputIterator __last, __false_type)`
- `void _M_range_check (size_type __n) const`
- `template<typename _InputIterator >`
`void _M_range_initialize (_InputIterator __first, _InputIterator __last, std::input_iterator_tag)`
- `template<typename _ForwardIterator >`
`void _M_range_initialize (_ForwardIterator __first, _ForwardIterator __last, std::forward_iterator_tag)`
- `template<typename _InputIterator >`
`void _M_range_insert (iterator __pos, _InputIterator __first, _InputIterator __last, std::input_iterator_tag)`
- `template<typename _ForwardIterator >`
`void _M_range_insert (iterator __pos, _ForwardIterator __first, _ForwardIterator __last, std::forward_iterator_tag)`
- `bool _M_shrink_to_fit ()`
- `allocator_type get_allocator () const noexcept`

Protected Attributes

- `_Vector_impl _M_impl`

4.711.1 Detailed Description

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
class std::vector< _Tp, _Alloc >
```

A standard container which offers fixed time access to individual elements in any order.

Template Parameters

<code>_Tp</code>	Type of element.
<code>_Alloc</code>	Allocator type, defaults to <code>allocator<_Tp></code> .

Meets the requirements of a [container](#), a [reversible container](#), and a [sequence](#), including the [optional sequence requirements](#) with the exception of `push_front` and `pop_front`.

In some terminology a vector can be described as a dynamic C-style array, it offers fast and efficient access to individual elements in any order and saves the user from worrying about memory and size allocation. Subscripting (`[]`) access is also provided as with C-style arrays.

Definition at line 214 of file `stl_vector.h`.

4.711.2 Constructor & Destructor Documentation

4.711.2.1 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::vector<_Tp, _Alloc>::vector ()`
`[inline], [noexcept]`

Creates a vector with no elements.

Definition at line 253 of file `stl_vector.h`.

4.711.2.2 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::vector<_Tp, _Alloc>::vector (const allocator_type &__a)` `[inline], [explicit], [noexcept]`

Creates a vector with no elements.

Parameters

<code>__a</code>	An allocator object.
------------------	----------------------

Definition at line 264 of file `stl_vector.h`.

4.711.2.3 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::vector<_Tp, _Alloc>::vector (size_type __n, const allocator_type &__a = allocator_type())` `[inline], [explicit]`

Creates a vector with default constructed elements.

Parameters

<code>__n</code>	The number of elements to initially create.
<code>__a</code>	An allocator.

This constructor fills the vector with `__n` default constructed elements.

Definition at line 277 of file `stl_vector.h`.

4.711.2.4 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::vector<_Tp, _Alloc>::vector (size_type __n, const value_type &__value, const allocator_type &__a = allocator_type())` `[inline]`

Creates a vector with copies of an exemplar element.

Parameters

<code>__n</code>	The number of elements to initially create.
<code>__value</code>	An element to copy.
<code>__a</code>	An allocator.

This constructor fills the vector with `__n` copies of `__value`.

Definition at line 289 of file `stl_vector.h`.

4.711.2.5 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::vector<_Tp, _Alloc>::vector (const vector<_Tp, _Alloc> &__x) [inline]`

Vector copy constructor.

Parameters

<code>__x</code>	A vector of identical element and allocator types.
------------------	--

The newly-created vector uses a copy of the allocation object used by `__x`. All the elements of `__x` are copied, but any extra memory in `__x` (for fast expansion) will not be copied.

Definition at line 318 of file `stl_vector.h`.

4.711.2.6 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::vector<_Tp, _Alloc>::vector (vector<_Tp, _Alloc> &&__x) [inline], [noexcept]`

Vector move constructor.

Parameters

<code>__x</code>	A vector of identical element and allocator types.
------------------	--

The newly-created vector contains the exact contents of `__x`. The contents of `__x` are a valid, but unspecified vector.

Definition at line 335 of file `stl_vector.h`.

4.711.2.7 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::vector<_Tp, _Alloc>::vector (const vector<_Tp, _Alloc> &__x, const allocator_type &__a) [inline]`

Copy constructor with alternative allocator.

Definition at line 339 of file `stl_vector.h`.

4.711.2.8 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::vector<_Tp, _Alloc>::vector (vector<_Tp, _Alloc> &&__rv, const allocator_type &__m) [inline], [noexcept]`

Move constructor with alternative allocator.

Definition at line 348 of file `stl_vector.h`.

4.711.2.9 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::vector<_Tp, _Alloc>::vector (initializer_list<value_type> __l, const allocator_type &__a = allocator_type()) [inline]`

Builds a vector from an initializer list.

Parameters

$_l$	An initializer_list.
$_a$	An allocator.

Create a vector consisting of copies of the elements in the initializer_list $_l$.

This will call the element type's copy constructor N times (where N is $_l.size()$) and do no memory reallocation.

Definition at line 373 of file stl_vector.h.

```
4.711.2.10 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> template<typename _InputIterator, typename =
std::_RequireInputIter<_InputIterator>> std::vector<_Tp, _Alloc>::vector ( _InputIterator __first, _InputIterator
__last, const allocator_type & __a = allocator_type() ) [inline]
```

Builds a vector from a range.

Parameters

$_first$	An input iterator.
$_last$	An input iterator.
$_a$	An allocator.

Create a vector consisting of copies of the elements from [first,last).

If the iterators are forward, bidirectional, or random-access, then this will call the elements' copy constructor N times (where N is distance(first,last)) and do no memory reallocation. But if only input iterators are used, then this will do at most 2N calls to the copy constructor, and logN memory reallocations.

Definition at line 401 of file stl_vector.h.

```
4.711.2.11 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::vector<_Tp, _Alloc>::~~vector ( )
[inline], [noexcept]
```

The dtor only erases the elements, and note that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 423 of file stl_vector.h.

4.711.3 Member Function Documentation

```
4.711.3.1 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> template<typename _ForwardIterator> pointer
std::vector<_Tp, _Alloc>::M_allocate_and_copy ( size_type __n, _ForwardIterator __first, _ForwardIterator __last )
[inline], [protected]
```

Memory expansion handler. Uses the member allocation function to obtain n bytes of memory, and then copies [first,last) into it.

Definition at line 1221 of file stl_vector.h.

4.711.3.2 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::vector<_Tp, _Alloc>::_M_range_check (size_type __n) const [inline], [protected]`

Safety check used only from `at()`.

Definition at line 800 of file `stl_vector.h`.

4.711.3.3 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::vector<_Tp, _Alloc>::assign (size_type __n, const value_type & __val) [inline]`

Assigns a given value to a vector.

Parameters

<code>__n</code>	Number of elements to be assigned.
<code>__val</code>	Value to be assigned.

This function fills a vector with `__n` copies of the given value. Note that the assignment completely changes the vector and that the resulting vector's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 488 of file `stl_vector.h`.

4.711.3.4 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>> void std::vector<_Tp, _Alloc>::assign (_InputIterator __first, _InputIterator __last) [inline]`

Assigns a range to a vector.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.

This function fills a vector with copies of the elements in the range `[__first, __last)`.

Note that the assignment completely changes the vector and that the resulting vector's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 507 of file `stl_vector.h`.

4.711.3.5 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::vector<_Tp, _Alloc>::assign (initializer_list<value_type> & __l) [inline]`

Assigns an initializer list to a vector.

Parameters

\leftrightarrow	An initializer_list.
$_ \leftrightarrow$	
\leftarrow	
$_ \leftarrow$	
l	

This function fills a vector with copies of the elements in the initializer list $_l$.

Note that the assignment completely changes the vector and that the resulting vector's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 533 of file stl_vector.h.

4.711.3.6 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> reference std::vector< _Tp, _Alloc >::at (size_type __n) [inline]`

Provides access to the data contained in the vector.

Parameters

$_ \leftrightarrow$	The index of the element for which data should be accessed.
$_n$	

Returns

Read/write reference to data.

Exceptions

<i>std::out_of_range</i>	If $_n$ is an invalid index.
--------------------------	-------------------------------

This function provides for safer data access. The parameter is first checked that it is in the range of the vector. The function throws *out_of_range* if the check fails.

Definition at line 822 of file stl_vector.h.

4.711.3.7 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reference std::vector< _Tp, _Alloc >::at (size_type __n) const [inline]`

Provides access to the data contained in the vector.

Parameters

$_ \leftrightarrow$	The index of the element for which data should be accessed.
$_n$	

Returns

Read-only (constant) reference to data.

Exceptions

<code>std::out_of_range</code>	If <code>__n</code> is an invalid index.
--------------------------------	--

This function provides for safer data access. The parameter is first checked that it is in the range of the vector. The function throws `out_of_range` if the check fails.

Definition at line 840 of file `stl_vector.h`.

4.711.3.8 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> reference std::vector<_Tp, _Alloc>::back ()`
`[inline], [noexcept]`

Returns a read/write reference to the data at the last element of the vector.

Definition at line 867 of file `stl_vector.h`.

Referenced by `std::piecewise_linear_distribution<_RealType>::max()`.

4.711.3.9 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reference std::vector<_Tp, _Alloc>::back () const` `[inline], [noexcept]`

Returns a read-only (constant) reference to the data at the last element of the vector.

Definition at line 875 of file `stl_vector.h`.

4.711.3.10 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> iterator std::vector<_Tp, _Alloc>::begin ()`
`[inline], [noexcept]`

Returns a read/write iterator that points to the first element in the vector. Iteration is done in ordinary element order.

Definition at line 547 of file `stl_vector.h`.

Referenced by `__gnu_parallel::multiseq_partition()`, `__gnu_parallel::multiseq_selection()`, `__gnu_parallel::multiway_↔merge_exact_splitting()`, `std::operator==()`, and `std::vector< sub_match<_Bi_iter>, allocator< sub_match<_Bi_iter>>>::vector()`.

4.711.3.11 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_iterator std::vector<_Tp, _Alloc>::begin () const` `[inline], [noexcept]`

Returns a read-only (constant) iterator that points to the first element in the vector. Iteration is done in ordinary element order.

Definition at line 556 of file `stl_vector.h`.

4.711.3.12 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> size_type std::vector<_Tp, _Alloc>::capacity () const [inline], [noexcept]`

Returns the total number of elements that the vector can hold before needing to allocate more memory.

Definition at line 734 of file `stl_vector.h`.

4.711.3.13 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_iterator std::vector<_Tp, _Alloc>::cbegin () const [inline], [noexcept]`

Returns a read-only (constant) iterator that points to the first element in the vector. Iteration is done in ordinary element order.

Definition at line 620 of file `stl_vector.h`.

4.711.3.14 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_iterator std::vector<_Tp, _Alloc>::cend () const [inline], [noexcept]`

Returns a read-only (constant) iterator that points one past the last element in the vector. Iteration is done in ordinary element order.

Definition at line 629 of file `stl_vector.h`.

4.711.3.15 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::vector<_Tp, _Alloc>::clear () [inline], [noexcept]`

Erases all the elements. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1211 of file `stl_vector.h`.

4.711.3.16 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reverse_iterator std::vector<_Tp, _Alloc>::rbegin () const [inline], [noexcept]`

Returns a read-only (constant) reverse iterator that points to the last element in the vector. Iteration is done in reverse element order.

Definition at line 638 of file `stl_vector.h`.

4.711.3.17 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reverse_iterator std::vector<_Tp, _Alloc>::rend () const [inline], [noexcept]`

Returns a read-only (constant) reverse iterator that points to one before the first element in the vector. Iteration is done in reverse element order.

Definition at line 647 of file `stl_vector.h`.

4.711.3.18 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> _Tp* std::vector<_Tp, _Alloc>::data () [inline], [noexcept]`

Returns a pointer such that `[data(), data() + size())` is a valid range. For a non-empty vector, `data() == &front()`.

Definition at line 890 of file `stl_vector.h`.

Referenced by `std::regex_traits<_Ch_type>::transform_primary()`.

4.711.3.19 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> template<typename... _Args> iterator std::vector<_Tp, _Alloc>::emplace (const_iterator __position, _Args &&... __args)`

Inserts an object in vector before specified iterator.

Parameters

<code>__position</code>	A <code>const_iterator</code> into the vector.
<code>__args</code>	Arguments.

Returns

An iterator that points to the inserted data.

This function will insert an object of type `T` constructed with `T(std::forward<Args>(args)...) before the specified location. Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using std::list.`

4.711.3.20 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> bool std::vector<_Tp, _Alloc>::empty ()`
`const [inline], [noexcept]`

Returns true if the vector is empty. (Thus `begin()` would equal `end()`.)

Definition at line 743 of file `stl_vector.h`.

Referenced by `std::piecewise_linear_distribution<_RealType>::densities()`, `std::piecewise_linear_distribution<_RealType>::intervals()`, `std::piecewise_linear_distribution<_RealType>::max()`, and `std::piecewise_linear_distribution<_RealType>::min()`.

4.711.3.21 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> iterator std::vector<_Tp, _Alloc>::end ()`
`[inline], [noexcept]`

Returns a read/write iterator that points one past the last element in the vector. Iteration is done in ordinary element order.

Definition at line 565 of file `stl_vector.h`.

Referenced by `__gnu_parallel::multiseq_partition()`, `__gnu_parallel::multiseq_selection()`, `__gnu_parallel::multiway_merge_exact_splitting()`, `std::operator==()`, and `std::vector<sub_match<_Bi_iter>, allocator<sub_match<_Bi_iter>>>::vector()`.

4.711.3.22 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_iterator std::vector<_Tp, _Alloc>::end ()`
`const [inline], [noexcept]`

Returns a read-only (constant) iterator that points one past the last element in the vector. Iteration is done in ordinary element order.

Definition at line 574 of file `stl_vector.h`.

4.711.3.23 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> iterator std::vector<_Tp, _Alloc>::erase (`
`const_iterator __position) [inline]`

Remove element at given position.

Parameters

<code>__position</code>	Iterator pointing to element to be erased.
-------------------------	--

Returns

An iterator pointing to the next element (or end()).

This function will erase the element at the given position and thus shorten the vector by one.

Note This operation could be expensive and if it is frequently used the user should consider using `std::list`. The user is also cautioned that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1146 of file `stl_vector.h`.

```
4.711.3.24 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> iterator std::vector< _Tp, _Alloc >::erase (
    const_iterator __first, const_iterator __last ) [inline]
```

Remove a range of elements.

Parameters

<code>__first</code>	Iterator pointing to the first element to be erased.
<code>__last</code>	Iterator pointing to one past the last element to be erased.

Returns

An iterator pointing to the element pointed to by `__last` prior to erasing (or end()).

This function will erase the elements in the range `[__first,__last)` and shorten the vector accordingly.

Note This operation could be expensive and if it is frequently used the user should consider using `std::list`. The user is also cautioned that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1173 of file `stl_vector.h`.

```
4.711.3.25 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> reference std::vector< _Tp, _Alloc >::front ( )
    [inline], [noexcept]
```

Returns a read/write reference to the data at the first element of the vector.

Definition at line 851 of file `stl_vector.h`.

Referenced by `std::piecewise_linear_distribution< _RealType >::min()`.

4.711.3.26 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reference std::vector<_Tp, _Alloc>::front () const [inline], [noexcept]`

Returns a read-only (constant) reference to the data at the first element of the vector.

Definition at line 859 of file `stl_vector.h`.

4.711.3.27 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> iterator std::vector<_Tp, _Alloc>::insert (const_iterator __position, const value_type & __x)`

Inserts given value into vector before specified iterator.

Parameters

<code>__position</code>	A <code>const_iterator</code> into the vector.
<code>__x</code>	Data to be inserted.

Returns

An iterator that points to the inserted data.

This function will insert a copy of the given value before the specified location. Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using `std::list`.

4.711.3.28 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> iterator std::vector<_Tp, _Alloc>::insert (const_iterator __position, value_type && __x) [inline]`

Inserts given rvalue into vector before specified iterator.

Parameters

<code>__position</code>	A <code>const_iterator</code> into the vector.
<code>__x</code>	Data to be inserted.

Returns

An iterator that points to the inserted data.

This function will insert a copy of the given rvalue before the specified location. Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using `std::list`.

Definition at line 1014 of file `stl_vector.h`.

4.711.3.29 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> iterator std::vector<_Tp, _Alloc>::insert (const_iterator __position, initializer_list<value_type> & __l) [inline]`

Inserts an `initializer_list` into the vector.

Parameters

<code>__position</code>	An iterator into the vector.
<code>__l</code>	An initializer_list.

This function will insert copies of the data in the initializer_list *l* into the vector before the location specified by *position*.

Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using `std::list`.

Definition at line 1031 of file `stl_vector.h`.

4.711.3.30 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> iterator std::vector<_Tp, _Alloc>::insert (const_iterator __position, size_type __n, const value_type & __x) [inline]`

Inserts a number of copies of given data into the vector.

Parameters

<code>__position</code>	A const_iterator into the vector.
<code>__n</code>	Number of elements to be inserted.
<code>__x</code>	Data to be inserted.

Returns

An iterator that points to the inserted data.

This function will insert a specified number of copies of the given data before the location specified by *position*.

Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using `std::list`.

Definition at line 1051 of file `stl_vector.h`.

4.711.3.31 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>> iterator std::vector<_Tp, _Alloc>::insert (const_iterator __position, _InputIterator __first, _InputIterator __last) [inline]`

Inserts a range into the vector.

Parameters

<code>__position</code>	A const_iterator into the vector.
<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.

Returns

An iterator that points to the inserted data.

This function will insert copies of the data in the range `[__first,__last)` into the vector before the location specified by *pos*.

Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using `std::list`.

Definition at line 1095 of file `stl_vector.h`.

```
4.711.3.32  template<typename _Tp, typename _Alloc = std::allocator<_Tp>> size_type std::vector<_Tp, _Alloc>::max_size (
            ) const    [inline], [noexcept]
```

Returns the `size()` of the largest possible vector.

Definition at line 659 of file `stl_vector.h`.

```
4.711.3.33  template<typename _Tp, typename _Alloc = std::allocator<_Tp>> vector& std::vector<_Tp, _Alloc>::operator= (
            const vector<_Tp, _Alloc> & __x )
```

Vector assignment operator.

Parameters

<code>__x</code>	A vector of identical element and allocator types.
------------------	--

All the elements of `__x` are copied, but any extra memory in `__x` (for fast expansion) will not be copied. Unlike the copy constructor, the allocator object is not copied.

```
4.711.3.34  template<typename _Tp, typename _Alloc = std::allocator<_Tp>> vector& std::vector<_Tp, _Alloc>::operator= (
            vector<_Tp, _Alloc> && __x )    [inline], [noexcept]
```

Vector move assignment operator.

Parameters

<code>__x</code>	A vector of identical element and allocator types.
------------------	--

The contents of `__x` are moved into this vector (without copying, if the allocators permit it). `__x` is a valid, but unspecified vector.

Definition at line 448 of file `stl_vector.h`.

```
4.711.3.35  template<typename _Tp, typename _Alloc = std::allocator<_Tp>> vector& std::vector<_Tp, _Alloc>::operator= (
            initializer_list< value_type > __l )    [inline]
```

Vector list assignment operator.

Parameters

\leftrightarrow	An initializer_list.
$_ \leftrightarrow$	
\leftrightarrow	
$_ \leftrightarrow$	
l	

This function fills a vector with copies of the elements in the initializer list $_l$.

Note that the assignment completely changes the vector and that the resulting vector's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 470 of file stl_vector.h.

```
4.711.3.36  template<typename _Tp, typename _Alloc = std::allocator<_Tp>> reference std::vector< _Tp, _Alloc >::operator[]
            ( size_type __n )  [inline], [noexcept]
```

Subscript access to the data contained in the vector.

Parameters

$_ \leftrightarrow$	The index of the element for which data should be accessed.
$_n$	

Returns

Read/write reference to data.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and out_of_range lookups are not defined. (For checked lookups see at().)

Definition at line 779 of file stl_vector.h.

```
4.711.3.37  template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reference std::vector< _Tp, _Alloc
            >::operator[]( size_type __n ) const  [inline], [noexcept]
```

Subscript access to the data contained in the vector.

Parameters

$_ \leftrightarrow$	The index of the element for which data should be accessed.
$_n$	

Returns

Read-only (constant) reference to data.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and `out_of_range` lookups are not defined. (For checked lookups see `at()`.)

Definition at line 794 of file `stl_vector.h`.

4.711.3.38 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::vector<_Tp, _Alloc>::pop_back ()`
`[inline], [noexcept]`

Removes last element.

This is a typical stack operation. It shrinks the vector by one.

Note that no data is returned, and if the last element's data is needed, it should be retrieved before `pop_back()` is called.

Definition at line 949 of file `stl_vector.h`.

4.711.3.39 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::vector<_Tp, _Alloc>::push_back (`
`const value_type &__x) [inline]`

Add data to the end of the vector.

Parameters

<code>__x</code>	Data to be added.
------------------	-------------------

This is a typical stack operation. The function creates an element at the end of the vector and assigns the given data to it. Due to the nature of a vector this operation can be done in constant time if the vector has preallocated space available.

Definition at line 913 of file `stl_vector.h`.

Referenced by `__gnu_parallel::multiseq_partition()`, and `__gnu_parallel::multiseq_selection()`.

4.711.3.40 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> reverse_iterator std::vector<_Tp, _Alloc>::rbegin ()`
`[inline], [noexcept]`

Returns a read/write reverse iterator that points to the last element in the vector. Iteration is done in reverse element order.

Definition at line 583 of file `stl_vector.h`.

4.711.3.41 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reverse_iterator std::vector<_Tp, _Alloc>::rbegin () const`
`[inline], [noexcept]`

Returns a read-only (constant) reverse iterator that points to the last element in the vector. Iteration is done in reverse element order.

Definition at line 592 of file `stl_vector.h`.

4.711.3.42 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> reverse_iterator std::vector<_Tp, _Alloc>::rend () [inline], [noexcept]`

Returns a read/write reverse iterator that points to one before the first element in the vector. Iteration is done in reverse element order.

Definition at line 601 of file `stl_vector.h`.

4.711.3.43 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reverse_iterator std::vector<_Tp, _Alloc>::rend () const [inline], [noexcept]`

Returns a read-only (constant) reverse iterator that points to one before the first element in the vector. Iteration is done in reverse element order.

Definition at line 610 of file `stl_vector.h`.

4.711.3.44 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::vector<_Tp, _Alloc>::reserve (size_type __n)`

Attempt to preallocate enough memory for specified number of elements.

Parameters

<code>__n</code>	Number of elements required.
------------------	------------------------------

Exceptions

<code>std::length_error</code>	If <code>n</code> exceeds <code>max_size()</code> .
--------------------------------	---

This function attempts to reserve enough memory for the vector to hold the specified number of elements. If the number requested is more than `max_size()`, `length_error` is thrown.

The advantage of this function is that if optimal code is a necessity and the user can determine the number of elements that will be required, the user can reserve the memory in advance, and thus prevent a possible reallocation of memory and copying of vector data.

4.711.3.45 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::vector<_Tp, _Alloc>::resize (size_type __new_size) [inline]`

Resizes the vector to the specified number of elements.

Parameters

<code>__new_size</code>	Number of elements the vector should contain.
-------------------------	---

This function will resize the vector to the specified number of elements. If the number is smaller than the vector's current size the vector is truncated, otherwise default constructed elements are appended.

Definition at line 673 of file `stl_vector.h`.

Referenced by `__gnu_parallel::__shrink_and_double()`, and `__gnu_parallel::multiway_merge_exact_splitting()`.

4.711.3.46 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::vector<_Tp, _Alloc>::resize (`
`size_type __new_size, const value_type & __x) [inline]`

Resizes the vector to the specified number of elements.

Parameters

<code>__new_size</code>	Number of elements the vector should contain.
<code>__x</code>	Data with which new elements should be populated.

This function will resize the vector to the specified number of elements. If the number is smaller than the vector's current size the vector is truncated, otherwise the vector is extended and new elements are populated with given data.

Definition at line 693 of file `stl_vector.h`.

4.711.3.47 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::vector<_Tp, _Alloc>::shrink_to_fit (`
`) [inline]`

A non-binding request to reduce `capacity()` to `size()`.

Definition at line 725 of file `stl_vector.h`.

4.711.3.48 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> size_type std::vector<_Tp, _Alloc>::size ()`
`const [inline], [noexcept]`

Returns the number of elements in the vector.

Definition at line 654 of file `stl_vector.h`.

Referenced by `__gnu_parallel::__shrink()`, `__gnu_parallel::__shrink_and_double()`, `__gnu_parallel::list_partition()`, `std::operator==()`, and `std::regex_traits<_Ch_type>::transform_primary()`.

4.711.3.49 `template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::vector<_Tp, _Alloc>::swap (`
`vector<_Tp, _Alloc> & __x) [inline], [noexcept]`

Swaps data with another vector.

Parameters

<code>__x</code>	A vector of the same element and allocator types.
------------------	---

This exchanges the elements between two vectors in constant time. (Three pointers, so it should be quite fast.) Note that the global `std::swap()` function is specialized such that `std::swap(v1,v2)` will feed to this function.

Definition at line 1194 of file `stl_vector.h`.

Referenced by `std::swap()`.

The documentation for this class was generated from the following file:

- [stl_vector.h](#)

4.712 `std::vector< bool, _Alloc >` Class Template Reference

Inherits `std::_Bvector_base< _Alloc >`.

Public Types

- typedef `_Alloc` **allocator_type**
- typedef `_Bit_const_iterator` **const_iterator**
- typedef `const bool *` **const_pointer**
- typedef `bool` **const_reference**
- typedef `std::reverse_iterator< const_iterator >` **const_reverse_iterator**
- typedef `ptrdiff_t` **difference_type**
- typedef `_Bit_iterator` **iterator**
- typedef `_Bit_reference *` **pointer**
- typedef `_Bit_reference` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse_iterator**
- typedef `size_t` **size_type**
- typedef `bool` **value_type**

Public Member Functions

- **vector** (`const allocator_type &__a`)
- **vector** (`size_type __n, const allocator_type &__a=allocator_type()`)
- **vector** (`size_type __n, const bool &__value, const allocator_type &__a=allocator_type()`)
- **vector** (`const vector &__x`)
- **vector** (`vector &&__x`) `noexcept`
- **vector** (`initializer_list< bool > __l, const allocator_type &__a=allocator_type()`)
- template<typename `_InputIterator` , typename = `std::_RequireInputIter<_InputIterator>>`
vector (`_InputIterator __first, _InputIterator __last, const allocator_type &__a=allocator_type()`)
- void **assign** (`size_type __n, const bool &__x`)
- template<typename `_InputIterator` , typename = `std::_RequireInputIter<_InputIterator>>`
void **assign** (`_InputIterator __first, _InputIterator __last`)
- void **assign** (`initializer_list< bool > __l`)
- reference **at** (`size_type __n`)
- const_reference **at** (`size_type __n`) `const`
- reference **back** ()
- const_reference **back** () `const`
- iterator **begin** () `noexcept`
- const_iterator **begin** () `const noexcept`

- `size_type capacity ()` const noexcept
- `const_iterator cbegin ()` const noexcept
- `const_iterator cend ()` const noexcept
- `void clear ()` noexcept
- `const_reverse_iterator crbegin ()` const noexcept
- `const_reverse_iterator crend ()` const noexcept
- `void data ()` noexcept
- `template<typename... _Args>`
`iterator emplace (const_iterator __pos, _Args &&... __args)`
- `template<typename... _Args>`
`void emplace_back (_Args &&... __args)`
- `bool empty ()` const noexcept
- `iterator end ()` noexcept
- `const_iterator end ()` const noexcept
- `iterator erase (const_iterator __position)`
- `iterator erase (const_iterator __first, const_iterator __last)`
- `void flip ()` noexcept
- `reference front ()`
- `const_reference front ()` const
- `allocator_type get_allocator ()` const
- `iterator insert (const_iterator __position, const bool &__x=bool())`
- `template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>>`
`iterator insert (const_iterator __position, _InputIterator __first, _InputIterator __last)`
- `iterator insert (const_iterator __position, size_type __n, const bool &__x)`
- `iterator insert (const_iterator __p, initializer_list< bool > __l)`
- `size_type max_size ()` const noexcept
- `vector & operator= (const vector &__x)`
- `vector & operator= (vector &&__x)`
- `vector & operator= (initializer_list< bool > __l)`
- `reference operator[] (size_type __n)`
- `const_reference operator[] (size_type __n)` const
- `void pop_back ()`
- `void push_back (bool __x)`
- `reverse_iterator rbegin ()` noexcept
- `const_reverse_iterator rbegin ()` const noexcept
- `reverse_iterator rend ()` noexcept
- `const_reverse_iterator rend ()` const noexcept
- `void reserve (size_type __n)`
- `void resize (size_type __new_size, bool __x=bool())`
- `void shrink_to_fit ()`
- `size_type size ()` const noexcept
- `void swap (vector &__x)`

Static Public Member Functions

- `static void swap (reference __x, reference __y)` noexcept

Protected Types

- `typedef _Alloc::template rebind< _Bit_type >::other _Bit_alloc_type`

Protected Member Functions

- `_Bit_type * _M_allocate (size_t __n)`
- `template<typename _InputIterator >`
`void _M_assign_aux (_InputIterator __first, _InputIterator __last, std::input_iterator_tag)`
- `template<typename _ForwardIterator >`
`void _M_assign_aux (_ForwardIterator __first, _ForwardIterator __last, std::forward_iterator_tag)`
- `template<typename _Integer >`
`void _M_assign_dispatch (_Integer __n, _Integer __val, __true_type)`
- `template<class _InputIterator >`
`void _M_assign_dispatch (_InputIterator __first, _InputIterator __last, __false_type)`
- `size_type _M_check_len (size_type __n, const char *__s) const`
- `iterator _M_copy_aligned (const_iterator __first, const_iterator __last, iterator __result)`
- `void _M_deallocate ()`
- `iterator _M_erase (iterator __pos)`
- `iterator _M_erase (iterator __first, iterator __last)`
- `void _M_erase_at_end (iterator __pos)`
- `void _M_fill_assign (size_t __n, bool __x)`
- `void _M_fill_insert (iterator __position, size_type __n, bool __x)`
- `_Bit_alloc_type & _M_get_Bit_allocator () noexcept`
- `const _Bit_alloc_type & _M_get_Bit_allocator () const noexcept`
- `void _M_initialize (size_type __n)`
- `template<typename _Integer >`
`void _M_initialize_dispatch (_Integer __n, _Integer __x, __true_type)`
- `template<typename _InputIterator >`
`void _M_initialize_dispatch (_InputIterator __first, _InputIterator __last, __false_type)`
- `template<typename _InputIterator >`
`void _M_initialize_range (_InputIterator __first, _InputIterator __last, std::input_iterator_tag)`
- `template<typename _ForwardIterator >`
`void _M_initialize_range (_ForwardIterator __first, _ForwardIterator __last, std::forward_iterator_tag)`
- `void _M_insert_aux (iterator __position, bool __x)`
- `template<typename _Integer >`
`void _M_insert_dispatch (iterator __pos, _Integer __n, _Integer __x, __true_type)`
- `template<typename _InputIterator >`
`void _M_insert_dispatch (iterator __pos, _InputIterator __first, _InputIterator __last, __false_type)`
- `template<typename _InputIterator >`
`void _M_insert_range (iterator __pos, _InputIterator __first, _InputIterator __last, std::input_iterator_tag)`
- `template<typename _ForwardIterator >`
`void _M_insert_range (iterator __position, _ForwardIterator __first, _ForwardIterator __last, std::forward_iterator_tag)`
- `void _M_range_check (size_type __n) const`
- `void _M_reallocate (size_type __n)`
- `bool _M_shrink_to_fit ()`

Static Protected Member Functions

- `static size_t _S_nword (size_t __n)`

Protected Attributes

- `_Bvector_impl _M_impl`

Friends

- `template<typename >`
`struct hash`

4.712.1 Detailed Description

```
template<typename _Alloc>
class std::vector< bool, _Alloc >
```

A specialization of vector for booleans which offers fixed time access to individual elements in any order.

Template Parameters

<code>_Alloc</code>	Allocator type.
---------------------	-----------------

Note that `vector<bool>` does not actually meet the requirements for being a container. This is because the reference and pointer types are not really references and pointers to `bool`. See DR96 for details.

See also

`vector` for function documentation.

In some terminology a vector can be described as a dynamic C-style array, it offers fast and efficient access to individual elements in any order and saves the user from worrying about memory and size allocation. Subscripting (`[]`) access is also provided as with C-style arrays.

Definition at line 526 of file `stl_bvector.h`.

The documentation for this class was generated from the following file:

- [stl_bvector.h](#)

4.713 `std::weak_ptr<_Tp>` Class Template Reference

Inherits `std::__weak_ptr<_Tp, _Lp>`.

Public Types

- `typedef _Tp element_type`

Public Member Functions

- `template<typename _Tp1, typename = typename std::enable_if<std::is_convertible<_Tp1*, _Tp*>::value>::type> weak_ptr (const weak_ptr<_Tp1> &__r) noexcept`
- `template<typename _Tp1, typename = typename std::enable_if<std::is_convertible<_Tp1*, _Tp*>::value>::type> weak_ptr (const shared_ptr<_Tp1> &__r) noexcept`
- `bool expired () const noexcept`
- `shared_ptr<_Tp> lock () const noexcept`
- `template<typename _Tp1> weak_ptr & operator= (const weak_ptr<_Tp1> &__r) noexcept`
- `template<typename _Tp1> weak_ptr & operator= (const shared_ptr<_Tp1> &__r) noexcept`
- `template<typename _Tp1> bool owner_before (const __shared_ptr<_Tp1, _Lp> &__rhs) const`
- `template<typename _Tp1> bool owner_before (const __weak_ptr<_Tp1, _Lp> &__rhs) const`
- `void reset () noexcept`
- `void swap (__weak_ptr &__s) noexcept`
- `long use_count () const noexcept`

4.713.1 Detailed Description

```
template<typename _Tp>
class std::weak_ptr<_Tp>
```

A smart pointer with weak semantics.

With forwarding constructors and assignment operators.

Definition at line 467 of file `shared_ptr.h`.

The documentation for this class was generated from the following file:

- [shared_ptr.h](#)

4.714 `std::weibull_distribution<_RealType>` Class Template Reference

Classes

- struct [param_type](#)

Public Types

- typedef `_RealType` [result_type](#)

Public Member Functions

- **weibull_distribution** (`_RealType __a=_RealType(1), _RealType __b=_RealType(1)`)
- **weibull_distribution** (`const param_type &__p`)
- `template<typename _ForwardIterator, typename _UniformRandomNumberGenerator >`
`void __generate (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng)`
- `template<typename _ForwardIterator, typename _UniformRandomNumberGenerator >`
`void __generate (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param_type &__p)`
- `template<typename _UniformRandomNumberGenerator >`
`void __generate (result_type *__f, result_type *__t, _UniformRandomNumberGenerator &__urng, const param_type &__p)`
- `_RealType a () const`
- `_RealType b () const`
- `result_type max () const`
- `result_type min () const`
- `template<typename _UniformRandomNumberGenerator >`
`result_type operator() (_UniformRandomNumberGenerator &__urng)`
- `template<typename _UniformRandomNumberGenerator >`
`result_type operator() (_UniformRandomNumberGenerator &__urng, const param_type &__p)`
- `param_type param () const`
- `void param (const param_type &__param)`
- `void reset ()`

Friends

- `bool operator== (const weibull_distribution &__d1, const weibull_distribution &__d2)`

4.714.1 Detailed Description

```
template<typename _RealType = double>
class std::weibull_distribution< _RealType >
```

A weibull_distribution random number distribution.

The formula for the normal probability density function is:

$$p(x|\alpha, \beta) = \frac{\alpha}{\beta} \left(\frac{x}{\beta}\right)^{\alpha-1} \exp\left(-\left(\frac{x}{\beta}\right)^{\alpha}\right)$$

Definition at line 4850 of file random.h.

4.714.2 Member Typedef Documentation

4.714.2.1 `template<typename _RealType = double> typedef _RealType std::weibull_distribution< _RealType >::result_type`

The type of the range of the distribution.

Definition at line 4853 of file random.h.

4.714.3 Member Function Documentation

4.714.3.1 `template<typename _RealType = double> _RealType std::weibull_distribution<_RealType>::a () const`
[inline]

Return the a parameter of the distribution.

Definition at line 4908 of file random.h.

4.714.3.2 `template<typename _RealType = double> _RealType std::weibull_distribution<_RealType>::b () const`
[inline]

Return the b parameter of the distribution.

Definition at line 4915 of file random.h.

4.714.3.3 `template<typename _RealType = double> result_type std::weibull_distribution<_RealType>::max () const`
[inline]

Returns the least upper bound value of the distribution.

Definition at line 4944 of file random.h.

References `std::max()`.

4.714.3.4 `template<typename _RealType = double> result_type std::weibull_distribution<_RealType>::min () const`
[inline]

Returns the greatest lower bound value of the distribution.

Definition at line 4937 of file random.h.

4.714.3.5 `template<typename _RealType = double> template<typename _UniformRandomNumberGenerator> result_type
std::weibull_distribution<_RealType>::operator() (_UniformRandomNumberGenerator &__urng)` [inline]

Generating functions.

Definition at line 4952 of file random.h.

4.714.3.6 `template<typename _RealType = double> param_type std::weibull_distribution<_RealType>::param () const`
[inline]

Returns the parameter set of the distribution.

Definition at line 4922 of file random.h.

4.714.3.7 `template<typename _RealType = double> void std::weibull_distribution<_RealType>::param (const
param_type &__param)` [inline]

Sets the parameter set of the distribution.

Parameters

<code>__param</code>	The new parameter set of the distribution.
----------------------	--

Definition at line 4930 of file random.h.

4.714.3.8 `template<typename _RealType = double> void std::weibull_distribution<_RealType>::reset() [inline]`

Resets the distribution state.

Definition at line 4901 of file random.h.

4.714.4 Friends And Related Function Documentation

4.714.4.1 `template<typename _RealType = double> bool operator==(const weibull_distribution<_RealType> &__d1, const weibull_distribution<_RealType> &__d2) [friend]`

Return true if two Weibull distributions have the same parameters.

Definition at line 4987 of file random.h.

The documentation for this class was generated from the following file:

- [random.h](#)

4.715 `std::weibull_distribution<_RealType>::param_type` Struct Reference

Public Types

- typedef `weibull_distribution<_RealType>` **distribution_type**

Public Member Functions

- **param_type** (`_RealType __a=_RealType(1), _RealType __b=_RealType(1)`)
- `_RealType a` () const
- `_RealType b` () const

Friends

- bool **operator==** (const `param_type` &__p1, const `param_type` &__p2)

4.715.1 Detailed Description

```
template<typename _RealType = double>
struct std::weibull_distribution< _RealType >::param_type
```

Parameter type.

Definition at line 4859 of file random.h.

The documentation for this struct was generated from the following file:

- [random.h](#)

5 File Documentation

5.1 algo.h File Reference

Classes

- struct [std::__parallel::__CRandNumber< _MustBeInt >](#)

Namespaces

- [std](#)
- [std::__parallel](#)

Functions

- template<typename _RAIter >
_RAIter **std::__parallel::__adjacent_find_switch** (_RAIter __begin, _RAIter __end, random_access_iterator↔
_tag)
- template<typename _Filterator , typename _IteratorTag >
_Filterator **std::__parallel::__adjacent_find_switch** (_Filterator __begin, _Filterator __end, _IteratorTag)
- template<typename _Filterator , typename _BinaryPredicate , typename _IteratorTag >
_Filterator **std::__parallel::__adjacent_find_switch** (_Filterator __begin, _Filterator __end, _BinaryPredicate ↔
__pred, _IteratorTag)
- template<typename _RAIter , typename _BinaryPredicate >
_RAIter **std::__parallel::__adjacent_find_switch** (_RAIter __begin, _RAIter __end, _BinaryPredicate __pred,
random_access_iterator_tag)
- template<typename _RAIter , typename _Predicate >
iterator_traits< _RAIter >::difference_type **std::__parallel::__count_if_switch** (_RAIter __begin, _RAIter ↔
__end, _Predicate __pred, random_access_iterator_tag, [__gnu_parallel::__Parallelism](#) __parallelism_tag)
- template<typename _Iter , typename _Predicate , typename _IteratorTag >
iterator_traits< _Iter >::difference_type **std::__parallel::__count_if_switch** (_Iter __begin, _Iter __end, ↔
_Predicate __pred, _IteratorTag)

- `template<typename _RAIter, typename _Tp >`
`iterator_traits< _RAIter >::difference_type std::__parallel::__count_switch (_RAIter __begin, _RAIter __end,`
`const _Tp &__value, random_access_iterator_tag, __gnu_parallel::__Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Tp, typename _IteratorTag >`
`iterator_traits< _Iter >::difference_type std::__parallel::__count_switch (_Iter __begin, _Iter __end, const`
`_Tp &__value, _IteratorTag)`
- `template<typename _Iter, typename _FIterator, typename _IteratorTag1, typename _IteratorTag2 >`
`_Iter std::__parallel::__find_first_of_switch (_Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator`
`__end2, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter, typename _FIterator, typename _BinaryPredicate, typename _IteratorTag >`
`_RAIter std::__parallel::__find_first_of_switch (_RAIter __begin1, _RAIter __end1, _FIterator __begin2, _F`
`Iterator __end2, _BinaryPredicate __comp, random_access_iterator_tag, _IteratorTag)`
- `template<typename _Iter, typename _FIterator, typename _BinaryPredicate, typename _IteratorTag1, typename _IteratorTag2 >`
`_Iter std::__parallel::__find_first_of_switch (_Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator`
`__end2, _BinaryPredicate __comp, _IteratorTag1, _IteratorTag2)`
- `template<typename _Iter, typename _Predicate, typename _IteratorTag >`
`_Iter std::__parallel::__find_if_switch (_Iter __begin, _Iter __end, _Predicate __pred, _IteratorTag)`
- `template<typename _RAIter, typename _Predicate >`
`_RAIter std::__parallel::__find_if_switch (_RAIter __begin, _RAIter __end, _Predicate __pred, random_`
`access_iterator_tag)`
- `template<typename _Iter, typename _Tp, typename _IteratorTag >`
`_Iter std::__parallel::__find_switch (_Iter __begin, _Iter __end, const _Tp &__val, _IteratorTag)`
- `template<typename _RAIter, typename _Tp >`
`_RAIter std::__parallel::__find_switch (_RAIter __begin, _RAIter __end, const _Tp &__val, random_access_`
`iterator_tag)`
- `template<typename _Iter, typename _Function, typename _IteratorTag >`
`_Function std::__parallel::__for_each_switch (_Iter __begin, _Iter __end, _Function __f, _IteratorTag)`
- `template<typename _RAIter, typename _Function >`
`_Function std::__parallel::__for_each_switch (_RAIter __begin, _RAIter __end, _Function __f, random_`
`access_iterator_tag, __gnu_parallel::__Parallelism __parallelism_tag)`
- `template<typename _OutputIterator, typename _Size, typename _Generator, typename _IteratorTag >`
`_OutputIterator std::__parallel::__generate_n_switch (_OutputIterator __begin, _Size __n, _Generator __gen,`
`_IteratorTag)`
- `template<typename _RAIter, typename _Size, typename _Generator >`
`_RAIter std::__parallel::__generate_n_switch (_RAIter __begin, _Size __n, _Generator __gen, random_`
`access_iterator_tag, __gnu_parallel::__Parallelism __parallelism_tag)`
- `template<typename _FIterator, typename _Generator, typename _IteratorTag >`
`void std::__parallel::__generate_switch (_FIterator __begin, _FIterator __end, _Generator __gen, _IteratorTag)`
- `template<typename _RAIter, typename _Generator >`
`void std::__parallel::__generate_switch (_RAIter __begin, _RAIter __end, _Generator __gen, random_`
`access_iterator_tag, __gnu_parallel::__Parallelism __parallelism_tag)`
- `template<typename _FIterator, typename _Compare, typename _IteratorTag >`
`_FIterator std::__parallel::__max_element_switch (_FIterator __begin, _FIterator __end, _Compare __comp,`
`_IteratorTag)`
- `template<typename _RAIter, typename _Compare >`
`_RAIter std::__parallel::__max_element_switch (_RAIter __begin, _RAIter __end, _Compare __comp,`
`random_access_iterator_tag, __gnu_parallel::__Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Compare, typename _IteratorTag1, typename _`
`IteratorTag2, typename _IteratorTag3 >`
`_OutputIterator std::__parallel::__merge_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2`
`__end2, _OutputIterator __result, _Compare __comp, _IteratorTag1, _IteratorTag2, _IteratorTag3)`

- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Compare >`
`_OutputIterator std::parallel::merge_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __result, _Compare __comp, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Filterator, typename _Compare, typename _IteratorTag >`
`_Filterator std::parallel::min_element_switch (_Filterator __begin, _Filterator __end, _Compare __comp, _IteratorTag)`
- `template<typename _RAIter, typename _Compare >`
`_RAIter std::parallel::min_element_switch (_RAIter __begin, _RAIter __end, _Compare __comp, random_access_iterator_tag, gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Filterator, typename _Predicate, typename _IteratorTag >`
`_Filterator std::parallel::partition_switch (_Filterator __begin, _Filterator __end, _Predicate __pred, _IteratorTag)`
- `template<typename _RAIter, typename _Predicate >`
`_RAIter std::parallel::partition_switch (_RAIter __begin, _RAIter __end, _Predicate __pred, random_access_iterator_tag)`
- `template<typename _Filterator, typename _Predicate, typename _Tp, typename _IteratorTag >`
`void std::parallel::replace_if_switch (_Filterator __begin, _Filterator __end, _Predicate __pred, const _Tp & __new_value, _IteratorTag)`
- `template<typename _RAIter, typename _Predicate, typename _Tp >`
`void std::parallel::replace_if_switch (_RAIter __begin, _RAIter __end, _Predicate __pred, const _Tp & __new_value, random_access_iterator_tag, gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Filterator, typename _Tp, typename _IteratorTag >`
`void std::parallel::replace_switch (_Filterator __begin, _Filterator __end, const _Tp & __old_value, const _Tp & __new_value, _IteratorTag)`
- `template<typename _RAIter, typename _Tp >`
`void std::parallel::replace_switch (_RAIter __begin, _RAIter __end, const _Tp & __old_value, const _Tp & __new_value, random_access_iterator_tag, gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _RAIter, typename _Integer, typename _Tp, typename _BinaryPredicate >`
`_RAIter std::parallel::search_n_switch (_RAIter __begin, _RAIter __end, _Integer __count, const _Tp & __val, _BinaryPredicate __binary_pred, random_access_iterator_tag)`
- `template<typename _Filterator, typename _Integer, typename _Tp, typename _BinaryPredicate, typename _IteratorTag >`
`_Filterator std::parallel::search_n_switch (_Filterator __begin, _Filterator __end, _Integer __count, const _Tp & __val, _BinaryPredicate __binary_pred, _IteratorTag)`
- `template<typename _RAIter1, typename _RAIter2 >`
`_RAIter1 std::parallel::search_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Filterator1, typename _Filterator2, typename _IteratorTag1, typename _IteratorTag2 >`
`_Filterator1 std::parallel::search_switch (_Filterator1 __begin1, _Filterator1 __end1, _Filterator2 __begin2, _Filterator2 __end2, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _BinaryPredicate >`
`_RAIter1 std::parallel::search_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _BinaryPredicate __pred, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Filterator1, typename _Filterator2, typename _BinaryPredicate, typename _IteratorTag1, typename _IteratorTag2 >`
`_Filterator1 std::parallel::search_switch (_Filterator1 __begin1, _Filterator1 __end1, _Filterator2 __begin2, _Filterator2 __end2, _BinaryPredicate __pred, _IteratorTag1, _IteratorTag2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3 >`
`_OutputIterator std::parallel::set_difference_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAIter, typename _Predicate >`
`_Output_RAIter std::parallel::set_difference_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAIter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)`

- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3 >`
`_OutputIterator std::parallel::set_intersection_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAlter, typename _Predicate >`
`_Output_RAlter std::parallel::set_intersection_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAlter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3 >`
`_OutputIterator std::parallel::set_symmetric_difference_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAlter, typename _Predicate >`
`_Output_RAlter std::parallel::set_symmetric_difference_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAlter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3 >`
`_OutputIterator std::parallel::set_union_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAlter, typename _Predicate >`
`_Output_RAlter std::parallel::set_union_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAlter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _RAIter1, typename _RAIter2, typename _UnaryOperation >`
`_RAIter2 std::parallel::transform1_switch (_RAIter1 __begin, _RAIter1 __end, _RAIter2 __result, _UnaryOperation __unary_op, random_access_iterator_tag, random_access_iterator_tag, gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _RAIter1, typename _RAIter2, typename _UnaryOperation, typename _IteratorTag1, typename _IteratorTag2 >`
`_RAIter2 std::parallel::transform1_switch (_RAIter1 __begin, _RAIter1 __end, _RAIter2 __result, _UnaryOperation __unary_op, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _RAIter3, typename _BinaryOperation >`
`_RAIter3 std::parallel::transform2_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter3 __result, _BinaryOperation __binary_op, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag, gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _BinaryOperation, typename _Tag1, typename _Tag2, typename _Tag3 >`
`_OutputIterator std::parallel::transform2_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _OutputIterator __result, _BinaryOperation __binary_op, _Tag1, _Tag2, _Tag3)`
- `template<typename _Iter, typename _OutputIterator, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2 >`
`_OutputIterator std::parallel::unique_copy_switch (_Iter __begin, _Iter __last, _OutputIterator __out, _Predicate __pred, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter, typename RandomAccessOutputIterator, typename _Predicate >`
`RandomAccessOutputIterator std::parallel::unique_copy_switch (_RAIter __begin, _RAIter __last, RandomAccessOutputIterator __out, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _FIterator >`
`_FIterator std::parallel::adjacent_find (_FIterator __begin, _FIterator __end, gnu_parallel::sequential_tag)`
- `template<typename _FIterator, typename _BinaryPredicate >`
`_FIterator std::parallel::adjacent_find (_FIterator __begin, _FIterator __end, _BinaryPredicate __binary_pred, gnu_parallel::sequential_tag)`
- `template<typename _FIterator >`
`_FIterator std::parallel::adjacent_find (_FIterator __begin, _FIterator __end)`

- `template<typename _FIterator, typename _BinaryPredicate >`
`_FIterator std::parallel::adjacent_find (_FIterator __begin, _FIterator __end, _BinaryPredicate __pred)`
- `template<typename _Iter, typename _Tp >`
`iterator_traits< _Iter >::difference_type std::parallel::count (_Iter __begin, _Iter __end, const _Tp &__↵`
`value, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _Tp >`
`iterator_traits< _Iter >::difference_type std::parallel::count (_Iter __begin, _Iter __end, const _Tp &__↵`
`value, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Tp >`
`iterator_traits< _Iter >::difference_type std::parallel::count (_Iter __begin, _Iter __end, const _Tp &__↵`
`value)`
- `template<typename _Iter, typename _Predicate >`
`iterator_traits< _Iter >::difference_type std::parallel::count_if (_Iter __begin, _Iter __end, _Predicate __↵`
`pred, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _Predicate >`
`iterator_traits< _Iter >::difference_type std::parallel::count_if (_Iter __begin, _Iter __end, _Predicate __↵`
`pred, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Predicate >`
`iterator_traits< _Iter >::difference_type std::parallel::count_if (_Iter __begin, _Iter __end, _Predicate __↵`
`pred)`
- `template<typename _Iter, typename _Tp >`
`_Iter std::parallel::find (_Iter __begin, _Iter __end, const _Tp &__val, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _Tp >`
`_Iter std::parallel::find (_Iter __begin, _Iter __end, const _Tp &__val)`
- `template<typename _Iter, typename _FIterator >`
`_Iter std::parallel::find_first_of (_Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator __end2, __↵`
`__gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _FIterator, typename _BinaryPredicate >`
`_Iter std::parallel::find_first_of (_Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator __end2, __↵`
`_BinaryPredicate __comp, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _FIterator, typename _BinaryPredicate >`
`_Iter std::parallel::find_first_of (_Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator __end2, __↵`
`_BinaryPredicate __comp)`
- `template<typename _Iter, typename _FIterator >`
`_Iter std::parallel::find_first_of (_Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator __end2)`
- `template<typename _Iter, typename _Predicate >`
`_Iter std::parallel::find_if (_Iter __begin, _Iter __end, _Predicate __pred, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _Predicate >`
`_Iter std::parallel::find_if (_Iter __begin, _Iter __end, _Predicate __pred)`
- `template<typename _Iter, typename _Function >`
`_Function std::parallel::for_each (_Iter __begin, _Iter __end, _Function __f, __gnu_parallel::sequential_tag)`
- `template<typename _Iterator, typename _Function >`
`_Function std::parallel::for_each (_Iterator __begin, _Iterator __end, _Function __f, __gnu_parallel::↵`
`Parallelism __parallelism_tag)`
- `template<typename _Iterator, typename _Function >`
`_Function std::parallel::for_each (_Iterator __begin, _Iterator __end, _Function __f)`
- `template<typename _FIterator, typename _Generator >`
`void std::parallel::generate (_FIterator __begin, _FIterator __end, _Generator __gen, __gnu_parallel::↵`
`sequential_tag)`
- `template<typename _FIterator, typename _Generator >`
`void std::parallel::generate (_FIterator __begin, _FIterator __end, _Generator __gen, __gnu_parallel::↵`
`Parallelism __parallelism_tag)`

- `template<typename _Filterator , typename _Generator >`
`void std::__parallel::generate (_Filterator __begin, _Filterator __end, _Generator __gen)`
- `template<typename _Outputlterator , typename _Size , typename _Generator >`
`_Outputlterator std::__parallel::generate_n (_Outputlterator __begin, _Size __n, _Generator __gen, __gnu_parallel::sequential_tag)`
- `template<typename _Outputlterator , typename _Size , typename _Generator >`
`_Outputlterator std::__parallel::generate_n (_Outputlterator __begin, _Size __n, _Generator __gen, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Outputlterator , typename _Size , typename _Generator >`
`_Outputlterator std::__parallel::generate_n (_Outputlterator __begin, _Size __n, _Generator __gen)`
- `template<typename _Filterator >`
`_Filterator std::__parallel::max_element (_Filterator __begin, _Filterator __end, __gnu_parallel::sequential_tag)`
- `template<typename _Filterator , typename _Compare >`
`_Filterator std::__parallel::max_element (_Filterator __begin, _Filterator __end, _Compare __comp, __gnu_parallel::sequential_tag)`
- `template<typename _Filterator >`
`_Filterator std::__parallel::max_element (_Filterator __begin, _Filterator __end, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Filterator >`
`_Filterator std::__parallel::max_element (_Filterator __begin, _Filterator __end)`
- `template<typename _Filterator , typename _Compare >`
`_Filterator std::__parallel::max_element (_Filterator __begin, _Filterator __end, _Compare __comp, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Filterator , typename _Compare >`
`_Filterator std::__parallel::max_element (_Filterator __begin, _Filterator __end, _Compare __comp)`
- `template<typename _lter1 , typename _lter2 , typename _Outputlterator >`
`_Outputlterator std::__parallel::merge (_lter1 __begin1, _lter1 __end1, _lter2 __begin2, _lter2 __end2, __gnu_parallel::sequential_tag)`
- `template<typename _lter1 , typename _lter2 , typename _Outputlterator , typename _Compare >`
`_Outputlterator std::__parallel::merge (_lter1 __begin1, _lter1 __end1, _lter2 __begin2, _lter2 __end2, __gnu_parallel::sequential_tag)`
- `template<typename _lter1 , typename _lter2 , typename _Outputlterator , typename _Compare >`
`_Outputlterator std::__parallel::merge (_lter1 __begin1, _lter1 __end1, _lter2 __begin2, _lter2 __end2, __gnu_parallel::sequential_tag)`
- `template<typename _lter1 , typename _lter2 , typename _Outputlterator >`
`_Outputlterator std::__parallel::merge (_lter1 __begin1, _lter1 __end1, _lter2 __begin2, _lter2 __end2, __gnu_parallel::sequential_tag)`
- `template<typename _Filterator >`
`_Filterator std::__parallel::min_element (_Filterator __begin, _Filterator __end, __gnu_parallel::sequential_tag)`
- `template<typename _Filterator , typename _Compare >`
`_Filterator std::__parallel::min_element (_Filterator __begin, _Filterator __end, _Compare __comp, __gnu_parallel::sequential_tag)`
- `template<typename _Filterator >`
`_Filterator std::__parallel::min_element (_Filterator __begin, _Filterator __end, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Filterator >`
`_Filterator std::__parallel::min_element (_Filterator __begin, _Filterator __end)`
- `template<typename _Filterator , typename _Compare >`
`_Filterator std::__parallel::min_element (_Filterator __begin, _Filterator __end, _Compare __comp, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Filterator , typename _Compare >`
`_Filterator std::__parallel::min_element (_Filterator __begin, _Filterator __end, _Compare __comp)`

- `template<typename _RAIter >`
`void std::__parallel::nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end, __gnu_parallel::sequential_tag)`
- `template<typename _RAIter, typename _Compare >`
`void std::__parallel::nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end, _Compare __comp, __gnu_parallel::sequential_tag)`
- `template<typename _RAIter, typename _Compare >`
`void std::__parallel::nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter >`
`void std::__parallel::nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end)`
- `template<typename _RAIter, typename _Compare >`
`void std::__parallel::partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end, _Compare __comp, __gnu_parallel::sequential_tag)`
- `template<typename _RAIter >`
`void std::__parallel::partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end, __gnu_parallel::sequential_tag)`
- `template<typename _RAIter, typename _Compare >`
`void std::__parallel::partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter >`
`void std::__parallel::partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end)`
- `template<typename _Filterator, typename _Predicate >`
`_Filterator std::__parallel::partition (_Filterator __begin, _Filterator __end, _Predicate __pred, __gnu_parallel::sequential_tag)`
- `template<typename _Filterator, typename _Predicate >`
`_Filterator std::__parallel::partition (_Filterator __begin, _Filterator __end, _Predicate __pred)`
- `template<typename _RAIter >`
`void std::__parallel::random_shuffle (_RAIter __begin, _RAIter __end, __gnu_parallel::sequential_tag)`
- `template<typename _RAIter, typename _RandomNumberGenerator >`
`void std::__parallel::random_shuffle (_RAIter __begin, _RAIter __end, _RandomNumberGenerator &__rand, __gnu_parallel::sequential_tag)`
- `template<typename _RAIter >`
`void std::__parallel::random_shuffle (_RAIter __begin, _RAIter __end)`
- `template<typename _RAIter, typename _RandomNumberGenerator >`
`void std::__parallel::random_shuffle (_RAIter __begin, _RAIter __end, _RandomNumberGenerator &&__rand)`
- `template<typename _Filterator, typename _Tp >`
`void std::__parallel::replace (_Filterator __begin, _Filterator __end, const _Tp &__old_value, const _Tp &__new_value, __gnu_parallel::sequential_tag)`
- `template<typename _Filterator, typename _Tp >`
`void std::__parallel::replace (_Filterator __begin, _Filterator __end, const _Tp &__old_value, const _Tp &__new_value, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Filterator, typename _Tp >`
`void std::__parallel::replace (_Filterator __begin, _Filterator __end, const _Tp &__old_value, const _Tp &__new_value)`
- `template<typename _Filterator, typename _Predicate, typename _Tp >`
`void std::__parallel::replace_if (_Filterator __begin, _Filterator __end, _Predicate __pred, const _Tp &__new_value, __gnu_parallel::sequential_tag)`
- `template<typename _Filterator, typename _Predicate, typename _Tp >`
`void std::__parallel::replace_if (_Filterator __begin, _Filterator __end, _Predicate __pred, const _Tp &__new_value, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Filterator, typename _Predicate, typename _Tp >`
`void std::__parallel::replace_if (_Filterator __begin, _Filterator __end, _Predicate __pred, const _Tp &__new_value)`

- `template<typename _FIterator1, typename _FIterator2 >`
`_FIterator1 std::parallel::search (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2`
`__end2, __gnu_parallel::sequential_tag)`
- `template<typename _FIterator1, typename _FIterator2 >`
`_FIterator1 std::parallel::search (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2`
`__end2)`
- `template<typename _FIterator1, typename _FIterator2, typename _BinaryPredicate >`
`_FIterator1 std::parallel::search (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2`
`__end2, _BinaryPredicate __pred, __gnu_parallel::sequential_tag)`
- `template<typename _FIterator1, typename _FIterator2, typename _BinaryPredicate >`
`_FIterator1 std::parallel::search (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2`
`__end2, _BinaryPredicate __pred)`
- `template<typename _FIterator, typename _Integer, typename _Tp >`
`_FIterator std::parallel::search_n (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp &__val,`
`__gnu_parallel::sequential_tag)`
- `template<typename _FIterator, typename _Integer, typename _Tp, typename _BinaryPredicate >`
`_FIterator std::parallel::search_n (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp &__val,`
`_BinaryPredicate __binary_pred, __gnu_parallel::sequential_tag)`
- `template<typename _FIterator, typename _Integer, typename _Tp >`
`_FIterator std::parallel::search_n (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp &__val)`
- `template<typename _FIterator, typename _Integer, typename _Tp, typename _BinaryPredicate >`
`_FIterator std::parallel::search_n (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp &__val,`
`_BinaryPredicate __binary_pred)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator >`
`_OutputIterator std::parallel::set_difference (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __↵`
`end2, _OutputIterator __out, __gnu_parallel::sequential_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Predicate >`
`_OutputIterator std::parallel::set_difference (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __↵`
`end2, _OutputIterator __out, _Predicate __pred, __gnu_parallel::sequential_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator >`
`_OutputIterator std::parallel::set_difference (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __↵`
`end2, _OutputIterator __out)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Predicate >`
`_OutputIterator std::parallel::set_difference (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __↵`
`end2, _OutputIterator __out, _Predicate __pred)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator >`
`_OutputIterator std::parallel::set_intersection (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 ↵`
`__end2, _OutputIterator __out, __gnu_parallel::sequential_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Predicate >`
`_OutputIterator std::parallel::set_intersection (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 ↵`
`__end2, _OutputIterator __out, _Predicate __pred, __gnu_parallel::sequential_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator >`
`_OutputIterator std::parallel::set_intersection (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 ↵`
`__end2, _OutputIterator __out)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Predicate >`
`_OutputIterator std::parallel::set_intersection (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 ↵`
`__end2, _OutputIterator __out, _Predicate __pred)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator >`
`_OutputIterator std::parallel::set_symmetric_difference (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2,`
`_IIter2 __end2, _OutputIterator __out, __gnu_parallel::sequential_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Predicate >`
`_OutputIterator std::parallel::set_symmetric_difference (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2,`
`_IIter2 __end2, _OutputIterator __out, _Predicate __pred, __gnu_parallel::sequential_tag)`

- `template<typename _Iter1, typename _Iter2, typename _OutputIterator >`
`_OutputIterator std::parallel::set_symmetric_difference (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2,`
`_Iter2 __end2, _OutputIterator __out)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate >`
`_OutputIterator std::parallel::set_symmetric_difference (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2,`
`_Iter2 __end2, _OutputIterator __out, _Predicate __pred)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator >`
`_OutputIterator std::parallel::set_union (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2,`
`_OutputIterator __out, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate >`
`_OutputIterator std::parallel::set_union (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2,`
`_OutputIterator __out, _Predicate __pred, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator >`
`_OutputIterator std::parallel::set_union (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2,`
`_OutputIterator __out)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate >`
`_OutputIterator std::parallel::set_union (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2,`
`_OutputIterator __out, _Predicate __pred)`
- `template<typename _RAIter >`
`void std::parallel::sort (_RAIter __begin, _RAIter __end, __gnu_parallel::sequential_tag)`
- `template<typename _RAIter, typename _Compare >`
`void std::parallel::sort (_RAIter __begin, _RAIter __end, _Compare __comp, __gnu_parallel::sequential_tag)`
- `template<typename _RAIter, typename _Compare, typename _Parallelism >`
`void std::parallel::sort (_RAIter __begin, _RAIter __end, _Compare __comp, _Parallelism __parallelism)`
- `template<typename _RAIter >`
`void std::parallel::sort (_RAIter __begin, _RAIter __end)`
- `template<typename _RAIter >`
`void std::parallel::sort (_RAIter __begin, _RAIter __end, __gnu_parallel::default_parallel_tag __parallelism)`
- `template<typename _RAIter >`
`void std::parallel::sort (_RAIter __begin, _RAIter __end, __gnu_parallel::parallel_tag __parallelism)`
- `template<typename _RAIter >`
`void std::parallel::sort (_RAIter __begin, _RAIter __end, __gnu_parallel::multiway_mergesort_tag __parallelism)`
- `template<typename _RAIter >`
`void std::parallel::sort (_RAIter __begin, _RAIter __end, __gnu_parallel::multiway_mergesort_sampling_tag __parallelism)`
- `template<typename _RAIter >`
`void std::parallel::sort (_RAIter __begin, _RAIter __end, __gnu_parallel::multiway_mergesort_exact_tag __parallelism)`
- `template<typename _RAIter >`
`void std::parallel::sort (_RAIter __begin, _RAIter __end, __gnu_parallel::quicksort_tag __parallelism)`
- `template<typename _RAIter >`
`void std::parallel::sort (_RAIter __begin, _RAIter __end, __gnu_parallel::balanced_quicksort_tag __parallelism)`
- `template<typename _RAIter, typename _Compare >`
`void std::parallel::sort (_RAIter __begin, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter >`
`void std::parallel::stable_sort (_RAIter __begin, _RAIter __end, __gnu_parallel::sequential_tag)`
- `template<typename _RAIter, typename _Compare >`
`void std::parallel::stable_sort (_RAIter __begin, _RAIter __end, _Compare __comp, __gnu_parallel::sequential_tag)`

- `template<typename _RAIter, typename _Compare, typename _Parallelism >`
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, _Compare __comp, _Parallelism __↵`
`parallelism)`
- `template<typename _RAIter >`
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end)`
- `template<typename _RAIter >`
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, __gnu_parallel::default_parallel_tag __↵`
`parallelism)`
- `template<typename _RAIter >`
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, __gnu_parallel::parallel_tag __parallelism)`
- `template<typename _RAIter >`
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, __gnu_parallel::multiway_mergesort_tag ↵`
`__parallelism)`
- `template<typename _RAIter >`
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, __gnu_parallel::quicksort_tag __parallelism)`
- `template<typename _RAIter >`
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, __gnu_parallel::balanced_quicksort_tag ↵`
`__parallelism)`
- `template<typename _RAIter, typename _Compare >`
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, _Compare __comp)`
- `template<typename _Iter, typename _OutputIterator, typename _UnaryOperation >`
`_OutputIterator std::__parallel::transform (_Iter __begin, _Iter __end, _OutputIterator __result, _Unary↵`
`Operation __unary_op, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _UnaryOperation >`
`_OutputIterator std::__parallel::transform (_Iter __begin, _Iter __end, _OutputIterator __result, _Unary↵`
`Operation __unary_op, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _UnaryOperation >`
`_OutputIterator std::__parallel::transform (_Iter __begin, _Iter __end, _OutputIterator __result, _Unary↵`
`Operation __unary_op)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _BinaryOperation >`
`_OutputIterator std::__parallel::transform (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _OutputIterator`
`__result, _BinaryOperation __binary_op, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _BinaryOperation >`
`_OutputIterator std::__parallel::transform (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _OutputIterator`
`__result, _BinaryOperation __binary_op, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _BinaryOperation >`
`_OutputIterator std::__parallel::transform (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _OutputIterator`
`__result, _BinaryOperation __binary_op)`
- `template<typename _Iter, typename _OutputIterator >`
`_OutputIterator std::__parallel::unique_copy (_Iter __begin1, _Iter __end1, _OutputIterator __out, __gnu_↵`
`parallel::sequential_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _Predicate >`
`_OutputIterator std::__parallel::unique_copy (_Iter __begin1, _Iter __end1, _OutputIterator __out, _Predicate`
`__pred, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _OutputIterator >`
`_OutputIterator std::__parallel::unique_copy (_Iter __begin1, _Iter __end1, _OutputIterator __out)`
- `template<typename _Iter, typename _OutputIterator, typename _Predicate >`
`_OutputIterator std::__parallel::unique_copy (_Iter __begin1, _Iter __end1, _OutputIterator __out, _Predicate`
`__pred)`

5.1.1 Detailed Description

Parallel STL function calls corresponding to the `stl_algo.h` header.

The functions defined here mainly do case switches and call the actual parallelized versions in other files. Inlining policy: Functions that basically only contain one function call, are declared inline. This file is a GNU parallel extension to the Standard C++ Library.

5.2 `algbase.h` File Reference

Namespaces

- `std`
- `std::__parallel`

Functions

- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2 >`
`bool std::__parallel::__lexicographical_compare_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2,`
`_Iter2 __end2, _Predicate __pred, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _Predicate >`
`bool std::__parallel::__lexicographical_compare_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2,`
`_RAIter2 __end2, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2 >`
`pair< _Iter1, _Iter2 > std::__parallel::__mismatch_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2,`
`_Predicate __pred, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _Predicate >`
`pair< _RAIter1, _RAIter2 > std::__parallel::__mismatch_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2,`
`_Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter1, typename _Iter2 >`
`bool std::__parallel::equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`
`bool std::__parallel::equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2 >`
`bool std::__parallel::equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`
`bool std::__parallel::equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred)`
- `template<typename _Iter1, typename _Iter2 >`
`bool std::__parallel::lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2,`
`__gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`
`bool std::__parallel::lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2,`
`_Predicate __pred, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2 >`
`bool std::__parallel::lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`
`bool std::__parallel::lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2,`
`_Predicate __pred)`

- `template<typename _Iter1, typename _Iter2 >`
`pair< _Iter1, _Iter2 > std::__parallel::mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`
`pair< _Iter1, _Iter2 > std::__parallel::mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, __gnu_parallel::sequential_tag, _Predicate __pred, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2 >`
`pair< _Iter1, _Iter2 > std::__parallel::mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`
`pair< _Iter1, _Iter2 > std::__parallel::mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, __gnu_parallel::sequential_tag, _Predicate __pred)`

5.2.1 Detailed Description

Parallel STL function calls corresponding to the `stl_algobase.h` header. The functions defined here mainly do case switches and call the actual parallelized versions in other files. Inlining policy: Functions that basically only contain one function call, are declared inline. This file is a GNU parallel extension to the Standard C++ Library.

5.3 `algorithmfwd.h` File Reference

Namespaces

- [std](#)

Functions

- `template<typename _Filter >`
`_Filter std::adjacent_find (_Filter, _Filter)`
- `template<typename _Filter, typename _BinaryPredicate >`
`_Filter std::adjacent_find (_Filter, _Filter, _BinaryPredicate)`
- `template<typename _Iter, typename _Predicate >`
`bool std::all_of (_Iter, _Iter, _Predicate)`
- `template<typename _Iter, typename _Predicate >`
`bool std::any_of (_Iter, _Iter, _Predicate)`
- `template<typename _Filter, typename _Tp >`
`bool std::binary_search (_Filter, _Filter, const _Tp &)`
- `template<typename _Filter, typename _Tp, typename _Compare >`
`bool std::binary_search (_Filter, _Filter, const _Tp &, _Compare)`
- `template<typename _Iter, typename _OIter >`
`_OIter std::copy (_Iter, _Iter, _OIter)`
- `template<typename _BIter1, typename _BIter2 >`
`_BIter2 std::copy_backward (_BIter1, _BIter1, _BIter2)`
- `template<typename _Iter, typename _OIter, typename _Predicate >`
`_OIter std::copy_if (_Iter, _Iter, _OIter, _Predicate)`
- `template<typename _Iter, typename _Size, typename _OIter >`
`_OIter std::copy_n (_Iter, _Size, _OIter)`
- `template<typename _Iter, typename _Tp >`
`iterator_traits< _Iter >::difference_type std::count (_Iter, _Iter, const _Tp &)`

- `template<typename _Iter, typename _Predicate >`
`iterator_traits< _Iter >::difference_type std::count_if (_Iter, _Iter, _Predicate)`
- `template<typename _Iter1, typename _Iter2 >`
`bool std::equal (_Iter1, _Iter1, _Iter2)`
- `template<typename _Iter1, typename _Iter2, typename _BinaryPredicate >`
`bool std::equal (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _BinaryPredicate __binary_pred)`
- `template<typename _Filter, typename _Tp >`
`pair< _Filter, _Filter > std::equal_range (_Filter, _Filter, const _Tp &)`
- `template<typename _Filter, typename _Tp, typename _Compare >`
`pair< _Filter, _Filter > std::equal_range (_Filter, _Filter, const _Tp &, _Compare)`
- `template<typename _Filter, typename _Tp >`
`void std::fill (_Filter, _Filter, const _Tp &)`
- `template<typename _OIter, typename _Size, typename _Tp >`
`_OIter std::fill_n (_OIter, _Size, const _Tp &)`
- `template<typename _Iter, typename _Tp >`
`_Iter std::find (_Iter, _Iter, const _Tp &)`
- `template<typename _Filter1, typename _Filter2 >`
`_Filter1 std::find_end (_Filter1, _Filter1, _Filter2, _Filter2)`
- `template<typename _Filter1, typename _Filter2, typename _BinaryPredicate >`
`_Filter1 std::find_end (_Filter1, _Filter1, _Filter2, _Filter2, _BinaryPredicate)`
- `template<typename _Filter1, typename _Filter2 >`
`_Filter1 std::find_first_of (_Filter1, _Filter1, _Filter2, _Filter2)`
- `template<typename _Filter1, typename _Filter2, typename _BinaryPredicate >`
`_Filter1 std::find_first_of (_Filter1, _Filter1, _Filter2, _Filter2, _BinaryPredicate)`
- `template<typename _Iter, typename _Predicate >`
`_Iter std::find_if (_Iter, _Iter, _Predicate)`
- `template<typename _Iter, typename _Predicate >`
`_Iter std::find_if_not (_Iter, _Iter, _Predicate)`
- `template<typename _Iter, typename _Funct >`
`_Funct std::for_each (_Iter, _Iter, _Funct)`
- `template<typename _Filter, typename _Generator >`
`void std::generate (_Filter, _Filter, _Generator)`
- `template<typename _OIter, typename _Size, typename _Generator >`
`_OIter std::generate_n (_OIter, _Size, _Generator)`
- `template<typename _Iter1, typename _Iter2 >`
`bool std::includes (_Iter1, _Iter1, _Iter2, _Iter2)`
- `template<typename _Iter1, typename _Iter2, typename _Compare >`
`bool std::includes (_Iter1, _Iter1, _Iter2, _Iter2, _Compare)`
- `template<typename _BIter >`
`void std::inplace_merge (_BIter, _BIter, _BIter)`
- `template<typename _BIter, typename _Compare >`
`void std::inplace_merge (_BIter, _BIter, _BIter, _Compare)`
- `template<typename _RAIter >`
`bool std::is_heap (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`
`bool std::is_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _RAIter >`
`_RAIter std::is_heap_until (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`
`_RAIter std::is_heap_until (_RAIter, _RAIter, _Compare)`
- `template<typename _Iter, typename _Predicate >`
`bool std::is_partitioned (_Iter, _Iter, _Predicate)`

- `template<typename _Filter1, typename _Filter2 >`
`bool std::is_permutation (_Filter1, _Filter1, _Filter2)`
- `template<typename _Filter1, typename _Filter2, typename _BinaryPredicate >`
`bool std::is_permutation (_Filter1, _Filter1, _Filter2, _BinaryPredicate)`
- `template<typename _Filter >`
`bool std::is_sorted (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare >`
`bool std::is_sorted (_Filter, _Filter, _Compare)`
- `template<typename _Filter >`
`_Filter std::is_sorted_until (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare >`
`_Filter std::is_sorted_until (_Filter, _Filter, _Compare)`
- `template<typename _Filter1, typename _Filter2 >`
`void std::iter_swap (_Filter1, _Filter2)`
- `template<typename _Iter1, typename _Iter2 >`
`bool std::lexicographical_compare (_Iter1, _Iter1, _Iter2, _Iter2)`
- `template<typename _Iter1, typename _Iter2, typename _Compare >`
`bool std::lexicographical_compare (_Iter1, _Iter1, _Iter2, _Iter2, _Compare)`
- `template<typename _Filter, typename _Tp >`
`_Filter std::lower_bound (_Filter, _Filter, const _Tp &)`
- `template<typename _Filter, typename _Tp, typename _Compare >`
`_Filter std::lower_bound (_Filter, _Filter, const _Tp &, _Compare)`
- `template<typename _RAIter >`
`void std::make_heap (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`
`void std::make_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _Tp >`
`const _Tp & std::max (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp, typename _Compare >`
`const _Tp & std::max (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _Tp >`
`_Tp std::max (initializer_list< _Tp >)`
- `template<typename _Tp, typename _Compare >`
`_Tp std::max (initializer_list< _Tp >, _Compare)`
- `template<typename _Filter >`
`_Filter std::max_element (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare >`
`_Filter std::max_element (_Filter, _Filter, _Compare)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`
`_OIter std::merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`
`_OIter std::merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _Tp >`
`const _Tp & std::min (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp, typename _Compare >`
`const _Tp & std::min (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _Tp >`
`_Tp std::min (initializer_list< _Tp >)`
- `template<typename _Tp, typename _Compare >`
`_Tp std::min (initializer_list< _Tp >, _Compare)`
- `template<typename _Filter >`
`_Filter std::min_element (_Filter, _Filter)`

- `template<typename _Filter, typename _Compare >`
`_Filter std::min_element (_Filter, _Filter, _Compare)`
- `template<typename _Tp >`
`pair< const _Tp &, const _Tp & > std::minmax (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp, typename _Compare >`
`pair< const _Tp &, const _Tp & > std::minmax (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _Tp >`
`pair< _Tp, _Tp > std::minmax (initializer_list< _Tp >)`
- `template<typename _Tp, typename _Compare >`
`pair< _Tp, _Tp > std::minmax (initializer_list< _Tp >, _Compare)`
- `template<typename _Filter >`
`pair< _Filter, _Filter > std::minmax_element (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare >`
`pair< _Filter, _Filter > std::minmax_element (_Filter, _Filter, _Compare)`
- `template<typename _Iter1, typename _Iter2 >`
`pair< _Iter1, _Iter2 > std::mismatch (_Iter1, _Iter1, _Iter2)`
- `template<typename _Iter1, typename _Iter2, typename _BinaryPredicate >`
`pair< _Iter1, _Iter2 > std::mismatch (_Iter1, _Iter1, _Iter2, _BinaryPredicate)`
- `template<typename _Biter >`
`bool std::next_permutation (_Biter, _Biter)`
- `template<typename _Biter, typename _Compare >`
`bool std::next_permutation (_Biter, _Biter, _Compare)`
- `template<typename _Iter, typename _Predicate >`
`bool std::none_of (_Iter, _Iter, _Predicate)`
- `template<typename _RAIter >`
`void std::nth_element (_RAIter, _RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`
`void std::nth_element (_RAIter, _RAIter, _RAIter, _Compare)`
- `template<typename _RAIter >`
`void std::partial_sort (_RAIter, _RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`
`void std::partial_sort (_RAIter, _RAIter, _RAIter, _Compare)`
- `template<typename _Iter, typename _RAIter >`
`_RAIter std::partial_sort_copy (_Iter, _Iter, _RAIter, _RAIter)`
- `template<typename _Iter, typename _RAIter, typename _Compare >`
`_RAIter std::partial_sort_copy (_Iter, _Iter, _RAIter, _RAIter, _Compare)`
- `template<typename _Biter, typename _Predicate >`
`_Biter std::partition (_Biter, _Biter, _Predicate)`
- `template<typename _Iter, typename _OIter1, typename _OIter2, typename _Predicate >`
`pair< _OIter1, _OIter2 > std::partition_copy (_Iter, _Iter, _OIter1, _OIter2, _Predicate)`
- `template<typename _Filter, typename _Predicate >`
`_Filter std::partition_point (_Filter, _Filter, _Predicate)`
- `template<typename _RAIter >`
`void std::pop_heap (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`
`void std::pop_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _Biter >`
`bool std::prev_permutation (_Biter, _Biter)`
- `template<typename _Biter, typename _Compare >`
`bool std::prev_permutation (_Biter, _Biter, _Compare)`
- `template<typename _RAIter >`
`void std::push_heap (_RAIter, _RAIter)`

- `template<typename _RAIter, typename _Compare >`
`void std::push_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _RAIter >`
`void std::random_shuffle (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Generator >`
`void std::random_shuffle (_RAIter, _RAIter, _Generator &&)`
- `template<typename _Filter, typename _Tp >`
`_Filter std::remove (_Filter, _Filter, const _Tp &)`
- `template<typename _Iter, typename _OIter, typename _Tp >`
`_OIter std::remove_copy (_Iter, _Iter, _OIter, const _Tp &)`
- `template<typename _Iter, typename _OIter, typename _Predicate >`
`_OIter std::remove_copy_if (_Iter, _Iter, _OIter, _Predicate)`
- `template<typename _Filter, typename _Predicate >`
`_Filter std::remove_if (_Filter, _Filter, _Predicate)`
- `template<typename _Filter, typename _Tp >`
`void std::replace (_Filter, _Filter, const _Tp &, const _Tp &)`
- `template<typename _Iter, typename _OIter, typename _Tp >`
`_OIter std::replace_copy (_Iter, _Iter, _OIter, const _Tp &, const _Tp &)`
- `template<typename _Iter, typename _OIter, typename _Predicate, typename _Tp >`
`_OIter std::replace_copy_if (_Iter, _Iter, _OIter, _Predicate, const _Tp &)`
- `template<typename _Filter, typename _Predicate, typename _Tp >`
`void std::replace_if (_Filter, _Filter, _Predicate, const _Tp &)`
- `template<typename _BIter >`
`void std::reverse (_BIter, _BIter)`
- `template<typename _BIter, typename _OIter >`
`_OIter std::reverse_copy (_BIter, _BIter, _OIter)`
- `template<typename _Filter >`
`void std::rotate (_Filter, _Filter, _Filter)`
- `template<typename _Filter, typename _OIter >`
`_OIter std::rotate_copy (_Filter, _Filter, _Filter, _OIter)`
- `template<typename _Filter1, typename _Filter2 >`
`_Filter1 std::search (_Filter1, _Filter1, _Filter2, _Filter2)`
- `template<typename _Filter1, typename _Filter2, typename _BinaryPredicate >`
`_Filter1 std::search (_Filter1, _Filter1, _Filter2, _Filter2, _BinaryPredicate)`
- `template<typename _Filter, typename _Size, typename _Tp >`
`_Filter std::search_n (_Filter, _Filter, _Size, const _Tp &)`
- `template<typename _Filter, typename _Size, typename _Tp, typename _BinaryPredicate >`
`_Filter std::search_n (_Filter, _Filter, _Size, const _Tp &, _BinaryPredicate)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`
`_OIter std::set_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`
`_OIter std::set_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`
`_OIter std::set_intersection (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`
`_OIter std::set_intersection (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`
`_OIter std::set_symmetric_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`
`_OIter std::set_symmetric_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`
`_OIter std::set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`

- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`
`_OIter std::set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _RAlter, typename _UGenerator >`
`void std::shuffle (_RAlter, _RAlter, _UGenerator &&)`
- `template<typename _RAlter >`
`void std::sort (_RAlter, _RAlter)`
- `template<typename _RAlter, typename _Compare >`
`void std::sort (_RAlter, _RAlter, _Compare)`
- `template<typename _RAlter >`
`void std::sort_heap (_RAlter, _RAlter)`
- `template<typename _RAlter, typename _Compare >`
`void std::sort_heap (_RAlter, _RAlter, _Compare)`
- `template<typename _Blter, typename _Predicate >`
`_Blter std::stable_partition (_Blter, _Blter, _Predicate)`
- `template<typename _RAlter >`
`void std::stable_sort (_RAlter, _RAlter)`
- `template<typename _RAlter, typename _Compare >`
`void std::stable_sort (_RAlter, _RAlter, _Compare)`
- `template<typename _Tp >`
`void std::swap (_Tp &__a, _Tp &__b) noexcept(__and_< is_nothrow_move_constructible< _Tp >, is_nothrow_↵`
`_move_assignable< _Tp >>::value)`
- `template<typename _Tp, size_t _Nm>`
`void std::swap (_Tp(&__a)[_Nm], _Tp(&__b)[_Nm]) noexcept(noexcept(swap(*__a,*__b)))`
- `template<typename _Filter1, typename _Filter2 >`
`_Filter2 std::swap_ranges (_Filter1, _Filter1, _Filter2)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation >`
`_OIter std::transform (_Iter, _Iter, _OIter, _UnaryOperation)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BinaryOperation >`
`_OIter std::transform (_Iter1, _Iter1, _Iter2, _OIter, _BinaryOperation)`
- `template<typename _Filter >`
`_Filter std::unique (_Filter, _Filter)`
- `template<typename _Filter, typename _BinaryPredicate >`
`_Filter std::unique (_Filter, _Filter, _BinaryPredicate)`
- `template<typename _Iter, typename _OIter >`
`_OIter std::unique_copy (_Iter, _Iter, _OIter)`
- `template<typename _Iter, typename _OIter, typename _BinaryPredicate >`
`_OIter std::unique_copy (_Iter, _Iter, _OIter, _BinaryPredicate)`
- `template<typename _Filter, typename _Tp >`
`_Filter std::upper_bound (_Filter, _Filter, const _Tp &)`
- `template<typename _Filter, typename _Tp, typename _Compare >`
`_Filter std::upper_bound (_Filter, _Filter, const _Tp &, _Compare)`

5.3.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<algorithm>`.

5.4 algorithmfwd.h File Reference

Namespaces

- [std](#)
- [std::__parallel](#)

Functions

- `template<typename _Filter, typename _IterTag >`
`_Filter std::parallel::adjacent_find_switch (_Filter, _Filter, _IterTag)`
- `template<typename _Filter, typename _BiPredicate, typename _IterTag >`
`_Filter std::parallel::adjacent_find_switch (_Filter, _Filter, _BiPredicate, _IterTag)`
- `template<typename _RAIter, typename _BiPredicate >`
`_RAIter std::parallel::adjacent_find_switch (_RAIter, _RAIter, _BiPredicate, random_access_iterator_tag)`
- `template<typename _RAIter >`
`_RAIter std::parallel::adjacent_find_switch (_RAIter __begin, _RAIter __end, random_access_iterator_tag)`
- `template<typename _Iter, typename _Predicate, typename _IterTag >`
`iterator_traits< _Iter >::difference_type std::parallel::count_if_switch (_Iter, _Iter, _Predicate, _IterTag)`
- `template<typename _RAIter, typename _Predicate >`
`iterator_traits< _RAIter >::difference_type std::parallel::count_if_switch (_RAIter __begin, _RAIter __end, _Predicate __pred, random_access_iterator_tag, gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Tp, typename _IterTag >`
`iterator_traits< _Iter >::difference_type std::parallel::count_switch (_Iter, _Iter, const _Tp &, _IterTag)`
- `template<typename _RAIter, typename _Tp >`
`iterator_traits< _RAIter >::difference_type std::parallel::count_switch (_RAIter __begin, _RAIter __end, const _Tp & __value, random_access_iterator_tag, gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Filter, typename _IterTag1, typename _IterTag2 >`
`_Iter std::parallel::find_first_of_switch (_Iter, _Iter, _Filter, _Filter, _IterTag1, _IterTag2)`
- `template<typename _RAIter, typename _Filter, typename _BiPredicate, typename _IterTag >`
`_RAIter std::parallel::find_first_of_switch (_RAIter, _RAIter, _Filter, _Filter, _BiPredicate, random_access_iterator_tag, _IterTag)`
- `template<typename _Iter, typename _Filter, typename _BiPredicate, typename _IterTag1, typename _IterTag2 >`
`_Iter std::parallel::find_first_of_switch (_Iter, _Iter, _Filter, _Filter, _BiPredicate, _IterTag1, _IterTag2)`
- `template<typename _Iter, typename _Predicate, typename _IterTag >`
`_Iter std::parallel::find_if_switch (_Iter, _Iter, _Predicate, _IterTag)`
- `template<typename _RAIter, typename _Predicate >`
`_RAIter std::parallel::find_if_switch (_RAIter __begin, _RAIter __end, _Predicate __pred, random_access_iterator_tag)`
- `template<typename _RAIter, typename _Tp >`
`_RAIter std::parallel::find_switch (_RAIter __begin, _RAIter __end, const _Tp & __val, random_access_iterator_tag)`
- `template<typename _Iter, typename _Tp, typename _IterTag >`
`_Iter std::parallel::find_switch (_Iter, _Iter, const _Tp &, _IterTag)`
- `template<typename _RAIter, typename _Function >`
`_Function std::parallel::for_each_switch (_RAIter __begin, _RAIter __end, _Function __f, random_access_iterator_tag, gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Function, typename _IterTag >`
`_Function std::parallel::for_each_switch (_Iter, _Iter, _Function, _IterTag)`
- `template<typename _OIter, typename _Size, typename _Generator, typename _IterTag >`
`_OIter std::parallel::generate_n_switch (_OIter, _Size, _Generator, _IterTag)`
- `template<typename _RAIter, typename _Size, typename _Generator >`
`_RAIter std::parallel::generate_n_switch (_RAIter __begin, _Size __n, _Generator __gen, random_access_iterator_tag, gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Filter, typename _Generator, typename _IterTag >`
`void std::parallel::generate_switch (_Filter, _Filter, _Generator, _IterTag)`
- `template<typename _RAIter, typename _Generator >`
`void std::parallel::generate_switch (_RAIter __begin, _RAIter __end, _Generator __gen, random_access_iterator_tag, gnu_parallel::Parallelism __parallelism_tag)`

- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IterTag1, typename _IterTag2 >`
`bool std::parallel::lexicographical_compare_switch (_Iter1, _Iter1, _Iter2, _Iter2, _Predicate, _IterTag1, _IterTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _Predicate >`
`bool std::parallel::lexicographical_compare_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Filter, typename _Compare, typename _IterTag >`
`_Filter std::parallel::max_element_switch (_Filter, _Filter, _Compare, _IterTag)`
- `template<typename _RAIter, typename _Compare >`
`_RAIter std::parallel::max_element_switch (_RAIter __begin, _RAIter __end, _Compare __comp, random_access_iterator_tag, gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare, typename _IterTag1, typename _IterTag2, typename _IterTag3 >`
`_OIter std::parallel::merge_switch (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare, _IterTag1, _IterTag2, _IterTag3)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`
`_OIter std::parallel::merge_switch (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Filter, typename _Compare, typename _IterTag >`
`_Filter std::parallel::min_element_switch (_Filter, _Filter, _Compare, _IterTag)`
- `template<typename _RAIter, typename _Compare >`
`_RAIter std::parallel::min_element_switch (_RAIter __begin, _RAIter __end, _Compare __comp, random_access_iterator_tag, gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _RAIter1, typename _RAIter2, typename _Predicate >`
`pair< _RAIter1, _RAIter2 > std::parallel::mismatch_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IterTag1, typename _IterTag2 >`
`pair< _Iter1, _Iter2 > std::parallel::mismatch_switch (_Iter1, _Iter1, _Iter2, _Predicate, _IterTag1, _IterTag2)`
- `template<typename _Filter, typename _Predicate, typename _IterTag >`
`_Filter std::parallel::partition_switch (_Filter, _Filter, _Predicate, _IterTag)`
- `template<typename _RAIter, typename _Predicate >`
`_RAIter std::parallel::partition_switch (_RAIter __begin, _RAIter __end, _Predicate __pred, random_access_iterator_tag)`
- `template<typename _Filter, typename _Predicate, typename _Tp, typename _IterTag >`
`void std::parallel::replace_if_switch (_Filter, _Filter, _Predicate, const _Tp &, _IterTag)`
- `template<typename _RAIter, typename _Predicate, typename _Tp >`
`void std::parallel::replace_if_switch (_RAIter __begin, _RAIter __end, _Predicate __pred, const _Tp &, _new_value, random_access_iterator_tag, gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Filter, typename _Tp, typename _IterTag >`
`void std::parallel::replace_switch (_Filter, _Filter, const _Tp &, const _Tp &, _IterTag)`
- `template<typename _RAIter, typename _Tp >`
`void std::parallel::replace_switch (_RAIter __begin, _RAIter __end, const _Tp & __old_value, const _Tp & __new_value, random_access_iterator_tag, gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _RAIter, typename _Integer, typename _Tp, typename _BiPredicate >`
`_RAIter std::parallel::search_n_switch (_RAIter, _RAIter, _Integer, const _Tp &, _BiPredicate, random_access_iterator_tag)`
- `template<typename _Filter, typename _Integer, typename _Tp, typename _BiPredicate, typename _IterTag >`
`_Filter std::parallel::search_n_switch (_Filter, _Filter, _Integer, const _Tp &, _BiPredicate, _IterTag)`
- `template<typename _Filter1, typename _Filter2, typename _IterTag1, typename _IterTag2 >`
`_Filter1 std::parallel::search_switch (_Filter1, _Filter1, _Filter2, _Filter2, _IterTag1, _IterTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _BiPredicate >`
`_RAIter1 std::parallel::search_switch (_RAIter1, _RAIter1, _RAIter2, _RAIter2, _BiPredicate, random_access_iterator_tag, random_access_iterator_tag)`

- `template<typename _Filter1, typename _Filter2, typename _BiPredicate, typename _IterTag1, typename _IterTag2 >`
`_Filter1 std::parallel::search_switch (_Filter1, _Filter1, _Filter2, _Filter2, _BiPredicate, _IterTag1, _IterTag2)`
- `template<typename _RAIter1, typename _RAIter2 >`
`_RAIter1 std::parallel::search_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2`
`__end2, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAlter, typename _Predicate >`
`_Output_RAlter std::parallel::set_difference_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __`
`begin2, _RAIter2 __end2, _Output_RAlter __result, _Predicate __pred, random_access_iterator_tag, random`
`__access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OIter, typename _IterTag1, typename _IterTag2, type-`
`name _IterTag3 >`
`_OIter std::parallel::set_difference_switch (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate, _IterTag1,`
`_IterTag2, _IterTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAlter, typename _Predicate >`
`_Output_RAlter std::parallel::set_intersection_switch (_RAIter1 __begin1, _RAIter1 __end1, _RA`
`Iter2 __begin2, _RAIter2 __end2, _Output_RAlter __result, _Predicate __pred, random_access_iterator_tag,`
`random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OIter, typename _IterTag1, typename _IterTag2, type-`
`name _IterTag3 >`
`_OIter std::parallel::set_intersection_switch (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate, _Iter`
`Tag1, _IterTag2, _IterTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAlter, typename _Predicate >`
`_Output_RAlter std::parallel::set_symmetric_difference_switch (_RAIter1 __begin1, _RAIter1 __end1,`
`_RAIter2 __begin2, _RAIter2 __end2, _Output_RAlter __result, _Predicate __pred, random_access_iterator`
`__tag, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OIter, typename _IterTag1, typename _IterTag2 ,`
`typename _IterTag3 >`
`_OIter std::parallel::set_symmetric_difference_switch (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, __`
`Predicate, _IterTag1, _IterTag2, _IterTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAlter, typename _Predicate >`
`_Output_RAlter std::parallel::set_union_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __`
`begin2, _RAIter2 __end2, _Output_RAlter __result, _Predicate __pred, random_access_iterator_tag, random`
`__access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OIter, typename _IterTag1, typename _IterTag2, type-`
`name _IterTag3 >`
`_OIter std::parallel::set_union_switch (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate, _IterTag1, __`
`IterTag2, _IterTag3)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation, typename _IterTag1, typename _IterTag2 >`
`_OIter std::parallel::transform1_switch (_Iter, _Iter, _OIter, _UnaryOperation, _IterTag1, _IterTag2)`
- `template<typename _RAIter, typename _RAOIter, typename _UnaryOperation >`
`_RAOIter std::parallel::transform1_switch (_RAIter, _RAIter, _RAOIter, _UnaryOperation, random`
`__access_iterator_tag, random_access_iterator_tag, gnu_parallel::Parallelism __parallelism=gnu_parallel`
`::parallel_balanced)`
- `template<typename _RAIter1, typename _RAIter2, typename _RAIter3, typename _BiOperation >`
`_RAIter3 std::parallel::transform2_switch (_RAIter1, _RAIter1, _RAIter2, _RAIter3, _BiOperation,`
`random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag, gnu_parallel::`
`Parallelism __parallelism=gnu_parallel::parallel_balanced)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BiOperation, typename _Tag1, typename _Tag2, typename`
`_Tag3 >`
`_OIter std::parallel::transform2_switch (_Iter1, _Iter1, _Iter2, _OIter, _BiOperation, _Tag1, _Tag2, __`
`Tag3)`
- `template<typename _Iter, typename _OIter, typename _Predicate, typename _IterTag1, typename _IterTag2 >`
`_OIter std::parallel::unique_copy_switch (_Iter, _Iter, _OIter, _Predicate, _IterTag1, _IterTag2)`

- `template<typename _RAIter, typename _RandomAccess_OIter, typename _Predicate >`
`_RandomAccess_OIter std::__parallel::__unique_copy_switch (_RAIter, _RAIter, _RandomAccess_OIter, __↵`
`Predicate, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Filter >`
`_Filter std::__parallel::adjacent_find (_Filter, _Filter)`
- `template<typename _Filter >`
`_Filter std::__parallel::adjacent_find (_Filter, _Filter, __gnu_parallel::sequential_tag)`
- `template<typename _Filter, typename _BiPredicate >`
`_Filter std::__parallel::adjacent_find (_Filter, _Filter, _BiPredicate)`
- `template<typename _Filter, typename _BiPredicate >`
`_Filter std::__parallel::adjacent_find (_Filter, _Filter, _BiPredicate, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _Tp >`
`iterator_traits< _Iter >::difference_type std::__parallel::count (_Iter __begin, _Iter __end, const _Tp &__↵`
`value, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _Tp >`
`iterator_traits< _Iter >::difference_type std::__parallel::count (_Iter __begin, _Iter __end, const _Tp &__↵`
`value, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Tp >`
`iterator_traits< _Iter >::difference_type std::__parallel::count (_Iter __begin, _Iter __end, const _Tp &__↵`
`value)`
- `template<typename _Iter, typename _Predicate >`
`iterator_traits< _Iter >::difference_type std::__parallel::count_if (_Iter __begin, _Iter __end, _Predicate __↵`
`pred, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _Predicate >`
`iterator_traits< _Iter >::difference_type std::__parallel::count_if (_Iter __begin, _Iter __end, _Predicate __↵`
`pred, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Predicate >`
`iterator_traits< _Iter >::difference_type std::__parallel::count_if (_Iter __begin, _Iter __end, _Predicate __↵`
`pred)`
- `template<typename _Iter1, typename _Iter2 >`
`bool std::__parallel::equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`
`bool std::__parallel::equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred, __gnu_↵`
`parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2 >`
`bool std::__parallel::equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`
`bool std::__parallel::equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred)`
- `template<typename _Iter, typename _Tp >`
`_Iter std::__parallel::find (_Iter __begin, _Iter __end, const _Tp &__val, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _Tp >`
`_Iter std::__parallel::find (_Iter __begin, _Iter __end, const _Tp &__val)`
- `template<typename _Iter, typename _Filter >`
`_Iter std::__parallel::find_first_of (_Iter, _Iter, _Filter, _Filter, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _Filter, typename _BiPredicate >`
`_Iter std::__parallel::find_first_of (_Iter, _Iter, _Filter, _Filter, _BiPredicate, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _Filter, typename _BiPredicate >`
`_Iter std::__parallel::find_first_of (_Iter, _Iter, _Filter, _Filter, _BiPredicate)`
- `template<typename _Iter, typename _Filter >`
`_Iter std::__parallel::find_first_of (_Iter, _Iter, _Filter, _Filter)`
- `template<typename _Iter, typename _Predicate >`
`_Iter std::__parallel::find_if (_Iter __begin, _Iter __end, _Predicate __pred, __gnu_parallel::sequential_tag)`

- `template<typename _Iter, typename _Predicate >`
`_Iter std::__parallel::find_if (_Iter __begin, _Iter __end, _Predicate __pred)`
- `template<typename _Iter, typename _Function >`
`_Function std::__parallel::for_each (_Iter __begin, _Iter __end, _Function __f, __gnu_parallel::sequential_tag)`
- `template<typename _Iterator, typename _Function >`
`_Function std::__parallel::for_each (_Iterator __begin, _Iterator __end, _Function __f, __gnu_parallel::__Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Function >`
`_Function std::__parallel::for_each (_Iter, _Iter, _Function)`
- `template<typename _Filter, typename _Generator >`
`void std::__parallel::generate (_Filter, _Filter, _Generator)`
- `template<typename _Filter, typename _Generator >`
`void std::__parallel::generate (_Filter, _Filter, _Generator, __gnu_parallel::sequential_tag)`
- `template<typename _Filter, typename _Generator >`
`void std::__parallel::generate (_Filter, _Filter, _Generator, __gnu_parallel::__Parallelism)`
- `template<typename _OIter, typename _Size, typename _Generator >`
`_OIter std::__parallel::generate_n (_OIter, _Size, _Generator)`
- `template<typename _OIter, typename _Size, typename _Generator >`
`_OIter std::__parallel::generate_n (_OIter, _Size, _Generator, __gnu_parallel::sequential_tag)`
- `template<typename _OIter, typename _Size, typename _Generator >`
`_OIter std::__parallel::generate_n (_OIter, _Size, _Generator, __gnu_parallel::__Parallelism)`
- `template<typename _Iter1, typename _Iter2 >`
`bool std::__parallel::lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`
`bool std::__parallel::lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _Predicate __pred, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2 >`
`bool std::__parallel::lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`
`bool std::__parallel::lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _Predicate __pred)`
- `template<typename _Filter >`
`_Filter std::__parallel::max_element (_Filter, _Filter)`
- `template<typename _Filter >`
`_Filter std::__parallel::max_element (_Filter, _Filter, __gnu_parallel::sequential_tag)`
- `template<typename _Filter >`
`_Filter std::__parallel::max_element (_Filter, _Filter, __gnu_parallel::__Parallelism)`
- `template<typename _Filter, typename _Compare >`
`_Filter std::__parallel::max_element (_Filter, _Filter, _Compare)`
- `template<typename _Filter, typename _Compare >`
`_Filter std::__parallel::max_element (_Filter, _Filter, _Compare, __gnu_parallel::sequential_tag)`
- `template<typename _Filter, typename _Compare >`
`_Filter std::__parallel::max_element (_Filter, _Filter, _Compare, __gnu_parallel::__Parallelism)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`
`_OIter std::__parallel::merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`
`_OIter std::__parallel::merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`
`_OIter std::__parallel::merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`

- `template<typename _Iter1, typename _Iter2, typename _OIter >`
`_OIter std::parallel::merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Filter >`
`_Filter std::parallel::min_element (_Filter, _Filter)`
- `template<typename _Filter >`
`_Filter std::parallel::min_element (_Filter, _Filter, __gnu_parallel::sequential_tag)`
- `template<typename _Filter >`
`_Filter std::parallel::min_element (_Filter, _Filter, __gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Filter, typename _Compare >`
`_Filter std::parallel::min_element (_Filter, _Filter, _Compare)`
- `template<typename _Filter, typename _Compare >`
`_Filter std::parallel::min_element (_Filter, _Filter, _Compare, __gnu_parallel::sequential_tag)`
- `template<typename _Filter, typename _Compare >`
`_Filter std::parallel::min_element (_Filter, _Filter, _Compare, __gnu_parallel::Parallelism)`
- `template<typename _Iter1, typename _Iter2 >`
`pair< _Iter1, _Iter2 > std::parallel::mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`
`pair< _Iter1, _Iter2 > std::parallel::mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, __gnu_parallel::sequential_tag, _Predicate __pred, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2 >`
`pair< _Iter1, _Iter2 > std::parallel::mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`
`pair< _Iter1, _Iter2 > std::parallel::mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, __gnu_parallel::sequential_tag, _Predicate __pred)`
- `template<typename _RAIter >`
`void std::parallel::nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end, __gnu_parallel::sequential_tag)`
- `template<typename _RAIter, typename _Compare >`
`void std::parallel::nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end, _Compare __comp, __gnu_parallel::sequential_tag)`
- `template<typename _RAIter, typename _Compare >`
`void std::parallel::nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter >`
`void std::parallel::nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end)`
- `template<typename _RAIter, typename _Compare >`
`void std::parallel::partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end, _Compare __comp, __gnu_parallel::sequential_tag)`
- `template<typename _RAIter >`
`void std::parallel::partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end, __gnu_parallel::sequential_tag)`
- `template<typename _RAIter, typename _Compare >`
`void std::parallel::partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter >`
`void std::parallel::partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end)`
- `template<typename _Filter, typename _Predicate >`
`_Filter std::parallel::partition (_Filter, _Filter, _Predicate, __gnu_parallel::sequential_tag)`
- `template<typename _Filter, typename _Predicate >`
`_Filter std::parallel::partition (_Filter, _Filter, _Predicate)`
- `template<typename _RAIter >`
`void std::parallel::random_shuffle (_RAIter __begin, _RAIter __end, __gnu_parallel::sequential_tag)`

- `template<typename _RAIter, typename _RandomNumberGenerator >`
`void std::__parallel::random_shuffle (_RAIter __begin, _RAIter __end, _RandomNumberGenerator &__rand,`
`__gnu_parallel::sequential_tag)`
- `template<typename _RAIter >`
`void std::__parallel::random_shuffle (_RAIter __begin, _RAIter __end)`
- `template<typename _RAIter, typename _RandomNumberGenerator >`
`void std::__parallel::random_shuffle (_RAIter __begin, _RAIter __end, _RandomNumberGenerator &&__rand)`
- `template<typename _Filter, typename _Tp >`
`void std::__parallel::replace (_Filter, _Filter, const _Tp &, const _Tp &)`
- `template<typename _Filter, typename _Tp >`
`void std::__parallel::replace (_Filter, _Filter, const _Tp &, const _Tp &, __gnu_parallel::sequential_tag)`
- `template<typename _Filter, typename _Tp >`
`void std::__parallel::replace (_Filter, _Filter, const _Tp &, const _Tp &, __gnu_parallel::Parallelism)`
- `template<typename _Filter, typename _Predicate, typename _Tp >`
`void std::__parallel::replace_if (_Filter, _Filter, _Predicate, const _Tp &)`
- `template<typename _Filter, typename _Predicate, typename _Tp >`
`void std::__parallel::replace_if (_Filter, _Filter, _Predicate, const _Tp &, __gnu_parallel::sequential_tag)`
- `template<typename _Filter, typename _Predicate, typename _Tp >`
`void std::__parallel::replace_if (_Filter, _Filter, _Predicate, const _Tp &, __gnu_parallel::Parallelism)`
- `template<typename _Filter1, typename _Filter2 >`
`_Filter1 std::__parallel::search (_Filter1, _Filter1, _Filter2, _Filter2, __gnu_parallel::sequential_tag)`
- `template<typename _Filter1, typename _Filter2 >`
`_Filter1 std::__parallel::search (_Filter1, _Filter1, _Filter2, _Filter2)`
- `template<typename _Filter1, typename _Filter2, typename _BiPredicate >`
`_Filter1 std::__parallel::search (_Filter1, _Filter1, _Filter2, _Filter2, _BiPredicate, __gnu_parallel::sequential_tag)`
- `template<typename _Filter1, typename _Filter2, typename _BiPredicate >`
`_Filter1 std::__parallel::search (_Filter1, _Filter1, _Filter2, _Filter2, _BiPredicate)`
- `template<typename _Filter, typename _Integer, typename _Tp >`
`_Filter std::__parallel::search_n (_Filter, _Filter, _Integer, const _Tp &, __gnu_parallel::sequential_tag)`
- `template<typename _Filter, typename _Integer, typename _Tp, typename _BiPredicate >`
`_Filter std::__parallel::search_n (_Filter, _Filter, _Integer, const _Tp &, _BiPredicate, __gnu_parallel::sequential_tag)`
- `template<typename _Filter, typename _Integer, typename _Tp >`
`_Filter std::__parallel::search_n (_Filter, _Filter, _Integer, const _Tp &)`
- `template<typename _Filter, typename _Integer, typename _Tp, typename _BiPredicate >`
`_Filter std::__parallel::search_n (_Filter, _Filter, _Integer, const _Tp &, _BiPredicate)`
- `template<typename _IIter1, typename _IIter2, typename _OIter >`
`_OIter std::__parallel::set_difference (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, __gnu_parallel::sequential_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OIter, typename _Predicate >`
`_OIter std::__parallel::set_difference (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate, __gnu_parallel::sequential_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OIter >`
`_OIter std::__parallel::set_difference (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)`
- `template<typename _IIter1, typename _IIter2, typename _OIter, typename _Predicate >`
`_OIter std::__parallel::set_difference (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate)`
- `template<typename _IIter1, typename _IIter2, typename _OIter >`
`_OIter std::__parallel::set_intersection (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, __gnu_parallel::sequential_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OIter, typename _Predicate >`
`_OIter std::__parallel::set_intersection (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate, __gnu_parallel::sequential_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OIter >`
`_OIter std::__parallel::set_intersection (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)`

- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate >`
`_OIter std::__parallel::set_intersection (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`
`_OIter std::__parallel::set_symmetric_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate >`
`_OIter std::__parallel::set_symmetric_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`
`_OIter std::__parallel::set_symmetric_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate >`
`_OIter std::__parallel::set_symmetric_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`
`_OIter std::__parallel::set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate >`
`_OIter std::__parallel::set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`
`_OIter std::__parallel::set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate >`
`_OIter std::__parallel::set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate)`
- `template<typename _RAIter >`
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, __gnu_parallel::sequential_tag)`
- `template<typename _RAIter, typename _Compare >`
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, _Compare __comp, __gnu_parallel::sequential_tag)`
- `template<typename _RAIter >`
`void std::__parallel::sort (_RAIter __begin, _RAIter __end)`
- `template<typename _RAIter, typename _Compare >`
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter >`
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, __gnu_parallel::sequential_tag)`
- `template<typename _RAIter, typename _Compare >`
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, _Compare __comp, __gnu_parallel::sequential_tag)`
- `template<typename _RAIter >`
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end)`
- `template<typename _RAIter, typename _Compare >`
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, _Compare __comp)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation >`
`_OIter std::__parallel::transform (_Iter, _Iter, _OIter, _UnaryOperation)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation >`
`_OIter std::__parallel::transform (_Iter, _Iter, _OIter, _UnaryOperation, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation >`
`_OIter std::__parallel::transform (_Iter, _Iter, _OIter, _UnaryOperation, __gnu_parallel::Parallelism)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BiOperation >`
`_OIter std::__parallel::transform (_Iter1, _Iter1, _Iter2, _OIter, _BiOperation)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BiOperation >`
`_OIter std::__parallel::transform (_Iter1, _Iter1, _Iter2, _OIter, _BiOperation, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BiOperation >`
`_OIter std::__parallel::transform (_Iter1, _Iter1, _Iter2, _OIter, _BiOperation, __gnu_parallel::Parallelism)`
- `template<typename _Iter, typename _OIter >`
`_OIter std::__parallel::unique_copy (_Iter, _Iter, _OIter, __gnu_parallel::sequential_tag)`

- `template<typename _Ilter, typename _Olter, typename _Predicate >`
`_Olter std::__parallel::unique_copy (_Ilter, _Ilter, _Olter, _Predicate, __gnu_parallel::sequential_tag)`
- `template<typename _Ilter, typename _Olter >`
`_Olter std::__parallel::unique_copy (_Ilter, _Ilter, _Olter)`
- `template<typename _Ilter, typename _Olter, typename _Predicate >`
`_Olter std::__parallel::unique_copy (_Ilter, _Ilter, _Olter, _Predicate)`

5.4.1 Detailed Description

This file is a GNU parallel extension to the Standard C++ Library.

5.5 `aligned_buffer.h` File Reference

Namespaces

- [__gnu_cxx](#)

5.5.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

5.6 `alloc_traits.h` File Reference

Classes

- struct [std::allocator_traits<_Alloc >](#)
- struct [std::allocator_traits< allocator<_Tp > >](#)

Namespaces

- [std](#)

Macros

- `#define _GLIBCXX_ALLOC_TR_NESTED_TYPE(_NTYPE, _ALT)`

Functions

- `template<typename _Alloc >`
`void std::__alloc_on_copy (_Alloc &__one, const _Alloc &__two)`
- `template<typename _Alloc >`
`_Alloc std::__alloc_on_copy (const _Alloc &__a)`
- `template<typename _Alloc >`
`void std::__alloc_on_move (_Alloc &__one, _Alloc &__two)`
- `template<typename _Alloc >`
`void std::__alloc_on_swap (_Alloc &__one, _Alloc &__two)`
- `template<typename _Alloc >`
`void std::__do_alloc_on_copy (_Alloc &__one, const _Alloc &__two, true_type)`
- `template<typename _Alloc >`
`void std::__do_alloc_on_copy (_Alloc &, const _Alloc &, false_type)`
- `template<typename _Alloc >`
`void std::__do_alloc_on_move (_Alloc &__one, _Alloc &__two, true_type)`
- `template<typename _Alloc >`
`void std::__do_alloc_on_move (_Alloc &, _Alloc &, false_type)`
- `template<typename _Alloc >`
`void std::__do_alloc_on_swap (_Alloc &__one, _Alloc &__two, true_type)`
- `template<typename _Alloc >`
`void std::__do_alloc_on_swap (_Alloc &, _Alloc &, false_type)`

5.6.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

5.7 `alloc_traits.h` File Reference

Classes

- `struct __gnu_cxx::__alloc_traits< _Alloc >`
- `class __gnu_cxx::array_allocator< _Tp, _Array >`
- `class __gnu_cxx::bitmap_allocator< _Tp >`
- `class __gnu_cxx::malloc_allocator< _Tp >`
- `class __gnu_cxx::new_allocator< _Tp >`

Namespaces

- `__gnu_cxx`

5.7.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

5.8 allocator.h File Reference

Classes

- class [std::allocator< _Tp >](#)
- class [std::allocator< void >](#)

Namespaces

- [std](#)

Functions

- `template<typename _T1 , typename _T2 >
bool std::operator!= (const allocator< _T1 > &, const allocator< _T2 > &)`
- `template<typename _Tp >
bool std::operator!= (const allocator< _Tp > &, const allocator< _Tp > &)`
- `template<typename _T1 , typename _T2 >
bool std::operator== (const allocator< _T1 > &, const allocator< _T2 > &)`
- `template<typename _Tp >
bool std::operator== (const allocator< _Tp > &, const allocator< _Tp > &)`

5.8.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

5.9 array_allocator.h File Reference

Classes

- class [__gnu_cxx::array_allocator< _Tp, _Array >](#)
- class [__gnu_cxx::array_allocator_base< _Tp >](#)

Namespaces

- [__gnu_cxx](#)

Functions

- `template<typename _Tp , typename _Array >
bool __gnu_cxx::operator!= (const array_allocator< _Tp, _Array > &, const array_allocator< _Tp, _Array > &)`
- `template<typename _Tp , typename _Array >
bool __gnu_cxx::operator== (const array_allocator< _Tp, _Array > &, const array_allocator< _Tp, _Array > &)`

5.9.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

5.10 `assoc_container.hpp` File Reference

Classes

- class [__gnu_pbds::basic_branch](#)< Key, Mapped, Tag, Node_Update, Policy_Tl, _Alloc >
- class [__gnu_pbds::basic_hash_table](#)< Key, Mapped, Hash_Fn, Eq_Fn, Resize_Policy, Store_Hash, Tag, Policy_Tl, _Alloc >
- class [__gnu_pbds::cc_hash_table](#)< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc >
- class [__gnu_pbds::gp_hash_table](#)< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >
- class [__gnu_pbds::list_update](#)< Key, Mapped, Eq_Fn, Update_Policy, _Alloc >
- class [__gnu_pbds::tree](#)< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc >
- class [__gnu_pbds::trie](#)< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc >

Namespaces

- [__gnu_pbds](#)

Macros

- `#define PB_DS_BRANCH_BASE`
- `#define PB_DS_CC_HASH_BASE`
- `#define PB_DS_GP_HASH_BASE`
- `#define PB_DS_HASH_BASE`
- `#define PB_DS_LU_BASE`
- `#define PB_DS_TREE_BASE`
- `#define PB_DS_TREE_NODE_AND_IT_TRAITS`
- `#define PB_DS_TRIE_BASE`
- `#define PB_DS_TRIE_NODE_AND_IT_TRAITS`

5.10.1 Detailed Description

Contains associative containers.

5.11 `atomic_base.h` File Reference

Classes

- struct [std::__atomic_base](#)< _ITp >
- struct [std::__atomic_base](#)< _ITp >
- struct [std::__atomic_base](#)< _PTp * >
- struct [std::__atomic_flag_base](#)
- struct [std::atomic_flag](#)

Namespaces

- [std](#)

Macros

- `#define _GLIBCXX_ALWAYS_INLINE`
- `#define ATOMIC_FLAG_INIT`
- `#define ATOMIC_VAR_INIT(_VI)`

Typedefs

- typedef unsigned char **std::__atomic_flag_data_type**
- typedef __atomic_base< char > [std::atomic_char](#)
- typedef __atomic_base< char16_t > [std::atomic_char16_t](#)
- typedef __atomic_base< char32_t > [std::atomic_char32_t](#)
- typedef __atomic_base< int > [std::atomic_int](#)
- typedef __atomic_base< int_fast16_t > [std::atomic_int_fast16_t](#)
- typedef __atomic_base< int_fast32_t > [std::atomic_int_fast32_t](#)
- typedef __atomic_base< int_fast64_t > [std::atomic_int_fast64_t](#)
- typedef __atomic_base< int_fast8_t > [std::atomic_int_fast8_t](#)
- typedef __atomic_base< int_least16_t > [std::atomic_int_least16_t](#)
- typedef __atomic_base< int_least32_t > [std::atomic_int_least32_t](#)
- typedef __atomic_base< int_least64_t > [std::atomic_int_least64_t](#)
- typedef __atomic_base< int_least8_t > [std::atomic_int_least8_t](#)
- typedef __atomic_base< intmax_t > [std::atomic_intmax_t](#)
- typedef __atomic_base< intptr_t > [std::atomic_intptr_t](#)
- typedef __atomic_base< long long > [std::atomic_llong](#)
- typedef __atomic_base< long > [std::atomic_long](#)
- typedef __atomic_base< ptrdiff_t > [std::atomic_ptrdiff_t](#)
- typedef __atomic_base< signed char > [std::atomic_schar](#)
- typedef __atomic_base< short > [std::atomic_short](#)
- typedef __atomic_base< size_t > [std::atomic_size_t](#)
- typedef __atomic_base< unsigned char > [std::atomic_uchar](#)
- typedef __atomic_base< unsigned int > [std::atomic_uint](#)
- typedef __atomic_base< uint_fast16_t > [std::atomic_uint_fast16_t](#)
- typedef __atomic_base< uint_fast32_t > [std::atomic_uint_fast32_t](#)
- typedef __atomic_base< uint_fast64_t > [std::atomic_uint_fast64_t](#)
- typedef __atomic_base< uint_fast8_t > [std::atomic_uint_fast8_t](#)
- typedef __atomic_base< uint_least16_t > [std::atomic_uint_least16_t](#)
- typedef __atomic_base< uint_least32_t > [std::atomic_uint_least32_t](#)
- typedef __atomic_base< uint_least64_t > [std::atomic_uint_least64_t](#)
- typedef __atomic_base< uint_least8_t > [std::atomic_uint_least8_t](#)
- typedef __atomic_base< uintmax_t > [std::atomic_uintmax_t](#)
- typedef __atomic_base< uintptr_t > [std::atomic_uintptr_t](#)
- typedef __atomic_base< unsigned long long > [std::atomic_ullong](#)
- typedef __atomic_base< unsigned long > [std::atomic_ulong](#)
- typedef __atomic_base< unsigned short > [std::atomic_ushort](#)
- typedef __atomic_base< wchar_t > [std::atomic_wchar_t](#)
- typedef enum [std::memory_order](#) [std::memory_order](#)

Enumerations

- enum `__memory_order_modifier` { `__memory_order_mask`, `__memory_order_modifier_mask`, `__memory_order_hle_acquire`, `__memory_order_hle_release` }
- enum `std::memory_order` { `memory_order_relaxed`, `memory_order_consume`, `memory_order_acquire`, `memory_order_release`, `memory_order_acq_rel`, `memory_order_seq_cst` }

Functions

- `std::__attribute__((always_inline)) void atomic_thread_fence(memory_order __m) noexcept`
- `constexpr memory_order std::__cmpexch_failure_order(memory_order __m) noexcept`
- `constexpr memory_order std::__cmpexch_failure_order2(memory_order __m) noexcept`
- `template<typename _Tp>
_Tp std::kill_dependency(_Tp __y) noexcept`
- `constexpr memory_order std::operator&(memory_order __m, __memory_order_modifier __mod)`
- `constexpr memory_order std::operator|(memory_order __m, __memory_order_modifier __mod)`

5.11.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<atomic>`.

5.12 atomic_lockfree_defines.h File Reference

Macros

- `#define ATOMIC_BOOL_LOCK_FREE`
- `#define ATOMIC_CHAR16_T_LOCK_FREE`
- `#define ATOMIC_CHAR32_T_LOCK_FREE`
- `#define ATOMIC_CHAR_LOCK_FREE`
- `#define ATOMIC_INT_LOCK_FREE`
- `#define ATOMIC_LLONG_LOCK_FREE`
- `#define ATOMIC_LONG_LOCK_FREE`
- `#define ATOMIC_POINTER_LOCK_FREE`
- `#define ATOMIC_SHORT_LOCK_FREE`
- `#define ATOMIC_WCHAR_T_LOCK_FREE`

5.12.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<atomic>`.

5.13 atomic_word.h File Reference

Typedefs

- typedef int **_Atomic_word**

5.13.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

5.14 atomicity.h File Reference

Namespaces

- [__gnu_cxx](#)

Macros

- #define **_GLIBCXX_READ_MEM_BARRIER**
- #define **_GLIBCXX_WRITE_MEM_BARRIER**

Functions

- static void **__gnu_cxx::__atomic_add_single** (_Atomic_word * __mem, int __val)
- else **__gnu_cxx::__atomic_add_single** (__mem, __val)
- _Atomic_word **__gnu_cxx::__attribute__** ((__unused__)) **__exchange_and_add**(volatile _Atomic_word *
- static _Atomic_word **__gnu_cxx::__exchange_and_add_single** (_Atomic_word * __mem, int __val)
- else return **__gnu_cxx::__exchange_and_add_single** (__mem, __val)
- _Atomic_word int **__gnu_cxx::throw** ()

Variables

- static _Atomic_word int **__gnu_cxx::__val**

5.14.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

5.15 auto_ptr.h File Reference

Classes

- class [std::auto_ptr<_Tp>](#)
- struct [std::auto_ptr_ref<_Tp1>](#)

Namespaces

- [std](#)

5.15.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

5.16 backward_warning.h File Reference

5.16.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iosfwd>`.

5.17 balanced_quicksort.h File Reference

Classes

- struct [__gnu_parallel::QSBThreadLocal<_RAIter>](#)

Namespaces

- [__gnu_parallel](#)

Functions

- template<typename _RAIter, typename _Compare>
void [__gnu_parallel::__parallel_sort_qsb](#) (_RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __num_threads)
- template<typename _RAIter, typename _Compare>
void [__gnu_parallel::__qsb_conquer](#) (QSBThreadLocal<_RAIter> *__tls, _RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __iam, _ThreadIndex __num_threads, bool __parent_wait)
- template<typename _RAIter, typename _Compare>
std::iterator_traits<_RAIter>::difference_type [__gnu_parallel::__qsb_divide](#) (_RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __num_threads)
- template<typename _RAIter, typename _Compare>
void [__gnu_parallel::__qsb_local_sort_with_helping](#) (QSBThreadLocal<_RAIter> *__tls, _Compare &__comp, _ThreadIndex __iam, bool __wait)

5.17.1 Detailed Description

Implementation of a dynamically load-balanced parallel quicksort.

It works in-place and needs only logarithmic extra memory. The algorithm is similar to the one proposed in

P. Tsigas and Y. Zhang. A simple, fast parallel implementation of quicksort and its performance evaluation on SUN enterprise 10000. In 11th Euromicro Conference on Parallel, Distributed and Network-Based Processing, page 372, 2003.

This file is a GNU parallel extension to the Standard C++ Library.

5.18 base.h File Reference

Namespaces

- [__gnu_profile](#)
- [std](#)
- [std::__profile](#)

5.18.1 Detailed Description

Sequential helper functions. This file is a GNU profile extension to the Standard C++ Library.

5.19 base.h File Reference

Classes

- class [__gnu_parallel::__binder1st<_Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >](#)
- class [__gnu_parallel::__binder2nd<_Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >](#)
- class [__gnu_parallel::__unary_negate<_Predicate, argument_type >](#)
- class [__gnu_parallel::__EqualFromLess<_T1, _T2, _Compare >](#)
- struct [__gnu_parallel::__EqualTo<_T1, _T2 >](#)
- struct [__gnu_parallel::__Less<_T1, _T2 >](#)
- struct [__gnu_parallel::__Multiplies<_Tp1, _Tp2, _Result >](#)
- struct [__gnu_parallel::__Plus<_Tp1, _Tp2, _Result >](#)
- class [__gnu_parallel::__PseudoSequence<_Tp, _DifferenceTp >](#)
- class [__gnu_parallel::__PseudoSequenceIterator<_Tp, _DifferenceTp >](#)

Namespaces

- [__gnu_parallel](#)
- [__gnu_sequential](#)
- [std](#)
- [std::__parallel](#)

Macros

- `#define _GLIBCXX_PARALLEL_ASSERT(_Condition)`

Functions

- `void __gnu_parallel::__decode2 (_CASable __x, int &__a, int &__b)`
- `_CASable __gnu_parallel::__encode2 (int __a, int __b)`
- `_ThreadIndex __gnu_parallel::__get_max_threads ()`
- `bool __gnu_parallel::__is_parallel (const _Parallelism __p)`
- `template<typename _RAlter, typename _Compare >
_RAlter __gnu_parallel::__median_of_three_iterators (_RAlter __a, _RAlter __b, _RAlter __c, _Compare __comp)`
- `template<typename _Size >
_Size __gnu_parallel::__rd_log2 (_Size __n)`
- `template<typename _Tp >
const _Tp & __gnu_parallel::max (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp >
const _Tp & __gnu_parallel::min (const _Tp &__a, const _Tp &__b)`

5.19.1 Detailed Description

Sequential helper functions. This file is a GNU parallel extension to the Standard C++ Library.

5.20 `basic_file.h` File Reference

Namespaces

- `std`

5.20.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ios>`.

5.21 `basic_ios.h` File Reference

Classes

- `class std::basic_ios< _CharT, _Traits >`

Namespaces

- `std`

Functions

- `template<typename _Facet >`
`const _Facet & std::__check_facet (const _Facet *__f)`

5.21.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ios>`.

5.22 `basic_iterator.h` File Reference

5.22.1 Detailed Description

Includes the original header files concerned with iterators except for stream iterators. This file is a GNU parallel extension to the Standard C++ Library.

5.23 `basic_string.h` File Reference

Classes

- class `std::basic_string< _CharT, _Traits, _Alloc >`
- struct `std::hash< string >`
- struct `std::hash< u16string >`
- struct `std::hash< u32string >`
- struct `std::hash< wstring >`

Namespaces

- `std`

Functions

- `template<typename _CharT, typename _Traits, typename _Alloc >`
`basic_istream< _CharT, _Traits > & std::getline (basic_istream< _CharT, _Traits > &__is, basic_string< _CharT, _Traits, _Alloc > &__str, _CharT __delim)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`basic_istream< _CharT, _Traits > & std::getline (basic_istream< _CharT, _Traits > &__is, basic_string< _CharT, _Traits, _Alloc > &__str)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`basic_istream< _CharT, _Traits > & std::getline (basic_istream< _CharT, _Traits > &&__is, basic_string< _CharT, _Traits, _Alloc > &__str, _CharT __delim)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`basic_istream< _CharT, _Traits > & std::getline (basic_istream< _CharT, _Traits > &&__is, basic_string< _CharT, _Traits, _Alloc > &__str)`

- `template<>`
`basic_istream< char > & std::getline (basic_istream< char > &__in, basic_string< char > &__str, char __delim)`
- `template<>`
`basic_istream< wchar_t > & std::getline (basic_istream< wchar_t > &__in, basic_string< wchar_t > &__str, wchar_t __delim)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool std::operator!= (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool std::operator!= (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool std::operator!= (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`basic_string< _CharT, _Traits, _Alloc > std::operator+ (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`basic_string< _CharT, _Traits, _Alloc > std::operator+ (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`basic_string< _CharT, _Traits, _Alloc > std::operator+ (_CharT __lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`basic_string< _CharT, _Traits, _Alloc > std::operator+ (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`basic_string< _CharT, _Traits, _Alloc > std::operator+ (const basic_string< _CharT, _Traits, _Alloc > &__lhs, _CharT __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`basic_string< _CharT, _Traits, _Alloc > std::operator+ (basic_string< _CharT, _Traits, _Alloc > &&__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`basic_string< _CharT, _Traits, _Alloc > std::operator+ (const basic_string< _CharT, _Traits, _Alloc > &__lhs, basic_string< _CharT, _Traits, _Alloc > &&__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`basic_string< _CharT, _Traits, _Alloc > std::operator+ (basic_string< _CharT, _Traits, _Alloc > &&__lhs, basic_string< _CharT, _Traits, _Alloc > &&__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`basic_string< _CharT, _Traits, _Alloc > std::operator+ (const _CharT *__lhs, basic_string< _CharT, _Traits, _Alloc > &&__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`basic_string< _CharT, _Traits, _Alloc > std::operator+ (_CharT __lhs, basic_string< _CharT, _Traits, _Alloc > &&__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`basic_string< _CharT, _Traits, _Alloc > std::operator+ (basic_string< _CharT, _Traits, _Alloc > &&__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`basic_string< _CharT, _Traits, _Alloc > std::operator+ (basic_string< _CharT, _Traits, _Alloc > &&__lhs, _CharT __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool std::operator< (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`

- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool std::operator< (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool std::operator< (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__os, const basic_string< _CharT, _Traits, _Alloc > &__str)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool std::operator<= (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool std::operator<= (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool std::operator== (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT >`
`__gnu_cxx::__enable_if< __is_char< _CharT >::value, bool >::type std::operator== (const basic_string< _CharT > &__lhs, const basic_string< _CharT > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool std::operator== (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool std::operator== (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool std::operator> (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool std::operator> (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool std::operator> (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool std::operator>= (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`bool std::operator>= (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__is, basic_string< _CharT, _Traits, _Alloc > &__str)`
- `template<>`
`basic_istream< char > & std::operator>> (basic_istream< char > &__is, basic_string< char > &__str)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`
`void std::swap (basic_string< _CharT, _Traits, _Alloc > &__lhs, basic_string< _CharT, _Traits, _Alloc > &__rhs)`

5.23.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<string>`.

5.24 bin_search_tree_.hpp File Reference

Namespaces

- [__gnu_pbds](#)

Macros

- `#define PB_DS_ASSERT_NODE_CONSISTENT(_Node)`
- `#define PB_DS_BIN_TREE_NAME`
- `#define PB_DS_BIN_TREE_TRAITS_BASE`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_STRUCT_ONLY_ASSERT_VALID(X)`

5.24.1 Detailed Description

Contains an implementation class for binary search tree.

5.25 binary_heap_.hpp File Reference

Classes

- class [__gnu_pbds::detail::binary_heap< Value_Type, Cmp_Fn, _Alloc >](#)

Namespaces

- [__gnu_pbds](#)

Macros

- `#define PB_DS_ASSERT_VALID(X)`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_DEBUG_VERIFY(_Cond)`
- `#define PB_DS_ENTRY_CMP_DEC`
- `#define PB_DS_RESIZE_POLICY_DEC`

5.25.1 Detailed Description

Contains an implementation class for a binary heap.

5.26 binders.h File Reference

Classes

- class [std::binder1st<_Operation>](#)
- class [std::binder2nd<_Operation>](#)

Namespaces

- [std](#)

Functions

- [template<typename _Operation, typename _Tp> binder1st<_Operation> std::bind1st](#) (const _Operation &__fn, const _Tp &__x)
- [template<typename _Operation, typename _Tp> binder2nd<_Operation> std::bind2nd](#) (const _Operation &__fn, const _Tp &__x)

5.26.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<functional>`.

5.27 binomial_heap_.hpp File Reference

Classes

- class [__gnu_pbds::detail::binomial_heap<Value_Type, Cmp_Fn, _Alloc>](#)

Namespaces

- [__gnu_pbds](#)

Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`

5.27.1 Detailed Description

Contains an implementation class for a binomial heap.

5.28 binomial_heap_base.hpp File Reference

Classes

- class [__gnu_pbds::detail::binomial_heap_base< Value_Type, Cmp_Fn, _Alloc >](#)

Namespaces

- [__gnu_pbds](#)

Macros

- #define **PB_DS_ASSERT_BASE_NODE_CONSISTENT**(_Node, _Bool)
- #define **PB_DS_ASSERT_VALID_COND**(X, _StrictlyBinomial)
- #define **PB_DS_B_HEAP_BASE**
- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_T_DEC**

5.28.1 Detailed Description

Contains an implementation class for a base of binomial heaps.

5.29 bitmap_allocator.h File Reference

Classes

- class [__gnu_cxx::__detail::__mini_vector< _Tp >](#)
- class [__gnu_cxx::__detail::Bitmap_counter< _Tp >](#)
- class [__gnu_cxx::__detail::Ffit_finder< _Tp >](#)
- class [__gnu_cxx::bitmap_allocator< _Tp >](#)
- class [__gnu_cxx::bitmap_allocator< _Tp >](#)
- class [__gnu_cxx::free_list](#)

Namespaces

- [__gnu_cxx](#)
- [__gnu_cxx::__detail](#)

Macros

- #define **_BALLOC_ALIGN_BYTES**

Enumerations

- enum { **bits_per_byte**, **bits_per_block** }

Functions

- void [__gnu_cxx::__detail::__bit_allocate](#) (size_t * __pmap, size_t __pos) throw ()
- void [__gnu_cxx::__detail::__bit_free](#) (size_t * __pmap, size_t __pos) throw ()
- template<typename _ForwardIterator, typename _Tp, typename _Compare >
_ForwardIterator [__gnu_cxx::__detail::__lower_bound](#) (_ForwardIterator __first, _ForwardIterator __last, const
_Tp & __val, _Compare __comp)
- template<typename _AddrPair >
size_t [__gnu_cxx::__detail::__num_bitmaps](#) (_AddrPair __ap)
- template<typename _AddrPair >
size_t [__gnu_cxx::__detail::__num_blocks](#) (_AddrPair __ap)
- size_t [__gnu_cxx::__Bit_scan_forward](#) (size_t __num)
- template<typename _Tp1, typename _Tp2 >
bool [__gnu_cxx::operator!=](#) (const bitmap_allocator< _Tp1 > &, const bitmap_allocator< _Tp2 > &) throw ()
- template<typename _Tp1, typename _Tp2 >
bool [__gnu_cxx::operator==](#) (const bitmap_allocator< _Tp1 > &, const bitmap_allocator< _Tp2 > &) throw ()

5.29.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

5.29.2 Macro Definition Documentation

5.29.2.1 #define _BALLOC_ALIGN_BYTES

The constant in the expression below is the alignment required in bytes.

Definition at line 43 of file bitmap_allocator.h.

Referenced by [__gnu_cxx::free_list::M_insert\(\)](#).

5.30 boost_concept_check.h File Reference

Namespaces

- [__gnu_cxx](#)

Macros

- `#define _GLIBCXX_CLASS_REQUIRES(_type_var, _ns, _concept)`
- `#define _GLIBCXX_CLASS_REQUIRES2(_type_var1, _type_var2, _ns, _concept)`
- `#define _GLIBCXX_CLASS_REQUIRES3(_type_var1, _type_var2, _type_var3, _ns, _concept)`
- `#define _GLIBCXX_CLASS_REQUIRES4(_type_var1, _type_var2, _type_var3, _type_var4, _ns, _concept)`
- `#define _GLIBCXX_DEFINE_BINARY_OPERATOR_CONSTRAINT(_OP, _NAME)`
- `#define _GLIBCXX_DEFINE_BINARY_PREDICATE_OP_CONSTRAINT(_OP, _NAME)`
- `#define _IsUnused`

Functions

- `template<class _Tp >`
`void __gnu_cxx::__aux_require_boolean_expr (const _Tp &__t)`
- `void __gnu_cxx::__error_type_must_be_a_signed_integer_type ()`
- `void __gnu_cxx::__error_type_must_be_an_integer_type ()`
- `void __gnu_cxx::__error_type_must_be_an_unsigned_integer_type ()`
- `template<class _Concept >`
`void __gnu_cxx::__function_requires ()`

5.30.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iterator>`.

5.31 branch_policy.hpp File Reference

Classes

- `struct __gnu_pbds::detail::branch_policy< Node_Cltr, Node_Itr, _Alloc >`
- `struct __gnu_pbds::detail::branch_policy< Node_Cltr, Node_Cltr, _Alloc >`

Namespaces

- [`__gnu_pbds`](#)

5.31.1 Detailed Description

Contains a base class for branch policies.

5.32 c++0x_warning.h File Reference

5.32.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iosfwd>`.

5.33 `c++14_warning.h` File Reference

5.33.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iosfwd>`.

5.34 `c++allocator.h` File Reference

Namespaces

- [std](#)

Typedefs

- `template<typename _Tp >`
`using std::__allocator_base = __gnu_cxx::new_allocator<_Tp >`

5.34.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

5.35 `c++config.h` File Reference

Namespaces

- [std](#)

Macros

- `#define __GLIBCXX__`
- `#define __glibcxx_assert(_Condition)`
- `#define __N(msgid)`
- `#define _GLIBCXX_ABI_TAG_CXX11`
- `#define _GLIBCXX_BEGIN_EXTERN_C`
- `#define _GLIBCXX_BEGIN_NAMESPACE_LDBL`
- `#define _GLIBCXX_BEGIN_NAMESPACE_VERSION`
- `#define _GLIBCXX_DEPRECATED`
- `#define _GLIBCXX_END_EXTERN_C`
- `#define _GLIBCXX_END_NAMESPACE_LDBL`
- `#define _GLIBCXX_END_NAMESPACE_VERSION`
- `#define _GLIBCXX_EXTERN_TEMPLATE`
- `#define _GLIBCXX_FAST_MATH`

- #define _GLIBCXX_FULLY_DYNAMIC_STRING
- #define _GLIBCXX_HAVE___CXA_THREAD_ATEXIT_IMPL
- #define _GLIBCXX_HAVE_ACOSF
- #define _GLIBCXX_HAVE_ACOSL
- #define _GLIBCXX_HAVE_AS_SYMVER_DIRECTIVE
- #define _GLIBCXX_HAVE_ASINF
- #define _GLIBCXX_HAVE_ASINL
- #define _GLIBCXX_HAVE_AT_QUICK_EXIT
- #define _GLIBCXX_HAVE_ATAN2F
- #define _GLIBCXX_HAVE_ATAN2L
- #define _GLIBCXX_HAVE_ATANF
- #define _GLIBCXX_HAVE_ATANL
- #define _GLIBCXX_HAVE_ATTRIBUTE_VISIBILITY
- #define _GLIBCXX_HAVE_CEILF
- #define _GLIBCXX_HAVE_CEILL
- #define _GLIBCXX_HAVE_COMPLEX_H
- #define _GLIBCXX_HAVE_COSF
- #define _GLIBCXX_HAVE_COSHF
- #define _GLIBCXX_HAVE_COSHL
- #define _GLIBCXX_HAVE_COSL
- #define _GLIBCXX_HAVE_DLFCN_H
- #define _GLIBCXX_HAVE_EBADMSG
- #define _GLIBCXX_HAVE_ECANCELED
- #define _GLIBCXX_HAVE_ECHILD
- #define _GLIBCXX_HAVE_EIDRM
- #define _GLIBCXX_HAVE_ENDIAN_H
- #define _GLIBCXX_HAVE_ENODATA
- #define _GLIBCXX_HAVE_ENOLINK
- #define _GLIBCXX_HAVE_ENOSPC
- #define _GLIBCXX_HAVE_ENOSR
- #define _GLIBCXX_HAVE_ENOSTR
- #define _GLIBCXX_HAVE_ENOTRECOVERABLE
- #define _GLIBCXX_HAVE_ENOTSUP
- #define _GLIBCXX_HAVE_EOVERFLOW
- #define _GLIBCXX_HAVE_EOWNERDEAD
- #define _GLIBCXX_HAVE_EPERM
- #define _GLIBCXX_HAVE_EPROTO
- #define _GLIBCXX_HAVE_ETIME
- #define _GLIBCXX_HAVE_ETIMEDOUT
- #define _GLIBCXX_HAVE_ETXTBSY
- #define _GLIBCXX_HAVE_EWOULDBLOCK
- #define _GLIBCXX_HAVE_EXECINFO_H
- #define _GLIBCXX_HAVE_EXPF
- #define _GLIBCXX_HAVE_EXPL
- #define _GLIBCXX_HAVE_FABSF
- #define _GLIBCXX_HAVE_FABSL
- #define _GLIBCXX_HAVE_FENV_H
- #define _GLIBCXX_HAVE_FINITE
- #define _GLIBCXX_HAVE_FINITEF
- #define _GLIBCXX_HAVE_FINITEL
- #define _GLIBCXX_HAVE_FLOAT_H

- `#define _GLIBCXX_HAVE_FLOORF`
- `#define _GLIBCXX_HAVE_FLOORL`
- `#define _GLIBCXX_HAVE_FMODF`
- `#define _GLIBCXX_HAVE_FMODL`
- `#define _GLIBCXX_HAVE_FREXPF`
- `#define _GLIBCXX_HAVE_FREXPL`
- `#define _GLIBCXX_HAVE_GETIPINFO`
- `#define _GLIBCXX_HAVE_HYPOT`
- `#define _GLIBCXX_HAVE_HYPOTF`
- `#define _GLIBCXX_HAVE_HYPOTL`
- `#define _GLIBCXX_HAVE_ICONV`
- `#define _GLIBCXX_HAVE_INT64_T`
- `#define _GLIBCXX_HAVE_INT64_T_LONG`
- `#define _GLIBCXX_HAVE_INTTYPES_H`
- `#define _GLIBCXX_HAVE_ISINF`
- `#define _GLIBCXX_HAVE_ISINFF`
- `#define _GLIBCXX_HAVE_ISINFL`
- `#define _GLIBCXX_HAVE_ISNAN`
- `#define _GLIBCXX_HAVE_ISNANF`
- `#define _GLIBCXX_HAVE_ISNANL`
- `#define _GLIBCXX_HAVE_ISWBLANK`
- `#define _GLIBCXX_HAVE_LC_MESSAGES`
- `#define _GLIBCXX_HAVE_LDEXPF`
- `#define _GLIBCXX_HAVE_LDEXPL`
- `#define _GLIBCXX_HAVE_LIBINTL_H`
- `#define _GLIBCXX_HAVE_LIMIT_AS`
- `#define _GLIBCXX_HAVE_LIMIT_DATA`
- `#define _GLIBCXX_HAVE_LIMIT_FSIZE`
- `#define _GLIBCXX_HAVE_LIMIT_RSS`
- `#define _GLIBCXX_HAVE_LIMIT_VMEM`
- `#define _GLIBCXX_HAVE_LINUX_FUTEX`
- `#define _GLIBCXX_HAVE_LOCALE_H`
- `#define _GLIBCXX_HAVE_LOG10F`
- `#define _GLIBCXX_HAVE_LOG10L`
- `#define _GLIBCXX_HAVE_LOGF`
- `#define _GLIBCXX_HAVE_LOGL`
- `#define _GLIBCXX_HAVE_MBSTATE_T`
- `#define _GLIBCXX_HAVE_MEMORY_H`
- `#define _GLIBCXX_HAVE_MODFF`
- `#define _GLIBCXX_HAVE_MODFL`
- `#define _GLIBCXX_HAVE_POLL`
- `#define _GLIBCXX_HAVE_POWF`
- `#define _GLIBCXX_HAVE_POWL`
- `#define _GLIBCXX_HAVE_QUICK_EXIT`
- `#define _GLIBCXX_HAVE_S_ISREG`
- `#define _GLIBCXX_HAVE_SETENV`
- `#define _GLIBCXX_HAVE_SINCOS`
- `#define _GLIBCXX_HAVE_SINCOSF`
- `#define _GLIBCXX_HAVE_SINCOSL`
- `#define _GLIBCXX_HAVE_SINF`
- `#define _GLIBCXX_HAVE_SINHF`

- #define `_GLIBCXX_HAVE_SINHL`
- #define `_GLIBCXX_HAVE_SINL`
- #define `_GLIBCXX_HAVE_SQRTF`
- #define `_GLIBCXX_HAVE_SQRTL`
- #define `_GLIBCXX_HAVE_STDALIGN_H`
- #define `_GLIBCXX_HAVE_STDBOOL_H`
- #define `_GLIBCXX_HAVE_STDINT_H`
- #define `_GLIBCXX_HAVE_STDLIB_H`
- #define `_GLIBCXX_HAVE_STRERROR_L`
- #define `_GLIBCXX_HAVE_STRERROR_R`
- #define `_GLIBCXX_HAVE_STRING_H`
- #define `_GLIBCXX_HAVE_STRINGS_H`
- #define `_GLIBCXX_HAVE_STRTOF`
- #define `_GLIBCXX_HAVE_STRTOLD`
- #define `_GLIBCXX_HAVE_STRXFRM_L`
- #define `_GLIBCXX_HAVE_SYMVER_SYMBOL_RENAMING_RUNTIME_SUPPORT`
- #define `_GLIBCXX_HAVE_SYS_IOCTL_H`
- #define `_GLIBCXX_HAVE_SYS_IPC_H`
- #define `_GLIBCXX_HAVE_SYS_PARAM_H`
- #define `_GLIBCXX_HAVE_SYS_RESOURCE_H`
- #define `_GLIBCXX_HAVE_SYS_SDT_H`
- #define `_GLIBCXX_HAVE_SYS_SEM_H`
- #define `_GLIBCXX_HAVE_SYS_STAT_H`
- #define `_GLIBCXX_HAVE_SYS_SYSINFO_H`
- #define `_GLIBCXX_HAVE_SYS_TIME_H`
- #define `_GLIBCXX_HAVE_SYS_TYPES_H`
- #define `_GLIBCXX_HAVE_SYS_UIO_H`
- #define `_GLIBCXX_HAVE_TANF`
- #define `_GLIBCXX_HAVE_TANHF`
- #define `_GLIBCXX_HAVE_TANHL`
- #define `_GLIBCXX_HAVE_TANL`
- #define `_GLIBCXX_HAVE_TGMATH_H`
- #define `_GLIBCXX_HAVE_TLS`
- #define `_GLIBCXX_HAVE_UNISTD_H`
- #define `_GLIBCXX_HAVE_VFWSCANF`
- #define `_GLIBCXX_HAVE_VSWSCANF`
- #define `_GLIBCXX_HAVE_VWSCANF`
- #define `_GLIBCXX_HAVE_WCHAR_H`
- #define `_GLIBCXX_HAVE_WCSTOF`
- #define `_GLIBCXX_HAVE_WCTYPE_H`
- #define `_GLIBCXX_HAVE_WRITEV`
- #define `_GLIBCXX_HOSTED`
- #define `_GLIBCXX_ICONV_CONST`
- #define `_GLIBCXX_INLINE_VERSION`
- #define `_GLIBCXX_NAMESPACE_LDBL`
- #define `_GLIBCXX_PACKAGE__GLIBCXX_VERSION`
- #define `_GLIBCXX_PACKAGE_BUGREPORT`
- #define `_GLIBCXX_PACKAGE_NAME`
- #define `_GLIBCXX_PACKAGE_STRING`
- #define `_GLIBCXX_PACKAGE_TARNAME`
- #define `_GLIBCXX_PACKAGE_URL`

- #define _GLIBCXX_PSEUDO_VISIBILITY(V)
- #define _GLIBCXX_RES_LIMITS
- #define _GLIBCXX_STDIO_EOF
- #define _GLIBCXX_STDIO_SEEK_CUR
- #define _GLIBCXX_STDIO_SEEK_END
- #define _GLIBCXX_SYMVER
- #define _GLIBCXX_SYMVER_GNU
- #define _GLIBCXX_SYNCHRONIZATION_HAPPENS_AFTER(A)
- #define _GLIBCXX_SYNCHRONIZATION_HAPPENS_BEFORE(A)
- #define _GLIBCXX_THROW_OR_ABORT(_EXC)
- #define _GLIBCXX_USE_C99
- #define _GLIBCXX_USE_C99_COMPLEX
- #define _GLIBCXX_USE_C99_COMPLEX_TR1
- #define _GLIBCXX_USE_C99_CTYPE_TR1
- #define _GLIBCXX_USE_C99_FENV_TR1
- #define _GLIBCXX_USE_C99_INTTYPES_TR1
- #define _GLIBCXX_USE_C99_INTTYPES_WCHAR_T_TR1
- #define _GLIBCXX_USE_C99_MATH
- #define _GLIBCXX_USE_C99_MATH_TR1
- #define _GLIBCXX_USE_C99_STDINT_TR1
- #define _GLIBCXX_USE_CLOCK_MONOTONIC
- #define _GLIBCXX_USE_CLOCK_REALTIME
- #define _GLIBCXX_USE_DECIMAL_FLOAT
- #define _GLIBCXX_USE_DEPRECATED
- #define _GLIBCXX_USE_FLOAT128
- #define _GLIBCXX_USE_GET_NPROCS
- #define _GLIBCXX_USE_GETTIMEOFDAY
- #define _GLIBCXX_USE_INT128
- #define _GLIBCXX_USE_LFS
- #define _GLIBCXX_USE_LONG_LONG
- #define _GLIBCXX_USE_NANOSLEEP
- #define _GLIBCXX_USE_NLS
- #define _GLIBCXX_USE_RANDOM_TR1
- #define _GLIBCXX_USE_SC_NPROCESSORS_ONLN
- #define _GLIBCXX_USE_SCHED_YIELD
- #define _GLIBCXX_USE_TMPNAM
- #define _GLIBCXX_USE_WCHAR_T
- #define _GLIBCXX_VERBOSE
- #define _GLIBCXX_VISIBILITY(V)
- #define _GLIBCXX_WEAK_DEFINITION
- #define _GLIBCXX_X86_RDRAND
- #define _GTHREAD_USE_MUTEX_TIMEDLOCK
- #define LT_OBJDIR
- #define STDC_HEADERS

Typedefs

- typedef __PTRDIFF_TYPE__ std::ptrdiff_t
- typedef __SIZE_TYPE__ std::size_t

Variables

- `decltype(nullptr)` typedef **`std::nullptr_t`**

5.35.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iosfwd>`.

5.36 `c++io.h` File Reference

Namespaces

- [`std`](#)

Typedefs

- typedef FILE **`std::__c_file`**
- typedef `__pthread_mutex_t` **`std::__c_lock`**

5.36.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ios>`.

5.37 `c++locale.h` File Reference

Namespaces

- [`std`](#)

Macros

- `#define` **`_GLIBCXX_C_LOCALE_GNU`**
- `#define` **`_GLIBCXX_NUM_CATEGORIES`**

Typedefs

- typedef `__locale_t` **`std::__c_locale`**

Functions

- int **std::__convert_from_v** (const __c_locale &__cloc __attribute__((__unused__)), char *__out, const int __size __attribute__((__unused__)), const char *__fmt,...)

5.37.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

5.38 c++locale_internal.h File Reference

5.38.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

5.39 cast.h File Reference

Classes

- struct [__gnu_cxx::_Caster<_ToType>](#)

Namespaces

- [__gnu_cxx](#)

Functions

- template<typename _ToType, typename _FromType >
_ToType [__gnu_cxx::const_pointer_cast](#) (const _FromType &__arg)
- template<typename _ToType, typename _FromType >
_ToType [__gnu_cxx::const_pointer_cast](#) (_FromType *__arg)
- template<typename _ToType, typename _FromType >
_ToType [__gnu_cxx::dynamic_pointer_cast](#) (const _FromType &__arg)
- template<typename _ToType, typename _FromType >
_ToType [__gnu_cxx::dynamic_pointer_cast](#) (_FromType *__arg)
- template<typename _ToType, typename _FromType >
_ToType [__gnu_cxx::reinterpret_pointer_cast](#) (const _FromType &__arg)
- template<typename _ToType, typename _FromType >
_ToType [__gnu_cxx::reinterpret_pointer_cast](#) (_FromType *__arg)
- template<typename _ToType, typename _FromType >
_ToType [__gnu_cxx::static_pointer_cast](#) (const _FromType &__arg)
- template<typename _ToType, typename _FromType >
_ToType [__gnu_cxx::static_pointer_cast](#) (_FromType *__arg)

5.39.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ext/pointer.h>`.

5.40 cc_hash_max_collision_check_resize_trigger_imp.hpp File Reference

5.40.1 Detailed Description

Contains a resize trigger implementation.

5.41 cc_ht_map.hpp File Reference

Classes

- class [__gnu_pbds::detail::cc_ht_map](#)< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >

Namespaces

- [__gnu_pbds](#)

Macros

- `#define PB_DS_CC_HASH_NAME`
- `#define PB_DS_CC_HASH_TRAITS_BASE`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_GEN_POS`
- `#define PB_DS_HASH_EQ_FN_C_DEC`
- `#define PB_DS_RANGED_HASH_FN_C_DEC`

5.41.1 Detailed Description

Contains an implementation class for `cc_ht_map_`.

5.42 char_traits.h File Reference

Classes

- struct [__gnu_cxx::Char_types](#)< _CharT >
- struct [__gnu_cxx::char_traits](#)< _CharT >
- struct [std::char_traits](#)< _CharT >
- struct [std::char_traits](#)< char >
- struct [std::char_traits](#)< wchar_t >

Namespaces

- [__gnu_cxx](#)
- [std](#)

5.42.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<string>`.

5.43 `checkers.h` File Reference

Namespaces

- [__gnu_parallel](#)

Functions

- `template<typename _Iter , typename _Compare >`
`bool __gnu_parallel::__is_sorted (_Iter __begin, _Iter __end, _Compare __comp)`

5.43.1 Detailed Description

Routines for checking the correctness of algorithm results. This file is a GNU parallel extension to the Standard C++ Library.

5.44 `cmp_fn_imps.hpp` File Reference

5.44.1 Detailed Description

Contains implementations of `cc_ht_map_'s` entire container comparison related functions.

5.45 `codecvt.h` File Reference

Classes

- class [std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >](#)
- class [std::codecvt< _InternT, _ExternT, _StateT >](#)
- class [std::codecvt< char, char, mbstate_t >](#)
- class [std::codecvt< wchar_t, char, mbstate_t >](#)
- class [std::codecvt_base](#)
- class [std::codecvt_byname< _InternT, _ExternT, _StateT >](#)

Namespaces

- [std](#)

5.45.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

5.46 `codecvt_specializations.h` File Reference

Classes

- struct [__gnu_cxx::encoding_char_traits<_CharT>](#)
- class [__gnu_cxx::encoding_state](#)
- class [std::codecvt<_InternT, _ExternT, encoding_state>](#)

Namespaces

- [__gnu_cxx](#)
- [std](#)

Functions

- template<typename _Tp>
size_t **std::__iconv_adaptor** (size_t(*__func)(iconv_t, _Tp, size_t *, char **, size_t *), iconv_t __cd, char **↵
__inbuf, size_t *__inbytes, char **__outbuf, size_t *__outbytes)

5.46.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

5.47 `compatibility.h` File Reference

5.47.1 Detailed Description

This is an internal header file, included by other library sources. You should not attempt to use it directly.

5.48 `compatibility.h` File Reference

Namespaces

- [__gnu_parallel](#)

Functions

- `template<typename _Tp >`
`_Tp __gnu_parallel::__add_omp (volatile _Tp *__ptr, _Tp __addend)`
- `template<typename _Tp >`
`bool __gnu_parallel::__cas_omp (volatile _Tp *__ptr, _Tp __comparand, _Tp __replacement)`
- `template<typename _Tp >`
`bool __gnu_parallel::__compare_and_swap (volatile _Tp *__ptr, _Tp __comparand, _Tp __replacement)`
- `template<typename _Tp >`
`_Tp __gnu_parallel::__fetch_and_add (volatile _Tp *__ptr, _Tp __addend)`
- `void __gnu_parallel::__yield ()`

5.48.1 Detailed Description

Compatibility layer, mostly concerned with atomic operations.

This file is a GNU parallel extension to the Standard C++ Library and contains implementation details for the library's internal use.

5.49 compiletime_settings.h File Reference

Macros

- `#define _GLIBCXX_ASSERTIONS`
- `#define _GLIBCXX_CALL(__n)`
- `#define _GLIBCXX_RANDOM_SHUFFLE_CONSIDER_L1`
- `#define _GLIBCXX_RANDOM_SHUFFLE_CONSIDER_TLB`
- `#define _GLIBCXX_SCALE_DOWN_FPU`
- `#define _GLIBCXX_VERBOSE_LEVEL`

5.49.1 Detailed Description

Defines on options concerning debugging and performance, at compile-time. This file is a GNU parallel extension to the Standard C++ Library.

5.49.2 Macro Definition Documentation

5.49.2.1 `#define _GLIBCXX_ASSERTIONS`

Switch on many `_GLIBCXX_PARALLEL_ASSERTIONS` in parallel code. Should be switched on only locally.

Definition at line 61 of file `compiletime_settings.h`.

Referenced by `__gnu_parallel::__qsb_local_sort_with_helping()`.

5.49.2.2 `#define _GLIBCXX_CALL(__n)`

Macro to produce log message when entering a function.

Parameters

<code>↔</code>	Input size.
<code>n</code>	

See also

`_GLIBCXX_VERBOSE_LEVEL`

Definition at line 44 of file `compiletime_settings.h`.

Referenced by `__gnu_parallel::__find_template()`, `__gnu_parallel::__for_each_template_random_access_workstealing()`, `__gnu_parallel::__merge_advance()`, `__gnu_parallel::__parallel_nth_element()`, `__gnu_parallel::__parallel_partial↔sum()`, `__gnu_parallel::__parallel_partition()`, `__gnu_parallel::__parallel_random_shuffle_drs()`, `__gnu_parallel::__↔parallel_sort()`, `__gnu_parallel::__parallel_sort_qs()`, `__gnu_parallel::__parallel_sort_qsb()`, `__gnu_parallel::__↔parallel_unique_copy()`, `__gnu_parallel::__search_template()`, `__gnu_parallel::__sequential_multiway_merge()`, `__↔gnu_parallel::__multiseq_partition()`, `__gnu_parallel::__multiseq_selection()`, `__gnu_parallel::__multiway_merge()`, `__gnu_↔parallel::__multiway_merge_3_variant()`, `__gnu_parallel::__multiway_merge_4_variant()`, `__gnu_parallel::__multiway_merge_↔loser_tree()`, `__gnu_parallel::__multiway_merge_loser_tree_sentinel()`, `__gnu_parallel::__multiway_merge_loser_tree_↔unguarded()`, `__gnu_parallel::__multiway_merge_sentinels()`, `__gnu_parallel::__parallel_multiway_merge()`, and `__gnu_↔parallel::__parallel_sort_mwms()`.

5.49.2.3 `#define _GLIBCXX_RANDOM_SHUFFLE_CONSIDER_L1`

Switch on many `_GLIBCXX_PARALLEL_ASSERTions` in parallel code. Consider the size of the L1 cache for `gnu_↔parallel::__parallel_random_shuffle()`.

Definition at line 68 of file `compiletime_settings.h`.

5.49.2.4 `#define _GLIBCXX_RANDOM_SHUFFLE_CONSIDER_TLB`

Switch on many `_GLIBCXX_PARALLEL_ASSERTions` in parallel code. Consider the size of the TLB for `gnu_parallel_↔::__parallel_random_shuffle()`.

Definition at line 74 of file `compiletime_settings.h`.

5.49.2.5 `#define _GLIBCXX_SCALE_DOWN_FPU`

Use floating-point scaling instead of modulo for mapping random numbers to a range. This can be faster on certain CPUs.

Definition at line 55 of file `compiletime_settings.h`.

5.49.2.6 `#define _GLIBCXX_VERBOSE_LEVEL`

Determine verbosity level of the parallel mode. Level 1 prints a message each time a parallel-mode function is entered.

Definition at line 37 of file `compiletime_settings.h`.

5.50 `complex.h` File Reference

Macros

- `#define __GLIBCXX_COMPLEX_H`

5.50.1 Detailed Description

This is a Standard C++ Library header.

5.51 `concept_check.h` File Reference

Macros

- `#define __glibcxx_class_requires(_a, _b)`
- `#define __glibcxx_class_requires2(_a, _b, _c)`
- `#define __glibcxx_class_requires3(_a, _b, _c, _d)`
- `#define __glibcxx_class_requires4(_a, _b, _c, _d, _e)`
- `#define __glibcxx_function_requires(...)`

5.51.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iterator>`.

5.52 `concurrency.h` File Reference

Classes

- class [`__gnu_cxx::__scoped_lock`](#)

Namespaces

- [`__gnu_cxx`](#)

Enumerations

- enum `_Lock_policy` { `_S_single`, `_S_mutex`, `_S_atomic` }

Functions

- void `__gnu_cxx::__throw_concurrency_lock_error()`
- void `__gnu_cxx::__throw_concurrency_unlock_error()`

Variables

- static const _Lock_policy [__gnu_cxx::__default_lock_policy](#)

5.52.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

5.53 cond_dealtor.hpp File Reference

Classes

- class [__gnu_pbds::detail::cond_dealtor< Entry, _Alloc >](#)

Namespaces

- [__gnu_pbds](#)

5.53.1 Detailed Description

Contains a conditional deallocator.

5.54 cond_key_dtor_entry_dealtor.hpp File Reference

Classes

- class [__gnu_pbds::detail::cond_dealtor< Entry, _Alloc >](#)

Namespaces

- [__gnu_pbds](#)

5.54.1 Detailed Description

Contains a conditional key destructor, used for exception handling.

5.55 const_iterator.hpp File Reference

Classes

- class [__gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >](#)

Namespaces

- [__gnu_pbds](#)

Macros

- `#define PB_DS_BIN_HEAP_CIT_BASE`

5.55.1 Detailed Description

Contains an iterator class returned by the table's const find and insert methods.

5.56 `const_iterator.hpp` File Reference

Classes

- class [__gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >](#)

Namespaces

- [__gnu_pbds](#)

Macros

- `#define PB_DS_BASIC_HEAP_CIT_BASE`
- `#define PB_DS_CLASS_C_DEC`

5.56.1 Detailed Description

Contains an iterator class returned by the table's const find and insert methods.

5.57 `const_iterator.hpp` File Reference

Classes

- class [const_iterator_](#)

5.57.1 Detailed Description

Contains an iterator class used for const ranging over the elements of the table.

5.58 constructor_destructor_fn_imps.hpp File Reference

5.58.1 Detailed Description

Contains implementations of cc_ht_map_'s constructors, destructor, and related functions.

5.59 constructor_destructor_fn_imps.hpp File Reference

5.59.1 Detailed Description

Contains implementations of gp_ht_map_'s constructors, destructor, and related functions.

5.60 constructor_destructor_fn_imps.hpp File Reference

5.61 constructor_destructor_no_store_hash_fn_imps.hpp File Reference

5.61.1 Detailed Description

Contains implementations of cc_ht_map_'s constructors, destructor, and related functions.

5.62 constructor_destructor_no_store_hash_fn_imps.hpp File Reference

5.62.1 Detailed Description

Contains implementations of gp_ht_map_'s constructors, destructor, and related functions.

5.63 constructor_destructor_store_hash_fn_imps.hpp File Reference

5.63.1 Detailed Description

Contains implementations of cc_ht_map_'s constructors, destructor, and related functions.

5.64 constructor_destructor_store_hash_fn_imps.hpp File Reference

5.64.1 Detailed Description

Contains implementations of gp_ht_map_'s constructors, destructor, and related functions.

5.65 constructors_destructor_fn_imps.hpp File Reference

5.65.1 Detailed Description

Contains an implementation class for binary_heap_.

5.66 constructors_destructor_fn_imps.hpp File Reference

5.66.1 Detailed Description

Contains an implementation for binomial_heap_.

5.67 constructors_destructor_fn_imps.hpp File Reference

5.67.1 Detailed Description

Contains an implementation class for a base of binomial heaps.

5.68 constructors_destructor_fn_imps.hpp File Reference

5.68.1 Detailed Description

Contains an implementation class for bin_search_tree_.

5.69 constructors_destructor_fn_imps.hpp File Reference

5.69.1 Detailed Description

Contains an implementation class for left_child_next_sibling_heap_.

5.70 constructors_destructor_fn_imps.hpp File Reference

5.70.1 Detailed Description

Contains an implementation class for ov_tree_.

5.71 constructors_destructor_fn_imps.hpp File Reference

5.71.1 Detailed Description

Contains an implementation class for a pairing heap.

5.72 constructors_destructor_fn_imps.hpp File Reference

5.72.1 Detailed Description

Contains an implementation class for pat_trie.

5.73 constructors_destructor_fn_imps.hpp File Reference

5.73.1 Detailed Description

Contains an implementation for rb_tree_.

5.74 constructors_destructor_fn_imps.hpp File Reference

5.74.1 Detailed Description

Contains an implementation for rc_binomial_heap_.

5.75 constructors_destructor_fn_imps.hpp File Reference

5.75.1 Detailed Description

Contains an implementation class for splay_tree_.

5.76 constructors_destructor_fn_imps.hpp File Reference

5.76.1 Detailed Description

Contains an implementation for thin_heap_.

5.77 container_base_dispatch.hpp File Reference

Classes

- struct [__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, cc_hash_tag, Policy_Tl >](#)
- struct [__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, gp_hash_tag, Policy_Tl >](#)
- struct [__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, list_update_tag, Policy_Tl >](#)
- struct [__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, ov_tree_tag, Policy_Tl >](#)
- struct [__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, pat_trie_tag, Policy_Tl >](#)
- struct [__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, rb_tree_tag, Policy_Tl >](#)
- struct [__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, splay_tree_tag, Policy_Tl >](#)
- struct [__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, cc_hash_tag, Policy_Tl >](#)
- struct [__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, gp_hash_tag, Policy_Tl >](#)
- struct [__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, list_update_tag, Policy_Tl >](#)
- struct [__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, ov_tree_tag, Policy_Tl >](#)
- struct [__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, pat_trie_tag, Policy_Tl >](#)
- struct [__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, rb_tree_tag, Policy_Tl >](#)
- struct [__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, splay_tree_tag, Policy_Tl >](#)

Namespaces

- [__gnu_pbds](#)

Macros

- **#define PB_DS_ASSERT_VALID(X)**
- **#define PB_DS_CHECK_KEY_DOES_NOT_EXIST(_Key)**
- **#define PB_DS_CHECK_KEY_EXISTS(_Key)**
- **#define PB_DS_DATA_FALSE_INDICATOR**
- **#define PB_DS_DATA_TRUE_INDICATOR**
- **#define PB_DS_DEBUG_VERIFY(_Cond)**
- **#define PB_DS_EP2VP(X)**
- **#define PB_DS_EP2VP(X)**
- **#define PB_DS_V2F(X)**
- **#define PB_DS_V2F(X)**
- **#define PB_DS_V2S(X)**
- **#define PB_DS_V2S(X)**

5.77.1 Detailed Description

Contains associative container dispatching.

5.78 cpp_type_traits.h File Reference

Classes

- class [std::move_iterator< _Iterator >](#)

Namespaces

- [__gnu_cxx](#)
- [std](#)

5.78.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ext/type_traits>`.

5.79 `cpu_defines.h` File Reference

5.79.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iosfwd>`.

5.80 `ctype_base.h` File Reference

Classes

- struct [std::ctype_base](#)

Namespaces

- [std](#)

5.80.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

5.81 `ctype_inline.h` File Reference

Namespaces

- [std](#)

5.81.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

5.82 cxxabi.h File Reference

Classes

- class [__gnu_cxx::recursive_init_error](#)

Namespaces

- [__gnu_cxx](#)
- [abi](#)

Typedefs

- typedef [__cxa_cdtor_return_type](#)(* [__cxxabiv1::__cxa_cdtor_type](#)) (void *)

Functions

- [__cxa_dependent_exception](#) * [__cxxabiv1::__cxa_allocate_dependent_exception](#) () noexcept
- void * [__cxxabiv1::__cxa_allocate_exception](#) (size_t) noexcept
- int [__cxxabiv1::__cxa_atexit](#) (void*)(void *), void *, void *) noexcept
- void [__cxxabiv1::__cxa_bad_cast](#) () __attribute__((__noreturn__))
- void [__cxxabiv1::__cxa_bad_typeid](#) () __attribute__((__noreturn__))
- void * [__cxxabiv1::__cxa_begin_catch](#) (void *) noexcept
- std::type_info * [__cxxabiv1::__cxa_current_exception_type](#) () noexcept __attribute__((__pure__))
- void [__cxxabiv1::__cxa_deleted_virtual](#) (void) __attribute__((__noreturn__))
- char * [__cxxabiv1::__cxa_demangle](#) (const char * __mangled_name, char * __output_buffer, size_t * __length, int * __status)
- void [__cxxabiv1::__cxa_end_catch](#) ()
- int [__cxxabiv1::__cxa_finalize](#) (void *)
- void [__cxxabiv1::__cxa_free_dependent_exception](#) (__cxa_dependent_exception *) noexcept
- void [__cxxabiv1::__cxa_free_exception](#) (void *) noexcept
- void * [__cxxabiv1::__cxa_get_exception_ptr](#) (void *) noexcept __attribute__((__pure__))
- [__cxa_eh_globals](#) * [__cxxabiv1::__cxa_get_globals](#) () noexcept __attribute__((__const__))
- [__cxa_eh_globals](#) * [__cxxabiv1::__cxa_get_globals_fast](#) () noexcept __attribute__((__const__))
- void [__cxxabiv1::__cxa_guard_abort](#) (__guard *) noexcept
- int [__cxxabiv1::__cxa_guard_acquire](#) (__guard *)
- void [__cxxabiv1::__cxa_guard_release](#) (__guard *) noexcept
- void [__cxxabiv1::__cxa_pure_virtual](#) (void) __attribute__((__noreturn__))
- void [__cxxabiv1::__cxa_rethrow](#) () __attribute__((__noreturn__))
- int [__cxxabiv1::__cxa_thread_atexit](#) (void*)(void *), void *, void *) noexcept
- void [__cxxabiv1::__cxa_throw](#) (void *, std::type_info *, void (*)(void *)) __attribute__((__noreturn__))
- void [__cxxabiv1::__cxa_throw_bad_array_length](#) () __attribute__((__noreturn__))
- void [__cxxabiv1::__cxa_throw_bad_array_new_length](#) () __attribute__((__noreturn__))
- [__cxa_vec_ctor_return_type](#) [__cxxabiv1::__cxa_vec_ctor](#) (void * __dest_array, void * __src_array, size_t __element_count, size_t __element_size, [__cxa_cdtor_return_type](#)(* __constructor)(void *, void *), [__cxa_cdtor_return_type](#) __destructor)
- void [__cxxabiv1::__cxa_vec_cleanup](#) (void * __array_address, size_t __element_count, size_t __s, [__cxa_cdtor_return_type](#) __destructor) noexcept

- `__cxa_vec_ctor_return_type __cxxabiv1::__cxa_vec_ctor` (void *__array_address, size_t __element_count, size_t __element_size, __cxa_ctor_type __constructor, __cxa_ctor_type __destructor)
- void `__cxxabiv1::__cxa_vec_delete` (void *__array_address, size_t __element_size, size_t __padding_size, \leftrightarrow __cxa_ctor_type __destructor)
- void `__cxxabiv1::__cxa_vec_delete2` (void *__array_address, size_t __element_size, size_t __padding_size, __cxa_ctor_type __destructor, void(* __dealloc)(void *))
- void `__cxxabiv1::__cxa_vec_delete3` (void *__array_address, size_t __element_size, size_t __padding_size, __cxa_ctor_type __destructor, void(* __dealloc)(void *, size_t))
- void `__cxxabiv1::__cxa_vec_dtor` (void *__array_address, size_t __element_count, size_t __element_size, \leftrightarrow __cxa_ctor_type __destructor)
- void * `__cxxabiv1::__cxa_vec_new` (size_t __element_count, size_t __element_size, size_t __padding_size, __cxa_ctor_type __constructor, __cxa_ctor_type __destructor)
- void * `__cxxabiv1::__cxa_vec_new2` (size_t __element_count, size_t __element_size, size_t __padding_size, __cxa_ctor_type __constructor, __cxa_ctor_type __destructor, void *(* __alloc)(size_t), void(* __dealloc)(void *))
- void * `__cxxabiv1::__cxa_vec_new3` (size_t __element_count, size_t __element_size, size_t __padding_size, __cxa_ctor_type __constructor, __cxa_ctor_type __destructor, void *(* __alloc)(size_t), void(* __dealloc)(void *, size_t))
- void * `__cxxabiv1::__dynamic_cast` (const void *__src_ptr, const __class_type_info *__src_type, const \leftrightarrow __class_type_info *__dst_type, ptrdiff_t __src2dst)

5.82.1 Detailed Description

The header provides an interface to the C++ ABI.

5.82.2 Function Documentation

5.82.2.1 `char* __cxxabiv1::__cxa_demangle (const char * __mangled_name, char * __output_buffer, size_t * __length, int * __status)`

Demangling routine. ABI-mandated entry point in the C++ runtime library for demangling.

Parameters

<code>__mangled_name</code>	A NUL-terminated character string containing the name to be demangled.
<code>__output_buffer</code>	A region of memory, allocated with malloc, of * <code>__length</code> bytes, into which the demangled name is stored. If <code>__output_buffer</code> is not long enough, it is expanded using realloc. <code>__output_buffer</code> may instead be NULL; in that case, the demangled name is placed in a region of memory allocated with malloc.
<code>__length</code>	If <code>__length</code> is non-NULL, the length of the buffer containing the demangled name is placed in * <code>__length</code> .
<code>__status</code>	* <code>__status</code> is set to one of the following values: 0: The demangling operation succeeded. -1: A memory allocation failure occurred. -2: <code>mangled_name</code> is not a valid name under the C++ ABI mangling rules. -3: One of the arguments is invalid.

Returns

A pointer to the start of the NUL-terminated demangled name, or NULL if the demangling fails. The caller is responsible for deallocating this memory using `free`.

The demangling is performed using the C++ ABI mangling rules, with GNU extensions. For example, this function is used in `__gnu_cxx::__verbose_terminate_handler`.

See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt12ch39.html> for other examples of use.

Note

The same demangling functionality is available via `libiberty` (`<libiberty/demangle.h>` and `libiberty.a`) in GCC 3.1 and later, but that requires explicit installation (`-enable-install-libiberty`) and uses a different API, although the ABI is unchanged.

5.83 cxxabi_forced.h File Reference**Classes**

- class `__cxxabiv1::__forced_unwind`

5.83.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<cxxabi.h>`.

5.84 cxxabi_tweaks.h File Reference**Macros**

- `#define _GLIBCXX_CXA_VEC_CTOR_RETURN(x)`
- `#define _GLIBCXX_GUARD_BIT`
- `#define _GLIBCXX_GUARD_PENDING_BIT`
- `#define _GLIBCXX_GUARD_SET(x)`
- `#define _GLIBCXX_GUARD_TEST(x)`
- `#define _GLIBCXX_GUARD_WAITING_BIT`

Typedefs

- `typedef void __cxxabiv1::__cxa_cdtor_return_type`
- `typedef void __cxxabiv1::__cxa_vec_ctor_return_type`

Functions

- `__extension__ typedef int __guard __cxxabiv1::__attribute__ ((mode(__DI__)))`

5.84.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<cxxabi.h>`.

5.85 debug.h File Reference

Namespaces

- [__gnu_debug](#)
- [std](#)
- [std::__debug](#)

Macros

- `#define __glibcxx_requires_cond(_Cond, _Msg)`
- `#define __glibcxx_requires_heap(_First, _Last)`
- `#define __glibcxx_requires_heap_pred(_First, _Last, _Pred)`
- `#define __glibcxx_requires_non_empty_range(_First, _Last)`
- `#define __glibcxx_requires_nonempty()`
- `#define __glibcxx_requires_partitioned_lower(_First, _Last, _Value)`
- `#define __glibcxx_requires_partitioned_lower_pred(_First, _Last, _Value, _Pred)`
- `#define __glibcxx_requires_partitioned_upper(_First, _Last, _Value)`
- `#define __glibcxx_requires_partitioned_upper_pred(_First, _Last, _Value, _Pred)`
- `#define __glibcxx_requires_sorted(_First, _Last)`
- `#define __glibcxx_requires_sorted_pred(_First, _Last, _Pred)`
- `#define __glibcxx_requires_sorted_set(_First1, _Last1, _First2)`
- `#define __glibcxx_requires_sorted_set_pred(_First1, _Last1, _First2, _Pred)`
- `#define __glibcxx_requires_string(_String)`
- `#define __glibcxx_requires_string_len(_String, _Len)`
- `#define __glibcxx_requires_subscript(_N)`
- `#define __glibcxx_requires_valid_range(_First, _Last)`
- `#define _GLIBCXX_DEBUG_ASSERT(_Condition)`
- `#define _GLIBCXX_DEBUG_ONLY(_Statement)`
- `#define _GLIBCXX_DEBUG_PEDASSERT(_Condition)`

5.85.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

5.86 debug_allocator.h File Reference

Classes

- class [__gnu_cxx::debug_allocator<_Alloc>](#)

Namespaces

- [__gnu_cxx](#)

Functions

- `template<typename _Alloc >
bool __gnu_cxx::operator!= (const debug_allocator< _Alloc > &__lhs, const debug_allocator< _Alloc > &__rhs)`

5.86.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

5.87 debug_fn_imps.hpp File Reference

5.87.1 Detailed Description

Contains an implementation class for a binary_heap.

5.88 debug_fn_imps.hpp File Reference

5.88.1 Detailed Description

Contains an implementation for binomial_heap_.

5.89 debug_fn_imps.hpp File Reference

5.89.1 Detailed Description

Contains an implementation class for a base of binomial heaps.

5.90 debug_fn_imps.hpp File Reference

5.90.1 Detailed Description

Contains an implementation class for bin_search_tree_.

5.91 `debug_fn_imps.hpp` File Reference

5.91.1 Detailed Description

Contains implementations of `cc_ht_map_`'s debug-mode functions.

5.92 `debug_fn_imps.hpp` File Reference

5.92.1 Detailed Description

Contains implementations of `gp_ht_map_`'s debug-mode functions.

5.93 `debug_fn_imps.hpp` File Reference

5.93.1 Detailed Description

Contains an implementation class for `left_child_next_sibling_heap_`.

5.94 `debug_fn_imps.hpp` File Reference

5.94.1 Detailed Description

Contains implementations of `cc_ht_map_`'s debug-mode functions.

5.95 `debug_fn_imps.hpp` File Reference

5.95.1 Detailed Description

Contains an implementation class for `ov_tree_`.

5.96 `debug_fn_imps.hpp` File Reference

5.96.1 Detailed Description

Contains an implementation class for a pairing heap.

5.97 `debug_fn_imps.hpp` File Reference

5.97.1 Detailed Description

Contains an implementation class for `pat_trie_`.

5.98 `debug_fn_imps.hpp` File Reference

5.98.1 Detailed Description

Contains an implementation for `rb_tree_`.

5.99 `debug_fn_imps.hpp` File Reference

5.99.1 Detailed Description

Contains an implementation for `rc_binomial_heap_`.

5.100 `debug_fn_imps.hpp` File Reference

5.100.1 Detailed Description

Contains an implementation class for `splay_tree_`.

5.101 `debug_fn_imps.hpp` File Reference

5.101.1 Detailed Description

Contains an implementation for `thin_heap_`.

5.102 `debug_map_base.hpp` File Reference

5.102.1 Detailed Description

Contains a debug-mode base for all maps.

5.103 `debug_no_store_hash_fn_imps.hpp` File Reference

5.103.1 Detailed Description

Contains implementations of `cc_ht_map_'s` debug-mode functions.

5.104 `debug_no_store_hash_fn_imps.hpp` File Reference

5.104.1 Detailed Description

Contains implementations of `gp_ht_map_'s` debug-mode functions.

5.105 `debug_store_hash_fn_imps.hpp` File Reference

5.105.1 Detailed Description

Contains implementations of `cc_ht_map_'s` debug-mode functions.

5.106 `debug_store_hash_fn_imps.hpp` File Reference

5.106.1 Detailed Description

Contains implementations of `gp_ht_map_'s` debug-mode functions.

5.107 `direct_mask_range_hashing_imp.hpp` File Reference

5.107.1 Detailed Description

Contains a range-hashing policy implementation

5.108 `direct_mod_range_hashing_imp.hpp` File Reference

5.108.1 Detailed Description

Contains a range-hashing policy implementation

5.109 `enable_special_members.h` File Reference

Classes

- struct [std::_Enable_copy_move<_Copy, _CopyAssignment, _Move, _MoveAssignment, _Tag >](#)
- struct [std::_Enable_default_constructor<_Switch, _Tag >](#)
- struct [std::_Enable_destructor<_Switch, _Tag >](#)
- struct [std::_Enable_special_members<_Default, _Destructor, _Copy, _CopyAssignment, _Move, _MoveAssignment, _Tag >](#)

Namespaces

- [std](#)

5.109.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly.

5.110 `enc_filebuf.h` File Reference

Classes

- class [__gnu_cxx::enc_filebuf<_CharT >](#)

Namespaces

- [__gnu_cxx](#)

5.110.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

5.111 `entry_cmp.hpp` File Reference

Classes

- struct [__gnu_pbds::detail::entry_cmp<_VTp, Cmp_Fn, _Alloc, No_Throw >](#)
- struct [__gnu_pbds::detail::entry_cmp<_VTp, Cmp_Fn, _Alloc, false >](#)
- struct [__gnu_pbds::detail::entry_cmp<_VTp, Cmp_Fn, _Alloc, false >::type](#)
- struct [__gnu_pbds::detail::entry_cmp<_VTp, Cmp_Fn, _Alloc, true >](#)

Namespaces

- [__gnu_pbds](#)

5.111.1 Detailed Description

Contains an implementation class for a binary_heap.

5.112 entry_list_fn_imps.hpp File Reference

5.112.1 Detailed Description

Contains implementations of cc_ht_map_'s entry-list related functions.

5.113 entry_metadata_base.hpp File Reference

Namespaces

- [__gnu_pbds](#)

5.113.1 Detailed Description

Contains an implementation for a list update map.

5.114 entry_pred.hpp File Reference

Classes

- struct [__gnu_pbds::detail::entry_pred](#)< _VTp, Pred, _Alloc, No_Throw >
- struct [__gnu_pbds::detail::entry_pred](#)< _VTp, Pred, _Alloc, false >
- struct [__gnu_pbds::detail::entry_pred](#)< _VTp, Pred, _Alloc, true >

Namespaces

- [__gnu_pbds](#)

5.114.1 Detailed Description

Contains an implementation class for a binary_heap.

5.115 `eq_by_less.hpp` File Reference

Classes

- [struct `__gnu_pbds::detail::eq_by_less< Key, Cmp_Fn >`](#)

Namespaces

- [__gnu_pbds](#)

5.115.1 Detailed Description

Contains an equivalence function.

5.116 `equally_split.h` File Reference

Namespaces

- [__gnu_parallel](#)

Functions

- `template<typename _DifferenceType , typename _OutputIterator >`
`_OutputIterator __gnu_parallel::__equally_split (_DifferenceType __n, _ThreadIndex __num_threads, _OutputIterator __s)`
- `template<typename _DifferenceType >`
`_DifferenceType __gnu_parallel::__equally_split_point (_DifferenceType __n, _ThreadIndex __num_threads, _ThreadIndex __thread_no)`

5.116.1 Detailed Description

This file is a GNU parallel extension to the Standard C++ Library.

5.117 `erase_fn_imps.hpp` File Reference

5.117.1 Detailed Description

Contains an implementation class for a binary_heap.

5.118 erase_fn_imps.hpp File Reference

5.118.1 Detailed Description

Contains an implementation class for a base of binomial heaps.

5.119 erase_fn_imps.hpp File Reference

5.119.1 Detailed Description

Contains an implementation class for bin_search_tree_.

5.120 erase_fn_imps.hpp File Reference

5.120.1 Detailed Description

Contains implementations of cc_ht_map_'s erase related functions.

5.121 erase_fn_imps.hpp File Reference

5.121.1 Detailed Description

Contains implementations of gp_ht_map_'s erase related functions.

5.122 erase_fn_imps.hpp File Reference

5.122.1 Detailed Description

Contains an implementation class for left_child_next_sibling_heap_.

5.123 erase_fn_imps.hpp File Reference

5.123.1 Detailed Description

Contains implementations of lu_map_.

5.124 `erase_fn_imps.hpp` File Reference

5.124.1 Detailed Description

Contains an implementation class for `ov_tree_`.

5.125 `erase_fn_imps.hpp` File Reference

5.125.1 Detailed Description

Contains an implementation class for a pairing heap.

5.126 `erase_fn_imps.hpp` File Reference

5.126.1 Detailed Description

Contains an implementation class for `pat_trie`.

5.127 `erase_fn_imps.hpp` File Reference

5.127.1 Detailed Description

Contains an implementation for `rb_tree_`.

5.128 `erase_fn_imps.hpp` File Reference

5.128.1 Detailed Description

Contains an implementation for `rc_binomial_heap_`.

5.129 `erase_fn_imps.hpp` File Reference

5.129.1 Detailed Description

Contains an implementation class for `splay_tree_`.

5.130 `erase_fn_imps.hpp` File Reference

5.130.1 Detailed Description

Contains an implementation for `thin_heap_`.

5.131 `erase_no_store_hash_fn_imps.hpp` File Reference

5.131.1 Detailed Description

Contains implementations of `cc_ht_map_`'s erase related functions, when the hash value is not stored.

5.132 `erase_no_store_hash_fn_imps.hpp` File Reference

5.132.1 Detailed Description

Contains implementations of `gp_ht_map_`'s erase related functions, when the hash value is not stored.

5.133 `erase_store_hash_fn_imps.hpp` File Reference

5.133.1 Detailed Description

Contains implementations of `cc_ht_map_`'s erase related functions, when the hash value is stored.

5.134 `erase_store_hash_fn_imps.hpp` File Reference

5.134.1 Detailed Description

Contains implementations of `gp_ht_map_`'s erase related functions, when the hash value is stored.

5.135 `error_constants.h` File Reference

Namespaces

- [std](#)

Enumerations

- enum `errc` {
`address_family_not_supported`, `address_in_use`, `address_not_available`, `already_connected`,
`argument_list_too_long`, `argument_out_of_domain`, `bad_address`, `bad_file_descriptor`,
`broken_pipe`, `connection_aborted`, `connection_already_in_progress`, `connection_refused`,
`connection_reset`, `cross_device_link`, `destination_address_required`, `device_or_resource_busy`,
`directory_not_empty`, `executable_format_error`, `file_exists`, `file_too_large`,
`filename_too_long`, `function_not_supported`, `host_unreachable`, `illegal_byte_sequence`,
`inappropriate_io_control_operation`, `interrupted`, `invalid_argument`, `invalid_seek`,
`io_error`, `is_a_directory`, `message_size`, `network_down`,
`network_reset`, `network_unreachable`, `no_buffer_space`, `no_child_process`,
`no_lock_available`, `no_message`, `no_protocol_option`, `no_space_on_device`,
`no_such_device_or_address`, `no_such_device`, `no_such_file_or_directory`, `no_such_process`,
`not_a_directory`, `not_a_socket`, `not_connected`, `not_enough_memory`,
`operation_in_progress`, `operation_not_permitted`, `operation_not_supported`, `operation_would_block`,
`permission_denied`, `protocol_not_supported`, `read_only_file_system`, `resource_deadlock_would_occur`,
`resource_unavailable_try_again`, `result_out_of_range`, `timed_out`, `too_many_files_open_in_system`,
`too_many_files_open`, `too_many_links`, `too_many_symbolic_link_levels`, `wrong_protocol_type` }

5.135.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<system_error>`.

5.136 `exception.hpp` File Reference

Classes

- struct [__gnu_pbds::container_error](#)
- struct [__gnu_pbds::insert_error](#)
- struct [__gnu_pbds::join_error](#)
- struct [__gnu_pbds::resize_error](#)

Namespaces

- [__gnu_pbds](#)

Functions

- void [__gnu_pbds::__throw_container_error\(\)](#)
- void [__gnu_pbds::__throw_insert_error\(\)](#)
- void [__gnu_pbds::__throw_join_error\(\)](#)
- void [__gnu_pbds::__throw_resize_error\(\)](#)

5.136.1 Detailed Description

Contains exception classes.

5.137 `exception_defines.h` File Reference

Macros

- `#define __catch(X)`
- `#define __throw_exception_again`
- `#define __try`

5.137.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<exception>`.

5.138 `exception_ptr.h` File Reference

Classes

- class [std::__exception_ptr::exception_ptr](#)

Namespaces

- [std](#)

Functions

- `template<typename _Ex >`
`exception_ptr std::copy_exception (_Ex __ex) noexcept 1`
- `exception_ptr std::current_exception () noexcept`
- `template<typename _Ex >`
`exception_ptr std::make_exception_ptr (_Ex __ex) noexcept`
- `bool std::__exception_ptr::operator!= (const exception_ptr &, const exception_ptr &) noexcept __attribute__((__pure__))`
- `bool std::__exception_ptr::operator== (const exception_ptr &, const exception_ptr &) noexcept __attribute__((__pure__))`
- `void std::rethrow_exception (exception_ptr) __attribute__((__noreturn__))`
- `void std::__exception_ptr::swap (exception_ptr &__lhs, exception_ptr &__rhs)`

5.138.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<exception>`.

5.139 extc++.h File Reference

5.139.1 Detailed Description

This is an implementation file for a precompiled header.

5.140 extptr_allocator.h File Reference

Classes

- class [__gnu_cxx::_ExtPtr_allocator<_Tp>](#)

Namespaces

- [__gnu_cxx](#)

Functions

- `template<typename _Tp>
void __gnu_cxx::swap (_ExtPtr_allocator<_Tp> &__larg, _ExtPtr_allocator<_Tp> &__rarg)`

5.140.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Author

Bob Walters

An example allocator which uses an alternative pointer type from bits/pointer.h. Supports test cases which confirm container support for alternative pointers.

5.141 features.h File Reference

Macros

- `#define _GLIBCXX_BAL_QUICKSORT`
- `#define _GLIBCXX_FIND_CONSTANT_SIZE_BLOCKS`
- `#define _GLIBCXX_FIND_EQUAL_SPLIT`
- `#define _GLIBCXX_FIND_GROWING_BLOCKS`
- `#define _GLIBCXX_MERGESORT`
- `#define _GLIBCXX_QUICKSORT`
- `#define _GLIBCXX_TREE_DYNAMIC_BALANCING`
- `#define _GLIBCXX_TREE_FULL_COPY`
- `#define _GLIBCXX_TREE_INITIAL_SPLITTING`

5.141.1 Detailed Description

Defines on whether to include algorithm variants.

Less variants reduce executable size and compile time. This file is a GNU parallel extension to the Standard C++ Library.

5.141.2 Macro Definition Documentation

5.141.2.1 `#define _GLIBCXX_BAL_QUICKSORT`

Include parallel dynamically load-balanced quicksort.

See also

`__gnu_parallel::_Settings::sort_algorithm`

Definition at line 55 of file features.h.

5.141.2.2 `#define _GLIBCXX_FIND_CONSTANT_SIZE_BLOCKS`

Include the equal-sized blocks variant for `std::find`.

See also

`__gnu_parallel::_Settings::find_algorithm`

Definition at line 67 of file features.h.

5.141.2.3 `#define _GLIBCXX_FIND_EQUAL_SPLIT`

Include the equal splitting variant for `std::find`.

See also

`__gnu_parallel::_Settings::find_algorithm`

Definition at line 74 of file features.h.

5.141.2.4 `#define _GLIBCXX_FIND_GROWING_BLOCKS`

Include the growing blocks variant for `std::find`.

See also

`__gnu_parallel::_Settings::find_algorithm`

Definition at line 61 of file features.h.

5.141.2.5 `#define _GLIBCXX_MERGESORT`

Include parallel multi-way mergesort.

See also

`__gnu_parallel::Settings::sort_algorithm`

Definition at line 41 of file features.h.

5.141.2.6 `#define _GLIBCXX_QUICKSORT`

Include parallel unbalanced quicksort.

See also

`__gnu_parallel::Settings::sort_algorithm`

Definition at line 48 of file features.h.

5.141.2.7 `#define _GLIBCXX_TREE_DYNAMIC_BALANCING`

Include the dynamic balancing variant for `_Rb_tree::insert_unique(_Iter beg, _Iter __end)`.

See also

`__gnu_parallel::_Rb_tree`

Definition at line 91 of file features.h.

5.141.2.8 `#define _GLIBCXX_TREE_FULL_COPY`

In order to sort the input sequence of `_Rb_tree::insert_unique(_Iter beg, _Iter __end)` a full copy of the input elements is done.

See also

`__gnu_parallel::_Rb_tree`

Definition at line 100 of file features.h.

5.141.2.9 `#define _GLIBCXX_TREE_INITIAL_SPLITTING`

Include the initial splitting variant for `_Rb_tree::insert_unique(_Iter beg, _Iter __end)`.

See also

`__gnu_parallel::_Rb_tree`

Definition at line 83 of file features.h.

5.142 fenv.h File Reference

5.142.1 Detailed Description

This is a Standard C++ Library header.

5.143 find.h File Reference

Namespaces

- [__gnu_parallel](#)

Functions

- `template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector >
std::pair<_RAIter1, _RAIter2 > __gnu_parallel::__find_template (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred, _Selector __selector)`
- `template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector >
std::pair<_RAIter1, _RAIter2 > __gnu_parallel::__find_template (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred, _Selector __selector, equal_split_tag)`
- `template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector >
std::pair<_RAIter1, _RAIter2 > __gnu_parallel::__find_template (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred, _Selector __selector, growing_blocks_tag)`
- `template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector >
std::pair<_RAIter1, _RAIter2 > __gnu_parallel::__find_template (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred, _Selector __selector, constant_size_blocks_tag)`

5.143.1 Detailed Description

Parallel implementation base for `std::find()`, `std::equal()` and related functions. This file is a GNU parallel extension to the Standard C++ Library.

5.144 find_fn_imps.hpp File Reference

5.144.1 Detailed Description

Contains an implementation class for a binary_heap.

5.145 find_fn_imps.hpp File Reference

5.145.1 Detailed Description

Contains an implementation class for a base of binomial heaps.

5.146 find_fn_imps.hpp File Reference

5.146.1 Detailed Description

Contains an implementation class for bin_search_tree_.

5.147 find_fn_imps.hpp File Reference

5.147.1 Detailed Description

Contains implementations of cc_ht_map_'s find related functions.

5.148 find_fn_imps.hpp File Reference

5.148.1 Detailed Description

Contains implementations of gp_ht_map_'s find related functions.

5.149 find_fn_imps.hpp File Reference

5.149.1 Detailed Description

Contains implementations of lu_map_.

5.150 find_fn_imps.hpp File Reference

5.150.1 Detailed Description

Contains an implementation class for a pairing heap.

5.151 find_fn_imps.hpp File Reference

5.151.1 Detailed Description

Contains an implementation class for pat_trie.

5.152 find_fn_imps.hpp File Reference

5.152.1 Detailed Description

Contains an implementation for rb_tree_.

5.153 find_fn_imps.hpp File Reference

5.153.1 Detailed Description

Contains an implementation class for splay_tree_.

5.154 find_fn_imps.hpp File Reference

5.154.1 Detailed Description

Contains an implementation for thin_heap_.

5.155 find_no_store_hash_fn_imps.hpp File Reference

5.155.1 Detailed Description

Contains implementations of gp_ht_map_'s find related functions, when the hash value is not stored.

5.156 find_selectors.h File Reference

Classes

- struct [__gnu_parallel::__adjacent_find_selector](#)
- struct [__gnu_parallel::__find_first_of_selector<_FIterator>](#)
- struct [__gnu_parallel::__find_if_selector](#)
- struct [__gnu_parallel::__generic_find_selector](#)
- struct [__gnu_parallel::__mismatch_selector](#)

Namespaces

- [__gnu_parallel](#)

5.156.1 Detailed Description

_Function objects representing different tasks to be plugged into the parallel find algorithm. This file is a GNU parallel extension to the Standard C++ Library.

5.157 find_store_hash_fn_imps.hpp File Reference

5.157.1 Detailed Description

Contains implementations of cc_ht_map_'s find related functions, when the hash value is stored.

5.158 find_store_hash_fn_imps.hpp File Reference

5.158.1 Detailed Description

Contains implementations of gp_ht_map_'s insert related functions, when the hash value is stored.

5.159 for_each.h File Reference

Namespaces

- [__gnu_parallel](#)

Functions

- `template<typename _Iter , typename _UserOp , typename _Functionality , typename _Red , typename _Result >
_UserOp __gnu_parallel::__for_each_template_random_access (_Iter __begin, _Iter __end, _UserOp __user↵
_op, _Functionality &__functionality, _Red __reduction, _Result __reduction_start, _Result &__output, typename
std::iterator_traits< _Iter >::difference_type __bound, _Parallelism __parallelism_tag)`

5.159.1 Detailed Description

Main interface for embarrassingly parallel functions.

The explicit implementation are in other header files, like workstealing.h, par_loop.h, omp_loop.h, and omp_loop_↵
static.h. This file is a GNU parallel extension to the Standard C++ Library.

5.160 `for_each_selectors.h` File Reference

Classes

- struct `__gnu_parallel::__accumulate_binop_reduct<_BinOp>`
- struct `__gnu_parallel::__accumulate_selector<_It>`
- struct `__gnu_parallel::__adjacent_difference_selector<_It>`
- struct `__gnu_parallel::__count_if_selector<_It, _Diff>`
- struct `__gnu_parallel::__count_selector<_It, _Diff>`
- struct `__gnu_parallel::__fill_selector<_It>`
- struct `__gnu_parallel::__for_each_selector<_It>`
- struct `__gnu_parallel::__generate_selector<_It>`
- struct `__gnu_parallel::__generic_for_each_selector<_It>`
- struct `__gnu_parallel::__identity_selector<_It>`
- struct `__gnu_parallel::__inner_product_selector<_It, _It2, _Tp>`
- struct `__gnu_parallel::__max_element_reduct<_Compare, _It>`
- struct `__gnu_parallel::__min_element_reduct<_Compare, _It>`
- struct `__gnu_parallel::__replace_if_selector<_It, _Op, _Tp>`
- struct `__gnu_parallel::__replace_selector<_It, _Tp>`
- struct `__gnu_parallel::__transform1_selector<_It>`
- struct `__gnu_parallel::__transform2_selector<_It>`
- struct `__gnu_parallel::__DummyReduct`
- struct `__gnu_parallel::__Nothing`

Namespaces

- `__gnu_parallel`

5.160.1 Detailed Description

Functors representing different tasks to be plugged into the generic parallelization methods for embarrassingly parallel functions. This file is a GNU parallel extension to the Standard C++ Library.

5.161 `formatter.h` File Reference

Classes

- class `__gnu_debug::__Safe_iterator<_Iterator, _Sequence>`
- class `__gnu_debug::__Safe_local_iterator<_Iterator, _Sequence>`
- class `__gnu_debug::__Safe_sequence<_Sequence>`

Namespaces

- `__gnu_debug`

Enumerations

- enum `_Debug_msg_id` {
`__msg_valid_range`, `__msg_insert_singular`, `__msg_insert_different`, `__msg_erase_bad`,
`__msg_erase_different`, `__msg_subscript_oob`, `__msg_empty`, `__msg_unpartitioned`,
`__msg_unpartitioned_pred`, `__msg_unsorted`, `__msg_unsorted_pred`, `__msg_not_heap`,
`__msg_not_heap_pred`, `__msg_bad_bitset_write`, `__msg_bad_bitset_read`, `__msg_bad_bitset_flip`,
`__msg_self_splice`, `__msg_splice_alloc`, `__msg_splice_bad`, `__msg_splice_other`,
`__msg_splice_overlap`, `__msg_init_singular`, `__msg_init_copy_singular`, `__msg_init_const_singular`,
`__msg_copy_singular`, `__msg_bad_deref`, `__msg_bad_inc`, `__msg_bad_dec`,
`__msg_iter_subscript_oob`, `__msg_advance_oob`, `__msg_retreat_oob`, `__msg_iter_compare_bad`,
`__msg_compare_different`, `__msg_iter_order_bad`, `__msg_order_different`, `__msg_distance_bad`,
`__msg_distance_different`, `__msg_deref_istream`, `__msg_inc_istream`, `__msg_output_ostream`,
`__msg_deref_istreambuf`, `__msg_inc_istreambuf`, `__msg_insert_after_end`, `__msg_erase_after_bad`,
`__msg_valid_range2`, `__msg_local_iter_compare_bad`, `__msg_non_empty_range`, `__msg_self_move_↵`
`assign`,
`__msg_bucket_index_oob`, `__msg_valid_load_factor`, `__msg_equal_allocs`, `__msg_insert_range_from_↵`
`_self` }

Functions

- template<typename `_Iterator` >
`bool __gnu_debug::__check_singular` (const `_Iterator` &)

5.161.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

5.162 `forward_list.h` File Reference

Classes

- struct `std::_Fwd_list_base<_Tp, _Alloc>`
- struct `std::_Fwd_list_const_iterator<_Tp>`
- struct `std::_Fwd_list_iterator<_Tp>`
- struct `std::_Fwd_list_node<_Tp>`
- struct `std::_Fwd_list_node_base`
- class `std::forward_list<_Tp, _Alloc>`

Namespaces

- `std`

Functions

- `template<typename _Tp >`
`bool std::operator!= (const _Fwd_list_iterator< _Tp > &__x, const _Fwd_list_const_iterator< _Tp > &__y) noexcept`
- `template<typename _Tp, typename _Alloc >`
`bool std::operator!= (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Alloc >`
`bool std::operator< (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Alloc >`
`bool std::operator<= (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp >`
`bool std::operator== (const _Fwd_list_iterator< _Tp > &__x, const _Fwd_list_const_iterator< _Tp > &__y) noexcept`
- `template<typename _Tp, typename _Alloc >`
`bool std::operator== (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Alloc >`
`bool std::operator> (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Alloc >`
`bool std::operator>= (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Alloc >`
`void std::swap (forward_list< _Tp, _Alloc > &__lx, forward_list< _Tp, _Alloc > &__ly)`

5.162.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<forward_list>`.

5.163 funtexcept.h File Reference

Namespaces

- `std`

Functions

- `void std::__throw_bad_alloc (void) __attribute__((__noreturn__))`
- `void std::__throw_bad_cast (void) __attribute__((__noreturn__))`
- `void std::__throw_bad_exception (void) __attribute__((__noreturn__))`
- `void std::__throw_bad_function_call () __attribute__((__noreturn__))`
- `void std::__throw_bad_typeid (void) __attribute__((__noreturn__))`
- `void std::__throw_domain_error (const char *) __attribute__((__noreturn__))`
- `void std::__throw_future_error (int) __attribute__((__noreturn__))`
- `void std::__throw_invalid_argument (const char *) __attribute__((__noreturn__))`
- `void std::__throw_ios_failure (const char *) __attribute__((__noreturn__))`
- `void std::__throw_length_error (const char *) __attribute__((__noreturn__))`
- `void std::__throw_logic_error (const char *) __attribute__((__noreturn__))`
- `void std::__throw_out_of_range (const char *) __attribute__((__noreturn__))`

- void **std::__throw_out_of_range_fmt** (const char *,...) __attribute__((__noreturn__)) __attribute__((__format__(__printf__,
- void **std::__throw_overflow_error** (const char *) __attribute__((__noreturn__))
- void **std::__throw_range_error** (const char *) __attribute__((__noreturn__))
- void void **std::__throw_runtime_error** (const char *) __attribute__((__noreturn__))
- void **std::__throw_system_error** (int) __attribute__((__noreturn__))
- void **std::__throw_underflow_error** (const char *) __attribute__((__noreturn__))

5.163.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<exception>`.

This header provides support for `-fno-exceptions`.

5.164 functional_hash.h File Reference

Classes

- struct [std::hash< _Tp >](#)
- struct [std::hash< _Tp * >](#)
- struct [std::hash< bool >](#)
- struct [std::hash< char >](#)
- struct [std::hash< char16_t >](#)
- struct [std::hash< char32_t >](#)
- struct [std::hash< double >](#)
- struct [std::hash< float >](#)
- struct [std::hash< int >](#)
- struct [std::hash< long >](#)
- struct [std::hash< long double >](#)
- struct [std::hash< long long >](#)
- struct [std::hash< short >](#)
- struct [std::hash< signed char >](#)
- struct [std::hash< unsigned char >](#)
- struct [std::hash< unsigned int >](#)
- struct [std::hash< unsigned long >](#)
- struct [std::hash< unsigned long long >](#)
- struct [std::hash< unsigned short >](#)
- struct [std::hash< wchar_t >](#)

Namespaces

- [std](#)

Macros

- `#define _Cxx_hashtable_define_trivial_hash(_Tp)`

5.164.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<functional>`.

5.165 functions.h File Reference

Classes

- class [__gnu_debug::__Safe_iterator<_Iterator, _Sequence>](#)
- class [__gnu_debug::__Safe_local_iterator<_Iterator, _Sequence>](#)

Namespaces

- [__gnu_debug](#)

Functions

- template<typename _Iterator >
_Siter_base<_Iterator>::iterator_type [__gnu_debug::__base](#) (_Iterator __it)
- template<typename _Iterator >
bool [__gnu_debug::__check_dereferenceable](#) (const _Iterator &)
- template<typename _Tp >
bool [__gnu_debug::__check_dereferenceable](#) (const _Tp *__ptr)
- template<typename _Iterator, typename _Sequence >
bool [__gnu_debug::__check_dereferenceable](#) (const __Safe_iterator<_Iterator, _Sequence> &__x)
- template<typename _Iterator, typename _Sequence >
bool [__gnu_debug::__check_dereferenceable](#) (const __Safe_local_iterator<_Iterator, _Sequence> &__x)
- template<typename _ForwardIterator, typename _Tp >
bool [__gnu_debug::__check_partitioned_lower](#) (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)
- template<typename _ForwardIterator, typename _Tp, typename _Pred >
bool [__gnu_debug::__check_partitioned_lower](#) (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value, _Pred __pred)
- template<typename _ForwardIterator, typename _Tp >
bool [__gnu_debug::__check_partitioned_upper](#) (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)
- template<typename _ForwardIterator, typename _Tp, typename _Pred >
bool [__gnu_debug::__check_partitioned_upper](#) (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value, _Pred __pred)
- template<typename _Iterator >
bool [__gnu_debug::__check_singular](#) (const _Iterator &)
- template<typename _Tp >
bool [__gnu_debug::__check_singular](#) (const _Tp *__ptr)
- bool [__gnu_debug::__check_singular_aux](#) (const void *)
- template<typename _InputIterator >
bool [__gnu_debug::__check_sorted](#) (const _InputIterator &__first, const _InputIterator &__last)

- `template<typename _InputIterator, typename _Predicate >`
`bool __gnu_debug::__check_sorted (const _InputIterator &__first, const _InputIterator &__last, _Predicate __pred)`
- `template<typename _InputIterator >`
`bool __gnu_debug::__check_sorted_aux (const _InputIterator &, const _InputIterator &, std::input_iterator_tag)`
- `template<typename _ForwardIterator >`
`bool __gnu_debug::__check_sorted_aux (_ForwardIterator __first, _ForwardIterator __last, std::forward_iterator_tag)`
- `template<typename _InputIterator, typename _Predicate >`
`bool __gnu_debug::__check_sorted_aux (const _InputIterator &, const _InputIterator &, _Predicate, std::input_iterator_tag)`
- `template<typename _ForwardIterator, typename _Predicate >`
`bool __gnu_debug::__check_sorted_aux (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, std::forward_iterator_tag)`
- `template<typename _InputIterator1, typename _InputIterator2 >`
`bool __gnu_debug::__check_sorted_set (const _InputIterator1 &__first, const _InputIterator1 &__last, const _InputIterator2 &)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Predicate >`
`bool __gnu_debug::__check_sorted_set (const _InputIterator1 &__first, const _InputIterator1 &__last, const _InputIterator2 &, _Predicate __pred)`
- `template<typename _InputIterator >`
`bool __gnu_debug::__check_sorted_set_aux (const _InputIterator &__first, const _InputIterator &__last, std::true_type)`
- `template<typename _InputIterator >`
`bool __gnu_debug::__check_sorted_set_aux (const _InputIterator &, const _InputIterator &, std::false_type)`
- `template<typename _InputIterator, typename _Predicate >`
`bool __gnu_debug::__check_sorted_set_aux (const _InputIterator &__first, const _InputIterator &__last, _Predicate __pred, std::true_type)`
- `template<typename _InputIterator, typename _Predicate >`
`bool __gnu_debug::__check_sorted_set_aux (const _InputIterator &, const _InputIterator &, _Predicate, std::false_type)`
- `template<typename _CharT, typename _Integer >`
`const _CharT * __gnu_debug::__check_string (const _CharT * __s, const _Integer & __n __attribute__\(\(__unused__\)\))`
- `template<typename _CharT >`
`const _CharT * __gnu_debug::__check_string (const _CharT * __s)`
- `template<typename _InputIterator >`
`_InputIterator __gnu_debug::__check_valid_range (const _InputIterator &__first, const _InputIterator &__last __attribute__\(\(__unused__\)\))`
- `template<typename _Iterator, typename _Sequence, typename _InputIterator >`
`bool __gnu_debug::__foreign_iterator (const _Safe_iterator< _Iterator, _Sequence > &__it, _InputIterator __other, _InputIterator __other_end)`
- `template<typename _Iterator, typename _Sequence, typename _Integral >`
`bool __gnu_debug::__foreign_iterator_aux (const _Safe_iterator< _Iterator, _Sequence > &, _Integral, std::true_type)`
- `template<typename _Iterator, typename _Sequence, typename _InputIterator >`
`bool __gnu_debug::__foreign_iterator_aux (const _Safe_iterator< _Iterator, _Sequence > &__it, _InputIterator __other, _InputIterator __other_end, std::false_type)`
- `template<typename _Iterator, typename _Sequence, typename _OtherIterator >`
`bool __gnu_debug::__foreign_iterator_aux2 (const _Safe_iterator< _Iterator, _Sequence > &__it, const _Safe_iterator< _OtherIterator, _Sequence > &__other, const _Safe_iterator< _OtherIterator, _Sequence > &)`
- `template<typename _Iterator, typename _Sequence, typename _OtherIterator, typename _OtherSequence >`
`bool __gnu_debug::__foreign_iterator_aux2 (const _Safe_iterator< _Iterator, _Sequence > &__it, const _Safe_iterator< _OtherIterator, _OtherSequence > &, const _Safe_iterator< _OtherIterator, _OtherSequence > &)`

- `template<typename _Iterator, typename _Sequence, typename _InputIterator >`
`bool __gnu_debug::__foreign_iterator_aux2 (const _Safe_iterator< _Iterator, _Sequence > &__it, const _InputIterator &__other, const _InputIterator &__other_end)`
- `template<typename _Iterator, typename _Sequence, typename _InputIterator >`
`bool __gnu_debug::__foreign_iterator_aux3 (const _Safe_iterator< _Iterator, _Sequence > &__it, const _InputIterator &__other, const _InputIterator &__other_end, std::__true_type)`
- `template<typename _Iterator, typename _Sequence, typename _InputIterator >`
`bool __gnu_debug::__foreign_iterator_aux3 (const _Safe_iterator< _Iterator, _Sequence > &, const _InputIterator &, const _InputIterator &, std::__false_type)`
- `template<typename _Iterator, typename _Sequence >`
`bool __gnu_debug::__foreign_iterator_aux4 (const _Safe_iterator< _Iterator, _Sequence > &__it, const typename _Sequence::value_type *__other)`
- `template<typename _Iterator, typename _Sequence >`
`bool __gnu_debug::__foreign_iterator_aux4 (const _Safe_iterator< _Iterator, _Sequence > &,...)`
- `template<typename _InputIterator >`
`bool __gnu_debug::__valid_range (const _InputIterator &__first, const _InputIterator &__last)`
- `template<typename _Iterator, typename _Sequence >`
`bool __gnu_debug::__valid_range (const _Safe_iterator< _Iterator, _Sequence > &__first, const _Safe_iterator< _Iterator, _Sequence > &__last)`
- `template<typename _Iterator, typename _Sequence >`
`bool __gnu_debug::__valid_range (const _Safe_local_iterator< _Iterator, _Sequence > &__first, const _Safe_local_iterator< _Iterator, _Sequence > &__last)`
- `template<typename _Integral >`
`bool __gnu_debug::__valid_range_aux (const _Integral &, const _Integral &, std::__true_type)`
- `template<typename _InputIterator >`
`bool __gnu_debug::__valid_range_aux (const _InputIterator &__first, const _InputIterator &__last, std::__false_type)`
- `template<typename _RandomAccessIterator >`
`bool __gnu_debug::__valid_range_aux2 (const _RandomAccessIterator &__first, const _RandomAccessIterator &__last, std::random_access_iterator_tag)`
- `template<typename _InputIterator >`
`bool __gnu_debug::__valid_range_aux2 (const _InputIterator &, const _InputIterator &, std::input_iterator_tag)`

5.165.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

5.166 gp_ht_map.hpp File Reference

Classes

- class `__gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >`

Namespaces

- `__gnu_pbds`

Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_GEN_POS`
- `#define PB_DS_GP_HASH_NAME`
- `#define PB_DS_GP_HASH_TRAITS_BASE`
- `#define PB_DS_HASH_EQ_FN_C_DEC`
- `#define PB_DS_RANGED_PROBE_FN_C_DEC`

Variables

- `empty_entry_status`
- `erased_entry_status`
- `valid_entry_status`

5.166.1 Detailed Description

Contains an implementation class for general probing hash.

5.167 `gslice.h` File Reference

Classes

- class [`std::gslice`](#)

Namespaces

- [`std`](#)

5.167.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<valarray>`.

5.168 `gslice_array.h` File Reference

Classes

- class [`std::gslice_array<_Tp>`](#)

Namespaces

- [std](#)

Macros

- `#define _DEFINE_VALARRAY_OPERATOR(_Op, _Name)`

5.168.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<valarray>`.

5.169 hash_bytes.h File Reference

Namespaces

- [std](#)

Functions

- `size_t std::Fnv_hash_bytes(const void *__ptr, size_t __len, size_t __seed)`
- `size_t std::Hash_bytes(const void *__ptr, size_t __len, size_t __seed)`

5.169.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<functional>`.

5.170 hash_eq_fn.hpp File Reference

Classes

- `struct __gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, Store_Hash >`
- `struct __gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, false >`
- `struct __gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, true >`

Namespaces

- [__gnu_pbds](#)

5.170.1 Detailed Description

Contains 2 equivalence functions, one employing a hash value, and one ignoring it.

5.171 `hash_exponential_size_policy_imp.hpp` File Reference

5.171.1 Detailed Description

Contains a resize size policy implementation.

5.172 `hash_fun.h` File Reference

Namespaces

- [__gnu_cxx](#)

Functions

- `size_t __gnu_cxx::__stl_hash_string` (const char *__s)

5.172.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

5.173 `hash_load_check_resize_trigger_imp.hpp` File Reference

Macros

- `#define PB_DS_ASSERT_VALID(X)`
- `#define PB_DS_ASSERT_VALID(X)`

5.173.1 Detailed Description

Contains a resize trigger implementation.

5.174 `hash_load_check_resize_trigger_size_base.hpp` File Reference

Classes

- class [__gnu_pbds::detail::hash_load_check_resize_trigger_size_base< Size_Type, Hold_Size >](#)
- class [__gnu_pbds::detail::hash_load_check_resize_trigger_size_base< Size_Type, true >](#)

Namespaces

- [__gnu_pbds](#)

5.174.1 Detailed Description

Contains an base holding size for some resize policies.

5.175 hash_policy.hpp File Reference

Classes

- class [__gnu_pbds::cc_hash_max_collision_check_resize_trigger](#)< External_Load_Access, Size_Type >
- class [__gnu_pbds::direct_mask_range_hashing](#)< Size_Type >
- class [__gnu_pbds::direct_mod_range_hashing](#)< Size_Type >
- class [__gnu_pbds::hash_exponential_size_policy](#)< Size_Type >
- class [__gnu_pbds::hash_load_check_resize_trigger](#)< External_Load_Access, Size_Type >
- class [__gnu_pbds::hash_prime_size_policy](#)
- class [__gnu_pbds::hash_standard_resize_policy](#)< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >
- class [__gnu_pbds::linear_probe_fn](#)< Size_Type >
- class [__gnu_pbds::quadratic_probe_fn](#)< Size_Type >

Namespaces

- [__gnu_pbds](#)

Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_SIZE_BASE_C_DEC`

Enumerations

- enum { **num_distinct_sizes_32_bit**, **num_distinct_sizes_64_bit**, **num_distinct_sizes** }

Variables

- static const std::size_t **__gnu_pbds::detail::g_a_sizes** [num_distinct_sizes_64_bit]

5.175.1 Detailed Description

Contains hash-related policies.

5.176 hash_prime_size_policy_imp.hpp File Reference

Enumerations

- enum { **num_distinct_sizes_32_bit**, **num_distinct_sizes_64_bit**, **num_distinct_sizes** }

Variables

- static const std::size_t **detail::g_a_sizes** [num_distinct_sizes_64_bit]

5.176.1 Detailed Description

Contains a resize size policy implementation.

5.177 hash_standard_resize_policy_imp.hpp File Reference

5.177.1 Detailed Description

Contains a resize policy implementation.

5.178 hashtable.h File Reference

Classes

- class [std::_Hashtable<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits>](#)

Namespaces

- [std](#)

Typedefs

- `template<typename _Tp, typename _Hash >`
`using std::__cache_default = __not_< __and_< __is_fast_hash< _Hash >, __detail::__is_noexcept_hash<`
`_Tp, _Hash >>>`

5.178.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<unordered_map>` or `<unordered_set>`.

5.179 hashtable.h File Reference

Namespaces

- [`__gnu_cxx`](#)

Enumerations

- `enum { _S_num_primes }`

Functions

- `unsigned long __gnu_cxx::__stl_next_prime (unsigned long __n)`
- `template<class _Val, class _Key, class _HF, class _Ex, class _Eq, class _All >`
`bool __gnu_cxx::operator!= (const hashtable< _Val, _Key, _HF, _Ex, _Eq, _All > &__ht1, const hashtable<`
`_Val, _Key, _HF, _Ex, _Eq, _All > &__ht2)`
- `template<class _Val, class _Key, class _HF, class _Ex, class _Eq, class _All >`
`bool __gnu_cxx::operator== (const hashtable< _Val, _Key, _HF, _Ex, _Eq, _All > &__ht1, const hashtable<`
`_Val, _Key, _HF, _Ex, _Eq, _All > &__ht2)`
- `template<class _Val, class _Key, class _HF, class _Extract, class _EqKey, class _All >`
`void __gnu_cxx::swap (hashtable< _Val, _Key, _HF, _Extract, _EqKey, _All > &__ht1, hashtable< _Val, _Key,`
`_HF, _Extract, _EqKey, _All > &__ht2)`

5.179.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

5.180 hashtable_policy.h File Reference

Classes

- struct std::__detail::__Default_ranged_hash
- struct std::__detail::__Equal_helper< _Key, _Value, _ExtractKey, _Equal, _HashCodeType, __cache_hash_code >
- struct std::__detail::__Equal_helper< _Key, _Value, _ExtractKey, _Equal, _HashCodeType, false >
- struct std::__detail::__Equal_helper< _Key, _Value, _ExtractKey, _Equal, _HashCodeType, true >
- struct std::__detail::__Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, _Unique_keys >
- struct std::__detail::__Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false >
- struct std::__detail::__Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true >
- struct std::__detail::__Equality_base
- struct std::__detail::__Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache_hash_code >
- struct std::__detail::__Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Default_ranged_hash, false >
- struct std::__detail::__Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Default_ranged_hash, true >
- struct std::__detail::__Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, false >
- struct std::__detail::__Hash_node< _Value, _Cache_hash_code >
- struct std::__detail::__Hash_node< _Value, false >
- struct std::__detail::__Hash_node< _Value, true >
- struct std::__detail::__Hash_node_base
- struct std::__detail::__Hash_node_value_base< _Value >
- struct std::__detail::__Hashtable_alloc< _NodeAlloc >
- struct std::__detail::__Hashtable_alloc< _NodeAlloc >
- struct std::__detail::__Hashtable_base< _Key, _Value, _ExtractKey, _Equal, _H1, _H2, _Hash, _Traits >
- struct std::__detail::__Hashtable_base< _Key, _Value, _ExtractKey, _Equal, _H1, _H2, _Hash, _Traits >
- struct std::__detail::__Hashtable_ebo_helper< _Nm, _Tp, __use_ebo >
- struct std::__detail::__Hashtable_ebo_helper< _Nm, _Tp, false >
- struct std::__detail::__Hashtable_ebo_helper< _Nm, _Tp, true >
- struct std::__detail::__Hashtable_traits< _Cache_hash_code, _Constant_iterators, _Unique_keys >
- struct std::__detail::__Insert< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, __ Traits, _Constant_iterators, _Unique_keys >
- struct std::__detail::__Insert< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, __ Traits, false, _Unique_keys >
- struct std::__detail::__Insert< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, __ Traits, true, false >
- struct std::__detail::__Insert< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, __ Traits, true, true >
- struct std::__detail::__Insert_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >
- struct std::__detail::__Local_const_iterator< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __constant_iterators, __cache >
- struct std::__detail::__Local_iterator< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __constant_iterators, __cache >
- struct std::__detail::__Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache_hash_code >
- struct std::__detail::__Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, true >
- struct std::__detail::__Map_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, _Unique_keys >

- `struct std::__detail::Map_base<_Key, _Pair, _Alloc, _Select1st, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false>`
- `struct std::__detail::Map_base<_Key, _Pair, _Alloc, _Select1st, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true>`
- `struct std::__detail::Mod_range_hashing`
- `struct std::__detail::Node_const_iterator<_Value, __constant_iterators, __cache>`
- `struct std::__detail::Node_iterator<_Value, __constant_iterators, __cache>`
- `struct std::__detail::Node_iterator_base<_Value, _Cache_hash_code>`
- `struct std::__detail::Prime_rehash_policy`
- `struct std::__detail::Rehash_base<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits>`
- `struct std::__detail::Rehash_base<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _Prime_rehash_policy, _Traits>`
- `class std::_Hashtable<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits>`

Namespaces

- `std`
- `std::__detail`

Typedefs

- `template<typename _Key, typename _Value, typename _ExtractKey, typename _H1, typename _H2, typename _Hash>
using std::__detail::__hash_code_for_local_iter = _Hash_code_storage<_Hash_code_base<_Key, _Value, _ExtractKey, _H1, _H2, _Hash, false>>`

Functions

- `template<class _Iterator>
std::iterator_traits<_Iterator>::difference_type std::__detail::__distance_fw(_Iterator __first, _Iterator __last, std::input_iterator_tag)`
- `template<class _Iterator>
std::iterator_traits<_Iterator>::difference_type std::__detail::__distance_fw(_Iterator __first, _Iterator __last, std::forward_iterator_tag)`
- `template<class _Iterator>
std::iterator_traits<_Iterator>::difference_type std::__detail::__distance_fw(_Iterator __first, _Iterator __last)`
- `template<typename _Value, bool _Cache_hash_code>
bool std::__detail::operator!= (const _Node_iterator_base<_Value, _Cache_hash_code> &__x, const _Node_iterator_base<_Value, _Cache_hash_code> &__y) noexcept`
- `template<typename _Key, typename _Value, typename _ExtractKey, typename _H1, typename _H2, typename _Hash, bool __cache>
bool std::__detail::operator!= (const _Local_iterator_base<_Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache> &__x, const _Local_iterator_base<_Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache> &__y)`
- `template<typename _Value, bool _Cache_hash_code>
bool std::__detail::operator== (const _Node_iterator_base<_Value, _Cache_hash_code> &__x, const _Node_iterator_base<_Value, _Cache_hash_code> &__y) noexcept`
- `template<typename _Key, typename _Value, typename _ExtractKey, typename _H1, typename _H2, typename _Hash, bool __cache>
bool std::__detail::operator== (const _Local_iterator_base<_Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache> &__x, const _Local_iterator_base<_Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache> &__y)`

5.180.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<unordered_map>` or `<unordered_set>`.

5.181 `indirect_array.h` File Reference

Classes

- class `std::indirect_array<_Tp>`

Namespaces

- `std`

Macros

- `#define _DEFINE_VALARRAY_OPERATOR(_Op, _Name)`

5.181.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<valarray>`.

5.182 `info_fn_imps.hpp` File Reference

5.182.1 Detailed Description

Contains an implementation class for a `binary_heap`.

5.183 `info_fn_imps.hpp` File Reference

5.183.1 Detailed Description

Contains an implementation class for `bin_search_tree_`.

5.184 `info_fn_imps.hpp` File Reference

5.184.1 Detailed Description

Contains implementations of `cc_ht_map_`'s entire container info related functions.

5.185 info_fn_imps.hpp File Reference

5.185.1 Detailed Description

Contains implementations of gp_ht_map_'s entire container info related functions.

5.186 info_fn_imps.hpp File Reference

5.186.1 Detailed Description

Contains an implementation class for left_child_next_sibling_heap_.

5.187 info_fn_imps.hpp File Reference

5.187.1 Detailed Description

Contains implementations of lu_map_.

5.188 info_fn_imps.hpp File Reference

5.188.1 Detailed Description

Contains an implementation class for ov_tree_.

5.189 info_fn_imps.hpp File Reference

5.189.1 Detailed Description

Contains an implementation class for pat_trie.

5.190 info_fn_imps.hpp File Reference

5.190.1 Detailed Description

Contains an implementation for rb_tree_.

5.191 info_fn_imps.hpp File Reference

5.191.1 Detailed Description

Contains an implementation.

5.192 insert_fn_imps.hpp File Reference

5.192.1 Detailed Description

Contains an implementation class for a binary_heap.

5.193 insert_fn_imps.hpp File Reference

5.193.1 Detailed Description

Contains an implementation class for a base of binomial heaps.

5.194 insert_fn_imps.hpp File Reference

5.194.1 Detailed Description

Contains an implementation class for bin_search_tree_.

5.195 insert_fn_imps.hpp File Reference

5.195.1 Detailed Description

Contains implementations of cc_ht_map_'s insert related functions.

5.196 insert_fn_imps.hpp File Reference

5.196.1 Detailed Description

Contains implementations of gp_ht_map_'s insert related functions.

5.197 insert_fn_imps.hpp File Reference

5.197.1 Detailed Description

Contains an implementation class for left_child_next_sibling_heap_.

5.198 insert_fn_imps.hpp File Reference

5.198.1 Detailed Description

Contains implementations of lu_map_.

5.199 insert_fn_imps.hpp File Reference

5.199.1 Detailed Description

Contains an implementation class for ov_tree_.

5.200 insert_fn_imps.hpp File Reference

5.200.1 Detailed Description

Contains an implementation class for a pairing heap.

5.201 insert_fn_imps.hpp File Reference

5.201.1 Detailed Description

Contains an implementation for rb_tree_.

5.202 insert_fn_imps.hpp File Reference

5.202.1 Detailed Description

Contains an implementation for rc_binomial_heap_.

5.203 `insert_fn_imps.hpp` File Reference

5.203.1 Detailed Description

Contains an implementation class for `splay_tree_`.

5.204 `insert_fn_imps.hpp` File Reference

5.204.1 Detailed Description

Contains an implementation for `thin_heap_`.

5.205 `insert_join_fn_imps.hpp` File Reference

5.205.1 Detailed Description

Contains an implementation class for `pat_trie`.

5.206 `insert_no_store_hash_fn_imps.hpp` File Reference

5.206.1 Detailed Description

Contains implementations of `cc_ht_map_`'s insert related functions, when the hash value is not stored.

5.207 `insert_no_store_hash_fn_imps.hpp` File Reference

5.207.1 Detailed Description

Contains implementations of `gp_ht_map_`'s insert related functions, when the hash value is not stored.

5.208 `insert_store_hash_fn_imps.hpp` File Reference

5.208.1 Detailed Description

Contains implementations of `cc_ht_map_`'s insert related functions, when the hash value is stored.

5.209 insert_store_hash_fn_imps.hpp File Reference

5.209.1 Detailed Description

Contains implementations of gp_ht_map_'s find related functions, when the hash value is stored.

5.210 ios_base.h File Reference

Classes

- class [std::ios_base](#)
- class [std::ios_base::failure](#)

Namespaces

- [std](#)

Enumerations

- enum [_ios_Fmtflags](#) {
[_S_boolalpha](#), [_S_dec](#), [_S_fixed](#), [_S_hex](#),
[_S_internal](#), [_S_left](#), [_S_oct](#), [_S_right](#),
[_S_scientific](#), [_S_showbase](#), [_S_showpoint](#), [_S_showpos](#),
[_S_skipws](#), [_S_unitbuf](#), [_S_uppercase](#), [_S_adjustfield](#),
[_S_basefield](#), [_S_floatfield](#), [_S_ios_fmtflags_end](#) }
- enum [_ios_Iostate](#) {
[_S_goodbit](#), [_S_badbit](#), [_S_eofbit](#), [_S_failbit](#),
[_S_ios_iostate_end](#) }
- enum [_ios_Openmode](#) {
[_S_app](#), [_S_ate](#), [_S_bin](#), [_S_in](#),
[_S_out](#), [_S_trunc](#), [_S_ios_openmode_end](#) }
- enum [_ios_Seekdir](#) { [_S_beg](#), [_S_cur](#), [_S_end](#), [_S_ios_seekdir_end](#) }

Functions

- [ios_base & std::boolalpha](#) ([ios_base & __base](#))
- [ios_base & std::dec](#) ([ios_base & __base](#))
- [ios_base & std::fixed](#) ([ios_base & __base](#))
- [ios_base & std::hex](#) ([ios_base & __base](#))
- [ios_base & std::internal](#) ([ios_base & __base](#))
- [ios_base & std::left](#) ([ios_base & __base](#))
- [ios_base & std::noboolalpha](#) ([ios_base & __base](#))
- [ios_base & std::noshowbase](#) ([ios_base & __base](#))
- [ios_base & std::noshowpoint](#) ([ios_base & __base](#))
- [ios_base & std::noshowpos](#) ([ios_base & __base](#))
- [ios_base & std::noskipws](#) ([ios_base & __base](#))
- [ios_base & std::nounitbuf](#) ([ios_base & __base](#))

- `ios_base & std::nouppercase (ios_base & __base)`
- `ios_base & std::oct (ios_base & __base)`
- `constexpr _ios_Fmtflags std::operator& (_ios_Fmtflags __a, _ios_Fmtflags __b)`
- `constexpr _ios_Openmode std::operator& (_ios_Openmode __a, _ios_Openmode __b)`
- `constexpr _ios_ostate std::operator& (_ios_ostate __a, _ios_ostate __b)`
- `const _ios_Fmtflags & std::operator&= (_ios_Fmtflags & __a, _ios_Fmtflags __b)`
- `const _ios_Openmode & std::operator&= (_ios_Openmode & __a, _ios_Openmode __b)`
- `const _ios_ostate & std::operator&= (_ios_ostate & __a, _ios_ostate __b)`
- `constexpr _ios_Fmtflags std::operator^ (_ios_Fmtflags __a, _ios_Fmtflags __b)`
- `constexpr _ios_Openmode std::operator^ (_ios_Openmode __a, _ios_Openmode __b)`
- `constexpr _ios_ostate std::operator^ (_ios_ostate __a, _ios_ostate __b)`
- `const _ios_Fmtflags & std::operator^= (_ios_Fmtflags & __a, _ios_Fmtflags __b)`
- `const _ios_Openmode & std::operator^= (_ios_Openmode & __a, _ios_Openmode __b)`
- `const _ios_ostate & std::operator^= (_ios_ostate & __a, _ios_ostate __b)`
- `constexpr _ios_Fmtflags std::operator| (_ios_Fmtflags __a, _ios_Fmtflags __b)`
- `constexpr _ios_Openmode std::operator| (_ios_Openmode __a, _ios_Openmode __b)`
- `constexpr _ios_ostate std::operator| (_ios_ostate __a, _ios_ostate __b)`
- `const _ios_Fmtflags & std::operator|= (_ios_Fmtflags & __a, _ios_Fmtflags __b)`
- `const _ios_Openmode & std::operator|= (_ios_Openmode & __a, _ios_Openmode __b)`
- `const _ios_ostate & std::operator|= (_ios_ostate & __a, _ios_ostate __b)`
- `constexpr _ios_Fmtflags std::operator~ (_ios_Fmtflags __a)`
- `constexpr _ios_Openmode std::operator~ (_ios_Openmode __a)`
- `constexpr _ios_ostate std::operator~ (_ios_ostate __a)`
- `ios_base & std::right (ios_base & __base)`
- `ios_base & std::scientific (ios_base & __base)`
- `ios_base & std::showbase (ios_base & __base)`
- `ios_base & std::showpoint (ios_base & __base)`
- `ios_base & std::showpos (ios_base & __base)`
- `ios_base & std::skipws (ios_base & __base)`
- `ios_base & std::unitbuf (ios_base & __base)`
- `ios_base & std::uppercase (ios_base & __base)`

5.210.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ios>`.

5.211 iterator.h File Reference

Classes

- class `__gnu_parallel::_IteratorPair< _Iterator1, _Iterator2, _IteratorCategory >`
- class `__gnu_parallel::_IteratorTriple< _Iterator1, _Iterator2, _Iterator3, _IteratorCategory >`

Namespaces

- `__gnu_parallel`

5.211.1 Detailed Description

Helper iterator classes for the `std::transform()` functions. This file is a GNU parallel extension to the Standard C++ Library.

5.212 iterator.hpp File Reference

Classes

- class [iterator_](#)

5.212.1 Detailed Description

Contains an `iterator_` class used for ranging over the elements of the table.

5.213 iterator_fn_imps.hpp File Reference

5.213.1 Detailed Description

Contains implementations of `gp_ht_map_`'s iterators related functions, e.g., `begin()`.

5.214 iterator_tracker.h File Reference

Namespaces

- [std](#)
- [std::__profile](#)

Functions

- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`
`bool std::__profile::operator!= (const __iterator_tracker< _IteratorL, _Sequence > &__lhs, const __iterator_↵`
`tracker< _IteratorR, _Sequence > &__rhs) noexcept`
- `template<typename _Iterator, typename _Sequence >`
`bool std::__profile::operator!= (const __iterator_tracker< _Iterator, _Sequence > &__lhs, const __iterator_↵`
`tracker< _Iterator, _Sequence > &__rhs) noexcept`
- `template<typename _Iterator, typename _Sequence >`
`__iterator_tracker< _Iterator, _Sequence > std::__profile::operator+ (typename __iterator_tracker< _Iterator,`
`_Sequence >::difference_type __n, const __iterator_tracker< _Iterator, _Sequence > &__i) noexcept`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`
`__iterator_tracker< _IteratorL, _Sequence >::difference_type std::__profile::operator- (const __iterator_↵`
`tracker< _IteratorL, _Sequence > &__lhs, const __iterator_tracker< _IteratorR, _Sequence > &__rhs) noexcept`

- `template<typename _Iterator, typename _Sequence>
__iterator_tracker< _Iterator, _Sequence>::difference_type std::__profile::operator- (const __iterator_↵
tracker< _Iterator, _Sequence> &__lhs, const __iterator_tracker< _Iterator, _Sequence> &__rhs) noexcept`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence>
bool std::__profile::operator< (const __iterator_tracker< _IteratorL, _Sequence> &__lhs, const __iterator_↵
tracker< _IteratorR, _Sequence> &__rhs) noexcept`
- `template<typename _Iterator, typename _Sequence>
bool std::__profile::operator< (const __iterator_tracker< _Iterator, _Sequence> &__lhs, const __iterator_↵
tracker< _Iterator, _Sequence> &__rhs) noexcept`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence>
bool std::__profile::operator<= (const __iterator_tracker< _IteratorL, _Sequence> &__lhs, const __iterator_↵
tracker< _IteratorR, _Sequence> &__rhs) noexcept`
- `template<typename _Iterator, typename _Sequence>
bool std::__profile::operator<= (const __iterator_tracker< _Iterator, _Sequence> &__lhs, const __iterator_↵
tracker< _Iterator, _Sequence> &__rhs) noexcept`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence>
bool std::__profile::operator== (const __iterator_tracker< _IteratorL, _Sequence> &__lhs, const __iterator_↵
tracker< _IteratorR, _Sequence> &__rhs) noexcept`
- `template<typename _Iterator, typename _Sequence>
bool std::__profile::operator== (const __iterator_tracker< _Iterator, _Sequence> &__lhs, const __iterator_↵
tracker< _Iterator, _Sequence> &__rhs) noexcept`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence>
bool std::__profile::operator> (const __iterator_tracker< _IteratorL, _Sequence> &__lhs, const __iterator_↵
tracker< _IteratorR, _Sequence> &__rhs) noexcept`
- `template<typename _Iterator, typename _Sequence>
bool std::__profile::operator> (const __iterator_tracker< _Iterator, _Sequence> &__lhs, const __iterator_↵
tracker< _Iterator, _Sequence> &__rhs) noexcept`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence>
bool std::__profile::operator>= (const __iterator_tracker< _IteratorL, _Sequence> &__lhs, const __iterator_↵
tracker< _IteratorR, _Sequence> &__rhs) noexcept`
- `template<typename _Iterator, typename _Sequence>
bool std::__profile::operator>= (const __iterator_tracker< _Iterator, _Sequence> &__lhs, const __iterator_↵
tracker< _Iterator, _Sequence> &__rhs) noexcept`

5.214.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

5.215 iterators_fn_imps.hpp File Reference

5.215.1 Detailed Description

Contains an implementation class for a binary_heap.

5.216 iterators_fn_imps.hpp File Reference

5.216.1 Detailed Description

Contains an implementation class for bin_search_tree_.

5.217 iterators_fn_imps.hpp File Reference

5.217.1 Detailed Description

Contains implementations of cc_ht_map_'s iterators related functions, e.g., begin().

5.218 iterators_fn_imps.hpp File Reference

5.218.1 Detailed Description

Contains an implementation class for left_child_next_sibling_heap_.

5.219 iterators_fn_imps.hpp File Reference

5.219.1 Detailed Description

Contains implementations of lu_map_.

5.220 iterators_fn_imps.hpp File Reference

5.220.1 Detailed Description

Contains an implementation class for ov_tree_.

5.221 iterators_fn_imps.hpp File Reference

5.221.1 Detailed Description

Contains an implementation class for pat_trie.

5.222 left_child_next_sibling_heap_.hpp File Reference

Classes

- class [__gnu_pbds::detail::left_child_next_sibling_heap< Value_Type, Cmp_Fn, Node_Metadata, _Alloc >](#)

Namespaces

- [__gnu_pbds](#)

Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`

5.222.1 Detailed Description

Contains an implementation class for a basic heap.

5.223 `linear_probe_fn_imp.hpp` File Reference

5.223.1 Detailed Description

Contains a probe policy implementation

5.224 `list_partition.h` File Reference

Namespaces

- [`__gnu_parallel`](#)

Functions

- `template<typename _Iter >`
`void __gnu_parallel::__shrink (std::vector< _Iter > &__os_starts, size_t &__count_to_two, size_t &__range_length)`
- `template<typename _Iter >`
`void __gnu_parallel::__shrink_and_double (std::vector< _Iter > &__os_starts, size_t &__count_to_two, size_t &__range_length, const bool __make_twice)`
- `template<typename _Iter, typename _FunctorType >`
`size_t __gnu_parallel::list_partition (const _Iter __begin, const _Iter __end, _Iter * __starts, size_t * __lengths, const int __num_parts, _FunctorType & __f, int __oversampling=0)`

5.224.1 Detailed Description

Functionality to split __sequence referenced by only input iterators. This file is a GNU parallel extension to the Standard C++ Library.

5.225 `list_update_policy.hpp` File Reference

Classes

- class [`__gnu_pbds::lu_counter_policy< Max_Count, _Alloc >`](#)
- class [`__gnu_pbds::lu_move_to_front_policy< _Alloc >`](#)

Namespaces

- [__gnu_pbds](#)

5.225.1 Detailed Description

Contains policies for list update containers.

5.226 locale_classes.h File Reference

Classes

- class [std::collate<_CharT>](#)
- class [std::collate_byname<_CharT>](#)
- class [std::locale](#)
- class [std::locale::facet](#)
- class [std::locale::id](#)

Namespaces

- [std](#)

5.226.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

5.227 locale_facets.h File Reference

Classes

- class [std::__ctype_abstract_base<_CharT>](#)
- class [std::ctype<_CharT>](#)
- class [std::ctype<char>](#)
- class [std::ctype<wchar_t>](#)
- class [std::ctype_byname<_CharT>](#)
- class [std::ctype_byname<char>](#)
- class [std::num_get<_CharT, _InIter>](#)
- class [std::num_put<_CharT, _OutIter>](#)
- class [std::numpunct<_CharT>](#)
- class [std::numpunct_byname<_CharT>](#)

Namespaces

- [std](#)

Macros

- `#define _GLIBCXX_NUM_FACETS`

Functions

- `template<typename _CharT >`
`_CharT * std::add_grouping (_CharT *__s, _CharT __sep, const char *__gbeg, size_t __gsize, const _CharT`
`* __first, const _CharT * __last)`
- `template<typename _Tp >`
`void std::__convert_to_v (const char *, _Tp &, ios_base::iostate &, const __c_locale &) throw ()`
- `template<>`
`void std::__convert_to_v (const char *, float &, ios_base::iostate &, const __c_locale &) throw ()`
- `template<>`
`void std::__convert_to_v (const char *, double &, ios_base::iostate &, const __c_locale &) throw ()`
- `template<>`
`void std::__convert_to_v (const char *, long double &, ios_base::iostate &, const __c_locale &) throw ()`
- `template<typename _CharT >`
`ostreambuf_iterator< _CharT > std::write (ostreambuf_iterator< _CharT > __s, const _CharT * __ws, int`
`__len)`
- `template<typename _CharT, typename _Outlter >`
`_Outlter std::write (_Outlter __s, const _CharT * __ws, int __len)`
- `template<typename _CharT >`
`bool std::isalnum (_CharT __c, const locale & __loc)`
- `template<typename _CharT >`
`bool std::isalpha (_CharT __c, const locale & __loc)`
- `template<typename _CharT >`
`bool std::iscntrl (_CharT __c, const locale & __loc)`
- `template<typename _CharT >`
`bool std::isdigit (_CharT __c, const locale & __loc)`
- `template<typename _CharT >`
`bool std::isgraph (_CharT __c, const locale & __loc)`
- `template<typename _CharT >`
`bool std::islower (_CharT __c, const locale & __loc)`
- `template<typename _CharT >`
`bool std::isprint (_CharT __c, const locale & __loc)`
- `template<typename _CharT >`
`bool std::ispunct (_CharT __c, const locale & __loc)`
- `template<typename _CharT >`
`bool std::isspace (_CharT __c, const locale & __loc)`
- `template<typename _CharT >`
`bool std::isupper (_CharT __c, const locale & __loc)`
- `template<typename _CharT >`
`bool std::isxdigit (_CharT __c, const locale & __loc)`
- `template<typename _CharT >`
`_CharT std::tolower (_CharT __c, const locale & __loc)`
- `template<typename _CharT >`
`_CharT std::toupper (_CharT __c, const locale & __loc)`

5.227.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

5.228 `locale_facets_nonio.h` File Reference

Classes

- class `std::messages<_CharT>`
- struct `std::messages_base`
- class `std::messages_byname<_CharT>`
- class `std::money_base`
- class `std::money_get<_CharT, _InIter>`
- class `std::money_put<_CharT, _OutIter>`
- class `std::moneypunct<_CharT, _Intl>`
- class `std::moneypunct_byname<_CharT, _Intl>`
- class `std::time_base`
- class `std::time_get<_CharT, _InIter>`
- class `std::time_get_byname<_CharT, _InIter>`
- class `std::time_put<_CharT, _OutIter>`
- class `std::time_put_byname<_CharT, _OutIter>`

Namespaces

- `std`

5.228.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

5.229 `localefwd.h` File Reference

Classes

- class `std::codecvt<_InternT, _ExternT, _StateT>`
- class `std::codecvt_byname<_InternT, _ExternT, _StateT>`
- class `std::collate<_CharT>`
- class `std::collate_byname<_CharT>`
- class `std::ctype<_CharT>`
- class `std::ctype_byname<_CharT>`
- class `std::messages<_CharT>`
- class `std::messages_byname<_CharT>`
- class `std::money_get<_CharT, _InIter>`

- class `std::money_put<_CharT, _OutIter>`
- class `std::moneypunct<_CharT, _Intl>`
- class `std::moneypunct_byname<_CharT, _Intl>`
- class `std::num_get<_CharT, _InIter>`
- class `std::num_put<_CharT, _OutIter>`
- class `std::numpunct<_CharT>`
- class `std::numpunct_byname<_CharT>`
- class `std::time_get<_CharT, _InIter>`
- class `std::time_get_byname<_CharT, _InIter>`
- class `std::time_put<_CharT, _OutIter>`
- class `std::time_put_byname<_CharT, _OutIter>`

Namespaces

- `std`

Functions

- `template<typename _Facet>`
`bool std::has_facet (const locale &) throw ()`
- `template<typename _CharT>`
`bool std::isalnum (_CharT __c, const locale &__loc)`
- `template<typename _CharT>`
`bool std::isalpha (_CharT __c, const locale &__loc)`
- `template<typename _CharT>`
`bool std::iscntrl (_CharT __c, const locale &__loc)`
- `template<typename _CharT>`
`bool std::isdigit (_CharT __c, const locale &__loc)`
- `template<typename _CharT>`
`bool std::isgraph (_CharT __c, const locale &__loc)`
- `template<typename _CharT>`
`bool std::islower (_CharT __c, const locale &__loc)`
- `template<typename _CharT>`
`bool std::isprint (_CharT __c, const locale &__loc)`
- `template<typename _CharT>`
`bool std::ispunct (_CharT __c, const locale &__loc)`
- `template<typename _CharT>`
`bool std::isspace (_CharT __c, const locale &__loc)`
- `template<typename _CharT>`
`bool std::isupper (_CharT __c, const locale &__loc)`
- `template<typename _CharT>`
`bool std::isxdigit (_CharT __c, const locale &__loc)`
- `template<typename _CharT>`
`_CharT std::tolower (_CharT __c, const locale &__loc)`
- `template<typename _CharT>`
`_CharT std::toupper (_CharT __c, const locale &__loc)`
- `template<typename _Facet>`
`const _Facet & std::use_facet (const locale &)`

5.229.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

5.230 losertree.h File Reference

Classes

- class [__gnu_parallel::_LoserTree< __stable, _Tp, _Compare >](#)
- class [__gnu_parallel::_LoserTree< false, _Tp, _Compare >](#)
- class [__gnu_parallel::_LoserTreeBase< _Tp, _Compare >](#)
- struct [__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser](#)
- class [__gnu_parallel::_LoserTreePointer< __stable, _Tp, _Compare >](#)
- class [__gnu_parallel::_LoserTreePointer< false, _Tp, _Compare >](#)
- class [__gnu_parallel::_LoserTreePointerBase< _Tp, _Compare >](#)
- struct [__gnu_parallel::_LoserTreePointerBase< _Tp, _Compare >::_Loser](#)
- class [__gnu_parallel::_LoserTreePointerUnguarded< __stable, _Tp, _Compare >](#)
- class [__gnu_parallel::_LoserTreePointerUnguarded< false, _Tp, _Compare >](#)
- class [__gnu_parallel::_LoserTreePointerUnguardedBase< _Tp, _Compare >](#)
- class [__gnu_parallel::_LoserTreeUnguarded< __stable, _Tp, _Compare >](#)
- class [__gnu_parallel::_LoserTreeUnguarded< false, _Tp, _Compare >](#)
- class [__gnu_parallel::_LoserTreeUnguardedBase< _Tp, _Compare >](#)

Namespaces

- [__gnu_parallel](#)

5.230.1 Detailed Description

Many generic loser tree variants. This file is a GNU parallel extension to the Standard C++ Library.

5.231 lu_counter_metadata.hpp File Reference

Classes

- class [__gnu_pbds::detail::lu_counter_metadata< Size_Type >](#)
- class [__gnu_pbds::detail::lu_counter_policy_base< Size_Type >](#)
- class [__gnu_pbds::detail::lu_counter_policy_base< Size_Type >](#)

Namespaces

- [__gnu_pbds](#)

5.231.1 Detailed Description

Contains implementation of a lu counter policy's metadata.

5.232 lu_map_.hpp File Reference

Classes

- class [__gnu_pbds::detail::lu_map< Key, Mapped, Eq_Fn, _Alloc, Update_Policy >](#)

Namespaces

- [__gnu_pbds](#)

Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_GEN_POS`
- `#define PB_DS_LU_NAME`
- `#define PB_DS_LU_TRAITS_BASE`

5.232.1 Detailed Description

Contains a list update map.

5.233 macros.h File Reference

Macros

- `#define __glibcxx_check_bucket_index(_N)`
- `#define __glibcxx_check_equal_allocs(_Other)`
- `#define __glibcxx_check_erase(_Position)`
- `#define __glibcxx_check_erase_after(_Position)`
- `#define __glibcxx_check_erase_range(_First, _Last)`
- `#define __glibcxx_check_erase_range_after(_First, _Last)`
- `#define __glibcxx_check_heap(_First, _Last)`
- `#define __glibcxx_check_heap_pred(_First, _Last, _Pred)`
- `#define __glibcxx_check_insert(_Position)`
- `#define __glibcxx_check_insert_after(_Position)`
- `#define __glibcxx_check_insert_range(_Position, _First, _Last)`
- `#define __glibcxx_check_insert_range_after(_Position, _First, _Last)`
- `#define __glibcxx_check_max_load_factor(_F)`
- `#define __glibcxx_check_non_empty_range(_First, _Last)`

- `#define __glibcxx_check_nonempty()`
- `#define __glibcxx_check_partitioned_lower(_First, _Last, _Value)`
- `#define __glibcxx_check_partitioned_lower_pred(_First, _Last, _Value, _Pred)`
- `#define __glibcxx_check_partitioned_upper(_First, _Last, _Value)`
- `#define __glibcxx_check_partitioned_upper_pred(_First, _Last, _Value, _Pred)`
- `#define __glibcxx_check_self_move_assign(_Other)`
- `#define __glibcxx_check_sorted(_First, _Last)`
- `#define __glibcxx_check_sorted_pred(_First, _Last, _Pred)`
- `#define __glibcxx_check_sorted_set(_First1, _Last1, _First2)`
- `#define __glibcxx_check_sorted_set_pred(_First1, _Last1, _First2, _Pred)`
- `#define __glibcxx_check_string(_String)`
- `#define __glibcxx_check_string_len(_String, _Len)`
- `#define __glibcxx_check_subscript(_N)`
- `#define __glibcxx_check_valid_range(_First, _Last)`
- `#define _GLIBCXX_DEBUG_VERIFY(_Condition, _ErrorMessage)`
- `#define _GLIBCXX_DEBUG_VERIFY_AT(_Condition, _ErrorMessage, _File, _Line)`

5.233.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

5.233.2 Macro Definition Documentation

5.233.2.1 `#define __glibcxx_check_erase(_Position)`

Verify that we can erase the element referenced by the iterator `_Position`. We can erase the element if the `_Position` iterator is dereferenceable and references this sequence.

Definition at line 139 of file `macros.h`.

5.233.2.2 `#define __glibcxx_check_erase_after(_Position)`

Verify that we can erase the element after the iterator `_Position`. We can erase the element if the `_Position` iterator is before a dereferenceable one and references this sequence.

Definition at line 153 of file `macros.h`.

5.233.2.3 `#define __glibcxx_check_erase_range(_First, _Last)`

Verify that we can erase the elements in the iterator range `[_First, _Last)`. We can erase the elements if `[_First, _Last)` is a valid iterator range within this sequence.

Definition at line 167 of file `macros.h`.

5.233.2.4 `#define __glibcxx_check_erase_range_after(_First, _Last)`

Verify that we can erase the elements in the iterator range `[_First, _Last)`. We can erase the elements if `[_First, _Last)` is a valid iterator range within this sequence.

Definition at line 179 of file `macros.h`.

5.233.2.5 `#define __glibcxx_check_heap_pred(_First, _Last, _Pred)`

Verify that the iterator range `[_First, _Last)` is a heap w.r.t. the predicate `_Pred`.

Definition at line 329 of file `macros.h`.

5.233.2.6 `#define __glibcxx_check_insert(_Position)`

Verify that we can insert into `*this` with the iterator `_Position`. Insertion into a container at a specific position requires that the iterator be nonsingular, either dereferenceable or past-the-end, and that it reference the sequence we are inserting into. Note that this macro is only valid when the container is a `_Safe_sequence` and the iterator is a `_Safe_iterator`.

Definition at line 73 of file `macros.h`.

5.233.2.7 `#define __glibcxx_check_insert_after(_Position)`

Verify that we can insert into `*this` after the iterator `_Position`. Insertion into a container after a specific position requires that the iterator be nonsingular, either dereferenceable or before-begin, and that it reference the sequence we are inserting into. Note that this macro is only valid when the container is a `_Safe_sequence` and the iterator is a `_Safe_iterator`.

Definition at line 90 of file `macros.h`.

5.233.2.8 `#define __glibcxx_check_insert_range(_Position, _First, _Last)`

Verify that we can insert the values in the iterator range `[_First, _Last)` into `*this` with the iterator `_Position`. Insertion into a container at a specific position requires that the iterator be nonsingular (i.e., either dereferenceable or past-the-end), that it reference the sequence we are inserting into, and that the iterator range `[_First, _Last)` is a valid (possibly empty) range which does not reference the sequence we are inserting into. Note that this macro is only valid when the container is a `_Safe_sequence` and the `_Position` iterator is a `_Safe_iterator`.

Definition at line 107 of file `macros.h`.

5.233.2.9 `#define __glibcxx_check_insert_range_after(_Position, _First, _Last)`

Verify that we can insert the values in the iterator range `[_First, _Last)` into `*this` after the iterator `_Position`. Insertion into a container after a specific position requires that the iterator be nonsingular (i.e., either dereferenceable or past-the-end), that it reference the sequence we are inserting into, and that the iterator range `[_First, _Last)` is a valid (possibly empty) range which does not reference the sequence we are inserting into. Note that this macro is only valid when the container is a `_Safe_sequence` and the `_Position` iterator is a `_Safe_iterator`.

Definition at line 126 of file `macros.h`.

5.233.2.10 `#define __glibcxx_check_partitioned_lower(_First, _Last, _Value)`

Verify that the iterator range `[_First, _Last)` is partitioned w.r.t. the value `_Value`.

Definition at line 273 of file `macros.h`.

5.233.2.11 `#define __glibcxx_check_partitioned_lower_pred(_First, _Last, _Value, _Pred)`

Verify that the iterator range `[_First, _Last)` is partitioned w.r.t. the value `_Value` and predicate `_Pred`.

Definition at line 295 of file `macros.h`.

5.233.2.12 `#define __glibcxx_check_partitioned_upper_pred(_First, _Last, _Value, _Pred)`

Verify that the iterator range `[_First, _Last)` is partitioned w.r.t. the value `_Value` and predicate `_Pred`.

Definition at line 308 of file `macros.h`.

5.233.2.13 `#define __glibcxx_check_sorted_pred(_First, _Last, _Pred)`

Verify that the iterator range `[_First, _Last)` is sorted by the predicate `_Pred`.

Definition at line 239 of file `macros.h`.

5.233.2.14 `#define _GLIBCXX_DEBUG_VERIFY_AT(_Condition, _ErrorMessage, _File, _Line)`

Macros used by the implementation to verify certain properties. These macros may only be used directly by the debug wrappers. Note that these are macros (instead of the more obviously *correct* choice of making them functions) because we need line and file information at the call site, to minimize the distance between the user error and where the error is reported.

Definition at line 41 of file `macros.h`.

5.234 malloc_allocator.h File Reference

Classes

- class [__gnu_cxx::malloc_allocator< _Tp >](#)

Namespaces

- [__gnu_cxx](#)

Functions

- `template<typename _Tp >`
`bool __gnu_cxx::operator!= (const malloc_allocator< _Tp > &, const malloc_allocator< _Tp > &)`
- `template<typename _Tp >`
`bool __gnu_cxx::operator== (const malloc_allocator< _Tp > &, const malloc_allocator< _Tp > &)`

5.234.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

5.235 map.h File Reference

Classes

- class [std::__debug::map< _Key, _Tp, _Compare, _Allocator >](#)

Namespaces

- [std](#)
- [std::__debug](#)

Functions

- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >
bool std::__debug::operator!= (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >
bool std::__debug::operator< (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >
bool std::__debug::operator<= (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >
bool std::__debug::operator== (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >
bool std::__debug::operator> (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >
bool std::__debug::operator>= (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >
void std::__debug::swap (map< _Key, _Tp, _Compare, _Allocator > &__lhs, map< _Key, _Tp, _Compare, _Allocator > &__rhs)`

5.235.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

5.236 map.h File Reference

Classes

- class [std::__profile::map< _Key, _Tp, _Compare, _Allocator >](#)

Namespaces

- [std](#)
- [std::__profile](#)

Functions

- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >
bool std::__profile::operator!= (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >
bool std::__profile::operator< (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >
bool std::__profile::operator<= (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >
bool std::__profile::operator== (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >
bool std::__profile::operator> (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >
bool std::__profile::operator>= (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >
void std::__profile::swap (map< _Key, _Tp, _Compare, _Allocator > &__lhs, map< _Key, _Tp, _Compare, _Allocator > &__rhs)`

5.236.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

5.237 mask_array.h File Reference

Classes

- class [std::mask_array< _Tp >](#)

Namespaces

- [std](#)

Macros

- `#define _DEFINE_VALARRAY_OPERATOR(_Op, _Name)`

5.237.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<valarray>`.

5.238 `mask_based_range_hashing.hpp` File Reference

Classes

- class [__gnu_pbds::detail::mask_based_range_hashing< Size_Type >](#)

Namespaces

- [__gnu_pbds](#)

5.238.1 Detailed Description

Contains a range hashing policy base.

5.239 `memoryfwd.h` File Reference

Classes

- class [std::allocator< _Tp >](#)
- struct [std::uses_allocator< _Tp, _Alloc >](#)

Namespaces

- [std](#)

5.239.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

5.240 `merge.h` File Reference

Namespaces

- [__gnu_parallel](#)

Functions

- `template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename _DifferenceTp, typename _Compare > _OutputIterator __gnu_parallel::__merge_advance (_RAIter1 &__begin1, _RAIter1 __end1, _RAIter2 &__begin2, _RAIter2 __end2, _OutputIterator __target, _DifferenceTp __max_length, _Compare __comp)`
- `template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename _DifferenceTp, typename _Compare > _OutputIterator __gnu_parallel::__merge_advance_movc (_RAIter1 &__begin1, _RAIter1 __end1, _RAIter2 &__begin2, _RAIter2 __end2, _OutputIterator __target, _DifferenceTp __max_length, _Compare __comp)`
- `template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename _DifferenceTp, typename _Compare > _OutputIterator __gnu_parallel::__merge_advance_usual (_RAIter1 &__begin1, _RAIter1 __end1, _RAIter2 &__begin2, _RAIter2 __end2, _OutputIterator __target, _DifferenceTp __max_length, _Compare __comp)`
- `template<typename _RAIter1, typename _RAIter2, typename _RAIter3, typename _Compare > _RAIter3 __gnu_parallel::__parallel_merge_advance (_RAIter1 &__begin1, _RAIter1 __end1, _RAIter2 &__begin2, _RAIter2 __end2, _RAIter3 __target, typename std::iterator_traits<_RAIter1 >::difference_type __max_length, _Compare __comp)`
- `template<typename _RAIter1, typename _RAIter3, typename _Compare > _RAIter3 __gnu_parallel::__parallel_merge_advance (_RAIter1 &__begin1, _RAIter1 __end1, _RAIter1 &__begin2, _RAIter1 __end2, _RAIter3 __target, typename std::iterator_traits<_RAIter1 >::difference_type __max_length, _Compare __comp)`

5.240.1 Detailed Description

Parallel implementation of `std::merge()`. This file is a GNU parallel extension to the Standard C++ Library.

5.241 messages_members.h File Reference

Namespaces

- [std](#)

5.241.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

5.242 mod_based_range_hashing.hpp File Reference

Classes

- class [__gnu_pbds::detail::mod_based_range_hashing< Size_Type >](#)

Namespaces

- [__gnu_pbds](#)

5.242.1 Detailed Description

Contains a range hashing policy base.

5.243 move.h File Reference

Namespaces

- [std](#)

Macros

- `#define _GLIBCXX_FORWARD(_Tp, __val)`
- `#define _GLIBCXX_MOVE(__val)`

Functions

- `template<typename _Tp >`
`_Tp * std::__addressof (_Tp &__r) noexcept`
- `template<typename _Tp >`
`_Tp * std::addressof (_Tp &__r) noexcept`
- `template<typename _Tp >`
`constexpr _Tp && std::forward (typename std::remove_reference< _Tp >::type &__t) noexcept`
- `template<typename _Tp >`
`constexpr _Tp && std::forward (typename std::remove_reference< _Tp >::type &&__t) noexcept`
- `template<typename _Tp >`
`constexpr std::remove_reference< _Tp >::type && std::move (_Tp &&__t) noexcept`
- `template<typename _Tp >`
`constexpr conditional< __move_if_noexcept_cond< _Tp >::value, const _Tp &, _Tp && >::type std::move_if_noexcept (_Tp &__x) noexcept`
- `template<typename _Tp >`
`void std::swap (_Tp &__a, _Tp &__b) noexcept(__and_< is_nothrow_move_constructible< _Tp >, is_nothrow_move_assignable< _Tp >>::value)`
- `template<typename _Tp, size_t _Nm>`
`void std::swap (_Tp(&__a)[_Nm], _Tp(&__b)[_Nm]) noexcept(noexcept(swap(*__a,*__b)))`

5.243.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<utility>`.

5.244 `mt_allocator.h` File Reference

Classes

- struct [__gnu_cxx::__common_pool_policy<_PoolTp, _Thread>](#)
- class [__gnu_cxx::__mt_alloc<_Tp, _Poolp>](#)
- class [__gnu_cxx::__mt_alloc_base<_Tp>](#)
- struct [__gnu_cxx::__per_type_pool_policy<_Tp, _PoolTp, _Thread>](#)
- class [__gnu_cxx::__pool<_Thread>](#)
- class [__gnu_cxx::__pool<false>](#)
- class [__gnu_cxx::__pool<true>](#)
- struct [__gnu_cxx::__pool_base](#)

Namespaces

- [__gnu_cxx](#)

Macros

- `#define __thread_default`

Typedefs

- typedef void(* [__gnu_cxx::__destroy_handler](#)) (void *)

Functions

- template<typename `_Tp`, typename `_Poolp`>
bool [__gnu_cxx::operator!=](#) (const [__mt_alloc<_Tp, _Poolp>](#) &, const [__mt_alloc<_Tp, _Poolp>](#) &)
- template<typename `_Tp`, typename `_Poolp`>
bool [__gnu_cxx::operator==](#) (const [__mt_alloc<_Tp, _Poolp>](#) &, const [__mt_alloc<_Tp, _Poolp>](#) &)

5.244.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

5.245 `multimap.h` File Reference

Classes

- class [std::__debug::multimap<_Key, _Tp, _Compare, _Allocator>](#)

Namespaces

- [std](#)
- [std::__debug](#)

Functions

- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >
bool std::__debug::operator!= (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >
bool std::__debug::operator< (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >
bool std::__debug::operator<= (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >
bool std::__debug::operator== (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >
bool std::__debug::operator> (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >
bool std::__debug::operator>= (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >
void std::__debug::swap (multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`

5.245.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

5.246 multimap.h File Reference

Classes

- [class std::__profile::multimap< _Key, _Tp, _Compare, _Allocator >](#)

Namespaces

- [std](#)
- [std::__profile](#)

Functions

- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool std::__profile::operator!= (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool std::__profile::operator< (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool std::__profile::operator<= (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool std::__profile::operator== (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool std::__profile::operator> (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`bool std::__profile::operator>= (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`
`void std::__profile::swap (multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`

5.246.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

5.247 multiseq_selection.h File Reference

Classes

- class [__gnu_parallel::_Lexicographic< _T1, _T2, _Compare >](#)
- class [__gnu_parallel::_LexicographicReverse< _T1, _T2, _Compare >](#)

Namespaces

- [__gnu_parallel](#)

Macros

- `#define __S(__i)`
- `#define __S(__i)`

Functions

- `template<typename _RanSeqs, typename _RankType, typename _RankIterator, typename _Compare >`
`void __gnu_parallel::multiseq_partition (_RanSeqs __begin_seqs, _RanSeqs __end_seqs, _RankType __rank,`
`_RankIterator __begin_offsets, _Compare __comp=std::less< typename std::iterator_traits< typename std::iterator_traits< _RanSeqs >::value_type::first_type >::value_type >>())`
- `template<typename _Tp, typename _RanSeqs, typename _RankType, typename _Compare >`
`_Tp __gnu_parallel::multiseq_selection (_RanSeqs __begin_seqs, _RanSeqs __end_seqs, _RankType __rank,`
`_RankType &__offset, _Compare __comp=std::less< _Tp >())`

5.247.1 Detailed Description

Functions to find elements of a certain global `__rank` in multiple sorted sequences. Also serves for splitting such sequence sets.

The algorithm description can be found in

P. J. Varman, S. D. Scheufler, B. R. Iyer, and G. R. Ricard. Merging Multiple Lists on Hierarchical-Memory Multiprocessors. *Journal of Parallel and Distributed Computing*, 12(2):171–177, 1991.

This file is a GNU parallel extension to the Standard C++ Library.

5.248 `multiset.h` File Reference

Classes

- class `std::__debug::multiset< _Key, _Compare, _Allocator >`

Namespaces

- `std`
- `std::__debug`

Functions

- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool std::__debug::operator!= (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key,`
`_Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool std::__debug::operator< (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key,`
`_Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool std::__debug::operator<= (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key,`
`_Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool std::__debug::operator== (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key,`
`_Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool std::__debug::operator> (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key,`
`_Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool std::__debug::operator>= (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key,`
`_Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`void std::__debug::swap (multiset< _Key, _Compare, _Allocator > &__x, multiset< _Key, _Compare, _Allocator`
`> &__y)`

5.248.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

5.249 multiset.h File Reference

Classes

- class [std::__profile::multiset<_Key, _Compare, _Allocator>](#)

Namespaces

- [std](#)
- [std::__profile](#)

Functions

- `template<typename _Key, typename _Compare, typename _Allocator>
bool std::__profile::operator!= (const multiset< _Key, _Compare, _Allocator> &__lhs, const multiset< _Key, _Compare, _Allocator> &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator>
bool std::__profile::operator< (const multiset< _Key, _Compare, _Allocator> &__lhs, const multiset< _Key, _Compare, _Allocator> &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator>
bool std::__profile::operator<= (const multiset< _Key, _Compare, _Allocator> &__lhs, const multiset< _Key, _Compare, _Allocator> &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator>
bool std::__profile::operator== (const multiset< _Key, _Compare, _Allocator> &__lhs, const multiset< _Key, _Compare, _Allocator> &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator>
bool std::__profile::operator> (const multiset< _Key, _Compare, _Allocator> &__lhs, const multiset< _Key, _Compare, _Allocator> &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator>
bool std::__profile::operator>= (const multiset< _Key, _Compare, _Allocator> &__lhs, const multiset< _Key, _Compare, _Allocator> &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator>
void std::__profile::swap (multiset< _Key, _Compare, _Allocator> &__x, multiset< _Key, _Compare, _Allocator> &__y)`

5.249.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

5.250 multiway_merge.h File Reference

Classes

- struct [__gnu_parallel::__multiway_merge_3_variant_sentinel_switch](#)< __sentinels, _RAIterIterator, _RAIter3, [_DifferenceTp](#), [_Compare](#) >
- struct [__gnu_parallel::__multiway_merge_3_variant_sentinel_switch](#)< true, _RAIterIterator, _RAIter3, [_DifferenceTp](#), [_Compare](#) >
- struct [__gnu_parallel::__multiway_merge_4_variant_sentinel_switch](#)< __sentinels, _RAIterIterator, _RAIter3, [_DifferenceTp](#), [_Compare](#) >
- struct [__gnu_parallel::__multiway_merge_4_variant_sentinel_switch](#)< true, _RAIterIterator, _RAIter3, [_DifferenceTp](#), [_Compare](#) >
- struct [__gnu_parallel::__multiway_merge_k_variant_sentinel_switch](#)< __sentinels, __stable, _RAIterIterator, [_RAIter3](#), [_DifferenceTp](#), [_Compare](#) >
- struct [__gnu_parallel::__multiway_merge_k_variant_sentinel_switch](#)< false, __stable, _RAIterIterator, _RAIter3, [_DifferenceTp](#), [_Compare](#) >
- class [__gnu_parallel::_GuardedIterator](#)< _RAIter, [_Compare](#) >
- struct [__gnu_parallel::_LoserTreeTraits](#)< [_Tp](#) >
- struct [__gnu_parallel::_SamplingSorter](#)< __stable, _RAIter, [_StrictWeakOrdering](#) >
- struct [__gnu_parallel::_SamplingSorter](#)< false, _RAIter, [_StrictWeakOrdering](#) >

Namespaces

- [__gnu_parallel](#)

Macros

- [#define GLIBCXX_PARALLEL_DECISION](#)(__a, __b, __c, __d)
- [#define GLIBCXX_PARALLEL_LENGTH](#)(__s)
- [#define GLIBCXX_PARALLEL_MERGE_3_CASE](#)(__a, __b, __c, __c0, __c1)
- [#define GLIBCXX_PARALLEL_MERGE_4_CASE](#)(__a, __b, __c, __d, __c0, __c1, __c2)

Functions

- [template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename _DifferenceTp, typename _Compare> _OutputIterator __gnu_parallel::merge_advance](#) (_RAIter1 & __begin1, _RAIter1 __end1, _RAIter2 & __begin2, _RAIter2 __end2, _OutputIterator __target, [_DifferenceTp](#) __max_length, [_Compare](#) __comp)
- [template<bool __stable, bool __sentinels, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare> _RAIter3 __gnu_parallel::sequential_multiway_merge](#) (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator >::value_type::first_type >::value_type & __sentinel, [_DifferenceTp](#) __length, [_Compare](#) __comp)
- [template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare> _RAIterOut __gnu_parallel::multiway_merge](#) (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, [_DifferenceTp](#) __length, [_Compare](#) __comp, [__gnu_parallel::sequential_tag](#))
- [template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare> _RAIterOut __gnu_parallel::multiway_merge](#) (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, [_DifferenceTp](#) __length, [_Compare](#) __comp, [__gnu_parallel::exact_tag](#) __tag)

- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`
`_RAIterOut gnu_parallel::multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, gnu_parallel::sampling_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`
`_RAIterOut gnu_parallel::multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, parallel_tag __tag=parallel_tag(0))`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`
`_RAIterOut gnu_parallel::multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, default_parallel_tag __tag)`
- `template<template< typename RAI, typename C > class iterator, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare >`
`_RAIter3 gnu_parallel::multiway_merge_3_variant (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp)`
- `template<template< typename RAI, typename C > class iterator, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare >`
`_RAIter3 gnu_parallel::multiway_merge_4_variant (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp)`
- `template<bool __stable, typename _RAIterIterator, typename _Compare, typename _DifferenceType >`
`void gnu_parallel::multiway_merge_exact_splitting (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _DifferenceType __length, _DifferenceType __total_length, _Compare __comp, std::vector< std::pair< _DifferenceType, _DifferenceType > > *__pieces)`
- `template<typename _LT, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare >`
`_RAIter3 gnu_parallel::multiway_merge_loser_tree (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp)`
- `template<typename UnguardedLoserTree, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare >`
`_RAIter3 gnu_parallel::multiway_merge_loser_tree_sentinel (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator >::value_type::first_type >::value_type &__sentinel, _DifferenceTp __length, _Compare __comp)`
- `template<typename _LT, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare >`
`_RAIter3 gnu_parallel::multiway_merge_loser_tree_unguarded (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator >::value_type::first_type >::value_type &__sentinel, _DifferenceTp __length, _Compare __comp)`
- `template<bool __stable, typename _RAIterIterator, typename _Compare, typename _DifferenceType >`
`void gnu_parallel::multiway_merge_sampling_splitting (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _DifferenceType __length, _DifferenceType __total_length, _Compare __comp, std::vector< std::pair< _DifferenceType, _DifferenceType > > *__pieces)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`
`_RAIterOut gnu_parallel::multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, gnu_parallel::sequential_tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`
`_RAIterOut gnu_parallel::multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, gnu_parallel::exact_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`
`_RAIterOut gnu_parallel::multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, sampling_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`
`_RAIterOut gnu_parallel::multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, parallel_tag __tag=parallel_tag(0))`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`
`_RAIterOut gnu_parallel::multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, default_parallel_tag __tag)`

- `template<bool __stable, bool __sentinels, typename _RAlterIterator, typename _RAlter3, typename _DifferenceTp, typename _Splitter, typename _Compare >`
`_RAlter3 __gnu_parallel::parallel_multiway_merge (_RAlterIterator __seqs_begin, _RAlterIterator __seqs_end,`
`_RAlter3 __target, _Splitter __splitter, _DifferenceTp __length, _Compare __comp, _ThreadIndex __num, ↵`
`threads)`
- `template<typename _RAlterPairIterator, typename _RAlterOut, typename _DifferenceTp, typename _Compare >`
`_RAlterOut __gnu_parallel::stable_multiway_merge (_RAlterPairIterator __seqs_begin, _RAlterPairIterator ↵`
`__seqs_end, _RAlterOut __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::sequential_tag)`
- `template<typename _RAlterPairIterator, typename _RAlterOut, typename _DifferenceTp, typename _Compare >`
`_RAlterOut __gnu_parallel::stable_multiway_merge (_RAlterPairIterator __seqs_begin, _RAlterPairIterator ↵`
`__seqs_end, _RAlterOut __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::exact_tag __tag)`
- `template<typename _RAlterPairIterator, typename _RAlterOut, typename _DifferenceTp, typename _Compare >`
`_RAlterOut __gnu_parallel::stable_multiway_merge (_RAlterPairIterator __seqs_begin, _RAlterPairIterator ↵`
`__seqs_end, _RAlterOut __target, _DifferenceTp __length, _Compare __comp, sampling_tag __tag)`
- `template<typename _RAlterPairIterator, typename _RAlterOut, typename _DifferenceTp, typename _Compare >`
`_RAlterOut __gnu_parallel::stable_multiway_merge (_RAlterPairIterator __seqs_begin, _RAlterPairIterator ↵`
`__seqs_end, _RAlterOut __target, _DifferenceTp __length, _Compare __comp, parallel_tag __tag=parallel ↵`
`tag(0))`
- `template<typename _RAlterPairIterator, typename _RAlterOut, typename _DifferenceTp, typename _Compare >`
`_RAlterOut __gnu_parallel::stable_multiway_merge (_RAlterPairIterator __seqs_begin, _RAlterPairIterator ↵`
`__seqs_end, _RAlterOut __target, _DifferenceTp __length, _Compare __comp, default_parallel_tag __tag)`
- `template<typename _RAlterPairIterator, typename _RAlterOut, typename _DifferenceTp, typename _Compare >`
`_RAlterOut __gnu_parallel::stable_multiway_merge_sentinels (_RAlterPairIterator __seqs_begin, _RAlter ↵`
`PairIterator __seqs_end, _RAlterOut __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel:: ↵`
`sequential_tag)`
- `template<typename _RAlterPairIterator, typename _RAlterOut, typename _DifferenceTp, typename _Compare >`
`_RAlterOut __gnu_parallel::stable_multiway_merge_sentinels (_RAlterPairIterator __seqs_begin, _RAlter ↵`
`PairIterator __seqs_end, _RAlterOut __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel:: ↵`
`exact_tag __tag)`
- `template<typename _RAlterPairIterator, typename _RAlterOut, typename _DifferenceTp, typename _Compare >`
`_RAlterOut __gnu_parallel::stable_multiway_merge_sentinels (_RAlterPairIterator __seqs_begin, _RAlter ↵`
`PairIterator __seqs_end, _RAlterOut __target, _DifferenceTp __length, _Compare __comp, sampling_tag __tag)`
- `template<typename _RAlterPairIterator, typename _RAlterOut, typename _DifferenceTp, typename _Compare >`
`_RAlterOut __gnu_parallel::stable_multiway_merge_sentinels (_RAlterPairIterator __seqs_begin, _RAlter ↵`
`PairIterator __seqs_end, _RAlterOut __target, _DifferenceTp __length, _Compare __comp, parallel_tag ↵`
`tag=parallel_tag(0))`
- `template<typename _RAlterPairIterator, typename _RAlterOut, typename _DifferenceTp, typename _Compare >`
`_RAlterOut __gnu_parallel::stable_multiway_merge_sentinels (_RAlterPairIterator __seqs_begin, _RAlter ↵`
`PairIterator __seqs_end, _RAlterOut __target, _DifferenceTp __length, _Compare __comp, default_parallel_tag ↵`
`tag)`

5.250.1 Detailed Description

Implementation of sequential and parallel multiway merge.

Explanations on the high-speed merging routines in the appendix of

P. Sanders. Fast priority queues for cached memory. ACM Journal of Experimental Algorithmics, 5, 2000.

This file is a GNU parallel extension to the Standard C++ Library.

5.250.2 Macro Definition Documentation

5.250.2.1 `#define _GLIBCXX_PARALLEL_LENGTH(__s)`

Length of a sequence described by a pair of iterators.

Definition at line 54 of file multiway_merge.h.

Referenced by `__gnu_parallel::sequential_multiway_merge()`, `__gnu_parallel::multiway_merge_exact_splitting()`, `__gnu_parallel::multiway_merge_loser_tree()`, `__gnu_parallel::multiway_merge_sampling_splitting()`, and `__gnu_parallel::parallel_multiway_merge()`.

5.251 multiway_mergesort.h File Reference

Classes

- struct [__gnu_parallel::_Piece< _DifferenceTp >](#)
- struct [__gnu_parallel::_PMWMSortingData< _RAIter >](#)
- struct [__gnu_parallel::_SplitConsistently< __exact, _RAIter, _Compare, _SortingPlacesIterator >](#)
- struct [__gnu_parallel::_SplitConsistently< false, _RAIter, _Compare, _SortingPlacesIterator >](#)
- struct [__gnu_parallel::_SplitConsistently< true, _RAIter, _Compare, _SortingPlacesIterator >](#)

Namespaces

- [__gnu_parallel](#)

Functions

- template<typename _RAIter, typename _DifferenceTp >
void [__gnu_parallel::__determine_samples](#) (_PMWMSortingData< _RAIter > *__sd, _DifferenceTp __num←
_samples)
- template<bool __stable, bool __exact, typename _RAIter, typename _Compare >
void [__gnu_parallel::parallel_sort_mwms](#) (_RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex
__num_threads)
- template<bool __stable, bool __exact, typename _RAIter, typename _Compare >
void [__gnu_parallel::parallel_sort_mwms_pu](#) (_PMWMSortingData< _RAIter > *__sd, _Compare &__comp)

5.251.1 Detailed Description

Parallel multiway merge sort. This file is a GNU parallel extension to the Standard C++ Library.

5.252 nested_exception.h File Reference

Classes

- class [std::nested_exception](#)

Namespaces

- [std](#)

Functions

- `template<typename _Ex >`
`const nested_exception * std::__get_nested_exception (const _Ex &__ex)`
- `template<typename _Ex >`
`void std::__throw_with_nested (_Ex &&, const nested_exception *=0) __attribute__((__noreturn__))`
- `template<typename _Ex >`
`void std::__throw_with_nested (_Ex &&,...) __attribute__((__noreturn__))`
- `template<typename _Ex >`
`void std::rethrow_if_nested (const _Ex &__ex)`
- `void std::rethrow_if_nested (const nested_exception &__ex)`
- `template<typename _Ex >`
`void std::throw_with_nested (_Ex __ex)`

5.252.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<exception>`.

5.253 `new_allocator.h` File Reference

Classes

- class [__gnu_cxx::new_allocator< _Tp >](#)

Namespaces

- [__gnu_cxx](#)

Functions

- `template<typename _Tp >`
`bool __gnu_cxx::operator!= (const new_allocator< _Tp > &, const new_allocator< _Tp > &)`
- `template<typename _Tp >`
`bool __gnu_cxx::operator== (const new_allocator< _Tp > &, const new_allocator< _Tp > &)`

5.253.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

5.254 node.hpp File Reference

Classes

- struct [__gnu_pbds::detail::left_child_next_sibling_heap_node_< _Value, _Metadata, _Alloc >](#)

Namespaces

- [__gnu_pbds](#)

5.254.1 Detailed Description

Contains an implementation struct for this type of heap's node.

5.255 node.hpp File Reference

Classes

- struct [__gnu_pbds::detail::rb_tree_node_< Value_Type, Metadata, _Alloc >](#)

Namespaces

- [__gnu_pbds](#)

5.255.1 Detailed Description

Contains an implementation for rb_tree_.

5.256 node.hpp File Reference

Classes

- struct [__gnu_pbds::detail::splay_tree_node_< Value_Type, Metadata, _Alloc >](#)

Namespaces

- [__gnu_pbds](#)

5.256.1 Detailed Description

Contains an implementation struct for splay_tree_'s node.

5.257 node_iterators.hpp File Reference

Classes

- class [__gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >](#)
- class [__gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc >](#)

Namespaces

- [__gnu_pbds](#)

Macros

- **#define PB_DS_TREE_CONST_NODE_ITERATOR_CLASS_C_DEC**
- **#define PB_DS_TREE_NODE_ITERATOR_CLASS_C_DEC**

5.257.1 Detailed Description

Contains an implementation class for bin_search_tree_.

5.258 node_iterators.hpp File Reference

Classes

- class [__gnu_pbds::detail::ov_tree_node_const_it_< Value_Type, Metadata_Type, _Alloc >](#)
- class [__gnu_pbds::detail::ov_tree_node_it_< Value_Type, Metadata_Type, _Alloc >](#)

Namespaces

- [__gnu_pbds](#)

Macros

- **#define PB_DS_OV_TREE_CONST_NODE_ITERATOR_C_DEC**
- **#define PB_DS_OV_TREE_NODE_ITERATOR_C_DEC**

5.258.1 Detailed Description

Contains an implementation class for ov_tree_.

5.259 node_metadata_selector.hpp File Reference

Classes

- struct [__gnu_pbds::detail::tree_metadata_helper< Node_Update, _BTp >](#)
- struct [__gnu_pbds::detail::tree_metadata_helper< Node_Update, false >](#)
- struct [__gnu_pbds::detail::tree_metadata_helper< Node_Update, true >](#)
- struct [__gnu_pbds::detail::tree_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc >](#)

Namespaces

- [__gnu_pbds](#)

5.259.1 Detailed Description

Contains an implementation class for trees.

5.260 node_metadata_selector.hpp File Reference

Classes

- struct [__gnu_pbds::detail::trie_metadata_helper< Node_Update, _BTp >](#)
- struct [__gnu_pbds::detail::trie_metadata_helper< Node_Update, false >](#)
- struct [__gnu_pbds::detail::trie_metadata_helper< Node_Update, true >](#)
- struct [__gnu_pbds::detail::trie_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc >](#)

Namespaces

- [__gnu_pbds](#)

5.260.1 Detailed Description

Contains an implementation class for tries.

5.261 null_node_metadata.hpp File Reference

Classes

- struct [__gnu_pbds::detail::dumnode_const_iterator< Key, Data, _Alloc >](#)

Namespaces

- [__gnu_pbds](#)

5.261.1 Detailed Description

Contains an implementation class for tree-like classes.

5.262 numeric_traits.h File Reference

Namespaces

- [__gnu_cxx](#)

Macros

- `#define __glibcxx_digits(_Tp)`
- `#define __glibcxx_digits10(_Tp)`
- `#define __glibcxx_floating(_Tp, _Fval, _Dval, _LDval)`
- `#define __glibcxx_max(_Tp)`
- `#define __glibcxx_max_digits10(_Tp)`
- `#define __glibcxx_max_exponent10(_Tp)`
- `#define __glibcxx_min(_Tp)`
- `#define __glibcxx_signed(_Tp)`

5.262.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

5.263 numericfwd.h File Reference

Namespaces

- [std](#)
- [std::__parallel](#)

Functions

- `template<typename _Iter, typename _Tp, typename _Tag >`
`_Tp std::__parallel::__accumulate_switch (_Iter, _Iter, _Tp, _Tag)`
- `template<typename _Iter, typename _Tp, typename _BinaryOper, typename _Tag >`
`_Tp std::__parallel::__accumulate_switch (_Iter, _Iter, _Tp, _BinaryOper, _Tag)`
- `template<typename _RAIter, typename _Tp, typename _BinaryOper >`
`_Tp std::__parallel::__accumulate_switch (_RAIter, _RAIter, _Tp, _BinaryOper, random_access_iterator_tag,`
`__gnu_parallel::__Parallelism __parallelism=__gnu_parallel::parallel_unbalanced)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper, typename _Tag1, typename _Tag2 >`
`_OIter std::__parallel::__adjacent_difference_switch (_Iter, _Iter, _OIter, _BinaryOper, _Tag1, _Tag2)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper >`
`_OIter std::__parallel::__adjacent_difference_switch (_Iter, _Iter, _OIter, _BinaryOper, random_access_iterator_tag, random_access_iterator_tag,`
`__gnu_parallel::__Parallelism __parallelism=__gnu_parallel::parallel_unbalanced)`
- `template<typename _RAIter1, typename _RAIter2, typename _Tp, typename BinaryFunction1, typename BinaryFunction2 >`
`_Tp std::__parallel::__inner_product_switch (_RAIter1, _RAIter1, _RAIter2, _Tp, BinaryFunction1, BinaryFunction2, random_access_iterator_tag, random_access_iterator_tag,`
`__gnu_parallel::__Parallelism=__gnu_parallel::parallel_unbalanced)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2, typename _Tag1, typename _Tag2 >`
`_Tp std::__parallel::__inner_product_switch (_Iter1, _Iter1, _Iter2, _Tp, _BinaryFunction1, _BinaryFunction2, _Tag1, _Tag2)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper, typename _Tag1, typename _Tag2 >`
`_OIter std::__parallel::__partial_sum_switch (_Iter, _Iter, _OIter, _BinaryOper, _Tag1, _Tag2)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper >`
`_OIter std::__parallel::__partial_sum_switch (_Iter, _Iter, _OIter, _BinaryOper, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter, typename _Tp >`
`_Tp std::__parallel::accumulate (_Iter, _Iter, _Tp)`
- `template<typename _Iter, typename _Tp >`
`_Tp std::__parallel::accumulate (_Iter, _Iter, _Tp, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _Tp >`
`_Tp std::__parallel::accumulate (_Iter, _Iter, _Tp, __gnu_parallel::Parallelism)`
- `template<typename _Iter, typename _Tp, typename _BinaryOper >`
`_Tp std::__parallel::accumulate (_Iter, _Iter, _Tp, _BinaryOper)`
- `template<typename _Iter, typename _Tp, typename _BinaryOper >`
`_Tp std::__parallel::accumulate (_Iter, _Iter, _Tp, _BinaryOper, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _Tp, typename _BinaryOper >`
`_Tp std::__parallel::accumulate (_Iter, _Iter, _Tp, _BinaryOper, __gnu_parallel::Parallelism)`
- `template<typename _Iter, typename _OIter >`
`_OIter std::__parallel::adjacent_difference (_Iter, _Iter, _OIter)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper >`
`_OIter std::__parallel::adjacent_difference (_Iter, _Iter, _OIter, _BinaryOper)`
- `template<typename _Iter, typename _OIter >`
`_OIter std::__parallel::adjacent_difference (_Iter, _Iter, _OIter, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper >`
`_OIter std::__parallel::adjacent_difference (_Iter, _Iter, _OIter, _BinaryOper, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _OIter >`
`_OIter std::__parallel::adjacent_difference (_Iter, _Iter, _OIter, __gnu_parallel::Parallelism)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper >`
`_OIter std::__parallel::adjacent_difference (_Iter, _Iter, _OIter, _BinaryOper, __gnu_parallel::Parallelism)`

- `template<typename _Iter1, typename _Iter2, typename _Tp >`
`_Tp std::__parallel::inner_product (_Iter1, _Iter1, _Iter2, _Tp)`
- `template<typename _Iter1, typename _Iter2, typename _Tp >`
`_Tp std::__parallel::inner_product (_Iter1, _Iter1, _Iter2, _Tp, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Tp >`
`_Tp std::__parallel::inner_product (_Iter1, _Iter1, _Iter2, _Tp, __gnu_parallel::Parallelism)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2 >`
`_Tp std::__parallel::inner_product (_Iter1, _Iter1, _Iter2, _Tp, _BinaryFunction1, _BinaryFunction2)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2 >`
`_Tp std::__parallel::inner_product (_Iter1, _Iter1, _Iter2, _Tp, _BinaryFunction1, _BinaryFunction2, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename BinaryFunction1, typename BinaryFunction2 >`
`_Tp std::__parallel::inner_product (_Iter1, _Iter1, _Iter2, _Tp, BinaryFunction1, BinaryFunction2, __gnu_parallel::Parallelism)`
- `template<typename _Iter, typename _OIter >`
`_OIter std::__parallel::partial_sum (_Iter, _Iter, _OIter, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper >`
`_OIter std::__parallel::partial_sum (_Iter, _Iter, _OIter, _BinaryOper, __gnu_parallel::sequential_tag)`
- `template<typename _Iter, typename _OIter >`
`_OIter std::__parallel::partial_sum (_Iter, _Iter, _OIter __result)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper >`
`_OIter std::__parallel::partial_sum (_Iter, _Iter, _OIter, _BinaryOper)`

5.263.1 Detailed Description

This file is a GNU parallel extension to the Standard C++ Library.

5.264 omp_loop.h File Reference

Namespaces

- [__gnu_parallel](#)

Functions

- `template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result >`
`_Op __gnu_parallel::for_each_template_random_access_omp_loop (_RAIter __begin, _RAIter __end, _Op __o, _Fu &__f, _Red __r, _Result __base, _Result &__output, typename std::iterator_traits< _RAIter >::difference_type __bound)`

5.264.1 Detailed Description

Parallelization of embarrassingly parallel execution by means of an OpenMP for loop. This file is a GNU parallel extension to the Standard C++ Library.

5.265 omp_loop_static.h File Reference

Namespaces

- [__gnu_parallel](#)

Functions

- `template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result >
_Op __gnu_parallel::__for_each_template_random_access_omp_loop_static (_RAIter __begin, _RAIter __end,
_Op __o, _Fu & __f, _Red __r, _Result __base, _Result & __output, typename std::iterator_traits< _RAIter >::difference_type __bound)`

5.265.1 Detailed Description

Parallelization of embarrassingly parallel execution by means of an OpenMP for loop with static scheduling. This file is a GNU parallel extension to the Standard C++ Library.

5.266 opt_random.h File Reference

Namespaces

- [std](#)

5.266.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<random>`.

5.267 order_statistics_imp.hpp File Reference

5.267.1 Detailed Description

Contains forward declarations for `order_statistics_key`

5.268 order_statistics_imp.hpp File Reference

5.268.1 Detailed Description

Contains forward declarations for `order_statistics_key`

5.269 `os_defines.h` File Reference

Macros

- `#define __NO_CTYPE`

5.269.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iosfwd>`.

5.270 `ostream_insert.h` File Reference

Namespaces

- [std](#)

Functions

- `template<typename _CharT, typename _Traits >`
`void std::__ostream_fill (basic_ostream< _CharT, _Traits > &__out, streamsize __n)`
- `template<typename _CharT, typename _Traits >`
`basic_ostream< _CharT, _Traits > & std::__ostream_insert (basic_ostream< _CharT, _Traits > &__out, const _CharT *__s, streamsize __n)`
- `template<typename _CharT, typename _Traits >`
`void std::__ostream_write (basic_ostream< _CharT, _Traits > &__out, const _CharT *__s, streamsize __n)`

5.270.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ostream>`.

5.271 `ov_tree_map.hpp` File Reference

Classes

- `class __gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >`
- `class __gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::cond_dtor< Size_Type >`

Namespaces

- [__gnu_pbds](#)

Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CONST_NODE_ITERATOR_NAME`
- `#define PB_DS_OV_TREE_NAME`
- `#define PB_DS_OV_TREE_TRAITS_BASE`

5.271.1 Detailed Description

Contains an implementation class for `ov_tree`.

5.272 pairing_heap.hpp File Reference

Classes

- class [__gnu_pbds::detail::pairing_heap< Value_Type, Cmp_Fn, _Alloc >](#)

Namespaces

- [__gnu_pbds](#)

Macros

- `#define PB_DS_ASSERT_NODE_CONSISTENT(_Node, _Bool)`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_P_HEAP_BASE`

5.272.1 Detailed Description

Contains an implementation class for a pairing heap.

5.273 par_loop.h File Reference

Namespaces

- [__gnu_parallel](#)

Functions

- `template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result >
_Op __gnu_parallel::__for_each_template_random_access_ed (_RAIter __begin, _RAIter __end, _Op __o, _Fu
& __f, _Red __r, _Result __base, _Result & __output, typename std::iterator_traits< _RAIter >::difference_type
__bound)`

5.273.1 Detailed Description

Parallelization of embarrassingly parallel execution by means of equal splitting. This file is a GNU parallel extension to the Standard C++ Library.

5.274 parallel.h File Reference

5.274.1 Detailed Description

End-user include file. Provides advanced settings and tuning options. This file is a GNU parallel extension to the Standard C++ Library.

5.275 parse_numbers.h File Reference

5.275.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<chrono>`.

5.276 partial_sum.h File Reference

Namespaces

- [__gnu_parallel](#)

Functions

- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >
_OutputIterator __gnu_parallel::__parallel_partial_sum (_Iter __begin, _Iter __end, _OutputIterator __result, ↔
_BinaryOperation __bin_op)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >
_OutputIterator __gnu_parallel::__parallel_partial_sum_basecase (_Iter __begin, _Iter __end, _OutputIterator
__result, _BinaryOperation __bin_op, typename std::iterator_traits< _Iter >::value_type __value)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >
_OutputIterator __gnu_parallel::__parallel_partial_sum_linear (_Iter __begin, _Iter __end, _OutputIterator __↔
result, _BinaryOperation __bin_op, typename std::iterator_traits< _Iter >::difference_type __n)`

5.276.1 Detailed Description

Parallel implementation of `std::partial_sum()`, i.e. prefix sums. This file is a GNU parallel extension to the Standard C++ Library.

5.277 partition.h File Reference

Namespaces

- [__gnu_parallel](#)

Macros

- [#define _GLIBCXX_VOLATILE](#)

Functions

- `template<typename _RAIter, typename _Compare >
void __gnu_parallel::__parallel_nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter, typename _Compare >
void __gnu_parallel::__parallel_partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter, typename _Predicate >
std::iterator_traits< _RAIter >::difference_type __gnu_parallel::__parallel_partition (_RAIter __begin, _RAIter __end, _Predicate __pred, _ThreadIndex __num_threads)`

5.277.1 Detailed Description

Parallel implementation of `std::partition()`, `std::nth_element()`, and `std::partial_sort()`. This file is a GNU parallel extension to the Standard C++ Library.

5.277.2 Macro Definition Documentation

5.277.2.1 [#define _GLIBCXX_VOLATILE](#)

Decide whether to declare certain variables volatile.

Definition at line 43 of file `partition.h`.

Referenced by `__gnu_parallel::__parallel_partition()`.

5.278 pat_trie.hpp File Reference

Classes

- class [__gnu_pbds::detail::pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >](#)

Namespaces

- [__gnu_pbds](#)

Macros

- `#define PB_DS_ASSERT_NODE_VALID(X)`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_PAT_TRIE_NAME`
- `#define PB_DS_PAT_TRIE_TRAITS_BASE`
- `#define PB_DS_RECURSIVE_COUNT_LEAFS(X)`

5.278.1 Detailed Description

Contains an implementation class for a patricia tree.

5.279 `pat_trie_base.hpp` File Reference

Classes

- struct [__gnu_pbds::detail::pat_trie_base](#)
- class [__gnu_pbds::detail::pat_trie_base::_Clter< Node, Leaf, Head, Inode, Is_Forward_Iterator >](#)
- struct [__gnu_pbds::detail::pat_trie_base::_Head< _ATraits, Metadata >](#)
- struct [__gnu_pbds::detail::pat_trie_base::_Inode< _ATraits, Metadata >](#)
- struct [__gnu_pbds::detail::pat_trie_base::_Inode< _ATraits, Metadata >::const_iterator](#)
- struct [__gnu_pbds::detail::pat_trie_base::_Inode< _ATraits, Metadata >::iterator](#)
- class [__gnu_pbds::detail::pat_trie_base::_Iter< Node, Leaf, Head, Inode, Is_Forward_Iterator >](#)
- struct [__gnu_pbds::detail::pat_trie_base::_Leaf< _ATraits, Metadata >](#)
- struct [__gnu_pbds::detail::pat_trie_base::_Metadata< Metadata, _Alloc >](#)
- struct [__gnu_pbds::detail::pat_trie_base::_Metadata< null_type, _Alloc >](#)
- struct [__gnu_pbds::detail::pat_trie_base::_Node_base< _ATraits, Metadata >](#)
- class [__gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _Clterator, Iterator, _Alloc >](#)
- class [__gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode, _Clterator, Iterator, _Alloc >](#)

Namespaces

- [__gnu_pbds](#)

Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CONST_IT_C_DEC`
- `#define PB_DS_CONST_ODIR_IT_C_DEC`
- `#define PB_DS_IT_C_DEC`
- `#define PB_DS_ODIR_IT_C_DEC`
- `#define PB_DS_PAT_TRIE_NODE_CONST_ITERATOR_C_DEC`
- `#define PB_DS_PAT_TRIE_NODE_ITERATOR_C_DEC`

5.279.1 Detailed Description

Contains the base class for a patricia tree.

5.280 `pod_char_traits.h` File Reference

Classes

- struct [__gnu_cxx::character< _Value, _Int, _St >](#)
- struct [std::char_traits< __gnu_cxx::character< _Value, _Int, _St > >](#)

Namespaces

- [__gnu_cxx](#)
- [std](#)

Functions

- `template<typename _Value , typename _Int , typename _St >`
`bool __gnu_cxx::operator< (const character< _Value, _Int, _St > &lhs, const character< _Value, _Int, _St > &rhs)`
- `template<typename _Value , typename _Int , typename _St >`
`bool __gnu_cxx::operator== (const character< _Value, _Int, _St > &lhs, const character< _Value, _Int, _St > &rhs)`

5.280.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

5.281 `point_const_iterator.hpp` File Reference

Classes

- class [__gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >](#)

Namespaces

- [__gnu_pbds](#)

5.281.1 Detailed Description

Contains an iterator class returned by the table's const find and insert methods.

5.282 `point_const_iterator.hpp` File Reference

Classes

- class [__gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >](#)

Namespaces

- [__gnu_pbds](#)

Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`

5.282.1 Detailed Description

Contains an iterator class returned by the table's const find and insert methods.

5.283 `point_const_iterator.hpp` File Reference

Classes

- class [point_const_iterator_](#)

5.283.1 Detailed Description

Contains an iterator class returned by the tables' const find and insert methods.

5.284 `point_iterator.hpp` File Reference

Classes

- class [point_iterator_](#)

5.284.1 Detailed Description

Contains an iterator class returned by the tables' find and insert methods.

5.285 point_iterators.hpp File Reference

Classes

- class [__gnu_pbds::detail::bin_search_tree_const_it< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc >](#)
- class [__gnu_pbds::detail::bin_search_tree_it< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc >](#)

Namespaces

- [__gnu_pbds](#)

Macros

- `#define PB_DS_TREE_CONST_IT_C_DEC`
- `#define PB_DS_TREE_CONST_ODIR_IT_C_DEC`
- `#define PB_DS_TREE_IT_C_DEC`
- `#define PB_DS_TREE_ODIR_IT_C_DEC`

5.285.1 Detailed Description

Contains an implementation class for `bin_search_tree_`.

5.286 pointer.h File Reference

Classes

- struct [__gnu_cxx::Invalid_type](#)
- class [__gnu_cxx::_Pointer_adapter< _Storage_policy >](#)
- class [__gnu_cxx::_Relative_pointer_impl< _Tp >](#)
- class [__gnu_cxx::_Relative_pointer_impl< const _Tp >](#)
- class [__gnu_cxx::_Std_pointer_impl< _Tp >](#)
- struct [__gnu_cxx::_Unqualified_type< _Tp >](#)

Namespaces

- [__gnu_cxx](#)
- [std](#)

Macros

- `#define _CXX_POINTER_ARITH_OPERATOR_SET(INT_TYPE)`
- `#define _GCC_CXX_POINTER_COMPARISON_OPERATION_SET(OPERATOR)`

Functions

- `template<typename _Tp1, typename _Tp2 >`
`bool __gnu_cxx::operator!= (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)`
- `template<typename _Tp1, typename _Tp2 >`
`bool __gnu_cxx::operator!= (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`
`bool __gnu_cxx::operator!= (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp >`
`bool __gnu_cxx::operator!= (const _Pointer_adapter< _Tp > &__lhs, int __rhs)`
- `template<typename _Tp >`
`bool __gnu_cxx::operator!= (int __lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<typename _Tp >`
`bool __gnu_cxx::operator!= (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`
`bool __gnu_cxx::operator< (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`
`bool __gnu_cxx::operator< (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`
`bool __gnu_cxx::operator< (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)`
- `template<typename _CharT, typename _Traits, typename _StoreT >`
`std::basic_ostream< _CharT, _Traits > & __gnu_cxx::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const _Pointer_adapter< _StoreT > &__p)`
- `template<typename _Tp1, typename _Tp2 >`
`bool __gnu_cxx::operator<= (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)`
- `template<typename _Tp1, typename _Tp2 >`
`bool __gnu_cxx::operator<= (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`
`bool __gnu_cxx::operator<= (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp >`
`bool __gnu_cxx::operator<= (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`
`bool __gnu_cxx::operator== (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`
`bool __gnu_cxx::operator== (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)`
- `template<typename _Tp1, typename _Tp2 >`
`bool __gnu_cxx::operator== (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp >`
`bool __gnu_cxx::operator== (const _Pointer_adapter< _Tp > &__lhs, int __rhs)`
- `template<typename _Tp >`
`bool __gnu_cxx::operator== (int __lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<typename _Tp >`
`bool __gnu_cxx::operator== (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`
`bool __gnu_cxx::operator> (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`
`bool __gnu_cxx::operator> (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)`

- `template<typename _Tp1, typename _Tp2 >`
`bool __gnu_cxx::operator> (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__`
`__rhs)`
- `template<typename _Tp >`
`bool __gnu_cxx::operator> (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`
`bool __gnu_cxx::operator>= (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)`
- `template<typename _Tp1, typename _Tp2 >`
`bool __gnu_cxx::operator>= (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__`
`__rhs)`
- `template<typename _Tp1, typename _Tp2 >`
`bool __gnu_cxx::operator>= (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp >`
`bool __gnu_cxx::operator>= (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__`
`rhs)`

5.286.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Author

Bob Walters

Provides reusable `_Pointer_adapter` for assisting in the development of custom pointer types that can be used with the standard containers via the `allocator::pointer` and `allocator::const_pointer` typedefs.

5.287 policy_access_fn_imps.hpp File Reference

5.287.1 Detailed Description

Contains an implementation class for a `binary_heap`.

5.288 policy_access_fn_imps.hpp File Reference

5.288.1 Detailed Description

Contains an implementation class for `bin_search_tree_`.

5.289 policy_access_fn_imps.hpp File Reference

5.289.1 Detailed Description

Contains implementations of `cc_ht_map_`'s policy access functions.

5.290 `policy_access_fn_imps.hpp` File Reference

5.290.1 Detailed Description

Contains implementations of `gp_ht_map_`'s policy access functions.

5.291 `policy_access_fn_imps.hpp` File Reference

5.291.1 Detailed Description

Contains an implementation class for `left_child_next_sibling_heap_`.

5.292 `policy_access_fn_imps.hpp` File Reference

5.292.1 Detailed Description

Contains an implementation class for `ov_tree`.

5.293 `policy_access_fn_imps.hpp` File Reference

5.293.1 Detailed Description

Contains an implementation class for `pat_trie`.

5.294 `pool_allocator.h` File Reference

Classes

- class [__gnu_cxx::__pool_alloc<_Tp>](#)
- class [__gnu_cxx::__pool_alloc_base](#)

Namespaces

- [__gnu_cxx](#)

Functions

- `template<typename _Tp>`
`bool __gnu_cxx::operator!= (const __pool_alloc<_Tp> &, const __pool_alloc<_Tp> &)`
- `template<typename _Tp>`
`bool __gnu_cxx::operator== (const __pool_alloc<_Tp> &, const __pool_alloc<_Tp> &)`

5.294.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

5.295 `postypes.h` File Reference

Classes

- class `std::fpos<_StateT>`

Namespaces

- `std`

Typedefs

- typedef long long `std::streamoff`
- typedef `fpos<mbstate_t>` `std::streampos`
- typedef `ptrdiff_t` `std::streamsize`
- typedef `fpos<mbstate_t>` `std::u16streampos`
- typedef `fpos<mbstate_t>` `std::u32streampos`
- typedef `fpos<mbstate_t>` `std::wstreampos`

Functions

- template<typename `_StateT`>
bool **`std::operator!=`** (const `fpos<_StateT>` &__lhs, const `fpos<_StateT>` &__rhs)
- template<typename `_StateT`>
bool **`std::operator==`** (const `fpos<_StateT>` &__lhs, const `fpos<_StateT>` &__rhs)

5.295.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iosfwd>`.

5.296 `predefined_ops.h` File Reference

Namespaces

- `__gnu_cxx`

Functions

- `template<typename _Compare >`
`_Iter_comp_iter< _Compare > __gnu_cxx::__ops::__iter_comp_iter (_Compare __comp)`
- `template<typename _Iterator >`
`_Iter_equals_iter< _Iterator > __gnu_cxx::__ops::__iter_comp_iter (_Iter_equal_to_iter, _Iterator __it)`
- `template<typename _Compare, typename _Iterator >`
`_Iter_comp_to_iter< _Compare, _Iterator > __gnu_cxx::__ops::__iter_comp_iter (_Iter_comp_iter< _Compare > __comp, _Iterator __it)`
- `_Iter_less_val __gnu_cxx::__ops::__iter_comp_val (_Iter_less_iter)`
- `_Iter_equal_to_val __gnu_cxx::__ops::__iter_comp_val (_Iter_equal_to_iter)`
- `template<typename _Compare >`
`_Iter_comp_val< _Compare > __gnu_cxx::__ops::__iter_comp_val (_Compare __comp)`
- `template<typename _Compare >`
`_Iter_comp_val< _Compare > __gnu_cxx::__ops::__iter_comp_val (_Iter_comp_iter< _Compare > __comp)`
- `template<typename _Compare, typename _Value >`
`_Iter_comp_to_val< _Compare, _Value > __gnu_cxx::__ops::__iter_comp_val (_Compare __comp, _Value &__val)`
- `_Iter_equal_to_iter __gnu_cxx::__ops::__iter_equal_to_iter ()`
- `_Iter_equal_to_val __gnu_cxx::__ops::__iter_equal_to_val ()`
- `template<typename _Value >`
`_Iter_equals_val< _Value > __gnu_cxx::__ops::__iter_equals_val (_Value &__val)`
- `_Iter_less_iter __gnu_cxx::__ops::__iter_less_iter ()`
- `_Iter_less_val __gnu_cxx::__ops::__iter_less_val ()`
- `template<typename _Predicate >`
`_Iter_negate< _Predicate > __gnu_cxx::__ops::__negate (_Iter_pred< _Predicate > __pred)`
- `template<typename _Predicate >`
`_Iter_pred< _Predicate > __gnu_cxx::__ops::__pred_iter (_Predicate __pred)`
- `_Val_less_iter __gnu_cxx::__ops::__val_comp_iter (_Iter_less_iter)`
- `template<typename _Compare >`
`_Val_comp_iter< _Compare > __gnu_cxx::__ops::__val_comp_iter (_Compare __comp)`
- `template<typename _Compare >`
`_Val_comp_iter< _Compare > __gnu_cxx::__ops::__val_comp_iter (_Iter_comp_iter< _Compare > __comp)`
- `_Val_less_iter __gnu_cxx::__ops::__val_less_iter ()`

5.296.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

5.297 prefix_search_node_update_imp.hpp File Reference

5.297.1 Detailed Description

Contains an implementation of `prefix_search_node_update`.

5.298 `priority_queue.hpp` File Reference

Classes

- class [__gnu_pbds::priority_queue<_Tv, Cmp_Fn, Tag, _Alloc>](#)

Namespaces

- [__gnu_pbds](#)

5.298.1 Detailed Description

Contains `priority_queues`.

5.299 `priority_queue_base_dispatch.hpp` File Reference

Classes

- struct [__gnu_pbds::detail::container_base_dispatch<_VTp, Cmp_Fn, _Alloc, binary_heap_tag, null_type>](#)
- struct [__gnu_pbds::detail::container_base_dispatch<_VTp, Cmp_Fn, _Alloc, binomial_heap_tag, null_type>](#)
- struct [__gnu_pbds::detail::container_base_dispatch<_VTp, Cmp_Fn, _Alloc, pairing_heap_tag, null_type>](#)
- struct [__gnu_pbds::detail::container_base_dispatch<_VTp, Cmp_Fn, _Alloc, rc_binomial_heap_tag, null_type>](#)
- struct [__gnu_pbds::detail::container_base_dispatch<_VTp, Cmp_Fn, _Alloc, thin_heap_tag, null_type>](#)

Namespaces

- [__gnu_pbds](#)

Macros

- `#define PB_DS_ASSERT_VALID(X)`
- `#define PB_DS_DEBUG_VERIFY(_Cond)`

5.299.1 Detailed Description

Contains an pqiative container dispatching base.

5.300 `probe_fn_base.hpp` File Reference

Classes

- class [__gnu_pbds::detail::probe_fn_base<_Alloc>](#)

Namespaces

- [__gnu_pbds](#)

5.300.1 Detailed Description

Contains a probe policy base.

5.301 profiler.h File Reference

Classes

- struct [__gnu_profile::__reentrance_guard](#)

Namespaces

- [__gnu_profile](#)

Macros

- #define [__profcxx_hashtable_construct\(__x...\)](#)
- #define [__profcxx_hashtable_construct2\(__x...\)](#)
- #define [__profcxx_hashtable_destruct\(__x...\)](#)
- #define [__profcxx_hashtable_destruct2\(__x...\)](#)
- #define [__profcxx_hashtable_resize\(__x...\)](#)
- #define [__profcxx_inefficient_hash_is_on\(\)](#)
- #define [__profcxx_is_invalid\(\)](#)
- #define [__profcxx_is_off\(\)](#)
- #define [__profcxx_is_on\(\)](#)
- #define [__profcxx_list_construct\(__x...\)](#)
- #define [__profcxx_list_construct2\(__x...\)](#)
- #define [__profcxx_list_destruct\(__x...\)](#)
- #define [__profcxx_list_destruct2\(__x...\)](#)
- #define [__profcxx_list_insert\(__x...\)](#)
- #define [__profcxx_list_invalid_operator\(__x...\)](#)
- #define [__profcxx_list_iterate\(__x...\)](#)
- #define [__profcxx_list_operation\(__x...\)](#)
- #define [__profcxx_list_rewind\(__x...\)](#)
- #define [__profcxx_map_to_unordered_map_construct\(__x...\)](#)
- #define [__profcxx_map_to_unordered_map_destruct\(__x...\)](#)
- #define [__profcxx_map_to_unordered_map_erase\(__x...\)](#)
- #define [__profcxx_map_to_unordered_map_find\(__x...\)](#)
- #define [__profcxx_map_to_unordered_map_insert\(__x...\)](#)
- #define [__profcxx_map_to_unordered_map_invalidate\(__x...\)](#)
- #define [__profcxx_map_to_unordered_map_iterate\(__x...\)](#)
- #define [__profcxx_report\(\)](#)

- `#define __profcxx_turn_off()`
- `#define __profcxx_turn_on()`
- `#define __profcxx_vector_construct(__x...)`
- `#define __profcxx_vector_construct2(__x...)`
- `#define __profcxx_vector_destruct(__x...)`
- `#define __profcxx_vector_destruct2(__x...)`
- `#define __profcxx_vector_find(__x...)`
- `#define __profcxx_vector_insert(__x...)`
- `#define __profcxx_vector_invalid_operator(__x...)`
- `#define __profcxx_vector_iterate(__x...)`
- `#define __profcxx_vector_resize(__x...)`
- `#define __profcxx_vector_resize2(__x...)`
- `#define _GLIBCXX_PROFILE_DATA(__name)`
- `#define _GLIBCXX_PROFILE_DEFINE_DATA(__type, __name, __initial_value...)`
- `#define _GLIBCXX_PROFILE_DEFINE_UNINIT_DATA(__type, __name)`
- `#define _GLIBCXX_PROFILE_MAX_STACK_DEPTH`
- `#define _GLIBCXX_PROFILE_MAX_STACK_DEPTH_ENV_VAR`
- `#define _GLIBCXX_PROFILE_MAX_WARN_COUNT`
- `#define _GLIBCXX_PROFILE_MAX_WARN_COUNT_ENV_VAR`
- `#define _GLIBCXX_PROFILE_MEM_PER_DIAGNOSTIC`
- `#define _GLIBCXX_PROFILE_MEM_PER_DIAGNOSTIC_ENV_VAR`
- `#define _GLIBCXX_PROFILE_REENTRANCE_GUARD(__x...)`
- `#define _GLIBCXX_PROFILE_TRACE_ENV_VAR`
- `#define _GLIBCXX_PROFILE_TRACE_PATH_ROOT`

Functions

- `bool __gnu_profile::__is_invalid ()`
- `bool __gnu_profile::__is_off ()`
- `bool __gnu_profile::__is_on ()`
- `void __gnu_profile::__report (void)`
- `void __gnu_profile::__trace_hash_func_construct (const void *)`
- `void __gnu_profile::__trace_hash_func_destruct (const void *, std::size_t, std::size_t, std::size_t)`
- `void __gnu_profile::__trace_hashtable_size_construct (const void *, std::size_t)`
- `void __gnu_profile::__trace_hashtable_size_destruct (const void *, std::size_t, std::size_t)`
- `void __gnu_profile::__trace_hashtable_size_resize (const void *, std::size_t, std::size_t)`
- `void __gnu_profile::__trace_list_to_set_construct (const void *)`
- `void __gnu_profile::__trace_list_to_set_destruct (const void *)`
- `void __gnu_profile::__trace_list_to_set_find (const void *, std::size_t)`
- `void __gnu_profile::__trace_list_to_set_insert (const void *, std::size_t, std::size_t)`
- `void __gnu_profile::__trace_list_to_set_invalid_operator (const void *)`
- `void __gnu_profile::__trace_list_to_set_iterate (const void *, std::size_t)`
- `void __gnu_profile::__trace_list_to_slist_construct (const void *)`
- `void __gnu_profile::__trace_list_to_slist_destruct (const void *)`
- `void __gnu_profile::__trace_list_to_slist_operation (const void *)`
- `void __gnu_profile::__trace_list_to_slist_rewind (const void *)`
- `void __gnu_profile::__trace_list_to_vector_construct (const void *)`
- `void __gnu_profile::__trace_list_to_vector_destruct (const void *)`
- `void __gnu_profile::__trace_list_to_vector_insert (const void *, std::size_t, std::size_t)`
- `void __gnu_profile::__trace_list_to_vector_invalid_operator (const void *)`

- void **__gnu_profile::__trace_list_to_vector_iterate** (const void *, std::size_t)
- void **__gnu_profile::__trace_list_to_vector_resize** (const void *, std::size_t, std::size_t)
- void **__gnu_profile::__trace_map_to_unordered_map_construct** (const void *)
- void **__gnu_profile::__trace_map_to_unordered_map_destruct** (const void *)
- void **__gnu_profile::__trace_map_to_unordered_map_erase** (const void *, std::size_t, std::size_t)
- void **__gnu_profile::__trace_map_to_unordered_map_find** (const void *, std::size_t)
- void **__gnu_profile::__trace_map_to_unordered_map_insert** (const void *, std::size_t, std::size_t)
- void **__gnu_profile::__trace_map_to_unordered_map_invalidate** (const void *)
- void **__gnu_profile::__trace_map_to_unordered_map_iterate** (const void *, std::size_t)
- void **__gnu_profile::__trace_vector_size_construct** (const void *, std::size_t)
- void **__gnu_profile::__trace_vector_size_destruct** (const void *, std::size_t, std::size_t)
- void **__gnu_profile::__trace_vector_size_resize** (const void *, std::size_t, std::size_t)
- void **__gnu_profile::__trace_vector_to_list_construct** (const void *)
- void **__gnu_profile::__trace_vector_to_list_destruct** (const void *)
- void **__gnu_profile::__trace_vector_to_list_find** (const void *, std::size_t)
- void **__gnu_profile::__trace_vector_to_list_insert** (const void *, std::size_t, std::size_t)
- void **__gnu_profile::__trace_vector_to_list_invalid_operator** (const void *)
- void **__gnu_profile::__trace_vector_to_list_iterate** (const void *, std::size_t)
- void **__gnu_profile::__trace_vector_to_list_resize** (const void *, std::size_t, std::size_t)
- bool **__gnu_profile::__turn_off** ()
- bool **__gnu_profile::__turn_on** ()

5.301.1 Detailed Description

Interface of the profiling runtime library.

5.302 profiler_algos.h File Reference

Namespaces

- [__gnu_profile](#)

Functions

- template<typename _InputIterator, typename _Function >
_Function **__gnu_profile::__for_each** (_InputIterator __first, _InputIterator __last, _Function __f)
- template<typename _Container >
void **__gnu_profile::__insert_top_n** (_Container &__output, const typename _Container::value_type &__value, typename _Container::size_type __n)
- template<typename _ForwardIterator, typename _Tp >
_ForwardIterator **__gnu_profile::__remove** (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)
- template<typename _Container >
void **__gnu_profile::__top_n** (const _Container &__input, _Container &__output, typename _Container::size_type __n)

5.302.1 Detailed Description

Algorithms used by the profile extension.

This file is needed to avoid including `<algorithm>` or `<bits/stl_algo.h>`. Including those files would result in recursive includes. These implementations are oversimplified. In general, efficiency may be sacrificed to minimize maintenance overhead.

5.303 profiler_container_size.h File Reference

Classes

- class [__gnu_profile::__container_size_info](#)
- class [__gnu_profile::__container_size_stack_info](#)
- class [__gnu_profile::__trace_container_size](#)

Namespaces

- [__gnu_profile](#)

5.303.1 Detailed Description

Diagnostics for container sizes.

5.304 profiler_hash_func.h File Reference

Classes

- class [__gnu_profile::__hashfunc_info](#)
- class [__gnu_profile::__hashfunc_stack_info](#)
- class [__gnu_profile::__trace_hash_func](#)

Namespaces

- [__gnu_profile](#)

Functions

- void [__gnu_profile::__trace_hash_func_construct](#) (const void *)
- void [__gnu_profile::__trace_hash_func_destruct](#) (const void *, std::size_t, std::size_t, std::size_t)
- void [__gnu_profile::__trace_hash_func_init](#) ()
- void [__gnu_profile::__trace_hash_func_report](#) (FILE * __f, __warning_vector_t & __warnings)

5.304.1 Detailed Description

Data structures to represent profiling traces.

5.305 profiler_hashtable_size.h File Reference

Classes

- class [__gnu_profile::__trace_hashtable_size](#)

Namespaces

- [__gnu_profile](#)

Functions

- void [__gnu_profile::__trace_hashtable_size_construct](#) (const void *, std::size_t)
- void [__gnu_profile::__trace_hashtable_size_destruct](#) (const void *, std::size_t, std::size_t)
- void [__gnu_profile::__trace_hashtable_size_init](#) ()
- void [__gnu_profile::__trace_hashtable_size_report](#) (FILE *__f, __warning_vector_t &__warnings)
- void [__gnu_profile::__trace_hashtable_size_resize](#) (const void *, std::size_t, std::size_t)

5.305.1 Detailed Description

Collection of hashtable size traces.

5.306 profiler_list_to_slist.h File Reference

Namespaces

- [__gnu_profile](#)

Functions

- void [__gnu_profile::__trace_list_to_slist_construct](#) (const void *)
- void [__gnu_profile::__trace_list_to_slist_destruct](#) (const void *)
- void [__gnu_profile::__trace_list_to_slist_init](#) ()
- void [__gnu_profile::__trace_list_to_slist_operation](#) (const void *)
- void [__gnu_profile::__trace_list_to_slist_report](#) (FILE *__f, __warning_vector_t &__warnings)
- void [__gnu_profile::__trace_list_to_slist_rewind](#) (const void *)

5.306.1 Detailed Description

Diagnostics for list to slist.

5.307 profiler_list_to_vector.h File Reference

Classes

- class [__gnu_profile::__list2vector_info](#)

Namespaces

- [__gnu_profile](#)

Functions

- void [__gnu_profile::__trace_list_to_vector_construct](#) (const void *)
- void [__gnu_profile::__trace_list_to_vector_destruct](#) (const void *)
- void [__gnu_profile::__trace_list_to_vector_init](#) ()
- void [__gnu_profile::__trace_list_to_vector_insert](#) (const void *, std::size_t, std::size_t)
- void [__gnu_profile::__trace_list_to_vector_invalid_operator](#) (const void *)
- void [__gnu_profile::__trace_list_to_vector_iterate](#) (const void *, std::size_t)
- void [__gnu_profile::__trace_list_to_vector_report](#) (FILE *__f, __warning_vector_t &__warnings)
- void [__gnu_profile::__trace_list_to_vector_resize](#) (const void *, std::size_t, std::size_t)

5.307.1 Detailed Description

diagnostics for list to vector.

5.308 profiler_map_to_unordered_map.h File Reference

Classes

- class [__gnu_profile::__map2umap_info](#)
- class [__gnu_profile::__map2umap_stack_info](#)
- class [__gnu_profile::__trace_map2umap](#)

Namespaces

- [__gnu_profile](#)

Functions

- `int __gnu_profile::__log2 (std::size_t __size)`
- `float __gnu_profile::__map_erase_cost (std::size_t __size)`
- `float __gnu_profile::__map_find_cost (std::size_t __size)`
- `float __gnu_profile::__map_insert_cost (std::size_t __size)`
- `void __gnu_profile::__trace_map_to_unordered_map_construct (const void *)`
- `void __gnu_profile::__trace_map_to_unordered_map_destruct (const void *)`
- `void __gnu_profile::__trace_map_to_unordered_map_erase (const void *, std::size_t, std::size_t)`
- `void __gnu_profile::__trace_map_to_unordered_map_find (const void *, std::size_t)`
- `void __gnu_profile::__trace_map_to_unordered_map_init ()`
- `void __gnu_profile::__trace_map_to_unordered_map_insert (const void *, std::size_t, std::size_t)`
- `void __gnu_profile::__trace_map_to_unordered_map_invalidate (const void *)`
- `void __gnu_profile::__trace_map_to_unordered_map_iterate (const void *, std::size_t)`
- `void __gnu_profile::__trace_map_to_unordered_map_report (FILE *__f, __warning_vector_t &__warnings)`

5.308.1 Detailed Description

Diagnostics for `map` to `unordered_map`.

5.309 profiler_node.h File Reference

Classes

- class [__gnu_profile::__object_info_base](#)
- class [__gnu_profile::__stack_hash](#)
- class [__gnu_profile::__stack_info_base](#) < [__object_info](#) >

Namespaces

- [__gnu_profile](#)

Typedefs

- `typedef void * __gnu_profile::__instruction_address_t`
- `typedef const void * __gnu_profile::__object_t`
- `typedef std::vector< __instruction_address_t > __gnu_profile::__stack_npt`
- `typedef __stack_npt * __gnu_profile::__stack_t`

Functions

- `__stack_t __gnu_profile::__get_stack ()`
- `std::size_t __gnu_profile::__size (__stack_t __stack)`
- `std::size_t __gnu_profile::__stack_max_depth ()`
- `void __gnu_profile::__write (FILE *__f, __stack_t __stack)`

5.309.1 Detailed Description

Data structures to represent a single profiling event.

5.310 profiler_state.h File Reference

Namespaces

- [__gnu_profile](#)

Enumerations

- enum `__state_type` { `__ON`, `__OFF`, `__INVALID` }

Functions

- bool `__gnu_profile::__is_invalid ()`
- bool `__gnu_profile::__is_off ()`
- bool `__gnu_profile::__is_on ()`
- bool `__gnu_profile::__turn (__state_type __s)`
- bool `__gnu_profile::__turn_off ()`
- bool `__gnu_profile::__turn_on ()`
- `__gnu_profile::__GLIBCXX_PROFILE_DEFINE_DATA (__state_type, __state, __INVALID)`

5.310.1 Detailed Description

Global profiler state.

5.311 profiler_trace.h File Reference

Classes

- class [__gnu_profile::__trace_base< __object_info, __stack_info >](#)
- struct [__gnu_profile::__warning_data](#)

Namespaces

- [__gnu_profile](#)

Macros

- `#define _GLIBCXX_IMPL_UNORDERED_MAP`

Typedefs

- typedef std::vector< __cost_factor * > **__gnu_profile::__cost_factor_vector**
- typedef std::unordered_map< [std::string](#), [std::string](#) > **__gnu_profile::__env_t**
- typedef std::vector< __warning_data > **__gnu_profile::__warning_vector_t**

Functions

- std::size_t **__gnu_profile::__env_to_size_t** (const char * __env_var, std::size_t __default_value)
- int **__gnu_profile::__log_magnitude** (float __f)
- std::size_t **__gnu_profile::__max_mem** ()
- FILE * **__gnu_profile::__open_output_file** (const char * __extension)
- bool **__gnu_profile::__profcxx_init** ()
- void **__gnu_profile::__profcxx_init_unconditional** ()
- void **__gnu_profile::__read_cost_factors** ()
- void **__gnu_profile::__report** (void)
- void **__gnu_profile::__set_cost_factors** ()
- void **__gnu_profile::__set_max_mem** ()
- void **__gnu_profile::__set_max_stack_trace_depth** ()
- void **__gnu_profile::__set_max_warn_count** ()
- void **__gnu_profile::__set_trace_path** ()
- std::size_t **__gnu_profile::__stack_max_depth** ()
- void **__gnu_profile::__trace_hash_func_init** ()
- void **__gnu_profile::__trace_hash_func_report** (FILE * __f, __warning_vector_t & __warnings)
- void **__gnu_profile::__trace_hashtable_size_init** ()
- void **__gnu_profile::__trace_hashtable_size_report** (FILE * __f, __warning_vector_t & __warnings)
- void **__gnu_profile::__trace_list_to_slist_init** ()
- void **__gnu_profile::__trace_list_to_slist_report** (FILE * __f, __warning_vector_t & __warnings)
- void **__gnu_profile::__trace_list_to_vector_init** ()
- void **__gnu_profile::__trace_list_to_vector_report** (FILE * __f, __warning_vector_t & __warnings)
- void **__gnu_profile::__trace_map_to_unordered_map_init** ()
- void **__gnu_profile::__trace_map_to_unordered_map_report** (FILE * __f, __warning_vector_t & __warnings)
- void **__gnu_profile::__trace_vector_size_init** ()
- void **__gnu_profile::__trace_vector_size_report** (FILE *, __warning_vector_t &)
- void **__gnu_profile::__trace_vector_to_list_init** ()
- void **__gnu_profile::__trace_vector_to_list_report** (FILE *, __warning_vector_t &)
- void **__gnu_profile::__write_cost_factors** ()
- **__gnu_profile::__GLIBCXX_PROFILE_DEFINE_DATA** (__trace_hash_func *, _S_hash_func, 0)
- **__gnu_profile::__GLIBCXX_PROFILE_DEFINE_DATA** (__trace_hashtable_size *, _S_hashtable_size, 0)
- **__gnu_profile::__GLIBCXX_PROFILE_DEFINE_DATA** (__trace_map2umap *, _S_map2umap, 0)
- **__gnu_profile::__GLIBCXX_PROFILE_DEFINE_DATA** (__trace_vector_size *, _S_vector_size, 0)
- **__gnu_profile::__GLIBCXX_PROFILE_DEFINE_DATA** (__trace_vector_to_list *, _S_vector_to_list, 0)
- **__gnu_profile::__GLIBCXX_PROFILE_DEFINE_DATA** (__trace_list_to_slist *, _S_list_to_slist, 0)
- **__gnu_profile::__GLIBCXX_PROFILE_DEFINE_DATA** (__trace_list_to_vector *, _S_list_to_vector, 0)
- **__gnu_profile::__GLIBCXX_PROFILE_DEFINE_DATA** (__cost_factor, __vector_shift_cost_factor, {"__vector_↵
shift_cost_factor", 1.0})
- **__gnu_profile::__GLIBCXX_PROFILE_DEFINE_DATA** (__cost_factor, __vector_iterate_cost_factor, {"__↵
vector_iterate_cost_factor", 1.0})
- **__gnu_profile::__GLIBCXX_PROFILE_DEFINE_DATA** (__cost_factor, __vector_resize_cost_factor, {"__↵
vector_resize_cost_factor", 1.0})

- `__gnu_profile::__GLIBCXX_PROFILE_DEFINE_DATA` (`__cost_factor`, `__list_shift_cost_factor`, {"__list_shift_↵
cost_factor", 0.0})
- `__gnu_profile::__GLIBCXX_PROFILE_DEFINE_DATA` (`__cost_factor`, `__list_iterate_cost_factor`, {"__list_↵
iterate_cost_factor", 10.0})
- `__gnu_profile::__GLIBCXX_PROFILE_DEFINE_DATA` (`__cost_factor`, `__list_resize_cost_factor`, {"__list_↵
resize_cost_factor", 0.0})
- `__gnu_profile::__GLIBCXX_PROFILE_DEFINE_DATA` (`__cost_factor`, `__map_insert_cost_factor`, {"__map_↵
insert_cost_factor", 1.5})
- `__gnu_profile::__GLIBCXX_PROFILE_DEFINE_DATA` (`__cost_factor`, `__map_erase_cost_factor`, {"__map_↵
erase_cost_factor", 1.5})
- `__gnu_profile::__GLIBCXX_PROFILE_DEFINE_DATA` (`__cost_factor`, `__map_find_cost_factor`, {"__map_find_↵
cost_factor", 1})
- `__gnu_profile::__GLIBCXX_PROFILE_DEFINE_DATA` (`__cost_factor`, `__map_iterate_cost_factor`, {"__map_↵
iterate_cost_factor", 2.3})
- `__gnu_profile::__GLIBCXX_PROFILE_DEFINE_DATA` (`__cost_factor`, `__umap_insert_cost_factor`, {"__umap_↵
insert_cost_factor", 12.0})
- `__gnu_profile::__GLIBCXX_PROFILE_DEFINE_DATA` (`__cost_factor`, `__umap_erase_cost_factor`, {"__umap_↵
erase_cost_factor", 12.0})
- `__gnu_profile::__GLIBCXX_PROFILE_DEFINE_DATA` (`__cost_factor`, `__umap_find_cost_factor`, {"__umap_↵
find_cost_factor", 10.0})
- `__gnu_profile::__GLIBCXX_PROFILE_DEFINE_DATA` (`__cost_factor`, `__umap_iterate_cost_factor`, {"__umap_↵
iterate_cost_factor", 1.7})
- `__gnu_profile::__GLIBCXX_PROFILE_DEFINE_DATA` (`__cost_factor_vector *`, `__cost_factors`, 0)
- `__gnu_profile::__GLIBCXX_PROFILE_DEFINE_DATA` (`const char *`, `_S_trace_file_name`, `_GLIBCXX_PROFI_↵
ILE_TRACE_PATH_ROOT`)
- `__gnu_profile::__GLIBCXX_PROFILE_DEFINE_DATA` (`std::size_t`, `_S_max_warn_count`, `_GLIBCXX_PROFI_↵
LE_MAX_WARN_COUNT`)
- `__gnu_profile::__GLIBCXX_PROFILE_DEFINE_DATA` (`std::size_t`, `_S_max_stack_depth`, `_GLIBCXX_PROFI_↵
ILE_MAX_STACK_DEPTH`)
- `__gnu_profile::__GLIBCXX_PROFILE_DEFINE_DATA` (`std::size_t`, `_S_max_mem`, `_GLIBCXX_PROFILE_M_↵
EM_PER_DIAGNOSTIC`)
- `__gnu_profile::__GLIBCXX_PROFILE_DEFINE_UNINIT_DATA` (`__env_t`, `__env`)
- `__gnu_profile::__GLIBCXX_PROFILE_DEFINE_UNINIT_DATA` (`__gnu_cxx::__mutex`, `__global_lock`)

5.311.1 Detailed Description

Data structures to represent profiling traces.

5.312 profiler_vector_size.h File Reference

Classes

- class [__gnu_profile::__trace_vector_size](#)

Namespaces

- [__gnu_profile](#)

Functions

- void `__gnu_profile::__trace_vector_size_construct` (const void *, std::size_t)
- void `__gnu_profile::__trace_vector_size_destruct` (const void *, std::size_t, std::size_t)
- void `__gnu_profile::__trace_vector_size_init` ()
- void `__gnu_profile::__trace_vector_size_report` (FILE *, __warning_vector_t &)
- void `__gnu_profile::__trace_vector_size_resize` (const void *, std::size_t, std::size_t)

5.312.1 Detailed Description

Collection of vector size traces.

5.313 profiler_vector_to_list.h File Reference

Classes

- class [__gnu_profile::__trace_vector_to_list](#)
- class [__gnu_profile::__vector2list_info](#)
- class [__gnu_profile::__vector2list_stack_info](#)

Namespaces

- [__gnu_profile](#)

Functions

- void `__gnu_profile::__trace_vector_to_list_construct` (const void *)
- void `__gnu_profile::__trace_vector_to_list_destruct` (const void *)
- void `__gnu_profile::__trace_vector_to_list_find` (const void *, std::size_t)
- void `__gnu_profile::__trace_vector_to_list_init` ()
- void `__gnu_profile::__trace_vector_to_list_insert` (const void *, std::size_t, std::size_t)
- void `__gnu_profile::__trace_vector_to_list_invalid_operator` (const void *)
- void `__gnu_profile::__trace_vector_to_list_iterate` (const void *, std::size_t)
- void `__gnu_profile::__trace_vector_to_list_report` (FILE *, __warning_vector_t &)
- void `__gnu_profile::__trace_vector_to_list_resize` (const void *, std::size_t, std::size_t)

5.313.1 Detailed Description

diagnostics for vector to list.

5.314 ptr_traits.h File Reference

Classes

- struct [std::pointer_traits<_Ptr>](#)
- struct [std::pointer_traits<_Tp*>](#)

Namespaces

- [std](#)

5.314.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

5.315 quadratic_probe_fn_imp.hpp File Reference

5.315.1 Detailed Description

Contains a probe policy implementation

5.316 queue.h File Reference

Classes

- class [__gnu_parallel::_RestrictedBoundedConcurrentQueue<_Tp>](#)

Namespaces

- [__gnu_parallel](#)

Macros

- [#define _GLIBCXX_VOLATILE](#)

5.316.1 Detailed Description

Lock-free double-ended queue. This file is a GNU parallel extension to the Standard C++ Library.

5.316.2 Macro Definition Documentation

5.316.2.1 [#define _GLIBCXX_VOLATILE](#)

Decide whether to declare certain variable volatile in this file.

Definition at line 40 of file queue.h.

5.317 quicksort.h File Reference

Namespaces

- [__gnu_parallel](#)

Functions

- `template<typename _RAIter, typename _Compare >`
`void __gnu_parallel::__parallel_sort_qs (_RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __num_threads)`
- `template<typename _RAIter, typename _Compare >`
`void __gnu_parallel::__parallel_sort_qs_conquer (_RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __num_threads)`
- `template<typename _RAIter, typename _Compare >`
`std::iterator_traits< _RAIter >::difference_type __gnu_parallel::__parallel_sort_qs_divide (_RAIter __begin, _RAIter __end, _Compare __comp, typename std::iterator_traits< _RAIter >::difference_type __pivot_rank, typename std::iterator_traits< _RAIter >::difference_type __num_samples, _ThreadIndex __num_threads)`

5.317.1 Detailed Description

Implementation of a unbalanced parallel quicksort (in-place). This file is a GNU parallel extension to the Standard C++ Library.

5.318 r_erase_fn_imps.hpp File Reference

5.318.1 Detailed Description

Contains an implementation class for `bin_search_tree_`.

5.319 r_erase_fn_imps.hpp File Reference

5.319.1 Detailed Description

Contains an implementation class for `pat_trie`.

5.320 random.h File Reference

Classes

- class `std::bernoulli_distribution`
- struct `std::bernoulli_distribution::param_type`
- class `std::binomial_distribution<_IntType>`
- struct `std::binomial_distribution<_IntType>::param_type`
- class `std::cauchy_distribution<_RealType>`
- struct `std::cauchy_distribution<_RealType>::param_type`
- class `std::chi_squared_distribution<_RealType>`
- struct `std::chi_squared_distribution<_RealType>::param_type`
- class `std::discard_block_engine<_RandomNumberEngine, __p, __r>`
- class `std::discrete_distribution<_IntType>`
- struct `std::discrete_distribution<_IntType>::param_type`
- class `std::exponential_distribution<_RealType>`
- struct `std::exponential_distribution<_RealType>::param_type`
- class `std::extreme_value_distribution<_RealType>`
- struct `std::extreme_value_distribution<_RealType>::param_type`
- class `std::fisher_f_distribution<_RealType>`
- struct `std::fisher_f_distribution<_RealType>::param_type`
- class `std::gamma_distribution<_RealType>`
- struct `std::gamma_distribution<_RealType>::param_type`
- class `std::geometric_distribution<_IntType>`
- struct `std::geometric_distribution<_IntType>::param_type`
- class `std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType>`
- class `std::linear_congruential_engine<_UIntType, __a, __c, __m>`
- class `std::lognormal_distribution<_RealType>`
- struct `std::lognormal_distribution<_RealType>::param_type`
- class `std::mersenne_twister_engine<_UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f>`
- class `std::negative_binomial_distribution<_IntType>`
- struct `std::negative_binomial_distribution<_IntType>::param_type`
- class `std::normal_distribution<_RealType>`
- struct `std::normal_distribution<_RealType>::param_type`
- class `std::piecewise_constant_distribution<_RealType>`
- struct `std::piecewise_constant_distribution<_RealType>::param_type`
- class `std::piecewise_linear_distribution<_RealType>`
- struct `std::piecewise_linear_distribution<_RealType>::param_type`
- class `std::poisson_distribution<_IntType>`
- struct `std::poisson_distribution<_IntType>::param_type`
- class `std::random_device`
- class `std::seed_seq`
- class `std::shuffle_order_engine<_RandomNumberEngine, __k>`
- class `std::student_t_distribution<_RealType>`
- struct `std::student_t_distribution<_RealType>::param_type`
- class `std::uniform_int_distribution<_IntType>`
- struct `std::uniform_int_distribution<_IntType>::param_type`
- class `std::uniform_real_distribution<_RealType>`
- struct `std::uniform_real_distribution<_RealType>::param_type`
- class `std::weibull_distribution<_RealType>`
- struct `std::weibull_distribution<_RealType>::param_type`

Namespaces

- [std](#)
- [std::__detail](#)

Typedefs

- typedef minstd_rand0 **std::default_random_engine**
- typedef shuffle_order_engine< minstd_rand0, 256 > **std::knuth_b**
- typedef linear_congruential_engine< uint_fast32_t, 48271UL, 0UL, 2147483647UL > [std::minstd_rand](#)
- typedef linear_congruential_engine< uint_fast32_t, 16807UL, 0UL, 2147483647UL > [std::minstd_rand0](#)
- typedef mersenne_twister_engine< uint_fast32_t, 32, 624, 397, 31, 0x9908b0dfUL, 11, 0xffffffffUL, 7, 0x9d2c5680UL, 15, 0xefc60000UL, 18, 1812433253UL > [std::mt19937](#)
- typedef mersenne_twister_engine< uint_fast64_t, 64, 312, 156, 31, 0xb5026f5aa96619e9ULL, 29, 0x5555555555555555ULL, 17, 0x71d67ffeda60000ULL, 37, 0xffff7eee00000000ULL, 43, 6364136223846793005ULL > [std::mt19937_64](#)
- typedef discard_block_engine< ranlux24_base, 223, 23 > **std::ranlux24**
- typedef subtract_with_carry_engine< uint_fast32_t, 24, 10, 24 > **std::ranlux24_base**
- typedef discard_block_engine< ranlux48_base, 389, 11 > **std::ranlux48**
- typedef subtract_with_carry_engine< uint_fast64_t, 48, 5, 12 > **std::ranlux48_base**

Functions

- template<typename _RealType, size_t __bits, typename _UniformRandomNumberGenerator >
_RealType [std::generate_canonical](#) (_UniformRandomNumberGenerator &__g)
- template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
bool [std::operator!=](#) (const [std::linear_congruential_engine](#)< _UIntType, __a, __c, __m > &__lhs, const [std::linear_congruential_engine](#)< _UIntType, __a, __c, __m > &__rhs)
- template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f>
bool [std::operator!=](#) (const [std::mersenne_twister_engine](#)< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > &__lhs, const [std::mersenne_twister_engine](#)< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > &__rhs)
- template<typename _UIntType, size_t __w, size_t __s, size_t __r>
bool [std::operator!=](#) (const [std::subtract_with_carry_engine](#)< _UIntType, __w, __s, __r > &__lhs, const [std::subtract_with_carry_engine](#)< _UIntType, __w, __s, __r > &__rhs)
- template<typename _RandomNumberEngine, size_t __p, size_t __r>
bool [std::operator!=](#) (const [std::discard_block_engine](#)< _RandomNumberEngine, __p, __r > &__lhs, const [std::discard_block_engine](#)< _RandomNumberEngine, __p, __r > &__rhs)
- template<typename _RandomNumberEngine, size_t __w, typename _UIntType >
bool [std::operator!=](#) (const [std::independent_bits_engine](#)< _RandomNumberEngine, __w, _UIntType > &__lhs, const [std::independent_bits_engine](#)< _RandomNumberEngine, __w, _UIntType > &__rhs)
- template<typename _RandomNumberEngine, size_t __k>
bool [std::operator!=](#) (const [std::shuffle_order_engine](#)< _RandomNumberEngine, __k > &__lhs, const [std::shuffle_order_engine](#)< _RandomNumberEngine, __k > &__rhs)
- template<typename _IntType >
bool [std::operator!=](#) (const [std::uniform_int_distribution](#)< _IntType > &__d1, const [std::uniform_int_distribution](#)< _IntType > &__d2)
- template<typename _IntType >
bool [std::operator!=](#) (const [std::uniform_real_distribution](#)< _IntType > &__d1, const [std::uniform_real_distribution](#)< _IntType > &__d2)

- `template<typename _RealType >`
`bool std::operator!= (const std::normal_distribution< _RealType > &__d1, const std::normal_distribution< _RealType > &__d2)`
- `template<typename _RealType >`
`bool std::operator!= (const std::lognormal_distribution< _RealType > &__d1, const std::lognormal_distribution< _RealType > &__d2)`
- `template<typename _RealType >`
`bool std::operator!= (const std::gamma_distribution< _RealType > &__d1, const std::gamma_distribution< _RealType > &__d2)`
- `template<typename _RealType >`
`bool std::operator!= (const std::chi_squared_distribution< _RealType > &__d1, const std::chi_squared_distribution< _RealType > &__d2)`
- `template<typename _RealType >`
`bool std::operator!= (const std::cauchy_distribution< _RealType > &__d1, const std::cauchy_distribution< _RealType > &__d2)`
- `template<typename _RealType >`
`bool std::operator!= (const std::fisher_f_distribution< _RealType > &__d1, const std::fisher_f_distribution< _RealType > &__d2)`
- `template<typename _RealType >`
`bool std::operator!= (const std::student_t_distribution< _RealType > &__d1, const std::student_t_distribution< _RealType > &__d2)`
- `bool std::operator!= (const std::bernoulli_distribution &__d1, const std::bernoulli_distribution &__d2)`
- `template<typename _IntType >`
`bool std::operator!= (const std::binomial_distribution< _IntType > &__d1, const std::binomial_distribution< _IntType > &__d2)`
- `template<typename _IntType >`
`bool std::operator!= (const std::geometric_distribution< _IntType > &__d1, const std::geometric_distribution< _IntType > &__d2)`
- `template<typename _IntType >`
`bool std::operator!= (const std::negative_binomial_distribution< _IntType > &__d1, const std::negative_binomial_distribution< _IntType > &__d2)`
- `template<typename _IntType >`
`bool std::operator!= (const std::poisson_distribution< _IntType > &__d1, const std::poisson_distribution< _IntType > &__d2)`
- `template<typename _RealType >`
`bool std::operator!= (const std::exponential_distribution< _RealType > &__d1, const std::exponential_distribution< _RealType > &__d2)`
- `template<typename _RealType >`
`bool std::operator!= (const std::weibull_distribution< _RealType > &__d1, const std::weibull_distribution< _RealType > &__d2)`
- `template<typename _RealType >`
`bool std::operator!= (const std::extreme_value_distribution< _RealType > &__d1, const std::extreme_value_distribution< _RealType > &__d2)`
- `template<typename _IntType >`
`bool std::operator!= (const std::discrete_distribution< _IntType > &__d1, const std::discrete_distribution< _IntType > &__d2)`
- `template<typename _RealType >`
`bool std::operator!= (const std::piecewise_constant_distribution< _RealType > &__d1, const std::piecewise_constant_distribution< _RealType > &__d2)`
- `template<typename _RealType >`
`bool std::operator!= (const std::piecewise_linear_distribution< _RealType > &__d1, const std::piecewise_linear_distribution< _RealType > &__d2)`

- `template<typename _RandomNumberEngine, size_t __w, typename _UIntType, typename _CharT, typename _Traits >
std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const
std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >
std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &, const std::←
::uniform_int_distribution< _IntType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >
std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &, const std::←
::uniform_real_distribution< _RealType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >
std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const
std::cauchy_distribution< _RealType > &__x)`
- `template<typename _CharT, typename _Traits >
std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const
std::bernoulli_distribution &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >
std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const
std::geometric_distribution< _IntType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >
std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const
std::exponential_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >
std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const
std::weibull_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >
std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const
std::extreme_value_distribution< _RealType > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >
std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &, std::←
::uniform_int_distribution< _IntType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >
std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &, std::←
::uniform_real_distribution< _RealType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >
std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::←
::cauchy_distribution< _RealType > &__x)`
- `template<typename _CharT, typename _Traits >
std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::←
::bernoulli_distribution &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >
std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::←
::geometric_distribution< _IntType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >
std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::←
::exponential_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >
std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::←
::weibull_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >
std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::←
::extreme_value_distribution< _RealType > &__x)`

5.320.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<random>`.

5.321 random_number.h File Reference

Classes

- class [__gnu_parallel::_RandomNumber](#)

Namespaces

- [__gnu_parallel](#)

5.321.1 Detailed Description

Random number generator based on the Mersenne twister. This file is a GNU parallel extension to the Standard C++ Library.

5.322 random_shuffle.h File Reference

Classes

- struct [__gnu_parallel::_DRandomShufflingGlobalData<_RAIter>](#)
- struct [__gnu_parallel::_DRSSorterPU<_RAIter, _RandomNumberGenerator>](#)

Namespaces

- [__gnu_parallel](#)

Typedefs

- typedef unsigned short [__gnu_parallel::_BinIndex](#)

Functions

- `template<typename _RAIter, typename _RandomNumberGenerator >`
`void __gnu_parallel::__parallel_random_shuffle (_RAIter __begin, _RAIter __end, _RandomNumberGenerator &__rng)`
- `template<typename _RAIter, typename _RandomNumberGenerator >`
`void __gnu_parallel::__parallel_random_shuffle_drs (_RAIter __begin, _RAIter __end, typename std::iterator_traits<_RAIter>::difference_type __n, _ThreadIndex __num_threads, _RandomNumberGenerator &__rng)`
- `template<typename _RAIter, typename _RandomNumberGenerator >`
`void __gnu_parallel::__parallel_random_shuffle_drs_pu (_DRSSorterPU<_RAIter, _RandomNumberGenerator> *__pus)`
- `template<typename _RandomNumberGenerator >`
`int __gnu_parallel::__random_number_pow2 (int __logp, _RandomNumberGenerator &__rng)`
- `template<typename _Tp >`
`_Tp __gnu_parallel::__round_up_to_pow2 (_Tp __x)`
- `template<typename _RAIter, typename _RandomNumberGenerator >`
`void __gnu_parallel::__sequential_random_shuffle (_RAIter __begin, _RAIter __end, _RandomNumberGenerator &__rng)`

5.322.1 Detailed Description

Parallel implementation of `std::random_shuffle()`. This file is a GNU parallel extension to the Standard C++ Library.

5.323 range_access.h File Reference

Namespaces

- `std`

Functions

- `template<class _Container >`
`auto std::begin (_Container &__cont) -> decltype(__cont.begin())`
- `template<class _Container >`
`auto std::begin (const _Container &__cont) -> decltype(__cont.begin())`
- `template<class _Tp, size_t _Nm >`
`_Tp * std::begin (_Tp(&__arr)[_Nm])`
- `template<class _Container >`
`auto std::end (_Container &__cont) -> decltype(__cont.end())`
- `template<class _Container >`
`auto std::end (const _Container &__cont) -> decltype(__cont.end())`
- `template<class _Tp, size_t _Nm >`
`_Tp * std::end (_Tp(&__arr)[_Nm])`

5.323.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iterator>`.

5.324 ranged_hash_fn.hpp File Reference

Classes

- class [__gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, Store_Hash >](#)
- class [__gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, false >](#)
- class [__gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, true >](#)
- class [__gnu_pbds::detail::ranged_hash_fn< Key, null_type, _Alloc, Comb_Hash_Fn, false >](#)
- class [__gnu_pbds::detail::ranged_hash_fn< Key, null_type, _Alloc, Comb_Hash_Fn, true >](#)

Namespaces

- [__gnu_pbds](#)

Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`

5.324.1 Detailed Description

Contains a unified ranged hash functor, allowing the hash tables to deal with a single class for ranged hashing.

5.325 ranged_probe_fn.hpp File Reference

Classes

- class [__gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, Store_Hash >](#)
- class [__gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, false >](#)
- class [__gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, true >](#)
- class [__gnu_pbds::detail::ranged_probe_fn< Key, null_type, _Alloc, Comb_Probe_Fn, null_type, false >](#)

Namespaces

- [__gnu_pbds](#)

Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`

5.325.1 Detailed Description

Contains a unified ranged probe functor, allowing the probe tables to deal with a single class for ranged probeing.

5.326 `rb_tree.hpp` File Reference

Classes

- class [__gnu_pbds::detail::rb_tree_map](#)< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >

Namespaces

- [__gnu_pbds](#)

Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_RB_TREE_BASE`
- `#define PB_DS_RB_TREE_BASE_NAME`
- `#define PB_DS_RB_TREE_NAME`
- `#define PB_DS_STRUCT_ONLY_ASSERT_VALID(X)`

5.326.1 Detailed Description

Contains an implementation for Red Black trees.

5.327 `rc.hpp` File Reference

Classes

- class [__gnu_pbds::detail::rc](#)< _Node, _Alloc >

Namespaces

- [__gnu_pbds](#)

5.327.1 Detailed Description

Contains a redundant (binary counter).

5.328 rc_binomial_heap_.hpp File Reference

Classes

- class [__gnu_pbds::detail::rc_binomial_heap< Value_Type, Cmp_Fn, _Alloc >](#)

Namespaces

- [__gnu_pbds](#)

Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_RC_C_DEC`

5.328.1 Detailed Description

Contains an implementation for redundant-counter binomial heap.

5.329 rc_string_base.h File Reference

Classes

- class [__gnu_cxx::__rc_string_base< _CharT, _Traits, _Alloc >](#)

Namespaces

- [__gnu_cxx](#)

5.329.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ext/vstring.h>`.

5.330 regex.h File Reference

Classes

- class [std::__detail::_Executor<_Bilter, _Alloc, _TraitsT, __dfs_mode>](#)
- class [std::basic_regex<_Ch_type, _Rx_traits>](#)
- class [std::basic_regex<_Ch_type, _Rx_traits>](#)
- class [std::match_results<_Bi_iter, _Alloc>](#)
- class [std::match_results<_Bi_iter, _Alloc>](#)
- class [std::regex_iterator<_Bi_iter, _Ch_type, _Rx_traits>](#)
- class [std::regex_token_iterator<_Bi_iter, _Ch_type, _Rx_traits>](#)
- struct [std::regex_traits<_Ch_type>](#)
- class [std::sub_match<_Bilter>](#)

Namespaces

- [std](#)
- [std::__detail](#)

Typedefs

- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc>
using std::__sub_match_string = basic_string< typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc >`
- `typedef match_results< const char * > std::cmatch`
- `typedef regex_iterator< const char * > std::cregex_iterator`
- `typedef regex_token_iterator< const char * > std::cregex_token_iterator`
- `typedef sub_match< const char * > std::csub_match`
- `typedef basic_regex< char > std::regex`
- `typedef match_results< string::const_iterator > std::smatch`
- `typedef regex_iterator< string::const_iterator > std::sregex_iterator`
- `typedef regex_token_iterator< string::const_iterator > std::sregex_token_iterator`
- `typedef sub_match< string::const_iterator > std::ssub_match`
- `typedef match_results< const wchar_t * > std::wcmatch`
- `typedef regex_iterator< const wchar_t * > std::wcregex_iterator`
- `typedef regex_token_iterator< const wchar_t * > std::wcregex_token_iterator`
- `typedef sub_match< const wchar_t * > std::wcs_sub_match`
- `typedef basic_regex< wchar_t > std::wregex`
- `typedef match_results< wstring::const_iterator > std::wsmatch`
- `typedef regex_iterator< wstring::const_iterator > std::wsregex_iterator`
- `typedef regex_token_iterator< wstring::const_iterator > std::wsregex_token_iterator`
- `typedef sub_match< wstring::const_iterator > std::wssub_match`

Enumerations

- enum **_RegexExecutorPolicy** : int { **_S_auto**, **_S_alterate** }

Functions

- `template<typename _TraitsT >`
`std::shared_ptr< _NFA< _TraitsT > > std::__detail::__compile_nfa` (const typename _TraitsT::char_type * __
`_first, const typename _TraitsT::char_type * __last, const _TraitsT & __traits, regex_constants::syntax_option_type`
`__flags)`
- `template<typename _Bilter, typename _Alloc, typename _CharT, typename _TraitsT, _RegexExecutorPolicy __policy, bool __match_`
`mode>`
`bool std::__detail::__regex_algo_impl` (_Bilter __s, _Bilter __e, match_results< _Bilter, _Alloc > & __m, const
`basic_regex< _CharT, _TraitsT > & __re, regex_constants::match_flag_type __flags)`
- `template<typename _Bi_iter >`
`const sub_match< _Bi_iter > & std::__unmatched_sub` ()
- `template<typename _Bilter >`
`bool std::operator!=` (const sub_match< _Bilter > & __lhs, const sub_match< _Bilter > & __rhs)
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool std::operator!=` (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > & __lhs, const sub_match<
`_Bi_iter > & __rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool std::operator!=` (const sub_match< _Bi_iter > & __lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _
`Ch_alloc > & __rhs)`
- `template<typename _Bi_iter >`
`bool std::operator!=` (typename iterator_traits< _Bi_iter >::value_type const * __lhs, const sub_match< _Bi_iter
`> & __rhs)`
- `template<typename _Bi_iter >`
`bool std::operator!=` (const sub_match< _Bi_iter > & __lhs, typename iterator_traits< _Bi_iter >::value_type
`const * __rhs)`
- `template<typename _Bi_iter >`
`bool std::operator!=` (typename iterator_traits< _Bi_iter >::value_type const & __lhs, const sub_match< _Bi_iter
`> & __rhs)`
- `template<typename _Bi_iter >`
`bool std::operator!=` (const sub_match< _Bi_iter > & __lhs, typename iterator_traits< _Bi_iter >::value_type
`const & __rhs)`
- `template<typename _Bi_iter, class _Alloc >`
`bool std::operator!=` (const match_results< _Bi_iter, _Alloc > & __m1, const match_results< _Bi_iter, _Alloc >
`& __m2)`
- `template<typename _Bilter >`
`bool std::operator<` (const sub_match< _Bilter > & __lhs, const sub_match< _Bilter > & __rhs)
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool std::operator<` (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > & __lhs, const sub_match<
`_Bi_iter > & __rhs)`
- `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc >`
`bool std::operator<` (const sub_match< _Bi_iter > & __lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _
`Ch_alloc > & __rhs)`
- `template<typename _Bi_iter >`
`bool std::operator<` (typename iterator_traits< _Bi_iter >::value_type const * __lhs, const sub_match< _Bi_iter
`> & __rhs)`
- `template<typename _Bi_iter >`
`bool std::operator<` (const sub_match< _Bi_iter > & __lhs, typename iterator_traits< _Bi_iter >::value_type
`const * __rhs)`
- `template<typename _Bi_iter >`
`bool std::operator<` (typename iterator_traits< _Bi_iter >::value_type const & __lhs, const sub_match< _Bi_iter
`> & __rhs)`

- `template<typename _Bi_iter >`
`bool std::operator< (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type`
`const &__rhs)`
- `template<typename _Ch_type, typename _Ch_traits, typename _Bi_iter >`
`basic_ostream< _Ch_type, _Ch_traits > & std::operator<< (basic_ostream< _Ch_type, _Ch_traits > &__os,`
`const sub_match< _Bi_iter > &__m)`
- `template<typename _Bilter >`
`bool std::operator<= (const sub_match< _Bilter > &__lhs, const sub_match< _Bilter > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool std::operator<= (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__lhs, const sub_match<`
`_Bi_iter > &__rhs)`
- `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc >`
`bool std::operator<= (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits,`
`_Ch_alloc > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator<= (typename iterator_traits< _Bi_iter >::value_type const *__lhs, const sub_match< _Bi_iter`
`> &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator<= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type`
`const *__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator<= (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter`
`> &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator<= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type`
`const &__rhs)`
- `template<typename _Bilter >`
`bool std::operator== (const sub_match< _Bilter > &__lhs, const sub_match< _Bilter > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool std::operator== (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__lhs, const sub_match<`
`_Bi_iter > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool std::operator== (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits,`
`_Ch_alloc > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator== (typename iterator_traits< _Bi_iter >::value_type const *__lhs, const sub_match< _Bi_iter`
`> &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator== (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type`
`const *__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator== (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter`
`> &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator== (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type`
`const &__rhs)`
- `template<typename _Bi_iter, typename _Alloc >`
`bool std::operator== (const match_results< _Bi_iter, _Alloc > &__m1, const match_results< _Bi_iter, _Alloc >`
`&__m2)`
- `template<typename _Bilter >`
`bool std::operator> (const sub_match< _Bilter > &__lhs, const sub_match< _Bilter > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool std::operator> (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__lhs, const sub_match<`
`_Bi_iter > &__rhs)`

- `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc >`
`bool std::operator> (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator> (typename iterator_traits< _Bi_iter >::value_type const *__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator> (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const *__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator> (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator> (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Bilter >`
`bool std::operator>= (const sub_match< _Bilter > &__lhs, const sub_match< _Bilter > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool std::operator>= (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter, class _Ch_traits, class _Ch_alloc >`
`bool std::operator>= (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator>= (typename iterator_traits< _Bi_iter >::value_type const *__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator>= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const *__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator>= (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`
`bool std::operator>= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Ch_type, typename _Rx_traits >`
`void std::swap (basic_regex< _Ch_type, _Rx_traits > &__lhs, basic_regex< _Ch_type, _Rx_traits > &__rhs)`
- `template<typename _Bi_iter, typename _Alloc >`
`void std::swap (match_results< _Bi_iter, _Alloc > &__lhs, match_results< _Bi_iter, _Alloc > &__rhs)`

Matching, Searching, and Replacing

- `template<typename _Bi_iter, typename _Alloc, typename _Ch_type, typename _Rx_traits >`
`bool std::regex_match (_Bi_iter __s, _Bi_iter __e, match_results< _Bi_iter, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits >`
`bool std::regex_match (_Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Ch_type, typename _Alloc, typename _Rx_traits >`
`bool std::regex_match (const _Ch_type *__s, match_results< const _Ch_type *, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __f=regex_constants::match_default)`

- `template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits >`
`bool std::regex_match (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > &__s, match_results< type-`
`name basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > &__m, const basic_regex<`
`_Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Ch_type, class _Rx_traits >`
`bool std::regex_match (const _Ch_type *__s, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_↵`
`constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _Str_allocator, typename _Ch_type, typename _Rx_traits >`
`bool std::regex_match (const basic_string< _Ch_type, _Ch_traits, _Str_allocator > &__s, const basic_regex<`
`_Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Bi_iter, typename _Alloc, typename _Ch_type, typename _Rx_traits >`
`bool std::regex_search (_Bi_iter __s, _Bi_iter __e, match_results< _Bi_iter, _Alloc > &__m, const basic_↵`
`regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_↵`
`_default)`
- `template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits >`
`bool std::regex_search (_Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits > &__re,`
`regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Ch_type, class _Alloc, class _Rx_traits >`
`bool std::regex_search (const _Ch_type *__s, match_results< const _Ch_type *, _Alloc > &__m, const`
`basic_regex< _Ch_type, _Rx_traits > &__e, regex_constants::match_flag_type __f=regex_constants_↵`
`::match_default)`
- `template<typename _Ch_type, typename _Rx_traits >`
`bool std::regex_search (const _Ch_type *__s, const basic_regex< _Ch_type, _Rx_traits > &__e, regex_↵`
`constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _String_allocator, typename _Ch_type, typename _Rx_traits >`
`bool std::regex_search (const basic_string< _Ch_type, _Ch_traits, _String_allocator > &__s, const basic_↵`
`regex< _Ch_type, _Rx_traits > &__e, regex_constants::match_flag_type __flags=regex_constants::match_↵`
`_default)`
- `template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits >`
`bool std::regex_search (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > &__s, match_results< type-`
`name basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > &__m, const basic_regex<`
`_Ch_type, _Rx_traits > &__e, regex_constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Out_iter, typename _Bi_iter, typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa >`
`_Out_iter std::regex_replace (_Out_iter __out, _Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type,`
`_Rx_traits > &__e, const basic_string< _Ch_type, _St, _Sa > &__fmt, regex_constants::match_flag_type`
`__flags=regex_constants::match_default)`
- `template<typename _Out_iter, typename _Bi_iter, typename _Rx_traits, typename _Ch_type >`
`_Out_iter std::regex_replace (_Out_iter __out, _Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type,`
`_Rx_traits > &__e, const _Ch_type *__fmt, regex_constants::match_flag_type __flags=regex_constants_↵`
`::match_default)`
- `template<typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa, typename _Fst, typename _Fsa >`
`basic_string< _Ch_type, _St, _Sa > std::regex_replace (const basic_string< _Ch_type, _St, _Sa > &__↵`
`s, const basic_regex< _Ch_type, _Rx_traits > &__e, const basic_string< _Ch_type, _Fst, _Fsa > &__fmt,`
`regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa >`
`basic_string< _Ch_type, _St, _Sa > std::regex_replace (const basic_string< _Ch_type, _St, _Sa > &__s,`
`const basic_regex< _Ch_type, _Rx_traits > &__e, const _Ch_type *__fmt, regex_constants::match_flag_↵`
`type __flags=regex_constants::match_default)`
- `template<typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa >`
`basic_string< _Ch_type > std::regex_replace (const _Ch_type *__s, const basic_regex< _Ch_type, _Rx_↵`
`_traits > &__e, const basic_string< _Ch_type, _St, _Sa > &__fmt, regex_constants::match_flag_type __↵`
`flags=regex_constants::match_default)`
- `template<typename _Rx_traits, typename _Ch_type >`
`basic_string< _Ch_type > std::regex_replace (const _Ch_type *__s, const basic_regex< _Ch_type, _Rx_↵`
`traits > &__e, const _Ch_type *__fmt, regex_constants::match_flag_type __flags=regex_constants::match_↵`
`_default)`

5.330.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<regex>`.

5.331 `regex_automaton.h` File Reference

Classes

- class `std::__detail::_StateSeq< _TraitsT >`

Namespaces

- `std`
- `std::__detail`

Typedefs

- `template<typename _CharT >`
using `std::__detail::_Matcher` = `std::function< bool(_CharT)>`
- `typedef long std::__detail::_StatelD`

Enumerations

- `enum std::__detail::_Opcode : int {`
`_S_opcode_unknown, _S_opcode_alternative, _S_opcode_backref, _S_opcode_line_begin_assertion,`
`_S_opcode_line_end_assertion, _S_opcode_word_boundary, _S_opcode_subexpr_lookahead, _S_`
`opcode_subexpr_begin,`
`_S_opcode_subexpr_end, _S_opcode_dummy, _S_opcode_match, _S_opcode_accept }`

Variables

- `static const _StatelD std::__detail::_S_invalid_state_id`

5.331.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<regex>`.

5.332 `regex_compiler.h` File Reference

Classes

- `struct std::__detail::_BracketMatcher< _TraitsT, __icase, __collate >`
- `struct std::__detail::_BracketMatcher< _TraitsT, __icase, __collate >`
- `class std::__detail::_Compiler< _TraitsT >`

Namespaces

- [std](#)
- [std::__detail](#)

Functions

- `template<typename _TraitsT >
std::shared_ptr<_NFA<_TraitsT > > std::__detail::__compile_nfa (const typename _TraitsT::char_type *__↵
_first, const typename _TraitsT::char_type *__last, const _TraitsT &__traits, regex_constants::syntax_option_type
__flags)`

5.332.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<regex>`.

5.333 `regex_constants.h` File Reference

Namespaces

- [std](#)
- [std::regex_constants](#)

5.1 Regular Expression Syntax Options

- `enum std::regex_constants::__syntax_option {
_S_icode, _S_nosubs, _S_optimize, _S_collate,
_S_ECMAScript, _S_basic, _S_extended, _S_awk,
_S_grep, _S_egrep, _S_syntax_last }`
- `enum std::regex_constants::syntax_option_type : unsigned int {
std::regex_constants::icode, std::regex_constants::nosubs, std::regex_constants::optimize, std::regex_↵
constants::collate,
std::regex_constants::ECMAScript, std::regex_constants::basic, std::regex_constants::extended, std::regex_↵
constants::awk,
std::regex_constants::grep, std::regex_constants::egrep }`
- `constexpr syntax_option_type std::regex_constants::operator& (syntax_option_type __a, syntax_option_type __↵
__b)`
- `constexpr syntax_option_type std::regex_constants::operator| (syntax_option_type __a, syntax_option_type __b)`
- `constexpr syntax_option_type std::regex_constants::operator^ (syntax_option_type __a, syntax_option_type __↵
__b)`
- `constexpr syntax_option_type std::regex_constants::operator~ (syntax_option_type __a)`
- `syntax_option_type & std::regex_constants::operator&= (syntax_option_type &__a, syntax_option_type __b)`
- `syntax_option_type & std::regex_constants::operator|= (syntax_option_type &__a, syntax_option_type __b)`
- `syntax_option_type & std::regex_constants::operator^= (syntax_option_type &__a, syntax_option_type __b)`

5.2 Matching Rules

Matching a regular expression against a sequence of characters [first, last) proceeds according to the rules of the grammar specified for the regular expression object, modified according to the effects listed below for any bitmask elements set.

- enum `std::regex_constants::__match_flag` {
`_S_not_bol`, `_S_not_eol`, `_S_not_bow`, `_S_not_eow`,
`_S_any`, `_S_not_null`, `_S_continuous`, `_S_prev_avail`,
`_S_sed`, `_S_no_copy`, `_S_first_only`, `_S_match_flag_last` }
- enum `std::regex_constants::match_flag_type` : unsigned int {
`std::regex_constants::match_default`, `std::regex_constants::match_not_bol`, `std::regex_constants::match_not_↵
eol`, `std::regex_constants::match_not_bow`,
`std::regex_constants::match_not_eow`, `std::regex_constants::match_any`, `std::regex_constants::match_not_null`,
`std::regex_constants::match_continuous`,
`std::regex_constants::match_prev_avail`, `std::regex_constants::format_default`, `std::regex_constants::format_sed`,
`std::regex_constants::format_no_copy`,
`std::regex_constants::format_first_only` }
- constexpr `match_flag_type` `std::regex_constants::operator&` (`match_flag_type __a`, `match_flag_type __b`)
- constexpr `match_flag_type` `std::regex_constants::operator|` (`match_flag_type __a`, `match_flag_type __b`)
- constexpr `match_flag_type` `std::regex_constants::operator^` (`match_flag_type __a`, `match_flag_type __b`)
- constexpr `match_flag_type` `std::regex_constants::operator~` (`match_flag_type __a`)
- `match_flag_type &` `std::regex_constants::operator&=` (`match_flag_type &__a`, `match_flag_type __b`)
- `match_flag_type &` `std::regex_constants::operator|=` (`match_flag_type &__a`, `match_flag_type __b`)
- `match_flag_type &` `std::regex_constants::operator^=` (`match_flag_type &__a`, `match_flag_type __b`)

5.333.1 Detailed Description

Constant definitions for the std regex library.

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<regex>`.

5.334 `regex_error.h` File Reference

Classes

- class `std::regex_error`

Namespaces

- `std`
- `std::regex_constants`

Functions

- void `std::__throw_regex_error` (`regex_constants::error_type __ecode`)

5.3 Error Types

- enum [std::regex_constants::error_type](#) {
[_S_error_collate](#), [_S_error_ctype](#), [_S_error_escape](#), [_S_error_backref](#),
[_S_error_brack](#), [_S_error_paren](#), [_S_error_brace](#), [_S_error_badbrace](#),
[_S_error_range](#), [_S_error_space](#), [_S_error_badrepeat](#), [_S_error_complexity](#),
[_S_error_stack](#) }
- constexpr error_type [std::regex_constants::error_collate](#) ([_S_error_collate](#))
- constexpr error_type [std::regex_constants::error_ctype](#) ([_S_error_ctype](#))
- constexpr error_type [std::regex_constants::error_escape](#) ([_S_error_escape](#))
- constexpr error_type [std::regex_constants::error_backref](#) ([_S_error_backref](#))
- constexpr error_type [std::regex_constants::error_brack](#) ([_S_error_brack](#))
- constexpr error_type [std::regex_constants::error_paren](#) ([_S_error_paren](#))
- constexpr error_type [std::regex_constants::error_brace](#) ([_S_error_brace](#))
- constexpr error_type [std::regex_constants::error_badbrace](#) ([_S_error_badbrace](#))
- constexpr error_type [std::regex_constants::error_range](#) ([_S_error_range](#))
- constexpr error_type [std::regex_constants::error_space](#) ([_S_error_space](#))
- constexpr error_type [std::regex_constants::error_badrepeat](#) ([_S_error_badrepeat](#))
- constexpr error_type [std::regex_constants::error_complexity](#) ([_S_error_complexity](#))
- constexpr error_type [std::regex_constants::error_stack](#) ([_S_error_stack](#))

5.334.1 Detailed Description

Error and exception objects for the std regex library.

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<regex>`.

5.335 `regex_executor.h` File Reference

Classes

- class [std::__detail::_Executor<_Bilter, _Alloc, _TraitsT, __dfs_mode>](#)

Namespaces

- [std](#)
- [std::__detail](#)

5.335.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<regex>`.

5.336 `regex_scanner.h` File Reference

Classes

- class `std::__detail::_Scanner<_CharT>`

Namespaces

- `std`
- `std::__detail`

5.336.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<regex>`.

5.337 `resize_fn_imps.hpp` File Reference

5.337.1 Detailed Description

Contains implementations of `cc_ht_map_'s` resize related functions.

5.338 `resize_fn_imps.hpp` File Reference

5.338.1 Detailed Description

Contains implementations of `gp_ht_map_'s` resize related functions.

5.339 `resize_no_store_hash_fn_imps.hpp` File Reference

5.339.1 Detailed Description

Contains implementations of `cc_ht_map_'s` resize related functions, when the hash value is not stored.

5.340 `resize_no_store_hash_fn_imps.hpp` File Reference

5.340.1 Detailed Description

Contains implementations of `gp_ht_map_'s` resize related functions, when the hash value is not stored.

5.341 `resize_policy.hpp` File Reference

Classes

- class [__gnu_pbds::detail::resize_policy< _Tp >](#)

Namespaces

- [__gnu_pbds](#)

5.341.1 Detailed Description

Contains an implementation class for a `binary_heap`.

5.342 `resize_store_hash_fn_imps.hpp` File Reference

5.342.1 Detailed Description

Contains implementations of `cc_ht_map_'s` `resize` related functions, when the hash value is stored.

5.343 `resize_store_hash_fn_imps.hpp` File Reference

5.343.1 Detailed Description

Contains implementations of `gp_ht_map_'s` `resize` related functions, when the hash value is stored.

5.344 `ropeimpl.h` File Reference

Namespaces

- [__gnu_cxx](#)

Functions

- `template<class _CharT, class _Traits >`
`void __gnu_cxx::Rope_fill (basic_ostream< _CharT, _Traits > &__o, size_t __n)`
- `template<class _CharT >`
`bool __gnu_cxx::Rope_is_simple (_CharT *)`
- `bool __gnu_cxx::Rope_is_simple (char *)`
- `bool __gnu_cxx::Rope_is_simple (wchar_t *)`
- `template<class _Rope_iterator >`
`void __gnu_cxx::Rope_rotate (_Rope_iterator __first, _Rope_iterator __middle, _Rope_iterator __last)`
- `template<class _CharT, class _Traits, class _Alloc >`
`basic_ostream< _CharT, _Traits > & __gnu_cxx::operator<< (basic_ostream< _CharT, _Traits > &__o, const rope< _CharT, _Alloc > &__r)`
- `void __gnu_cxx::rotate (_Rope_iterator< char, __STL_DEFAULT_ALLOCATOR(char)> __first, _Rope_iterator< char, __STL_DEFAULT_ALLOCATOR(char)> __middle, _Rope_iterator< char, __STL_DEFAULT_ALLOCATOR(char)> __last)`

5.344.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ext/rope>`.

5.345 rotate_fn_imps.hpp File Reference

5.345.1 Detailed Description

Contains imps for rotating nodes.

5.346 rotate_fn_imps.hpp File Reference

5.346.1 Detailed Description

Contains imps for rotating nodes.

5.347 safe_base.h File Reference

Classes

- class [__gnu_debug::_Safe_iterator_base](#)
- class [__gnu_debug::_Safe_sequence_base](#)

Namespaces

- [__gnu_debug](#)

5.347.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

5.348 safe_iterator.h File Reference

Classes

- struct [__gnu_debug::_BeforeBeginHelper<_Sequence>](#)
- class [__gnu_debug::_Safe_iterator<_Iterator, _Sequence>](#)

Namespaces

- [__gnu_debug](#)

Enumerations

- enum [__gnu_debug::Distance_precision](#) { [__dp_equality](#), [__dp_sign](#), [__dp_exact](#) }

Functions

- bool [__gnu_debug::check_singular_aux](#) (const _Safe_iterator_base * __x)
- template<typename _Iterator >
[std::pair](#)< typename std::iterator_traits< _Iterator >::difference_type, _Distance_precision > [__gnu_debug::__get_distance](#) (const _Iterator & __lhs, const _Iterator & __rhs, [std::random_access_iterator_tag](#))
- template<typename _Iterator >
[std::pair](#)< typename std::iterator_traits< _Iterator >::difference_type, _Distance_precision > [__gnu_debug::__get_distance](#) (const _Iterator & __lhs, const _Iterator & __rhs, [std::forward_iterator_tag](#))
- template<typename _Iterator >
[std::pair](#)< typename std::iterator_traits< _Iterator >::difference_type, _Distance_precision > [__gnu_debug::__get_distance](#) (const _Iterator & __lhs, const _Iterator & __rhs)
- template<typename _IteratorL, typename _IteratorR, typename _Sequence >
bool [__gnu_debug::operator!=](#) (const _Safe_iterator< _IteratorL, _Sequence > & __lhs, const _Safe_iterator< _IteratorR, _Sequence > & __rhs) noexcept
- template<typename _Iterator, typename _Sequence >
bool [__gnu_debug::operator!=](#) (const _Safe_iterator< _Iterator, _Sequence > & __lhs, const _Safe_iterator< _Iterator, _Sequence > & __rhs) noexcept
- template<typename _Iterator, typename _Sequence >
_Safe_iterator< _Iterator, _Sequence > [__gnu_debug::operator+](#) (typename _Safe_iterator< _Iterator, _Sequence >::difference_type __n, const _Safe_iterator< _Iterator, _Sequence > & __i) noexcept
- template<typename _IteratorL, typename _IteratorR, typename _Sequence >
_Safe_iterator< _IteratorL, _Sequence >::difference_type [__gnu_debug::operator-](#) (const _Safe_iterator< _IteratorL, _Sequence > & __lhs, const _Safe_iterator< _IteratorR, _Sequence > & __rhs) noexcept
- template<typename _Iterator, typename _Sequence >
_Safe_iterator< _Iterator, _Sequence >::difference_type [__gnu_debug::operator-](#) (const _Safe_iterator< _Iterator, _Sequence > & __lhs, const _Safe_iterator< _Iterator, _Sequence > & __rhs) noexcept
- template<typename _IteratorL, typename _IteratorR, typename _Sequence >
bool [__gnu_debug::operator<](#) (const _Safe_iterator< _IteratorL, _Sequence > & __lhs, const _Safe_iterator< _IteratorR, _Sequence > & __rhs) noexcept
- template<typename _Iterator, typename _Sequence >
bool [__gnu_debug::operator<](#) (const _Safe_iterator< _Iterator, _Sequence > & __lhs, const _Safe_iterator< _Iterator, _Sequence > & __rhs) noexcept
- template<typename _IteratorL, typename _IteratorR, typename _Sequence >
bool [__gnu_debug::operator<=](#) (const _Safe_iterator< _IteratorL, _Sequence > & __lhs, const _Safe_iterator< _IteratorR, _Sequence > & __rhs) noexcept
- template<typename _Iterator, typename _Sequence >
bool [__gnu_debug::operator<=](#) (const _Safe_iterator< _Iterator, _Sequence > & __lhs, const _Safe_iterator< _Iterator, _Sequence > & __rhs) noexcept
- template<typename _IteratorL, typename _IteratorR, typename _Sequence >
bool [__gnu_debug::operator==](#) (const _Safe_iterator< _IteratorL, _Sequence > & __lhs, const _Safe_iterator< _IteratorR, _Sequence > & __rhs) noexcept

- `template<typename _Iterator, typename _Sequence >`
`bool __gnu_debug::operator== (const _Safe_iterator< _Iterator, _Sequence > &__lhs, const _Safe_iterator< _Iterator, _Sequence > &__rhs) noexcept`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`
`bool __gnu_debug::operator> (const _Safe_iterator< _IteratorL, _Sequence > &__lhs, const _Safe_iterator< _IteratorR, _Sequence > &__rhs) noexcept`
- `template<typename _Iterator, typename _Sequence >`
`bool __gnu_debug::operator> (const _Safe_iterator< _Iterator, _Sequence > &__lhs, const _Safe_iterator< _Iterator, _Sequence > &__rhs) noexcept`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`
`bool __gnu_debug::operator>= (const _Safe_iterator< _IteratorL, _Sequence > &__lhs, const _Safe_iterator< _IteratorR, _Sequence > &__rhs) noexcept`
- `template<typename _Iterator, typename _Sequence >`
`bool __gnu_debug::operator>= (const _Safe_iterator< _Iterator, _Sequence > &__lhs, const _Safe_iterator< _Iterator, _Sequence > &__rhs) noexcept`

5.348.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

5.349 `safe_local_iterator.h` File Reference

Classes

- class [`__gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >`](#)

Namespaces

- [`__gnu_debug`](#)

Functions

- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`
`bool __gnu_debug::operator!= (const _Safe_local_iterator< _IteratorL, _Sequence > &__lhs, const _Safe_local_iterator< _IteratorR, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`
`bool __gnu_debug::operator!= (const _Safe_local_iterator< _Iterator, _Sequence > &__lhs, const _Safe_local_iterator< _Iterator, _Sequence > &__rhs)`
- `template<typename _IteratorL, typename _IteratorR, typename _Sequence >`
`bool __gnu_debug::operator== (const _Safe_local_iterator< _IteratorL, _Sequence > &__lhs, const _Safe_local_iterator< _IteratorR, _Sequence > &__rhs)`
- `template<typename _Iterator, typename _Sequence >`
`bool __gnu_debug::operator== (const _Safe_local_iterator< _Iterator, _Sequence > &__lhs, const _Safe_local_iterator< _Iterator, _Sequence > &__rhs)`

5.349.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

5.350 `safe_sequence.h` File Reference

Classes

- class [__gnu_debug::_After_nth_from<_Iterator>](#)
- class [__gnu_debug::_Equal_to<_Type>](#)
- class [__gnu_debug::_Not_equal_to<_Type>](#)
- class [__gnu_debug::_Safe_iterator<_Iterator, _Sequence>](#)
- class [__gnu_debug::_Safe_sequence<_Sequence>](#)

Namespaces

- [__gnu_debug](#)

5.350.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

5.351 `safe_unordered_base.h` File Reference

Classes

- class [__gnu_debug::_Safe_local_iterator_base](#)
- class [__gnu_debug::_Safe_unordered_container_base](#)

Namespaces

- [__gnu_debug](#)

5.351.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

5.352 `safe_unordered_container.h` File Reference

Classes

- class [__gnu_debug::_Safe_unordered_container<_Container>](#)

Namespaces

- [__gnu_debug](#)

5.352.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

5.353 sample_probe_fn.hpp File Reference

Classes

- class [__gnu_pbds::sample_probe_fn](#)

Namespaces

- [__gnu_pbds](#)

5.353.1 Detailed Description

Contains a sample probe policy.

5.354 sample_range_hashing.hpp File Reference

Classes

- class [__gnu_pbds::sample_range_hashing](#)

Namespaces

- [__gnu_pbds](#)

5.354.1 Detailed Description

Contains a range hashing policy.

5.355 sample_ranged_hash_fn.hpp File Reference

Classes

- class [__gnu_pbds::sample_ranged_hash_fn](#)

Namespaces

- [__gnu_pbds](#)

5.355.1 Detailed Description

Contains a ranged hash policy.

5.356 `sample_ranged_probe_fn.hpp` File Reference

Classes

- class [__gnu_pbds::sample_ranged_probe_fn](#)

Namespaces

- [__gnu_pbds](#)

5.356.1 Detailed Description

Contains a ranged probe policy.

5.357 `sample_resize_policy.hpp` File Reference

Classes

- class [__gnu_pbds::sample_resize_policy](#)

Namespaces

- [__gnu_pbds](#)

5.357.1 Detailed Description

Contains a sample resize policy for hash tables.

5.358 `sample_resize_trigger.hpp` File Reference

Classes

- class [__gnu_pbds::sample_resize_trigger](#)

Namespaces

- [__gnu_pbds](#)

5.358.1 Detailed Description

Contains a sample resize trigger policy class.

5.359 `sample_size_policy.hpp` File Reference

Classes

- class [__gnu_pbds::sample_size_policy](#)

Namespaces

- [__gnu_pbds](#)

5.359.1 Detailed Description

Contains a sample size resize-policy.

5.360 `sample_tree_node_update.hpp` File Reference

Classes

- class [__gnu_pbds::sample_tree_node_update< Const_Node_Iter, Node_Iter, Cmp_Fn, _Alloc >](#)

Namespaces

- [__gnu_pbds](#)

5.360.1 Detailed Description

Contains a samle node update functor.

5.361 `sample_trie_access_traits.hpp` File Reference

Classes

- struct [__gnu_pbds::sample_trie_access_traits](#)

Namespaces

- [__gnu_pbds](#)

5.361.1 Detailed Description

Contains a sample probe policy.

5.362 `sample_trie_node_update.hpp` File Reference

Classes

- class [__gnu_pbds::sample_trie_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >](#)

Namespaces

- [__gnu_pbds](#)

5.362.1 Detailed Description

Contains a samle node update functor.

5.363 `sample_update_policy.hpp` File Reference

Classes

- struct [__gnu_pbds::sample_update_policy](#)

Namespaces

- [__gnu_pbds](#)

5.363.1 Detailed Description

Contains a sample policy for list update containers.

5.364 `search.h` File Reference

Namespaces

- [__gnu_parallel](#)

Functions

- `template<typename _RAIter, typename _DifferenceTp >`
`void gnu_parallel::__calc_borders (_RAIter __elements, _DifferenceTp __length, _DifferenceTp *__off)`
- `template<typename __RAIter1, typename __RAIter2, typename _Pred >`
`gnu_parallel::__search_template (__RAIter1 __begin1, __RAIter1 __end1, __RAIter2 __begin2, ↵
__RAIter2 __end2, _Pred __pred)`

5.364.1 Detailed Description

Parallel implementation base for `std::search()` and `std::search_n()`. This file is a GNU parallel extension to the Standard C++ Library.

5.365 set.h File Reference

Classes

- `class std::__debug::set<_Key, _Compare, _Allocator>`

Namespaces

- `std`
- `std::__debug`

Functions

- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool std::__debug::operator!= (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool std::__debug::operator< (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool std::__debug::operator<= (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool std::__debug::operator== (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool std::__debug::operator> (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`bool std::__debug::operator>= (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`
`void std::__debug::swap (set< _Key, _Compare, _Allocator > &__x, set< _Key, _Compare, _Allocator > &__y)`

5.365.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

5.366 set.h File Reference

Classes

- class [std::__profile::set<_Key, _Compare, _Allocator>](#)

Namespaces

- [std](#)
- [std::__profile](#)

Functions

- `template<typename _Key, typename _Compare, typename _Allocator>
bool std::__profile::operator!= (const set< _Key, _Compare, _Allocator> &__lhs, const set< _Key, _Compare, _Allocator> &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator>
bool std::__profile::operator< (const set< _Key, _Compare, _Allocator> &__lhs, const set< _Key, _Compare, _Allocator> &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator>
bool std::__profile::operator<= (const set< _Key, _Compare, _Allocator> &__lhs, const set< _Key, _Compare, _Allocator> &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator>
bool std::__profile::operator== (const set< _Key, _Compare, _Allocator> &__lhs, const set< _Key, _Compare, _Allocator> &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator>
bool std::__profile::operator> (const set< _Key, _Compare, _Allocator> &__lhs, const set< _Key, _Compare, _Allocator> &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator>
bool std::__profile::operator>= (const set< _Key, _Compare, _Allocator> &__lhs, const set< _Key, _Compare, _Allocator> &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator>
void std::__profile::swap (set< _Key, _Compare, _Allocator> &__x, set< _Key, _Compare, _Allocator> &__y)`

5.366.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

5.367 set_operations.h File Reference

Namespaces

- [__gnu_parallel](#)

Functions

- `template<typename _Iter, typename _OutputIterator >`
`_OutputIterator gnu_parallel::copy_tail (std::pair< _Iter, _Iter > __b, std::pair< _Iter, _Iter > __e, _↵`
`_OutputIterator __r)`
- `template<typename _Iter, typename _OutputIterator, typename _Compare >`
`_OutputIterator gnu_parallel::parallel_set_difference (_Iter __begin1, _Iter __end1, _Iter __begin2, _↵`
`_Iter __end2, _OutputIterator __result, _Compare __comp)`
- `template<typename _Iter, typename _OutputIterator, typename _Compare >`
`_OutputIterator gnu_parallel::parallel_set_intersection (_Iter __begin1, _Iter __end1, _Iter __begin2,`
`_Iter __end2, _OutputIterator __result, _Compare __comp)`
- `template<typename _Iter, typename _OutputIterator, typename _Operation >`
`_OutputIterator gnu_parallel::parallel_set_operation (_Iter __begin1, _Iter __end1, _Iter __begin2, _↵`
`_Iter __end2, _OutputIterator __result, _Operation __op)`
- `template<typename _Iter, typename _OutputIterator, typename _Compare >`
`_OutputIterator gnu_parallel::parallel_set_symmetric_difference (_Iter __begin1, _Iter __end1, _Iter`
`__begin2, _Iter __end2, _OutputIterator __result, _Compare __comp)`
- `template<typename _Iter, typename _OutputIterator, typename _Compare >`
`_OutputIterator gnu_parallel::parallel_set_union (_Iter __begin1, _Iter __end1, _Iter __begin2, _Iter`
`__end2, _OutputIterator __result, _Compare __comp)`

5.367.1 Detailed Description

Parallel implementations of set operations for random-access iterators. This file is a GNU parallel extension to the Standard C++ Library.

5.368 settings.h File Reference

Classes

- `struct __gnu_parallel::Settings`

Namespaces

- `__gnu_parallel`

Macros

- `#define _GLIBCXX_PARALLEL_CONDITION(__c)`

5.368.1 Detailed Description

Runtime settings and tuning parameters, heuristics to decide whether to use parallelized algorithms. This file is a GNU parallel extension to the Standard C++ Library.

5.368.2 parallelization_decision

The decision whether to run an algorithm in parallel.

There are several ways the user can switch on and __off the parallel execution of an algorithm, both at compile- and run-time.

Only sequential execution can be forced at compile-time. This reduces code size and protects code parts that have non-thread-safe side effects.

Ultimately, forcing parallel execution at compile-time makes sense. Often, the sequential algorithm implementation is used as a subroutine, so no reduction in code size can be achieved. Also, the machine the program is run on might have only one processor core, so to avoid overhead, the algorithm is executed sequentially.

To force sequential execution of an algorithm ultimately at compile-time, the user must add the tag `gnu_parallel↵::sequential_tag()` to the end of the parameter list, e. g.

```
std::sort(__v.begin(), __v.end(), __gnu_parallel::sequential_tag());
```

This is compatible with all overloaded algorithm variants. No additional code will be instantiated, at all. The same holds for most algorithm calls with iterators not providing random access.

If the algorithm call is not forced to be executed sequentially at compile-time, the decision is made at run-time. The global variable `__gnu_parallel::_Settings::algorithm_strategy` is checked. `__`It is a tristate variable corresponding to:

a. `force_sequential`, meaning the sequential algorithm is executed. b. `force_parallel`, meaning the parallel algorithm is executed. c. `heuristic`

For heuristic, the parallel algorithm implementation is called only if the input size is sufficiently large. For most algorithms, the input size is the (combined) length of the input sequence(`__s`). The threshold can be set by the user, individually for each algorithm. The according variables are called `gnu_parallel::_Settings::[algorithm]_minimal_n`.

For some of the algorithms, there are even more tuning options, e. g. the ability to choose from multiple algorithm variants. See below for details.

5.368.3 Macro Definition Documentation

5.368.3.1 `#define GLIBCXX_PARALLEL_CONDITION(__c)`

Determine at compile(?) -time if the parallel variant of an algorithm should be called.

Parameters

<code>__↵</code>	A condition that is convertible to bool that is overruled by <code>__gnu_parallel::_Settings::algorithm_strategy</code> .
<code>__c</code>	Usually a decision based on the input size.

Definition at line 95 of file `settings.h`.

Referenced by `__gnu_parallel::multiway_merge()`, and `__gnu_parallel::multiway_merge_sentinels()`.

5.369 shared_ptr.h File Reference

Classes

- class [std::enable_shared_from_this<_Tp>](#)
- struct [std::hash<shared_ptr<_Tp>>](#)
- struct [std::owner_less<_Tp>](#)
- struct [std::owner_less<shared_ptr<_Tp>>](#)
- struct [std::owner_less<weak_ptr<_Tp>>](#)
- class [std::shared_ptr<_Tp>](#)
- class [std::weak_ptr<_Tp>](#)

Namespaces

- [std](#)

Functions

- [template<typename _Tp1, typename _Tp2> void std::__enable_shared_from_this_helper \(const __shared_count<> &__pn, const enable_shared_from_this<_Tp1> *__pe, const _Tp2 *__px\) noexcept](#)
- [template<typename _Tp, typename _Alloc, typename... _Args> shared_ptr<_Tp> std::allocate_shared \(const _Alloc &__a, _Args &&...__args\)](#)
- [template<typename _Tp, typename _Tp1> shared_ptr<_Tp> std::const_pointer_cast \(const shared_ptr<_Tp1> &__r\) noexcept](#)
- [template<typename _Tp, typename _Tp1> shared_ptr<_Tp> std::dynamic_pointer_cast \(const shared_ptr<_Tp1> &__r\) noexcept](#)
- [template<typename _Del, typename _Tp, _Lock_policy _Lp> _Del * std::get_deleter \(const __shared_ptr<_Tp, _Lp> &__p\) noexcept](#)
- [template<typename _Tp, typename... _Args> shared_ptr<_Tp> std::make_shared \(_Args &&...__args\)](#)
- [template<typename _Tp1, typename _Tp2> bool std::operator!= \(const shared_ptr<_Tp1> &__a, const shared_ptr<_Tp2> &__b\) noexcept](#)
- [template<typename _Tp> bool std::operator!= \(const shared_ptr<_Tp> &__a, nullptr_t\) noexcept](#)
- [template<typename _Tp> bool std::operator!= \(nullptr_t, const shared_ptr<_Tp> &__a\) noexcept](#)
- [template<typename _Tp1, typename _Tp2> bool std::operator< \(const shared_ptr<_Tp1> &__a, const shared_ptr<_Tp2> &__b\) noexcept](#)
- [template<typename _Tp> bool std::operator< \(const shared_ptr<_Tp> &__a, nullptr_t\) noexcept](#)
- [template<typename _Tp> bool std::operator< \(nullptr_t, const shared_ptr<_Tp> &__a\) noexcept](#)
- [template<typename _Ch, typename _Tr, typename _Tp, _Lock_policy _Lp> std::basic_ostream<_Ch, _Tr> & std::operator<< \(std::basic_ostream<_Ch, _Tr> &__os, const __shared_ptr<_Tp, _Lp> &__p\)](#)
- [template<typename _Tp1, typename _Tp2> bool std::operator<= \(const shared_ptr<_Tp1> &__a, const shared_ptr<_Tp2> &__b\) noexcept](#)
- [template<typename _Tp> bool std::operator<= \(const shared_ptr<_Tp> &__a, nullptr_t\) noexcept](#)

- `template<typename _Tp >`
`bool std::operator<= (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<typename _Tp1 , typename _Tp2 >`
`bool std::operator== (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept`
- `template<typename _Tp >`
`bool std::operator== (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp >`
`bool std::operator== (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<typename _Tp1 , typename _Tp2 >`
`bool std::operator> (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept`
- `template<typename _Tp >`
`bool std::operator> (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp >`
`bool std::operator> (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<typename _Tp1 , typename _Tp2 >`
`bool std::operator>= (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept`
- `template<typename _Tp >`
`bool std::operator>= (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp >`
`bool std::operator>= (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<typename _Tp , typename _Tp1 >`
`shared_ptr< _Tp > std::static_pointer_cast (const shared_ptr< _Tp1 > &__r) noexcept`
- `template<typename _Tp >`
`void std::swap (shared_ptr< _Tp > &__a, shared_ptr< _Tp > &__b) noexcept`
- `template<typename _Tp >`
`void std::swap (weak_ptr< _Tp > &__a, weak_ptr< _Tp > &__b) noexcept`

5.369.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

5.370 `shared_ptr_base.h` File Reference

Classes

- `struct std::_Sp_ebo_helper< _Nm, _Tp, false >`
- `struct std::_Sp_ebo_helper< _Nm, _Tp, true >`
- `class std::bad_weak_ptr`
- `class std::enable_shared_from_this< _Tp >`
- `struct std::hash< __shared_ptr< _Tp, _Lp > >`
- `struct std::owner_less< _Tp >`
- `class std::shared_ptr< _Tp >`
- `class std::weak_ptr< _Tp >`

Namespaces

- `std`

Functions

- `template<typename _Tp, _Lock_policy _Lp, typename _Alloc, typename... _Args>`
`__shared_ptr< _Tp, _Lp > std::__allocate_shared (const _Alloc &__a, _Args &&...__args)`
- `template<typename _Tp1, typename _Tp2 >`
`void std::__enable_shared_from_this_helper (const __shared_count<> &__pn, const enable_shared_from_←`
`_this< _Tp1 > *__pe, const _Tp2 *__px) noexcept`
- `template<_Lock_policy _Lp, typename _Tp1, typename _Tp2 >`
`void std::__enable_shared_from_this_helper (const __shared_count< _Lp > &, const __enable_shared_←`
`from_this< _Tp1, _Lp > *, const _Tp2 *) noexcept`
- `template<_Lock_policy _Lp>`
`void std::__enable_shared_from_this_helper (const __shared_count< _Lp > &,...) noexcept`
- `template<_Lock_policy _Lp1, typename _Tp1, typename _Tp2 >`
`void std::__enable_shared_from_this_helper (const __shared_count< _Lp1 > &__pn, const __enable_←`
`shared_from_this< _Tp1, _Lp1 > *__pe, const _Tp2 *__px) noexcept`
- `template<typename _Tp, _Lock_policy _Lp, typename... _Args>`
`__shared_ptr< _Tp, _Lp > std::__make_shared (_Args &&...__args)`
- `void std::__throw_bad_weak_ptr ()`
- `template<typename _Tp, typename _Tp1, _Lock_policy _Lp>`
`__shared_ptr< _Tp, _Lp > std::const_pointer_cast (const __shared_ptr< _Tp1, _Lp > &__r) noexcept`
- `template<typename _Tp, typename _Tp1, _Lock_policy _Lp>`
`__shared_ptr< _Tp, _Lp > std::dynamic_pointer_cast (const __shared_ptr< _Tp1, _Lp > &__r) noexcept`
- `template<typename _Tp1, typename _Tp2, _Lock_policy _Lp>`
`bool std::operator!= (const __shared_ptr< _Tp1, _Lp > &__a, const __shared_ptr< _Tp2, _Lp > &__b) noex-`
`cept`
- `template<typename _Tp, _Lock_policy _Lp>`
`bool std::operator!= (const __shared_ptr< _Tp, _Lp > &__a, nullptr_t) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>`
`bool std::operator!= (nullptr_t, const __shared_ptr< _Tp, _Lp > &__a) noexcept`
- `template<typename _Tp1, typename _Tp2, _Lock_policy _Lp>`
`bool std::operator< (const __shared_ptr< _Tp1, _Lp > &__a, const __shared_ptr< _Tp2, _Lp > &__b) noex-`
`cept`
- `template<typename _Tp, _Lock_policy _Lp>`
`bool std::operator< (const __shared_ptr< _Tp, _Lp > &__a, nullptr_t) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>`
`bool std::operator< (nullptr_t, const __shared_ptr< _Tp, _Lp > &__a) noexcept`
- `template<typename _Tp1, typename _Tp2, _Lock_policy _Lp>`
`bool std::operator<= (const __shared_ptr< _Tp1, _Lp > &__a, const __shared_ptr< _Tp2, _Lp > &__b) noex-`
`cept`
- `template<typename _Tp, _Lock_policy _Lp>`
`bool std::operator<= (const __shared_ptr< _Tp, _Lp > &__a, nullptr_t) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>`
`bool std::operator<= (nullptr_t, const __shared_ptr< _Tp, _Lp > &__a) noexcept`
- `template<typename _Tp1, typename _Tp2, _Lock_policy _Lp>`
`bool std::operator== (const __shared_ptr< _Tp1, _Lp > &__a, const __shared_ptr< _Tp2, _Lp > &__b) noex-`
`cept`
- `template<typename _Tp, _Lock_policy _Lp>`
`bool std::operator== (const __shared_ptr< _Tp, _Lp > &__a, nullptr_t) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>`
`bool std::operator== (nullptr_t, const __shared_ptr< _Tp, _Lp > &__a) noexcept`
- `template<typename _Tp1, typename _Tp2, _Lock_policy _Lp>`
`bool std::operator> (const __shared_ptr< _Tp1, _Lp > &__a, const __shared_ptr< _Tp2, _Lp > &__b) noex-`
`cept`

- `template<typename _Tp, _Lock_policy _Lp>`
`bool std::operator> (const __shared_ptr< _Tp, _Lp > &__a, nullptr_t) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>`
`bool std::operator> (nullptr_t, const __shared_ptr< _Tp, _Lp > &__a) noexcept`
- `template<typename _Tp1, typename _Tp2, _Lock_policy _Lp>`
`bool std::operator>= (const __shared_ptr< _Tp1, _Lp > &__a, const __shared_ptr< _Tp2, _Lp > &__b) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>`
`bool std::operator>= (const __shared_ptr< _Tp, _Lp > &__a, nullptr_t) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>`
`bool std::operator>= (nullptr_t, const __shared_ptr< _Tp, _Lp > &__a) noexcept`
- `template<typename _Tp, typename _Tp1, _Lock_policy _Lp>`
`__shared_ptr< _Tp, _Lp > std::static_pointer_cast (const __shared_ptr< _Tp1, _Lp > &__r) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>`
`void std::swap (__shared_ptr< _Tp, _Lp > &__a, __shared_ptr< _Tp, _Lp > &__b) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>`
`void std::swap (__weak_ptr< _Tp, _Lp > &__a, __weak_ptr< _Tp, _Lp > &__b) noexcept`

5.370.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

5.371 `size_fn_imps.hpp` File Reference

5.371.1 Detailed Description

Contains implementations of `cc_ht_map_'s` entire container size related functions.

5.372 `slice_array.h` File Reference

Classes

- class `std::slice`
- class `std::slice_array< _Tp >`

Namespaces

- `std`

Macros

- `#define _DEFINE_VALARRAY_OPERATOR(_Op, _Name)`

5.372.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<valarray>`.

5.373 sort.h File Reference

Namespaces

- [__gnu_parallel](#)

Functions

- `template<bool __stable, typename _RAIter, typename _Compare, typename _Parallelism >
void __gnu_parallel::__parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, _Parallelism __↵
parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare >
void __gnu_parallel::__parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, multiway_mergesort↵
__tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare >
void __gnu_parallel::__parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, multiway_mergesort↵
__exact_tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare >
void __gnu_parallel::__parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, multiway_mergesort↵
__sampling_tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare >
void __gnu_parallel::__parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, quicksort_tag __↵
parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare >
void __gnu_parallel::__parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, balanced_quicksort↵
__tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare >
void __gnu_parallel::__parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, default_parallel_tag
__parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare >
void __gnu_parallel::__parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, parallel_tag __↵
parallelism)`

5.373.1 Detailed Description

Parallel sorting algorithm switch. This file is a GNU parallel extension to the Standard C++ Library.

5.374 splay_fn_imps.hpp File Reference

5.374.1 Detailed Description

Contains an implementation class for `splay_tree_`.

5.375 `splay_tree_.hpp` File Reference

Classes

- class [__gnu_pbds::detail::splay_tree_map](#)< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >

Namespaces

- [__gnu_pbds](#)

Macros

- `#define PB_DS_ASSERT_BASE_NODE_CONSISTENT(_Node)`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_S_TREE_BASE`
- `#define PB_DS_S_TREE_BASE_NAME`
- `#define PB_DS_S_TREE_NAME`

5.375.1 Detailed Description

Contains an implementation class for splay trees.

5.376 `split_fn_imps.hpp` File Reference

5.376.1 Detailed Description

Contains an implementation class for `pat_trie`.

5.377 `split_join_fn_imps.hpp` File Reference

5.377.1 Detailed Description

Contains an implementation class for a `binary_heap`.

5.378 `split_join_fn_imps.hpp` File Reference

5.378.1 Detailed Description

Contains an implementation class for a base of binomial heaps.

5.379 `split_join_fn_imps.hpp` File Reference

5.379.1 Detailed Description

Contains an implementation class for `bin_search_tree_`.

5.380 `split_join_fn_imps.hpp` File Reference

5.380.1 Detailed Description

Contains an implementation class for `ov_tree_`.

5.381 `split_join_fn_imps.hpp` File Reference

5.381.1 Detailed Description

Contains an implementation class for a pairing heap.

5.382 `split_join_fn_imps.hpp` File Reference

5.382.1 Detailed Description

Contains an implementation for `rb_tree_`.

5.383 `split_join_fn_imps.hpp` File Reference

5.383.1 Detailed Description

Contains an implementation for `rc_binomial_heap_`.

5.384 `split_join_fn_imps.hpp` File Reference

5.384.1 Detailed Description

Contains an implementation class for `splay_tree_`.

5.385 `split_join_fn_imps.hpp` File Reference

5.385.1 Detailed Description

Contains an implementation for `thin_heap_`.

5.386 `sso_string_base.h` File Reference

Namespaces

- [__gnu_cxx](#)

5.386.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ext/vstring.h>`.

5.387 `standard_policies.hpp` File Reference

Classes

- struct [__gnu_pbds::detail::default_comb_hash_fn](#)
- struct [__gnu_pbds::detail::default_eq_fn< Key >](#)
- struct [__gnu_pbds::detail::default_hash_fn< Key >](#)
- struct [__gnu_pbds::detail::default_probe_fn< Comb_Probe_Fn >](#)
- struct [__gnu_pbds::detail::default_resize_policy< Comb_Hash_Fn >](#)
- struct [__gnu_pbds::detail::default_trie_access_traits< Key >](#)
- struct [__gnu_pbds::detail::default_trie_access_traits< std::basic_string< Char, Char_Traits, std::allocator< char > > >](#)
- struct [__gnu_pbds::detail::default_update_policy](#)

Namespaces

- [__gnu_pbds](#)

Macros

- `#define __dtrie_alloc`
- `#define __dtrie_string`

Enumerations

- enum { `default_store_hash` }

5.387.1 Detailed Description

Contains standard policies for containers.

5.387.2 Enumeration Type Documentation

5.387.2.1 anonymous enum

Enumeration for default behavior of stored hash data.

Definition at line 74 of file `standard_policies.hpp`.

5.388 `stdc++.h` File Reference

5.388.1 Detailed Description

This is an implementation file for a precompiled header.

5.389 `stdio_filebuf.h` File Reference

Classes

- class [__gnu_cxx::stdio_filebuf<_CharT, _Traits>](#)

Namespaces

- [__gnu_cxx](#)

5.389.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

5.390 `stdio_sync_filebuf.h` File Reference

Classes

- class [__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>](#)

Namespaces

- [__gnu_cxx](#)

5.390.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

5.391 `stdtr1c++.h` File Reference

5.391.1 Detailed Description

This is an implementation file for a precompiled header.

5.392 `stl_algo.h` File Reference

Namespaces

- [std](#)

Enumerations

- enum { **_S_threshold** }
- enum { **_S_chunk_size** }

Functions

- template<typename `_ForwardIterator` , typename `_BinaryPredicate` >
`_ForwardIterator` **std::adjacent_find** (`_ForwardIterator` __first, `_ForwardIterator` __last, `_BinaryPredicate` __↵
`binary_pred`)
- template<typename `_RandomAccessIterator` , typename `_Distance` , typename `_Compare` >
void **std::chunk_insertion_sort** (`_RandomAccessIterator` __first, `_RandomAccessIterator` __last, `_Distance`
`__chunk_size`, `_Compare` __comp)
- template<typename `_InputIterator` , typename `_Size` , typename `_OutputIterator` >
`_OutputIterator` **std::copy_n** (`_InputIterator` __first, `_Size` __n, `_OutputIterator` __result, `input_iterator_tag`)
- template<typename `_RandomAccessIterator` , typename `_Size` , typename `_OutputIterator` >
`_OutputIterator` **std::copy_n** (`_RandomAccessIterator` __first, `_Size` __n, `_OutputIterator` __result, `random_`↵
`access_iterator_tag`)
- template<typename `_InputIterator` , typename `_Predicate` >
iterator_traits< `_InputIterator` >::difference_type **std::count_if** (`_InputIterator` __first, `_InputIterator` __last, ↵
`_Predicate` __pred)
- template<typename `_ForwardIterator` , typename `_Tp` , typename `_CompareItTp` , typename `_CompareTplt` >
pair< `_ForwardIterator`, `_ForwardIterator` > **std::equal_range** (`_ForwardIterator` __first, `_ForwardIterator` __↵
`last`, const `_Tp` &__val, `_CompareItTp` __comp_it_val, `_CompareTplt` __comp_val_it)
- template<typename `_RandomAccessIterator` , typename `_Compare` >
void **std::final_insertion_sort** (`_RandomAccessIterator` __first, `_RandomAccessIterator` __last, `_Compare` __↵
`comp`)
- template<typename `_ForwardIterator1` , typename `_ForwardIterator2` , typename `_BinaryPredicate` >
`_ForwardIterator1` **std::find_end** (`_ForwardIterator1` __first1, `_ForwardIterator1` __last1, `_ForwardIterator2` ↵
`__first2`, `_ForwardIterator2` __last2, `forward_iterator_tag`, `forward_iterator_tag`, `_BinaryPredicate` __comp)

- `template<typename _BidirectionalIterator1, typename _BidirectionalIterator2, typename _BinaryPredicate >`
`_BidirectionalIterator1 std::find_end (_BidirectionalIterator1 __first1, _BidirectionalIterator1 __last1, _BidirectionalIterator2 __first2, _BidirectionalIterator2 __last2, bidirectional_iterator_tag, bidirectional_iterator_tag, _BinaryPredicate __comp)`
- `template<typename _InputIterator, typename _Predicate >`
`_InputIterator std::find_if (_InputIterator __first, _InputIterator __last, _Predicate __pred, input_iterator_tag)`
- `template<typename _RandomAccessIterator, typename _Predicate >`
`_RandomAccessIterator std::find_if (_RandomAccessIterator __first, _RandomAccessIterator __last, _Predicate __pred, random_access_iterator_tag)`
- `template<typename _Iterator, typename _Predicate >`
`_Iterator std::find_if (_Iterator __first, _Iterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _Predicate >`
`_InputIterator std::find_if_not (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _Predicate, typename _Distance >`
`_InputIterator std::find_if_not_n (_InputIterator __first, _Distance & __len, _Predicate __pred)`
- `template<typename _EuclideanRingElement >`
`_EuclideanRingElement std::gcd (_EuclideanRingElement __m, _EuclideanRingElement __n)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void std::heap_select (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Compare >`
`bool std::includes (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _Compare __comp)`
- `template<typename _BidirectionalIterator, typename _Compare >`
`void std::inplace_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Predicate, typename _Distance >`
`_ForwardIterator std::inplace_stable_partition (_ForwardIterator __first, _Predicate __pred, _Distance __len)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void std::inplace_stable_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void std::insertion_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Size, typename _Compare >`
`void std::introsort (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __last, _Size __depth_limit, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Size, typename _Compare >`
`void std::introsort_loop (_RandomAccessIterator __first, _RandomAccessIterator __last, _Size __depth_limit, _Compare __comp)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate >`
`bool std::is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _BinaryPredicate __pred)`
- `template<typename _ForwardIterator, typename _Compare >`
`_ForwardIterator std::is_sorted_until (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Compare >`
`_ForwardIterator std::max_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`
`_OutputIterator std::merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _BidirectionalIterator, typename _Distance, typename _Pointer, typename _Compare >`
`void std::merge_adaptive (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Distance __len1, _Distance __len2, _Pointer __buffer, _Distance __buffer_size, _Compare __comp)`

- `template<typename _RandomAccessIterator1, typename _RandomAccessIterator2, typename _Distance, typename _Compare >`
`void std::merge_sort_loop (_RandomAccessIterator1 __first, _RandomAccessIterator1 __last, _RandomAccessIterator2 __result, _Distance __step_size, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Pointer, typename _Compare >`
`void std::merge_sort_with_buffer (_RandomAccessIterator __first, _RandomAccessIterator __last, _Pointer __buffer, _Compare __comp)`
- `template<typename _BidirectionalIterator, typename _Distance, typename _Compare >`
`void std::merge_without_buffer (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Distance __len1, _Distance __len2, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Compare >`
`_ForwardIterator std::min_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Compare >`
`pair< _ForwardIterator, _ForwardIterator > std::minmax_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _Iterator, typename _Compare >`
`void std::move_median_to_first (_Iterator __result, _Iterator __a, _Iterator __b, _Iterator __c, _Compare __comp)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Compare >`
`_OutputIterator std::move_merge (_InputIterator __first1, _InputIterator __last1, _InputIterator __first2, _InputIterator __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`
`void std::move_merge_adaptive (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _BidirectionalIterator1, typename _BidirectionalIterator2, typename _BidirectionalIterator3, typename _Compare >`
`void std::move_merge_adaptive_backward (_BidirectionalIterator1 __first1, _BidirectionalIterator1 __last1, _BidirectionalIterator2 __first2, _BidirectionalIterator2 __last2, _BidirectionalIterator3 __result, _Compare __comp)`
- `template<typename _BidirectionalIterator, typename _Compare >`
`bool std::next_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void std::partial_sort (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _Compare >`
`_RandomAccessIterator std::partial_sort_copy (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __result_first, _RandomAccessIterator __result_last, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Predicate >`
`_ForwardIterator std::partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, forward_iterator_tag)`
- `template<typename _BidirectionalIterator, typename _Predicate >`
`_BidirectionalIterator std::partition (_BidirectionalIterator __first, _BidirectionalIterator __last, _Predicate __pred, bidirectional_iterator_tag)`
- `template<typename _BidirectionalIterator, typename _Compare >`
`bool std::prev_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate >`
`_OutputIterator std::remove_copy_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Predicate >`
`_ForwardIterator std::remove_if (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate, typename _Tp >`
`_OutputIterator std::replace_copy_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred, const _Tp &__new_value)`
- `template<typename _BidirectionalIterator >`
`void std::reverse (_BidirectionalIterator __first, _BidirectionalIterator __last, bidirectional_iterator_tag)`

- `template<typename _RandomAccessIterator >`
`void std::__reverse (_RandomAccessIterator __first, _RandomAccessIterator __last, random_access_iterator_tag)`
- `template<typename _ForwardIterator >`
`void std::__rotate (_ForwardIterator __first, _ForwardIterator __middle, _ForwardIterator __last, forward_iterator_tag)`
- `template<typename _BidirectionalIterator >`
`void std::__rotate (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, bidirectional_iterator_tag)`
- `template<typename _RandomAccessIterator >`
`void std::__rotate (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last, random_access_iterator_tag)`
- `template<typename _BidirectionalIterator1, typename _BidirectionalIterator2, typename _Distance >`
`_BidirectionalIterator1 std::__rotate_adaptive (_BidirectionalIterator1 __first, _BidirectionalIterator1 __middle, _BidirectionalIterator1 __last, _Distance __len1, _Distance __len2, _BidirectionalIterator2 __buffer, _Distance __buffer_size)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate >`
`_ForwardIterator1 std::__search (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate __predicate)`
- `template<typename _ForwardIterator, typename _Integer, typename _UnaryPredicate >`
`_ForwardIterator std::__search_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, _UnaryPredicate __unary_pred)`
- `template<typename _ForwardIterator, typename _Integer, typename _UnaryPredicate >`
`_ForwardIterator std::__search_n_aux (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, _UnaryPredicate __unary_pred, std::forward_iterator_tag)`
- `template<typename _RandomAccessIter, typename _Integer, typename _UnaryPredicate >`
`_RandomAccessIter std::__search_n_aux (_RandomAccessIter __first, _RandomAccessIter __last, _Integer __count, _UnaryPredicate __unary_pred, std::random_access_iterator_tag)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`
`_OutputIterator std::__set_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`
`_OutputIterator std::__set_intersection (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`
`_OutputIterator std::__set_symmetric_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`
`_OutputIterator std::__set_union (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void std::__sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Predicate >`
`_ForwardIterator std::__stable_partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Pointer, typename _Predicate, typename _Distance >`
`_ForwardIterator std::__stable_partition_adaptive (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, _Distance __len, _Pointer __buffer, _Distance __buffer_size)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void std::__stable_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Pointer, typename _Distance, typename _Compare >`
`void std::__stable_sort_adaptive (_RandomAccessIterator __first, _RandomAccessIterator __last, _Pointer __buffer, _Distance __buffer_size, _Compare __comp)`

- `template<typename _RandomAccessIterator, typename _Compare >`
`void std::unguarded_insertion_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void std::unguarded_linear_insert (_RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`_RandomAccessIterator std::unguarded_partition (_RandomAccessIterator __first, _RandomAccessIterator __last, _RandomAccessIterator __pivot, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`_RandomAccessIterator std::unguarded_partition_pivot (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator, typename _BinaryPredicate >`
`_ForwardIterator std::unique (_ForwardIterator __first, _ForwardIterator __last, _BinaryPredicate __binary_pred)`
- `template<typename _ForwardIterator, typename _OutputIterator, typename _BinaryPredicate >`
`_OutputIterator std::unique_copy (_ForwardIterator __first, _ForwardIterator __last, _OutputIterator __result, _BinaryPredicate __binary_pred, forward_iterator_tag, output_iterator_tag)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryPredicate >`
`_OutputIterator std::unique_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryPredicate __binary_pred, input_iterator_tag, output_iterator_tag)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _BinaryPredicate >`
`_ForwardIterator std::unique_copy (_InputIterator __first, _InputIterator __last, _ForwardIterator __result, _BinaryPredicate __binary_pred, input_iterator_tag, forward_iterator_tag)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`
`_ForwardIterator std::upper_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)`
- `template<typename _ForwardIterator >`
`_ForwardIterator std::adjacent_find (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _BinaryPredicate >`
`_ForwardIterator std::adjacent_find (_ForwardIterator __first, _ForwardIterator __last, _BinaryPredicate __binary_pred)`
- `template<typename _InputIterator, typename _Predicate >`
`bool std::all_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _Predicate >`
`bool std::any_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Tp >`
`bool std::binary_search (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`
`bool std::binary_search (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate >`
`_OutputIterator std::copy_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred)`
- `template<typename _InputIterator, typename _Size, typename _OutputIterator >`
`_OutputIterator std::copy_n (_InputIterator __first, _Size __n, _OutputIterator __result)`
- `template<typename _InputIterator, typename _Tp >`
`iterator_traits< _InputIterator >::difference_type std::count (_InputIterator __first, _InputIterator __last, const _Tp &__value)`
- `template<typename _InputIterator, typename _Predicate >`
`iterator_traits< _InputIterator >::difference_type std::count_if (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Tp >`
`pair< _ForwardIterator, _ForwardIterator > std::equal_range (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)`

- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`
`pair< _ForwardIterator, _ForwardIterator > std::equal_range (_ForwardIterator __first, _ForwardIterator __last,`
`const _Tp &__val, _Compare __comp)`
- `template<typename _InputIterator, typename _Tp >`
`_InputIterator std::find (_InputIterator __first, _InputIterator __last, const _Tp &__val)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`
`_ForwardIterator1 std::find_end (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __↵`
`first2, _ForwardIterator2 __last2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate >`
`_ForwardIterator1 std::find_end (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __↵`
`first2, _ForwardIterator2 __last2, _BinaryPredicate __comp)`
- `template<typename _InputIterator, typename _ForwardIterator >`
`_InputIterator std::find_first_of (_InputIterator __first1, _InputIterator __last1, _ForwardIterator __first2, ↵`
`ForwardIterator __last2)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _BinaryPredicate >`
`_InputIterator std::find_first_of (_InputIterator __first1, _InputIterator __last1, _ForwardIterator __first2, ↵`
`ForwardIterator __last2, _BinaryPredicate __comp)`
- `template<typename _InputIterator, typename _Predicate >`
`_InputIterator std::find_if (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _Predicate >`
`_InputIterator std::find_if_not (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _Function >`
`_Function std::for_each (_InputIterator __first, _InputIterator __last, _Function __f)`
- `template<typename _ForwardIterator, typename _Generator >`
`void std::generate (_ForwardIterator __first, _ForwardIterator __last, _Generator __gen)`
- `template<typename _OutputIterator, typename _Size, typename _Generator >`
`_OutputIterator std::generate_n (_OutputIterator __first, _Size __n, _Generator __gen)`
- `template<typename _InputIterator1, typename _InputIterator2 >`
`bool std::includes (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __↵`
`last2)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Compare >`
`bool std::includes (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __↵`
`last2, _Compare __comp)`
- `template<typename _BidirectionalIterator >`
`void std::inplace_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __↵`
`last)`
- `template<typename _BidirectionalIterator, typename _Compare >`
`void std::inplace_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __↵`
`last, _Compare __comp)`
- `template<typename _InputIterator, typename _Predicate >`
`bool std::is_partitioned (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`
`bool std::is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate >`
`bool std::is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, ↵`
`BinaryPredicate __pred)`
- `template<typename _ForwardIterator >`
`bool std::is_sorted (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >`
`bool std::is_sorted (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator >`
`_ForwardIterator std::is_sorted_until (_ForwardIterator __first, _ForwardIterator __last)`

- `template<typename _ForwardIterator, typename _Compare >`
`_ForwardIterator std::is_sorted_until (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`
`_ForwardIterator std::lower_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)`
- `template<typename _Tp >`
`_Tp std::max (initializer_list< _Tp >)`
- `template<typename _Tp, typename _Compare >`
`_Tp std::max (initializer_list< _Tp >, _Compare)`
- `template<typename _ForwardIterator >`
`_ForwardIterator std::max_element (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >`
`_ForwardIterator std::max_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`
`_OutputIterator std::merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`
`_OutputIterator std::merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _Tp >`
`_Tp std::min (initializer_list< _Tp >)`
- `template<typename _Tp, typename _Compare >`
`_Tp std::min (initializer_list< _Tp >, _Compare)`
- `template<typename _ForwardIterator >`
`_ForwardIterator std::min_element (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >`
`_ForwardIterator std::min_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _Tp >`
`pair< const _Tp &, const _Tp & > std::minmax (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp, typename _Compare >`
`pair< const _Tp &, const _Tp & > std::minmax (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _Tp >`
`pair< _Tp, _Tp > std::minmax (initializer_list< _Tp >)`
- `template<typename _Tp, typename _Compare >`
`pair< _Tp, _Tp > std::minmax (initializer_list< _Tp >, _Compare)`
- `template<typename _ForwardIterator >`
`pair< _ForwardIterator, _ForwardIterator > std::minmax_element (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >`
`pair< _ForwardIterator, _ForwardIterator > std::minmax_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _BidirectionalIterator >`
`bool std::next_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _Compare >`
`bool std::next_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _InputIterator, typename _Predicate >`
`bool std::none_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _RandomAccessIterator >`
`void std::nth_element (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void std::nth_element (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __last, _Compare __comp)`

- `template<typename _RandomAccessIterator >`
`void std::partial_sort (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void std::partial_sort (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _InputIterator, typename _RandomAccessIterator >`
`_RandomAccessIterator std::partial_sort_copy (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __result_first, _RandomAccessIterator __result_last)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _Compare >`
`_RandomAccessIterator std::partial_sort_copy (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __result_first, _RandomAccessIterator __result_last, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Predicate >`
`_ForwardIterator std::partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _OutputIterator1, typename _OutputIterator2, typename _Predicate >`
`pair< _OutputIterator1, _OutputIterator2 > std::partition_copy (_InputIterator __first, _InputIterator __last, _OutputIterator1 __out_true, _OutputIterator2 __out_false, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Predicate >`
`_ForwardIterator std::partition_point (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _BidirectionalIterator >`
`bool std::prev_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _Compare >`
`bool std::prev_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`void std::random_shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _RandomNumberGenerator >`
`void std::random_shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last, _RandomNumberGenerator &&__rand)`
- `template<typename _ForwardIterator, typename _Tp >`
`_ForwardIterator std::remove (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Tp >`
`_OutputIterator std::remove_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, const _Tp &__value)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate >`
`_OutputIterator std::remove_copy_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Predicate >`
`_ForwardIterator std::remove_if (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Tp >`
`void std::replace (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__old_value, const _Tp &__new_value)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Tp >`
`_OutputIterator std::replace_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, const _Tp &__old_value, const _Tp &__new_value)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate, typename _Tp >`
`_OutputIterator std::replace_copy_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred, const _Tp &__new_value)`
- `template<typename _ForwardIterator, typename _Predicate, typename _Tp >`
`void std::replace_if (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, const _Tp &__new_value)`
- `template<typename _BidirectionalIterator >`
`void std::reverse (_BidirectionalIterator __first, _BidirectionalIterator __last)`

- `template<typename _BidirectionalIterator, typename _OutputIterator >`
`_OutputIterator std::reverse_copy (_BidirectionalIterator __first, _BidirectionalIterator __last, _OutputIterator __result)`
- `template<typename _ForwardIterator >`
`void std::rotate (_ForwardIterator __first, _ForwardIterator __middle, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _OutputIterator >`
`_OutputIterator std::rotate_copy (_ForwardIterator __first, _ForwardIterator __middle, _ForwardIterator __last, _OutputIterator __result)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`
`_ForwardIterator1 std::search (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate >`
`_ForwardIterator1 std::search (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate __predicate)`
- `template<typename _ForwardIterator, typename _Integer, typename _Tp >`
`_ForwardIterator std::search_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp &__val)`
- `template<typename _ForwardIterator, typename _Integer, typename _Tp, typename _BinaryPredicate >`
`_ForwardIterator std::search_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`
`_OutputIterator std::set_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`
`_OutputIterator std::set_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`
`_OutputIterator std::set_intersection (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`
`_OutputIterator std::set_intersection (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`
`_OutputIterator std::set_symmetric_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`
`_OutputIterator std::set_symmetric_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`
`_OutputIterator std::set_union (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`
`_OutputIterator std::set_union (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _UniformRandomNumberGenerator >`
`void std::shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last, _UniformRandomNumberGenerator &&__g)`
- `template<typename _RandomAccessIterator >`
`void std::sort (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void std::sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Predicate >`
`_ForwardIterator std::stable_partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`

- `template<typename _RandomAccessIterator >`
`void std::stable_sort (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void std::stable_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _InputIterator, typename _OutputIterator, typename _UnaryOperation >`
`_OutputIterator std::transform (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _UnaryOperation __unary_op)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _BinaryOperation >`
`_OutputIterator std::transform (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _OutputIterator __result, _BinaryOperation __binary_op)`
- `template<typename _ForwardIterator >`
`_ForwardIterator std::unique (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _BinaryPredicate >`
`_ForwardIterator std::unique (_ForwardIterator __first, _ForwardIterator __last, _BinaryPredicate __binary_pred)`
- `template<typename _InputIterator, typename _OutputIterator >`
`_OutputIterator std::unique_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryPredicate >`
`_OutputIterator std::unique_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryPredicate __binary_pred)`
- `template<typename _ForwardIterator, typename _Tp >`
`_ForwardIterator std::upper_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`
`_ForwardIterator std::upper_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)`

5.392.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<algorithm>`.

5.393 `std_algobase.h` File Reference

Classes

- `struct std::char_traits< _CharT >`
- `class std::istreambuf_iterator< _CharT, _Traits >`
- `class std::ostreambuf_iterator< _CharT, _Traits >`

Namespaces

- `std`

Macros

- `#define _GLIBCXX_MOVE3(_Tp, _Up, _Vp)`
- `#define _GLIBCXX_MOVE_BACKWARD3(_Tp, _Up, _Vp)`

Functions

- `template<bool _IsMove, typename _II, typename _OI >`
`_OI std::__copy_move_a (_II __first, _II __last, _OI __result)`
- `template<bool _IsMove, typename _CharT >`
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT, char_traits< _CharT >`
`> >::__type std::__copy_move_a2 (_CharT *, _CharT *, ostreambuf_iterator< _CharT, char_traits< _CharT`
`> >)`
- `template<bool _IsMove, typename _CharT >`
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT, char_traits< _CharT`
`> > >::__type std::__copy_move_a2 (const _CharT *, const _CharT *, ostreambuf_iterator< _CharT, char_`
`traits< _CharT > >)`
- `template<bool _IsMove, typename _CharT >`
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, _CharT * >::__type std::__copy_move_a2`
`(istreambuf_iterator< _CharT, char_traits< _CharT > >, istreambuf_iterator< _CharT, char_traits< _CharT`
`> >, _CharT *)`
- `template<bool _IsMove, typename _II, typename _OI >`
`_OI std::__copy_move_a2 (_II __first, _II __last, _OI __result)`
- `template<bool _IsMove, typename _BI1, typename _BI2 >`
`_BI2 std::__copy_move_backward_a (_BI1 __first, _BI1 __last, _BI2 __result)`
- `template<bool _IsMove, typename _BI1, typename _BI2 >`
`_BI2 std::__copy_move_backward_a2 (_BI1 __first, _BI1 __last, _BI2 __result)`
- `template<typename _II1, typename _II2 >`
`bool std::__equal_aux (_II1 __first1, _II1 __last1, _II2 __first2)`
- `template<typename _ForwardIterator, typename _Tp >`
`__gnu_cxx::__enable_if< !__is_scalar< _Tp >::__value, void >::__type std::__fill_a (_ForwardIterator __first,`
`_ForwardIterator __last, const _Tp &__value)`
- `template<typename _ForwardIterator, typename _Tp >`
`__gnu_cxx::__enable_if< __is_scalar< _Tp >::__value, void >::__type std::__fill_a (_ForwardIterator __first,`
`_ForwardIterator __last, const _Tp &__value)`
- `template<typename _Tp >`
`__gnu_cxx::__enable_if< __is_byte< _Tp >::__value, void >::__type std::__fill_a (_Tp *__first, _Tp *__last,`
`const _Tp &__c)`
- `template<typename _OutputIterator, typename _Size, typename _Tp >`
`__gnu_cxx::__enable_if< !__is_scalar< _Tp >::__value, _OutputIterator >::__type std::__fill_n_a (_Output`
`Iterator __first, _Size __n, const _Tp &__value)`
- `template<typename _OutputIterator, typename _Size, typename _Tp >`
`__gnu_cxx::__enable_if< __is_scalar< _Tp >::__value, _OutputIterator >::__type std::__fill_n_a (_Output`
`Iterator __first, _Size __n, const _Tp &__value)`
- `template<typename _Size, typename _Tp >`
`__gnu_cxx::__enable_if< __is_byte< _Tp >::__value, _Tp * >::__type std::__fill_n_a (_Tp *__first, _Size __n,`
`const _Tp &__c)`
- `template<typename _II1, typename _II2 >`
`bool std::__lexicographical_compare_aux (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2)`
- `template<typename _II1, typename _II2, typename _Compare >`
`bool std::__lexicographical_compare_impl (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2, _Compare`
`__comp)`
- `constexpr int std::__lg (int __n)`
- `constexpr unsigned std::__lg (unsigned __n)`
- `constexpr long std::__lg (long __n)`
- `constexpr unsigned long std::__lg (unsigned long __n)`
- `constexpr long long std::__lg (long long __n)`
- `constexpr unsigned long long std::__lg (unsigned long long __n)`

- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`
`_ForwardIterator std::lower_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate >`
`pair< _InputIterator1, _InputIterator2 > std::mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _BinaryPredicate __binary_pred)`
- `template<typename _Iterator >`
`_Miter_base< _Iterator >::iterator_type std::miter_base (_Iterator __it)`
- `template<typename _Iterator >`
`_Niter_base< _Iterator >::iterator_type std::niter_base (_Iterator __it)`
- `template<typename _II, typename _OI >`
`_OI std::copy (_II __first, _II __last, _OI __result)`
- `template<typename _BI1, typename _BI2 >`
`_BI2 std::copy_backward (_BI1 __first, _BI1 __last, _BI2 __result)`
- `template<typename _Iiter1, typename _Iiter2, typename _BinaryPredicate >`
`bool std::equal (_Iiter1 __first1, _Iiter1 __last1, _Iiter2 __first2, _BinaryPredicate __binary_pred)`
- `template<typename _II1, typename _II2 >`
`bool std::equal (_II1 __first1, _II1 __last1, _II2 __first2)`
- `template<typename _ForwardIterator, typename _Tp >`
`void std::fill (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)`
- `template<typename _OI, typename _Size, typename _Tp >`
`_OI std::fill_n (_OI __first, _Size __n, const _Tp &__value)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`
`void std::iter_swap (_ForwardIterator1 __a, _ForwardIterator2 __b)`
- `template<typename _II1, typename _II2 >`
`bool std::lexicographical_compare (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2)`
- `template<typename _II1, typename _II2, typename _Compare >`
`bool std::lexicographical_compare (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Tp >`
`_ForwardIterator std::lower_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)`
- `template<typename _Tp >`
`const _Tp & std::max (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp, typename _Compare >`
`const _Tp & std::max (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _Tp >`
`const _Tp & std::min (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp, typename _Compare >`
`const _Tp & std::min (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2 >`
`pair< _InputIterator1, _InputIterator2 > std::mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate >`
`pair< _InputIterator1, _InputIterator2 > std::mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _BinaryPredicate __binary_pred)`
- `template<typename _II, typename _OI >`
`_OI std::move (_II __first, _II __last, _OI __result)`
- `template<typename _BI1, typename _BI2 >`
`_BI2 std::move_backward (_BI1 __first, _BI1 __last, _BI2 __result)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`
`_ForwardIterator2 std::swap_ranges (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2)`

5.393.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<algorithm>`.

5.394 `std_bvector.h` File Reference

Classes

- struct `std::hash<::vector< bool, _Alloc > >`
- class `std::vector< bool, _Alloc >`

Namespaces

- `std`

Typedefs

- typedef unsigned long `std::_Bit_type`

Enumerations

- enum { `_S_word_bit` }

Functions

- void `std::_fill_bvector` (`_Bit_iterator __first`, `_Bit_iterator __last`, `bool __x`)
- void `std::fill` (`_Bit_iterator __first`, `_Bit_iterator __last`, `const bool &__x`)
- `_Bit_iterator std::operator+` (`ptrdiff_t __n`, `const _Bit_iterator &__x`)
- `_Bit_const_iterator std::operator+` (`ptrdiff_t __n`, `const _Bit_const_iterator &__x`)
- `ptrdiff_t std::operator-` (`const _Bit_iterator_base &__x`, `const _Bit_iterator_base &__y`)
- void `std::swap` (`_Bit_reference __x`, `_Bit_reference __y`) `noexcept`
- void `std::swap` (`_Bit_reference __x`, `bool &__y`) `noexcept`
- void `std::swap` (`bool &__x`, `_Bit_reference __y`) `noexcept`

5.394.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<vector>`.

5.395 `std_construct.h` File Reference

Namespaces

- `std`

Functions

- `template<typename _T1, typename... _Args>`
`void std::_Construct (_T1 *__p, _Args &&...__args)`
- `template<typename _Tp >`
`void std::_Destroy (_Tp *__pointer)`
- `template<typename _ForwardIterator >`
`void std::_Destroy (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Allocator >`
`void std::_Destroy (_ForwardIterator __first, _ForwardIterator __last, _Allocator &__alloc)`
- `template<typename _ForwardIterator, typename _Tp >`
`void std::_Destroy (_ForwardIterator __first, _ForwardIterator __last, allocator<_Tp > &)`

5.395.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

5.396 `std_deque.h` File Reference

Classes

- `class std::_Deque_base<_Tp, _Alloc >`
- `struct std::_Deque_iterator<_Tp, _Ref, _Ptr >`
- `class std::deque<_Tp, _Alloc >`

Namespaces

- `std`

Macros

- `#define _GLIBCXX_DEQUE_BUF_SIZE`

Functions

- `size_t std::_deque_buf_size (size_t __size)`
- `template<typename _Tp >`
`_Deque_iterator<_Tp, _Tp &, _Tp * > std::copy (_Deque_iterator<_Tp, const _Tp &, const _Tp * >, _Deque_iterator<_Tp, const _Tp &, const _Tp * >, _Deque_iterator<_Tp, _Tp &, _Tp * >)`
- `template<typename _Tp >`
`_Deque_iterator<_Tp, _Tp &, _Tp * > std::copy (_Deque_iterator<_Tp, _Tp &, _Tp * > __first, _Deque_iterator<_Tp, _Tp &, _Tp * > __last, _Deque_iterator<_Tp, _Tp &, _Tp * > __result)`
- `template<typename _Tp >`
`_Deque_iterator<_Tp, _Tp &, _Tp * > std::copy_backward (_Deque_iterator<_Tp, const _Tp &, const _Tp * >, _Deque_iterator<_Tp, const _Tp &, const _Tp * >, _Deque_iterator<_Tp, _Tp &, _Tp * >)`

- `template<typename _Tp >`
`_Deque_iterator< _Tp, _Tp &, _Tp * > std::copy_backward (_Deque_iterator< _Tp, _Tp &, _Tp * > __first,`
`_Deque_iterator< _Tp, _Tp &, _Tp * > __last, _Deque_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _Tp >`
`void std::fill (const _Deque_iterator< _Tp, _Tp &, _Tp * > &, const _Deque_iterator< _Tp, _Tp &, _Tp * > &,`
`const _Tp &)`
- `template<typename _Tp >`
`_Deque_iterator< _Tp, _Tp &, _Tp * > std::move (_Deque_iterator< _Tp, const _Tp &, const _Tp * >, _Deque_iterator< _Tp, const _Tp &, const _Tp * >, _Deque_iterator< _Tp, _Tp &, _Tp * >)`
`_Deque_iterator< _Tp, _Tp &, _Tp * > std::move (_Deque_iterator< _Tp, _Tp &, _Tp * > __first, _Deque_iterator< _Tp, _Tp &, _Tp * > __last, _Deque_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _Tp >`
`_Deque_iterator< _Tp, _Tp &, _Tp * > std::move_backward (_Deque_iterator< _Tp, const _Tp &, const _Tp * >, _Deque_iterator< _Tp, const _Tp &, const _Tp * >, _Deque_iterator< _Tp, _Tp &, _Tp * >)`
- `template<typename _Tp >`
`_Deque_iterator< _Tp, _Tp &, _Tp * > std::move_backward (_Deque_iterator< _Tp, _Tp &, _Tp * > __first, _Deque_iterator< _Tp, _Tp &, _Tp * > __last, _Deque_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<typename _Tp, typename _Ref, typename _Ptr >`
`bool std::operator!= (const _Deque_iterator< _Tp, _Ref, _Ptr > &__x, const _Deque_iterator< _Tp, _Ref, _Ptr > &__y) noexcept`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR >`
`bool std::operator!= (const _Deque_iterator< _Tp, _RefL, _PtrL > &__x, const _Deque_iterator< _Tp, _RefR, _PtrR > &__y) noexcept`
- `template<typename _Tp, typename _Alloc >`
`bool std::operator!= (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Ref, typename _Ptr >`
`_Deque_iterator< _Tp, _Ref, _Ptr > std::operator+ (ptrdiff_t __n, const _Deque_iterator< _Tp, _Ref, _Ptr > &__x) noexcept`
- `template<typename _Tp, typename _Ref, typename _Ptr >`
`_Deque_iterator< _Tp, _Ref, _Ptr >::difference_type std::operator- (const _Deque_iterator< _Tp, _Ref, _Ptr > &__x, const _Deque_iterator< _Tp, _Ref, _Ptr > &__y) noexcept`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR >`
`_Deque_iterator< _Tp, _RefL, _PtrL >::difference_type std::operator- (const _Deque_iterator< _Tp, _RefL, _PtrL > &__x, const _Deque_iterator< _Tp, _RefR, _PtrR > &__y) noexcept`
- `template<typename _Tp, typename _Ref, typename _Ptr >`
`bool std::operator< (const _Deque_iterator< _Tp, _Ref, _Ptr > &__x, const _Deque_iterator< _Tp, _Ref, _Ptr > &__y) noexcept`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR >`
`bool std::operator< (const _Deque_iterator< _Tp, _RefL, _PtrL > &__x, const _Deque_iterator< _Tp, _RefR, _PtrR > &__y) noexcept`
- `template<typename _Tp, typename _Alloc >`
`bool std::operator< (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Ref, typename _Ptr >`
`bool std::operator<= (const _Deque_iterator< _Tp, _Ref, _Ptr > &__x, const _Deque_iterator< _Tp, _Ref, _Ptr > &__y) noexcept`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR >`
`bool std::operator<= (const _Deque_iterator< _Tp, _RefL, _PtrL > &__x, const _Deque_iterator< _Tp, _RefR, _PtrR > &__y) noexcept`
- `template<typename _Tp, typename _Alloc >`
`bool std::operator<= (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Ref, typename _Ptr >`
`bool std::operator== (const _Deque_iterator< _Tp, _Ref, _Ptr > &__x, const _Deque_iterator< _Tp, _Ref, _Ptr > &__y) noexcept`

- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR >`
`bool std::operator== (const _Deque_iterator< _Tp, _RefL, _PtrL > &__x, const _Deque_iterator< _Tp, _RefR, _PtrR > &__y) noexcept`
- `template<typename _Tp, typename _Alloc >`
`bool std::operator== (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Ref, typename _Ptr >`
`bool std::operator> (const _Deque_iterator< _Tp, _Ref, _Ptr > &__x, const _Deque_iterator< _Tp, _Ref, _Ptr > &__y) noexcept`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR >`
`bool std::operator> (const _Deque_iterator< _Tp, _RefL, _PtrL > &__x, const _Deque_iterator< _Tp, _RefR, _PtrR > &__y) noexcept`
- `template<typename _Tp, typename _Alloc >`
`bool std::operator> (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Ref, typename _Ptr >`
`bool std::operator>= (const _Deque_iterator< _Tp, _Ref, _Ptr > &__x, const _Deque_iterator< _Tp, _Ref, _Ptr > &__y) noexcept`
- `template<typename _Tp, typename _RefL, typename _PtrL, typename _RefR, typename _PtrR >`
`bool std::operator>= (const _Deque_iterator< _Tp, _RefL, _PtrL > &__x, const _Deque_iterator< _Tp, _RefR, _PtrR > &__y) noexcept`
- `template<typename _Tp, typename _Alloc >`
`bool std::operator>= (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`
`void std::swap (deque< _Tp, _Alloc > &__x, deque< _Tp, _Alloc > &__y)`

5.396.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<deque>`.

5.396.2 Macro Definition Documentation

5.396.2.1 `#define _GLIBCXX_DEQUE_BUF_SIZE`

This function controls the size of memory nodes.

Parameters

<code>__size</code>	The size of an element.
---------------------	-------------------------

Returns

The number (not byte size) of elements per node.

This function started off as a compiler kludge from SGI, but seems to be a useful wrapper around a repeated constant expression. The **512** is tunable (and no other code needs to change), but no investigation has been done since inheriting the SGI code. Touch `_GLIBCXX_DEQUE_BUF_SIZE` only if you know what you are doing, however: changing it breaks the binary compatibility!!

Definition at line 85 of file `stl_deque.h`.

5.397 `std_function.h` File Reference

Classes

- struct `std::binary_function<_Arg1, _Arg2, _Result >`
- class `std::binary_negate<_Predicate >`
- class `std::const_mem_fun1_ref_t<_Ret, _Tp, _Arg >`
- class `std::const_mem_fun1_t<_Ret, _Tp, _Arg >`
- class `std::const_mem_fun_ref_t<_Ret, _Tp >`
- class `std::const_mem_fun_t<_Ret, _Tp >`
- struct `std::divides<_Tp >`
- struct `std::equal_to<_Tp >`
- struct `std::greater<_Tp >`
- struct `std::greater_equal<_Tp >`
- struct `std::less<_Tp >`
- struct `std::less_equal<_Tp >`
- struct `std::logical_and<_Tp >`
- struct `std::logical_not<_Tp >`
- struct `std::logical_or<_Tp >`
- class `std::mem_fun1_ref_t<_Ret, _Tp, _Arg >`
- class `std::mem_fun1_t<_Ret, _Tp, _Arg >`
- class `std::mem_fun_ref_t<_Ret, _Tp >`
- class `std::mem_fun_t<_Ret, _Tp >`
- struct `std::minus<_Tp >`
- struct `std::modulus<_Tp >`
- struct `std::multiplies<_Tp >`
- struct `std::negate<_Tp >`
- struct `std::not_equal_to<_Tp >`
- struct `std::plus<_Tp >`
- class `std::pointer_to_binary_function<_Arg1, _Arg2, _Result >`
- class `std::pointer_to_unary_function<_Arg, _Result >`
- struct `std::unary_function<_Arg, _Result >`
- class `std::unary_negate<_Predicate >`

Namespaces

- `std`

Functions

- `template<typename _Ret, typename _Tp >`
`mem_fun_t<_Ret, _Tp > std::mem_fun (_Ret(_Tp::*__f)())`
- `template<typename _Ret, typename _Tp >`
`const_mem_fun_t<_Ret, _Tp > std::mem_fun (_Ret(_Tp::*__f)() const)`
- `template<typename _Ret, typename _Tp, typename _Arg >`
`mem_fun1_t<_Ret, _Tp, _Arg > std::mem_fun (_Ret(_Tp::*__f)(_Arg))`
- `template<typename _Ret, typename _Tp, typename _Arg >`
`const_mem_fun1_t<_Ret, _Tp, _Arg > std::mem_fun (_Ret(_Tp::*__f)(_Arg) const)`

- `template<typename _Ret, typename _Tp >`
`mem_fun_ref_t< _Ret, _Tp > std::mem_fun_ref (_Ret(_Tp::*__f)())`
- `template<typename _Ret, typename _Tp >`
`const_mem_fun_ref_t< _Ret, _Tp > std::mem_fun_ref (_Ret(_Tp::*__f)() const)`
- `template<typename _Ret, typename _Tp, typename _Arg >`
`mem_fun1_ref_t< _Ret, _Tp, _Arg > std::mem_fun_ref (_Ret(_Tp::*__f)(_Arg))`
- `template<typename _Ret, typename _Tp, typename _Arg >`
`const_mem_fun1_ref_t< _Ret, _Tp, _Arg > std::mem_fun_ref (_Ret(_Tp::*__f)(_Arg) const)`
- `template<typename _Predicate >`
`unary_negate< _Predicate > std::not1 (const _Predicate &__pred)`
- `template<typename _Predicate >`
`binary_negate< _Predicate > std::not2 (const _Predicate &__pred)`
- `template<typename _Arg, typename _Result >`
`pointer_to_unary_function< _Arg, _Result > std::ptr_fun (_Result(*__x)(_Arg))`
- `template<typename _Arg1, typename _Arg2, typename _Result >`
`pointer_to_binary_function< _Arg1, _Arg2, _Result > std::ptr_fun (_Result(*__x)(_Arg1, _Arg2))`

5.397.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<functional>`.

5.398 `std_heap.h` File Reference

Namespaces

- [std](#)

Functions

- `template<typename _RandomAccessIterator, typename _Distance, typename _Tp, typename _Compare >`
`void std::__adjust_heap (_RandomAccessIterator __first, _Distance __holeIndex, _Distance __len, _Tp __↵
value, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Distance >`
`bool std::__is_heap (_RandomAccessIterator __first, _Distance __n)`
- `template<typename _RandomAccessIterator, typename _Compare, typename _Distance >`
`bool std::__is_heap (_RandomAccessIterator __first, _Compare __comp, _Distance __n)`
- `template<typename _RandomAccessIterator >`
`bool std::__is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`bool std::__is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Distance, typename _Compare >`
`_Distance std::__is_heap_until (_RandomAccessIterator __first, _Distance __n, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void std::__make_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void std::__pop_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _RandomAccessIterator
__result, _Compare __comp)`

- `template<typename _RandomAccessIterator, typename _Distance, typename _Tp, typename _Compare >`
`void std::push_heap (_RandomAccessIterator __first, _Distance __holeIndex, _Distance __topIndex, _Tp &↵`
`__value, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void std::sort_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`bool std::is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`bool std::is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`_RandomAccessIterator std::is_heap_until (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`_RandomAccessIterator std::is_heap_until (_RandomAccessIterator __first, _RandomAccessIterator __last, &↵`
`_Compare __comp)`
- `template<typename _RandomAccessIterator >`
`void std::make_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void std::make_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`void std::pop_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void std::pop_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`void std::push_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void std::push_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`void std::sort_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void std::sort_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`

5.398.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<queue>`.

5.399 stl_iterator.h File Reference

Classes

- class `std::back_insert_iterator< _Container >`
- class `std::front_insert_iterator< _Container >`
- class `std::insert_iterator< _Container >`
- class `std::move_iterator< _Iterator >`
- class `std::reverse_iterator< _Iterator >`

Namespaces

- `__gnu_cxx`
- `std`

Macros

- `#define _GLIBCXX_MAKE_MOVE_IF_NOEXCEPT_ITERATOR(_Iter)`
- `#define _GLIBCXX_MAKE_MOVE_ITERATOR(_Iter)`

Functions

- `template<typename _Iterator, typename _ReturnType = typename conditional<__move_if_noexcept_cond<typename iterator_traits<_Iterator>::value_type>::value, _Iterator, move_iterator<_Iterator>>::type>
_ReturnType std::make_move_if_noexcept_iterator(_Iterator __i)`
- `template<typename _Container >
back_insert_iterator<_Container> std::back_inserter(_Container &__x)`
- `template<typename _Container >
front_insert_iterator<_Container> std::front_inserter(_Container &__x)`
- `template<typename _Container, typename _Iterator >
insert_iterator<_Container> std::inserter(_Container &__x, _Iterator __i)`
- `template<typename _Iterator >
move_iterator<_Iterator> std::make_move_iterator(_Iterator __i)`
- `template<typename _Iterator >
bool std::operator!= (const reverse_iterator<_Iterator> &__x, const reverse_iterator<_Iterator> &__y)`
- `template<typename _IteratorL, typename _IteratorR >
bool std::operator!= (const reverse_iterator<_IteratorL> &__x, const reverse_iterator<_IteratorR> &__y)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >
bool __gnu_cxx::operator!= (const __normal_iterator<_IteratorL, _Container> &__lhs, const __normal_iterator<_IteratorR, _Container> &__rhs) noexcept`
- `template<typename _Iterator, typename _Container >
bool __gnu_cxx::operator!= (const __normal_iterator<_Iterator, _Container> &__lhs, const __normal_iterator<_Iterator, _Container> &__rhs) noexcept`
- `template<typename _IteratorL, typename _IteratorR >
bool std::operator!= (const move_iterator<_IteratorL> &__x, const move_iterator<_IteratorR> &__y)`
- `template<typename _Iterator >
bool std::operator!= (const move_iterator<_Iterator> &__x, const move_iterator<_Iterator> &__y)`
- `template<typename _Iterator >
reverse_iterator<_Iterator> std::operator+ (typename reverse_iterator<_Iterator>::difference_type __n, const reverse_iterator<_Iterator> &__x)`
- `template<typename _Iterator, typename _Container >
__normal_iterator<_Iterator, _Container> __gnu_cxx::operator+ (typename __normal_iterator<_Iterator, _Container>::difference_type __n, const __normal_iterator<_Iterator, _Container> &__i) noexcept`
- `template<typename _Iterator >
move_iterator<_Iterator> std::operator+ (typename move_iterator<_Iterator>::difference_type __n, const move_iterator<_Iterator> &__x)`
- `template<typename _Iterator >
reverse_iterator<_Iterator>::difference_type std::operator- (const reverse_iterator<_Iterator> &__x, const reverse_iterator<_Iterator> &__y)`
- `template<typename _IteratorL, typename _IteratorR >
auto std::operator- (const reverse_iterator<_IteratorL> &__x, const reverse_iterator<_IteratorR> &__y) -> decltype(__y.base()-__x.base())`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >
auto __gnu_cxx::operator- (const __normal_iterator<_IteratorL, _Container> &__lhs, const __normal_iterator<_IteratorR, _Container> &__rhs) noexcept -> decltype(__lhs.base()-__rhs.base())`

- `template<typename _IteratorL, typename _IteratorR >`
`bool std::operator> (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >`
`bool __gnu_cxx::operator> (const __normal_iterator< _IteratorL, _Container > &__lhs, const __normal_iterator< _IteratorR, _Container > &__rhs) noexcept`
- `template<typename _Iterator, typename _Container >`
`bool __gnu_cxx::operator> (const __normal_iterator< _Iterator, _Container > &__lhs, const __normal_iterator< _Iterator, _Container > &__rhs) noexcept`
- `template<typename _IteratorL, typename _IteratorR >`
`bool std::operator> (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`
- `template<typename _Iterator >`
`bool std::operator> (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)`
- `template<typename _Iterator >`
`bool std::operator>= (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`bool std::operator>= (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >`
`bool __gnu_cxx::operator>= (const __normal_iterator< _IteratorL, _Container > &__lhs, const __normal_iterator< _IteratorR, _Container > &__rhs) noexcept`
- `template<typename _Iterator, typename _Container >`
`bool __gnu_cxx::operator>= (const __normal_iterator< _Iterator, _Container > &__lhs, const __normal_iterator< _Iterator, _Container > &__rhs) noexcept`
- `template<typename _IteratorL, typename _IteratorR >`
`bool std::operator>= (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`
- `template<typename _Iterator >`
`bool std::operator>= (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)`

5.399.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iterator>`.

This file implements `reverse_iterator`, `back_insert_iterator`, `front_insert_iterator`, `insert_iterator`, `__normal_iterator`, and their supporting functions and overloaded operators.

5.400 `stl_iterator_base_funcs.h` File Reference

Namespaces

- [std](#)

Functions

- `template<typename _InputIterator, typename _Distance>`
`void std::__advance (_InputIterator &__i, _Distance __n, input_iterator_tag)`
- `template<typename _BidirectionalIterator, typename _Distance>`
`void std::__advance (_BidirectionalIterator &__i, _Distance __n, bidirectional_iterator_tag)`
- `template<typename _RandomAccessIterator, typename _Distance>`
`void std::__advance (_RandomAccessIterator &__i, _Distance __n, random_access_iterator_tag)`
- `template<typename _InputIterator>`
`iterator_traits< _InputIterator >::difference_type std::__distance (_InputIterator __first, _InputIterator __last, input_iterator_tag)`
- `template<typename _RandomAccessIterator>`
`iterator_traits< _RandomAccessIterator >::difference_type std::__distance (_RandomAccessIterator __first, ↵
_ RandomAccessIterator __last, random_access_iterator_tag)`
- `template<typename _InputIterator, typename _Distance>`
`void std::advance (_InputIterator &__i, _Distance __n)`
- `template<typename _InputIterator>`
`iterator_traits< _InputIterator >::difference_type std::distance (_InputIterator __first, _InputIterator __last)`
- `template<typename _ForwardIterator>`
`_ForwardIterator std::next (_ForwardIterator __x, typename iterator_traits< _ForwardIterator >::difference_type
__n=1)`
- `template<typename _BidirectionalIterator>`
`_BidirectionalIterator std::prev (_BidirectionalIterator __x, typename iterator_traits< _BidirectionalIterator >↵
::difference_type __n=1)`

5.400.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iterator>`.

This file contains all of the general iterator-related utility functions, such as `distance()` and `advance()`.

5.401 `std_iterator_base_types.h` File Reference

Classes

- `class std::__has_iterator_category_helper< _Tp >`
- `struct std::bidirectional_iterator_tag`
- `struct std::forward_iterator_tag`
- `struct std::input_iterator_tag`
- `struct std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >`
- `struct std::iterator_traits< _Tp * >`
- `struct std::iterator_traits< const _Tp * >`
- `struct std::output_iterator_tag`
- `struct std::random_access_iterator_tag`

Namespaces

- `std`

Typedefs

- `template<typename _InIter >`
`using std::RequireInputIter = typename enable_if< is_convertible< typename iterator_traits< _InIter >::iterator_category, input_iterator_tag >::value >::type`

Functions

- `template<typename _Iter >`
`iterator_traits< _Iter >::iterator_category std::__iterator_category (const _Iter &)`

5.401.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iterator>`.

This file contains all of the general iterator-related utility types, such as `iterator_traits` and `struct iterator`.

5.402 `std_list.h` File Reference

Classes

- `struct std::__detail::_List_node_base`
- `class std::_List_base< _Tp, _Alloc >`
- `struct std::_List_const_iterator< _Tp >`
- `struct std::_List_iterator< _Tp >`
- `struct std::_List_node< _Tp >`
- `class std::list< _Tp, _Alloc >`

Namespaces

- `std`
- `std::__detail`

Functions

- `template<typename _Val >`
`bool std::operator!= (const _List_iterator< _Val > &__x, const _List_const_iterator< _Val > &__y) noexcept`
- `template<typename _Tp, typename _Alloc >`
`bool std::operator!= (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool std::operator< (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool std::operator<= (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`
- `template<typename _Val >`
`bool std::operator== (const _List_iterator< _Val > &__x, const _List_const_iterator< _Val > &__y) noexcept`
- `template<typename _Tp, typename _Alloc >`
`bool std::operator== (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool std::operator> (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool std::operator>= (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`
`void std::swap (list< _Tp, _Alloc > &__x, list< _Tp, _Alloc > &__y)`

5.402.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<list>`.

5.403 `std::map.h` File Reference

Classes

- class `std::map<_Key, _Tp, _Compare, _Alloc>`

Namespaces

- `std`

Functions

- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
bool std::operator!= (const map< _Key, _Tp, _Compare, _Alloc> &__x, const map< _Key, _Tp, _Compare, _Alloc> &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
bool std::operator< (const map< _Key, _Tp, _Compare, _Alloc> &__x, const map< _Key, _Tp, _Compare, _Alloc> &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
bool std::operator<= (const map< _Key, _Tp, _Compare, _Alloc> &__x, const map< _Key, _Tp, _Compare, _Alloc> &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
bool std::operator== (const map< _Key, _Tp, _Compare, _Alloc> &__x, const map< _Key, _Tp, _Compare, _Alloc> &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
bool std::operator> (const map< _Key, _Tp, _Compare, _Alloc> &__x, const map< _Key, _Tp, _Compare, _Alloc> &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
bool std::operator>= (const map< _Key, _Tp, _Compare, _Alloc> &__x, const map< _Key, _Tp, _Compare, _Alloc> &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
void std::swap (map< _Key, _Tp, _Compare, _Alloc> &__x, map< _Key, _Tp, _Compare, _Alloc> &__y)`

5.403.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<map>`.

5.404 `std::multimap.h` File Reference

Classes

- class `std::multimap<_Key, _Tp, _Compare, _Alloc>`

Namespaces

- [std](#)

Functions

- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`
`bool std::operator!= (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`
`bool std::operator< (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`
`bool std::operator<= (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`
`bool std::operator== (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`
`bool std::operator> (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`
`bool std::operator>= (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`
`void std::swap (multimap< _Key, _Tp, _Compare, _Alloc > &__x, multimap< _Key, _Tp, _Compare, _Alloc > &__y)`

5.404.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<map>`.

5.405 `std_multiset.h` File Reference

Classes

- class [std::multiset< _Key, _Compare, _Alloc >](#)

Namespaces

- [std](#)

Functions

- `template<typename _Key, typename _Compare, typename _Alloc >`
`bool std::operator!= (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`
`bool std::operator< (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`
`bool std::operator<= (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`
`bool std::operator== (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`
`bool std::operator> (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`
`bool std::operator>= (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`
`void std::swap (multiset< _Key, _Compare, _Alloc > &__x, multiset< _Key, _Compare, _Alloc > &__y)`

5.405.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<set>`.

5.406 `stl_numeric.h` File Reference

Namespaces

- [std](#)

Functions

- `template<typename _InputIterator, typename _Tp >`
`_Tp std::accumulate (_InputIterator __first, _InputIterator __last, _Tp __init)`
- `template<typename _InputIterator, typename _Tp, typename _BinaryOperation >`
`_Tp std::accumulate (_InputIterator __first, _InputIterator __last, _Tp __init, _BinaryOperation __binary_op)`
- `template<typename _InputIterator, typename _OutputIterator >`
`_OutputIterator std::adjacent_difference (_InputIterator __first, _InputIterator __last, _OutputIterator __result)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation >`
`_OutputIterator std::adjacent_difference (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryOperation __binary_op)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Tp >`
`_Tp std::inner_product (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Tp __init)`

- `template<typename _InputIterator1, typename _InputIterator2, typename _Tp, typename _BinaryOperation1, typename _BinaryOperation2>`
`>`
`_Tp std::inner_product` (`_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Tp __init, _BinaryOperation1 __binary_op1, _BinaryOperation2 __binary_op2`)
- `template<typename _ForwardIterator, typename _Tp>`
`void std::iota` (`_ForwardIterator __first, _ForwardIterator __last, _Tp __value`)
- `template<typename _InputIterator, typename _OutputIterator>`
`_OutputIterator std::partial_sum` (`_InputIterator __first, _InputIterator __last, _OutputIterator __result`)
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation>`
`_OutputIterator std::partial_sum` (`_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryOperation __binary_op`)

5.406.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<numeric>`.

5.407 `std::pair.h` File Reference

Classes

- struct `std::pair<_T1, _T2>`
- struct `std::piecewise_construct_t`

Namespaces

- `std`

Functions

- `template<class _T1, class _T2>`
`constexpr pair< typename __decay_and_strip< _T1 >::__type, typename __decay_and_strip< _T2 >::__type`
`> std::make_pair` (`_T1 &&__x, _T2 &&__y`)
- `template<class _T1, class _T2>`
`constexpr bool std::operator!=` (`const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y`)
- `template<class _T1, class _T2>`
`constexpr bool std::operator<` (`const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y`)
- `template<class _T1, class _T2>`
`constexpr bool std::operator<=` (`const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y`)
- `template<class _T1, class _T2>`
`constexpr bool std::operator==` (`const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y`)
- `template<class _T1, class _T2>`
`constexpr bool std::operator>` (`const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y`)
- `template<class _T1, class _T2>`
`constexpr bool std::operator>=` (`const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y`)
- `template<class _T1, class _T2>`
`void std::swap` (`pair< _T1, _T2 > &__x, pair< _T1, _T2 > &__y`) `noexcept(noexcept(__x.swap(__y)))`

Variables

- `constexpr piecewise_construct_t` [std::piecewise_construct](#)

5.407.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<utility>`.

5.408 `std_queue.h` File Reference

Classes

- class [std::priority_queue<_Tp, _Sequence, _Compare>](#)
- class [std::queue<_Tp, _Sequence>](#)

Namespaces

- [std](#)

Functions

- `template<typename _Tp, typename _Seq>`
`bool` [std::operator!=](#) (const queue<_Tp, _Seq> &__x, const queue<_Tp, _Seq> &__y)
- `template<typename _Tp, typename _Seq>`
`bool` [std::operator<](#) (const queue<_Tp, _Seq> &__x, const queue<_Tp, _Seq> &__y)
- `template<typename _Tp, typename _Seq>`
`bool` [std::operator<=](#) (const queue<_Tp, _Seq> &__x, const queue<_Tp, _Seq> &__y)
- `template<typename _Tp, typename _Seq>`
`bool` [std::operator==](#) (const queue<_Tp, _Seq> &__x, const queue<_Tp, _Seq> &__y)
- `template<typename _Tp, typename _Seq>`
`bool` [std::operator>](#) (const queue<_Tp, _Seq> &__x, const queue<_Tp, _Seq> &__y)
- `template<typename _Tp, typename _Seq>`
`bool` [std::operator>=](#) (const queue<_Tp, _Seq> &__x, const queue<_Tp, _Seq> &__y)
- `template<typename _Tp, typename _Seq>`
`void` [std::swap](#) (queue<_Tp, _Seq> &__x, queue<_Tp, _Seq> &__y) noexcept(noexcept(__x.swap(__y)))
- `template<typename _Tp, typename _Sequence, typename _Compare>`
`void` [std::swap](#) (priority_queue<_Tp, _Sequence, _Compare> &__x, priority_queue<_Tp, _Sequence, _Compare> &__y) noexcept(noexcept(__x.swap(__y)))

5.408.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<queue>`.

5.409 `std_raw_storage_iter.h` File Reference

Classes

- class `std::raw_storage_iterator<_OutputIterator, _Tp>`

Namespaces

- `std`

5.409.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

5.410 `std_relops.h` File Reference

Namespaces

- `std`
- `std::rel_ops`

Functions

- `template<class _Tp>`
`bool std::rel_ops::operator!= (const _Tp &__x, const _Tp &__y)`
- `template<class _Tp>`
`bool std::rel_ops::operator<= (const _Tp &__x, const _Tp &__y)`
- `template<class _Tp>`
`bool std::rel_ops::operator> (const _Tp &__x, const _Tp &__y)`
- `template<class _Tp>`
`bool std::rel_ops::operator>= (const _Tp &__x, const _Tp &__y)`

5.410.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<utility>`.

Inclusion of this file has been removed from all of the other STL headers for safety reasons, except `std_utility.h`. For more information, see the thread of about twenty messages starting with <http://gcc.gnu.org/ml/libstdc++/2001-01/msg00223.html>, or http://gcc.gnu.org/onlinedocs/libstdc++/faq.html#faq.ambiguous_overloads

Short summary: the `rel_ops` operators should be avoided for the present.

5.411 `std_set.h` File Reference

Classes

- class `std::set<_Key, _Compare, _Alloc>`

Namespaces

- `std`

Functions

- `template<typename _Key, typename _Compare, typename _Alloc>`
`bool std::operator!= (const set<_Key, _Compare, _Alloc> &__x, const set<_Key, _Compare, _Alloc> &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc>`
`bool std::operator< (const set<_Key, _Compare, _Alloc> &__x, const set<_Key, _Compare, _Alloc> &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc>`
`bool std::operator<= (const set<_Key, _Compare, _Alloc> &__x, const set<_Key, _Compare, _Alloc> &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc>`
`bool std::operator== (const set<_Key, _Compare, _Alloc> &__x, const set<_Key, _Compare, _Alloc> &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc>`
`bool std::operator> (const set<_Key, _Compare, _Alloc> &__x, const set<_Key, _Compare, _Alloc> &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc>`
`bool std::operator>= (const set<_Key, _Compare, _Alloc> &__x, const set<_Key, _Compare, _Alloc> &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc>`
`void std::swap (set<_Key, _Compare, _Alloc> &__x, set<_Key, _Compare, _Alloc> &__y)`

5.411.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<set>`.

5.412 `std_stack.h` File Reference

Classes

- class `std::stack<_Tp, _Sequence>`

Namespaces

- `std`

Functions

- `template<typename _Tp, typename _Seq >`
`bool std::operator!= (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq >`
`bool std::operator< (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq >`
`bool std::operator<= (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq >`
`bool std::operator== (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq >`
`bool std::operator> (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq >`
`bool std::operator>= (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq >`
`void std::swap (stack< _Tp, _Seq > &__x, stack< _Tp, _Seq > &__y) noexcept(noexcept(__x.swap(__y)))`

5.412.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<stack>`.

5.413 `std_tempbuf.h` File Reference

Classes

- class `std::_Temporary_buffer<_ForwardIterator, _Tp >`

Namespaces

- `std`

Functions

- `template<typename _Pointer, typename _ForwardIterator >`
`void std::__uninitialized_construct_buf (_Pointer __first, _Pointer __last, _ForwardIterator __seed)`
- `template<typename _Tp >`
`pair< _Tp *, ptrdiff_t > std::get_temporary_buffer (ptrdiff_t __len) noexcept`
- `template<typename _Tp >`
`void std::return_temporary_buffer (_Tp *__p)`

5.413.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

5.414 `std::tree.h` File Reference

Namespaces

- [std](#)

Enumerations

- enum `_Rb_tree_color` { `_S_red`, `_S_black` }

Functions

- unsigned int `std::Rb_tree_black_count` (const `_Rb_tree_node_base *`__node, const `_Rb_tree_node_base *`__root) throw ()
- `_Rb_tree_node_base *` `std::Rb_tree_decrement` (`_Rb_tree_node_base *`__x) throw ()
- const `_Rb_tree_node_base *` `std::Rb_tree_decrement` (const `_Rb_tree_node_base *`__x) throw ()
- `_Rb_tree_node_base *` `std::Rb_tree_increment` (`_Rb_tree_node_base *`__x) throw ()
- const `_Rb_tree_node_base *` `std::Rb_tree_increment` (const `_Rb_tree_node_base *`__x) throw ()
- void `std::Rb_tree_insert_and_rebalance` (const bool __insert_left, `_Rb_tree_node_base *`__x, `_Rb_tree_node_base *`__p, `_Rb_tree_node_base &`__header) throw ()
- `_Rb_tree_node_base *` `std::Rb_tree_rebalance_for_erase` (`_Rb_tree_node_base *`const __z, `_Rb_tree_node_base &`__header) throw ()
- template<typename `_Val` >
bool `std::operator!=` (const `_Rb_tree_iterator`< `_Val` > &__x, const `_Rb_tree_const_iterator`< `_Val` > &__y) noexcept
- template<typename `_Key`, typename `_Val`, typename `_KeyOfValue`, typename `_Compare`, typename `_Alloc` >
bool `std::operator!=` (const `_Rb_tree`< `_Key`, `_Val`, `_KeyOfValue`, `_Compare`, `_Alloc` > &__x, const `_Rb_tree`< `_Key`, `_Val`, `_KeyOfValue`, `_Compare`, `_Alloc` > &__y)
- template<typename `_Key`, typename `_Val`, typename `_KeyOfValue`, typename `_Compare`, typename `_Alloc` >
bool `std::operator<` (const `_Rb_tree`< `_Key`, `_Val`, `_KeyOfValue`, `_Compare`, `_Alloc` > &__x, const `_Rb_tree`< `_Key`, `_Val`, `_KeyOfValue`, `_Compare`, `_Alloc` > &__y)
- template<typename `_Key`, typename `_Val`, typename `_KeyOfValue`, typename `_Compare`, typename `_Alloc` >
bool `std::operator<=` (const `_Rb_tree`< `_Key`, `_Val`, `_KeyOfValue`, `_Compare`, `_Alloc` > &__x, const `_Rb_tree`< `_Key`, `_Val`, `_KeyOfValue`, `_Compare`, `_Alloc` > &__y)
- template<typename `_Val` >
bool `std::operator==` (const `_Rb_tree_iterator`< `_Val` > &__x, const `_Rb_tree_const_iterator`< `_Val` > &__y) noexcept
- template<typename `_Key`, typename `_Val`, typename `_KeyOfValue`, typename `_Compare`, typename `_Alloc` >
bool `std::operator==` (const `_Rb_tree`< `_Key`, `_Val`, `_KeyOfValue`, `_Compare`, `_Alloc` > &__x, const `_Rb_tree`< `_Key`, `_Val`, `_KeyOfValue`, `_Compare`, `_Alloc` > &__y)
- template<typename `_Key`, typename `_Val`, typename `_KeyOfValue`, typename `_Compare`, typename `_Alloc` >
bool `std::operator>` (const `_Rb_tree`< `_Key`, `_Val`, `_KeyOfValue`, `_Compare`, `_Alloc` > &__x, const `_Rb_tree`< `_Key`, `_Val`, `_KeyOfValue`, `_Compare`, `_Alloc` > &__y)
- template<typename `_Key`, typename `_Val`, typename `_KeyOfValue`, typename `_Compare`, typename `_Alloc` >
bool `std::operator>=` (const `_Rb_tree`< `_Key`, `_Val`, `_KeyOfValue`, `_Compare`, `_Alloc` > &__x, const `_Rb_tree`< `_Key`, `_Val`, `_KeyOfValue`, `_Compare`, `_Alloc` > &__y)
- template<typename `_Key`, typename `_Val`, typename `_KeyOfValue`, typename `_Compare`, typename `_Alloc` >
void `std::swap` (`_Rb_tree`< `_Key`, `_Val`, `_KeyOfValue`, `_Compare`, `_Alloc` > &__x, `_Rb_tree`< `_Key`, `_Val`, `_KeyOfValue`, `_Compare`, `_Alloc` > &__y)

5.414.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<map>` or `<set>`.

5.415 `std_uninitialized.h` File Reference

Namespaces

- [std](#)

Functions

- `template<typename _InputIterator, typename _ForwardIterator, typename _Allocator >`
`_ForwardIterator std::uninitialized_copy_a (_InputIterator __first, _InputIterator __last, _ForwardIterator __result, _Allocator &__alloc)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _Tp >`
`_ForwardIterator std::uninitialized_copy_a (_InputIterator __first, _InputIterator __last, _ForwardIterator __result, allocator<_Tp> &)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _ForwardIterator, typename _Allocator >`
`_ForwardIterator std::uninitialized_copy_move (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _ForwardIterator __result, _Allocator &__alloc)`
- `template<typename _InputIterator, typename _Size, typename _ForwardIterator >`
`_ForwardIterator std::uninitialized_copy_n (_InputIterator __first, _Size __n, _ForwardIterator __result, input_iterator_tag)`
- `template<typename _RandomAccessIterator, typename _Size, typename _ForwardIterator >`
`_ForwardIterator std::uninitialized_copy_n (_RandomAccessIterator __first, _Size __n, _ForwardIterator __result, random_access_iterator_tag)`
- `template<typename _ForwardIterator >`
`void std::uninitialized_default (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Allocator >`
`void std::uninitialized_default_a (_ForwardIterator __first, _ForwardIterator __last, _Allocator &__alloc)`
- `template<typename _ForwardIterator, typename _Tp >`
`void std::uninitialized_default_a (_ForwardIterator __first, _ForwardIterator __last, allocator<_Tp> &)`
- `template<typename _ForwardIterator, typename _Size >`
`void std::uninitialized_default_n (_ForwardIterator __first, _Size __n)`
- `template<typename _ForwardIterator, typename _Size, typename _Allocator >`
`void std::uninitialized_default_n_a (_ForwardIterator __first, _Size __n, _Allocator &__alloc)`
- `template<typename _ForwardIterator, typename _Size, typename _Tp >`
`void std::uninitialized_default_n_a (_ForwardIterator __first, _Size __n, allocator<_Tp> &)`
- `template<typename _ForwardIterator, typename _Tp, typename _Allocator >`
`void std::uninitialized_fill_a (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__x, _Allocator &__alloc)`
- `template<typename _ForwardIterator, typename _Tp, typename _Tp2 >`
`void std::uninitialized_fill_a (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__x, allocator<_Tp2> &)`
- `template<typename _ForwardIterator, typename _Tp, typename _InputIterator, typename _Allocator >`
`_ForwardIterator std::uninitialized_fill_move (_ForwardIterator __result, _ForwardIterator __mid, const _Tp &__x, _InputIterator __first, _InputIterator __last, _Allocator &__alloc)`

- `template<typename _ForwardIterator, typename _Size, typename _Tp, typename _Allocator >`
`void std::__uninitialized_fill_n_a (_ForwardIterator __first, _Size __n, const _Tp &__x, _Allocator &__alloc)`
- `template<typename _ForwardIterator, typename _Size, typename _Tp, typename _Tp2 >`
`void std::__uninitialized_fill_n_a (_ForwardIterator __first, _Size __n, const _Tp &__x, allocator<_Tp2> &__a)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _Allocator >`
`_ForwardIterator std::__uninitialized_move_a (_InputIterator __first, _InputIterator __last, _ForwardIterator __result, _Allocator &__alloc)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _ForwardIterator, typename _Allocator >`
`_ForwardIterator std::__uninitialized_move_copy (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _ForwardIterator __result, _Allocator &__alloc)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _Tp, typename _Allocator >`
`void std::__uninitialized_move_fill (_InputIterator __first1, _InputIterator __last1, _ForwardIterator __first2, _ForwardIterator __last2, const _Tp &__x, _Allocator &__alloc)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _Allocator >`
`_ForwardIterator std::__uninitialized_move_if_noexcept_a (_InputIterator __first, _InputIterator __last, _ForwardIterator __result, _Allocator &__alloc)`
- `template<typename _InputIterator, typename _ForwardIterator >`
`_ForwardIterator std::uninitialized_copy (_InputIterator __first, _InputIterator __last, _ForwardIterator __result)`
- `template<typename _InputIterator, typename _Size, typename _ForwardIterator >`
`_ForwardIterator std::uninitialized_copy_n (_InputIterator __first, _Size __n, _ForwardIterator __result)`
- `template<typename _ForwardIterator, typename _Tp >`
`void std::uninitialized_fill (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__x)`
- `template<typename _ForwardIterator, typename _Size, typename _Tp >`
`void std::uninitialized_fill_n (_ForwardIterator __first, _Size __n, const _Tp &__x)`

5.415.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

5.416 `std::vector.h` File Reference

Classes

- struct `std::_Vector_base<_Tp, _Alloc>`
- class `std::vector<_Tp, _Alloc>`

Namespaces

- `std`

Functions

- `template<typename _Tp, typename _Alloc >`
`bool std::operator!= (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool std::operator< (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool std::operator<= (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool std::operator== (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool std::operator> (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`
`bool std::operator>= (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`
`void std::swap (vector< _Tp, _Alloc > &__x, vector< _Tp, _Alloc > &__y)`

5.416.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<vector>`.

5.417 `stream_iterator.h` File Reference

Classes

- class `std::istream_iterator< _Tp, _CharT, _Traits, _Dist >`
- class `std::ostream_iterator< _Tp, _CharT, _Traits >`

Namespaces

- `std`

Functions

- `template<class _Tp, class _CharT, class _Traits, class _Dist >`
`bool std::operator!= (const istream_iterator< _Tp, _CharT, _Traits, _Dist > &__x, const istream_iterator< _Tp, _CharT, _Traits, _Dist > &__y)`
- `template<typename _Tp, typename _CharT, typename _Traits, typename _Dist >`
`bool std::operator== (const istream_iterator< _Tp, _CharT, _Traits, _Dist > &__x, const istream_iterator< _Tp, _CharT, _Traits, _Dist > &__y)`

5.417.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iterator>`.

5.418 `streambuf_iterator.h` File Reference

Classes

- class [std::istreambuf_iterator<_CharT, _Traits>](#)
- class [std::ostreambuf_iterator<_CharT, _Traits>](#)

Namespaces

- [std](#)

Functions

- `template<bool _IsMove, typename _CharT >`
`__gnu_cxx::__enable_if<__is_char<_CharT>::__value, ostreambuf_iterator<_CharT>>::__type std::__↵`
`copy_move_a2 (_CharT * __first, _CharT * __last, ostreambuf_iterator<_CharT> __result)`
- `template<bool _IsMove, typename _CharT >`
`__gnu_cxx::__enable_if<__is_char<_CharT>::__value, ostreambuf_iterator<_CharT>>::__type std::__↵`
`copy_move_a2 (const _CharT * __first, const _CharT * __last, ostreambuf_iterator<_CharT> __result)`
- `template<bool _IsMove, typename _CharT >`
`__gnu_cxx::__enable_if<__is_char<_CharT>::__value, _CharT * >::__type std::__copy_move_a2`
`(istreambuf_iterator<_CharT> __first, istreambuf_iterator<_CharT> __last, _CharT * __result)`
- `template<typename _CharT >`
`__gnu_cxx::__enable_if<__is_char<_CharT>::__value, ostreambuf_iterator<_CharT>>::__type std::__copy`
`(istreambuf_iterator<_CharT> __first, istreambuf_iterator<_CharT> __last, ostreambuf_iterator<_CharT>`
`__result)`
- `template<typename _CharT >`
`__gnu_cxx::__enable_if<__is_char<_CharT>::__value, istreambuf_iterator<_CharT>>::__type std::__find`
`(istreambuf_iterator<_CharT> __first, istreambuf_iterator<_CharT> __last, const _CharT & __val)`
- `template<typename _CharT, typename _Traits >`
`bool std::__operator!= (const istreambuf_iterator<_CharT, _Traits> & __a, const istreambuf_iterator<_CharT,`
`_Traits> & __b)`
- `template<typename _CharT, typename _Traits >`
`bool std::__operator== (const istreambuf_iterator<_CharT, _Traits> & __a, const istreambuf_iterator<_CharT,`
`_Traits> & __b)`

5.418.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iterator>`.

5.419 `string_conversions.h` File Reference

Namespaces

- [__gnu_cxx](#)

Functions

- `template<typename _TRet , typename _Ret = _TRet, typename _CharT , typename... _Base>
_Ret gnu_cxx::__stoa (_TRet(*__convf)(const _CharT *, _CharT **, _Base...), const char *__name, const
_CharT *__str, std::size_t *__idx, _Base...__base)`
- `template<typename _String , typename _CharT = typename _String::value_type>
_String gnu_cxx::__to_xstring (int(*__convf)(_CharT *, std::size_t, const _CharT *, __builtin_va_list), std::size_t __n, const _CharT *__fmt,...)`

5.419.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

5.420 stringfwd.h File Reference

Classes

- class `std::basic_string< _CharT, _Traits, _Alloc >`
- struct `std::char_traits< _CharT >`

Namespaces

- `std`

Typedefs

- `typedef basic_string< char > std::string`
- `typedef basic_string< char16_t > std::u16string`
- `typedef basic_string< char32_t > std::u32string`
- `typedef basic_string< wchar_t > std::wstring`

5.420.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<string>`.

5.421 synth_access_traits.hpp File Reference

Classes

- struct `__gnu_pbds::detail::synth_access_traits< Type_Traits, Set, _ATraits >`

Namespaces

- [__gnu_pbds](#)

Macros

- `#define PB_DS_SYNTH_E_ACCESS_TRAITS_C_DEC`
- `#define PB_DS_SYNTH_E_ACCESS_TRAITS_T_DEC`

5.421.1 Detailed Description

Contains an implementation class for a patricia tree.

5.422 tag_and_trait.hpp File Reference

Classes

- struct [__gnu_pbds::associative_tag](#)
- struct [__gnu_pbds::basic_branch_tag](#)
- struct [__gnu_pbds::basic_hash_tag](#)
- struct [__gnu_pbds::basic_invalidation_guarantee](#)
- struct [__gnu_pbds::binary_heap_tag](#)
- struct [__gnu_pbds::binomial_heap_tag](#)
- struct [__gnu_pbds::cc_hash_tag](#)
- struct [__gnu_pbds::container_tag](#)
- struct [__gnu_pbds::container_traits< Cntnr >](#)
- struct [__gnu_pbds::container_traits_base< _Tag >](#)
- struct [__gnu_pbds::container_traits_base< binary_heap_tag >](#)
- struct [__gnu_pbds::container_traits_base< binomial_heap_tag >](#)
- struct [__gnu_pbds::container_traits_base< cc_hash_tag >](#)
- struct [__gnu_pbds::container_traits_base< gp_hash_tag >](#)
- struct [__gnu_pbds::container_traits_base< list_update_tag >](#)
- struct [__gnu_pbds::container_traits_base< ov_tree_tag >](#)
- struct [__gnu_pbds::container_traits_base< pairing_heap_tag >](#)
- struct [__gnu_pbds::container_traits_base< pat_trie_tag >](#)
- struct [__gnu_pbds::container_traits_base< rb_tree_tag >](#)
- struct [__gnu_pbds::container_traits_base< rc_binomial_heap_tag >](#)
- struct [__gnu_pbds::container_traits_base< splay_tree_tag >](#)
- struct [__gnu_pbds::container_traits_base< thin_heap_tag >](#)
- struct [__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, Tag, Policy_TI >](#)
- struct [__gnu_pbds::gp_hash_tag](#)
- struct [__gnu_pbds::list_update_tag](#)
- struct [__gnu_pbds::null_node_update< _Tp1, _Tp2, _Tp3, _Tp4 >](#)
- struct [__gnu_pbds::null_type](#)
- struct [__gnu_pbds::ov_tree_tag](#)
- struct [__gnu_pbds::pairing_heap_tag](#)
- struct [__gnu_pbds::pat_trie_tag](#)

- struct [__gnu_pbds::point_invalidation_guarantee](#)
- struct [__gnu_pbds::priority_queue_tag](#)
- struct [__gnu_pbds::range_invalidation_guarantee](#)
- struct [__gnu_pbds::rb_tree_tag](#)
- struct [__gnu_pbds::rc_binomial_heap_tag](#)
- struct [__gnu_pbds::sequence_tag](#)
- struct [__gnu_pbds::splay_tree_tag](#)
- struct [__gnu_pbds::string_tag](#)
- struct [__gnu_pbds::thin_heap_tag](#)
- struct [__gnu_pbds::tree_tag](#)
- struct [__gnu_pbds::trie_tag](#)
- struct [__gnu_pbds::trivial_iterator_tag](#)

Namespaces

- [__gnu_pbds](#)

Typedefs

- typedef void [__gnu_pbds::trivial_iterator_difference_type](#)

5.422.1 Detailed Description

Contains tags and traits, e.g., ones describing underlying data structures.

5.423 tags.h File Reference

Classes

- struct [__gnu_parallel::balanced_quicksort_tag](#)
- struct [__gnu_parallel::balanced_tag](#)
- struct [__gnu_parallel::constant_size_blocks_tag](#)
- struct [__gnu_parallel::default_parallel_tag](#)
- struct [__gnu_parallel::equal_split_tag](#)
- struct [__gnu_parallel::exact_tag](#)
- struct [__gnu_parallel::find_tag](#)
- struct [__gnu_parallel::growing_blocks_tag](#)
- struct [__gnu_parallel::multiway_mergesort_exact_tag](#)
- struct [__gnu_parallel::multiway_mergesort_sampling_tag](#)
- struct [__gnu_parallel::multiway_mergesort_tag](#)
- struct [__gnu_parallel::omp_loop_static_tag](#)
- struct [__gnu_parallel::omp_loop_tag](#)
- struct [__gnu_parallel::parallel_tag](#)
- struct [__gnu_parallel::quicksort_tag](#)
- struct [__gnu_parallel::sampling_tag](#)
- struct [__gnu_parallel::sequential_tag](#)
- struct [__gnu_parallel::unbalanced_tag](#)

Namespaces

- [__gnu_parallel](#)

5.423.1 Detailed Description

Tags for compile-time selection. This file is a GNU parallel extension to the Standard C++ Library.

5.424 `tgmath.h` File Reference

Macros

- `#define _GLIBCXX_TGMATH_H`

5.424.1 Detailed Description

This is a Standard C++ Library header.

5.425 `thin_heap_.hpp` File Reference

Classes

- class [__gnu_pbds::detail::thin_heap< Value_Type, Cmp_Fn, _Alloc >](#)

Namespaces

- [__gnu_pbds](#)

Macros

- `#define PB_DS_ASSERT_AUX_NULL(X)`
- `#define PB_DS_ASSERT_NODE_CONSISTENT(_Node, _Bool)`
- `#define PB_DS_BASE_T_P`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`

Enumerations

- enum { `num_distinct_rank_bounds` }

Variables

- static const std::size_t `__gnu_pbds::detail::g_a_rank_bounds` [num_distinct_rank_bounds]

5.425.1 Detailed Description

Contains an implementation class for a thin heap.

5.426 `throw_allocator.h` File Reference

Classes

- struct `__gnu_cxx::annotate_base`
- struct `__gnu_cxx::condition_base`
- struct `__gnu_cxx::forced_error`
- struct `__gnu_cxx::limit_condition`
- struct `__gnu_cxx::limit_condition::always_adjustor`
- struct `__gnu_cxx::limit_condition::limit_adjustor`
- struct `__gnu_cxx::limit_condition::never_adjustor`
- struct `__gnu_cxx::random_condition`
- struct `__gnu_cxx::random_condition::always_adjustor`
- struct `__gnu_cxx::random_condition::group_adjustor`
- struct `__gnu_cxx::random_condition::never_adjustor`
- class `__gnu_cxx::throw_allocator_base< _Tp, _Cond >`
- struct `__gnu_cxx::throw_allocator_limit< _Tp >`
- struct `__gnu_cxx::throw_allocator_random< _Tp >`
- struct `__gnu_cxx::throw_value_base< _Cond >`
- struct `__gnu_cxx::throw_value_limit`
- struct `__gnu_cxx::throw_value_random`
- struct `std::hash< __gnu_cxx::throw_value_limit >`
- struct `std::hash< __gnu_cxx::throw_value_random >`

Namespaces

- `__gnu_cxx`
- `std`

Functions

- void **__gnu_cxx::__throw_forced_error** ()
- template<typename _Tp, typename _Cond >
bool **__gnu_cxx::operator!=** (const throw_allocator_base< _Tp, _Cond > &, const throw_allocator_base< _Tp, _Cond > &)
- template<typename _Cond >
throw_value_base< _Cond > **__gnu_cxx::operator*** (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)
- template<typename _Cond >
throw_value_base< _Cond > **__gnu_cxx::operator+** (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)
- template<typename _Cond >
throw_value_base< _Cond > **__gnu_cxx::operator-** (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)
- template<typename _Cond >
bool **__gnu_cxx::operator<** (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)
- std::ostream & **__gnu_cxx::operator<<** (std::ostream &os, const annotate_base &__b)
- template<typename _Cond >
bool **__gnu_cxx::operator==** (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)
- template<typename _Tp, typename _Cond >
bool **__gnu_cxx::operator==** (const throw_allocator_base< _Tp, _Cond > &, const throw_allocator_base< _Tp, _Cond > &)
- template<typename _Cond >
void **__gnu_cxx::swap** (throw_value_base< _Cond > &__a, throw_value_base< _Cond > &__b)

5.426.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Contains two exception-generating types (throw_value, throw_allocator) intended to be used as value and allocator types while testing exception safety in templated containers and algorithms. The allocator has additional log and debug features. The exception generated is of type forced_exception_error.

5.427 time_members.h File Reference

Namespaces

- [std](#)

5.427.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <locale>.

5.428 `trace_fn_imps.hpp` File Reference

5.428.1 Detailed Description

Contains an implementation class for a `binary_heap`.

5.429 `trace_fn_imps.hpp` File Reference

5.429.1 Detailed Description

Contains implementations of `cc_ht_map_`'s trace-mode functions.

5.430 `trace_fn_imps.hpp` File Reference

5.430.1 Detailed Description

Contains implementations of `gp_ht_map_`'s trace-mode functions.

5.431 `trace_fn_imps.hpp` File Reference

5.431.1 Detailed Description

Contains an implementation class for `left_child_next_sibling_heap_`.

5.432 `trace_fn_imps.hpp` File Reference

5.432.1 Detailed Description

Contains implementations of `lu_map_`.

5.433 `trace_fn_imps.hpp` File Reference

5.433.1 Detailed Description

Contains an implementation class for `pat_trie_`.

5.434 `trace_fn_imps.hpp` File Reference

5.434.1 Detailed Description

Contains an implementation for `rc_binomial_heap_`.

5.435 `trace_fn_imps.hpp` File Reference

5.435.1 Detailed Description

Contains an implementation class for `left_child_next_sibling_heap_`.

5.436 `traits.hpp` File Reference

Classes

- struct [__gnu_pbds::detail::bin_search_tree_traits< Key, Mapped, Cmp_Fn, Node_Update, Node, _Alloc >](#)
- struct [__gnu_pbds::detail::bin_search_tree_traits< Key, null_type, Cmp_Fn, Node_Update, Node, _Alloc >](#)

Namespaces

- [__gnu_pbds](#)

5.436.1 Detailed Description

Contains an implementation for `bin_search_tree_`.

5.437 `traits.hpp` File Reference

Classes

- struct [__gnu_pbds::detail::tree_traits< Key, Data, Cmp_Fn, Node_Update, Tag, _Alloc >](#)
- struct [__gnu_pbds::detail::trie_traits< Key, Data, _ATraits, Node_Update, Tag, _Alloc >](#)

Namespaces

- [__gnu_pbds](#)

Macros

- `#define PB_DS_DEBUG_VERIFY(_Cond)`

5.437.1 Detailed Description

Contains an implementation class for tree-like classes.

5.438 traits.hpp File Reference

Classes

- struct [__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >](#)
- struct [__gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >](#)

Namespaces

- [__gnu_pbds](#)

5.438.1 Detailed Description

Contains an implementation class for ov_tree_.

5.439 traits.hpp File Reference

Classes

- struct [__gnu_pbds::detail::trie_traits< Key, Mapped, _ATraits, Node_Update, pat_trie_tag, _Alloc >](#)
- struct [__gnu_pbds::detail::trie_traits< Key, null_type, _ATraits, Node_Update, pat_trie_tag, _Alloc >](#)

Namespaces

- [__gnu_pbds](#)

5.439.1 Detailed Description

Contains an implementation class for pat_trie_.

5.440 traits.hpp File Reference

Classes

- struct [__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >](#)
- struct [__gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >](#)

Namespaces

- [__gnu_pbds](#)

5.440.1 Detailed Description

Contains an implementation for `rb_tree_`.

5.441 `traits.hpp` File Reference

Classes

- struct [__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >](#)
- struct [__gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >](#)

Namespaces

- [__gnu_pbds](#)

5.441.1 Detailed Description

Contains an implementation for `splay_tree_`.

5.442 `tree_policy.hpp` File Reference

Classes

- class [__gnu_pbds::tree_order_statistics_node_update< Node_Cltr, Node_Itr, Cmp_Fn, _Alloc >](#)

Namespaces

- [__gnu_pbds](#)

Macros

- `#define PB_DS_BRANCH_POLICY_BASE`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`

5.442.1 Detailed Description

Contains tree-related policies.

5.443 tree_trace_base.hpp File Reference

5.443.1 Detailed Description

Contains tree-related policies.

5.444 trie_policy.hpp File Reference

Classes

- class [__gnu_pbds::trie_order_statistics_node_update](#)< Node_Cltr, Node_Itr, _ATraits, _Alloc >
- class [__gnu_pbds::trie_prefix_search_node_update](#)< Node_Cltr, Node_Itr, _ATraits, _Alloc >
- struct [__gnu_pbds::trie_string_access_traits](#)< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >

Namespaces

- [__gnu_pbds](#)

Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_TRIE_POLICY_BASE`

5.444.1 Detailed Description

Contains trie-related policies.

5.445 trie_policy_base.hpp File Reference

Classes

- class [__gnu_pbds::detail::trie_policy_base](#)< Node_Cltr, Node_Itr, _ATraits, _Alloc >

Namespaces

- [__gnu_pbds](#)

Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`

5.445.1 Detailed Description

Contains an implementation of `trie_policy_base`.

5.446 `trie_string_access_traits_imp.hpp` File Reference

5.446.1 Detailed Description

Contains a policy for extracting character positions from a string for a vector-based PATRICIA tree

5.447 `type_traits.h` File Reference

Namespaces

- [`__gnu_cxx`](#)

Functions

- `template<typename _Type >`
`bool __gnu_cxx::__is_null_pointer (_Type *__ptr)`
- `template<typename _Type >`
`bool __gnu_cxx::__is_null_pointer (_Type)`
- `bool __gnu_cxx::__is_null_pointer (std::nullptr_t)`

5.447.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

5.448 `type_utils.hpp` File Reference

Namespaces

- [`__gnu_pbds`](#)

Macros

- `#define PB_DS_STATIC_ASSERT(UNIQUE, E)`

Typedefs

- typedef std::tr1::integral_constant< int, 0 > **__gnu_pbds::detail::false_type**
- typedef std::tr1::integral_constant< int, 1 > **__gnu_pbds::detail::true_type**

5.448.1 Detailed Description

Contains utilities for handling types. All of these classes are based on Modern C++ by Andrei Alexandrescu.

5.449 typelist.h File Reference

Namespaces

- [__gnu_cxx](#)
- [__gnu_cxx::typelist](#)

Macros

- #define **_GLIBCXX_TYPELIST_CHAIN1**(X0)
- #define **_GLIBCXX_TYPELIST_CHAIN10**(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9)
- #define **_GLIBCXX_TYPELIST_CHAIN11**(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10)
- #define **_GLIBCXX_TYPELIST_CHAIN12**(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11)
- #define **_GLIBCXX_TYPELIST_CHAIN13**(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12)
- #define **_GLIBCXX_TYPELIST_CHAIN14**(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13)
- #define **_GLIBCXX_TYPELIST_CHAIN15**(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14)
- #define **_GLIBCXX_TYPELIST_CHAIN16**(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X15)
- #define **_GLIBCXX_TYPELIST_CHAIN17**(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X15, X16)
- #define **_GLIBCXX_TYPELIST_CHAIN18**(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X15, X16, X17)
- #define **_GLIBCXX_TYPELIST_CHAIN19**(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X15, X16, X17, X18)
- #define **_GLIBCXX_TYPELIST_CHAIN2**(X0, X1)
- #define **_GLIBCXX_TYPELIST_CHAIN20**(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X15, X16, X17, X18, X19)
- #define **_GLIBCXX_TYPELIST_CHAIN3**(X0, X1, X2)
- #define **_GLIBCXX_TYPELIST_CHAIN4**(X0, X1, X2, X3)
- #define **_GLIBCXX_TYPELIST_CHAIN5**(X0, X1, X2, X3, X4)
- #define **_GLIBCXX_TYPELIST_CHAIN6**(X0, X1, X2, X3, X4, X5)
- #define **_GLIBCXX_TYPELIST_CHAIN7**(X0, X1, X2, X3, X4, X5, X6)
- #define **_GLIBCXX_TYPELIST_CHAIN8**(X0, X1, X2, X3, X4, X5, X6, X7)
- #define **_GLIBCXX_TYPELIST_CHAIN9**(X0, X1, X2, X3, X4, X5, X6, X7, X8)

Functions

- `template<typename Fn , typename Typelist >`
`void __gnu_cxx::typelist::apply (Fn &, Typelist)`
- `template<typename Gn , typename Typelist >`
`void __gnu_cxx::typelist::apply_generator (Gn &, Typelist)`
- `template<typename Gn , typename TypelistT , typename TypelistV >`
`void __gnu_cxx::typelist::apply_generator (Gn &, TypelistT, TypelistV)`
- `template<typename Fn , typename Typelist >`
`void __gnu_cxx::typelist::apply_generator (Fn &fn, Typelist)`
- `template<typename Fn , typename TypelistT , typename TypelistV >`
`void __gnu_cxx::typelist::apply_generator (Fn &fn, TypelistT, TypelistV)`

5.449.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Contains `typelist_chain` definitions. Typelists are an idea by Andrei Alexandrescu.

5.450 `types.h` File Reference

Namespaces

- [__gnu_parallel](#)

Typedefs

- `typedef int64_t __gnu_parallel::_CASable`
- `typedef uint64_t __gnu_parallel::_SequenceIndex`
- `typedef uint16_t __gnu_parallel::_ThreadIndex`

Enumerations

- `enum __gnu_parallel::_AlgorithmStrategy { heuristic, force_sequential, force_parallel }`
- `enum __gnu_parallel::_FindAlgorithm { GROWING_BLOCKS, CONSTANT_SIZE_BLOCKS, EQUAL_SPLIT }`
- `enum __gnu_parallel::_MultiwayMergeAlgorithm { LOSER_TREE }`
- `enum __gnu_parallel::_Parallelism { __gnu_parallel::sequential, __gnu_parallel::parallel_unbalanced, __gnu_parallel::parallel_balanced, __gnu_parallel::parallel_omp_loop, __gnu_parallel::parallel_omp_loop_static, __gnu_parallel::parallel_taskqueue }`
- `enum __gnu_parallel::_PartialSumAlgorithm { RECURSIVE, LINEAR }`
- `enum __gnu_parallel::_SortAlgorithm { MWMS, QS, QS_BALANCED }`
- `enum __gnu_parallel::_SplittingAlgorithm { SAMPLING, EXACT }`

Variables

- static const int [__gnu_parallel::_CASable_bits](#)
- static const _CASable [__gnu_parallel::_CASable_mask](#)

5.450.1 Detailed Description

Basic types and typedefs. This file is a GNU parallel extension to the Standard C++ Library.

5.451 types_traits.hpp File Reference

Classes

- struct [__gnu_pbds::detail::no_throw_copies< Key, Mapped >](#)
- struct [__gnu_pbds::detail::no_throw_copies< Key, null_type >](#)
- struct [__gnu_pbds::detail::stored_data< _Tv, _Th >](#)
- struct [__gnu_pbds::detail::stored_data< _Tv, null_type >](#)
- struct [__gnu_pbds::detail::stored_hash< _Th >](#)
- struct [__gnu_pbds::detail::stored_value< _Tv >](#)
- struct [__gnu_pbds::detail::type_base< Key, Mapped, _Alloc, Store_Hash >](#)
- struct [__gnu_pbds::detail::type_base< Key, Mapped, _Alloc, false >](#)
- struct [__gnu_pbds::detail::type_base< Key, Mapped, _Alloc, true >](#)
- struct [__gnu_pbds::detail::type_base< Key, null_type, _Alloc, false >](#)
- struct [__gnu_pbds::detail::type_base< Key, null_type, _Alloc, true >](#)
- struct [__gnu_pbds::detail::type_dispatch< Key, Mapped, _Alloc, Store_Hash >](#)
- struct [__gnu_pbds::detail::types_traits< Key, Mapped, _Alloc, Store_Hash >](#)

Namespaces

- [__gnu_pbds](#)

5.451.1 Detailed Description

Contains a traits class of types used by containers.

5.452 unique_copy.h File Reference

Namespaces

- [__gnu_parallel](#)

Functions

- `template<typename _Iter, class _OutputIterator, class _BinaryPredicate >`
`_OutputIterator __gnu_parallel::__parallel_unique_copy (_Iter __first, _Iter __last, _OutputIterator __result, _↔`
`BinaryPredicate __binary_pred)`
- `template<typename _Iter, class _OutputIterator >`
`_OutputIterator __gnu_parallel::__parallel_unique_copy (_Iter __first, _Iter __last, _OutputIterator __result)`

5.452.1 Detailed Description

Parallel implementations of `std::unique_copy()`. This file is a GNU parallel extension to the Standard C++ Library.

5.453 `unique_ptr.h` File Reference

Classes

- struct `std::default_delete<_Tp >`
- struct `std::default_delete<_Tp[]>`
- struct `std::hash< unique_ptr<_Tp, _Dp > >`
- class `std::unique_ptr<_Tp, _Dp >`
- class `std::unique_ptr<_Tp[], _Dp >`

Namespaces

- `std`

Functions

- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`
`bool std::operator!= (const unique_ptr<_Tp, _Dp > &__x, const unique_ptr<_Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp >`
`bool std::operator!= (const unique_ptr<_Tp, _Dp > &__x, nullptr_t) noexcept`
- `template<typename _Tp, typename _Dp >`
`bool std::operator!= (nullptr_t, const unique_ptr<_Tp, _Dp > &__x) noexcept`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`
`bool std::operator< (const unique_ptr<_Tp, _Dp > &__x, const unique_ptr<_Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp >`
`bool std::operator< (const unique_ptr<_Tp, _Dp > &__x, nullptr_t)`
- `template<typename _Tp, typename _Dp >`
`bool std::operator< (nullptr_t, const unique_ptr<_Tp, _Dp > &__x)`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`
`bool std::operator<= (const unique_ptr<_Tp, _Dp > &__x, const unique_ptr<_Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp >`
`bool std::operator<= (const unique_ptr<_Tp, _Dp > &__x, nullptr_t)`
- `template<typename _Tp, typename _Dp >`
`bool std::operator<= (nullptr_t, const unique_ptr<_Tp, _Dp > &__x)`

- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`
`bool std::operator== (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp >`
`bool std::operator== (const unique_ptr< _Tp, _Dp > &__x, nullptr_t) noexcept`
- `template<typename _Tp, typename _Dp >`
`bool std::operator== (nullptr_t, const unique_ptr< _Tp, _Dp > &__x) noexcept`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`
`bool std::operator> (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp >`
`bool std::operator> (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)`
- `template<typename _Tp, typename _Dp >`
`bool std::operator> (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`
`bool std::operator>= (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp >`
`bool std::operator>= (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)`
- `template<typename _Tp, typename _Dp >`
`bool std::operator>= (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)`
- `template<typename _Tp, typename _Dp >`
`void std::swap (unique_ptr< _Tp, _Dp > &__x, unique_ptr< _Tp, _Dp > &__y) noexcept`

5.453.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

5.454 unordered_base.h File Reference

Namespaces

- [std](#)
- [std::__profile](#)

Functions

- `template<typename _UnorderedCont, typename _Value, bool _Cache_hash_code>`
`bool std::__profile::are_equal (const _UnorderedCont &__uc, const __detail::Hash_node< _Value, _↵
Cache_hash_code > * __lhs, const __detail::Hash_node< _Value, _Cache_hash_code > * __rhs)`
- `template<typename _UnorderedCont, typename _Value, bool _Cache_hash_code>`
`std::size_t std::__profile::get_bucket_index (const _UnorderedCont &__uc, const __detail::Hash_node<
_Value, _Cache_hash_code > * __node)`

5.454.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

5.455 unordered_map.h File Reference

Classes

- class `std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>`
- class `std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>`

Namespaces

- `std`

Typedefs

- `template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = std::equal_to<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp>>, typename _Tr = __umap_traits<__cache_default<_Key, _Hash>::value>>`
`using std::__umap_hashtable = _Hashtable<_Key, std::pair< const _Key, _Tp >, _Alloc, __detail:: Select1st,`
`_Pred, _Hash, __detail:: Mod_range_hashing, __detail:: Default_ranged_hash, __detail:: Prime_rehash_policy,`
`_Tr >`
- `template<bool _Cache>`
`using std::__umap_traits = __detail:: _Hashtable_traits< _Cache, false, true >`
- `template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = std::equal_to<_Key>, typename _Alloc =`
`std::allocator<std::pair<const _Key, _Tp>>, typename _Tr = __ummap_traits<__cache_default<_Key, _Hash>::value>>`
`using std::__ummap_hashtable = _Hashtable<_Key, std::pair< const _Key, _Tp >, _Alloc, __detail::↵`
`Select1st, _Pred, _Hash, __detail:: Mod_range_hashing, __detail:: Default_ranged_hash, __detail:: Prime↵`
`rehash_policy, _Tr >`
- `template<bool _Cache>`
`using std::__ummap_traits = __detail:: _Hashtable_traits< _Cache, false, false >`

Functions

- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc >`
`bool std::operator!= (const unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc> &__x, const unordered_map<`
`_Key, _Tp, _Hash, _Pred, _Alloc> &__y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc >`
`bool std::operator!= (const unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc> &__x, const unordered↵`
`_multimap<_Key, _Tp, _Hash, _Pred, _Alloc> &__y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc >`
`bool std::operator== (const unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc> &__x, const unordered_map<`
`_Key, _Tp, _Hash, _Pred, _Alloc> &__y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc >`
`bool std::operator== (const unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc> &__x, const unordered↵`
`_multimap<_Key, _Tp, _Hash, _Pred, _Alloc> &__y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc >`
`void std::swap (unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc> &__x, unordered_map<_Key, _Tp, ↵`
`_Hash, _Pred, _Alloc> &__y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc >`
`void std::swap (unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc> &__x, unordered_multimap<_Key,`
`_Tp, _Hash, _Pred, _Alloc> &__y)`

5.455.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<unordered_map>`.

5.456 unordered_set.h File Reference

Classes

- class [std::unordered_multiset<_Value, _Hash, _Pred, _Alloc>](#)
- class [std::unordered_set<_Value, _Hash, _Pred, _Alloc>](#)

Namespaces

- [std](#)

Typedefs

- `template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = std::equal_to<_Value>, typename _Alloc = std::allocator<_Value>, typename _Tr = __umset_traits<__cache_default<_Value, _Hash>::value>>
using std::__umset_hashtable = _Hashtable<_Value, _Value, _Alloc, __detail::__Identity, _Pred, _Hash, __detail::__Mod_range_hashing, __detail::__Default_ranged_hash, __detail::__Prime_rehash_policy, _Tr>`
- `template<bool _Cache>
using std::__umset_traits = __detail::__Hashtable_traits<_Cache, true, false>`
- `template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = std::equal_to<_Value>, typename _Alloc = std::allocator<_Value>, typename _Tr = __uset_traits<__cache_default<_Value, _Hash>::value>>
using std::__uset_hashtable = _Hashtable<_Value, _Value, _Alloc, __detail::__Identity, _Pred, _Hash, __detail::__Mod_range_hashing, __detail::__Default_ranged_hash, __detail::__Prime_rehash_policy, _Tr>`
- `template<bool _Cache>
using std::__uset_traits = __detail::__Hashtable_traits<_Cache, true, true>`

Functions

- `template<class _Value, class _Hash, class _Pred, class _Alloc>
bool std::operator!= (const unordered_set<_Value, _Hash, _Pred, _Alloc> &__x, const unordered_set<_Value, _Hash, _Pred, _Alloc> &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc>
bool std::operator!= (const unordered_multiset<_Value, _Hash, _Pred, _Alloc> &__x, const unordered_multiset<_Value, _Hash, _Pred, _Alloc> &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc>
bool std::operator== (const unordered_set<_Value, _Hash, _Pred, _Alloc> &__x, const unordered_set<_Value, _Hash, _Pred, _Alloc> &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc>
bool std::operator== (const unordered_multiset<_Value, _Hash, _Pred, _Alloc> &__x, const unordered_multiset<_Value, _Hash, _Pred, _Alloc> &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc>
void std::swap (unordered_set<_Value, _Hash, _Pred, _Alloc> &__x, unordered_set<_Value, _Hash, _Pred, _Alloc> &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc>
void std::swap (unordered_multiset<_Value, _Hash, _Pred, _Alloc> &__x, unordered_multiset<_Value, _Hash, _Pred, _Alloc> &__y)`

5.456.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<unordered_set>`.

5.457 `update_fn_imps.hpp` File Reference

5.457.1 Detailed Description

Contains an implementation class for `pat_trie_`.

5.458 `valarray_after.h` File Reference

Namespaces

- [std](#)

Macros

- `#define DEFINE_EXPR_BINARY_FUNCTION(_Fun, _UFun)`
- `#define DEFINE_EXPR_BINARY_OPERATOR(_Op, _Name)`
- `#define DEFINE_EXPR_UNARY_FUNCTION(_Name, _UName)`
- `#define DEFINE_EXPR_UNARY_OPERATOR(_Op, _Name)`

Functions

- `template<class _Dom >`
`_Expr< _UnClos< _Abs, _Expr, _Dom >, typename _Dom::value_type > std::abs (const _Expr< _Dom, type-`
`name _Dom::value_type > &__e)`
- `template<typename _Tp >`
`_Expr< _UnClos< _Abs, _ValArray, _Tp >, _Tp > std::abs (const valarray< _Tp > &__v)`
- `template<class _Dom >`
`_Expr< _UnClos< _Acos, _Expr, _Dom >, typename _Dom::value_type > std::acos (const _Expr< _Dom,`
`typename _Dom::value_type > &__e)`
- `template<typename _Tp >`
`_Expr< _UnClos< _Acos, _ValArray, _Tp >, _Tp > std::acos (const valarray< _Tp > &__v)`
- `template<class _Dom >`
`_Expr< _UnClos< _Asin, _Expr, _Dom >, typename _Dom::value_type > std::asin (const _Expr< _Dom, type-`
`name _Dom::value_type > &__e)`
- `template<typename _Tp >`
`_Expr< _UnClos< _Asin, _ValArray, _Tp >, _Tp > std::asin (const valarray< _Tp > &__v)`
- `template<class _Dom >`
`_Expr< _UnClos< _Atan, _Expr, _Dom >, typename _Dom::value_type > std::atan (const _Expr< _Dom,`
`typename _Dom::value_type > &__e)`
- `template<typename _Tp >`
`_Expr< _UnClos< _Atan, _ValArray, _Tp >, _Tp > std::atan (const valarray< _Tp > &__v)`

- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< _Atan2, _Expr, _Expr, _Dom1, _Dom2 >, typename _Dom1::value_type > std::atan2 (const`
`_Expr< _Dom1, typename _Dom1::value_type > &__e1, const _Expr< _Dom2, typename _Dom2::value_type`
`> &__e2)`
- `template<class _Dom >`
`_Expr< _BinClos< _Atan2, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename _Dom::value_↵`
`_type > std::atan2 (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _↵`
`_Dom::value_type > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< _Atan2, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename _Dom::value_↵`
`type > std::atan2 (const valarray< typename _Dom::valarray > &__v, const _Expr< _Dom, typename _Dom_↵`
`::value_type > &__e)`
- `template<class _Dom >`
`_Expr< _BinClos< _Atan2, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename _Dom::value_↵`
`_type > std::atan2 (const _Expr< _Dom, typename _Dom::value_type > &__e, const typename _Dom::value_↵`
`_type &__t)`
- `template<class _Dom >`
`_Expr< _BinClos< _Atan2, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename _Dom::value_↵`
`_type > std::atan2 (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type`
`> &__e)`
- `template<typename _Tp >`
`_Expr< _BinClos< _Atan2, _ValArray, _ValArray, _Tp, _Tp >, _Tp > std::atan2 (const valarray< _Tp > &__v,`
`const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< _Atan2, _ValArray, _Constant, _Tp, _Tp >, _Tp > std::atan2 (const valarray< _Tp > &__v,`
`const _Tp &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< _Atan2, _Constant, _ValArray, _Tp, _Tp >, _Tp > std::atan2 (const _Tp &__t, const`
`valarray< _Tp > &__v)`
- `template<class _Dom >`
`_Expr< _UnClos< _Cos, _Expr, _Dom >, typename _Dom::value_type > std::cos (const _Expr< _Dom, type-`
`name _Dom::value_type > &__e)`
- `template<typename _Tp >`
`_Expr< _UnClos< _Cos, _ValArray, _Tp >, _Tp > std::cos (const valarray< _Tp > &__v)`
- `template<class _Dom >`
`_Expr< _UnClos< _Cosh, _Expr, _Dom >, typename _Dom::value_type > std::cosh (const _Expr< _Dom,`
`typename _Dom::value_type > &__e)`
- `template<typename _Tp >`
`_Expr< _UnClos< _Cosh, _ValArray, _Tp >, _Tp > std::cosh (const valarray< _Tp > &__v)`
- `template<class _Dom >`
`_Expr< _UnClos< _Exp, _Expr, _Dom >, typename _Dom::value_type > std::exp (const _Expr< _Dom, type-`
`name _Dom::value_type > &__e)`
- `template<typename _Tp >`
`_Expr< _UnClos< _Exp, _ValArray, _Tp >, _Tp > std::exp (const valarray< _Tp > &__v)`
- `template<typename _Tp >`
`_Expr< _UnClos< _Log, _ValArray, _Tp >, _Tp > std::log (const valarray< _Tp > &__v)`
- `template<class _Dom >`
`_Expr< _UnClos< _Log, _Expr, _Dom >, typename _Dom::value_type > std::log (const _Expr< _Dom, type-`
`name _Dom::value_type > &__e)`
- `template<class _Dom >`
`_Expr< _UnClos< _Log10, _Expr, _Dom >, typename _Dom::value_type > std::log10 (const _Expr< _Dom,`
`typename _Dom::value_type > &__e)`

- `template<typename _Tp >`
`_Expr< _BinClos< _Log10, _ValArray, _Tp >, _Tp > std::log10 (const valarray< _Tp > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __not_equal_to, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __not_equal_to, typename _Dom::value_type >::result_type > std::operator!= (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __not_equal_to, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< __not_equal_to, typename _Dom::value_type >::result_type > std::operator!= (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __not_equal_to, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __not_equal_to, typename _Dom::value_type >::result_type > std::operator!= (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __not_equal_to, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __not_equal_to, typename _Dom1::value_type >::result_type > std::operator!= (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`
`_Expr< _BinClos< __not_equal_to, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< __not_equal_to, typename _Dom::value_type >::result_type > std::operator!= (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`
`_Expr< _BinClos< __modulus, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< __modulus, typename _Dom::value_type >::result_type > std::operator% (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __modulus, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __modulus, typename _Dom1::value_type >::result_type > std::operator% (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`
`_Expr< _BinClos< __modulus, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __modulus, typename _Dom::value_type >::result_type > std::operator% (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __modulus, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< __modulus, typename _Dom::value_type >::result_type > std::operator% (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __modulus, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __modulus, typename _Dom::value_type >::result_type > std::operator% (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`
`_Expr< _BinClos< __bitwise_and, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< __bitwise_and, typename _Dom::value_type >::result_type > std::operator& (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`
`_Expr< _BinClos< __bitwise_and, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __bitwise_and, typename _Dom::value_type >::result_type > std::operator& (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __bitwise_and, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< __bitwise_and, typename _Dom::value_type >::result_type > std::operator& (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`

- `template<class _Dom >`
`_Expr< _BinClos< __bitwise_and, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __bitwise_and, typename _Dom::value_type >::result_type > std::operator& (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __bitwise_and, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __bitwise_and, typename _Dom1::value_type >::result_type > std::operator& (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __logical_and, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __logical_and, typename _Dom1::value_type >::result_type > std::operator&& (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`
`_Expr< _BinClos< __logical_and, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __logical_and, typename _Dom::value_type >::result_type > std::operator&& (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __logical_and, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< __logical_and, typename _Dom::value_type >::result_type > std::operator&& (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __logical_and, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __logical_and, typename _Dom::value_type >::result_type > std::operator&& (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`
`_Expr< _BinClos< __logical_and, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< __logical_and, typename _Dom::value_type >::result_type > std::operator&& (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __multiplies, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __multiplies, typename _Dom1::value_type >::result_type > std::operator* (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`
`_Expr< _BinClos< __multiplies, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< __multiplies, typename _Dom::value_type >::result_type > std::operator* (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`
`_Expr< _BinClos< __multiplies, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __multiplies, typename _Dom::value_type >::result_type > std::operator* (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __multiplies, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< __multiplies, typename _Dom::value_type >::result_type > std::operator* (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __multiplies, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __multiplies, typename _Dom::value_type >::result_type > std::operator* (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`
`_Expr< _BinClos< __plus, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< __plus, typename _Dom::value_type >::result_type > std::operator+ (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`

- `template<class _Dom >`
`_Expr< _BinClos< __plus, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< __plus,`
`typename _Dom::value_type >::result_type > std::operator+ (const _Expr< _Dom, typename _Dom::value_↵`
`type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`
`_Expr< _BinClos< __plus, ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __plus,`
`typename _Dom::value_type >::result_type > std::operator+ (const valarray< typename _Dom::value_type >`
`&__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __plus, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __plus, typename _Dom1↵`
`::value_type >::result_type > std::operator+ (const _Expr< _Dom1, typename _Dom1::value_type > &__v,`
`const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`
`_Expr< _BinClos< __plus, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __plus,`
`typename _Dom::value_type >::result_type > std::operator+ (const typename _Dom::value_type &__t, const`
`_Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __minus, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __↵`
`_minus, typename _Dom::value_type >::result_type > std::operator- (const typename _Dom::value_type &__t,`
`const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __minus, _Expr, ValArray, _Dom, typename _Dom::value_type >, typename __fun< __↵`
`minus, typename _Dom::value_type >::result_type > std::operator- (const _Expr< _Dom, typename _Dom↵`
`::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __minus, ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __↵`
`minus, typename _Dom::value_type >::result_type > std::operator- (const valarray< typename _Dom::value↵`
`_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __minus, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __minus, typename _Dom1↵`
`::value_type >::result_type > std::operator- (const _Expr< _Dom1, typename _Dom1::value_type > &__↵`
`v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`
`_Expr< _BinClos< __minus, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< __↵`
`minus, typename _Dom::value_type >::result_type > std::operator- (const _Expr< _Dom, typename _Dom↵`
`::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __divides, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __divides, typename ↵`
`_Dom1::value_type >::result_type > std::operator/ (const _Expr< _Dom1, typename _Dom1::value_type > &↵`
`__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`
`_Expr< _BinClos< __divides, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< __↵`
`divides, typename _Dom::value_type >::result_type > std::operator/ (const _Expr< _Dom, typename _Dom↵`
`::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`
`_Expr< _BinClos< __divides, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __↵`
`divides, typename _Dom::value_type >::result_type > std::operator/ (const typename _Dom::value_type &__t,`
`const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __divides, _Expr, ValArray, _Dom, typename _Dom::value_type >, typename __fun< __↵`
`divides, typename _Dom::value_type >::result_type > std::operator/ (const _Expr< _Dom, typename _Dom↵`
`::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`

- `template<class _Dom >`
`_Expr< _BinClos< __divides, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __divides, typename _Dom::value_type >::result_type > std::operator/ (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __less, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __less, typename _Dom1::value_type >::result_type > std::operator< (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`
`_Expr< _BinClos< __less, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< __less, typename _Dom::value_type >::result_type > std::operator< (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`
`_Expr< _BinClos< __less, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __less, typename _Dom::value_type >::result_type > std::operator< (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __less, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< __less, typename _Dom::value_type >::result_type > std::operator< (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __less, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __less, typename _Dom::value_type >::result_type > std::operator< (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __shift_left, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __shift_left, typename _Dom1::value_type >::result_type > std::operator<< (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`
`_Expr< _BinClos< __shift_left, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< __shift_left, typename _Dom::value_type >::result_type > std::operator<< (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`
`_Expr< _BinClos< __shift_left, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __shift_left, typename _Dom::value_type >::result_type > std::operator<< (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __shift_left, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< __shift_left, typename _Dom::value_type >::result_type > std::operator<< (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __shift_left, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __shift_left, typename _Dom::value_type >::result_type > std::operator<< (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __less_equal, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __less_equal, typename _Dom1::value_type >::result_type > std::operator<= (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`
`_Expr< _BinClos< __less_equal, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< __less_equal, typename _Dom::value_type >::result_type > std::operator<= (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`

- `template<class _Dom >`
`_Expr< _BinClos< __less_equal, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun<`
`__less_equal, typename _Dom::value_type >::result_type > std::operator<= (const _Expr< _Dom, typename`
`_Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __less_equal, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun<`
`__less_equal, typename _Dom::value_type >::result_type > std::operator<= (const typename _Dom::value_`
`__type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __less_equal, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun<`
`__less_equal, typename _Dom::value_type >::result_type > std::operator<= (const valarray< typename _`
`Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`
`_Expr< _BinClos< __equal_to, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun<`
`__equal_to, typename _Dom::value_type >::result_type > std::operator== (const _Expr< _Dom, typename`
`_Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __equal_to, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun<`
`__equal_to, typename _Dom::value_type >::result_type > std::operator== (const valarray< typename _Dom`
`::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`
`_Expr< _BinClos< __equal_to, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun<`
`__equal_to, typename _Dom::value_type >::result_type > std::operator== (const typename _Dom::value_type`
`&__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __equal_to, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __equal_to, typename`
`_Dom1::value_type >::result_type > std::operator== (const _Expr< _Dom1, typename _Dom1::value_type >`
`&__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`
`_Expr< _BinClos< __equal_to, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun<`
`__equal_to, typename _Dom::value_type >::result_type > std::operator== (const _Expr< _Dom, typename`
`_Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`
`_Expr< _BinClos< __greater, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun<`
`__greater, typename _Dom::value_type >::result_type > std::operator> (const _Expr< _Dom, typename`
`_Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`
`_Expr< _BinClos< __greater, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun<`
`__greater, typename _Dom::value_type >::result_type > std::operator> (const _Expr< _Dom, typename _Dom`
`::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __greater, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun<`
`__greater, typename _Dom::value_type >::result_type > std::operator> (const valarray< typename _Dom`
`::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __greater, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __greater, typename`
`_Dom1::value_type >::result_type > std::operator> (const _Expr< _Dom1, typename _Dom1::value_type >`
`&__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`
`_Expr< _BinClos< __greater, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun<`
`__greater, typename _Dom::value_type >::result_type > std::operator> (const typename _Dom::value_type`
`&__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`

- `template<class _Dom >`
`_Expr< _BinClos< __greater_equal, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< __greater_equal, typename _Dom::value_type >::result_type > std::operator>= (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __greater_equal, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __greater_equal, typename _Dom::value_type >::result_type > std::operator>= (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __greater_equal, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __greater_equal, typename _Dom::value_type >::result_type > std::operator>= (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __greater_equal, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __greater_equal, typename _Dom1::value_type >::result_type > std::operator>= (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`
`_Expr< _BinClos< __greater_equal, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< __greater_equal, typename _Dom::value_type >::result_type > std::operator>= (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __shift_right, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __shift_right, typename _Dom1::value_type >::result_type > std::operator>> (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`
`_Expr< _BinClos< __shift_right, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< __shift_right, typename _Dom::value_type >::result_type > std::operator>> (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`
`_Expr< _BinClos< __shift_right, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __shift_right, typename _Dom::value_type >::result_type > std::operator>> (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __shift_right, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< __shift_right, typename _Dom::value_type >::result_type > std::operator>> (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __shift_right, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __shift_right, typename _Dom::value_type >::result_type > std::operator>> (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`
`_Expr< _BinClos< __bitwise_xor, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< __bitwise_xor, typename _Dom::value_type >::result_type > std::operator^ (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`
`_Expr< _BinClos< __bitwise_xor, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< __bitwise_xor, typename _Dom::value_type >::result_type > std::operator^ (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __bitwise_xor, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __bitwise_xor, typename _Dom1::value_type >::result_type > std::operator^ (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`

- `template<class _Dom >`
`_Expr< _BinClos< __bitwise_xor, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun<`
`__bitwise_xor, typename _Dom::value_type >::result_type > std::operator^ (const _Expr< _Dom, typename`
`_Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`
`_Expr< _BinClos< __bitwise_xor, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun<`
`__bitwise_xor, typename _Dom::value_type >::result_type > std::operator^ (const typename _Dom::value_←`
`type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __bitwise_or, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __bitwise_or, typename`
`_Dom1::value_type >::result_type > std::operator| (const _Expr< _Dom1, typename _Dom1::value_type >`
`&__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`
`_Expr< _BinClos< __bitwise_or, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun<`
`__bitwise_or, typename _Dom::value_type >::result_type > std::operator| (const _Expr< _Dom, typename ←`
`_Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`
`_Expr< _BinClos< __bitwise_or, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun<`
`__bitwise_or, typename _Dom::value_type >::result_type > std::operator| (const _Expr< _Dom, typename ←`
`_Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __bitwise_or, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun<`
`__bitwise_or, typename _Dom::value_type >::result_type > std::operator| (const typename _Dom::value_type`
`&__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __bitwise_or, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun<`
`__bitwise_or, typename _Dom::value_type >::result_type > std::operator| (const valarray< typename _Dom←`
`::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`
`_Expr< _BinClos< __logical_or, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun<`
`__logical_or, typename _Dom::value_type >::result_type > std::operator|| (const valarray< typename _Dom←`
`::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`
`_Expr< _BinClos< __logical_or, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun<`
`__logical_or, typename _Dom::value_type >::result_type > std::operator|| (const _Expr< _Dom, typename ←`
`_Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`
`_Expr< _BinClos< __logical_or, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun<`
`__logical_or, typename _Dom::value_type >::result_type > std::operator|| (const _Expr< _Dom, typename ←`
`_Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom >`
`_Expr< _BinClos< __logical_or, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun<`
`__logical_or, typename _Dom::value_type >::result_type > std::operator|| (const typename _Dom::value_type`
`&__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< __logical_or, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< __logical_or, typename`
`_Dom1::value_type >::result_type > std::operator|| (const _Expr< _Dom1, typename _Dom1::value_type >`
`&__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`
`_Expr< _BinClos< _Pow, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename _Dom::value_type`
`> std::pow (const valarray< typename _Dom::valarray > &__v, const _Expr< _Dom, typename _Dom::value_←`
`type > &__e)`

- `template<class _Dom >`
`_Expr< _BinClos< _Pow, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename _Dom::value_type >`
`> std::pow (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type >`
`&__e)`
- `template<typename _Tp >`
`_Expr< _BinClos< _Pow, _ValArray, _Constant, _Tp, _Tp >, _Tp > std::pow (const valarray< _Tp > &__v,`
`const _Tp &__t)`
- `template<class _Dom >`
`_Expr< _BinClos< _Pow, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename _Dom::value_type >`
`> std::pow (const _Expr< _Dom, typename _Dom::value_type > &__e, const typename _Dom::value_type`
`&__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< _Pow, _ValArray, _ValArray, _Tp, _Tp >, _Tp > std::pow (const valarray< _Tp > &__v,`
`const valarray< _Tp > &__w)`
- `template<class _Dom >`
`_Expr< _BinClos< _Pow, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename _Dom::value_type`
`> std::pow (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom1, class _Dom2 >`
`_Expr< _BinClos< _Pow, _Expr, _Expr, _Dom1, _Dom2 >, typename _Dom1::value_type > std::pow (const`
`_Expr< _Dom1, typename _Dom1::value_type > &__e1, const _Expr< _Dom2, typename _Dom2::value_type`
`> &__e2)`
- `template<typename _Tp >`
`_Expr< _BinClos< _Pow, _Constant, _ValArray, _Tp, _Tp >, _Tp > std::pow (const _Tp &__t, const valarray<`
`_Tp > &__v)`
- `template<class _Dom >`
`_Expr< _UnClos< _Sin, _Expr, _Dom >, typename _Dom::value_type > std::sin (const _Expr< _Dom, type-`
`name _Dom::value_type > &__e)`
- `template<typename _Tp >`
`_Expr< _UnClos< _Sin, _ValArray, _Tp >, _Tp > std::sin (const valarray< _Tp > &__v)`
- `template<class _Dom >`
`_Expr< _UnClos< _Sinh, _Expr, _Dom >, typename _Dom::value_type > std::sinh (const _Expr< _Dom,`
`typename _Dom::value_type > &__e)`
- `template<typename _Tp >`
`_Expr< _UnClos< _Sinh, _ValArray, _Tp >, _Tp > std::sinh (const valarray< _Tp > &__v)`
- `template<typename _Tp >`
`_Expr< _UnClos< _Sqrt, _ValArray, _Tp >, _Tp > std::sqrt (const valarray< _Tp > &__v)`
- `template<class _Dom >`
`_Expr< _UnClos< _Sqrt, _Expr, _Dom >, typename _Dom::value_type > std::sqrt (const _Expr< _Dom, type-`
`name _Dom::value_type > &__e)`
- `template<typename _Tp >`
`_Expr< _UnClos< _Tan, _ValArray, _Tp >, _Tp > std::tan (const valarray< _Tp > &__v)`
- `template<class _Dom >`
`_Expr< _UnClos< _Tan, _Expr, _Dom >, typename _Dom::value_type > std::tan (const _Expr< _Dom, type-`
`name _Dom::value_type > &__e)`
- `template<class _Dom >`
`_Expr< _UnClos< _Tanh, _Expr, _Dom >, typename _Dom::value_type > std::tanh (const _Expr< _Dom,`
`typename _Dom::value_type > &__e)`
- `template<typename _Tp >`
`_Expr< _UnClos< _Tanh, _ValArray, _Tp >, _Tp > std::tanh (const valarray< _Tp > &__v)`

5.458.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<valarray>`.

5.459 `valarray_array.h` File Reference

Namespaces

- [std](#)

Macros

- `#define _DEFINE_ARRAY_FUNCTION(_Op, _Name)`

Functions

- `template<typename _Tp >`
`void std::__valarray_copy (const _Tp *__restrict __a, size_t __n, _Tp *__restrict __b)`
- `template<typename _Tp >`
`void std::__valarray_copy (const _Tp *__restrict __a, size_t __n, size_t __s, _Tp *__restrict __b)`
- `template<typename _Tp >`
`void std::__valarray_copy (const _Tp *__restrict __a, _Tp *__restrict __b, size_t __n, size_t __s)`
- `template<typename _Tp >`
`void std::__valarray_copy (const _Tp *__restrict __src, size_t __n, size_t __s1, _Tp *__restrict __dst, size_t __s2)`
- `template<typename _Tp >`
`void std::__valarray_copy (const _Tp *__restrict __a, const size_t *__restrict __i, _Tp *__restrict __b, size_t __n)`
- `template<typename _Tp >`
`void std::__valarray_copy (const _Tp *__restrict __a, size_t __n, _Tp *__restrict __b, const size_t *__restrict __i)`
- `template<typename _Tp >`
`void std::__valarray_copy (const _Tp *__restrict __src, size_t __n, const size_t *__restrict __i, _Tp *__restrict __dst, const size_t *__restrict __j)`
- `template<typename _Tp >`
`void std::__valarray_copy (_Array<_Tp> __a, size_t __n, _Array<_Tp> __b)`
- `template<typename _Tp >`
`void std::__valarray_copy (_Array<_Tp> __a, size_t __n, size_t __s, _Array<_Tp> __b)`
- `template<typename _Tp >`
`void std::__valarray_copy (_Array<_Tp> __a, _Array<_Tp> __b, size_t __n, size_t __s)`
- `template<typename _Tp >`
`void std::__valarray_copy (_Array<_Tp> __a, size_t __n, size_t __s1, _Array<_Tp> __b, size_t __s2)`
- `template<typename _Tp >`
`void std::__valarray_copy (_Array<_Tp> __a, _Array<size_t> __i, _Array<_Tp> __b, size_t __n)`
- `template<typename _Tp >`
`void std::__valarray_copy (_Array<_Tp> __a, size_t __n, _Array<_Tp> __b, _Array<size_t> __i)`

- `template<typename _Tp >`
`void std::__valarray_copy (_Array< _Tp > __src, size_t __n, _Array< size_t > __i, _Array< _Tp > __dst,`
`_Array< size_t > __j)`
- `template<typename _Tp >`
`void std::__valarray_copy_construct (const _Tp *__b, const _Tp *__e, _Tp *__restrict __o)`
- `template<typename _Tp >`
`void std::__valarray_copy_construct (const _Tp *__restrict __a, size_t __n, size_t __s, _Tp *__restrict`
`__o)`
- `template<typename _Tp >`
`void std::__valarray_copy_construct (const _Tp *__restrict __a, const size_t *__restrict __i, _Tp *__`
`restrict __o, size_t __n)`
- `template<typename _Tp >`
`void std::__valarray_copy_construct (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t`
`__n)`
- `template<typename _Tp >`
`void std::__valarray_copy_construct (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`
`void std::__valarray_default_construct (_Tp *__b, _Tp *__e)`
- `template<typename _Tp >`
`void std::__valarray_destroy_elements (_Tp *__b, _Tp *__e)`
- `template<typename _Tp >`
`void std::__valarray_fill (_Tp *__restrict __a, size_t __n, const _Tp & __t)`
- `template<typename _Tp >`
`void std::__valarray_fill (_Tp *__restrict __a, size_t __n, size_t __s, const _Tp & __t)`
- `template<typename _Tp >`
`void std::__valarray_fill (_Tp *__restrict __a, const size_t *__restrict __i, size_t __n, const _Tp & __t)`
- `template<typename _Tp >`
`void std::__valarray_fill (_Array< _Tp > __a, size_t __n, const _Tp & __t)`
- `template<typename _Tp >`
`void std::__valarray_fill (_Array< _Tp > __a, size_t __n, size_t __s, const _Tp & __t)`
- `template<typename _Tp >`
`void std::__valarray_fill (_Array< _Tp > __a, _Array< size_t > __i, size_t __n, const _Tp & __t)`
- `template<typename _Tp >`
`void std::__valarray_fill_construct (_Tp *__b, _Tp *__e, const _Tp & __t)`
- `void * std::__valarray_get_memory (size_t __n)`
- `template<typename _Tp >`
`_Tp *__restrict std::__valarray_get_storage (size_t __n)`
- `template<typename _Ta >`
`_Ta::value_type std::__valarray_max (const _Ta & __a)`
- `template<typename _Ta >`
`_Ta::value_type std::__valarray_min (const _Ta & __a)`
- `template<typename _Tp >`
`_Tp std::__valarray_product (const _Tp *__f, const _Tp *__l)`
- `void std::__valarray_release_memory (void *__p)`
- `template<typename _Tp >`
`_Tp std::__valarray_sum (const _Tp *__f, const _Tp *__l)`
- `template<typename _Tp, class _Dom >`
`void std::__Array_augmented_bitwise_and (_Array< _Tp > __a, _Array< size_t > __i, const Expr< _Dom,`
`_Tp > & __e, size_t __n)`
- `template<typename _Tp >`
`void std::__Array_augmented_bitwise_and (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b,`
`size_t __n)`

- `template<typename _Tp >`
`void std::Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__bitwise_and (_Array< _Tp > __a, _Array< bool > __m, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp >`
`void std::Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`
`void std::Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__bitwise_and (_Array< _Tp > __a, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`
`void std::Array_augmented__bitwise_and (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __s, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__bitwise_and (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__bitwise_or (_Array< _Tp > __a, _Array< bool > __m, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`
`void std::Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__bitwise_or (_Array< _Tp > __a, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`
`void std::Array_augmented__bitwise_or (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __s, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__bitwise_or (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__bitwise_or (_Array< _Tp > __a, _Array< size_t > __i, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__bitwise_or (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`

- `template<typename _Tp >`
`void std::_Array_augmented___bitwise_or (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`
`void std::_Array_augmented___bitwise_or (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp >`
`void std::_Array_augmented___bitwise_xor (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`
`void std::_Array_augmented___bitwise_xor (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`
`void std::_Array_augmented___bitwise_xor (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::_Array_augmented___bitwise_xor (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`
`void std::_Array_augmented___bitwise_xor (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`
`void std::_Array_augmented___bitwise_xor (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::_Array_augmented___bitwise_xor (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`
`void std::_Array_augmented___bitwise_xor (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom >`
`void std::_Array_augmented___bitwise_xor (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::_Array_augmented___bitwise_xor (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`
`void std::_Array_augmented___bitwise_xor (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`
`void std::_Array_augmented___bitwise_xor (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp >`
`void std::_Array_augmented___divides (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`
`void std::_Array_augmented___divides (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`
`void std::_Array_augmented___divides (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::_Array_augmented___divides (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`
`void std::_Array_augmented___divides (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`
`void std::_Array_augmented___divides (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)`

- `template<typename _Tp >`
`void std::Array_augmented__divides (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b,`
`size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__divides (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t`
`> __i)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__divides (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp`
`> &__e, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__divides (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t`
`__n)`
- `template<typename _Tp >`
`void std::Array_augmented__divides (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool >`
`__m)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__divides (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp`
`> &__e, size_t __n)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__minus (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp`
`> &__e, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__minus (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`
`void std::Array_augmented__minus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp >`
`void std::Array_augmented__minus (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t`
`__n)`
- `template<typename _Tp >`
`void std::Array_augmented__minus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t`
`> __i)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__minus (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e,`
`size_t __n)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__minus (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp`
`> &__e, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__minus (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp >`
`void std::Array_augmented__minus (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__minus (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__minus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool >`
`__m)`
- `template<typename _Tp >`
`void std::Array_augmented__minus (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t`
`__n)`
- `template<typename _Tp >`
`void std::Array_augmented__modulus (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`
`void std::Array_augmented__modulus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`

- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__modulus (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__modulus (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__modulus (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__modulus (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__modulus (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__modulus (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__modulus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__modulus (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__modulus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp >`
`void std::Array_augmented__modulus (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`
`void std::Array_augmented__multiplies (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp >`
`void std::Array_augmented__multiplies (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__multiplies (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__multiplies (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__multiplies (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__multiplies (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`
`void std::Array_augmented__multiplies (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__multiplies (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__multiplies (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__multiplies (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)`

- `template<typename _Tp >`
`void std::Array_augmented__multiplies (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`
`void std::Array_augmented__multiplies (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp >`
`void std::Array_augmented__plus (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__plus (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__plus (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`
`void std::Array_augmented__plus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp >`
`void std::Array_augmented__plus (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__plus (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__plus (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__plus (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__plus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp >`
`void std::Array_augmented__plus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`
`void std::Array_augmented__plus (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__plus (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__shift_left (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__shift_left (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__shift_left (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__shift_left (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__shift_left (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`
`void std::Array_augmented__shift_left (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__shift_left (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`

- `template<typename _Tp >`
`void std::Array_augmented__shift_left (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__shift_left (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__shift_left (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp >`
`void std::Array_augmented__shift_left (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__shift_left (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__shift_right (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp >`
`void std::Array_augmented__shift_right (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`
`void std::Array_augmented__shift_right (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__shift_right (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__shift_right (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__shift_right (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__shift_right (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`
`void std::Array_augmented__shift_right (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__shift_right (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`
`void std::Array_augmented__shift_right (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp >`
`void std::Array_augmented__shift_right (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp, class _Dom >`
`void std::Array_augmented__shift_right (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`

5.459.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<valarray>`.

5.460 valarray_before.h File Reference

Namespaces

- [std](#)

5.460.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<valarray>`.

5.461 vstring.h File Reference

Classes

- class [__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>](#)
- struct [std::hash<__gnu_cxx::__u16vstring>](#)
- struct [std::hash<__gnu_cxx::__u32vstring>](#)
- struct [std::hash<__gnu_cxx::__vstring>](#)
- struct [std::hash<__gnu_cxx::__wvstring>](#)

Namespaces

- [__gnu_cxx](#)
- [std](#)

Functions

- [template<typename _CharT, typename _Traits, typename _Alloc, template<typename, typename, typename> class _Base> basic_istream<_CharT, _Traits> & std::getline\(basic_istream<_CharT, _Traits> & __is, __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base> & __str, _CharT __delim\)](#)
- [template<typename _CharT, typename _Traits, typename _Alloc, template<typename, typename, typename> class _Base> basic_istream<_CharT, _Traits> & std::getline\(basic_istream<_CharT, _Traits> & __is, __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base> & __str\)](#)
- [template<typename _CharT, typename _Traits, typename _Alloc, template<typename, typename, typename> class _Base> bool __gnu_cxx::operator!= \(const __versa_string<_CharT, _Traits, _Alloc, _Base> & __lhs, const __versa_string<_CharT, _Traits, _Alloc, _Base> & __rhs\)](#)
- [template<typename _CharT, typename _Traits, typename _Alloc, template<typename, typename, typename> class _Base> bool __gnu_cxx::operator!= \(const _CharT * __lhs, const __versa_string<_CharT, _Traits, _Alloc, _Base> & __rhs\)](#)
- [template<typename _CharT, typename _Traits, typename _Alloc, template<typename, typename, typename> class _Base> bool __gnu_cxx::operator!= \(const __versa_string<_CharT, _Traits, _Alloc, _Base> & __lhs, const _CharT * __rhs\)](#)
- [template<typename _CharT, typename _Traits, typename _Alloc, template<typename, typename, typename> class _Base> __versa_string<_CharT, _Traits, _Alloc, _Base> __gnu_cxx::operator+ \(const __versa_string<_CharT, _Traits, _Alloc, _Base> & __lhs, const __versa_string<_CharT, _Traits, _Alloc, _Base> & __rhs\)](#)

- [illegible]

- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool __gnu_cxx::operator==(const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_↵`
`string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, template< typename, typename, typename > class _Base>`
`__enable_if< std::is_char< _CharT >::value, bool >::type __gnu_cxx::operator==(const __versa_↵`
`string< _CharT, std::char_traits< _CharT >, std::allocator< _CharT >, _Base > &__lhs, const __versa_string<`
`_CharT, std::char_traits< _CharT >, std::allocator< _CharT >, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool __gnu_cxx::operator==(const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &↵`
`__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool __gnu_cxx::operator==(const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *↵`
`__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool __gnu_cxx::operator>(const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_↵`
`string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool __gnu_cxx::operator>(const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *↵`
`__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool __gnu_cxx::operator>(const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &↵`
`__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool __gnu_cxx::operator>=(const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_↵`
`string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool __gnu_cxx::operator>=(const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT`
`*__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool __gnu_cxx::operator>=(const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base >`
`&__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__is, __gnu_cxx::↵`
`__versa_string< _CharT, _Traits, _Alloc, _Base > &__str)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`void __gnu_cxx::swap (__versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, __versa_string< _CharT, ↵`
`_Traits, _Alloc, _Base > &__rhs)`

5.461.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

5.462 vstring_fwd.h File Reference

Classes

- class `__gnu_cxx::__rc_string_base< _CharT, _Traits, _Alloc >`
- class `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >`

Namespaces

- [__gnu_cxx](#)

Typedefs

- typedef `__versa_string< char, std::char_traits< char >, std::allocator< char >, __rc_string_base >` **`__gnu_cxx::__rc_string`**
- typedef `__vstring` **`__gnu_cxx::__sso_string`**
- typedef `__versa_string< char16_t, std::char_traits< char16_t >, std::allocator< char16_t >, __rc_string_base >` **`__gnu_cxx::__u16rc_string`**
- typedef `__u16vstring` **`__gnu_cxx::__u16sso_string`**
- typedef `__versa_string< char16_t >` **`__gnu_cxx::__u16vstring`**
- typedef `__versa_string< char32_t, std::char_traits< char32_t >, std::allocator< char32_t >, __rc_string_base >` **`__gnu_cxx::__u32rc_string`**
- typedef `__u32vstring` **`__gnu_cxx::__u32sso_string`**
- typedef `__versa_string< char32_t >` **`__gnu_cxx::__u32vstring`**
- typedef `__versa_string< char >` **`__gnu_cxx::__vstring`**
- typedef `__versa_string< wchar_t, std::char_traits< wchar_t >, std::allocator< wchar_t >, __rc_string_base >` **`__gnu_cxx::__wrc_string`**
- typedef `__wvstring` **`__gnu_cxx::__wsso_string`**
- typedef `__versa_string< wchar_t >` **`__gnu_cxx::__wvstring`**

5.462.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ext/vstring.h>`.

5.463 `vstring_util.h` File Reference

Namespaces

- [__gnu_cxx](#)

5.463.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ext/vstring.h>`.

5.464 `workstealing.h` File Reference

Classes

- struct [__gnu_parallel::__Job< _DifferenceTp >](#)

Namespaces

- [__gnu_parallel](#)

Macros

- `#define _GLIBCXX_JOB_VOLATILE`

Functions

- `template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result >
_Op __gnu_parallel::__for_each_template_random_access_workstealing (_RAIter __begin, _RAIter __end, _↵
_Op __op, _Fu &__f, _Red __r, _Result __base, _Result &__output, typename std::iterator_traits< _RAIter >↵
::difference_type __bound)`

5.464.1 Detailed Description

Parallelization of embarrassingly parallel execution by means of work-stealing.

Work stealing is described in

R. D. Blumofe and C. E. Leiserson. Scheduling multithreaded computations by work stealing. *Journal of the ACM*, 46(5):720–748, 1999.

This file is a GNU parallel extension to the Standard C++ Library.

Index

- `_AlgorithmStrategy`
 - `__gnu_parallel`, 254
- `_BALLOC_ALIGN_BYTES`
 - `bitmap_allocator.h`, 2084
- `_BinIndex`
 - `__gnu_parallel`, 253
- `_Bit_scan_forward`
 - `__gnu_cxx`, 226
- `_CASable`
 - `__gnu_parallel`, 253
- `_CASable_bits`
 - `__gnu_parallel`, 293
- `_CASable_mask`
 - `__gnu_parallel`, 293
- `_Construct`
 - `std`, 377
- `_DRandomShufflingGlobalData`
 - `__gnu_parallel::_DRandomShufflingGlobalData`, 662
- `_Destroy`
 - `std`, 377
- `_Distance_precision`
 - `__gnu_debug`, 241
- `_FindAlgorithm`
 - `__gnu_parallel`, 254
- `_GLIBCXX_ASSERTIONS`
 - `compiletime_settings.h`, 2096
- `_GLIBCXX_BAL_QUICKSORT`
 - `features.h`, 2123
- `_GLIBCXX_CALL`
 - `compiletime_settings.h`, 2096
- `_GLIBCXX_DEBUG_VERIFY_AT`
 - `macros.h`, 2163
- `_GLIBCXX_DEQUE_BUF_SIZE`
 - `stl_deque.h`, 2269
- `_GLIBCXX_FIND_CONSTANT_SIZE_BLOCKS`
 - `features.h`, 2123
- `_GLIBCXX_FIND_EQUAL_SPLIT`
 - `features.h`, 2123
- `_GLIBCXX_FIND_GROWING_BLOCKS`
 - `features.h`, 2123
- `_GLIBCXX_MERGESORT`
 - `features.h`, 2123
- `_GLIBCXX_PARALLEL_CONDITION`
 - `settings.h`, 2244
- `_GLIBCXX_PARALLEL_LENGTH`
 - `multiway_merge.h`, 2177
- `_GLIBCXX_PROFILE_DEFINE_UNINIT_DATA`
 - `__gnu_profile`, 299
- `_GLIBCXX_QUICKSORT`
 - `features.h`, 2124
- `_GLIBCXX_RANDOM_SHUFFLE_CONSIDER_L1`
 - `compiletime_settings.h`, 2097
- `_GLIBCXX_RANDOM_SHUFFLE_CONSIDER_TLB`
 - `compiletime_settings.h`, 2097
- `_GLIBCXX_SCALE_DOWN_FPU`
 - `compiletime_settings.h`, 2097
- `_GLIBCXX_TREE_DYNAMIC_BALANCING`
 - `features.h`, 2124
- `_GLIBCXX_TREE_FULL_COPY`
 - `features.h`, 2124
- `_GLIBCXX_TREE_INITIAL_SPLITTING`
 - `features.h`, 2124
- `_GLIBCXX_VERBOSE_LEVEL`
 - `compiletime_settings.h`, 2097
- `_GLIBCXX_VOLATILE`
 - `partition.h`, 2189
 - `queue.h`, 2211
- `_GuardedIterator`
 - `__gnu_parallel::_GuardedIterator`, 668
- `_LoserTreeBase`
 - `__gnu_parallel::_LoserTreeBase`, 685
- `_M_allocate_and_copy`
 - `std::vector`, 2021
- `_M_allocate_single_object`
 - `__gnu_cxx::bitmap_allocator`, 549
- `_M_attach`
 - `__gnu_debug::_Safe_iterator`, 583
 - `__gnu_debug::_Safe_iterator_base`, 591
 - `__gnu_debug::_Safe_local_iterator`, 597
 - `__gnu_debug::_Safe_local_iterator_base`, 605
 - `__gnu_debug::_Safe_sequence`, 608
 - `__gnu_debug::_Safe_sequence_base`, 612
 - `__gnu_debug::_Safe_unordered_container`, 615
 - `__gnu_debug::_Safe_unordered_container_base`, 620
 - `std::__debug::map`, 1059
 - `std::__debug::multimap`, 1063
 - `std::__debug::multiset`, 1068
 - `std::__debug::set`, 1073
- `_M_attach_local`
 - `__gnu_debug::_Safe_unordered_container`, 615
 - `__gnu_debug::_Safe_unordered_container_base`, 620
- `_M_attach_local_single`
 - `__gnu_debug::_Safe_unordered_container`, 615
 - `__gnu_debug::_Safe_unordered_container_base`, 620
- `_M_attach_single`
 - `__gnu_debug::_Safe_iterator`, 583, 584
 - `__gnu_debug::_Safe_iterator_base`, 591
 - `__gnu_debug::_Safe_local_iterator`, 597
 - `__gnu_debug::_Safe_local_iterator_base`, 605

- `__gnu_debug::Safe_sequence`, 608
 - `__gnu_debug::Safe_sequence_base`, 612
 - `__gnu_debug::Safe_unordered_container`, 616
 - `__gnu_debug::Safe_unordered_container_base`, 620
- `std::__debug::map`, 1059
- `std::__debug::multimap`, 1063
- `std::__debug::multiset`, 1068
- `std::__debug::set`, 1073
- `_M_attached_to`
 - `__gnu_debug::Safe_iterator`, 584
 - `__gnu_debug::Safe_iterator_base`, 591
 - `__gnu_debug::Safe_local_iterator`, 597
 - `__gnu_debug::Safe_local_iterator_base`, 605
- `_M_before_dereferenceable`
 - `__gnu_debug::Safe_iterator`, 584
- `_M_begin`
 - `__gnu_parallel::Piece`, 702
- `_M_bin_proc`
 - `__gnu_parallel::DRandomShufflingGlobalData`, 662
- `_M_bins_begin`
 - `__gnu_parallel::DRSSorterPU`, 664
- `_M_can_compare`
 - `__gnu_debug::Safe_iterator`, 584
 - `__gnu_debug::Safe_iterator_base`, 591
 - `__gnu_debug::Safe_local_iterator`, 597
 - `__gnu_debug::Safe_local_iterator_base`, 605
- `_M_clear`
 - `__gnu_cxx::free_list`, 557
- `_M_comp`
 - `__gnu_parallel::LoserTree`, 680
 - `__gnu_parallel::LoserTree< false, _Tp, _Compare >`, 683
 - `__gnu_parallel::LoserTreeBase`, 687
- `_M_const_iterators`
 - `__gnu_debug::Safe_sequence`, 610
 - `__gnu_debug::Safe_sequence_base`, 613
 - `__gnu_debug::Safe_unordered_container`, 617
 - `__gnu_debug::Safe_unordered_container_base`, 622
- `std::__debug::map`, 1060
- `std::__debug::multimap`, 1065
- `std::__debug::multiset`, 1070
- `std::__debug::set`, 1075
- `_M_const_local_iterators`
 - `__gnu_debug::Safe_unordered_container`, 617
 - `__gnu_debug::Safe_unordered_container_base`, 622
- `_M_create_node`
 - `std::list`, 1560
- `_M_data`
 - `std::List_node`, 1149
- `_M_deallocate_single_object`
 - `__gnu_cxx::bitmap_allocator`, 549
- `_M_dereferenceable`
 - `__gnu_debug::Safe_iterator`, 584
 - `__gnu_debug::Safe_local_iterator`, 598
- `_M_detach`
 - `__gnu_debug::Safe_iterator`, 584
 - `__gnu_debug::Safe_iterator_base`, 592
 - `__gnu_debug::Safe_local_iterator`, 598
 - `__gnu_debug::Safe_local_iterator_base`, 605
 - `__gnu_debug::Safe_sequence`, 609
 - `__gnu_debug::Safe_sequence_base`, 612
 - `__gnu_debug::Safe_unordered_container`, 616
 - `__gnu_debug::Safe_unordered_container_base`, 620
- `std::__debug::map`, 1059
- `std::__debug::multimap`, 1064
- `std::__debug::multiset`, 1068
- `std::__debug::set`, 1073
- `_M_detach_all`
 - `__gnu_debug::Safe_sequence`, 609
 - `__gnu_debug::Safe_sequence_base`, 612
 - `__gnu_debug::Safe_unordered_container`, 616
 - `__gnu_debug::Safe_unordered_container_base`, 620
- `std::__debug::map`, 1059
- `std::__debug::multimap`, 1064
- `std::__debug::multiset`, 1069
- `std::__debug::set`, 1074
- `_M_detach_local`
 - `__gnu_debug::Safe_unordered_container`, 616
 - `__gnu_debug::Safe_unordered_container_base`, 620
- `_M_detach_local_single`
 - `__gnu_debug::Safe_unordered_container`, 616
 - `__gnu_debug::Safe_unordered_container_base`, 620
- `_M_detach_single`
 - `__gnu_debug::Safe_iterator`, 585
 - `__gnu_debug::Safe_iterator_base`, 592
 - `__gnu_debug::Safe_local_iterator`, 598
 - `__gnu_debug::Safe_local_iterator_base`, 605
 - `__gnu_debug::Safe_sequence`, 609
 - `__gnu_debug::Safe_sequence_base`, 612
 - `__gnu_debug::Safe_unordered_container`, 616
 - `__gnu_debug::Safe_unordered_container_base`, 620
- `std::__debug::map`, 1059
- `std::__debug::multimap`, 1064
- `std::__debug::multiset`, 1069
- `std::__debug::set`, 1074
- `_M_detach_singular`
 - `__gnu_debug::Safe_sequence`, 609
 - `__gnu_debug::Safe_sequence_base`, 612
 - `__gnu_debug::Safe_unordered_container`, 616

- __gnu_debug::Safe_unordered_container_base, 621
- std::__debug::map, 1059
- std::__debug::multimap, 1064
- std::__debug::multiset, 1069
- std::__debug::set, 1074
- _M_dist
 - __gnu_parallel::DRandomShufflingGlobalData, 662
- _M_elements_leftover
 - __gnu_parallel::QSBThreadLocal, 711
- _M_end
 - __gnu_parallel::Piece, 702
- _M_fill_initialize
 - std::deque, 1411
- _M_finish_iterator
 - __gnu_parallel::__accumulate_selector, 624
 - __gnu_parallel::__adjacent_difference_selector, 625
 - __gnu_parallel::__count_if_selector, 631
 - __gnu_parallel::__count_selector, 632
 - __gnu_parallel::__fill_selector, 634
 - __gnu_parallel::__for_each_selector, 638
 - __gnu_parallel::__generate_selector, 640
 - __gnu_parallel::__generic_for_each_selector, 643
 - __gnu_parallel::__identity_selector, 644
 - __gnu_parallel::__inner_product_selector, 646
 - __gnu_parallel::__replace_if_selector, 654
 - __gnu_parallel::__replace_selector, 656
 - __gnu_parallel::__transform1_selector, 658
 - __gnu_parallel::__transform2_selector, 659
- _M_first
 - __gnu_parallel::Job, 673
- _M_first_insert
 - __gnu_parallel::LoserTree, 680
 - __gnu_parallel::LoserTree< false, _Tp, _Compare >, 683
 - __gnu_parallel::LoserTreeBase, 687
- _M_get
 - __gnu_cxx::free_list, 557
- _M_get_mutex
 - __gnu_debug::Safe_iterator, 585
 - __gnu_debug::Safe_iterator_base, 592
 - __gnu_debug::Safe_local_iterator, 598
 - __gnu_debug::Safe_local_iterator_base, 606
 - __gnu_debug::Safe_sequence, 609
 - __gnu_debug::Safe_sequence_base, 613
 - __gnu_debug::Safe_unordered_container, 616
 - __gnu_debug::Safe_unordered_container_base, 621
 - std::__debug::map, 1059
 - std::__debug::multimap, 1064
 - std::__debug::multiset, 1069
 - std::__debug::set, 1074
- _M_getloc
 - std::basic_ios, 1184
- std::ios_base, 1527
- _M_global
 - __gnu_parallel::QSBThreadLocal, 711
- _M_in_same_bucket
 - __gnu_debug::Safe_local_iterator, 598
- _M_incrementable
 - __gnu_debug::Safe_iterator, 585
 - __gnu_debug::Safe_local_iterator, 598
- _M_initial
 - __gnu_parallel::QSBThreadLocal, 711
- _M_initialize_map
 - std::Deque_base, 1129
 - std::deque, 1412
- _M_insert
 - __gnu_cxx::free_list, 558
- _M_invalidate
 - __gnu_debug::Safe_iterator, 585
 - __gnu_debug::Safe_iterator_base, 592
 - __gnu_debug::Safe_local_iterator, 599
 - __gnu_debug::Safe_local_iterator_base, 606
- _M_invalidate_all
 - __gnu_debug::Safe_sequence, 609
 - __gnu_debug::Safe_sequence_base, 613
 - __gnu_debug::Safe_unordered_container, 616
 - __gnu_debug::Safe_unordered_container_base, 621
 - std::__debug::map, 1059
 - std::__debug::multimap, 1064
 - std::__debug::multiset, 1069
 - std::__debug::set, 1074
- _M_invalidate_if
 - __gnu_debug::Safe_sequence, 609
 - __gnu_debug::Safe_unordered_container, 617
 - std::__debug::map, 1060
 - std::__debug::multimap, 1064
 - std::__debug::multiset, 1069
 - std::__debug::set, 1074
- _M_invalidate_local_if
 - __gnu_debug::Safe_unordered_container, 617
- _M_is_before_begin
 - __gnu_debug::Safe_iterator, 585
- _M_is_begin
 - __gnu_debug::Safe_iterator, 585
 - __gnu_debug::Safe_local_iterator, 599
- _M_is_binnest
 - __gnu_debug::Safe_iterator, 585
- _M_is_end
 - __gnu_debug::Safe_iterator, 586
 - __gnu_debug::Safe_local_iterator, 599
- _M_iterators
 - __gnu_debug::Safe_sequence, 610
 - __gnu_debug::Safe_sequence_base, 613
 - __gnu_debug::Safe_unordered_container, 618

- __gnu_debug::Safe_unordered_container_base, 622
- std::__debug::map, 1060
- std::__debug::multimap, 1065
- std::__debug::multiset, 1070
- std::__debug::set, 1075
- _M_key
 - __gnu_parallel::LoserTreeBase::_Loser, 688
- _M_last
 - __gnu_parallel::Job, 673
- _M_leftover_parts
 - __gnu_parallel::QSBThreadLocal, 711
- _M_load
 - __gnu_parallel::Job, 674
- _M_local_iterators
 - __gnu_debug::Safe_unordered_container, 618
 - __gnu_debug::Safe_unordered_container_base, 622
- _M_log_k
 - __gnu_parallel::LoserTree, 680
 - __gnu_parallel::LoserTree< false, _Tp, _Compare >, 684
 - __gnu_parallel::LoserTreeBase, 687
- _M_losers
 - __gnu_parallel::LoserTree, 681
 - __gnu_parallel::LoserTree< false, _Tp, _Compare >, 684
 - __gnu_parallel::LoserTreeBase, 687
- _M_new_elements_at_back
 - std::deque, 1412
- _M_new_elements_at_front
 - std::deque, 1412
- _M_next
 - __gnu_debug::Safe_iterator, 589
 - __gnu_debug::Safe_iterator_base, 593
 - __gnu_debug::Safe_local_iterator, 602
 - __gnu_debug::Safe_local_iterator_base, 607
- _M_num_bins
 - __gnu_parallel::DRandomShufflingGlobalData, 662
- _M_num_bits
 - __gnu_parallel::DRandomShufflingGlobalData, 662
- _M_num_threads
 - __gnu_parallel::DRSSorterPU, 664
 - __gnu_parallel::PMWMSSortingData, 705
 - __gnu_parallel::QSBThreadLocal, 711
- _M_offsets
 - __gnu_parallel::PMWMSSortingData, 705
- _M_pieces
 - __gnu_parallel::PMWMSSortingData, 705
- _M_pop_back_aux
 - std::deque, 1412
- _M_pop_front_aux
 - std::deque, 1412
- _M_prior
 - __gnu_debug::Safe_iterator, 589
 - __gnu_debug::Safe_iterator_base, 593
 - __gnu_debug::Safe_local_iterator, 602
 - __gnu_debug::Safe_local_iterator_base, 607
- _M_push_back_aux
 - std::deque, 1413
- _M_push_front_aux
 - std::deque, 1413
- _M_range_check
 - std::deque, 1413
 - std::vector, 2021
- _M_range_initialize
 - std::deque, 1413
- _M_reallocate_map
 - std::deque, 1414
- _M_reserve_elements_at_back
 - std::deque, 1414
- _M_reserve_elements_at_front
 - std::deque, 1414
- _M_reserve_map_at_back
 - std::deque, 1414
- _M_reserve_map_at_front
 - std::deque, 1414
- _M_reset
 - __gnu_debug::Safe_iterator, 586
 - __gnu_debug::Safe_iterator_base, 592
 - __gnu_debug::Safe_local_iterator, 599
 - __gnu_debug::Safe_local_iterator_base, 606
- _M_revalidate_singular
 - __gnu_debug::Safe_sequence, 609
 - __gnu_debug::Safe_sequence_base, 613
 - __gnu_debug::Safe_unordered_container, 617
 - __gnu_debug::Safe_unordered_container_base, 621
 - std::__debug::map, 1060
 - std::__debug::multimap, 1064
 - std::__debug::multiset, 1069
 - std::__debug::set, 1074
- _M_samples
 - __gnu_parallel::PMWMSSortingData, 705
- _M_sd
 - __gnu_parallel::DRSSorterPU, 664
- _M_seed
 - __gnu_parallel::DRSSorterPU, 664
- _M_sequence
 - __gnu_debug::Safe_iterator, 589
 - __gnu_debug::Safe_iterator_base, 593
 - __gnu_debug::Safe_local_iterator, 602
 - __gnu_debug::Safe_local_iterator_base, 607
- _M_sequential_algorithm
 - __gnu_parallel::__adjacent_find_selector, 626
 - __gnu_parallel::__find_first_of_selector, 635
 - __gnu_parallel::__find_if_selector, 636
 - __gnu_parallel::__mismatch_selector, 649

- `_M_set_node`
 - `std::__Deque_iterator`, 1131
- `_M_singular`
 - `__gnu_debug::__Safe_iterator`, 586
 - `__gnu_debug::__Safe_iterator_base`, 592
 - `__gnu_debug::__Safe_local_iterator`, 599
 - `__gnu_debug::__Safe_local_iterator_base`, 606
- `_M_source`
 - `__gnu_parallel::__DRandomShufflingGlobalData`, 662
 - `__gnu_parallel::__LoserTreeBase::__Loser`, 688
 - `__gnu_parallel::__PMWMSSortingData`, 705
- `_M_starts`
 - `__gnu_parallel::__DRandomShufflingGlobalData`, 663
 - `__gnu_parallel::__PMWMSSortingData`, 705
- `_M_sup`
 - `__gnu_parallel::__LoserTreeBase::__Loser`, 688
- `_M_swap`
 - `__gnu_debug::__Safe_sequence`, 610
 - `__gnu_debug::__Safe_sequence_base`, 613
 - `__gnu_debug::__Safe_unordered_container`, 617
 - `__gnu_debug::__Safe_unordered_container_base`, 621
 - `std::__debug::map`, 1060
 - `std::__debug::multimap`, 1065
 - `std::__debug::multiset`, 1069
 - `std::__debug::set`, 1074
- `_M_temporaries`
 - `__gnu_parallel::__DRandomShufflingGlobalData`, 663
- `_M_temporary`
 - `__gnu_parallel::__PMWMSSortingData`, 705
- `_M_transfer_from_if`
 - `__gnu_debug::__Safe_sequence`, 610
 - `std::__debug::map`, 1060
 - `std::__debug::multimap`, 1065
 - `std::__debug::multiset`, 1070
 - `std::__debug::set`, 1075
- `_M_unlink`
 - `__gnu_debug::__Safe_iterator`, 586
 - `__gnu_debug::__Safe_iterator_base`, 592
 - `__gnu_debug::__Safe_local_iterator`, 600
 - `__gnu_debug::__Safe_local_iterator_base`, 606
- `_M_use_pointer`
 - `__gnu_parallel::__LoserTreeTraits`, 696
- `_M_version`
 - `__gnu_debug::__Safe_iterator`, 589
 - `__gnu_debug::__Safe_iterator_base`, 593
 - `__gnu_debug::__Safe_local_iterator`, 602
 - `__gnu_debug::__Safe_local_iterator_base`, 607
 - `__gnu_debug::__Safe_sequence`, 610
 - `__gnu_debug::__Safe_sequence_base`, 613
 - `__gnu_debug::__Safe_unordered_container`, 618
 - `__gnu_debug::__Safe_unordered_container_base`, 622
 - `std::__debug::map`, 1060
 - `std::__debug::multimap`, 1065
 - `std::__debug::multiset`, 1070
 - `std::__debug::set`, 1075
- `_MultiwayMergeAlgorithm`
 - `__gnu_parallel`, 254
- `_Opcode`
 - Base and Implementation Classes, 21
- `_Parallelism`
 - `__gnu_parallel`, 254
- `_PartialSumAlgorithm`
 - `__gnu_parallel`, 254
- `_Piece`
 - `__gnu_parallel::__QSBThreadLocal`, 710
- `_PseudoSequence`
 - `__gnu_parallel::__PseudoSequence`, 706
- `_QSBThreadLocal`
 - `__gnu_parallel::__QSBThreadLocal`, 710
- `_RandomNumber`
 - `__gnu_parallel::__RandomNumber`, 712
- `_RestrictedBoundedConcurrentQueue`
 - `__gnu_parallel::__RestrictedBoundedConcurrentQueue`, 714
- `_Safe_iterator`
 - `__gnu_debug::__Safe_iterator`, 582, 583
- `_Safe_iterator_base`
 - `__gnu_debug::__Safe_iterator_base`, 590, 591
- `_Safe_local_iterator`
 - `__gnu_debug::__Safe_local_iterator`, 596
- `_Safe_local_iterator_base`
 - `__gnu_debug::__Safe_local_iterator_base`, 604
- `_SequenceIndex`
 - `__gnu_parallel`, 253
- `_SortAlgorithm`
 - `__gnu_parallel`, 254
- `_SplittingAlgorithm`
 - `__gnu_parallel`, 255
- `_Temporary_buffer`
 - `std::__Temporary_buffer`, 1151
- `_ThreadIndex`
 - `__gnu_parallel`, 253
- `_TokenT`
 - `std::__detail::__Scanner`, 1118
- `__addressof`
 - Utilities, 212
- `__allocator_base`
 - Allocators, 6
- `__base`
 - `__gnu_debug`, 242
- `__begin1_iterator`
 - `__gnu_parallel::__inner_product_selector`, 646
- `__begin2_iterator`
 - `__gnu_parallel::__inner_product_selector`, 646
- `__bins_end`
 - `__gnu_parallel::__DRSSorterPU`, 664

- `__bit_allocate`
 - `__gnu_cxx::__detail`, 236
- `__bit_free`
 - `__gnu_cxx::__detail`, 236
- `__calc_borders`
 - `__gnu_parallel`, 255
- `__check_dereferenceable`
 - `__gnu_debug`, 242
- `__check_singular`
 - `__gnu_debug`, 242
- `__check_singular_aux`
 - `__gnu_debug`, 242
- `__check_string`
 - `__gnu_debug`, 243
- `__compare_and_swap`
 - `__gnu_parallel`, 255
- `__ctype_type`
 - `std::basic_ios`, 1180
- `__cxa_demangle`
 - `cxxabi.h`, 2107
- `__cxxabiv1::__forced_unwind`, 458
- `__decode2`
 - `__gnu_parallel`, 256
- `__delete_min_insert`
 - `__gnu_parallel::_LoserTree`, 679
 - `__gnu_parallel::_LoserTree< false, _Tp, _Compare >`, 682
- `__determine_samples`
 - `__gnu_parallel`, 256
- `__encode2`
 - `__gnu_parallel`, 256
- `__env_t`
 - `__gnu_profile`, 299
- `__equally_split`
 - `__gnu_parallel`, 257
- `__equally_split_point`
 - `__gnu_parallel`, 257
- `__fetch_and_add`
 - `__gnu_parallel`, 258
- `__final_insertion_sort`
 - `std`, 370
- `__find_if`
 - `std`, 370
- `__find_if_not`
 - `std`, 370
- `__find_if_not_n`
 - `std`, 370
- `__find_template`
 - `__gnu_parallel`, 258–260
- `__for_each_template_random_access`
 - `__gnu_parallel`, 260
- `__for_each_template_random_access_ed`
 - `__gnu_parallel`, 261
- `__for_each_template_random_access_omp_loop`
 - `__gnu_parallel`, 261
- `__for_each_template_random_access_omp_loop_static`
 - `__gnu_parallel`, 262
- `__for_each_template_random_access_workstealing`
 - `__gnu_parallel`, 262
- `__foreign_iterator_aux2`
 - `__gnu_debug`, 243
- `__gcd`
 - `std`, 371
- `__genrand_bits`
 - `__gnu_parallel::_RandomNumber`, 712
- `__get_distance`
 - `__gnu_debug`, 243
- `__get_min_source`
 - `__gnu_parallel::_LoserTree`, 679
 - `__gnu_parallel::_LoserTree< false, _Tp, _Compare >`, 682
 - `__gnu_parallel::_LoserTreeBase`, 686
- `__get_num_threads`
 - `__gnu_parallel::balanced_quicksort_tag`, 725
 - `__gnu_parallel::balanced_tag`, 726
 - `__gnu_parallel::default_parallel_tag`, 728
 - `__gnu_parallel::exact_tag`, 730
 - `__gnu_parallel::multiway_mergesort_exact_tag`, 733
 - `__gnu_parallel::multiway_mergesort_sampling_tag`, 734
 - `__gnu_parallel::multiway_mergesort_tag`, 736
 - `__gnu_parallel::omp_loop_static_tag`, 737
 - `__gnu_parallel::omp_loop_tag`, 738
 - `__gnu_parallel::parallel_tag`, 741
 - `__gnu_parallel::quicksort_tag`, 743
 - `__gnu_parallel::sampling_tag`, 744
 - `__gnu_parallel::unbalanced_tag`, 745
- `__glibcxx_check_erase`
 - `macros.h`, 2161
- `__glibcxx_check_erase_after`
 - `macros.h`, 2161
- `__glibcxx_check_erase_range`
 - `macros.h`, 2161
- `__glibcxx_check_erase_range_after`
 - `macros.h`, 2161
- `__glibcxx_check_heap_pred`
 - `macros.h`, 2162
- `__glibcxx_check_insert`
 - `macros.h`, 2162
- `__glibcxx_check_insert_after`
 - `macros.h`, 2162
- `__glibcxx_check_insert_range`
 - `macros.h`, 2162
- `__glibcxx_check_insert_range_after`
 - `macros.h`, 2162
- `__glibcxx_check_partitioned_lower`
 - `macros.h`, 2162
- `__glibcxx_check_partitioned_lower_pred`

- macros.h, [2163](#)
- __glibcxx_check_partitioned_upper_pred
 - macros.h, [2163](#)
- __glibcxx_check_sorted_pred
 - macros.h, [2163](#)
- __gnu_cxx, [218](#)
 - _Bit_scan_forward, [226](#)
 - __static_pointer_cast, [226](#)
 - operator!=, [226](#), [227](#)
 - operator<, [229](#), [230](#)
 - operator<=, [230](#), [231](#)
 - operator>, [233](#)
 - operator>=, [234](#), [235](#)
 - operator+, [227](#), [228](#)
 - operator==, [231](#), [232](#)
 - swap, [235](#)
- __gnu_cxx::Caster<_ToType>, [536](#)
- __gnu_cxx::Char_types<_CharT>, [537](#)
- __gnu_cxx::ExtPtr_allocator<_Tp>, [537](#)
- __gnu_cxx::Invalid_type, [539](#)
- __gnu_cxx::Pointer_adapter<_Storage_policy>, [539](#)
- __gnu_cxx::Relative_pointer_impl<_Tp>, [541](#)
- __gnu_cxx::Relative_pointer_impl<const _Tp>, [542](#)
- __gnu_cxx::Std_pointer_impl<_Tp>, [543](#)
- __gnu_cxx::Unqualified_type<_Tp>, [543](#)
- __gnu_cxx::__alloc_traits
 - allocate, [461](#)
 - const_void_pointer, [460](#)
 - construct, [462](#)
 - deallocate, [462](#)
 - destroy, [462](#)
 - max_size, [463](#)
 - propagate_on_container_copy_assignment, [460](#)
 - propagate_on_container_move_assignment, [460](#)
 - propagate_on_container_swap, [461](#)
 - select_on_container_copy_construction, [463](#)
 - void_pointer, [461](#)
- __gnu_cxx::__alloc_traits<_Alloc>, [459](#)
- __gnu_cxx::__common_pool_policy<_PoolTp, _Thread>, [464](#)
- __gnu_cxx::__detail, [235](#)
 - __bit_allocate, [236](#)
 - __bit_free, [236](#)
 - __num_bitmaps, [236](#)
 - __num_blocks, [236](#)
- __gnu_cxx::__detail::Bitmap_counter<_Tp>, [465](#)
- __gnu_cxx::__detail::Ffit_finder
 - argument_type, [466](#)
 - result_type, [466](#)
- __gnu_cxx::__detail::Ffit_finder<_Tp>, [466](#)
- __gnu_cxx::__detail::__mini_vector<_Tp>, [464](#)
- __gnu_cxx::__mt_alloc<_Tp, _Poolp>, [467](#)
- __gnu_cxx::__mt_alloc_base<_Tp>, [468](#)
- __gnu_cxx::__per_type_pool_policy<_Tp, _PoolTp, _Thread>, [469](#)
- __gnu_cxx::__pool<_Thread>, [470](#)
- __gnu_cxx::__pool<false>, [470](#)
- __gnu_cxx::__pool<true>, [471](#)
- __gnu_cxx::__pool_alloc<_Tp>, [472](#)
- __gnu_cxx::__pool_alloc_base, [474](#)
- __gnu_cxx::__pool_base, [475](#)
- __gnu_cxx::__rc_string_base<_CharT, _Traits, _Alloc>, [476](#)
- __gnu_cxx::__scoped_lock, [479](#)
- __gnu_cxx::__versa_string
 - __versa_string, [483–485](#)
 - ~__versa_string, [486](#)
 - append, [486](#), [487](#), [489](#), [490](#)
 - assign, [490–493](#)
 - at, [494](#)
 - back, [495](#)
 - begin, [495](#)
 - c_str, [496](#)
 - capacity, [496](#)
 - cbegin, [496](#)
 - cend, [496](#)
 - clear, [496](#)
 - compare, [496–499](#)
 - copy, [499](#)
 - crbegin, [500](#)
 - crend, [500](#)
 - data, [500](#)
 - empty, [500](#)
 - end, [501](#)
 - erase, [501](#), [502](#)
 - find, [502–504](#)
 - find_first_not_of, [504–506](#)
 - find_first_of, [506](#), [507](#)
 - find_last_not_of, [508](#), [509](#)
 - find_last_of, [510](#), [511](#)
 - front, [511](#), [512](#)
 - get_allocator, [512](#)
 - insert, [512–517](#)
 - length, [517](#)
 - max_size, [518](#)
 - npos, [536](#)
 - operator+=, [518](#), [519](#)
 - operator=, [519–521](#)
 - operator[], [521](#), [522](#)
 - pop_back, [522](#)
 - push_back, [522](#)
 - rbegin, [523](#)
 - rend, [523](#)
 - replace, [523–528](#), [530](#), [531](#)
 - reserve, [531](#)
 - resize, [532](#)
 - rfind, [533](#), [534](#)

- shrink_to_fit, 534
- size, 534
- substr, 535
- swap, 535
- __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _←
Base >, 479
- __gnu_cxx::annotate_base, 544
- __gnu_cxx::array_allocator< _Tp, _Array >, 545
- __gnu_cxx::array_allocator_base< _Tp >, 546
- __gnu_cxx::bitmap_allocator
 - _M_allocate_single_object, 549
 - _M_deallocate_single_object, 549
- __gnu_cxx::bitmap_allocator< _Tp >, 547
- __gnu_cxx::char_traits< _CharT >, 550
- __gnu_cxx::character< _Value, _Int, _St >, 551
- __gnu_cxx::condition_base, 552
- __gnu_cxx::debug_allocator< _Alloc >, 552
- __gnu_cxx::enc_filebuf< _CharT >, 553
- __gnu_cxx::encoding_char_traits< _CharT >, 554
- __gnu_cxx::encoding_state, 555
- __gnu_cxx::forced_error, 556
- __gnu_cxx::free_list, 557
 - _M_clear, 557
 - _M_get, 557
 - _M_insert, 558
- __gnu_cxx::limit_condition, 558
- __gnu_cxx::limit_condition::always_adjustor, 559
- __gnu_cxx::limit_condition::limit_adjustor, 559
- __gnu_cxx::limit_condition::never_adjustor, 560
- __gnu_cxx::malloc_allocator< _Tp >, 560
- __gnu_cxx::new_allocator< _Tp >, 561
- __gnu_cxx::random_condition, 563
- __gnu_cxx::random_condition::always_adjustor, 563
- __gnu_cxx::random_condition::group_adjustor, 564
- __gnu_cxx::random_condition::never_adjustor, 564
- __gnu_cxx::recursive_init_error, 565
- __gnu_cxx::stdio_filebuf
 - ~stdio_filebuf, 567
 - fd, 567
 - file, 567
 - stdio_filebuf, 566
- __gnu_cxx::stdio_filebuf< _CharT, _Traits >, 565
- __gnu_cxx::stdio_sync_filebuf
 - file, 569
- __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 568
- __gnu_cxx::throw_allocator_base< _Tp, _Cond >, 569
- __gnu_cxx::throw_allocator_limit< _Tp >, 571
- __gnu_cxx::throw_allocator_random< _Tp >, 573
- __gnu_cxx::throw_value_base< _Cond >, 574
- __gnu_cxx::throw_value_limit, 576
- __gnu_cxx::throw_value_random, 577
- __gnu_cxx::typelist, 237
 - apply_generator, 237
- __gnu_debug, 237
 - _Distance_precision, 241
 - __base, 242
 - __check_dereferenceable, 242
 - __check_singular, 242
 - __check_singular_aux, 242
 - __check_string, 243
 - __foreign_iterator_aux2, 243
 - __get_distance, 243
 - __valid_range, 243, 244
 - __valid_range_aux, 244
 - __valid_range_aux2, 244
- __gnu_debug:: After_nth_from< _Iterator >, 578
- __gnu_debug:: BeforeBeginHelper< _Sequence >, 579
- __gnu_debug:: Equal_to< _Type >, 579
- __gnu_debug:: Not_equal_to< _Type >, 580
- __gnu_debug:: Safe_iterator
 - _M_attach, 583
 - _M_attach_single, 583, 584
 - _M_attached_to, 584
 - _M_before_dereferenceable, 584
 - _M_can_compare, 584
 - _M_dereferenceable, 584
 - _M_detach, 584
 - _M_detach_single, 585
 - _M_get_mutex, 585
 - _M_incrementable, 585
 - _M_invalidate, 585
 - _M_is_before_begin, 585
 - _M_is_begin, 585
 - _M_is_beginnest, 585
 - _M_is_end, 586
 - _M_next, 589
 - _M_prior, 589
 - _M_reset, 586
 - _M_sequence, 589
 - _M_singular, 586
 - _M_unlink, 586
 - _M_version, 589
- base, 586
- operator _Iterator, 586
- operator*, 587
- operator++, 587
- operator->, 588
- operator--, 587
- operator=, 588
- __gnu_debug:: Safe_iterator< _Iterator, _Sequence >, 580
- __gnu_debug:: Safe_iterator_base, 589
 - _M_attach, 591
 - _M_attach_single, 591
 - _M_attached_to, 591
 - _M_can_compare, 591
 - _M_detach, 592

- [_M_detach_single](#), 592
- [_M_get_mutex](#), 592
- [_M_invalidate](#), 592
- [_M_next](#), 593
- [_M_prior](#), 593
- [_M_reset](#), 592
- [_M_sequence](#), 593
- [_M_singular](#), 592
- [_M_unlink](#), 592
- [_M_version](#), 593
- [_Safe_iterator_base](#), 590, 591
- [__gnu_debug:: Safe_local_iterator](#)
 - [_M_attach](#), 597
 - [_M_attach_single](#), 597
 - [_M_attached_to](#), 597
 - [_M_can_compare](#), 597
 - [_M_dereferenceable](#), 598
 - [_M_detach](#), 598
 - [_M_detach_single](#), 598
 - [_M_get_mutex](#), 598
 - [_M_in_same_bucket](#), 598
 - [_M_incrementable](#), 598
 - [_M_invalidate](#), 599
 - [_M_is_begin](#), 599
 - [_M_is_end](#), 599
 - [_M_next](#), 602
 - [_M_prior](#), 602
 - [_M_reset](#), 599
 - [_M_sequence](#), 602
 - [_M_singular](#), 599
 - [_M_unlink](#), 600
 - [_M_version](#), 602
 - [_Safe_local_iterator](#), 596
 - [base](#), 600
 - [bucket](#), 600
 - [operator_iterator](#), 600
 - [operator*](#), 600
 - [operator++](#), 601
 - [operator->](#), 601
 - [operator=](#), 601
- [__gnu_debug:: Safe_local_iterator< _Iterator, Sequence >](#), 594
- [__gnu_debug:: Safe_iterator_base](#), 603
 - [_M_attach](#), 605
 - [_M_attach_single](#), 605
 - [_M_attached_to](#), 605
 - [_M_can_compare](#), 605
 - [_M_detach](#), 605
 - [_M_detach_single](#), 605
 - [_M_get_mutex](#), 606
 - [_M_invalidate](#), 606
 - [_M_next](#), 607
 - [_M_prior](#), 607
 - [_M_reset](#), 606
 - [_M_sequence](#), 607
 - [_M_singular](#), 606
 - [_M_unlink](#), 606
 - [_M_version](#), 607
 - [_Safe_local_iterator_base](#), 604
- [__gnu_debug:: Safe_sequence](#)
 - [_M_attach](#), 608
 - [_M_attach_single](#), 608
 - [_M_const_iterators](#), 610
 - [_M_detach](#), 609
 - [_M_detach_all](#), 609
 - [_M_detach_single](#), 609
 - [_M_detach_singular](#), 609
 - [_M_get_mutex](#), 609
 - [_M_invalidate_all](#), 609
 - [_M_invalidate_if](#), 609
 - [_M_iterators](#), 610
 - [_M_revalidate_singular](#), 609
 - [_M_swap](#), 610
 - [_M_transfer_from_if](#), 610
 - [_M_version](#), 610
- [__gnu_debug:: Safe_sequence< Sequence >](#), 607
- [__gnu_debug:: Safe_sequence_base](#), 611
 - [_M_attach](#), 612
 - [_M_attach_single](#), 612
 - [_M_const_iterators](#), 613
 - [_M_detach](#), 612
 - [_M_detach_all](#), 612
 - [_M_detach_single](#), 612
 - [_M_detach_singular](#), 612
 - [_M_get_mutex](#), 613
 - [_M_invalidate_all](#), 613
 - [_M_iterators](#), 613
 - [_M_revalidate_singular](#), 613
 - [_M_swap](#), 613
 - [_M_version](#), 613
 - [~ Safe_sequence_base](#), 612
- [__gnu_debug:: Safe_unordered_container](#)
 - [_M_attach](#), 615
 - [_M_attach_local](#), 615
 - [_M_attach_local_single](#), 615
 - [_M_attach_single](#), 616
 - [_M_const_iterators](#), 617
 - [_M_const_local_iterators](#), 617
 - [_M_detach](#), 616
 - [_M_detach_all](#), 616
 - [_M_detach_local](#), 616
 - [_M_detach_local_single](#), 616
 - [_M_detach_single](#), 616
 - [_M_detach_singular](#), 616
 - [_M_get_mutex](#), 616
 - [_M_invalidate_all](#), 616
 - [_M_invalidate_if](#), 617
 - [_M_invalidate_local_if](#), 617

- [_M_iterators](#), 618
- [_M_local_iterators](#), 618
- [_M_revalidate_singular](#), 617
- [_M_swap](#), 617
- [_M_version](#), 618
- [__gnu_debug::_Safe_unordered_container< _Container](#)
[>](#), 614
- [__gnu_debug::_Safe_unordered_container_base](#), 618
- [_M_attach](#), 620
- [_M_attach_local](#), 620
- [_M_attach_local_single](#), 620
- [_M_attach_single](#), 620
- [_M_const_iterators](#), 622
- [_M_const_local_iterators](#), 622
- [_M_detach](#), 620
- [_M_detach_all](#), 620
- [_M_detach_local](#), 620
- [_M_detach_local_single](#), 620
- [_M_detach_single](#), 620
- [_M_detach_singular](#), 621
- [_M_get_mutex](#), 621
- [_M_invalidate_all](#), 621
- [_M_iterators](#), 622
- [_M_local_iterators](#), 622
- [_M_revalidate_singular](#), 621
- [_M_swap](#), 621
- [_M_version](#), 622
- [~_Safe_unordered_container_base](#), 620
- [__gnu_internal](#), 245
- [__gnu_parallel](#), 245
 - [_AlgorithmStrategy](#), 254
 - [_BinIndex](#), 253
 - [_CASable](#), 253
 - [_CASable_bits](#), 293
 - [_CASable_mask](#), 293
 - [_FindAlgorithm](#), 254
 - [_MultiwayMergeAlgorithm](#), 254
 - [_Parallelism](#), 254
 - [_PartialSumAlgorithm](#), 254
 - [_SequenceIndex](#), 253
 - [_SortAlgorithm](#), 254
 - [_SplittingAlgorithm](#), 255
 - [_ThreadIndex](#), 253
 - [_calc_borders](#), 255
 - [_compare_and_swap](#), 255
 - [_decode2](#), 256
 - [_determine_samples](#), 256
 - [_encode2](#), 256
 - [_equally_split](#), 257
 - [_equally_split_point](#), 257
 - [_fetch_and_add](#), 258
 - [_find_template](#), 258–260
 - [_for_each_template_random_access](#), 260
 - [_for_each_template_random_access_ed](#), 261
 - [_for_each_template_random_access_omp_loop](#),
[261](#)
 - [_for_each_template_random_access_omp_loop↔](#)
[_static](#), 262
 - [_for_each_template_random_access_workstealing](#),
[262](#)
 - [_is_sorted](#), 263
 - [_median_of_three_iterators](#), 263
 - [_merge_advance](#), 264
 - [_merge_advance_movc](#), 264
 - [_merge_advance_usual](#), 265
 - [_parallel_merge_advance](#), 266
 - [_parallel_nth_element](#), 267
 - [_parallel_partial_sort](#), 267
 - [_parallel_partial_sum](#), 267
 - [_parallel_partial_sum_basecase](#), 268
 - [_parallel_partial_sum_linear](#), 268
 - [_parallel_partition](#), 269
 - [_parallel_random_shuffle](#), 269
 - [_parallel_random_shuffle_drs](#), 270
 - [_parallel_random_shuffle_drs_pu](#), 270
 - [_parallel_sort](#), 271–275
 - [_parallel_sort_qs](#), 275
 - [_parallel_sort_qs_conquer](#), 276
 - [_parallel_sort_qs_divide](#), 276
 - [_parallel_sort_qsb](#), 277
 - [_parallel_unique_copy](#), 277
 - [_qsb_conquer](#), 278
 - [_qsb_divide](#), 278
 - [_qsb_local_sort_with_helping](#), 279
 - [_random_number_pow2](#), 279
 - [_rd_log2](#), 280
 - [_round_up_to_pow2](#), 280
 - [_search_template](#), 280
 - [_sequential_multiway_merge](#), 281
 - [_sequential_random_shuffle](#), 281
 - [_shrink](#), 282
 - [_shrink_and_double](#), 282
 - [_yield](#), 282
- [list_partition](#), 282
- [max](#), 283
- [min](#), 283
- [multiseq_partition](#), 283
- [multiseq_selection](#), 284
- [multiway_merge](#), 284
- [multiway_merge_3_variant](#), 286
- [multiway_merge_4_variant](#), 287
- [multiway_merge_exact_splitting](#), 287
- [multiway_merge_loser_tree](#), 287
- [multiway_merge_loser_tree_sentinel](#), 288
- [multiway_merge_loser_tree_unguarded](#), 289
- [multiway_merge_sampling_splitting](#), 289
- [multiway_merge_sentinels](#), 289
- [parallel_balanced](#), 254

- parallel_multiway_merge, [291](#)
- parallel_omp_loop, [254](#)
- parallel_omp_loop_static, [254](#)
- parallel_sort_mwms, [292](#)
- parallel_sort_mwms_pu, [292](#)
- parallel_taskqueue, [254](#)
- parallel_unbalanced, [254](#)
- sequential, [254](#)
- __gnu_parallel::DRSSorterPU
 - _M_bins_begin, [664](#)
 - _M_num_threads, [664](#)
 - _M_sd, [664](#)
 - _M_seed, [664](#)
 - _bins_end, [664](#)
- __gnu_parallel::DRSSorterPU< _RAIter, _Random↵
 - NumberGenerator >, [663](#)
- __gnu_parallel::DRandomShufflingGlobalData
 - _DRandomShufflingGlobalData, [662](#)
 - _M_bin_proc, [662](#)
 - _M_dist, [662](#)
 - _M_num_bins, [662](#)
 - _M_num_bits, [662](#)
 - _M_source, [662](#)
 - _M_starts, [663](#)
 - _M_temporaries, [663](#)
- __gnu_parallel::DRandomShufflingGlobalData< _RAIter
 - >, [661](#)
- __gnu_parallel::DummyReduct, [665](#)
- __gnu_parallel::EqualFromLess
 - first_argument_type, [666](#)
 - result_type, [666](#)
 - second_argument_type, [666](#)
- __gnu_parallel::EqualFromLess< _T1, _T2, _Compare
 - >, [665](#)
- __gnu_parallel::EqualTo
 - first_argument_type, [667](#)
 - result_type, [667](#)
 - second_argument_type, [667](#)
- __gnu_parallel::EqualTo< _T1, _T2 >, [666](#)
- __gnu_parallel::GuardedIterator
 - _GuardedIterator, [668](#)
 - operator _RAIter, [669](#)
 - operator<, [669](#)
 - operator<=, [670](#)
 - operator*, [669](#)
 - operator++, [669](#)
- __gnu_parallel::GuardedIterator< _RAIter, _Compare >,
 - [668](#)
- __gnu_parallel::IteratorPair
 - first, [672](#)
 - second, [672](#)
 - second_type, [671](#)
- __gnu_parallel::IteratorPair< _Iterator1, _Iterator2, _↵
 - IteratorCategory >, [670](#)
- __gnu_parallel::IteratorTriple< _Iterator1, _Iterator2, _↵
 - Iterator3, _IteratorCategory >, [672](#)
- __gnu_parallel::Job
 - _M_first, [673](#)
 - _M_last, [673](#)
 - _M_load, [674](#)
- __gnu_parallel::Job< _DifferenceTp >, [673](#)
- __gnu_parallel::Less
 - first_argument_type, [675](#)
 - result_type, [675](#)
 - second_argument_type, [675](#)
- __gnu_parallel::Less< _T1, _T2 >, [674](#)
- __gnu_parallel::Lexicographic
 - first_argument_type, [676](#)
 - result_type, [676](#)
 - second_argument_type, [677](#)
- __gnu_parallel::Lexicographic< _T1, _T2, _Compare >,
 - [676](#)
- __gnu_parallel::LexicographicReverse
 - first_argument_type, [678](#)
 - result_type, [678](#)
 - second_argument_type, [678](#)
- __gnu_parallel::LexicographicReverse< _T1, _T2, _↵
 - Compare >, [677](#)
- __gnu_parallel::LoserTree
 - _M_comp, [680](#)
 - _M_first_insert, [680](#)
 - _M_log_k, [680](#)
 - _M_losers, [681](#)
 - _delete_min_insert, [679](#)
 - _get_min_source, [679](#)
 - _insert_start, [680](#)
- __gnu_parallel::LoserTree< __stable, _Tp, _Compare >,
 - [678](#)
- __gnu_parallel::LoserTree< false, _Tp, _Compare >,
 - [681](#)
 - _M_comp, [683](#)
 - _M_first_insert, [683](#)
 - _M_log_k, [684](#)
 - _M_losers, [684](#)
 - _delete_min_insert, [682](#)
 - _get_min_source, [682](#)
 - _init_winner, [682](#)
 - _insert_start, [683](#)
- __gnu_parallel::LoserTreeBase
 - _LoserTreeBase, [685](#)
 - _M_comp, [687](#)
 - _M_first_insert, [687](#)
 - _M_log_k, [687](#)
 - _M_losers, [687](#)
 - _get_min_source, [686](#)
 - _insert_start, [686](#)
 - ~_LoserTreeBase, [686](#)
- __gnu_parallel::LoserTreeBase< _Tp, _Compare >, [684](#)

[__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser](#), 688
[__gnu_parallel::_LoserTreeBase::_Loser](#)
 [_M_key](#), 688
 [_M_source](#), 688
 [_M_sup](#), 688
[__gnu_parallel::_LoserTreePointer< __stable, _Tp, _Compare >](#), 689
[__gnu_parallel::_LoserTreePointer< false, _Tp, _Compare >](#), 690
[__gnu_parallel::_LoserTreePointerBase< _Tp, _Compare >](#), 691
[__gnu_parallel::_LoserTreePointerBase< _Tp, _Compare >::_Loser](#), 692
[__gnu_parallel::_LoserTreePointerUnguarded< __stable, _Tp, _Compare >](#), 693
[__gnu_parallel::_LoserTreePointerUnguarded< false, _Tp, _Compare >](#), 694
[__gnu_parallel::_LoserTreePointerUnguardedBase< _Tp, _Compare >](#), 695
[__gnu_parallel::_LoserTreeTraits](#)
 [_M_use_pointer](#), 696
[__gnu_parallel::_LoserTreeTraits< _Tp >](#), 696
[__gnu_parallel::_LoserTreeUnguarded< __stable, _Tp, _Compare >](#), 697
[__gnu_parallel::_LoserTreeUnguarded< false, _Tp, _Compare >](#), 698
[__gnu_parallel::_LoserTreeUnguardedBase< _Tp, _Compare >](#), 699
[__gnu_parallel::_Multiplies](#)
 [first_argument_type](#), 700
 [result_type](#), 700
 [second_argument_type](#), 701
[__gnu_parallel::_Multiplies< _Tp1, _Tp2, _Result >](#), 700
[__gnu_parallel::_Nothing](#), 701
 [operator\(\)](#), 701
[__gnu_parallel::_PMWMSortingData](#)
 [_M_num_threads](#), 705
 [_M_offsets](#), 705
 [_M_pieces](#), 705
 [_M_samples](#), 705
 [_M_source](#), 705
 [_M_starts](#), 705
 [_M_temporary](#), 705
[__gnu_parallel::_PMWMSortingData< _RAIter >](#), 704
[__gnu_parallel::_Piece](#)
 [_M_begin](#), 702
 [_M_end](#), 702
[__gnu_parallel::_Piece< _DifferenceTp >](#), 702
[__gnu_parallel::_Plus](#)
 [first_argument_type](#), 703
 [result_type](#), 703
 [second_argument_type](#), 704
[__gnu_parallel::_Plus< _Tp1, _Tp2, _Result >](#), 703
[__gnu_parallel::_PseudoSequence](#)
 [_PseudoSequence](#), 706
 [begin](#), 707
 [end](#), 707
[__gnu_parallel::_PseudoSequence< _Tp, _DifferenceTp >](#), 706
[__gnu_parallel::_PseudoSequenceIterator< _Tp, _DifferenceTp >](#), 707
[__gnu_parallel::_QSBThreadLocal](#)
 [_M_elements_leftover](#), 711
 [_M_global](#), 711
 [_M_initial](#), 711
 [_M_leftover_parts](#), 711
 [_M_num_threads](#), 711
 [_Piece](#), 710
 [_QSBThreadLocal](#), 710
[__gnu_parallel::_QSBThreadLocal< _RAIter >](#), 709
[__gnu_parallel::_RandomNumber](#), 712
 [_RandomNumber](#), 712
 [_genrand_bits](#), 712
 [operator\(\)](#), 713
[__gnu_parallel::_RestrictedBoundedConcurrentQueue](#)
 [_RestrictedBoundedConcurrentQueue](#), 714
 [~_RestrictedBoundedConcurrentQueue](#), 714
 [pop_back](#), 714
 [pop_front](#), 714
 [push_front](#), 714
[__gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp >](#), 713
[__gnu_parallel::_SamplingSorter< __stable, _RAIter, _Compare >](#), 715
[__gnu_parallel::_SamplingSorter< false, _RAIter, _Compare >](#), 715
[__gnu_parallel::_Settings](#), 716
 [accumulate_minimal_n](#), 717
 [adjacent_difference_minimal_n](#), 717
 [cache_line_size](#), 718
 [count_minimal_n](#), 718
 [fill_minimal_n](#), 718
 [find_increasing_factor](#), 718
 [find_initial_block_size](#), 718
 [find_maximum_block_size](#), 718
 [find_scale_factor](#), 718
 [find_sequential_search_size](#), 719
 [for_each_minimal_n](#), 719
 [generate_minimal_n](#), 719
 [get](#), 717
 [L1_cache_size](#), 719
 [L2_cache_size](#), 719
 [max_element_minimal_n](#), 719
 [merge_minimal_n](#), 719
 [merge_oversampling](#), 720
 [min_element_minimal_n](#), 720
 [multiway_merge_minimal_k](#), 720

- multiway_merge_minimal_n, 720
- multiway_merge_oversampling, 720
- nth_element_minimal_n, 720
- partial_sort_minimal_n, 720
- partial_sum_dilation, 720
- partial_sum_minimal_n, 721
- partition_chunk_share, 721
- partition_chunk_size, 721
- partition_minimal_n, 721
- qsb_steals, 721
- random_shuffle_minimal_n, 721
- replace_minimal_n, 721
- search_minimal_n, 722
- set, 717
- set_difference_minimal_n, 722
- set_intersection_minimal_n, 722
- set_symmetric_difference_minimal_n, 722
- set_union_minimal_n, 722
- sort_minimal_n, 722
- sort_mwms_oversampling, 722
- sort_qs_num_samples_preset, 722
- sort_qsb_base_case_maximal_n, 723
- TLB_size, 723
- transform_minimal_n, 723
- unique_copy_minimal_n, 723
- __gnu_parallel::SplitConsistently< __exact, _RAIter, _Compare, _SortingPlacesIter >, 723
- __gnu_parallel::SplitConsistently< false, _RAIter, _Compare, _SortingPlacesIter >, 724
- __gnu_parallel::SplitConsistently< true, _RAIter, _Compare, _SortingPlacesIter >, 724
- __gnu_parallel::__accumulate_binop_reduct< _BinOp >, 622
- __gnu_parallel::__accumulate_selector
 - _M_finish_iterator, 624
 - operator(), 624
- __gnu_parallel::__accumulate_selector< _It >, 623
- __gnu_parallel::__adjacent_difference_selector
 - _M_finish_iterator, 625
- __gnu_parallel::__adjacent_difference_selector< _It >, 624
- __gnu_parallel::__adjacent_find_selector, 625
 - _M_sequential_algorithm, 626
 - operator(), 626
- __gnu_parallel::__binder1st
 - argument_type, 628
 - result_type, 628
- __gnu_parallel::__binder1st< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >, 627
- __gnu_parallel::__binder2nd
 - argument_type, 629
 - result_type, 629
- __gnu_parallel::__binder2nd< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >, 628
- __gnu_parallel::__count_if_selector
 - _M_finish_iterator, 631
 - operator(), 630
- __gnu_parallel::__count_if_selector< _It, _Diff >, 630
- __gnu_parallel::__count_selector
 - _M_finish_iterator, 632
 - operator(), 632
- __gnu_parallel::__count_selector< _It, _Diff >, 631
- __gnu_parallel::__fill_selector
 - _M_finish_iterator, 634
 - operator(), 633
- __gnu_parallel::__fill_selector< _It >, 633
- __gnu_parallel::__find_first_of_selector
 - _M_sequential_algorithm, 635
 - operator(), 635
- __gnu_parallel::__find_first_of_selector< _Filterator >, 634
- __gnu_parallel::__find_if_selector, 636
 - _M_sequential_algorithm, 636
 - operator(), 637
- __gnu_parallel::__for_each_selector
 - _M_finish_iterator, 638
 - operator(), 638
- __gnu_parallel::__for_each_selector< _It >, 637
- __gnu_parallel::__generate_selector
 - _M_finish_iterator, 640
 - operator(), 639
- __gnu_parallel::__generate_selector< _It >, 639
- __gnu_parallel::__generic_find_selector, 640
- __gnu_parallel::__generic_for_each_selector
 - _M_finish_iterator, 643
- __gnu_parallel::__generic_for_each_selector< _It >, 642
- __gnu_parallel::__identity_selector
 - _M_finish_iterator, 644
 - operator(), 644
- __gnu_parallel::__identity_selector< _It >, 643
- __gnu_parallel::__inner_product_selector
 - _M_finish_iterator, 646
 - __begin1_iterator, 646
 - __begin2_iterator, 646
 - __inner_product_selector, 645
 - operator(), 646
- __gnu_parallel::__inner_product_selector< _It, _It2, _Tp >, 645
- __gnu_parallel::__max_element_reduct< _Compare, _It >, 647
- __gnu_parallel::__min_element_reduct< _Compare, _It >, 647
- __gnu_parallel::__mismatch_selector, 648
 - _M_sequential_algorithm, 649
 - operator(), 649

- `__gnu_parallel::__multiway_merge_3_variant_sentinel` \leftrightarrow
 - `switch< __sentinels, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >`, 649
- `__gnu_parallel::__multiway_merge_3_variant_sentinel` \leftrightarrow
 - `switch< true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >`, 650
- `__gnu_parallel::__multiway_merge_4_variant_sentinel` \leftrightarrow
 - `switch< __sentinels, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >`, 650
- `__gnu_parallel::__multiway_merge_4_variant_sentinel` \leftrightarrow
 - `switch< true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >`, 651
- `__gnu_parallel::__multiway_merge_k_variant_sentinel` \leftrightarrow
 - `switch< __sentinels, __stable, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >`, 651
- `__gnu_parallel::__multiway_merge_k_variant_sentinel` \leftrightarrow
 - `switch< false, __stable, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >`, 652
- `__gnu_parallel::__replace_if_selector`
 - `_M_finish_iterator`, 654
 - `_new_val`, 654
 - `_replace_if_selector`, 653
 - `operator()`, 654
- `__gnu_parallel::__replace_if_selector< _It, _Op, _Tp >`, 653
- `__gnu_parallel::__replace_selector`
 - `_M_finish_iterator`, 656
 - `_new_val`, 656
 - `_replace_selector`, 655
 - `operator()`, 656
- `__gnu_parallel::__replace_selector< _It, _Tp >`, 655
- `__gnu_parallel::__transform1_selector`
 - `_M_finish_iterator`, 658
 - `operator()`, 657
- `__gnu_parallel::__transform1_selector< _It >`, 657
- `__gnu_parallel::__transform2_selector`
 - `_M_finish_iterator`, 659
 - `operator()`, 659
- `__gnu_parallel::__transform2_selector< _It >`, 658
- `__gnu_parallel::__unary_negate`
 - `argument_type`, 661
 - `result_type`, 661
- `__gnu_parallel::__unary_negate< _Predicate, argument` \leftrightarrow
 - `_type >`, 660
- `__gnu_parallel::balanced_quicksort_tag`, 725
 - `__get_num_threads`, 725
 - `set_num_threads`, 725
- `__gnu_parallel::balanced_tag`, 726
 - `__get_num_threads`, 726
 - `set_num_threads`, 726
- `__gnu_parallel::constant_size_blocks_tag`, 727
- `__gnu_parallel::default_parallel_tag`, 728
 - `__get_num_threads`, 728
 - `set_num_threads`, 728
- `__gnu_parallel::equal_split_tag`, 729
- `__gnu_parallel::exact_tag`, 730
 - `__get_num_threads`, 730
 - `set_num_threads`, 730
- `__gnu_parallel::find_tag`, 731
- `__gnu_parallel::growing_blocks_tag`, 732
- `__gnu_parallel::multiway_mergesort_exact_tag`, 732
 - `__get_num_threads`, 733
 - `set_num_threads`, 733
- `__gnu_parallel::multiway_mergesort_sampling_tag`, 734
 - `__get_num_threads`, 734
 - `set_num_threads`, 734
- `__gnu_parallel::multiway_mergesort_tag`, 735
 - `__get_num_threads`, 736
 - `set_num_threads`, 736
- `__gnu_parallel::omp_loop_static_tag`, 736
 - `__get_num_threads`, 737
 - `set_num_threads`, 737
- `__gnu_parallel::omp_loop_tag`, 738
 - `__get_num_threads`, 738
 - `set_num_threads`, 738
- `__gnu_parallel::parallel_tag`, 740
 - `__get_num_threads`, 741
 - `parallel_tag`, 741
 - `set_num_threads`, 741
- `__gnu_parallel::quicksort_tag`, 742
 - `__get_num_threads`, 743
 - `set_num_threads`, 743
- `__gnu_parallel::sampling_tag`, 743
 - `__get_num_threads`, 744
 - `set_num_threads`, 744
- `__gnu_parallel::sequential_tag`, 744
- `__gnu_parallel::unbalanced_tag`, 745
 - `__get_num_threads`, 745
 - `set_num_threads`, 745
- `__gnu_pbds`, 293
- `__gnu_pbds::associative_tag`, 746
- `__gnu_pbds::basic_branch< Key, Mapped, Tag, Node` \leftrightarrow
 - `Update, Policy_Tl, _Alloc >`, 746
- `__gnu_pbds::basic_branch_tag`, 748
- `__gnu_pbds::basic_hash_table< Key, Mapped, Hash` \leftrightarrow
 - `_Fn, Eq_Fn, Resize_Policy, Store_Hash, Tag, Policy_Tl, _Alloc >`, 748
- `__gnu_pbds::basic_hash_tag`, 750
- `__gnu_pbds::basic_invalidation_guarantee`, 751
- `__gnu_pbds::binary_heap_tag`, 752
- `__gnu_pbds::binomial_heap_tag`, 753
- `__gnu_pbds::cc_hash_max_collision_check_resize` \leftrightarrow
 - `trigger`
 - `cc_hash_max_collision_check_resize_trigger`, 754
 - `external_load_access`, 754
 - `get_load`, 755
 - `is_grow_needed`, 755
 - `is_resize_needed`, 755

- notify_cleared, [755](#)
- notify_erase_search_collision, [755](#)
- notify_erase_search_end, [755](#)
- notify_erase_search_start, [755](#)
- notify_erased, [756](#)
- notify_externally_resized, [756](#)
- notify_find_search_collision, [756](#)
- notify_find_search_end, [756](#)
- notify_find_search_start, [756](#)
- notify_insert_search_collision, [756](#)
- notify_insert_search_end, [757](#)
- notify_insert_search_start, [757](#)
- notify_inserted, [757](#)
- notify_resized, [757](#)
- set_load, [757](#)
- __gnu_pbds::cc_hash_max_collision_check_resize_↵
trigger< External_Load_Access, Size_Type >, [753](#)
- __gnu_pbds::cc_hash_table
cc_hash_table, [759–761](#)
- __gnu_pbds::cc_hash_table< Key, Mapped, Hash_↵
Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy,
Store_Hash, _Alloc >, [758](#)
- __gnu_pbds::cc_hash_tag, [762](#)
- __gnu_pbds::container_error, [763](#)
- __gnu_pbds::container_tag, [763](#)
- __gnu_pbds::container_traits
erase_can_throw, [765](#)
order_preserving, [765](#)
reverse_iteration, [765](#)
split_join_can_throw, [765](#)
- __gnu_pbds::container_traits< Cntnr >, [764](#)
- __gnu_pbds::container_traits_base< _Tag >, [765](#)
- __gnu_pbds::container_traits_base< binary_heap_tag >, [765](#)
- __gnu_pbds::container_traits_base< binomial_heap_tag
>, [766](#)
- __gnu_pbds::container_traits_base< cc_hash_tag >, [766](#)
- __gnu_pbds::container_traits_base< gp_hash_tag >, [767](#)
- __gnu_pbds::container_traits_base< list_update_tag >, [767](#)
- __gnu_pbds::container_traits_base< ov_tree_tag >, [768](#)
- __gnu_pbds::container_traits_base< pairing_heap_tag
>, [768](#)
- __gnu_pbds::container_traits_base< pat_trie_tag >, [769](#)
- __gnu_pbds::container_traits_base< rb_tree_tag >, [769](#)
- __gnu_pbds::container_traits_base< rc_binomial_heap_↵
tag >, [770](#)
- __gnu_pbds::container_traits_base< splay_tree_tag >, [770](#)
- __gnu_pbds::container_traits_base< thin_heap_tag >, [771](#)
- __gnu_pbds::detail::bin_search_tree_const_it_↵
_Pointer, Value_Type, Pointer, Const_Pointer,
Reference, Const_Reference, Is_Forward_↵
Iterator, _Alloc >, [771](#)
- __gnu_pbds::detail::bin_search_tree_const_node_it_↵
const_reference, [774](#)
difference_type, [774](#)
get_l_child, [775](#)
get_metadata, [775](#)
get_r_child, [775](#)
iterator_category, [774](#)
metadata_const_reference, [774](#)
metadata_type, [775](#)
operator!=, [776](#)
operator*, [776](#)
operator==, [776](#)
reference, [775](#)
value_type, [775](#)
- __gnu_pbds::detail::bin_search_tree_const_node_it_↵
Node, Const_Iterator, Iterator, _Alloc >, [773](#)
- __gnu_pbds::detail::bin_search_tree_it_↵
Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference,
Const_Reference, Is_Forward_Iterator, _Alloc
>, [776](#)
- __gnu_pbds::detail::bin_search_tree_node_it_↵
const_reference, [779](#)
difference_type, [779](#)
get_l_child, [780](#)
get_metadata, [780](#)
get_r_child, [780](#)
iterator_category, [779](#)
metadata_const_reference, [779](#)
metadata_type, [780](#)
operator!=, [781](#)
operator*, [781](#)
operator==, [781](#)
reference, [780](#)
value_type, [780](#)
- __gnu_pbds::detail::bin_search_tree_node_it_↵
Node, Const_Iterator, Iterator, _Alloc >, [778](#)
- __gnu_pbds::detail::bin_search_tree_traits
node_const_iterator, [782](#)
- __gnu_pbds::detail::bin_search_tree_traits< Key, null_↵
type, Cmp_Fn, Node_Update, Node, _Alloc >, [783](#)
node_const_iterator, [783](#)
- __gnu_pbds::detail::bin_search_tree_traits< Key, Mapped,
Cmp_Fn, Node_Update, Node, _Alloc >, [781](#)
- __gnu_pbds::detail::binary_heap< Value_Type, Cmp_Fn,
_Alloc >, [784](#)
- __gnu_pbds::detail::binary_heap_const_iterator_↵
binary_heap_const_iterator_, [788](#)
const_pointer, [787](#)
const_reference, [787](#)
difference_type, [787](#)

- iterator_category, 788
- operator!=, 789
- operator*, 789
- operator->, 789
- operator==, 789
- pointer, 788
- reference, 788
- value_type, 788
- __gnu_pbds::detail::binary_heap_const_iterator_↵
 - < Value_Type, Entry, Simple, _Alloc >, 786
- __gnu_pbds::detail::binary_heap_point_const_iterator_
 - binary_heap_point_const_iterator_, 792
 - const_pointer, 791
 - const_reference, 791
 - difference_type, 791
 - iterator_category, 791
 - operator!=, 793
 - operator*, 793
 - operator->, 793
 - operator==, 793
 - pointer, 792
 - reference, 792
 - value_type, 792
- __gnu_pbds::detail::binary_heap_point_const_iterator_ <
 - Value_Type, Entry, Simple, _Alloc >, 790
- __gnu_pbds::detail::binomial_heap< Value_Type, Cmp_↵
 - _Fn, _Alloc >, 793
- __gnu_pbds::detail::binomial_heap_base< Value_Type,
 - Cmp_Fn, _Alloc >, 796
- __gnu_pbds::detail::branch_policy< Node_Cltr, Node_↵
 - Cltr, _Alloc >, 799
- __gnu_pbds::detail::branch_policy< Node_Cltr, Node_ltr,
 - _Alloc >, 798
- __gnu_pbds::detail::cc_ht_map
 - empty, 803
 - get_comb_hash_fn, 803
 - get_eq_fn, 803
 - get_hash_fn, 803, 804
 - get_resize_policy, 804
- __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_↵
 - Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_↵
 - _Fn, Resize_Policy >, 800
- __gnu_pbds::detail::cond_dealtor< Entry, _Alloc >, 804
- __gnu_pbds::detail::container_base_dispatch< Key,
 - Mapped, _Alloc, Tag, Policy_TI >, 805
- __gnu_pbds::detail::container_base_dispatch< Key,
 - Mapped, _Alloc, cc_hash_tag, Policy_TI >, 809
 - type, 809
- __gnu_pbds::detail::container_base_dispatch< Key,
 - Mapped, _Alloc, gp_hash_tag, Policy_TI >, 810
 - type, 810
- __gnu_pbds::detail::container_base_dispatch< Key,
 - Mapped, _Alloc, list_update_tag, Policy_TI >, 810
 - type, 811
- __gnu_pbds::detail::container_base_dispatch< Key,
 - Mapped, _Alloc, ov_tree_tag, Policy_TI >, 811
 - type, 811
- __gnu_pbds::detail::container_base_dispatch< Key,
 - Mapped, _Alloc, pat_trie_tag, Policy_TI >, 812
- __gnu_pbds::detail::container_base_dispatch< Key,
 - Mapped, _Alloc, rb_tree_tag, Policy_TI >, 812
 - type, 812
- __gnu_pbds::detail::container_base_dispatch< Key,
 - Mapped, _Alloc, splay_tree_tag, Policy_TI >, 813
 - type, 813
- __gnu_pbds::detail::container_base_dispatch< Key,
 - null_type, _Alloc, cc_hash_tag, Policy_TI >, 813
 - type, 814
- __gnu_pbds::detail::container_base_dispatch< Key,
 - null_type, _Alloc, gp_hash_tag, Policy_TI >, 814
 - type, 814
- __gnu_pbds::detail::container_base_dispatch< Key,
 - null_type, _Alloc, list_update_tag, Policy_TI >, 815
 - type, 815
- __gnu_pbds::detail::container_base_dispatch< Key,
 - null_type, _Alloc, ov_tree_tag, Policy_TI >, 815
 - type, 816
- __gnu_pbds::detail::container_base_dispatch< Key,
 - null_type, _Alloc, pat_trie_tag, Policy_TI >, 816
 - type, 816
- __gnu_pbds::detail::container_base_dispatch< Key,
 - null_type, _Alloc, rb_tree_tag, Policy_TI >, 817
- __gnu_pbds::detail::container_base_dispatch< Key,
 - null_type, _Alloc, splay_tree_tag, Policy_TI >, 817
 - type, 817
- __gnu_pbds::detail::container_base_dispatch< _VTp,
 - Cmp_Fn, _Alloc, binary_heap_tag, null_type >, 806
 - type, 806
- __gnu_pbds::detail::container_base_dispatch< _VTp,
 - Cmp_Fn, _Alloc, binomial_heap_tag, null_type >, 806
 - type, 807
- __gnu_pbds::detail::container_base_dispatch< _VTp,
 - Cmp_Fn, _Alloc, pairing_heap_tag, null_type >, 807
 - type, 807

- `__gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, rc_binomial_heap_tag, null_type >`, [808](#)
- `type`, [808](#)
- `__gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, thin_heap_tag, null_type >`, [808](#)
- `type`, [809](#)
- `__gnu_pbds::detail::default_comb_hash_fn`, [818](#)
- `type`, [818](#)
- `__gnu_pbds::detail::default_eq_fn`
- `type`, [819](#)
- `__gnu_pbds::detail::default_eq_fn< Key >`, [818](#)
- `__gnu_pbds::detail::default_hash_fn`
- `type`, [819](#)
- `__gnu_pbds::detail::default_hash_fn< Key >`, [819](#)
- `__gnu_pbds::detail::default_probe_fn`
- `type`, [820](#)
- `__gnu_pbds::detail::default_probe_fn< Comb_Probe_Fn >`, [819](#)
- `__gnu_pbds::detail::default_resize_policy`
- `type`, [820](#)
- `__gnu_pbds::detail::default_resize_policy< Comb_Hash_Fn >`, [820](#)
- `__gnu_pbds::detail::default_trie_access_traits< Key >`, [821](#)
- `__gnu_pbds::detail::default_trie_access_traits< std::basic_string< Char, Char_Traits, std::allocator< char > > >`, [821](#)
- `type`, [821](#)
- `__gnu_pbds::detail::default_update_policy`, [822](#)
- `type`, [822](#)
- `__gnu_pbds::detail::dumnode_const_iterator< Key, Data, _Alloc >`, [822](#)
- `__gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, No_Throw >`, [823](#)
- `__gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, false >`, [823](#)
- `__gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, false >::type`, [823](#)
- `__gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, true >`, [824](#)
- `type`, [824](#)
- `__gnu_pbds::detail::entry_pred< _VTp, Pred, _Alloc, No_Throw >`, [825](#)
- `__gnu_pbds::detail::entry_pred< _VTp, Pred, _Alloc, false >`, [825](#)
- `__gnu_pbds::detail::entry_pred< _VTp, Pred, _Alloc, true >`, [825](#)
- `__gnu_pbds::detail::eq_by_less< Key, Cmp_Fn >`, [826](#)
- `__gnu_pbds::detail::gp_ht_map`
- `empty`, [829](#)
- `get_comb_probe_fn`, [829](#)
- `get_eq_fn`, [830](#)
- `get_hash_fn`, [830](#)
- `get_probe_fn`, [830](#)
- `get_resize_policy`, [831](#)
- `__gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >`, [826](#)
- `__gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, Store_Hash >`, [831](#)
- `__gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, false >`, [832](#)
- `__gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, true >`, [833](#)
- `__gnu_pbds::detail::hash_load_check_resize_trigger_size_base< Size_Type, Hold_Size >`, [833](#)
- `__gnu_pbds::detail::hash_load_check_resize_trigger_size_base< Size_Type, true >`, [834](#)
- `__gnu_pbds::detail::left_child_next_sibling_heap< Value_Type, Cmp_Fn, Node_Metadata, _Alloc >`, [834](#)
- `__gnu_pbds::detail::left_child_next_sibling_heap_const_iterator`
- `const_pointer`, [837](#)
- `const_reference`, [837](#)
- `difference_type`, [837](#)
- `iterator_category`, [837](#)
- `left_child_next_sibling_heap_const_iterator`, [838](#)
- `operator!=`, [838](#)
- `operator*`, [838](#)
- `operator->`, [839](#)
- `operator==`, [839](#)
- `pointer`, [837](#)
- `reference`, [837](#)
- `value_type`, [838](#)
- `__gnu_pbds::detail::left_child_next_sibling_heap_const_iterator< Node, _Alloc >`, [836](#)
- `__gnu_pbds::detail::left_child_next_sibling_heap_node< _Value, _Metadata, _Alloc >`, [839](#)
- `__gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator`
- `const_pointer`, [841](#)
- `const_reference`, [841](#)
- `difference_type`, [842](#)
- `iterator_category`, [842](#)
- `left_child_next_sibling_heap_node_point_const_iterator`, [843](#)
- `operator!=`, [843](#)
- `operator*`, [843](#)
- `operator->`, [843](#)
- `operator==`, [843](#)
- `pointer`, [842](#)
- `reference`, [842](#)
- `value_type`, [842](#)
- `__gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator< Node, _Alloc >`, [840](#)
- `__gnu_pbds::detail::lu_counter_metadata< Size_Type >`,

- [844](#)
- `__gnu_pbds::detail::lu_counter_policy_base< Size_Type`
`>`, [844](#)
- `__gnu_pbds::detail::lu_map< Key, Mapped, Eq_Fn, _C↵`
`Alloc, Update_Policy >`, [845](#)
- `__gnu_pbds::detail::mask_based_range_hashing<`
`Size_Type >`, [847](#)
- `__gnu_pbds::detail::mod_based_range_hashing< Size_↵`
`_Type >`, [848](#)
- `__gnu_pbds::detail::no_throw_copies< Key, Mapped >`,
[849](#)
- `__gnu_pbds::detail::no_throw_copies< Key, null_type >`,
[850](#)
- `__gnu_pbds::detail::ov_tree_map`
`node_begin`, [852](#)
`node_end`, [853](#)
- `__gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_↵`
`_Fn, Node_And_It_Traits, _Alloc >`, [850](#)
- `__gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_↵`
`_Fn, Node_And_It_Traits, _Alloc >::cond_dtor<`
`Size_Type >`, [853](#)
- `__gnu_pbds::detail::ov_tree_node_const_it`
`get_l_child`, [855](#)
`get_r_child`, [855](#)
- `__gnu_pbds::detail::ov_tree_node_const_it< Value_↵`
`Type, Metadata_Type, _Alloc >`, [854](#)
- `__gnu_pbds::detail::ov_tree_node_it`
`get_l_child`, [857](#)
`get_r_child`, [857](#)
`operator*`, [857](#)
- `__gnu_pbds::detail::ov_tree_node_it< Value_Type,`
`Metadata_Type, _Alloc >`, [856](#)
- `__gnu_pbds::detail::pairing_heap< Value_Type, Cmp_Fn,`
`_Alloc >`, [858](#)
- `__gnu_pbds::detail::pat_trie_base`, [860](#)
`node_type`, [861](#)
- `__gnu_pbds::detail::pat_trie_base::_CIter< Node, Leaf,`
`Head, Inode, Is_Forward_Iterator >`, [861](#)
- `__gnu_pbds::detail::pat_trie_base::_Head< _ATraits,`
`Metadata >`, [863](#)
- `__gnu_pbds::detail::pat_trie_base::_Inode< _ATraits,`
`Metadata >`, [864](#)
- `__gnu_pbds::detail::pat_trie_base::_Inode< _ATraits,`
`Metadata >::const_iterator`, [866](#)
- `__gnu_pbds::detail::pat_trie_base::_Inode< _ATraits,`
`Metadata >::iterator`, [867](#)
- `__gnu_pbds::detail::pat_trie_base::_Iter< Node, Leaf,`
`Head, Inode, Is_Forward_Iterator >`, [869](#)
- `__gnu_pbds::detail::pat_trie_base::_Leaf< _ATraits,`
`Metadata >`, [871](#)
- `__gnu_pbds::detail::pat_trie_base::_Metadata< Meta-`
`data, _Alloc >`, [872](#)
- `__gnu_pbds::detail::pat_trie_base::_Metadata< null_↵`
`type, _Alloc >`, [873](#)
- `__gnu_pbds::detail::pat_trie_base::_Node_base< _A_↵`
`Traits, Metadata >`, [873](#)
- `__gnu_pbds::detail::pat_trie_base::_Node_citer`
`__rebind_m`, [876](#)
`get_child`, [876](#)
`get_metadata`, [876](#)
`metadata_type`, [876](#)
`num_children`, [876](#)
`operator!=`, [876](#)
`operator*`, [877](#)
`operator==`, [877](#)
`valid_prefix`, [877](#)
- `__gnu_pbds::detail::pat_trie_base::_Node_citer< Node,`
`Leaf, Head, Inode, _CIterator, Iterator, _Alloc >`,
[874](#)
- `__gnu_pbds::detail::pat_trie_base::_Node_iter`
`__rebind_m`, [879](#)
`get_child`, [879](#)
`get_metadata`, [879](#)
`metadata_type`, [879](#)
`num_children`, [879](#)
`operator!=`, [879](#)
`operator*`, [880](#)
`operator==`, [880](#)
`valid_prefix`, [880](#)
- `__gnu_pbds::detail::pat_trie_base::_Node_iter< Node,`
`Leaf, Head, Inode, _CIterator, Iterator, _Alloc >`,
[877](#)
- `__gnu_pbds::detail::pat_trie_map`
`node_begin`, [883](#)
`node_end`, [883](#)
`node_type`, [883](#)
- `__gnu_pbds::detail::pat_trie_map< Key, Mapped, Node_↵`
`_And_It_Traits, _Alloc >`, [880](#)
- `__gnu_pbds::detail::probe_fn_base< _Alloc >`, [884](#)
- `__gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _C↵`
`Alloc, Comb_Hash_Fn, Store_Hash >`, [884](#)
- `__gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _C↵`
`Alloc, Comb_Hash_Fn, false >`, [885](#)
- `__gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _C↵`
`Alloc, Comb_Hash_Fn, true >`, [885](#)
- `__gnu_pbds::detail::ranged_hash_fn< Key, null_type, _C↵`
`Alloc, Comb_Hash_Fn, false >`, [886](#)
- `__gnu_pbds::detail::ranged_hash_fn< Key, null_type, _C↵`
`Alloc, Comb_Hash_Fn, true >`, [887](#)
- `__gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _C↵`
`_Alloc, Comb_Probe_Fn, Probe_Fn, Store_↵`
`Hash >`, [888](#)
- `__gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _C↵`
`_Alloc, Comb_Probe_Fn, Probe_Fn, false >`,
[888](#)
- `__gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _C↵`
`_Alloc, Comb_Probe_Fn, Probe_Fn, true >`, [889](#)
- `__gnu_pbds::detail::ranged_probe_fn< Key, null_type, _C↵`

- `_Alloc, Comb_Probe_Fn, null_type, false >`, [890](#)
- `__gnu_pbds::detail::rb_tree_map`
 - `node_begin`, [894](#)
 - `node_end`, [894](#)
- `__gnu_pbds::detail::rb_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >`, [891](#)
- `__gnu_pbds::detail::rb_tree_node< Value_Type, Metadata, _Alloc >`, [894](#)
- `__gnu_pbds::detail::rc< _Node, _Alloc >`, [895](#)
- `__gnu_pbds::detail::rc_binomial_heap< Value_Type, Cmp_Fn, _Alloc >`, [896](#)
- `__gnu_pbds::detail::resize_policy< _Tp >`, [898](#)
- `__gnu_pbds::detail::splay_tree_map`
 - `node_begin`, [902](#)
 - `node_end`, [902](#)
- `__gnu_pbds::detail::splay_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >`, [899](#)
- `__gnu_pbds::detail::splay_tree_node< Value_Type, Metadata, _Alloc >`, [903](#)
- `__gnu_pbds::detail::stored_data< _Tv, _Th >`, [904](#)
- `__gnu_pbds::detail::stored_data< _Tv, null_type >`, [905](#)
- `__gnu_pbds::detail::stored_hash< _Th >`, [906](#)
- `__gnu_pbds::detail::stored_value< _Tv >`, [907](#)
- `__gnu_pbds::detail::synth_access_traits< Type_Traits, Set, _ATraits >`, [907](#)
- `__gnu_pbds::detail::thin_heap< Value_Type, Cmp_Fn, _Alloc >`, [908](#)
- `__gnu_pbds::detail::tree_metadata_helper< Node_↔ Update, _BTp >`, [911](#)
- `__gnu_pbds::detail::tree_metadata_helper< Node_↔ Update, false >`, [911](#)
- `__gnu_pbds::detail::tree_metadata_helper< Node_↔ Update, true >`, [911](#)
- `__gnu_pbds::detail::tree_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc >`, [912](#)
- `__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >`, [913](#)
 - `node_const_iterator`, [913](#)
- `__gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >`, [913](#)
 - `node_const_iterator`, [914](#)
- `__gnu_pbds::detail::tree_traits< Key, Data, Cmp_Fn, Node_Update, Tag, _Alloc >`, [912](#)
- `__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >`, [914](#)
 - `node_const_iterator`, [915](#)
- `__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >`, [916](#)
 - `node_const_iterator`, [917](#)
- `__gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >`, [917](#)
 - `node_const_iterator`, [918](#)
- `__gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >`, [919](#)
 - `node_const_iterator`, [920](#)
- `__gnu_pbds::detail::trie_metadata_helper< Node_↔ Update, _BTp >`, [920](#)
- `__gnu_pbds::detail::trie_metadata_helper< Node_↔ Update, false >`, [920](#)
- `__gnu_pbds::detail::trie_metadata_helper< Node_↔ Update, true >`, [921](#)
- `__gnu_pbds::detail::trie_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc >`, [921](#)
- `__gnu_pbds::detail::trie_policy_base< Node_Cltr, Node_Itr, _ATraits, _Alloc >`, [922](#)
- `__gnu_pbds::detail::trie_traits< Key, Data, _ATraits, Node_Update, Tag, _Alloc >`, [924](#)
- `__gnu_pbds::detail::trie_traits< Key, Mapped, _ATraits, Node_Update, pat_trie_tag, _Alloc >`, [924](#)
 - `node_const_iterator`, [925](#)
 - `node_update`, [925](#)
 - `synth_access_traits`, [925](#)
- `__gnu_pbds::detail::trie_traits< Key, null_type, _ATraits, Node_Update, pat_trie_tag, _Alloc >`, [925](#)
 - `node_const_iterator`, [926](#)
 - `node_update`, [926](#)
 - `synth_access_traits`, [926](#)
- `__gnu_pbds::detail::type_base< Key, Mapped, _Alloc, Store_Hash >`, [927](#)
- `__gnu_pbds::detail::type_base< Key, Mapped, _Alloc, false >`, [927](#)
- `__gnu_pbds::detail::type_base< Key, Mapped, _Alloc, true >`, [928](#)
- `__gnu_pbds::detail::type_base< Key, null_type, _Alloc, false >`, [929](#)
- `__gnu_pbds::detail::type_base< Key, null_type, _Alloc, true >`, [929](#)
- `__gnu_pbds::detail::type_dispatch< Key, Mapped, _Alloc, Store_Hash >`, [930](#)
- `__gnu_pbds::detail::types_traits< Key, Mapped, _Alloc, Store_Hash >`, [930](#)
- `__gnu_pbds::direct_mask_range_hashing`
 - `operator()`, [932](#)
- `__gnu_pbds::direct_mask_range_hashing< Size_Type >`, [931](#)
- `__gnu_pbds::direct_mod_range_hashing`
 - `operator()`, [934](#)
- `__gnu_pbds::direct_mod_range_hashing< Size_Type >`, [933](#)
- `__gnu_pbds::gp_hash_table`
 - `gp_hash_table`, [936–938](#)
- `__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_↔ Policy, Store_Hash, _Alloc >`, [934](#)
- `__gnu_pbds::gp_hash_tag`, [939](#)
- `__gnu_pbds::hash_exponential_size_policy`
 - `hash_exponential_size_policy`, [940](#)
- `__gnu_pbds::hash_exponential_size_policy< Size_Type`

- >, 940
- __gnu_pbds::hash_load_check_resize_trigger
 - external_load_access, 942
 - get_loads, 942
 - hash_load_check_resize_trigger, 942
 - notify_cleared, 942
 - notify_inserted, 942
 - notify_resized, 943
 - set_loads, 943
- __gnu_pbds::hash_load_check_resize_trigger< External_↵
_Load_Access, Size_Type >, 941
- __gnu_pbds::hash_prime_size_policy, 943
 - hash_prime_size_policy, 944
 - size_type, 944
- __gnu_pbds::hash_standard_resize_policy
 - get_actual_size, 946
 - get_new_size, 946
 - get_size_policy, 946
 - get_trigger_policy, 947
 - hash_standard_resize_policy, 945, 946
 - resize, 947
- __gnu_pbds::hash_standard_resize_policy< Size_Policy,
Trigger_Policy, External_Size_Access, Size_↵
Type >, 944
- __gnu_pbds::insert_error, 947
- __gnu_pbds::join_error, 948
- __gnu_pbds::linear_probe_fn
 - operator(), 949
- __gnu_pbds::linear_probe_fn< Size_Type >, 948
- __gnu_pbds::list_update
 - list_update, 950
- __gnu_pbds::list_update< Key, Mapped, Eq_Fn,
Update_Policy, _Alloc >, 949
- __gnu_pbds::list_update_tag, 950
- __gnu_pbds::lu_counter_policy
 - max_count, 952
 - metadata_reference, 952
 - metadata_type, 952
 - operator(), 952
- __gnu_pbds::lu_counter_policy< Max_Count, _Alloc >,
951
- __gnu_pbds::lu_move_to_front_policy
 - metadata_reference, 953
 - metadata_type, 953
 - operator(), 954
- __gnu_pbds::lu_move_to_front_policy< _Alloc >, 953
- __gnu_pbds::null_node_update< _Tp1, _Tp2, _Tp3, _↵
Tp4 >, 954
- __gnu_pbds::null_type, 955
- __gnu_pbds::ov_tree_tag, 956
- __gnu_pbds::pairing_heap_tag, 957
- __gnu_pbds::pat_trie_tag, 958
- __gnu_pbds::point_invalidation_guarantee, 959
- __gnu_pbds::priority_queue< _Tv, Cmp_Fn, Tag, _Alloc
>, 959
- __gnu_pbds::priority_queue_tag, 961
- __gnu_pbds::quadratic_probe_fn
 - operator(), 962
- __gnu_pbds::quadratic_probe_fn< Size_Type >, 961
- __gnu_pbds::range_invalidation_guarantee, 962
- __gnu_pbds::rb_tree_tag, 963
- __gnu_pbds::rc_binomial_heap_tag, 964
- __gnu_pbds::resize_error, 964
- __gnu_pbds::sample_probe_fn, 965
 - operator(), 966
 - sample_probe_fn, 965
 - swap, 966
- __gnu_pbds::sample_range_hashing, 966
 - notify_resized, 967
 - operator(), 967
 - sample_range_hashing, 967
 - size_type, 966
 - swap, 967
- __gnu_pbds::sample_ranged_hash_fn, 967
 - notify_resized, 968
 - operator(), 968
 - sample_ranged_hash_fn, 968
 - swap, 968
- __gnu_pbds::sample_ranged_probe_fn, 968
- __gnu_pbds::sample_resize_policy, 969
 - get_new_size, 970
 - is_resize_needed, 970
 - notify_cleared, 970
 - notify_erase_search_collision, 970
 - notify_erase_search_end, 970
 - notify_erase_search_start, 970
 - notify_erased, 971
 - notify_find_search_collision, 971
 - notify_find_search_end, 971
 - notify_find_search_start, 971
 - notify_insert_search_collision, 971
 - notify_insert_search_end, 971
 - notify_insert_search_start, 971
 - notify_inserted, 971
 - notify_resized, 971
 - sample_range_hashing, 971
 - sample_resize_policy, 970
 - size_type, 970
 - swap, 971
- __gnu_pbds::sample_resize_trigger, 972
 - is_grow_needed, 973
 - is_resize_needed, 973
 - notify_cleared, 973
 - notify_erase_search_collision, 973
 - notify_erase_search_end, 973
 - notify_erase_search_start, 973
 - notify_erased, 973

- notify_externally_resized, 973
- notify_find_search_collision, 974
- notify_find_search_end, 974
- notify_find_search_start, 974
- notify_insert_search_collision, 974
- notify_insert_search_end, 974
- notify_insert_search_start, 974
- notify_inserted, 974
- notify_resized, 974
- sample_range_hashing, 974
- sample_resize_trigger, 973
- size_type, 973
- swap, 974
- __gnu_pbds::sample_size_policy, 975
 - get_nearest_larger_size, 975
 - get_nearest_smaller_size, 975
 - sample_range_hashing, 976
 - sample_size_policy, 975
 - size_type, 975
 - swap, 976
- __gnu_pbds::sample_tree_node_update< Const_Node↔
_Iter, Node_Iter, Cmp_Fn, _Alloc >, 976
- __gnu_pbds::sample_trie_access_traits, 976
 - begin, 977
 - e_pos, 977
 - e_type, 977
 - end, 977
- __gnu_pbds::sample_trie_node_update
 - operator(), 978
 - sample_trie_node_update, 978
- __gnu_pbds::sample_trie_node_update< Node_Cltr, Node_Itr, ATraits, _Alloc >, 977
- __gnu_pbds::sample_update_policy, 978
 - metadata_type, 979
 - operator(), 979
 - sample_update_policy, 979
 - swap, 979
- __gnu_pbds::sequence_tag, 980
- __gnu_pbds::splay_tree_tag, 981
- __gnu_pbds::string_tag, 982
- __gnu_pbds::thin_heap_tag, 983
- __gnu_pbds::tree
 - cmp_fn, 984
 - tree, 985
- __gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node↔
Update, _Alloc >, 983
- __gnu_pbds::tree_order_statistics_node_update
 - find_by_order, 987
 - operator(), 987
 - order_of_key, 988
- __gnu_pbds::tree_order_statistics_node_update< Node↔
_Cltr, Node_Itr, Cmp_Fn, _Alloc >, 986
- __gnu_pbds::tree_tag, 988
- __gnu_pbds::trie
 - access_traits, 990
 - trie, 990
- __gnu_pbds::trie< Key, Mapped, ATraits, Tag, Node↔
Update, _Alloc >, 989
- __gnu_pbds::trie_order_statistics_node_update
 - find_by_order, 993
 - operator(), 993
 - order_of_key, 993
 - order_of_prefix, 993
- __gnu_pbds::trie_order_statistics_node_update< Node↔
_Cltr, Node_Itr, ATraits, _Alloc >, 991
- __gnu_pbds::trie_prefix_search_node_update
 - a_const_iterator, 996
 - access_traits, 996
 - allocator_type, 996
 - operator(), 996
 - prefix_range, 996, 997
 - size_type, 996
- __gnu_pbds::trie_prefix_search_node_update< Node↔
_Cltr, Node_Itr, ATraits, _Alloc >, 994
- __gnu_pbds::trie_string_access_traits
 - begin, 999
 - const_iterator, 998
 - e_pos, 999
 - e_type, 999
 - end, 999
- __gnu_pbds::trie_string_access_traits< String, Min_E↔
Val, Max_E_Val, Reverse, _Alloc >, 998
- __gnu_pbds::trie_tag, 1000
- __gnu_pbds::trivial_iterator_tag, 1001
- __gnu_profile, 295
 - _GLIBCXX_PROFILE_DEFINE_UNINIT_DATA, 299
 - __env_t, 299
 - __profcxx_init, 299
 - __report, 299
- __gnu_profile::__container_size_info, 1001
- __gnu_profile::__container_size_stack_info, 1002
- __gnu_profile::__hashfunc_info, 1003
- __gnu_profile::__hashfunc_stack_info, 1004
- __gnu_profile::__list2vector_info, 1005
- __gnu_profile::__map2umap_info, 1007
- __gnu_profile::__map2umap_stack_info, 1008
- __gnu_profile::__object_info_base, 1009
- __gnu_profile::__reentrance_guard, 1009
- __gnu_profile::__stack_hash, 1010
- __gnu_profile::__stack_info_base< __object_info >, 1010
- __gnu_profile::__trace_base< __object_info, __stack↔
info >, 1011
- __gnu_profile::__trace_container_size, 1011
- __gnu_profile::__trace_hash_func, 1012
- __gnu_profile::__trace_hashtable_size, 1013
- __gnu_profile::__trace_map2umap, 1014
- __gnu_profile::__trace_vector_size, 1015

- `__gnu_profile::__trace_vector_to_list`, 1016
- `__gnu_profile::__vector2list_info`, 1017
- `__gnu_profile::__vector2list_stack_info`, 1018
- `__gnu_profile::__warning_data`, 1019
- `__gnu_sequential`, 300
- `__heap_select`
 - `std`, 371
- `__init_winner`
 - `__gnu_parallel::_LoserTree< false, _Tp, _Compare >`, 682
- `__inner_product_selector`
 - `__gnu_parallel::_inner_product_selector`, 645
- `__inplace_stable_partition`
 - `std`, 371
- `__inplace_stable_sort`
 - `std`, 371
- `__insert_start`
 - `__gnu_parallel::_LoserTree`, 680
 - `__gnu_parallel::_LoserTree< false, _Tp, _Compare >`, 683
 - `__gnu_parallel::_LoserTreeBase`, 686
- `__insertion_sort`
 - `std`, 371
- `__introsort_loop`
 - `std`, 372
- `__is_sorted`
 - `__gnu_parallel`, 263
- `__iterator_category`
 - `Iterators`, 65
- `__lg`
 - `std`, 372
- `__match_flag`
 - `std::regex_constants`, 449
- `__median_of_three_iterators`
 - `__gnu_parallel`, 263
- `__merge_adaptive`
 - `std`, 372
- `__merge_advance`
 - `__gnu_parallel`, 264
- `__merge_advance_movc`
 - `__gnu_parallel`, 264
- `__merge_advance_usual`
 - `__gnu_parallel`, 265
- `__merge_without_buffer`
 - `std`, 372
- `__move_median_to_first`
 - `std`, 372
- `__move_merge`
 - `std`, 373
- `__move_merge_adaptive`
 - `std`, 373
- `__move_merge_adaptive_backward`
 - `std`, 373
- `__new_val`
 - `__gnu_parallel::_replace_if_selector`, 654
 - `__gnu_parallel::_replace_selector`, 656
- `__num_bitmaps`
 - `__gnu_cxx::_detail`, 236
- `__num_blocks`
 - `__gnu_cxx::_detail`, 236
- `__num_get_type`
 - `std::basic_ios`, 1180
- `__num_put_type`
 - `std::basic_ios`, 1180
- `__parallel_merge_advance`
 - `__gnu_parallel`, 266
- `__parallel_nth_element`
 - `__gnu_parallel`, 267
- `__parallel_partial_sort`
 - `__gnu_parallel`, 267
- `__parallel_partial_sum`
 - `__gnu_parallel`, 267
- `__parallel_partial_sum_basecase`
 - `__gnu_parallel`, 268
- `__parallel_partial_sum_linear`
 - `__gnu_parallel`, 268
- `__parallel_partition`
 - `__gnu_parallel`, 269
- `__parallel_random_shuffle`
 - `__gnu_parallel`, 269
- `__parallel_random_shuffle_drs`
 - `__gnu_parallel`, 270
- `__parallel_random_shuffle_drs_pu`
 - `__gnu_parallel`, 270
- `__parallel_sort`
 - `__gnu_parallel`, 271–275
- `__parallel_sort_qs`
 - `__gnu_parallel`, 275
- `__parallel_sort_qs_conquer`
 - `__gnu_parallel`, 276
- `__parallel_sort_qs_divide`
 - `__gnu_parallel`, 276
- `__parallel_sort_qsb`
 - `__gnu_parallel`, 277
- `__parallel_unique_copy`
 - `__gnu_parallel`, 277
- `__partition`
 - `std`, 373
- `__profcxx_init`
 - `__gnu_profile`, 299
- `__qsb_conquer`
 - `__gnu_parallel`, 278
- `__qsb_divide`
 - `__gnu_parallel`, 278
- `__qsb_local_sort_with_helping`
 - `__gnu_parallel`, 279
- `__random_number_pow2`
 - `__gnu_parallel`, 279

- __rd_log2
 - __gnu_parallel, 280
- __rebind_m
 - __gnu_pbds::detail::pat_trie_base::_Node_citer, 876
 - __gnu_pbds::detail::pat_trie_base::_Node_iter, 879
- __replace_if_selector
 - __gnu_parallel::__replace_if_selector, 653
- __replace_selector
 - __gnu_parallel::__replace_selector, 655
- __report
 - __gnu_profile, 299
- __reverse
 - std, 374
- __rotate
 - std, 374
- __rotate_adaptive
 - std, 375
- __round_up_to_pow2
 - __gnu_parallel, 280
- __search_n_aux
 - std, 375
- __search_template
 - __gnu_parallel, 280
- __sequential_multiway_merge
 - __gnu_parallel, 281
- __sequential_random_shuffle
 - __gnu_parallel, 281
- __shrink
 - __gnu_parallel, 282
- __shrink_and_double
 - __gnu_parallel, 282
- __stable_partition_adaptive
 - std, 375
- __static_pointer_cast
 - __gnu_cxx, 226
- __syntax_option
 - std::regex_constants, 449
- __umap_traits
 - std, 368
- __ummap_traits
 - std, 368
- __umset_traits
 - std, 368
- __unguarded_insertion_sort
 - std, 375
- __unguarded_linear_insert
 - std, 376
- __unguarded_partition
 - std, 376
- __unguarded_partition_pivot
 - std, 376
- __unique_copy
 - std, 376, 377
- __uset_traits
 - std, 369
- __valid_range
 - __gnu_debug, 243, 244
- __valid_range_aux
 - __gnu_debug, 244
- __valid_range_aux2
 - __gnu_debug, 244
- __versa_string
 - __gnu_cxx::__versa_string, 483–485
- __yield
 - __gnu_parallel, 282
- ~_LoserTreeBase
 - __gnu_parallel::_LoserTreeBase, 686
- ~_RestrictedBoundedConcurrentQueue
 - __gnu_parallel::_RestrictedBoundedConcurrentQueue, 714
- ~_Safe_sequence_base
 - __gnu_debug::_Safe_sequence_base, 612
- ~_Safe_unordered_container_base
 - __gnu_debug::_Safe_unordered_container_base, 620
- ~__versa_string
 - __gnu_cxx::__versa_string, 486
- ~auto_ptr
 - std::auto_ptr, 1169
- ~basic_ios
 - std::basic_ios, 1183
- ~basic_regex
 - std::basic_regex, 1207
- ~basic_string
 - std::basic_string, 1220
- ~collate
 - std::collate, 1316
- ~ctype
 - std::ctype< char >, 1346
 - std::ctype< wchar_t >, 1360
- ~deque
 - std::deque, 1411
- ~facet
 - std::locale::facet, 1585
- ~forward_list
 - std::forward_list, 1458
- ~gslice
 - Numeric_arrays, 113
- ~ios_base
 - std::ios_base, 1527
- ~locale
 - std::locale, 1578
- ~match_results
 - std::match_results, 1622
- ~messages
 - std::messages, 1641
- ~money_get
 - std::money_get, 1651

- ~money_put
 - std::money_put, 1656
- ~moneypunct
 - std::moneypunct, 1662
- ~num_get
 - std::num_get, 1731
- ~num_put
 - std::num_put, 1746
- ~numpunct
 - std::numpunct, 1760
- ~stdio_filebuf
 - __gnu_cxx::stdio_filebuf, 567
- ~time_get
 - std::time_get, 1881
- ~time_put
 - std::time_put, 1898
- ~unique_ptr
 - std::unique_ptr, 1916
 - std::unique_ptr< _Tp[], _Dp >, 1921
- ~vector
 - std::vector, 2021
- a
 - std::extreme_value_distribution, 1445
 - std::weibull_distribution, 2041
- a_const_iterator
 - __gnu_pbds::trie_prefix_search_node_update, 996
- ATOMIC_BOOL_LOCK_FREE
 - Atomics, 11
- abi, 300
- access_traits
 - __gnu_pbds::trie, 990
 - __gnu_pbds::trie_prefix_search_node_update, 996
- accumulate
 - std, 377, 378
- accumulate_minimal_n
 - __gnu_parallel::Settings, 717
- Adaptors for pointers to functions, 2
 - ptr_fun, 2
- Adaptors for pointers to members, 4
- addressof
 - Utilities, 212
- adjacent_difference
 - std, 378, 379
- adjacent_difference_minimal_n
 - __gnu_parallel::Settings, 717
- adjacent_find
 - Non-Mutating, 93
- adjustfield
 - std::basic_ios, 1197
 - std::ios_base, 1533
- advance
 - std, 379
- algo.h, 2043
- algbase.h, 2053
- algorithmfwd.h, 2054, 2059
- Algorithms, 5
- aligned_buffer.h, 2068
- all
 - std::locale, 1581
- all_of
 - Non-Mutating, 94
- alloc_traits.h, 2068, 2069
- allocate
 - __gnu_cxx::__alloc_traits, 461
 - std::allocator_traits, 1158, 1159
 - std::allocator_traits< allocator< _Tp > >, 1163, 1164
- allocate_shared
 - Pointer_abstractions, 124
 - std::shared_ptr, 1859
- allocator.h, 2070
- allocator_type
 - __gnu_pbds::trie_prefix_search_node_update, 996
 - std::allocator_traits, 1157
 - std::allocator_traits< allocator< _Tp > >, 1162
 - std::set, 1835
 - std::unordered_map, 1927
 - std::unordered_multimap, 1950
 - std::unordered_multiset, 1973
 - std::unordered_set, 1996
- Allocators, 6
 - __allocator_base, 6
- alpha
 - std::gamma_distribution, 1479
- any_of
 - Non-Mutating, 94
- app
 - std::basic_ios, 1197
 - std::ios_base, 1533
- append
 - __gnu_cxx::__versa_string, 486, 487, 489, 490
 - std::basic_string, 1220–1222
- apply_generator
 - __gnu_cxx::typelist, 237
- argument_type
 - __gnu_cxx::__detail::_Ffit_finder, 466
 - __gnu_parallel::__binder1st, 628
 - __gnu_parallel::__binder2nd, 629
 - __gnu_parallel::__unary_negate, 661
 - std::binder1st, 1272
 - std::binder2nd, 1273
 - std::const_mem_fun_ref_t, 1327
 - std::const_mem_fun_t, 1329
 - std::hash< __gnu_cxx::throw_value_limit >, 1493
 - std::hash< __gnu_cxx::throw_value_random >, 1494
 - std::logical_not, 1589

- std::mem_fun_ref_t, 1632
- std::mem_fun_t, 1633
- std::negate, 1716
- std::pointer_to_unary_function, 1795
- std::unary_function, 1905
- std::unary_negate, 1906
- Arithmetic Classes, 8
- array_allocator.h, 2070
- assign
 - __gnu_cxx::__versa_string, 490–493
 - std::basic_regex, 1207–1209
 - std::basic_string, 1223–1226
 - std::deque, 1414, 1415
 - std::forward_list, 1459
 - std::list, 1561
 - std::vector, 2022
- assoc_container.hpp, 2071
- Associative, 9
- at
 - __gnu_cxx::__versa_string, 494
 - std::basic_string, 1226, 1227
 - std::deque, 1415, 1416
 - std::map, 1600
 - std::unordered_map, 1931, 1932
 - std::vector, 2023
- ate
 - std::basic_ios, 1197
 - std::ios_base, 1533
- atomic_base.h, 2071
- atomic_char
 - Atomics, 12
- atomic_char16_t
 - Atomics, 12
- atomic_char32_t
 - Atomics, 12
- atomic_int
 - Atomics, 12
- atomic_int_fast16_t
 - Atomics, 12
- atomic_int_fast32_t
 - Atomics, 12
- atomic_int_fast64_t
 - Atomics, 12
- atomic_int_fast8_t
 - Atomics, 12
- atomic_int_least16_t
 - Atomics, 12
- atomic_int_least32_t
 - Atomics, 13
- atomic_int_least64_t
 - Atomics, 13
- atomic_int_least8_t
 - Atomics, 13
- atomic_intmax_t
 - Atomics, 13
- Atomics, 13
- atomic_intptr_t
 - Atomics, 13
- atomic_llong
 - Atomics, 13
- atomic_lockfree_defines.h, 2073
- atomic_long
 - Atomics, 13
- atomic_ptrdiff_t
 - Atomics, 13
- atomic_schar
 - Atomics, 14
- atomic_short
 - Atomics, 14
- atomic_size_t
 - Atomics, 14
- atomic_uchar
 - Atomics, 14
- atomic_uint
 - Atomics, 14
- atomic_uint_fast16_t
 - Atomics, 14
- atomic_uint_fast32_t
 - Atomics, 14
- atomic_uint_fast64_t
 - Atomics, 14
- atomic_uint_fast8_t
 - Atomics, 15
- atomic_uint_least16_t
 - Atomics, 15
- atomic_uint_least32_t
 - Atomics, 15
- atomic_uint_least64_t
 - Atomics, 15
- atomic_uint_least8_t
 - Atomics, 15
- atomic_uintmax_t
 - Atomics, 15
- atomic_uintptr_t
 - Atomics, 15
- atomic_ullong
 - Atomics, 15
- atomic_ulong
 - Atomics, 16
- atomic_ushort
 - Atomics, 16
- atomic_wchar_t
 - Atomics, 16
- atomic_word.h, 2074
- atomicity.h, 2074
- Atomics, 10
 - ATOMIC_BOOL_LOCK_FREE, 11
 - atomic_char, 12
 - atomic_char16_t, 12

- atomic_char32_t, [12](#)
- atomic_int, [12](#)
- atomic_int_fast16_t, [12](#)
- atomic_int_fast32_t, [12](#)
- atomic_int_fast64_t, [12](#)
- atomic_int_fast8_t, [12](#)
- atomic_int_least16_t, [12](#)
- atomic_int_least32_t, [13](#)
- atomic_int_least64_t, [13](#)
- atomic_int_least8_t, [13](#)
- atomic_intmax_t, [13](#)
- atomic_intptr_t, [13](#)
- atomic_llong, [13](#)
- atomic_long, [13](#)
- atomic_ptrdiff_t, [13](#)
- atomic_schar, [14](#)
- atomic_short, [14](#)
- atomic_size_t, [14](#)
- atomic_uchar, [14](#)
- atomic_uint, [14](#)
- atomic_uint_fast16_t, [14](#)
- atomic_uint_fast32_t, [14](#)
- atomic_uint_fast64_t, [14](#)
- atomic_uint_fast8_t, [15](#)
- atomic_uint_least16_t, [15](#)
- atomic_uint_least32_t, [15](#)
- atomic_uint_least64_t, [15](#)
- atomic_uint_least8_t, [15](#)
- atomic_uintmax_t, [15](#)
- atomic_uintptr_t, [15](#)
- atomic_ullong, [15](#)
- atomic_ulong, [16](#)
- atomic_ushort, [16](#)
- atomic_wchar_t, [16](#)
- kill_dependency, [16](#)
- memory_order, [16](#)
- auto_ptr
 - std::auto_ptr, [1168](#), [1169](#)
- auto_ptr.h, [2074](#)
- awk
 - std::regex_constants, [451](#)
- b
 - std::extreme_value_distribution, [1445](#)
 - std::weibull_distribution, [2041](#)
- back
 - __gnu_cxx::__versa_string, [495](#)
 - std::basic_string, [1227](#)
 - std::deque, [1416](#), [1417](#)
 - std::list, [1562](#)
 - std::queue, [1807](#)
 - std::vector, [2024](#)
- back_insert_iterator
 - std::back_insert_iterator, [1175](#)
- back_inserter
 - Iterators, [65](#)
- backward_warning.h, [2075](#)
- bad
 - std::basic_ios, [1184](#)
- badbit
 - std::basic_ios, [1198](#)
 - std::ios_base, [1533](#)
- balanced_quicksort.h, [2075](#)
- base
 - __gnu_debug::__Safe_iterator, [586](#)
 - __gnu_debug::__Safe_local_iterator, [600](#)
 - std::discard_block_engine, [1430](#)
 - std::independent_bits_engine, [1514](#)
 - std::reverse_iterator, [1829](#)
 - std::shuffle_order_engine, [1863](#)
- Base and Implementation Classes, [17](#), [20](#)
 - _Opcode, [21](#)
- Base and Policy Classes, [22–24](#)
- base.h, [2076](#)
- basefield
 - std::basic_ios, [1198](#)
 - std::ios_base, [1533](#)
- basic
 - std::regex_constants, [451](#)
- basic_file.h, [2077](#)
- basic_ios
 - std::basic_ios, [1183](#), [1184](#)
- basic_ios.h, [2077](#)
- basic_iterator.h, [2078](#)
- basic_regex
 - std::basic_regex, [1204–1206](#)
- basic_string
 - std::basic_string, [1216–1220](#)
- basic_string.h, [2078](#)
- before_begin
 - std::forward_list, [1459](#), [1460](#)
- beg
 - std::basic_ios, [1198](#)
 - std::ios_base, [1533](#)
- begin
 - __gnu_cxx::__versa_string, [495](#)
 - __gnu_parallel::__PseudoSequence, [707](#)
 - __gnu_pbds::sample_trie_access_traits, [977](#)
 - __gnu_pbds::trie_string_access_traits, [999](#)
 - std, [380](#)
 - std::_Temporary_buffer, [1151](#)
 - std::basic_string, [1227](#)
 - std::deque, [1417](#)
 - std::forward_list, [1460](#)
 - std::list, [1562](#)
 - std::map, [1601](#)
 - std::match_results, [1622](#)
 - std::multimap, [1685](#)

- std::multiset, 1704
- std::set, 1840
- std::unordered_map, 1932, 1933
- std::unordered_multimap, 1955
- std::unordered_multiset, 1977, 1978
- std::unordered_set, 2000, 2001
- std::vector, 2024
- Bernoulli Distributions, 25
 - operator!=, 26
 - operator<<, 26
 - operator>>, 27
- bernoulli_distribution
 - std::bernoulli_distribution, 1265
- beta
 - std::gamma_distribution, 1479
- bin_search_tree_.hpp, 2081
- binary
 - std::basic_ios, 1198
 - std::ios_base, 1534
- Binary Search, 29
 - binary_search, 30
 - equal_range, 30, 31
 - lower_bound, 31, 32
 - upper_bound, 32, 33
- binary_heap.hpp, 2081
- binary_heap_const_iterator_
 - __gnu_pbds::detail::binary_heap_const_iterator_, 788
- binary_heap_point_const_iterator_
 - __gnu_pbds::detail::binary_heap_point_const_iterator_, 792
- binary_search
 - Binary Search, 30
- bind1st
 - Binder Classes, 35
- bind2nd
 - Binder Classes, 35
- Binder Classes, 34
 - bind1st, 35
 - bind2nd, 35
- binders.h, 2082
- binomial_heap.hpp, 2082
- binomial_heap_base.hpp, 2083
- bitmap_allocator.h, 2083
 - _BALLOC_ALIGN_BYTES, 2084
- boolalpha
 - std, 381
 - std::basic_ios, 1198
 - std::ios_base, 1534
- Boolean Operations Classes, 36
- boost_concept_check.h, 2084
- Branch-Based, 37
- branch_policy.hpp, 2085
- bucket
 - __gnu_debug::_Safe_local_iterator, 600
- bucket_count
 - std::unordered_map, 1933
 - std::unordered_multimap, 1956
 - std::unordered_multiset, 1978
 - std::unordered_set, 2001
- c
 - std::queue, 1808
- c++0x_warning.h, 2085
- c++14_warning.h, 2086
- c++allocator.h, 2086
- c++config.h, 2086
- c++io.h, 2091
- c++locale.h, 2091
- c++locale_internal.h, 2092
- c_str
 - __gnu_cxx::__versa_string, 496
 - std::basic_string, 1228
- cache_line_size
 - __gnu_parallel::_Settings, 718
- capacity
 - __gnu_cxx::__versa_string, 496
 - std::basic_string, 1228
 - std::vector, 2024
- cast.h, 2092
- category
 - std::locale, 1576
- cbefore_begin
 - std::forward_list, 1460
- cbegin
 - __gnu_cxx::__versa_string, 496
 - std::basic_string, 1228
 - std::deque, 1417
 - std::forward_list, 1460
 - std::list, 1562
 - std::map, 1601
 - std::match_results, 1622
 - std::multimap, 1685
 - std::multiset, 1704
 - std::set, 1840
 - std::unordered_map, 1933
 - std::unordered_multimap, 1956
 - std::unordered_multiset, 1978
 - std::unordered_set, 2001
 - std::vector, 2025
- cc_hash_max_collision_check_resize_trigger
 - __gnu_pbds::cc_hash_max_collision_check_resize_trigger, 754
- cc_hash_max_collision_check_resize_trigger_imp.hpp, 2093
- cc_hash_table
 - __gnu_pbds::cc_hash_table, 759–761
- cc_ht_map.hpp, 2093

cend

- `__gnu_cxx::__versa_string`, 496
- `std::basic_string`, 1228
- `std::deque`, 1417
- `std::forward_list`, 1460
- `std::list`, 1562
- `std::map`, 1601
- `std::match_results`, 1622
- `std::multimap`, 1685
- `std::multiset`, 1704
- `std::set`, 1840
- `std::unordered_map`, 1934
- `std::unordered_multimap`, 1956, 1957
- `std::unordered_multiset`, 1979
- `std::unordered_set`, 2002
- `std::vector`, 2025

`char_traits.h`, 2093

`char_type`

- `std::__ctype_abstract_base`, 1044
- `std::basic_ios`, 1180
- `std::collate`, 1315
- `std::collate_byname`, 1321
- `std::ctype< char >`, 1345
- `std::ctype< wchar_t >`, 1359
- `std::ctype_byname< char >`, 1391
- `std::istreambuf_iterator`, 1542
- `std::messages`, 1641
- `std::money_get`, 1651
- `std::money_put`, 1655
- `std::moneypunct`, 1661
- `std::num_get`, 1730
- `std::num_put`, 1746
- `std::numpunct`, 1759
- `std::ostream_iterator`, 1770
- `std::ostreambuf_iterator`, 1773
- `std::time_get`, 1881
- `std::time_put`, 1897

`checkers.h`, 2094

`classic`

- `std::locale`, 1578

`classic_table`

- `std::ctype< char >`, 1346
- `std::ctype_byname< char >`, 1391

clear

- `__gnu_cxx::__versa_string`, 496
- `std::basic_ios`, 1184
- `std::basic_string`, 1228
- `std::deque`, 1417
- `std::forward_list`, 1461
- `std::list`, 1563
- `std::map`, 1601
- `std::multimap`, 1685
- `std::multiset`, 1704
- `std::set`, 1840

- `std::unordered_map`, 1934
- `std::unordered_multimap`, 1957
- `std::unordered_multiset`, 1979
- `std::unordered_set`, 2002
- `std::vector`, 2025

`cmp_fn`

- `__gnu_pbds::tree`, 984

`cmp_fn_imps.hpp`, 2094

code

- `std::regex_error`, 1813

`codecvt.h`, 2094

`codecvt_specializations.h`, 2095

collate

- `std::collate`, 1315
- `std::locale`, 1581
- `std::regex_constants`, 450

combine

- `std::locale`, 1578

compare

- `__gnu_cxx::__versa_string`, 496–499
- `std::basic_string`, 1228–1231
- `std::collate`, 1316
- `std::collate_byname`, 1321
- `std::sub_match`, 1875, 1876

Comparison Classes, 38

`compatibility.h`, 2095

`compiletime_settings.h`, 2096

- `_GLIBCXX_ASSERTIONS`, 2096

- `_GLIBCXX_CALL`, 2096

- `_GLIBCXX_RANDOM_SHUFFLE_CONSIDER_L1`, 2097

- `_GLIBCXX_RANDOM_SHUFFLE_CONSIDER_T↔LB`, 2097

- `_GLIBCXX_SCALE_DOWN_FPU`, 2097

- `_GLIBCXX_VERBOSE_LEVEL`, 2097

`complex.h`, 2098

`concept_check.h`, 2098

`concurrency.h`, 2098

Concurrency, 39

`cond_dealtor.hpp`, 2099

`cond_key_dtor_entry_dealtor.hpp`, 2099

`const_iterator`

- `__gnu_pbds::trie_string_access_traits`, 998

- `std::set`, 1835

- `std::unordered_map`, 1927

- `std::unordered_multimap`, 1950

- `std::unordered_multiset`, 1973

- `std::unordered_set`, 1996

`const_iterator.hpp`, 2099, 2100

`const_iterator_`, 1020

- `const_iterator_`, 1022

- `const_pointer`, 1021

- `const_reference`, 1021

- `difference_type`, 1021

- iterator_category, 1021
- m_p_tbl, 1023
- operator!=, 1022
- operator*, 1022
- operator++, 1023
- operator->, 1023
- operator==, 1023
- pointer, 1021
- reference, 1022
- value_type, 1022
- const_local_iterator
 - std::unordered_map, 1927
 - std::unordered_multimap, 1951
 - std::unordered_multiset, 1973
 - std::unordered_set, 1996
- const_pointer
 - __gnu_pbds::detail::binary_heap_const_iterator_, 787
 - __gnu_pbds::detail::binary_heap_point_const_iterator_, 791
 - __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_, 837
 - __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_, 841
 - const_iterator_, 1021
 - iterator_, 1025
 - point_const_iterator_, 1029
 - point_iterator_, 1032
 - std::allocator_traits, 1157
 - std::allocator_traits< allocator< _Tp > >, 1162
 - std::set, 1835
 - std::unordered_map, 1927
 - std::unordered_multimap, 1951
 - std::unordered_multiset, 1973
 - std::unordered_set, 1996
- const_pointer_cast
 - std, 381
- const_reference
 - __gnu_pbds::detail::bin_search_tree_const_node_it_, 774
 - __gnu_pbds::detail::bin_search_tree_node_it_, 779
 - __gnu_pbds::detail::binary_heap_const_iterator_, 787
 - __gnu_pbds::detail::binary_heap_point_const_iterator_, 791
 - __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_, 837
 - __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_, 841
 - const_iterator_, 1021
 - iterator_, 1025
 - point_const_iterator_, 1029
 - point_iterator_, 1032
 - std::set, 1835
 - std::unordered_map, 1927
 - std::unordered_multimap, 1951
 - std::unordered_multiset, 1973
 - std::unordered_set, 1996
- const_reverse_iterator
 - std::set, 1835
- const_void_pointer
 - __gnu_cxx::__alloc_traits, 460
 - std::allocator_traits, 1157
 - std::allocator_traits< allocator< _Tp > >, 1162
- construct
 - __gnu_cxx::__alloc_traits, 462
 - std::allocator_traits, 1159
 - std::allocator_traits< allocator< _Tp > >, 1164
- constructor_destructor_fn_imps.hpp, 2101
- constructor_destructor_no_store_hash_fn_imps.hpp, 2101
- constructor_destructor_store_hash_fn_imps.hpp, 2101
- constructors_destructor_fn_imps.hpp, 2102, 2103
- container_base_dispatch.hpp, 2104
- container_type
 - std::back_insert_iterator, 1174
 - std::front_insert_iterator, 1475
 - std::insert_iterator, 1520
- Containers, 40, 41
- copy
 - __gnu_cxx::__versa_string, 499
 - Mutating, 72
 - std::basic_string, 1231
- copy_backward
 - Mutating, 72
- copy_exception
 - Exceptions, 46
- copy_if
 - Mutating, 73
- copy_n
 - Mutating, 73
- copyfmt
 - std::basic_ios, 1185
- count
 - Non-Mutating, 95
 - std::map, 1602
 - std::multimap, 1685
 - std::multiset, 1704
 - std::set, 1841
 - std::unordered_map, 1934
 - std::unordered_multimap, 1957
 - std::unordered_multiset, 1979
 - std::unordered_set, 2002
- count_if
 - Non-Mutating, 95
- count_minimal_n
 - __gnu_parallel::_Settings, 718
- cpp_type_traits.h, 2104

- cpu_defines.h, 2105
- crbegin
 - __gnu_cxx::__versa_string, 500
 - std::basic_string, 1232
 - std::deque, 1418
 - std::list, 1563
 - std::map, 1603
 - std::multimap, 1686
 - std::multiset, 1705
 - std::set, 1841
 - std::vector, 2025
- cregex_token_iterator
 - Regular Expressions, 145
- crend
 - __gnu_cxx::__versa_string, 500
 - std::basic_string, 1232
 - std::deque, 1418
 - std::list, 1563
 - std::map, 1603
 - std::multimap, 1686
 - std::multiset, 1705
 - std::set, 1841
 - std::vector, 2025
- csub_match
 - Regular Expressions, 145
- ctype
 - std::ctype< char >, 1345, 1346
 - std::ctype< wchar_t >, 1359, 1360
 - std::locale, 1581
- ctype_base.h, 2105
- ctype_inline.h, 2105
- cur
 - std::basic_ios, 1198
 - std::ios_base, 1534
- curr_symbol
 - std::moneypunct, 1662
 - std::moneypunct_byname, 1671
- current_exception
 - Exceptions, 46
- cxxabi.h, 2106
 - __cxa_demangle, 2107
- cxxabi_forced.h, 2108
- cxxabi_tweaks.h, 2108
- data
 - __gnu_cxx::__versa_string, 500
 - std::basic_string, 1232
 - std::vector, 2025
- Data Structure Type, 42
- date_order
 - std::time_get, 1881
 - std::time_get_byname, 1890
- deallocate
 - __gnu_cxx::__alloc_traits, 462
 - std::allocator_traits, 1159
 - std::allocator_traits< allocator< _Tp > >, 1164
- debug.h, 2109
- debug_allocator.h, 2109
- debug_fn_imps.hpp, 2110–2112
- debug_map_base.hpp, 2112
- debug_no_store_hash_fn_imps.hpp, 2113
- debug_store_hash_fn_imps.hpp, 2113
- dec
 - std, 381
 - std::basic_ios, 1198
 - std::ios_base, 1534
- decimal_point
 - std::moneypunct, 1662
 - std::moneypunct_byname, 1671
 - std::numpunct, 1760
 - std::numpunct_byname, 1765
- default_delete
 - std::default_delete, 1402
 - std::default_delete< _Tp[]>, 1403
- densities
 - std::piecewise_constant_distribution, 1783
 - std::piecewise_linear_distribution, 1787
- deque
 - std::deque, 1409–1411
- destroy
 - __gnu_cxx::__alloc_traits, 462
 - std::allocator_traits, 1160
 - std::allocator_traits< allocator< _Tp > >, 1165
- Diagnostics, 43
- difference_type
 - __gnu_pbds::detail::bin_search_tree_const_node_↔ it_, 774
 - __gnu_pbds::detail::bin_search_tree_node_it_, 779
 - __gnu_pbds::detail::binary_heap_const_iterator_, 787
 - __gnu_pbds::detail::binary_heap_point_const_↔ iterator_, 791
 - __gnu_pbds::detail::left_child_next_sibling_heap_↔ const_iterator_, 837
 - __gnu_pbds::detail::left_child_next_sibling_heap_↔ node_point_const_iterator_, 842
 - const_iterator_, 1021
 - iterator_, 1025
 - point_const_iterator_, 1030
 - point_iterator_, 1033
 - std::allocator_traits, 1157
 - std::allocator_traits< allocator< _Tp > >, 1162
 - std::back_insert_iterator, 1174
 - std::front_insert_iterator, 1475
 - std::insert_iterator, 1520
 - std::istream_iterator, 1539
 - std::istreambuf_iterator, 1542
 - std::iterator, 1545

- std::ostream_iterator, 1770
- std::ostreambuf_iterator, 1773
- std::pointer_traits, 1796
- std::pointer_traits< _Tp * >, 1797
- std::raw_storage_iterator, 1811
- std::set, 1836
- std::unordered_map, 1928
- std::unordered_multimap, 1951
- std::unordered_multiset, 1973
- std::unordered_set, 1996
- direct_mask_range_hashing_imp.hpp, 2113
- direct_mod_range_hashing_imp.hpp, 2113
- discard
 - std::discard_block_engine, 1430
 - std::independent_bits_engine, 1514
 - std::linear_congruential_engine, 1551
 - std::mersenne_twister_engine, 1636
 - std::shuffle_order_engine, 1863
- discard_block_engine
 - std::discard_block_engine, 1428, 1429
- distance
 - std, 381
- do_compare
 - std::collate, 1316
 - std::collate_byname, 1321
- do_curr_symbol
 - std::moneypunct, 1662
 - std::moneypunct_byname, 1671
- do_date_order
 - std::time_get, 1882
 - std::time_get_byname, 1890
- do_decimal_point
 - std::moneypunct, 1663
 - std::moneypunct_byname, 1672
 - std::numpunct, 1760
 - std::numpunct_byname, 1765
- do_falsename
 - std::numpunct, 1760
 - std::numpunct_byname, 1766
- do_frac_digits
 - std::moneypunct, 1663
 - std::moneypunct_byname, 1672
- do_get
 - std::messages, 1642
 - std::messages_byname, 1645
 - std::money_get, 1652
 - std::num_get, 1731–1736
- do_get_date
 - std::time_get, 1882
 - std::time_get_byname, 1890
- do_get_monthname
 - std::time_get, 1882
 - std::time_get_byname, 1891
- do_get_time
 - std::time_get, 1883
 - std::time_get_byname, 1891
- do_get_weekday
 - std::time_get, 1883
 - std::time_get_byname, 1892
- do_get_year
 - std::time_get, 1884
 - std::time_get_byname, 1892
- do_grouping
 - std::moneypunct, 1663
 - std::moneypunct_byname, 1672
 - std::numpunct, 1761
 - std::numpunct_byname, 1766
- do_hash
 - std::collate, 1317
 - std::collate_byname, 1322
- do_is
 - std::__ctype_abstract_base, 1044, 1045
 - std::ctype, 1331, 1332
 - std::ctype< wchar_t >, 1360
 - std::ctype_byname, 1376
- do_narrow
 - std::__ctype_abstract_base, 1045, 1046
 - std::ctype, 1332, 1333
 - std::ctype< char >, 1346, 1347
 - std::ctype< wchar_t >, 1361
 - std::ctype_byname, 1376, 1377
 - std::ctype_byname< char >, 1391
- do_neg_format
 - std::moneypunct, 1664
 - std::moneypunct_byname, 1673
- do_negative_sign
 - std::moneypunct, 1664
 - std::moneypunct_byname, 1673
- do_out
 - std::__codecvt_abstract_base, 1039
 - std::codecvt, 1292
 - std::codecvt< _InternT, _ExternT, encoding_state >, 1296
 - std::codecvt< char, char, mbstate_t >, 1301
 - std::codecvt< wchar_t, char, mbstate_t >, 1305
 - std::codecvt_byname, 1311
- do_pos_format
 - std::moneypunct, 1664
 - std::moneypunct_byname, 1673
- do_positive_sign
 - std::moneypunct, 1665
 - std::moneypunct_byname, 1674
- do_put
 - std::money_put, 1656
 - std::num_put, 1747–1750
 - std::time_put, 1898
 - std::time_put_byname, 1902
- do_scan_is

- std::__ctype_abstract_base, 1046
- std::ctype, 1333
- std::ctype< wchar_t >, 1362
- std::ctype_byname, 1378
- do_scan_not
 - std::__ctype_abstract_base, 1047
 - std::ctype, 1334
 - std::ctype< wchar_t >, 1363
 - std::ctype_byname, 1378
- do_thousands_sep
 - std::moneypunct, 1665
 - std::moneypunct_byname, 1674
 - std::numpunct, 1761
 - std::numpunct_byname, 1766
- do_tolower
 - std::__ctype_abstract_base, 1047, 1048
 - std::ctype, 1334, 1335
 - std::ctype< char >, 1347, 1348
 - std::ctype< wchar_t >, 1363, 1364
 - std::ctype_byname, 1379
 - std::ctype_byname< char >, 1392
- do_toupper
 - std::__ctype_abstract_base, 1048, 1049
 - std::ctype, 1335, 1336
 - std::ctype< char >, 1348, 1349
 - std::ctype< wchar_t >, 1364
 - std::ctype_byname, 1380
 - std::ctype_byname< char >, 1393
- do_transform
 - std::collate, 1317
 - std::collate_byname, 1322
- do_truename
 - std::numpunct, 1761
 - std::numpunct_byname, 1767
- do_widen
 - std::__ctype_abstract_base, 1049, 1050
 - std::ctype, 1336, 1337
 - std::ctype< char >, 1349
 - std::ctype< wchar_t >, 1365
 - std::ctype_byname, 1381
 - std::ctype_byname< char >, 1394
- dynamic_pointer_cast
 - std, 382
- e_pos
 - __gnu_pbds::sample_trie_access_traits, 977
 - __gnu_pbds::trie_string_access_traits, 999
- e_type
 - __gnu_pbds::sample_trie_access_traits, 977
 - __gnu_pbds::trie_string_access_traits, 999
- ECMAScript
 - std::regex_constants, 450
- egrep
 - std::regex_constants, 451
- element_type
 - std::auto_ptr, 1168
 - std::pointer_traits, 1796
 - std::pointer_traits< _Tp * >, 1797
- emplace
 - std::deque, 1418
 - std::list, 1563
 - std::map, 1603
 - std::multimap, 1686
 - std::multiset, 1705
 - std::set, 1841
 - std::unordered_map, 1935
 - std::unordered_multimap, 1958
 - std::unordered_multiset, 1980
 - std::unordered_set, 2003
 - std::vector, 2025
- emplace_after
 - std::forward_list, 1461
- emplace_front
 - std::forward_list, 1461
- emplace_hint
 - std::map, 1604
 - std::multimap, 1686
 - std::multiset, 1705
 - std::set, 1842
 - std::unordered_map, 1935
 - std::unordered_multimap, 1958
 - std::unordered_multiset, 1980
 - std::unordered_set, 2003
- empty
 - __gnu_cxx::__versa_string, 500
 - __gnu_pbds::detail::cc_ht_map, 803
 - __gnu_pbds::detail::gp_ht_map, 829
 - std::basic_string, 1232
 - std::deque, 1418
 - std::forward_list, 1462
 - std::list, 1563
 - std::map, 1604
 - std::match_results, 1622
 - std::multimap, 1687
 - std::multiset, 1706
 - std::priority_queue, 1804
 - std::queue, 1807
 - std::set, 1842
 - std::stack, 1869
 - std::unordered_map, 1936
 - std::unordered_multimap, 1959
 - std::unordered_multiset, 1981
 - std::unordered_set, 2004
 - std::vector, 2026
- enable_special_members.h, 2114
- enc_filebuf.h, 2114
- end
 - __gnu_cxx::__versa_string, 501

- [__gnu_parallel::PseudoSequence, 707](#)
- [__gnu_pbds::sample_trie_access_traits, 977](#)
- [__gnu_pbds::trie_string_access_traits, 999](#)
- [std, 382, 384](#)
- [std::_Temporary_buffer, 1151](#)
- [std::basic_ios, 1199](#)
- [std::basic_string, 1232, 1233](#)
- [std::deque, 1418, 1419](#)
- [std::forward_list, 1462](#)
- [std::ios_base, 1534](#)
- [std::list, 1564](#)
- [std::map, 1604](#)
- [std::match_results, 1623](#)
- [std::multimap, 1687](#)
- [std::multiset, 1706](#)
- [std::set, 1842](#)
- [std::unordered_map, 1936, 1937](#)
- [std::unordered_multimap, 1959](#)
- [std::unordered_multiset, 1981](#)
- [std::unordered_set, 2004, 2005](#)
- [std::vector, 2026](#)
- [entry_cmp.hpp, 2114](#)
- [entry_list_fn_imps.hpp, 2115](#)
- [entry_metadata_base.hpp, 2115](#)
- [entry_pred.hpp, 2115](#)
- [eof](#)
 - [std::basic_ios, 1185](#)
- [eofbit](#)
 - [std::basic_ios, 1199](#)
 - [std::ios_base, 1534](#)
- [eq_by_less.hpp, 2116](#)
- [equal](#)
 - [Non-Mutating, 95, 96](#)
 - [std::istreambuf_iterator, 1544](#)
- [equal_range](#)
 - [Binary Search, 30, 31](#)
 - [std::map, 1605](#)
 - [std::multimap, 1687, 1688](#)
 - [std::multiset, 1706, 1707](#)
 - [std::set, 1843](#)
 - [std::unordered_map, 1937](#)
 - [std::unordered_multimap, 1960](#)
 - [std::unordered_multiset, 1982](#)
 - [std::unordered_set, 2005](#)
- [equally_split.h, 2116](#)
- [erase](#)
 - [__gnu_cxx::__versa_string, 501, 502](#)
 - [std::basic_string, 1233, 1234](#)
 - [std::deque, 1419](#)
 - [std::list, 1564](#)
 - [std::map, 1606, 1607](#)
 - [std::multimap, 1688, 1689](#)
 - [std::multiset, 1707, 1708](#)
 - [std::set, 1844, 1845](#)
 - [std::unordered_map, 1938, 1939](#)
 - [std::unordered_multimap, 1960–1962](#)
 - [std::unordered_multiset, 1983, 1984](#)
 - [std::unordered_set, 2006, 2007](#)
 - [std::vector, 2026, 2027](#)
- [erase_after](#)
 - [std::forward_list, 1462, 1463](#)
- [erase_can_throw](#)
 - [__gnu_pbds::container_traits, 765](#)
- [erase_fn_imps.hpp, 2116–2119](#)
- [erase_no_store_hash_fn_imps.hpp, 2119](#)
- [erase_store_hash_fn_imps.hpp, 2119](#)
- [error_backref](#)
 - [std::regex_constants, 451](#)
- [error_badbrace](#)
 - [std::regex_constants, 451](#)
- [error_badrepeat](#)
 - [std::regex_constants, 451](#)
- [error_brace](#)
 - [std::regex_constants, 451](#)
- [error_brack](#)
 - [std::regex_constants, 451](#)
- [error_collate](#)
 - [std::regex_constants, 451](#)
- [error_complexity](#)
 - [std::regex_constants, 452](#)
- [error_constants.h, 2119](#)
- [error_ctype](#)
 - [std::regex_constants, 452](#)
- [error_escape](#)
 - [std::regex_constants, 452](#)
- [error_paren](#)
 - [std::regex_constants, 452](#)
- [error_range](#)
 - [std::regex_constants, 452](#)
- [error_space](#)
 - [std::regex_constants, 452](#)
- [error_stack](#)
 - [std::regex_constants, 452](#)
- [error_type](#)
 - [std::regex_constants, 449](#)
- [event](#)
 - [std::basic_ios, 1183](#)
 - [std::ios_base, 1527](#)
- [event_callback](#)
 - [std::basic_ios, 1180](#)
 - [std::ios_base, 1525](#)
- [exception.hpp, 2120](#)
- [exception_defines.h, 2121](#)
- [exception_ptr.h, 2121](#)
- [Exceptions, 44, 45](#)
 - [copy_exception, 46](#)
 - [current_exception, 46](#)
 - [make_exception_ptr, 46](#)

- rethrow_exception, 46
 - rethrow_if_nested, 46
 - throw_with_nested, 46
- exceptions
 - std::basic_ios, 1185, 1186
- exponential_distribution
 - std::exponential_distribution, 1441
- extc++.h, 2122
- extended
 - std::regex_constants, 451
- Extensions, 47
- external_load_access
 - __gnu_pbds::cc_hash_max_collision_check_↔
resize_trigger, 754
 - __gnu_pbds::hash_load_check_resize_trigger, 942
- extptr_allocator.h, 2122
- facet
 - std::locale::facet, 1585
- fail
 - std::basic_ios, 1186
- failbit
 - std::basic_ios, 1199
 - std::ios_base, 1534
- failed
 - std::ostreambuf_iterator, 1775
- falsename
 - std::numpunct, 1762
 - std::numpunct_byname, 1767
- fd
 - __gnu_cxx::stdio_filebuf, 567
- features.h, 2122
 - _GLIBCXX_BAL_QUICKSORT, 2123
 - _GLIBCXX_FIND_CONSTANT_SIZE_BLOCKS,
2123
 - _GLIBCXX_FIND_EQUAL_SPLIT, 2123
 - _GLIBCXX_FIND_GROWING_BLOCKS, 2123
 - _GLIBCXX_MERGESORT, 2123
 - _GLIBCXX_QUICKSORT, 2124
 - _GLIBCXX_TREE_DYNAMIC_BALANCING, 2124
 - _GLIBCXX_TREE_FULL_COPY, 2124
 - _GLIBCXX_TREE_INITIAL_SPLITTING, 2124
- fenv.h, 2125
- file
 - __gnu_cxx::stdio_filebuf, 567
 - __gnu_cxx::stdio_sync_filebuf, 569
- fill
 - Mutating, 74
 - std::basic_ios, 1187
- fill_minimal_n
 - __gnu_parallel::_Settings, 718
- fill_n
 - Mutating, 74
- find
 - __gnu_cxx::__versa_string, 502–504
 - Non-Mutating, 96
 - std::basic_string, 1234, 1235
 - std::map, 1607, 1608
 - std::multimap, 1690
 - std::multiset, 1709
 - std::set, 1845, 1846
 - std::unordered_map, 1940
 - std::unordered_multimap, 1962, 1963
 - std::unordered_multiset, 1984, 1985
 - std::unordered_set, 2008
- find.h, 2125
- find_by_order
 - __gnu_pbds::tree_order_statistics_node_update,
987
 - __gnu_pbds::trie_order_statistics_node_update, 993
- find_end
 - Non-Mutating, 97
- find_first_not_of
 - __gnu_cxx::__versa_string, 504–506
 - std::basic_string, 1236, 1237
- find_first_of
 - __gnu_cxx::__versa_string, 506, 507
 - Non-Mutating, 98
 - std::basic_string, 1237–1239
- find_fn_imps.hpp, 2125–2127
- find_if
 - Non-Mutating, 99
- find_if_not
 - Non-Mutating, 99
- find_increasing_factor
 - __gnu_parallel::_Settings, 718
- find_initial_block_size
 - __gnu_parallel::_Settings, 718
- find_last_not_of
 - __gnu_cxx::__versa_string, 508, 509
 - std::basic_string, 1239, 1240
- find_last_of
 - __gnu_cxx::__versa_string, 510, 511
 - std::basic_string, 1241, 1242
- find_maximum_block_size
 - __gnu_parallel::_Settings, 718
- find_no_store_hash_fn_imps.hpp, 2127
- find_scale_factor
 - __gnu_parallel::_Settings, 718
- find_selectors.h, 2127
- find_sequential_search_size
 - __gnu_parallel::_Settings, 719
- find_store_hash_fn_imps.hpp, 2128
- first
 - __gnu_parallel::_IteratorPair, 672
 - std::pair, 1781
 - std::sub_match, 1877
- first_argument_type

- `__gnu_parallel::_EqualFromLess`, 666
- `__gnu_parallel::_EqualTo`, 667
- `__gnu_parallel::_Less`, 675
- `__gnu_parallel::_Lexicographic`, 676
- `__gnu_parallel::_LexicographicReverse`, 678
- `__gnu_parallel::_Multiplies`, 700
- `__gnu_parallel::_Plus`, 703
- `std::binary_function`, 1269
- `std::binary_negate`, 1270
- `std::const_mem_fun1_ref_t`, 1325
- `std::const_mem_fun1_t`, 1326
- `std::divides`, 1437
- `std::equal_to`, 1439
- `std::greater`, 1486
- `std::greater_equal`, 1487
- `std::less`, 1548
- `std::less_equal`, 1549
- `std::logical_and`, 1587
- `std::logical_or`, 1590
- `std::mem_fun1_ref_t`, 1629
- `std::mem_fun1_t`, 1631
- `std::minus`, 1646
- `std::modulus`, 1647
- `std::multiplies`, 1698
- `std::not_equal_to`, 1728
- `std::owner_less< shared_ptr< _Tp > >`, 1777
- `std::owner_less< weak_ptr< _Tp > >`, 1778
- `std::plus`, 1791
- `std::pointer_to_binary_function`, 1793
- fixed
 - `std`, 384
 - `std::basic_ios`, 1199
 - `std::ios_base`, 1535
- flags
 - `std::basic_ios`, 1188
 - `std::basic_regex`, 1210
 - `std::ios_base`, 1527, 1528
- floatfield
 - `std::basic_ios`, 1199
 - `std::ios_base`, 1535
- fmtflags
 - `std::basic_ios`, 1181
 - `std::ios_base`, 1525
- for_each
 - Non-Mutating, 100
- `for_each.h`, 2128
- `for_each_minimal_n`
 - `__gnu_parallel::_Settings`, 719
- `for_each_selectors.h`, 2129
- format
 - `std::match_results`, 1623
- format_default
 - `std::regex_constants`, 450
- format_first_only
 - `std::regex_constants`, 450
- format_no_copy
 - `std::regex_constants`, 450
- format_sed
 - `std::regex_constants`, 450
- formatter.h, 2129
- forward
 - Utilities, 213
- forward_list
 - `std::forward_list`, 1456–1458
- `forward_list.h`, 2130
- fpos
 - `std::fpos`, 1473
- frac_digits
 - `std::moneypunct`, 1665
 - `std::moneypunct_byname`, 1674
- front
 - `__gnu_cxx::__versa_string`, 511, 512
 - `std::basic_string`, 1242
 - `std::deque`, 1420
 - `std::forward_list`, 1463
 - `std::list`, 1565
 - `std::queue`, 1807
 - `std::vector`, 2027
- front_insert_iterator
 - `std::front_insert_iterator`, 1476
- front_inserter
 - Iterators, 65
- functexcept.h, 2131
- Function Objects, 48
- `functional_hash.h`, 2132
- `functions.h`, 2133
- gamma_distribution
 - `std::gamma_distribution`, 1478
- generate
 - Mutating, 74
- generate_canonical
 - Random Number Generation, 133
- generate_minimal_n
 - `__gnu_parallel::_Settings`, 719
- generate_n
 - Mutating, 75
- get
 - `__gnu_parallel::_Settings`, 717
 - `std::auto_ptr`, 1170
 - `std::money_get`, 1652, 1653
 - `std::num_get`, 1737–1743
 - `std::unique_ptr`, 1917
 - `std::unique_ptr< _Tp[], _Dp >`, 1922
- get_actual_size
 - `__gnu_pbds::hash_standard_resize_policy`, 946
- get_allocator
 - `__gnu_cxx::__versa_string`, 512

- std::basic_string, 1243
- std::deque, 1420
- std::forward_list, 1463
- std::list, 1565
- std::map, 1608
- std::match_results, 1624
- std::multimap, 1690
- std::multiset, 1709
- std::set, 1846
- std::unordered_map, 1940
- std::unordered_multimap, 1963
- std::unordered_multiset, 1985
- std::unordered_set, 2008
- get_child
 - __gnu_pbds::detail::pat_trie_base::_Node_citer, 876
 - __gnu_pbds::detail::pat_trie_base::_Node_iter, 879
- get_comb_hash_fn
 - __gnu_pbds::detail::cc_ht_map, 803
- get_comb_probe_fn
 - __gnu_pbds::detail::gp_ht_map, 829
- get_date
 - std::time_get, 1885
 - std::time_get_byname, 1893
- get_deleter
 - Pointer_abstractions, 124
 - std::unique_ptr, 1917
 - std::unique_ptr< _Tp[], _Dp >, 1922
- get_eq_fn
 - __gnu_pbds::detail::cc_ht_map, 803
 - __gnu_pbds::detail::gp_ht_map, 830
- get_hash_fn
 - __gnu_pbds::detail::cc_ht_map, 803, 804
 - __gnu_pbds::detail::gp_ht_map, 830
- get_l_child
 - __gnu_pbds::detail::bin_search_tree_const_node_↵ it_, 775
 - __gnu_pbds::detail::bin_search_tree_node_it_, 780
 - __gnu_pbds::detail::ov_tree_node_const_it_, 855
 - __gnu_pbds::detail::ov_tree_node_it_, 857
- get_load
 - __gnu_pbds::cc_hash_max_collision_check_↵ resize_trigger, 755
- get_loads
 - __gnu_pbds::hash_load_check_resize_trigger, 942
- get_metadata
 - __gnu_pbds::detail::bin_search_tree_const_node_↵ it_, 775
 - __gnu_pbds::detail::bin_search_tree_node_it_, 780
 - __gnu_pbds::detail::pat_trie_base::_Node_citer, 876
 - __gnu_pbds::detail::pat_trie_base::_Node_iter, 879
- get_monthname
 - std::time_get, 1885
 - std::time_get_byname, 1893
- get_nearest_larger_size
 - __gnu_pbds::sample_size_policy, 975
- get_nearest_smaller_size
 - __gnu_pbds::sample_size_policy, 975
- get_new_size
 - __gnu_pbds::hash_standard_resize_policy, 946
 - __gnu_pbds::sample_resize_policy, 970
- get_probe_fn
 - __gnu_pbds::detail::gp_ht_map, 830
- get_r_child
 - __gnu_pbds::detail::bin_search_tree_const_node_↵ it_, 775
 - __gnu_pbds::detail::bin_search_tree_node_it_, 780
 - __gnu_pbds::detail::ov_tree_node_const_it_, 855
 - __gnu_pbds::detail::ov_tree_node_it_, 857
- get_resize_policy
 - __gnu_pbds::detail::cc_ht_map, 804
 - __gnu_pbds::detail::gp_ht_map, 831
- get_size_policy
 - __gnu_pbds::hash_standard_resize_policy, 946
- get_temporary_buffer
 - std, 384
- get_time
 - std::time_get, 1886
 - std::time_get_byname, 1894
- get_trigger_policy
 - __gnu_pbds::hash_standard_resize_policy, 947
- get_weekday
 - std::time_get, 1886
 - std::time_get_byname, 1894
- get_year
 - std::time_get, 1887
 - std::time_get_byname, 1895
- getline
 - std, 384–386
- getloc
 - std::basic_ios, 1188
 - std::basic_regex, 1210
 - std::ios_base, 1528
 - std::regex_traits, 1821
- global
 - std::locale, 1579
- good
 - std::basic_ios, 1188
- goodbit
 - std::basic_ios, 1199
 - std::ios_base, 1535
- gp_hash_table
 - __gnu_pbds::gp_hash_table, 936–938
- gp_ht_map.hpp, 2135
- grep
 - std::regex_constants, 451
- grouping
 - std::moneypunct, 1666
 - std::moneypunct_byname, 1675

- std::numpunct, [1762](#)
 - std::numpunct_byname, [1767](#)
- gslice
 - Numeric_arrays, [111](#), [112](#)
- gslice.h, [2136](#)
- gslice_array
 - Numeric_arrays, [112](#)
- gslice_array.h, [2136](#)
- hash
 - std::collate, [1318](#)
 - std::collate_byname, [1323](#)
- Hash-Based, [50](#)
- hash_bytes.h, [2137](#)
- hash_eq_fn.hpp, [2137](#)
- hash_exponential_size_policy
 - __gnu_pbds::hash_exponential_size_policy, [940](#)
- hash_exponential_size_policy_imp.hpp, [2138](#)
- hash_fun.h, [2138](#)
- hash_function
 - std::unordered_map, [1941](#)
 - std::unordered_multimap, [1963](#)
 - std::unordered_multiset, [1985](#)
 - std::unordered_set, [2009](#)
- hash_load_check_resize_trigger
 - __gnu_pbds::hash_load_check_resize_trigger, [942](#)
- hash_load_check_resize_trigger_imp.hpp, [2138](#)
- hash_load_check_resize_trigger_size_base.hpp, [2138](#)
- hash_policy.hpp, [2139](#)
- hash_prime_size_policy
 - __gnu_pbds::hash_prime_size_policy, [944](#)
- hash_prime_size_policy_imp.hpp, [2140](#)
- hash_standard_resize_policy
 - __gnu_pbds::hash_standard_resize_policy, [945](#), [946](#)
- hash_standard_resize_policy_imp.hpp, [2140](#)
- hasher
 - std::unordered_map, [1928](#)
 - std::unordered_multimap, [1951](#)
 - std::unordered_multiset, [1974](#)
 - std::unordered_set, [1997](#)
- Hashes, [51](#)
- hashtable.h, [2140](#), [2141](#)
- hashtable_policy.h, [2142](#)
- Heap, [52](#)
 - is_heap, [52](#), [53](#)
 - is_heap_until, [53](#), [54](#)
 - make_heap, [54](#)
 - pop_heap, [55](#)
 - push_heap, [55](#), [56](#)
 - sort_heap, [56](#)
- Heap-Based, [58](#)
 - priority_queue, [59](#)
- hex
 - std, [387](#)
- std::basic_ios, [1200](#)
- std::ios_base, [1535](#)
- icase
 - std::regex_constants, [450](#)
- id
 - std::collate, [1319](#)
 - std::collate_byname, [1324](#)
 - std::ctype, [1343](#)
 - std::ctype< char >, [1356](#)
 - std::ctype< wchar_t >, [1372](#)
 - std::ctype_byname, [1388](#)
 - std::ctype_byname< char >, [1401](#)
 - std::locale::id, [1586](#)
 - std::messages, [1642](#)
 - std::messages_byname, [1645](#)
 - std::money_get, [1653](#)
 - std::money_put, [1658](#)
 - std::moneypunct, [1669](#)
 - std::moneypunct_byname, [1678](#)
 - std::num_get, [1744](#)
 - std::num_put, [1756](#)
 - std::numpunct, [1763](#)
 - std::numpunct_byname, [1769](#)
 - std::time_get, [1887](#)
 - std::time_get_byname, [1895](#)
 - std::time_put, [1900](#)
 - std::time_put_byname, [1904](#)
- imbue
 - std::basic_ios, [1189](#)
 - std::basic_regex, [1210](#)
 - std::ios_base, [1528](#)
 - std::regex_traits, [1821](#)
- in
 - std::__codecvt_abstract_base, [1039](#)
 - std::basic_ios, [1200](#)
 - std::codecvt, [1292](#)
 - std::codecvt< _InternT, _ExternT, encoding_state >, [1296](#)
 - std::codecvt< char, char, mbstate_t >, [1301](#)
 - std::codecvt< wchar_t, char, mbstate_t >, [1305](#)
 - std::codecvt_byname, [1311](#)
 - std::ios_base, [1535](#)
- includes
 - Set Operation, [179](#)
- increment
 - std::linear_congruential_engine, [1554](#)
- independent_bits_engine
 - std::independent_bits_engine, [1512](#), [1513](#)
- indirect_array
 - Numeric_arrays, [112](#)
- indirect_array.h, [2144](#)
- info_fn_imps.hpp, [2144](#)–[2146](#)
- init

- std::basic_ios, 1189
- inner_product
 - std, 387
- inplace_merge
 - Sorting, 186
- insert
 - __gnu_cxx::__versa_string, 512–517
 - std::basic_string, 1243–1247
 - std::deque, 1420–1422
 - std::list, 1565–1567
 - std::map, 1608–1610
 - std::multimap, 1691, 1692
 - std::multiset, 1710, 1711
 - std::set, 1846, 1847
 - std::unordered_map, 1941–1943
 - std::unordered_multimap, 1963–1966
 - std::unordered_multiset, 1985, 1987–1989
 - std::unordered_set, 2009–2011
 - std::vector, 2028, 2029
- insert_after
 - std::forward_list, 1463–1465
- insert_fn_imps.hpp, 2146–2148
- insert_iterator
 - std::insert_iterator, 1521
- insert_join_fn_imps.hpp, 2148
- insert_no_store_hash_fn_imps.hpp, 2148
- insert_store_hash_fn_imps.hpp, 2148, 2149
- inserter
 - Iterators, 65
- int_type
 - std::basic_ios, 1181
 - std::istreambuf_iterator, 1542
- internal
 - std, 388
 - std::basic_ios, 1200
 - std::ios_base, 1535
- intervals
 - std::piecewise_constant_distribution, 1783
 - std::piecewise_linear_distribution, 1787
- intl
 - std::moneypunct, 1669
- Invalidation Guarantees, 60
- ios_base.h, 2149
- iostate
 - std::basic_ios, 1182
 - std::ios_base, 1526
- iota
 - std, 388
- is
 - std::__ctype_abstract_base, 1050, 1051
 - std::ctype, 1337, 1338
 - std::ctype< char >, 1350
 - std::ctype< wchar_t >, 1366
 - std::ctype_byname, 1382
 - std::ctype_byname< char >, 1395
- is_grow_needed
 - __gnu_pbds::cc_hash_max_collision_check_↔
resize_trigger, 755
 - __gnu_pbds::sample_resize_trigger, 973
- is_heap
 - Heap, 52, 53
- is_heap_until
 - Heap, 53, 54
- is_partitioned
 - Mutating, 75
- is_permutation
 - Non-Mutating, 100
- is_resize_needed
 - __gnu_pbds::cc_hash_max_collision_check_↔
resize_trigger, 755
 - __gnu_pbds::sample_resize_policy, 970
 - __gnu_pbds::sample_resize_trigger, 973
- is_sorted
 - Sorting, 187
- is_sorted_until
 - Sorting, 188
- isalnum
 - std, 388
- isalpha
 - std, 388
- isctrl
 - std, 389
- isctype
 - std::regex_traits, 1822
- isdigit
 - std, 389
- isgraph
 - std, 389
- islower
 - std, 389
- isprint
 - std, 389
- ispunct
 - std, 389
- isspace
 - std, 390
- istream_iterator
 - std::istream_iterator, 1540
- istream_type
 - std::istreambuf_iterator, 1542
- istreambuf_iterator
 - std::istreambuf_iterator, 1543, 1544
- isupper
 - std, 390
- isxdigit
 - std, 390
- iter_swap
 - Mutating, 76

- iter_type
 - std::money_get, 1651
 - std::money_put, 1655
 - std::num_get, 1730
 - std::num_put, 1746
 - std::time_get, 1881
 - std::time_put, 1897
- iterator
 - std::set, 1836
 - std::unordered_map, 1928
 - std::unordered_multimap, 1951
 - std::unordered_multiset, 1974
 - std::unordered_set, 1997
- Iterator Tags, 61
- iterator.h, 2150
- iterator.hpp, 2151
- iterator_, 1024
 - const_pointer, 1025
 - const_reference, 1025
 - difference_type, 1025
 - iterator_, 1026
 - iterator_category, 1025
 - m_p_tbl, 1028
 - operator const point_iterator_, 1026
 - operator point_iterator_, 1026
 - operator!=, 1026, 1027
 - operator*, 1027
 - operator++, 1027
 - operator->, 1027
 - operator==, 1027
 - pointer, 1025
 - reference, 1026
 - value_type, 1026
- iterator_category
 - __gnu_pbds::detail::bin_search_tree_const_node_↔ it_, 774
 - __gnu_pbds::detail::bin_search_tree_node_it_, 779
 - __gnu_pbds::detail::binary_heap_const_iterator_, 788
 - __gnu_pbds::detail::binary_heap_point_const_↔ iterator_, 791
 - __gnu_pbds::detail::left_child_next_sibling_heap_↔ const_iterator_, 837
 - __gnu_pbds::detail::left_child_next_sibling_heap_↔ node_point_const_iterator_, 842
 - const_iterator_, 1021
 - iterator_, 1025
 - point_const_iterator_, 1030
 - point_iterator_, 1033
 - std::back_insert_iterator, 1174
 - std::front_insert_iterator, 1475
 - std::insert_iterator, 1520
 - std::istream_iterator, 1539
 - std::istreambuf_iterator, 1542
 - std::iterator, 1545
 - std::ostream_iterator, 1770
 - std::ostreambuf_iterator, 1774
 - std::raw_storage_iterator, 1811
 - std::reverse_iterator, 1828
- iterator_fn_imps.hpp, 2151
- iterator_tracker.h, 2151
- Iterators, 62
 - __iterator_category, 65
 - back_inserter, 65
 - front_inserter, 65
 - inserter, 65
 - operator!=, 66
 - operator==, 66
- iterators_fn_imps.hpp, 2152, 2153
- iword
 - std::basic_ios, 1189
 - std::ios_base, 1529
- k
 - std::negative_binomial_distribution, 1718
- key_comp
 - std::map, 1610
 - std::multimap, 1692
 - std::multiset, 1711
 - std::set, 1848
- key_compare
 - std::set, 1836
- key_eq
 - std::unordered_map, 1944
 - std::unordered_multimap, 1966
 - std::unordered_multiset, 1989
 - std::unordered_set, 2011
- key_equal
 - std::unordered_map, 1928
 - std::unordered_multimap, 1952
 - std::unordered_multiset, 1974
 - std::unordered_set, 1997
- key_type
 - std::set, 1836
 - std::unordered_map, 1928
 - std::unordered_multimap, 1952
 - std::unordered_multiset, 1974
 - std::unordered_set, 1997
- kill_dependency
 - Atomics, 16
- L1_cache_size
 - __gnu_parallel::Settings, 719
- L2_cache_size
 - __gnu_parallel::Settings, 719
- lambda
 - std::exponential_distribution, 1442
- left
 - std, 390

- std::basic_ios, 1200
- std::ios_base, 1536
- left_child_next_sibling_heap_.hpp, 2153
- left_child_next_sibling_heap_const_iterator_
 - __gnu_pbds::detail::left_child_next_sibling_heap_↔
 - const_iterator_, 838
- left_child_next_sibling_heap_node_point_const_iterator↔
 -
 - __gnu_pbds::detail::left_child_next_sibling_heap_↔
 - node_point_const_iterator_, 843
- length
 - __gnu_cxx::__versa_string, 517
 - std::basic_string, 1248
 - std::match_results, 1624
 - std::regex_traits, 1822
 - std::sub_match, 1877
- lexicographical_compare
 - Sorting, 188, 189
- linear_congruential_engine
 - std::linear_congruential_engine, 1551
- linear_probe_fn_imp.hpp, 2154
- list
 - std::list, 1558–1560
- List-Based, 68
- list_partition
 - __gnu_parallel, 282
- list_partition.h, 2154
- list_update
 - __gnu_pbds::list_update, 950
- list_update_policy.hpp, 2154
- load_factor
 - std::unordered_map, 1944
 - std::unordered_multimap, 1966
 - std::unordered_multiset, 1989
 - std::unordered_set, 2012
- local_iterator
 - std::unordered_map, 1928
 - std::unordered_multimap, 1952
 - std::unordered_multiset, 1974
 - std::unordered_set, 1997
- locale
 - std::locale, 1577, 1578
- locale_classes.h, 2155
- locale_facets.h, 2155
- locale_facets_nonio.h, 2157
- localefwd.h, 2157
- Locales, 69
- lookup_classname
 - std::regex_traits, 1823
- lookup_collatename
 - std::regex_traits, 1823
- losertree.h, 2159
- lower_bound
 - Binary Search, 31, 32
- std::map, 1610, 1611
- std::multimap, 1693
- std::multiset, 1711, 1712
- std::set, 1848
- lu_counter_metadata.hpp, 2159
- lu_map_.hpp, 2160
- m_p_tbl
 - const_iterator_, 1023
 - iterator_, 1028
- macros.h, 2160
 - _GLIBCXX_DEBUG_VERIFY_AT, 2163
 - __glibcxx_check_erase, 2161
 - __glibcxx_check_erase_after, 2161
 - __glibcxx_check_erase_range, 2161
 - __glibcxx_check_erase_range_after, 2161
 - __glibcxx_check_heap_pred, 2162
 - __glibcxx_check_insert, 2162
 - __glibcxx_check_insert_after, 2162
 - __glibcxx_check_insert_range, 2162
 - __glibcxx_check_insert_range_after, 2162
 - __glibcxx_check_partitioned_lower, 2162
 - __glibcxx_check_partitioned_lower_pred, 2163
 - __glibcxx_check_partitioned_upper_pred, 2163
 - __glibcxx_check_sorted_pred, 2163
- make_exception_ptr
 - Exceptions, 46
- make_heap
 - Heap, 54
- make_pair
 - Utilities, 213
- make_shared
 - Pointer_abstractions, 125
- malloc_allocator.h, 2163
- map
 - std::map, 1597–1600
- map.h, 2164
- mapped_type
 - std::unordered_map, 1929
 - std::unordered_multimap, 1952
- mark_count
 - std::basic_regex, 1210
- mask_array
 - Numeric_arrays, 112
- mask_array.h, 2165
- mask_based_range_hashing.hpp, 2166
- match_any
 - std::regex_constants, 449
- match_continuous
 - std::regex_constants, 449
- match_default
 - std::regex_constants, 449
- match_flag_type
 - std::regex_constants, 449

- match_not_bol
 - std::regex_constants, 449
- match_not_bow
 - std::regex_constants, 449
- match_not_eol
 - std::regex_constants, 449
- match_not_eow
 - std::regex_constants, 449
- match_not_null
 - std::regex_constants, 449
- match_prev_avail
 - std::regex_constants, 449
- match_results
 - std::match_results, 1622
- max
 - __gnu_parallel, 283
 - Sorting, 189, 190
 - std::bernoulli_distribution, 1265
 - std::binomial_distribution, 1275
 - std::cauchy_distribution, 1279
 - std::chi_squared_distribution, 1287
 - std::discard_block_engine, 1430
 - std::discrete_distribution, 1434
 - std::exponential_distribution, 1442
 - std::extreme_value_distribution, 1445
 - std::fisher_f_distribution, 1449
 - std::gamma_distribution, 1479
 - std::geometric_distribution, 1483
 - std::independent_bits_engine, 1514
 - std::linear_congruential_engine, 1551
 - std::lognormal_distribution, 1592
 - std::mersenne_twister_engine, 1637
 - std::negative_binomial_distribution, 1718
 - std::normal_distribution, 1723
 - std::piecewise_constant_distribution, 1783
 - std::piecewise_linear_distribution, 1787
 - std::poisson_distribution, 1799
 - std::shuffle_order_engine, 1863
 - std::student_t_distribution, 1871
 - std::uniform_int_distribution, 1908
 - std::uniform_real_distribution, 1911
 - std::weibull_distribution, 2041
- max_bucket_count
 - std::unordered_map, 1944
 - std::unordered_multimap, 1966
 - std::unordered_multiset, 1989
 - std::unordered_set, 2012
- max_count
 - __gnu_pbds::lu_counter_policy, 952
- max_element
 - Sorting, 190, 191
- max_element_minimal_n
 - __gnu_parallel::_Settings, 719
- max_load_factor
 - std::unordered_map, 1944
 - std::unordered_multimap, 1966
 - std::unordered_multiset, 1989
 - std::unordered_set, 2012
- max_size
 - __gnu_cxx::__alloc_traits, 463
 - __gnu_cxx::__versa_string, 518
 - std::allocator_traits, 1160
 - std::allocator_traits< allocator< _Tp > >, 1165
 - std::basic_string, 1248
 - std::deque, 1422
 - std::forward_list, 1465
 - std::list, 1567
 - std::map, 1611
 - std::match_results, 1624
 - std::multimap, 1693
 - std::multiset, 1712
 - std::set, 1849
 - std::unordered_map, 1944
 - std::unordered_multimap, 1968
 - std::unordered_multiset, 1991
 - std::unordered_set, 2012
 - std::vector, 2030
- mean
 - std::normal_distribution, 1723
 - std::poisson_distribution, 1799
- memory_order
 - Atomics, 16
- memoryfwd.h, 2166
- merge
 - Sorting, 191
 - std::forward_list, 1465, 1466
 - std::list, 1567, 1568
- merge.h, 2166
- merge_minimal_n
 - __gnu_parallel::_Settings, 719
- merge_oversampling
 - __gnu_parallel::_Settings, 720
- mersenne_twister_engine
 - std::mersenne_twister_engine, 1636
- messages
 - std::locale, 1582
 - std::messages, 1641
- messages_members.h, 2167
- metadata_const_reference
 - __gnu_pbds::detail::bin_search_tree_const_node_↵
it_, 774
 - __gnu_pbds::detail::bin_search_tree_node_it_, 779
- metadata_reference
 - __gnu_pbds::lu_counter_policy, 952
 - __gnu_pbds::lu_move_to_front_policy, 953
- metadata_type
 - __gnu_pbds::detail::bin_search_tree_const_node_↵
it_, 775

- [__gnu_pbds::detail::bin_search_tree_node_it](#), 780
 - [__gnu_pbds::detail::pat_trie_base::_Node_citer](#), 876
 - [__gnu_pbds::detail::pat_trie_base::_Node_iter](#), 879
 - [__gnu_pbds::lu_counter_policy](#), 952
 - [__gnu_pbds::lu_move_to_front_policy](#), 953
 - [__gnu_pbds::sample_update_policy](#), 979
- min
 - [__gnu_parallel](#), 283
 - Sorting, 192, 193
 - [std::bernoulli_distribution](#), 1265
 - [std::binomial_distribution](#), 1275
 - [std::cauchy_distribution](#), 1279
 - [std::chi_squared_distribution](#), 1287
 - [std::discard_block_engine](#), 1430
 - [std::discrete_distribution](#), 1434
 - [std::exponential_distribution](#), 1442
 - [std::extreme_value_distribution](#), 1446
 - [std::fisher_f_distribution](#), 1449
 - [std::gamma_distribution](#), 1479
 - [std::geometric_distribution](#), 1483
 - [std::independent_bits_engine](#), 1514
 - [std::linear_congruential_engine](#), 1552
 - [std::lognormal_distribution](#), 1592
 - [std::mersenne_twister_engine](#), 1637
 - [std::negative_binomial_distribution](#), 1718
 - [std::normal_distribution](#), 1723
 - [std::piecewise_constant_distribution](#), 1783
 - [std::piecewise_linear_distribution](#), 1788
 - [std::poisson_distribution](#), 1799
 - [std::shuffle_order_engine](#), 1863
 - [std::student_t_distribution](#), 1871
 - [std::uniform_int_distribution](#), 1908
 - [std::uniform_real_distribution](#), 1912
 - [std::weibull_distribution](#), 2041
- min_element
 - Sorting, 193
- min_element_minimal_n
 - [__gnu_parallel::_Settings](#), 720
- minmax
 - Sorting, 194
- minmax_element
 - Sorting, 195
- minstd_rand
 - Random Number Generators, 135
- minstd_rand0
 - Random Number Generators, 135
- mismatch
 - Non-Mutating, 101
- mod_based_range_hashing.hpp, 2167
- modulus
 - [std::linear_congruential_engine](#), 1554
- monetary
 - [std::locale](#), 1582
- money_get
 - [std::money_get](#), 1651
- money_put
 - [std::money_put](#), 1655
- money_punct
 - [std::money_punct](#), 1661
- move
 - Mutating, 76
 - Utilities, 214
- move.h, 2168
- move_backward
 - Mutating, 77
- move_if_noexcept
 - Utilities, 215
- mt19937
 - Random Number Generators, 135
- mt19937_64
 - Random Number Generators, 135
- mt_allocator.h, 2169
- multimap
 - [std::multimap](#), 1682–1684
- multimap.h, 2169, 2170
- multiplier
 - [std::linear_congruential_engine](#), 1554
- multisec_partition
 - [__gnu_parallel](#), 283
- multisec_selection
 - [__gnu_parallel](#), 284
- multisec_selection.h, 2171
- multiset
 - [std::multiset](#), 1701–1703
- multiset.h, 2172, 2173
- multiway_merge
 - [__gnu_parallel](#), 284
- multiway_merge.h, 2174
 - [_GLIBCXX_PARALLEL_LENGTH](#), 2177
- multiway_merge_3_variant
 - [__gnu_parallel](#), 286
- multiway_merge_4_variant
 - [__gnu_parallel](#), 287
- multiway_merge_exact_splitting
 - [__gnu_parallel](#), 287
- multiway_merge_loser_tree
 - [__gnu_parallel](#), 287
- multiway_merge_loser_tree_sentinel
 - [__gnu_parallel](#), 288
- multiway_merge_loser_tree_unguarded
 - [__gnu_parallel](#), 289
- multiway_merge_minimal_k
 - [__gnu_parallel::_Settings](#), 720
- multiway_merge_minimal_n
 - [__gnu_parallel::_Settings](#), 720
- multiway_merge_oversampling
 - [__gnu_parallel::_Settings](#), 720
- multiway_merge_sampling_splitting

- __gnu_parallel, [289](#)
- multiway_merge_sentinels
 - __gnu_parallel, [289](#)
- multiway_mergesort.h, [2177](#)
- Mutating, [70](#)
 - copy, [72](#)
 - copy_backward, [72](#)
 - copy_if, [73](#)
 - copy_n, [73](#)
 - fill, [74](#)
 - fill_n, [74](#)
 - generate, [74](#)
 - generate_n, [75](#)
 - is_partitioned, [75](#)
 - iter_swap, [76](#)
 - move, [76](#)
 - move_backward, [77](#)
 - partition, [77](#)
 - partition_copy, [78](#)
 - partition_point, [78](#)
 - random_shuffle, [79](#)
 - remove, [79](#)
 - remove_copy, [80](#)
 - remove_copy_if, [80](#)
 - remove_if, [81](#)
 - replace, [81](#)
 - replace_copy_if, [82](#)
 - replace_if, [82](#)
 - reverse, [83](#)
 - reverse_copy, [83](#)
 - rotate, [84](#)
 - rotate_copy, [84](#)
 - shuffle, [85](#)
 - stable_partition, [85](#)
 - swap_ranges, [86](#)
 - transform, [86](#), [87](#)
 - unique, [87](#), [88](#)
 - unique_copy, [88](#), [89](#)
- name
 - std::locale, [1579](#)
- narrow
 - std::__ctype_abstract_base, [1051](#), [1052](#)
 - std::basic_ios, [1190](#)
 - std::ctype, [1338](#)
 - std::ctype< char >, [1351](#)
 - std::ctype< wchar_t >, [1367](#)
 - std::ctype_byname, [1382](#), [1383](#)
 - std::ctype_byname< char >, [1396](#)
- neg_format
 - std::moneypunct, [1666](#)
 - std::moneypunct_byname, [1675](#)
- negative_sign
 - std::moneypunct, [1667](#)
- std::moneypunct_byname, [1676](#)
- Negators, [90](#)
 - not1, [91](#)
 - not2, [91](#)
- nested_exception.h, [2177](#)
- new_allocator.h, [2178](#)
- next_permutation
 - Sorting, [195](#), [196](#)
- noboolalpha
 - std, [390](#)
- node.hpp, [2179](#)
- node_begin
 - __gnu_pbds::detail::ov_tree_map, [852](#)
 - __gnu_pbds::detail::pat_trie_map, [883](#)
 - __gnu_pbds::detail::rb_tree_map, [894](#)
 - __gnu_pbds::detail::splay_tree_map, [902](#)
- node_const_iterator
 - __gnu_pbds::detail::bin_search_tree_traits, [782](#)
 - __gnu_pbds::detail::bin_search_tree_traits< Key, null_type, Cmp_Fn, Node_Update, Node, _Alloc >, [783](#)
 - __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >, [913](#)
 - __gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >, [914](#)
 - __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >, [915](#)
 - __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >, [917](#)
 - __gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >, [918](#)
 - __gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >, [920](#)
 - __gnu_pbds::detail::trie_traits< Key, Mapped, _Alloc, ATraits, Node_Update, pat_trie_tag, _Alloc >, [925](#)
 - __gnu_pbds::detail::trie_traits< Key, null_type, _Alloc, ATraits, Node_Update, pat_trie_tag, _Alloc >, [926](#)
- node_end
 - __gnu_pbds::detail::ov_tree_map, [853](#)
 - __gnu_pbds::detail::pat_trie_map, [883](#)
 - __gnu_pbds::detail::rb_tree_map, [894](#)
 - __gnu_pbds::detail::splay_tree_map, [902](#)
- node_iterators.hpp, [2180](#)
- node_metadata_selector.hpp, [2181](#)
- node_type
 - __gnu_pbds::detail::pat_trie_base, [861](#)

- __gnu_pbds::detail::pat_trie_map, 883
- node_update
 - __gnu_pbds::detail::trie_traits< Key, Mapped, _↔
ATraits, Node_Update, pat_trie_tag, _Alloc >, 925
 - __gnu_pbds::detail::trie_traits< Key, null_type, _↔
ATraits, Node_Update, pat_trie_tag, _Alloc >, 926
- Non-Mutating, 92
 - adjacent_find, 93
 - all_of, 94
 - any_of, 94
 - count, 95
 - count_if, 95
 - equal, 95, 96
 - find, 96
 - find_end, 97
 - find_first_of, 98
 - find_if, 99
 - find_if_not, 99
 - for_each, 100
 - is_permutation, 100
 - mismatch, 101
 - none_of, 102
 - search, 102, 103
 - search_n, 103, 104
- none
 - std::locale, 1582
- none_of
 - Non-Mutating, 102
- Normal Distributions, 105
 - operator!=, 106
 - operator<<, 106
 - operator>>, 107
- normal_distribution
 - std::normal_distribution, 1723
- noshowbase
 - std, 390
- noshowpoint
 - std, 391
- noshowpos
 - std, 391
- noskipws
 - std, 391
- nosubs
 - std::regex_constants, 450
- not1
 - Negators, 91
- not2
 - Negators, 91
- notify_cleared
 - __gnu_pbds::cc_hash_max_collision_check_↔
resize_trigger, 755
 - __gnu_pbds::hash_load_check_resize_trigger, 942
 - __gnu_pbds::sample_resize_policy, 970
 - __gnu_pbds::sample_resize_trigger, 973
- notify_erase_search_collision
 - __gnu_pbds::cc_hash_max_collision_check_↔
resize_trigger, 755
 - __gnu_pbds::sample_resize_policy, 970
 - __gnu_pbds::sample_resize_trigger, 973
- notify_erase_search_end
 - __gnu_pbds::cc_hash_max_collision_check_↔
resize_trigger, 755
 - __gnu_pbds::sample_resize_policy, 970
 - __gnu_pbds::sample_resize_trigger, 973
- notify_erase_search_start
 - __gnu_pbds::cc_hash_max_collision_check_↔
resize_trigger, 755
 - __gnu_pbds::sample_resize_policy, 970
 - __gnu_pbds::sample_resize_trigger, 973
- notify_erased
 - __gnu_pbds::cc_hash_max_collision_check_↔
resize_trigger, 756
 - __gnu_pbds::sample_resize_policy, 971
 - __gnu_pbds::sample_resize_trigger, 973
- notify_externally_resized
 - __gnu_pbds::cc_hash_max_collision_check_↔
resize_trigger, 756
 - __gnu_pbds::sample_resize_trigger, 973
- notify_find_search_collision
 - __gnu_pbds::cc_hash_max_collision_check_↔
resize_trigger, 756
 - __gnu_pbds::sample_resize_policy, 971
 - __gnu_pbds::sample_resize_trigger, 974
- notify_find_search_end
 - __gnu_pbds::cc_hash_max_collision_check_↔
resize_trigger, 756
 - __gnu_pbds::sample_resize_policy, 971
 - __gnu_pbds::sample_resize_trigger, 974
- notify_find_search_start
 - __gnu_pbds::cc_hash_max_collision_check_↔
resize_trigger, 756
 - __gnu_pbds::sample_resize_policy, 971
 - __gnu_pbds::sample_resize_trigger, 974
- notify_insert_search_collision
 - __gnu_pbds::cc_hash_max_collision_check_↔
resize_trigger, 756
 - __gnu_pbds::sample_resize_policy, 971
 - __gnu_pbds::sample_resize_trigger, 974
- notify_insert_search_end
 - __gnu_pbds::cc_hash_max_collision_check_↔
resize_trigger, 757
 - __gnu_pbds::sample_resize_policy, 971
 - __gnu_pbds::sample_resize_trigger, 974
- notify_insert_search_start
 - __gnu_pbds::cc_hash_max_collision_check_↔
resize_trigger, 757

- __gnu_pbds::sample_resize_policy, 971
- __gnu_pbds::sample_resize_trigger, 974
- notify_inserted
 - __gnu_pbds::cc_hash_max_collision_check_↔
resize_trigger, 757
 - __gnu_pbds::hash_load_check_resize_trigger, 942
 - __gnu_pbds::sample_resize_policy, 971
 - __gnu_pbds::sample_resize_trigger, 974
- notify_resized
 - __gnu_pbds::cc_hash_max_collision_check_↔
resize_trigger, 757
 - __gnu_pbds::hash_load_check_resize_trigger, 943
 - __gnu_pbds::sample_range_hashing, 967
 - __gnu_pbds::sample_ranged_hash_fn, 968
 - __gnu_pbds::sample_resize_policy, 971
 - __gnu_pbds::sample_resize_trigger, 974
- nounitbuf
 - std, 391
- nouppercase
 - std, 391
- npos
 - __gnu_cxx::_versa_string, 536
 - std::basic_string, 1263
- nth_element
 - Sorting, 196, 197
- nth_element_minimal_n
 - __gnu_parallel::Settings, 720
- null_node_metadata.hpp, 2181
- num_children
 - __gnu_pbds::detail::pat_trie_base::_Node_citer, 876
 - __gnu_pbds::detail::pat_trie_base::_Node_iter, 879
- num_get
 - std::num_get, 1731
- num_put
 - std::num_put, 1746
- numeric
 - std::locale, 1582
- Numeric_arrays, 108
 - ~gslice, 113
 - gslice, 111, 112
 - gslice_array, 112
 - indirect_array, 112
 - mask_array, 112
 - operator<=, 116, 117
 - operator>=, 119
 - operator*=, 114
 - operator^=, 119, 120
 - operator+=, 114, 115
 - operator-=, 115
 - operator/=, 116
 - operator=, 117, 118
 - operator%=, 113
 - operator&=, 113, 114
 - operator|=, 120
 - size, 120
 - slice, 112
 - slice_array, 112
 - start, 121
 - stride, 121
- numeric_traits.h, 2182
- numeric_fwd.h, 2182
- numpunct
 - std::numpunct, 1759
- oct
 - std, 391
 - std::basic_ios, 1200
 - std::ios_base, 1536
- off_type
 - std::basic_ios, 1182
- omp_loop.h, 2184
- omp_loop_static.h, 2185
- openmode
 - std::basic_ios, 1182
 - std::ios_base, 1526
- operator_iterator
 - __gnu_debug::_Safe_iterator, 586
 - __gnu_debug::_Safe_local_iterator, 600
- operator_RAlter
 - __gnu_parallel::_GuardedIterator, 669
- operator bool
 - std::unique_ptr, 1917
 - std::unique_ptr<_Tp[], _Dp>, 1922
- operator const_point_iterator_
 - iterator_, 1026
- operator point_iterator_
 - iterator_, 1026
- operator streamoff
 - std::fpos, 1473
- operator string_type
 - std::sub_match, 1877
- operator void *
 - std::basic_ios, 1190
- operator!
 - std::basic_ios, 1190
- operator!=
 - __gnu_cxx, 226, 227
 - __gnu_pbds::detail::bin_search_tree_const_node_↔
it_, 776
 - __gnu_pbds::detail::bin_search_tree_node_it_, 781
 - __gnu_pbds::detail::binary_heap_const_iterator_,
789
 - __gnu_pbds::detail::binary_heap_point_const_↔
iterator_, 793
 - __gnu_pbds::detail::left_child_next_sibling_heap_↔
const_iterator_, 838
 - __gnu_pbds::detail::left_child_next_sibling_heap_↔
node_point_const_iterator_, 843

- `__gnu_pbds::detail::pat_trie_base::_Node_citer`, 876
 - `__gnu_pbds::detail::pat_trie_base::_Node_iter`, 879
 - Bernoulli Distributions, 26
 - `const_iterator_`, 1022
 - `iterator_`, 1026, 1027
 - Iterators, 66
 - Normal Distributions, 106
 - `point_const_iterator_`, 1031
 - `point_iterator_`, 1034
 - Poisson Distributions, 127, 128
 - Random Number Generators, 136–138
 - Regular Expressions, 146, 148, 149
 - `std`, 392–394
 - `std::locale`, 1579
 - `std::regex_iterator`, 1815
 - `std::regex_token_iterator`, 1819
 - `std::rel_ops`, 456
 - Uniform Distributions, 207
 - Utilities, 215
- `operator<`
 - `__gnu_cxx`, 229, 230
 - `__gnu_parallel::_GuardedIterator`, 669
 - Regular Expressions, 150–152
 - `std`, 396–402
 - Utilities, 215
- `operator<<`
 - Bernoulli Distributions, 26
 - Normal Distributions, 106
 - Pointer abstractions, 125
 - Poisson Distributions, 128, 129
 - Random Number Generators, 138
 - Regular Expressions, 152
 - `std`, 402
 - `std::binomial_distribution`, 1276
 - `std::chi_squared_distribution`, 1288
 - `std::discard_block_engine`, 1431
 - `std::discrete_distribution`, 1435
 - `std::fisher_f_distribution`, 1450
 - `std::gamma_distribution`, 1480
 - `std::linear_congruential_engine`, 1553
 - `std::lognormal_distribution`, 1593
 - `std::mersenne_twister_engine`, 1637
 - `std::negative_binomial_distribution`, 1719
 - `std::normal_distribution`, 1725
 - `std::piecewise_constant_distribution`, 1784
 - `std::piecewise_linear_distribution`, 1789
 - `std::poisson_distribution`, 1800
 - `std::shuffle_order_engine`, 1864
 - `std::student_t_distribution`, 1872
 - Uniform Distributions, 208
- `operator<=<=`
 - Numeric arrays, 116, 117
- `operator<=`
 - `__gnu_cxx`, 230, 231
 - `__gnu_parallel::_GuardedIterator`, 670
 - Regular Expressions, 153–155
 - `std`, 403–405
 - `std::rel_ops`, 456
 - Utilities, 215
- `operator>`
 - `__gnu_cxx`, 233
 - Regular Expressions, 158–160
 - `std`, 412–414
 - `std::rel_ops`, 456
 - Utilities, 216
- `operator>>`
 - Bernoulli Distributions, 27
 - Normal Distributions, 107
 - Poisson Distributions, 129, 130
 - `std`, 417
 - `std::binomial_distribution`, 1276
 - `std::chi_squared_distribution`, 1288
 - `std::discard_block_engine`, 1431
 - `std::discrete_distribution`, 1435
 - `std::fisher_f_distribution`, 1450
 - `std::gamma_distribution`, 1480
 - `std::independent_bits_engine`, 1515
 - `std::linear_congruential_engine`, 1553
 - `std::lognormal_distribution`, 1593
 - `std::mersenne_twister_engine`, 1638
 - `std::negative_binomial_distribution`, 1720
 - `std::normal_distribution`, 1726
 - `std::piecewise_constant_distribution`, 1784
 - `std::piecewise_linear_distribution`, 1789
 - `std::poisson_distribution`, 1800
 - `std::shuffle_order_engine`, 1864
 - `std::student_t_distribution`, 1873
 - Uniform Distributions, 208, 209
- `operator>>=`
 - Numeric arrays, 119
- `operator>=`
 - `__gnu_cxx`, 234, 235
 - Regular Expressions, 161–163
 - `std`, 414–416
 - `std::rel_ops`, 457
 - Utilities, 216
- `operator*`
 - `__gnu_debug::_Safe_iterator`, 587
 - `__gnu_debug::_Safe_local_iterator`, 600
 - `__gnu_parallel::_GuardedIterator`, 669
 - `__gnu_pbds::detail::bin_search_tree_const_node_↵
it_`, 776
 - `__gnu_pbds::detail::bin_search_tree_node_it_`, 781
 - `__gnu_pbds::detail::binary_heap_const_iterator_`, 789
 - `__gnu_pbds::detail::binary_heap_point_const_↵
iterator_`, 793

- __gnu_pbds::detail::left_child_next_sibling_heap_↵
const_iterator_, 838
- __gnu_pbds::detail::left_child_next_sibling_heap_↵
node_point_const_iterator_, 843
- __gnu_pbds::detail::ov_tree_node_it_, 857
- __gnu_pbds::detail::pat_trie_base::_Node_citer, 877
- __gnu_pbds::detail::pat_trie_base::_Node_iter, 880
- const_iterator_, 1022
- iterator_, 1027
- point_const_iterator_, 1031
- point_iterator_, 1034
- std::auto_ptr, 1170
- std::back_insert_iterator, 1175
- std::front_insert_iterator, 1476
- std::insert_iterator, 1521
- std::istreambuf_iterator, 1544
- std::ostreambuf_iterator, 1775
- std::regex_iterator, 1815
- std::regex_token_iterator, 1819
- std::reverse_iterator, 1829
- std::unique_ptr, 1917
- operator*=
Numeric_arrays, 114
- operator^
std::regex_constants, 453
- operator^=
Numeric_arrays, 119, 120
- std::regex_constants, 453, 454
- operator()
__gnu_parallel::_Nothing, 701
- __gnu_parallel::_RandomNumber, 713
- __gnu_parallel::_accumulate_selector, 624
- __gnu_parallel::_adjacent_find_selector, 626
- __gnu_parallel::_count_if_selector, 630
- __gnu_parallel::_count_selector, 632
- __gnu_parallel::_fill_selector, 633
- __gnu_parallel::_find_first_of_selector, 635
- __gnu_parallel::_find_if_selector, 637
- __gnu_parallel::_for_each_selector, 638
- __gnu_parallel::_generate_selector, 639
- __gnu_parallel::_identity_selector, 644
- __gnu_parallel::_inner_product_selector, 646
- __gnu_parallel::_mismatch_selector, 649
- __gnu_parallel::_replace_if_selector, 654
- __gnu_parallel::_replace_selector, 656
- __gnu_parallel::_transform1_selector, 657
- __gnu_parallel::_transform2_selector, 659
- __gnu_pbds::direct_mask_range_hashing, 932
- __gnu_pbds::direct_mod_range_hashing, 934
- __gnu_pbds::linear_probe_fn, 949
- __gnu_pbds::lu_counter_policy, 952
- __gnu_pbds::lu_move_to_front_policy, 954
- __gnu_pbds::quadratic_probe_fn, 962
- __gnu_pbds::sample_probe_fn, 966
- __gnu_pbds::sample_range_hashing, 967
- __gnu_pbds::sample_ranged_hash_fn, 968
- __gnu_pbds::sample_trie_node_update, 978
- __gnu_pbds::sample_update_policy, 979
- __gnu_pbds::tree_order_statistics_node_update,
987
- __gnu_pbds::trie_order_statistics_node_update, 993
- __gnu_pbds::trie_prefix_search_node_update, 996
- std::bernoulli_distribution, 1265
- std::binomial_distribution, 1275
- std::cauchy_distribution, 1279
- std::chi_squared_distribution, 1287
- std::default_delete, 1402
- std::default_delete< _Tp[]>, 1403
- std::discard_block_engine, 1430
- std::discrete_distribution, 1434
- std::exponential_distribution, 1442
- std::extreme_value_distribution, 1446
- std::fisher_f_distribution, 1449
- std::gamma_distribution, 1479
- std::geometric_distribution, 1483
- std::independent_bits_engine, 1514
- std::linear_congruential_engine, 1552
- std::locale, 1580
- std::lognormal_distribution, 1592
- std::negative_binomial_distribution, 1719
- std::normal_distribution, 1724
- std::piecewise_constant_distribution, 1783
- std::piecewise_linear_distribution, 1788
- std::poisson_distribution, 1799
- std::shuffle_order_engine, 1863
- std::student_t_distribution, 1872
- std::uniform_int_distribution, 1908
- std::uniform_real_distribution, 1912
- std::weibull_distribution, 2041
- operator+
__gnu_cxx, 227, 228
- std, 394–396
- std::fpos, 1473
- std::reverse_iterator, 1829
- operator++
__gnu_debug::_Safe_iterator, 587
- __gnu_debug::_Safe_local_iterator, 601
- __gnu_parallel::_GuardedIterator, 669
- const_iterator_, 1023
- iterator_, 1027
- std::back_insert_iterator, 1175
- std::front_insert_iterator, 1476
- std::insert_iterator, 1521
- std::istreambuf_iterator, 1544
- std::ostreambuf_iterator, 1775
- std::regex_iterator, 1815
- std::regex_token_iterator, 1819
- std::reverse_iterator, 1829

- operator+=
 - `__gnu_cxx::__versa_string`, 518, 519
 - `Numeric_arrays`, 114, 115
 - `std::basic_string`, 1248, 1249
 - `std::fpos`, 1473
 - `std::reverse_iterator`, 1830
- operator-
 - `std::fpos`, 1473
 - `std::reverse_iterator`, 1830
- operator->
 - `__gnu_debug::__Safe_iterator`, 588
 - `__gnu_debug::__Safe_local_iterator`, 601
 - `__gnu_pbds::detail::binary_heap_const_iterator_`, 789
 - `__gnu_pbds::detail::binary_heap_point_const_iterator_`, 793
 - `__gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_`, 839
 - `__gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_`, 843
 - `const_iterator_`, 1023
 - `iterator_`, 1027
 - `point_const_iterator_`, 1031
 - `point_iterator_`, 1034
 - `std::auto_ptr`, 1170
 - `std::regex_iterator`, 1815
 - `std::regex_token_iterator`, 1819
 - `std::reverse_iterator`, 1831
 - `std::unique_ptr`, 1917
- operator--
 - `__gnu_debug::__Safe_iterator`, 587
 - `std::reverse_iterator`, 1830
- operator-=
 - `Numeric_arrays`, 115
 - `std::fpos`, 1473
 - `std::reverse_iterator`, 1831
- operator/=
 - `Numeric_arrays`, 116
- operator=
 - `__gnu_cxx::__versa_string`, 519–521
 - `__gnu_debug::__Safe_iterator`, 588
 - `__gnu_debug::__Safe_local_iterator`, 601
 - `Numeric_arrays`, 117, 118
 - `std::auto_ptr`, 1170, 1171
 - `std::back_insert_iterator`, 1175
 - `std::basic_regex`, 1210–1212
 - `std::basic_string`, 1249, 1250
 - `std::deque`, 1422, 1423
 - `std::forward_list`, 1466
 - `std::front_insert_iterator`, 1476
 - `std::insert_iterator`, 1521
 - `std::list`, 1568, 1569
 - `std::locale`, 1580
 - `std::map`, 1611, 1612
 - `std::match_results`, 1624, 1625
 - `std::multimap`, 1694
 - `std::multiset`, 1712, 1713
 - `std::ostream_iterator`, 1772
 - `std::ostreambuf_iterator`, 1775
 - `std::regex_iterator`, 1815
 - `std::regex_token_iterator`, 1820
 - `std::set`, 1849, 1850
 - `std::unique_ptr`, 1917, 1918
 - `std::unique_ptr< _Tp[], _Dp >`, 1922, 1923
 - `std::unordered_map`, 1945
 - `std::unordered_multimap`, 1968
 - `std::unordered_multiset`, 1991
 - `std::unordered_set`, 2013
 - `std::vector`, 2030
- operator==
 - `__gnu_cxx`, 231, 232
 - `__gnu_pbds::detail::bin_search_tree_const_node_iterator_`, 776
 - `__gnu_pbds::detail::bin_search_tree_node_iterator_`, 781
 - `__gnu_pbds::detail::binary_heap_const_iterator_`, 789
 - `__gnu_pbds::detail::binary_heap_point_const_iterator_`, 793
 - `__gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_`, 839
 - `__gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_`, 843
 - `__gnu_pbds::detail::pat_trie_base::_Node_citer`, 877
 - `__gnu_pbds::detail::pat_trie_base::_Node_iter`, 880
 - `const_iterator_`, 1023
 - `iterator_`, 1027
 - `Iterators`, 66
 - `point_const_iterator_`, 1031
 - `point_iterator_`, 1034
 - `Regular Expressions`, 155–158
 - `std`, 405–411
 - `std::bernoulli_distribution`, 1266
 - `std::binomial_distribution`, 1276
 - `std::cauchy_distribution`, 1280
 - `std::chi_squared_distribution`, 1288
 - `std::discard_block_engine`, 1431
 - `std::discrete_distribution`, 1435
 - `std::exponential_distribution`, 1443
 - `std::extreme_value_distribution`, 1447
 - `std::fisher_f_distribution`, 1450
 - `std::gamma_distribution`, 1480
 - `std::geometric_distribution`, 1484
 - `std::independent_bits_engine`, 1515
 - `std::linear_congruential_engine`, 1553
 - `std::locale`, 1581
 - `std::lognormal_distribution`, 1593
 - `std::mersenne_twister_engine`, 1637
 - `std::negative_binomial_distribution`, 1720

- std::normal_distribution, 1726
- std::piecewise_constant_distribution, 1784
- std::piecewise_linear_distribution, 1789
- std::poisson_distribution, 1800
- std::regex_iterator, 1815
- std::regex_token_iterator, 1820
- std::shuffle_order_engine, 1864
- std::student_t_distribution, 1873
- std::uniform_int_distribution, 1909
- std::uniform_real_distribution, 1913
- std::weibull_distribution, 2042
- Utilities, 215
- operator%=
 - Numeric_arrays, 113
- operator&
 - std::regex_constants, 452
- operator&=
 - Numeric_arrays, 113, 114
 - std::regex_constants, 453
- operator[]
 - __gnu_cxx::__versa_string, 521, 522
 - std::basic_string, 1251
 - std::deque, 1423, 1424
 - std::map, 1612
 - std::match_results, 1625
 - std::reverse_iterator, 1831
 - std::unique_ptr<_Tp[], _Dp>, 1923
 - std::unordered_map, 1945, 1946
 - std::vector, 2031
- operator|
 - std::regex_constants, 454
- operator|=
 - Numeric_arrays, 120
 - std::regex_constants, 454, 455
- operator~
 - std::regex_constants, 455
- opt_random.h, 2185
- optimize
 - std::regex_constants, 450
- order_of_key
 - __gnu_pbds::tree_order_statistics_node_update, 988
 - __gnu_pbds::trie_order_statistics_node_update, 993
- order_of_prefix
 - __gnu_pbds::trie_order_statistics_node_update, 993
- order_preserving
 - __gnu_pbds::container_traits, 765
- order_statistics_imp.hpp, 2185
- os_defines.h, 2186
- ostream_insert.h, 2186
- ostream_iterator
 - std::ostream_iterator, 1771, 1772
- ostream_type
 - std::ostream_iterator, 1770
- std::ostreambuf_iterator, 1774
- ostreambuf_iterator
 - std::ostreambuf_iterator, 1775
- out
 - std::__codecvt_abstract_base, 1040
 - std::basic_ios, 1200
 - std::codecvt, 1293
 - std::codecvt<_InternT, _ExternT, encoding_state>, 1297
 - std::codecvt<char, char, mbstate_t>, 1302
 - std::codecvt<wchar_t, char, mbstate_t>, 1306
 - std::codecvt_byname, 1312
 - std::ios_base, 1536
- ov_tree_map.hpp, 2186
- p
 - std::bernoulli_distribution, 1265
 - std::binomial_distribution, 1275
 - std::geometric_distribution, 1483
 - std::negative_binomial_distribution, 1719
- pair
 - std::pair, 1780
- pairing_heap.hpp, 2187
- par_loop.h, 2187
- parallel.h, 2188
- parallel_balanced
 - __gnu_parallel, 254
- parallel_multiway_merge
 - __gnu_parallel, 291
- parallel_omp_loop
 - __gnu_parallel, 254
- parallel_omp_loop_static
 - __gnu_parallel, 254
- parallel_sort_mwms
 - __gnu_parallel, 292
- parallel_sort_mwms_pu
 - __gnu_parallel, 292
- parallel_tag
 - __gnu_parallel::parallel_tag, 741
- parallel_taskqueue
 - __gnu_parallel, 254
- parallel_unbalanced
 - __gnu_parallel, 254
- param
 - std::bernoulli_distribution, 1266
 - std::binomial_distribution, 1275
 - std::cauchy_distribution, 1279
 - std::chi_squared_distribution, 1287
 - std::discrete_distribution, 1434
 - std::exponential_distribution, 1442
 - std::extreme_value_distribution, 1446
 - std::fisher_f_distribution, 1449
 - std::gamma_distribution, 1479
 - std::geometric_distribution, 1483

- std::lognormal_distribution, 1592
- std::negative_binomial_distribution, 1719
- std::normal_distribution, 1724
- std::piecewise_constant_distribution, 1783
- std::piecewise_linear_distribution, 1788
- std::poisson_distribution, 1799
- std::student_t_distribution, 1872
- std::uniform_int_distribution, 1908
- std::uniform_real_distribution, 1912
- std::weibull_distribution, 2041
- parse_numbers.h, 2188
- partial_sort
 - Sorting, 197, 198
- partial_sort_copy
 - Sorting, 198, 199
- partial_sort_minimal_n
 - __gnu_parallel::Settings, 720
- partial_sum
 - std, 417, 418
- partial_sum.h, 2188
- partial_sum_dilation
 - __gnu_parallel::Settings, 720
- partial_sum_minimal_n
 - __gnu_parallel::Settings, 721
- partition
 - Mutating, 77
- partition.h, 2189
 - _GLIBCXX_VOLATILE, 2189
- partition_chunk_share
 - __gnu_parallel::Settings, 721
- partition_chunk_size
 - __gnu_parallel::Settings, 721
- partition_copy
 - Mutating, 78
- partition_minimal_n
 - __gnu_parallel::Settings, 721
- partition_point
 - Mutating, 78
- pat_trie.hpp, 2189
- pat_trie_base.hpp, 2190
- piecewise_construct
 - Utilities, 217
- pod_char_traits.h, 2191
- point_const_iterator.hpp, 2191, 2192
- point_const_iterator_, 1028
 - const_pointer, 1029
 - const_reference, 1029
 - difference_type, 1030
 - iterator_category, 1030
 - operator!=, 1031
 - operator*, 1031
 - operator->, 1031
 - operator==, 1031
 - point_const_iterator_, 1030
 - pointer, 1030
 - reference, 1030
 - value_type, 1030
- point_iterator.hpp, 2192
- point_iterator_, 1032
 - const_pointer, 1032
 - const_reference, 1032
 - difference_type, 1033
 - iterator_category, 1033
 - operator!=, 1034
 - operator*, 1034
 - operator->, 1034
 - operator==, 1034
 - point_iterator_, 1033
 - pointer, 1033
 - reference, 1033
 - value_type, 1033
- point_iterators.hpp, 2193
- pointer
 - __gnu_pbds::detail::binary_heap_const_iterator_, 788
 - __gnu_pbds::detail::binary_heap_point_const_iterator_, 792
 - __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_, 837
 - __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_, 842
 - const_iterator_, 1021
 - iterator_, 1025
 - point_const_iterator_, 1030
 - point_iterator_, 1033
 - std::allocator_traits, 1157
 - std::allocator_traits< allocator< _Tp > >, 1162
 - std::back_insert_iterator, 1174
 - std::front_insert_iterator, 1475
 - std::insert_iterator, 1520
 - std::istream_iterator, 1539
 - std::istreambuf_iterator, 1543
 - std::iterator, 1545
 - std::ostream_iterator, 1771
 - std::ostreambuf_iterator, 1774
 - std::pointer_traits, 1796
 - std::pointer_traits< _Tp * >, 1797
 - std::raw_storage_iterator, 1811
 - std::set, 1836
 - std::unordered_map, 1929
 - std::unordered_multimap, 1952
 - std::unordered_multiset, 1974
 - std::unordered_set, 1997
- pointer.h, 2193
- Pointer_abstractions, 122
 - allocate_shared, 124
 - get_deleter, 124
 - make_shared, 125

- operator<<, 125
- pointer_to
 - std::pointer_traits< _Tp * >, 1797
- Poisson Distributions, 126
 - operator!=, 127, 128
 - operator<<, 128, 129
 - operator>>, 129, 130
- Policy-Based Data Structures, 131
- policy_access_fnimps.hpp, 2195, 2196
- pool_allocator.h, 2196
- pop
 - std::priority_queue, 1804
 - std::queue, 1807
 - std::stack, 1869
- pop_back
 - __gnu_cxx::__versa_string, 522
 - __gnu_parallel::RestrictedBoundedConcurrent←
Queue, 714
 - std::basic_string, 1252
 - std::deque, 1424
 - std::list, 1569
 - std::vector, 2032
- pop_front
 - __gnu_parallel::RestrictedBoundedConcurrent←
Queue, 714
 - std::deque, 1424
 - std::forward_list, 1467
 - std::list, 1569
- pop_heap
 - Heap, 55
- pos_format
 - std::moneypunct, 1667
 - std::moneypunct_byname, 1676
- pos_type
 - std::basic_ios, 1182
- position
 - std::match_results, 1625
- positive_sign
 - std::moneypunct, 1668
 - std::moneypunct_byname, 1677
- postypes.h, 2197
- precision
 - std::basic_ios, 1191
 - std::ios_base, 1529
- predefined_ops.h, 2197
- prefix
 - std::match_results, 1626
- prefix_range
 - __gnu_pbds::trie_prefix_search_node_update, 996,
997
- prefix_search_node_update_imp.hpp, 2198
- prev_permutation
 - Sorting, 199, 200
- priority_queue
 - Heap-Based, 59
 - std::priority_queue, 1803
- priority_queue.hpp, 2199
- priority_queue_base_dispatch.hpp, 2199
- probabilities
 - std::discrete_distribution, 1434
- probe_fn_base.hpp, 2199
- profiler.h, 2200
- profiler_algos.h, 2202
- profiler_container_size.h, 2203
- profiler_hash_func.h, 2203
- profiler_hashtable_size.h, 2204
- profiler_list_to_slist.h, 2204
- profiler_list_to_vector.h, 2205
- profiler_map_to_unordered_map.h, 2205
- profiler_node.h, 2206
- profiler_state.h, 2207
- profiler_trace.h, 2207
- profiler_vector_size.h, 2209
- profiler_vector_to_list.h, 2210
- propagate_on_container_copy_assignment
 - __gnu_cxx::__alloc_traits, 460
 - std::allocator_traits, 1157
 - std::allocator_traits< allocator< _Tp > >, 1163
- propagate_on_container_move_assignment
 - __gnu_cxx::__alloc_traits, 460
 - std::allocator_traits, 1158
 - std::allocator_traits< allocator< _Tp > >, 1163
- propagate_on_container_swap
 - __gnu_cxx::__alloc_traits, 461
 - std::allocator_traits, 1158
 - std::allocator_traits< allocator< _Tp > >, 1163
- ptr_fun
 - Adaptors for pointers to functions, 2
- ptr_traits.h, 2210
- push
 - std::priority_queue, 1804
 - std::queue, 1807
 - std::stack, 1869
- push_back
 - __gnu_cxx::__versa_string, 522
 - std::basic_string, 1252
 - std::deque, 1424
 - std::list, 1569
 - std::vector, 2032
- push_front
 - __gnu_parallel::RestrictedBoundedConcurrent←
Queue, 714
 - std::deque, 1425
 - std::forward_list, 1467
 - std::list, 1570
- push_heap
 - Heap, 55, 56
- put

- std::money_put, 1657
- std::num_put, 1751–1756
- std::time_put, 1898, 1900
- std::time_put_byname, 1902, 1903
- pwd
 - std::basic_ios, 1191
 - std::ios_base, 1530
- qsb_steals
 - __gnu_parallel::Settings, 721
- quadratic_probe_fn_imp.hpp, 2211
- queue
 - std::queue, 1806
- queue.h, 2211
 - _GLIBCXX_VOLATILE, 2211
- quicksort.h, 2212
- r_erase_fn_imps.hpp, 2212
- Random Number Distributions, 132
- Random Number Generation, 133
 - generate_canonical, 133
- Random Number Generators, 134
 - minstd_rand, 135
 - minstd_rand0, 135
 - mt19937, 135
 - mt19937_64, 135
 - operator!=, 136–138
 - operator<<, 138
- Random Number Utilities, 139
- random.h, 2213
- random_number.h, 2217
- random_shuffle
 - Mutating, 79
- random_shuffle.h, 2217
- random_shuffle_minimal_n
 - __gnu_parallel::Settings, 721
- range_access.h, 2218
- ranged_hash_fn.hpp, 2219
- ranged_probe_fn.hpp, 2219
- rb_tree.hpp, 2220
- rbegin
 - __gnu_cxx::__versa_string, 523
 - std::basic_string, 1252
 - std::deque, 1425
 - std::list, 1570
 - std::map, 1613
 - std::multimap, 1695
 - std::multiset, 1713
 - std::set, 1850
 - std::vector, 2032
- rc.hpp, 2220
- rc_binomial_heap.hpp, 2221
- rc_string_base.h, 2221
- rdbuf
 - std::basic_ios, 1192
- rdstate
 - std::basic_ios, 1192
- ready
 - std::match_results, 1626
- reference
 - __gnu_pbds::detail::bin_search_tree_const_node_↔
it_, 775
 - __gnu_pbds::detail::bin_search_tree_node_it_, 780
 - __gnu_pbds::detail::binary_heap_const_iterator_,
788
 - __gnu_pbds::detail::binary_heap_point_const_↔
iterator_, 792
 - __gnu_pbds::detail::left_child_next_sibling_heap_↔
const_iterator_, 837
 - __gnu_pbds::detail::left_child_next_sibling_heap_↔
node_point_const_iterator_, 842
 - const_iterator_, 1022
 - iterator_, 1026
 - point_const_iterator_, 1030
 - point_iterator_, 1033
 - std::back_insert_iterator, 1174
 - std::front_insert_iterator, 1476
 - std::insert_iterator, 1520
 - std::istream_iterator, 1540
 - std::istreambuf_iterator, 1543
 - std::iterator, 1545
 - std::ostream_iterator, 1771
 - std::ostreambuf_iterator, 1774
 - std::raw_storage_iterator, 1812
 - std::set, 1836
 - std::unordered_map, 1929
 - std::unordered_multimap, 1952
 - std::unordered_multiset, 1975
 - std::unordered_set, 1998
- regex
 - Regular Expressions, 145
- regex.h, 2222
- regex_automaton.h, 2227
- regex_compiler.h, 2227
- regex_constants.h, 2228
- regex_error
 - std::regex_error, 1812
- regex_error.h, 2229
- regex_executor.h, 2230
- regex_iterator
 - std::regex_iterator, 1814
- regex_match
 - Regular Expressions, 163–166
- regex_replace
 - Regular Expressions, 167–170
- regex_scanner.h, 2231
- regex_search
 - Regular Expressions, 170–174
- regex_token_iterator

- std::regex_token_iterator, 1817–1819
- regex_traits
 - std::regex_traits, 1821
- register_callback
 - std::basic_ios, 1193
 - std::ios_base, 1530
- Regular Expressions, 140
 - cregex_token_iterator, 145
 - csub_match, 145
 - operator!=, 146, 148, 149
 - operator<, 150–152
 - operator<<, 152
 - operator<=, 153–155
 - operator>, 158–160
 - operator>=, 161–163
 - operator==, 155–158
 - regex, 145
 - regex_match, 163–166
 - regex_replace, 167–170
 - regex_search, 170–174
 - sregex_token_iterator, 145
 - ssub_match, 145
 - swap, 174
 - wcregex_token_iterator, 145
 - wcsub_match, 145
 - wregex, 146
 - wsregex_token_iterator, 146
 - wssub_match, 146
- rehash
 - std::unordered_map, 1946
 - std::unordered_multimap, 1968
 - std::unordered_multiset, 1991
 - std::unordered_set, 2013
- release
 - std::auto_ptr, 1171
 - std::unique_ptr, 1918
 - std::unique_ptr< _Tp[], _Dp >, 1923
- remove
 - Mutating, 79
 - std::forward_list, 1467
 - std::list, 1570
- remove_copy
 - Mutating, 80
- remove_copy_if
 - Mutating, 80
- remove_if
 - Mutating, 81
 - std::forward_list, 1468
 - std::list, 1571
- rend
 - __gnu_cxx::__versa_string, 523
 - std::basic_string, 1252
 - std::deque, 1425
 - std::list, 1571
- std::map, 1613
- std::multimap, 1695
- std::multiset, 1714
- std::set, 1850
- std::vector, 2032, 2033
- replace
 - __gnu_cxx::__versa_string, 523–528, 530, 531
 - Mutating, 81
 - std::basic_string, 1253–1258
- replace_copy
 - std, 418
- replace_copy_if
 - Mutating, 82
- replace_if
 - Mutating, 82
- replace_minimal_n
 - __gnu_parallel::Settings, 721
- requested_size
 - std::_Temporary_buffer, 1152
- reserve
 - __gnu_cxx::__versa_string, 531
 - std::basic_string, 1259
 - std::unordered_map, 1947
 - std::unordered_multimap, 1969
 - std::unordered_multiset, 1992
 - std::unordered_set, 2014
 - std::vector, 2033
- reset
 - std::auto_ptr, 1171
 - std::bernoulli_distribution, 1266
 - std::binomial_distribution, 1276
 - std::cauchy_distribution, 1280
 - std::chi_squared_distribution, 1288
 - std::discrete_distribution, 1434
 - std::exponential_distribution, 1443
 - std::extreme_value_distribution, 1446
 - std::fisher_f_distribution, 1450
 - std::gamma_distribution, 1480
 - std::geometric_distribution, 1484
 - std::lognormal_distribution, 1593
 - std::negative_binomial_distribution, 1719
 - std::normal_distribution, 1724
 - std::piecewise_constant_distribution, 1784
 - std::piecewise_linear_distribution, 1788
 - std::poisson_distribution, 1800
 - std::student_t_distribution, 1872
 - std::uniform_int_distribution, 1909
 - std::uniform_real_distribution, 1912
 - std::unique_ptr, 1918
 - std::unique_ptr< _Tp[], _Dp >, 1923
 - std::weibull_distribution, 2042
- resize
 - __gnu_cxx::__versa_string, 532
 - __gnu_pbds::hash_standard_resize_policy, 947

- std::basic_string, 1260
- std::deque, 1426
- std::forward_list, 1468
- std::list, 1571
- std::vector, 2033, 2034
- resize_fn_imps.hpp, 2231
- resize_no_store_hash_fn_imps.hpp, 2231
- resize_policy.hpp, 2232
- resize_store_hash_fn_imps.hpp, 2232
- result_type
 - __gnu_cxx::__detail::_Ffit_finder, 466
 - __gnu_parallel::_EqualFromLess, 666
 - __gnu_parallel::_EqualTo, 667
 - __gnu_parallel::_Less, 675
 - __gnu_parallel::_Lexicographic, 676
 - __gnu_parallel::_LexicographicReverse, 678
 - __gnu_parallel::_Multiplies, 700
 - __gnu_parallel::_Plus, 703
 - __gnu_parallel::_binder1st, 628
 - __gnu_parallel::_binder2nd, 629
 - __gnu_parallel::_unary_negate, 661
- std::bernoulli_distribution, 1265
- std::binary_function, 1269
- std::binary_negate, 1270
- std::binder1st, 1272
- std::binder2nd, 1273
- std::binomial_distribution, 1275
- std::cauchy_distribution, 1279
- std::chi_squared_distribution, 1287
- std::const_mem_fun1_ref_t, 1325
- std::const_mem_fun1_t, 1326
- std::const_mem_fun_ref_t, 1327
- std::const_mem_fun_t, 1329
- std::discard_block_engine, 1428
- std::discrete_distribution, 1433
- std::divides, 1437
- std::equal_to, 1439
- std::exponential_distribution, 1441
- std::extreme_value_distribution, 1445
- std::fisher_f_distribution, 1449
- std::gamma_distribution, 1478
- std::geometric_distribution, 1483
- std::greater, 1486
- std::greater_equal, 1487
- std::hash< __gnu_cxx::throw_value_limit >, 1493
- std::hash< __gnu_cxx::throw_value_random >, 1494
- std::independent_bits_engine, 1512
- std::less, 1548
- std::less_equal, 1549
- std::linear_congruential_engine, 1551
- std::logical_and, 1587
- std::logical_not, 1589
- std::logical_or, 1590
- std::lognormal_distribution, 1592
- std::mem_fun1_ref_t, 1629
- std::mem_fun1_t, 1631
- std::mem_fun_ref_t, 1632
- std::mem_fun_t, 1633
- std::mersenne_twister_engine, 1636
- std::minus, 1646
- std::modulus, 1647
- std::multiplies, 1698
- std::negate, 1716
- std::negative_binomial_distribution, 1718
- std::normal_distribution, 1723
- std::not_equal_to, 1728
- std::owner_less< shared_ptr< _Tp > >, 1777
- std::owner_less< weak_ptr< _Tp > >, 1778
- std::piecewise_constant_distribution, 1783
- std::piecewise_linear_distribution, 1787
- std::plus, 1791
- std::pointer_to_binary_function, 1793
- std::pointer_to_unary_function, 1795
- std::poisson_distribution, 1799
- std::random_device, 1810
- std::seed_seq, 1832
- std::shuffle_order_engine, 1861
- std::student_t_distribution, 1871
- std::unary_function, 1905
- std::unary_negate, 1906
- std::uniform_int_distribution, 1907
- std::uniform_real_distribution, 1911
- std::weibull_distribution, 2040
- rethrow_exception
 - Exceptions, 46
- rethrow_if_nested
 - Exceptions, 46
- return_temporary_buffer
 - std, 419
- reverse
 - Mutating, 83
 - std::forward_list, 1468
 - std::list, 1572
- reverse_copy
 - Mutating, 83
- reverse_iteration
 - __gnu_pbds::container_traits, 765
- reverse_iterator
 - std::reverse_iterator, 1828
 - std::set, 1836
- rfind
 - __gnu_cxx::__versa_string, 533, 534
 - std::basic_string, 1260, 1261
- right
 - std, 419
 - std::basic_ios, 1201
 - std::ios_base, 1536

- ropeimpl.h, [2232](#)
- rotate
 - Mutating, [84](#)
- rotate_copy
 - Mutating, [84](#)
- rotate_fn_imps.hpp, [2233](#)
- SGL, [176](#)
- safe_base.h, [2233](#)
- safe_iterator.h, [2233](#)
- safe_local_iterator.h, [2235](#)
- safe_sequence.h, [2236](#)
- safe_unordered_base.h, [2236](#)
- safe_unordered_container.h, [2236](#)
- sample_probe_fn
 - __gnu_pbds::sample_probe_fn, [965](#)
- sample_probe_fn.hpp, [2237](#)
- sample_range_hashing
 - __gnu_pbds::sample_range_hashing, [967](#)
 - __gnu_pbds::sample_resize_policy, [971](#)
 - __gnu_pbds::sample_resize_trigger, [974](#)
 - __gnu_pbds::sample_size_policy, [976](#)
- sample_range_hashing.hpp, [2237](#)
- sample_ranged_hash_fn
 - __gnu_pbds::sample_ranged_hash_fn, [968](#)
- sample_ranged_hash_fn.hpp, [2237](#)
- sample_ranged_probe_fn.hpp, [2238](#)
- sample_resize_policy
 - __gnu_pbds::sample_resize_policy, [970](#)
- sample_resize_policy.hpp, [2238](#)
- sample_resize_trigger
 - __gnu_pbds::sample_resize_trigger, [973](#)
- sample_resize_trigger.hpp, [2238](#)
- sample_size_policy
 - __gnu_pbds::sample_size_policy, [975](#)
- sample_size_policy.hpp, [2239](#)
- sample_tree_node_update.hpp, [2239](#)
- sample_trie_access_traits.hpp, [2239](#)
- sample_trie_node_update
 - __gnu_pbds::sample_trie_node_update, [978](#)
- sample_trie_node_update.hpp, [2240](#)
- sample_update_policy
 - __gnu_pbds::sample_update_policy, [979](#)
- sample_update_policy.hpp, [2240](#)
- scan_is
 - std::__ctype_abstract_base, [1052](#)
 - std::ctype, [1339](#)
 - std::ctype< char >, [1352](#)
 - std::ctype< wchar_t >, [1368](#)
 - std::ctype_byname, [1383](#)
 - std::ctype_byname< char >, [1397](#)
- scan_not
 - std::__ctype_abstract_base, [1053](#)
 - std::ctype, [1339](#)
- std::ctype< char >, [1353](#)
- std::ctype< wchar_t >, [1369](#)
- std::ctype_byname, [1384](#)
- std::ctype_byname< char >, [1397](#)
- scientific
 - std, [419](#)
 - std::basic_ios, [1201](#)
 - std::ios_base, [1536](#)
- search
 - Non-Mutating, [102](#), [103](#)
- search.h, [2240](#)
- search_minimal_n
 - __gnu_parallel::Settings, [722](#)
- search_n
 - Non-Mutating, [103](#), [104](#)
- second
 - __gnu_parallel::IteratorPair, [672](#)
 - std::pair, [1781](#)
 - std::sub_match, [1877](#)
- second_argument_type
 - __gnu_parallel::EqualFromLess, [666](#)
 - __gnu_parallel::EqualTo, [667](#)
 - __gnu_parallel::Less, [675](#)
 - __gnu_parallel::Lexicographic, [677](#)
 - __gnu_parallel::LexicographicReverse, [678](#)
 - __gnu_parallel::Multiplies, [701](#)
 - __gnu_parallel::Plus, [704](#)
 - std::binary_function, [1269](#)
 - std::binary_negate, [1270](#)
 - std::const_mem_fun1_ref_t, [1325](#)
 - std::const_mem_fun1_t, [1326](#)
 - std::divides, [1438](#)
 - std::equal_to, [1440](#)
 - std::greater, [1486](#)
 - std::greater_equal, [1487](#)
 - std::less, [1548](#)
 - std::less_equal, [1549](#)
 - std::logical_and, [1588](#)
 - std::logical_or, [1590](#)
 - std::mem_fun1_ref_t, [1630](#)
 - std::mem_fun1_t, [1631](#)
 - std::minus, [1646](#)
 - std::modulus, [1648](#)
 - std::multiplies, [1698](#)
 - std::not_equal_to, [1728](#)
 - std::owner_less< shared_ptr< _Tp > >, [1777](#)
 - std::owner_less< weak_ptr< _Tp > >, [1778](#)
 - std::plus, [1792](#)
 - std::pointer_to_binary_function, [1793](#)
- second_type
 - __gnu_parallel::IteratorPair, [671](#)
 - std::pair, [1780](#)
 - std::sub_match, [1875](#)
- seed

- std::discard_block_engine, 1430
- std::independent_bits_engine, 1514
- std::linear_congruential_engine, 1552
- std::shuffle_order_engine, 1863
- seed_seq
 - std::seed_seq, 1832
- seekdir
 - std::basic_ios, 1183
 - std::ios_base, 1526
- select_on_container_copy_construction
 - __gnu_cxx::__alloc_traits, 463
 - std::allocator_traits, 1161
 - std::allocator_traits< allocator< _Tp > >, 1165
- Sequences, 177
- sequential
 - __gnu_parallel, 254
- set
 - __gnu_parallel::Settings, 717
 - std::set, 1837–1840
- Set Operation, 178
 - includes, 179
 - set_difference, 179, 180
 - set_intersection, 180, 181
 - set_symmetric_difference, 181, 182
 - set_union, 182, 183
- set.h, 2241, 2242
- set_difference
 - Set Operation, 179, 180
- set_difference_minimal_n
 - __gnu_parallel::Settings, 722
- set_intersection
 - Set Operation, 180, 181
- set_intersection_minimal_n
 - __gnu_parallel::Settings, 722
- set_load
 - __gnu_pbds::cc_hash_max_collision_check_↔
resize_trigger, 757
- set_loads
 - __gnu_pbds::hash_load_check_resize_trigger, 943
- set_num_threads
 - __gnu_parallel::balanced_quicksort_tag, 725
 - __gnu_parallel::balanced_tag, 726
 - __gnu_parallel::default_parallel_tag, 728
 - __gnu_parallel::exact_tag, 730
 - __gnu_parallel::multiway_mergesort_exact_tag, 733
 - __gnu_parallel::multiway_mergesort_sampling_tag,
734
 - __gnu_parallel::multiway_mergesort_tag, 736
 - __gnu_parallel::omp_loop_static_tag, 737
 - __gnu_parallel::omp_loop_tag, 738
 - __gnu_parallel::parallel_tag, 741
 - __gnu_parallel::quicksort_tag, 743
 - __gnu_parallel::sampling_tag, 744
 - __gnu_parallel::unbalanced_tag, 745
- set_operations.h, 2242
- set_symmetric_difference
 - Set Operation, 181, 182
- set_symmetric_difference_minimal_n
 - __gnu_parallel::Settings, 722
- set_union
 - Set Operation, 182, 183
- set_union_minimal_n
 - __gnu_parallel::Settings, 722
- setf
 - std::basic_ios, 1193
 - std::ios_base, 1530, 1531
- setstate
 - std::basic_ios, 1194
- settings.h, 2243
 - _GLIBCXX_PARALLEL_CONDITION, 2244
- shared_ptr
 - std::shared_ptr, 1854–1859
- shared_ptr.h, 2245
- shared_ptr_base.h, 2246
- showbase
 - std, 420
 - std::basic_ios, 1201
 - std::ios_base, 1536
- showpoint
 - std, 420
 - std::basic_ios, 1201
 - std::ios_base, 1537
- showpos
 - std, 420
 - std::basic_ios, 1201
 - std::ios_base, 1537
- shrink_to_fit
 - __gnu_cxx::__versa_string, 534
 - std::basic_string, 1262
 - std::deque, 1426
 - std::vector, 2034
- shuffle
 - Mutating, 85
- shuffle_order_engine
 - std::shuffle_order_engine, 1861, 1862
- size
 - __gnu_cxx::__versa_string, 534
 - Numeric_arrays, 120
 - std::_Temporary_buffer, 1152
 - std::basic_string, 1262
 - std::deque, 1426
 - std::list, 1572
 - std::map, 1614
 - std::match_results, 1626
 - std::multimap, 1695
 - std::multiset, 1714
 - std::priority_queue, 1805
 - std::queue, 1808

- std::set, 1850
- std::stack, 1869
- std::unordered_map, 1947
- std::unordered_multimap, 1969
- std::unordered_multiset, 1992
- std::unordered_set, 2014
- std::vector, 2034
- size_fn_imps.hpp, 2248
- size_type
 - __gnu_pbds::hash_prime_size_policy, 944
 - __gnu_pbds::sample_range_hashing, 966
 - __gnu_pbds::sample_resize_policy, 970
 - __gnu_pbds::sample_resize_trigger, 973
 - __gnu_pbds::sample_size_policy, 975
 - __gnu_pbds::trie_prefix_search_node_update, 996
- std::allocator_traits, 1158
- std::allocator_traits< allocator< _Tp > >, 1163
- std::set, 1837
- std::unordered_map, 1929
- std::unordered_multimap, 1953
- std::unordered_multiset, 1975
- std::unordered_set, 1998
- skipws
 - std, 420
 - std::basic_ios, 1201
 - std::ios_base, 1537
- slice
 - Numeric_arrays, 112
- slice_array
 - Numeric_arrays, 112
- slice_array.h, 2248
- sort
 - Sorting, 200, 201
 - std::forward_list, 1469
 - std::list, 1572
- sort.h, 2249
- sort_heap
 - Heap, 56
- sort_minimal_n
 - __gnu_parallel::_Settings, 722
- sort_mwms_oversampling
 - __gnu_parallel::_Settings, 722
- sort_qs_num_samples_preset
 - __gnu_parallel::_Settings, 722
- sort_qsb_base_case_maximal_n
 - __gnu_parallel::_Settings, 723
- Sorting, 184
 - inplace_merge, 186
 - is_sorted, 187
 - is_sorted_until, 188
 - lexicographical_compare, 188, 189
 - max, 189, 190
 - max_element, 190, 191
 - merge, 191
 - min, 192, 193
 - min_element, 193
 - minmax, 194
 - minmax_element, 195
 - next_permutation, 195, 196
 - nth_element, 196, 197
 - partial_sort, 197, 198
 - partial_sort_copy, 198, 199
 - prev_permutation, 199, 200
 - sort, 200, 201
 - stable_sort, 201, 202
- splay_fn_imps.hpp, 2249
- splay_tree_.hpp, 2250
- splice
 - std::list, 1572–1574
- splice_after
 - std::forward_list, 1469, 1470
- split_fn_imps.hpp, 2250
- split_join_can_throw
 - __gnu_pbds::container_traits, 765
- split_join_fn_imps.hpp, 2250–2252
- sregex_token_iterator
 - Regular Expressions, 145
- sso_string_base.h, 2252
- ssub_match
 - Regular Expressions, 145
- stable_partition
 - Mutating, 85
- stable_sort
 - Sorting, 201, 202
- stack
 - std::stack, 1868
- standard_policies.hpp, 2252
- start
 - Numeric_arrays, 121
- state
 - std::fpos, 1474
- static_pointer_cast
 - std, 420
- std, 300
 - _Construct, 377
 - _Destroy, 377
 - _final_insertion_sort, 370
 - _find_if, 370
 - _find_if_not, 370
 - _find_if_not_n, 370
 - _gcd, 371
 - _heap_select, 371
 - _inplace_stable_partition, 371
 - _inplace_stable_sort, 371
 - _insertion_sort, 371
 - _introsort_loop, 372
 - _lg, 372
 - _merge_adaptive, 372

- [__merge_without_buffer, 372](#)
- [__move_median_to_first, 372](#)
- [__move_merge, 373](#)
- [__move_merge_adaptive, 373](#)
- [__move_merge_adaptive_backward, 373](#)
- [__partition, 373](#)
- [__reverse, 374](#)
- [__rotate, 374](#)
- [__rotate_adaptive, 375](#)
- [__search_n_aux, 375](#)
- [__stable_partition_adaptive, 375](#)
- [__umap_traits, 368](#)
- [__ummap_traits, 368](#)
- [__umset_traits, 368](#)
- [__unguarded_insertion_sort, 375](#)
- [__unguarded_linear_insert, 376](#)
- [__unguarded_partition, 376](#)
- [__unguarded_partition_pivot, 376](#)
- [__unique_copy, 376, 377](#)
- [__uset_traits, 369](#)
- [accumulate, 377, 378](#)
- [adjacent_difference, 378, 379](#)
- [advance, 379](#)
- [begin, 380](#)
- [boolalpha, 381](#)
- [const_pointer_cast, 381](#)
- [dec, 381](#)
- [distance, 381](#)
- [dynamic_pointer_cast, 382](#)
- [end, 382, 384](#)
- [fixed, 384](#)
- [get_temporary_buffer, 384](#)
- [getline, 384–386](#)
- [hex, 387](#)
- [inner_product, 387](#)
- [internal, 388](#)
- [iota, 388](#)
- [isalnum, 388](#)
- [isalpha, 388](#)
- [isctrl, 389](#)
- [isdigit, 389](#)
- [isgraph, 389](#)
- [islower, 389](#)
- [isprint, 389](#)
- [ispunct, 389](#)
- [isspace, 390](#)
- [isupper, 390](#)
- [isxdigit, 390](#)
- [left, 390](#)
- [noboolalpha, 390](#)
- [noshowbase, 390](#)
- [noshowpoint, 391](#)
- [noshowpos, 391](#)
- [noskipws, 391](#)
- [nounitbuf, 391](#)
- [nouppercase, 391](#)
- [oct, 391](#)
- [operator!=, 392–394](#)
- [operator<, 396–402](#)
- [operator<<, 402](#)
- [operator<=, 403–405](#)
- [operator>, 412–414](#)
- [operator>>, 417](#)
- [operator>=, 414–416](#)
- [operator+, 394–396](#)
- [operator==, 405–411](#)
- [partial_sum, 417, 418](#)
- [replace_copy, 418](#)
- [return_temporary_buffer, 419](#)
- [right, 419](#)
- [scientific, 419](#)
- [showbase, 420](#)
- [showpoint, 420](#)
- [showpos, 420](#)
- [skipws, 420](#)
- [static_pointer_cast, 420](#)
- [streamoff, 369](#)
- [streampos, 369](#)
- [streamsize, 369](#)
- [swap, 420–422](#)
- [tolower, 422](#)
- [toupper, 422](#)
- [u16streampos, 369](#)
- [u32streampos, 369](#)
- [uninitialized_copy, 422](#)
- [uninitialized_copy_n, 423](#)
- [uninitialized_fill, 423](#)
- [uninitialized_fill_n, 424](#)
- [unitbuf, 424](#)
- [uppercase, 424](#)
- [wstreampos, 369](#)
- [std::__atomic_base< _ITp >, 1035](#)
- [std::__atomic_base< _PTp * >, 1036](#)
- [std::__atomic_flag_base, 1037](#)
- [std::__codecvt_abstract_base](#)
 - [do_out, 1039](#)
 - [in, 1039](#)
 - [out, 1040](#)
 - [unshift, 1041](#)
- [std::__codecvt_abstract_base< _InternT, _ExternT, _↵ StateT >, 1038](#)
- [std::__ctype_abstract_base](#)
 - [char_type, 1044](#)
 - [do_is, 1044, 1045](#)
 - [do_narrow, 1045, 1046](#)
 - [do_scan_is, 1046](#)
 - [do_scan_not, 1047](#)
 - [do_tolower, 1047, 1048](#)

- do_toupper, [1048](#), [1049](#)
- do_widen, [1049](#), [1050](#)
- is, [1050](#), [1051](#)
- narrow, [1051](#), [1052](#)
- scan_is, [1052](#)
- scan_not, [1053](#)
- tolower, [1053](#), [1054](#)
- toupper, [1054](#)
- widen, [1055](#)
- std::__ctype_abstract_base< _CharT >, [1042](#)
- std::__debug, [425](#)
- std::__debug::map
 - _M_attach, [1059](#)
 - _M_attach_single, [1059](#)
 - _M_const_iterators, [1060](#)
 - _M_detach, [1059](#)
 - _M_detach_all, [1059](#)
 - _M_detach_single, [1059](#)
 - _M_detach_singular, [1059](#)
 - _M_get_mutex, [1059](#)
 - _M_invalidate_all, [1059](#)
 - _M_invalidate_if, [1060](#)
 - _M_iterators, [1060](#)
 - _M_revalidate_singular, [1060](#)
 - _M_swap, [1060](#)
 - _M_transfer_from_if, [1060](#)
 - _M_version, [1060](#)
- std::__debug::map< _Key, _Tp, _Compare, _Allocator >, [1056](#)
- std::__debug::multimap
 - _M_attach, [1063](#)
 - _M_attach_single, [1063](#)
 - _M_const_iterators, [1065](#)
 - _M_detach, [1064](#)
 - _M_detach_all, [1064](#)
 - _M_detach_single, [1064](#)
 - _M_detach_singular, [1064](#)
 - _M_get_mutex, [1064](#)
 - _M_invalidate_all, [1064](#)
 - _M_invalidate_if, [1064](#)
 - _M_iterators, [1065](#)
 - _M_revalidate_singular, [1064](#)
 - _M_swap, [1065](#)
 - _M_transfer_from_if, [1065](#)
 - _M_version, [1065](#)
- std::__debug::multimap< _Key, _Tp, _Compare, _Allocator >, [1061](#)
- std::__debug::multiset
 - _M_attach, [1068](#)
 - _M_attach_single, [1068](#)
 - _M_const_iterators, [1070](#)
 - _M_detach, [1068](#)
 - _M_detach_all, [1069](#)
 - _M_detach_single, [1069](#)
 - _M_detach_singular, [1069](#)
 - _M_get_mutex, [1069](#)
 - _M_invalidate_all, [1069](#)
 - _M_invalidate_if, [1069](#)
 - _M_iterators, [1070](#)
 - _M_revalidate_singular, [1069](#)
 - _M_swap, [1069](#)
 - _M_transfer_from_if, [1070](#)
 - _M_version, [1070](#)
- std::__debug::multiset< _Key, _Compare, _Allocator >, [1066](#)
- std::__debug::set
 - _M_attach, [1073](#)
 - _M_attach_single, [1073](#)
 - _M_const_iterators, [1075](#)
 - _M_detach, [1073](#)
 - _M_detach_all, [1074](#)
 - _M_detach_single, [1074](#)
 - _M_detach_singular, [1074](#)
 - _M_get_mutex, [1074](#)
 - _M_invalidate_all, [1074](#)
 - _M_invalidate_if, [1074](#)
 - _M_iterators, [1075](#)
 - _M_revalidate_singular, [1074](#)
 - _M_swap, [1074](#)
 - _M_transfer_from_if, [1075](#)
 - _M_version, [1075](#)
- std::__debug::set< _Key, _Compare, _Allocator >, [1071](#)
- std::__detail, [427](#)
- std::__detail::BracketMatcher< _TraitsT, __icase, __collate >, [1075](#)
- std::__detail::Compiler< _TraitsT >, [1076](#)
- std::__detail::Default_ranged_hash, [1077](#)
- std::__detail::Equal_helper< _Key, _Value, _ExtractKey, _Equal, _HashCodeType, __cache_hash_code >, [1077](#)
- std::__detail::Equal_helper< _Key, _Value, _ExtractKey, _Equal, _HashCodeType, false >, [1077](#)
- std::__detail::Equal_helper< _Key, _Value, _ExtractKey, _Equal, _HashCodeType, true >, [1078](#)
- std::__detail::Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false >, [1079](#)
- std::__detail::Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true >, [1080](#)
- std::__detail::Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, _Unique_keys >, [1078](#)
- std::__detail::Equality_base, [1081](#)
- std::__detail::Executor< _Biliter, _Alloc, _TraitsT, __dfs_mode >, [1081](#)
- std::__detail::Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Default_ranged_hash,

- [false >, 1083](#)
[std::__detail::_Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Default_ranged_hash, true >, 1085](#)
[std::__detail::_Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache_hash_code >, 1083](#)
[std::__detail::_Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, false >, 1086](#)
[std::__detail::_Hash_node< _Value, _Cache_hash_code >, 1087](#)
[std::__detail::_Hash_node< _Value, false >, 1088](#)
[std::__detail::_Hash_node< _Value, true >, 1089](#)
[std::__detail::_Hash_node_base, 1090](#)
[std::__detail::_Hash_node_value_base< _Value >, 1091](#)
[std::__detail::_Hashtable_alloc< _NodeAlloc >, 1092](#)
[std::__detail::_Hashtable_base< _Key, _Value, _ExtractKey, _Equal, _H1, _H2, _Hash, _Traits >, 1093](#)
[std::__detail::_Hashtable_ebo_helper< _Nm, _Tp, __use_ebo >, 1095](#)
[std::__detail::_Hashtable_ebo_helper< _Nm, _Tp, false >, 1095](#)
[std::__detail::_Hashtable_ebo_helper< _Nm, _Tp, true >, 1095](#)
[std::__detail::_Hashtable_traits< _Cache_hash_code, __Constant_iterators, _Unique_keys >, 1096](#)
[std::__detail::_Insert< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, __Traits, false, _Unique_keys >, 1098](#)
[std::__detail::_Insert< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, __Traits, true, false >, 1099](#)
[std::__detail::_Insert< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, __Traits, true, true >, 1101](#)
[std::__detail::_Insert< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, __Traits, __Constant_iterators, _Unique_keys >, 1097](#)
[std::__detail::_Insert_base< _Key, _Value, _Alloc, __ExtractKey, _Equal, _H1, _H2, _Hash, __RehashPolicy, _Traits >, 1102](#)
[std::__detail::_List_node_base, 1104](#)
[std::__detail::_Local_const_iterator< _Key, _Value, __ExtractKey, _H1, _H2, _Hash, __constant_iterators, __cache >, 1105](#)
[std::__detail::_Local_iterator< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __constant_iterators, __cache >, 1106](#)
[std::__detail::_Local_iterator_base< _Key, _Value, __ExtractKey, _H1, _H2, _Hash, __cache_hash_code >, 1107](#)
[std::__detail::_Local_iterator_base< _Key, _Value, __ExtractKey, _H1, _H2, _Hash, true >, 1107](#)
[std::__detail::_Map_base< _Key, _Pair, _Alloc, __Select1st, _Equal, _H1, _H2, _Hash, __RehashPolicy, _Traits, false >, 1109](#)
[std::__detail::_Map_base< _Key, _Pair, _Alloc, __Select1st, _Equal, _H1, _H2, _Hash, __RehashPolicy, _Traits, true >, 1109](#)
[std::__detail::_Map_base< _Key, _Value, _Alloc, __ExtractKey, _Equal, _H1, _H2, _Hash, __RehashPolicy, _Traits, _Unique_keys >, 1108](#)
[std::__detail::_Mod_range_hashing, 1110](#)
[std::__detail::_Node_const_iterator< _Value, __constant_iterators, __cache >, 1111](#)
[std::__detail::_Node_iterator< _Value, __constant_iterators, __cache >, 1112](#)
[std::__detail::_Node_iterator_base< _Value, _Cache_hash_code >, 1113](#)
[std::__detail::_Prime_rehash_policy, 1114](#)
[std::__detail::_Rehash_base< _Key, _Value, _Alloc, __ExtractKey, _Equal, _H1, _H2, _Hash, _Prime_rehash_policy, _Traits >, 1115](#)
[std::__detail::_Rehash_base< _Key, _Value, _Alloc, __ExtractKey, _Equal, _H1, _H2, _Hash, __RehashPolicy, _Traits >, 1115](#)
[std::__detail::_Scanner](#)
[\[_TokenT, 1118\]\(#\)](#)
[std::__detail::_Scanner< _CharT >, 1116](#)
[std::__detail::_StateSeq< _TraitsT >, 1118](#)
[std::__exception_ptr::exception_ptr, 1119](#)
[std::__has_iterator_category_helper< _Tp >, 1119](#)
[std::__parallel, 429](#)
[std::__parallel::_CRandNumber< _MustBeInt >, 1120](#)
[std::__profile, 444](#)
[std::__profile::map< _Key, _Tp, _Compare, _Allocator >, 1120](#)
[std::__profile::multimap< _Key, _Tp, _Compare, __Allocator >, 1122](#)
[std::__profile::multiset< _Key, _Compare, _Allocator >, 1124](#)
[std::__profile::set< _Key, _Compare, _Allocator >, 1126](#)
[std::__Deque_base](#)
[\[_M_initialize_map, 1129\]\(#\)](#)
[std::__Deque_base< _Tp, _Alloc >, 1128](#)
[std::__Deque_iterator](#)
[\[_M_set_node, 1131\]\(#\)](#)
[std::__Deque_iterator< _Tp, _Ref, _Ptr >, 1130](#)
[std::__Enable_copy_move< _Copy, _CopyAssignment, __Move, _MoveAssignment, _Tag >, 1132](#)
[std::__Enable_default_constructor< _Switch, _Tag >, 1132](#)
[std::__Enable_destructor< _Switch, _Tag >, 1133](#)
[std::__Enable_special_members< _Default, _Destructor, __Copy, __CopyAssignment, __Move, __MoveAssignment, _Tag >, 1133](#)
[std::__Fwd_list_base< _Tp, _Alloc >, 1134](#)
[std::__Fwd_list_const_iterator< _Tp >, 1135](#)

std::_Fwd_list_iterator< _Tp >, 1136
 std::_Fwd_list_node< _Tp >, 1137
 std::_Fwd_list_node_base, 1138
 std::_Hashtable< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >, 1139
 std::_List_base< _Tp, _Alloc >, 1145
 std::_List_const_iterator< _Tp >, 1146
 std::_List_iterator< _Tp >, 1147
 std::_List_node
 _M_data, 1149
 std::_List_node< _Tp >, 1148
 std::_Sp_ebo_helper< _Nm, _Tp, false >, 1149
 std::_Sp_ebo_helper< _Nm, _Tp, true >, 1150
 std::_Temporary_buffer
 _Temporary_buffer, 1151
 begin, 1151
 end, 1151
 requested_size, 1152
 size, 1152
 std::_Temporary_buffer< _ForwardIterator, _Tp >, 1150
 std::_Vector_base< _Tp, _Alloc >, 1152
 std::allocator< _Tp >, 1153
 std::allocator< void >, 1155
 std::allocator_arg_t, 1155
 std::allocator_traits
 allocate, 1158, 1159
 allocator_type, 1157
 const_pointer, 1157
 const_void_pointer, 1157
 construct, 1159
 deallocate, 1159
 destroy, 1160
 difference_type, 1157
 max_size, 1160
 pointer, 1157
 propagate_on_container_copy_assignment, 1157
 propagate_on_container_move_assignment, 1158
 propagate_on_container_swap, 1158
 select_on_container_copy_construction, 1161
 size_type, 1158
 value_type, 1158
 void_pointer, 1158
 std::allocator_traits< _Alloc >, 1156
 std::allocator_traits< allocator< _Tp > >, 1161
 allocate, 1163, 1164
 allocator_type, 1162
 const_pointer, 1162
 const_void_pointer, 1162
 construct, 1164
 deallocate, 1164
 destroy, 1165
 difference_type, 1162
 max_size, 1165
 pointer, 1162
 propagate_on_container_copy_assignment, 1163
 propagate_on_container_move_assignment, 1163
 propagate_on_container_swap, 1163
 select_on_container_copy_construction, 1165
 size_type, 1163
 value_type, 1163
 void_pointer, 1163
 std::atomic_flag, 1166
 std::auto_ptr
 ~auto_ptr, 1169
 auto_ptr, 1168, 1169
 element_type, 1168
 get, 1170
 operator*, 1170
 operator->, 1170
 operator=, 1170, 1171
 release, 1171
 reset, 1171
 std::auto_ptr< _Tp >, 1167
 std::auto_ptr_ref< _Tp1 >, 1172
 std::back_insert_iterator
 back_insert_iterator, 1175
 container_type, 1174
 difference_type, 1174
 iterator_category, 1174
 operator*, 1175
 operator++, 1175
 operator=, 1175
 pointer, 1174
 reference, 1174
 value_type, 1174
 std::back_insert_iterator< _Container >, 1173
 std::bad_weak_ptr, 1176
 std::basic_ios
 _M_getloc, 1184
 __ctype_type, 1180
 __num_get_type, 1180
 __num_put_type, 1180
 ~basic_ios, 1183
 adjustfield, 1197
 app, 1197
 ate, 1197
 bad, 1184
 badbit, 1198
 basefield, 1198
 basic_ios, 1183, 1184
 beg, 1198
 binary, 1198
 boolalpha, 1198
 char_type, 1180
 clear, 1184
 copyfmt, 1185
 cur, 1198

dec, 1198
end, 1199
eof, 1185
eofbit, 1199
event, 1183
event_callback, 1180
exceptions, 1185, 1186
fail, 1186
failbit, 1199
fill, 1187
fixed, 1199
flags, 1188
floatfield, 1199
fmtflags, 1181
getloc, 1188
good, 1188
goodbit, 1199
hex, 1200
imbue, 1189
in, 1200
init, 1189
int_type, 1181
internal, 1200
iostate, 1182
iword, 1189
left, 1200
narrow, 1190
oct, 1200
off_type, 1182
openmode, 1182
operator void *, 1190
operator!, 1190
out, 1200
pos_type, 1182
precision, 1191
pword, 1191
rdbuf, 1192
rdstate, 1192
register_callback, 1193
right, 1201
scientific, 1201
seekdir, 1183
setf, 1193
setstate, 1194
showbase, 1201
showpoint, 1201
showpos, 1201
skipws, 1201
sync_with_stdio, 1194
tie, 1195
traits_type, 1183
trunc, 1202
unitbuf, 1202
unsetf, 1195
uppercase, 1202
widen, 1196
width, 1196
xalloc, 1197
std::basic_ios< _CharT, _Traits >, 1176
std::basic_regex
 ~basic_regex, 1207
 assign, 1207–1209
 basic_regex, 1204–1206
 flags, 1210
 getloc, 1210
 imbue, 1210
 mark_count, 1210
 operator=, 1210–1212
 swap, 1212
std::basic_regex< _Ch_type, _Rx_traits >, 1202
std::basic_string
 ~basic_string, 1220
 append, 1220–1222
 assign, 1223–1226
 at, 1226, 1227
 back, 1227
 basic_string, 1216–1220
 begin, 1227
 c_str, 1228
 capacity, 1228
 cbegin, 1228
 cend, 1228
 clear, 1228
 compare, 1228–1231
 copy, 1231
 crbegin, 1232
 crend, 1232
 data, 1232
 empty, 1232
 end, 1232, 1233
 erase, 1233, 1234
 find, 1234, 1235
 find_first_not_of, 1236, 1237
 find_first_of, 1237–1239
 find_last_not_of, 1239, 1240
 find_last_of, 1241, 1242
 front, 1242
 get_allocator, 1243
 insert, 1243–1247
 length, 1248
 max_size, 1248
 npos, 1263
 operator+=, 1248, 1249
 operator=, 1249, 1250
 operator[], 1251
 pop_back, 1252
 push_back, 1252
 rbegin, 1252

- rend, [1252](#)
- replace, [1253–1258](#)
- reserve, [1259](#)
- resize, [1260](#)
- rfind, [1260](#), [1261](#)
- shrink_to_fit, [1262](#)
- size, [1262](#)
- substr, [1262](#)
- swap, [1263](#)
- std::basic_string< _CharT, _Traits, _Alloc >, [1212](#)
- std::bernoulli_distribution, [1264](#)
 - bernoulli_distribution, [1265](#)
 - max, [1265](#)
 - min, [1265](#)
 - operator(), [1265](#)
 - operator==, [1266](#)
 - p, [1265](#)
 - param, [1266](#)
 - reset, [1266](#)
 - result_type, [1265](#)
- std::bernoulli_distribution::param_type, [1267](#)
- std::bidirectional_iterator_tag, [1267](#)
- std::binary_function
 - first_argument_type, [1269](#)
 - result_type, [1269](#)
 - second_argument_type, [1269](#)
- std::binary_function< _Arg1, _Arg2, _Result >, [1268](#)
- std::binary_negate
 - first_argument_type, [1270](#)
 - result_type, [1270](#)
 - second_argument_type, [1270](#)
- std::binary_negate< _Predicate >, [1269](#)
- std::binder1st
 - argument_type, [1272](#)
 - result_type, [1272](#)
- std::binder1st< _Operation >, [1271](#)
- std::binder2nd
 - argument_type, [1273](#)
 - result_type, [1273](#)
- std::binder2nd< _Operation >, [1272](#)
- std::binomial_distribution
 - max, [1275](#)
 - min, [1275](#)
 - operator<<, [1276](#)
 - operator>>, [1276](#)
 - operator(), [1275](#)
 - operator==, [1276](#)
 - p, [1275](#)
 - param, [1275](#)
 - reset, [1276](#)
 - result_type, [1275](#)
 - t, [1276](#)
- std::binomial_distribution< _IntType >, [1273](#)
- std::binomial_distribution< _IntType >::param_type, [1277](#)
- std::cauchy_distribution
 - max, [1279](#)
 - min, [1279](#)
 - operator(), [1279](#)
 - operator==, [1280](#)
 - param, [1279](#)
 - reset, [1280](#)
 - result_type, [1279](#)
- std::cauchy_distribution< _RealType >, [1278](#)
- std::cauchy_distribution< _RealType >::param_type, [1280](#)
- std::char_traits< __gnu_cxx::character< _Value, _Int, _St > >, [1282](#)
- std::char_traits< _CharT >, [1281](#)
- std::char_traits< char >, [1283](#)
- std::char_traits< wchar_t >, [1284](#)
- std::chi_squared_distribution
 - max, [1287](#)
 - min, [1287](#)
 - operator<<, [1288](#)
 - operator>>, [1288](#)
 - operator(), [1287](#)
 - operator==, [1288](#)
 - param, [1287](#)
 - reset, [1288](#)
 - result_type, [1287](#)
- std::chi_squared_distribution< _RealType >, [1285](#)
- std::chi_squared_distribution< _RealType >::param_type, [1289](#)
- std::codecvt
 - do_out, [1292](#)
 - in, [1292](#)
 - out, [1293](#)
 - unshift, [1293](#)
- std::codecvt< _InternT, _ExternT, _StateT >, [1290](#)
- std::codecvt< _InternT, _ExternT, encoding_state >, [1295](#)
 - do_out, [1296](#)
 - in, [1296](#)
 - out, [1297](#)
 - unshift, [1298](#)
- std::codecvt< char, char, mbstate_t >, [1299](#)
 - do_out, [1301](#)
 - in, [1301](#)
 - out, [1302](#)
 - unshift, [1302](#)
- std::codecvt< wchar_t, char, mbstate_t >, [1304](#)
 - do_out, [1305](#)
 - in, [1305](#)
 - out, [1306](#)
 - unshift, [1307](#)
- std::codecvt_base, [1308](#)
- std::codecvt_byname
 - do_out, [1311](#)
 - in, [1311](#)

- out, [1312](#)
- unshift, [1312](#)
- std::codecvt_byname< _InternT, _ExternT, _StateT >, [1309](#)
- std::collate
 - ~collate, [1316](#)
 - char_type, [1315](#)
 - collate, [1315](#)
 - compare, [1316](#)
 - do_compare, [1316](#)
 - do_hash, [1317](#)
 - do_transform, [1317](#)
 - hash, [1318](#)
 - id, [1319](#)
 - string_type, [1315](#)
 - transform, [1318](#)
- std::collate< _CharT >, [1313](#)
- std::collate_byname
 - char_type, [1321](#)
 - compare, [1321](#)
 - do_compare, [1321](#)
 - do_hash, [1322](#)
 - do_transform, [1322](#)
 - hash, [1323](#)
 - id, [1324](#)
 - string_type, [1321](#)
 - transform, [1323](#)
- std::collate_byname< _CharT >, [1319](#)
- std::const_mem_fun1_ref_t
 - first_argument_type, [1325](#)
 - result_type, [1325](#)
 - second_argument_type, [1325](#)
- std::const_mem_fun1_ref_t< _Ret, _Tp, _Arg >, [1324](#)
- std::const_mem_fun1_t
 - first_argument_type, [1326](#)
 - result_type, [1326](#)
 - second_argument_type, [1326](#)
- std::const_mem_fun1_t< _Ret, _Tp, _Arg >, [1325](#)
- std::const_mem_fun_ref_t
 - argument_type, [1327](#)
 - result_type, [1327](#)
- std::const_mem_fun_ref_t< _Ret, _Tp >, [1327](#)
- std::const_mem_fun_t
 - argument_type, [1329](#)
 - result_type, [1329](#)
- std::const_mem_fun_t< _Ret, _Tp >, [1328](#)
- std::ctype
 - do_is, [1331](#), [1332](#)
 - do_narrow, [1332](#), [1333](#)
 - do_scan_is, [1333](#)
 - do_scan_not, [1334](#)
 - do_tolower, [1334](#), [1335](#)
 - do_toupper, [1335](#), [1336](#)
 - do_widen, [1336](#), [1337](#)
 - id, [1343](#)
 - is, [1337](#), [1338](#)
 - narrow, [1338](#)
 - scan_is, [1339](#)
 - scan_not, [1339](#)
 - tolower, [1340](#)
 - toupper, [1341](#)
 - widen, [1342](#)
- std::ctype< _CharT >, [1329](#)
- std::ctype< char >, [1343](#)
 - ~ctype, [1346](#)
 - char_type, [1345](#)
 - classic_table, [1346](#)
 - ctype, [1345](#), [1346](#)
 - do_narrow, [1346](#), [1347](#)
 - do_tolower, [1347](#), [1348](#)
 - do_toupper, [1348](#), [1349](#)
 - do_widen, [1349](#)
 - id, [1356](#)
 - is, [1350](#)
 - narrow, [1351](#)
 - scan_is, [1352](#)
 - scan_not, [1353](#)
 - table, [1353](#)
 - table_size, [1356](#)
 - tolower, [1353](#), [1354](#)
 - toupper, [1354](#), [1355](#)
 - widen, [1355](#), [1356](#)
- std::ctype< wchar_t >, [1357](#)
 - ~ctype, [1360](#)
 - char_type, [1359](#)
 - ctype, [1359](#), [1360](#)
 - do_is, [1360](#)
 - do_narrow, [1361](#)
 - do_scan_is, [1362](#)
 - do_scan_not, [1363](#)
 - do_tolower, [1363](#), [1364](#)
 - do_toupper, [1364](#)
 - do_widen, [1365](#)
 - id, [1372](#)
 - is, [1366](#)
 - narrow, [1367](#)
 - scan_is, [1368](#)
 - scan_not, [1369](#)
 - tolower, [1369](#), [1370](#)
 - toupper, [1370](#), [1371](#)
 - widen, [1371](#)
- std::ctype_base, [1372](#)
- std::ctype_byname
 - do_is, [1376](#)
 - do_narrow, [1376](#), [1377](#)
 - do_scan_is, [1378](#)
 - do_scan_not, [1378](#)
 - do_tolower, [1379](#)

- do_toupper, 1380
- do_widen, 1381
- id, 1388
- is, 1382
- narrow, 1382, 1383
- scan_is, 1383
- scan_not, 1384
- tolower, 1384, 1386
- toupper, 1386, 1387
- widen, 1387
- std::ctype_byname< _CharT >, 1374
- std::ctype_byname< char >, 1389
 - char_type, 1391
 - classic_table, 1391
 - do_narrow, 1391
 - do_tolower, 1392
 - do_toupper, 1393
 - do_widen, 1394
 - id, 1401
 - is, 1395
 - narrow, 1396
 - scan_is, 1397
 - scan_not, 1397
 - table, 1398
 - table_size, 1401
 - tolower, 1398
 - toupper, 1399
 - widen, 1400
- std::default_delete
 - default_delete, 1402
 - operator(), 1402
- std::default_delete< _Tp >, 1401
- std::default_delete< _Tp[] >, 1402
 - default_delete, 1403
 - operator(), 1403
- std::deque
 - _M_fill_initialize, 1411
 - _M_initialize_map, 1412
 - _M_new_elements_at_back, 1412
 - _M_new_elements_at_front, 1412
 - _M_pop_back_aux, 1412
 - _M_pop_front_aux, 1412
 - _M_push_back_aux, 1413
 - _M_push_front_aux, 1413
 - _M_range_check, 1413
 - _M_range_initialize, 1413
 - _M_reallocate_map, 1414
 - _M_reserve_elements_at_back, 1414
 - _M_reserve_elements_at_front, 1414
 - _M_reserve_map_at_back, 1414
 - _M_reserve_map_at_front, 1414
 - ~deque, 1411
 - assign, 1414, 1415
 - at, 1415, 1416
 - back, 1416, 1417
 - begin, 1417
 - cbegin, 1417
 - cend, 1417
 - clear, 1417
 - crbegin, 1418
 - crend, 1418
 - deque, 1409–1411
 - emplace, 1418
 - empty, 1418
 - end, 1418, 1419
 - erase, 1419
 - front, 1420
 - get_allocator, 1420
 - insert, 1420–1422
 - max_size, 1422
 - operator=, 1422, 1423
 - operator[], 1423, 1424
 - pop_back, 1424
 - pop_front, 1424
 - push_back, 1424
 - push_front, 1425
 - rbegin, 1425
 - rend, 1425
 - resize, 1426
 - shrink_to_fit, 1426
 - size, 1426
 - swap, 1426
- std::deque< _Tp, _Alloc >, 1404
- std::discard_block_engine
 - base, 1430
 - discard, 1430
 - discard_block_engine, 1428, 1429
 - max, 1430
 - min, 1430
 - operator<<, 1431
 - operator>>, 1431
 - operator(), 1430
 - operator==, 1431
 - result_type, 1428
 - seed, 1430
- std::discard_block_engine< _RandomNumberEngine, _<_p, __r >, 1427
- std::discrete_distribution
 - max, 1434
 - min, 1434
 - operator<<, 1435
 - operator>>, 1435
 - operator(), 1434
 - operator==, 1435
 - param, 1434
 - probabilities, 1434
 - reset, 1434
 - result_type, 1433

std::discrete_distribution< _IntType >, 1432
 std::discrete_distribution< _IntType >::param_type, 1436
 std::divides
 first_argument_type, 1437
 result_type, 1437
 second_argument_type, 1438
 std::divides< _Tp >, 1437
 std::enable_shared_from_this< _Tp >, 1438
 std::equal_to
 first_argument_type, 1439
 result_type, 1439
 second_argument_type, 1440
 std::equal_to< _Tp >, 1439
 std::exponential_distribution
 exponential_distribution, 1441
 lambda, 1442
 max, 1442
 min, 1442
 operator(), 1442
 operator==, 1443
 param, 1442
 reset, 1443
 result_type, 1441
 std::exponential_distribution< _RealType >, 1440
 std::exponential_distribution< _RealType >::param_type, 1443
 std::extreme_value_distribution
 a, 1445
 b, 1445
 max, 1445
 min, 1446
 operator(), 1446
 operator==, 1447
 param, 1446
 reset, 1446
 result_type, 1445
 std::extreme_value_distribution< _RealType >, 1444
 std::extreme_value_distribution< _RealType >::param_type, 1447
 std::fisher_f_distribution
 max, 1449
 min, 1449
 operator<<, 1450
 operator>>, 1450
 operator(), 1449
 operator==, 1450
 param, 1449
 reset, 1450
 result_type, 1449
 std::fisher_f_distribution< _RealType >, 1448
 std::fisher_f_distribution< _RealType >::param_type, 1451
 std::forward_iterator_tag, 1452
 std::forward_list
 ~forward_list, 1458
 assign, 1459
 before_begin, 1459, 1460
 begin, 1460
 cbefore_begin, 1460
 cbegin, 1460
 cend, 1460
 clear, 1461
 emplace_after, 1461
 emplace_front, 1461
 empty, 1462
 end, 1462
 erase_after, 1462, 1463
 forward_list, 1456–1458
 front, 1463
 get_allocator, 1463
 insert_after, 1463–1465
 max_size, 1465
 merge, 1465, 1466
 operator=, 1466
 pop_front, 1467
 push_front, 1467
 remove, 1467
 remove_if, 1468
 resize, 1468
 reverse, 1468
 sort, 1469
 splice_after, 1469, 1470
 swap, 1470
 unique, 1470, 1471
 std::forward_list< _Tp, _Alloc >, 1453
 std::fpos
 fpos, 1473
 operator streamoff, 1473
 operator+, 1473
 operator+=, 1473
 operator-, 1473
 operator=, 1473
 state, 1474
 std::fpos< _StateT >, 1471
 std::front_insert_iterator
 container_type, 1475
 difference_type, 1475
 front_insert_iterator, 1476
 iterator_category, 1475
 operator*, 1476
 operator++, 1476
 operator=, 1476
 pointer, 1475
 reference, 1476
 value_type, 1476
 std::front_insert_iterator< _Container >, 1474
 std::gamma_distribution
 alpha, 1479

- beta, 1479
- gamma_distribution, 1478
- max, 1479
- min, 1479
- operator<<, 1480
- operator>>, 1480
- operator(), 1479
- operator==, 1480
- param, 1479
- reset, 1480
- result_type, 1478
- std::gamma_distribution< _RealType >, 1477
- std::gamma_distribution< _RealType >::param_type, 1481
- std::geometric_distribution
 - max, 1483
 - min, 1483
 - operator(), 1483
 - operator==, 1484
 - p, 1483
 - param, 1483
 - reset, 1484
 - result_type, 1483
- std::geometric_distribution< _IntType >, 1482
- std::geometric_distribution< _IntType >::param_type, 1484
- std::greater
 - first_argument_type, 1486
 - result_type, 1486
 - second_argument_type, 1486
- std::greater< _Tp >, 1485
- std::greater_equal
 - first_argument_type, 1487
 - result_type, 1487
 - second_argument_type, 1487
- std::greater_equal< _Tp >, 1486
- std::gslice, 1487
- std::gslice_array< _Tp >, 1488
- std::hash< __gnu_cxx::__u16vstring >, 1490
- std::hash< __gnu_cxx::__u32vstring >, 1490
- std::hash< __gnu_cxx::__vstring >, 1491
- std::hash< __gnu_cxx::__wvstring >, 1492
- std::hash< __gnu_cxx::throw_value_limit >, 1492
 - argument_type, 1493
 - result_type, 1493
- std::hash< __gnu_cxx::throw_value_random >, 1494
 - argument_type, 1494
 - result_type, 1494
- std::hash< __shared_ptr< _Tp, _Lp > >, 1495
- std::hash< _Tp >, 1490
- std::hash< _Tp * >, 1495
- std::hash< bool >, 1496
- std::hash< char >, 1497
- std::hash< char16_t >, 1497
- std::hash< char32_t >, 1498
- std::hash< double >, 1498
- std::hash< float >, 1499
- std::hash< int >, 1500
- std::hash< long >, 1500
- std::hash< long double >, 1501
- std::hash< long long >, 1501
- std::hash< shared_ptr< _Tp > >, 1502
- std::hash< short >, 1503
- std::hash< signed char >, 1503
- std::hash< string >, 1504
- std::hash< u16string >, 1504
- std::hash< u32string >, 1505
- std::hash< unique_ptr< _Tp, _Dp > >, 1506
- std::hash< unsigned char >, 1506
- std::hash< unsigned int >, 1507
- std::hash< unsigned long >, 1507
- std::hash< unsigned long long >, 1508
- std::hash< unsigned short >, 1509
- std::hash< wchar_t >, 1509
- std::hash< wstring >, 1510
- std::hash<::vector< bool, _Alloc > >, 1510
- std::independent_bits_engine
 - base, 1514
 - discard, 1514
 - independent_bits_engine, 1512, 1513
 - max, 1514
 - min, 1514
 - operator>>, 1515
 - operator(), 1514
 - operator==, 1515
 - result_type, 1512
 - seed, 1514
- std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >, 1511
- std::indirect_array< _Tp >, 1516
- std::input_iterator_tag, 1518
- std::insert_iterator
 - container_type, 1520
 - difference_type, 1520
 - insert_iterator, 1521
 - iterator_category, 1520
 - operator*, 1521
 - operator++, 1521
 - operator=, 1521
 - pointer, 1520
 - reference, 1520
 - value_type, 1520
- std::insert_iterator< _Container >, 1519
- std::ios_base, 1522
 - _M_getloc, 1527
 - ~ios_base, 1527
 - adjustfield, 1533
 - app, 1533

- ate, 1533
- badbit, 1533
- basefield, 1533
- beg, 1533
- binary, 1534
- boolalpha, 1534
- cur, 1534
- dec, 1534
- end, 1534
- eofbit, 1534
- event, 1527
- event_callback, 1525
- failbit, 1534
- fixed, 1535
- flags, 1527, 1528
- floatfield, 1535
- fmtflags, 1525
- getloc, 1528
- goodbit, 1535
- hex, 1535
- imbue, 1528
- in, 1535
- internal, 1535
- iostate, 1526
- isword, 1529
- left, 1536
- oct, 1536
- openmode, 1526
- out, 1536
- precision, 1529
- pword, 1530
- register_callback, 1530
- right, 1536
- scientific, 1536
- seekdir, 1526
- setf, 1530, 1531
- showbase, 1536
- showpoint, 1537
- showpos, 1537
- skipws, 1537
- sync_with_stdio, 1531
- trunc, 1537
- unitbuf, 1537
- unsetf, 1531
- uppercase, 1537
- width, 1532
- xalloc, 1532
- std::ios_base::failure, 1538
- std::istream_iterator
 - difference_type, 1539
 - istream_iterator, 1540
 - iterator_category, 1539
 - pointer, 1539
 - reference, 1540
 - value_type, 1540
- std::istream_iterator< _Tp, _CharT, _Traits, _Dist >, 1538
- std::istreambuf_iterator
 - char_type, 1542
 - difference_type, 1542
 - equal, 1544
 - int_type, 1542
 - istream_type, 1542
 - istreambuf_iterator, 1543, 1544
 - iterator_category, 1542
 - operator*, 1544
 - operator++, 1544
 - pointer, 1543
 - reference, 1543
 - streambuf_type, 1543
 - traits_type, 1543
 - value_type, 1543
- std::istreambuf_iterator< _CharT, _Traits >, 1541
- std::iterator
 - difference_type, 1545
 - iterator_category, 1545
 - pointer, 1545
 - reference, 1545
 - value_type, 1546
- std::iterator< _Category, _Tp, _Distance, _Pointer, _↵
Reference >, 1545
- std::iterator_traits< _Tp * >, 1546
- std::iterator_traits< const _Tp * >, 1546
- std::less
 - first_argument_type, 1548
 - result_type, 1548
 - second_argument_type, 1548
- std::less< _Tp >, 1547
- std::less_equal
 - first_argument_type, 1549
 - result_type, 1549
 - second_argument_type, 1549
- std::less_equal< _Tp >, 1548
- std::linear_congruential_engine
 - discard, 1551
 - increment, 1554
 - linear_congruential_engine, 1551
 - max, 1551
 - min, 1552
 - modulus, 1554
 - multiplier, 1554
 - operator<<, 1553
 - operator>>, 1553
 - operator(), 1552
 - operator==, 1553
 - result_type, 1551
 - seed, 1552
- std::linear_congruential_engine< _UIntType, __a, __c, ↵
__m >, 1549

- std::list
 - _M_create_node, 1560
 - assign, 1561
 - back, 1562
 - begin, 1562
 - cbegin, 1562
 - cend, 1562
 - clear, 1563
 - crbegin, 1563
 - crend, 1563
 - emplace, 1563
 - empty, 1563
 - end, 1564
 - erase, 1564
 - front, 1565
 - get_allocator, 1565
 - insert, 1565–1567
 - list, 1558–1560
 - max_size, 1567
 - merge, 1567, 1568
 - operator=, 1568, 1569
 - pop_back, 1569
 - pop_front, 1569
 - push_back, 1569
 - push_front, 1570
 - rbegin, 1570
 - remove, 1570
 - remove_if, 1571
 - rend, 1571
 - resize, 1571
 - reverse, 1572
 - size, 1572
 - sort, 1572
 - splice, 1572–1574
 - swap, 1574
 - unique, 1574
- std::list< _Tp, _Alloc >, 1554
- std::locale, 1575
 - ~locale, 1578
 - all, 1581
 - category, 1576
 - classic, 1578
 - collate, 1581
 - combine, 1578
 - ctype, 1581
 - global, 1579
 - locale, 1577, 1578
 - messages, 1582
 - monetary, 1582
 - name, 1579
 - none, 1582
 - numeric, 1582
 - operator!=, 1579
 - operator(), 1580
 - operator=, 1580
 - operator==, 1581
 - time, 1582
- std::locale::facet, 1584
 - ~facet, 1585
 - facet, 1585
- std::locale::id, 1586
 - id, 1586
- std::logical_and
 - first_argument_type, 1587
 - result_type, 1587
 - second_argument_type, 1588
- std::logical_and< _Tp >, 1587
- std::logical_not
 - argument_type, 1589
 - result_type, 1589
- std::logical_not< _Tp >, 1588
- std::logical_or
 - first_argument_type, 1590
 - result_type, 1590
 - second_argument_type, 1590
- std::logical_or< _Tp >, 1589
- std::lognormal_distribution
 - max, 1592
 - min, 1592
 - operator<<, 1593
 - operator>>, 1593
 - operator(), 1592
 - operator==, 1593
 - param, 1592
 - reset, 1593
 - result_type, 1592
- std::lognormal_distribution< _RealType >, 1591
- std::lognormal_distribution< _RealType >::param_type, 1594
- std::map
 - at, 1600
 - begin, 1601
 - cbegin, 1601
 - cend, 1601
 - clear, 1601
 - count, 1602
 - crbegin, 1603
 - crend, 1603
 - emplace, 1603
 - emplace_hint, 1604
 - empty, 1604
 - end, 1604
 - equal_range, 1605
 - erase, 1606, 1607
 - find, 1607, 1608
 - get_allocator, 1608
 - insert, 1608–1610
 - key_comp, 1610

lower_bound, 1610, 1611
 map, 1597–1600
 max_size, 1611
 operator=, 1611, 1612
 operator[], 1612
 rbegin, 1613
 rend, 1613
 size, 1614
 swap, 1614
 upper_bound, 1614, 1615
 value_comp, 1615
 std::map< _Key, _Tp, _Compare, _Alloc >, 1595
 std::mask_array< _Tp >, 1615
 std::match_results
 ~match_results, 1622
 begin, 1622
 cbegin, 1622
 cend, 1622
 empty, 1622
 end, 1623
 format, 1623
 get_allocator, 1624
 length, 1624
 match_results, 1622
 max_size, 1624
 operator=, 1624, 1625
 operator[], 1625
 position, 1625
 prefix, 1626
 ready, 1626
 size, 1626
 str, 1626
 suffix, 1628
 swap, 1628
 std::match_results< _Bi_iter, _Alloc >, 1617
 std::mem_fun1_ref_t
 first_argument_type, 1629
 result_type, 1629
 second_argument_type, 1630
 std::mem_fun1_ref_t< _Ret, _Tp, _Arg >, 1629
 std::mem_fun1_t
 first_argument_type, 1631
 result_type, 1631
 second_argument_type, 1631
 std::mem_fun1_t< _Ret, _Tp, _Arg >, 1630
 std::mem_fun_ref_t
 argument_type, 1632
 result_type, 1632
 std::mem_fun_ref_t< _Ret, _Tp >, 1631
 std::mem_fun_t
 argument_type, 1633
 result_type, 1633
 std::mem_fun_t< _Ret, _Tp >, 1633
 std::mersenne_twister_engine
 discard, 1636
 max, 1637
 mersenne_twister_engine, 1636
 min, 1637
 operator<<, 1637
 operator>>, 1638
 operator==, 1637
 result_type, 1636
 std::mersenne_twister_engine< _UIntType, __w, __n, __←
 __m, __r, __a, __u, __d, __s, __b, __t, __c, __l,
 __f >, 1634
 std::messages
 ~messages, 1641
 char_type, 1641
 do_get, 1642
 id, 1642
 messages, 1641
 string_type, 1641
 std::messages< _CharT >, 1639
 std::messages_base, 1642
 std::messages_byname
 do_get, 1645
 id, 1645
 std::messages_byname< _CharT >, 1643
 std::minus
 first_argument_type, 1646
 result_type, 1646
 second_argument_type, 1646
 std::minus< _Tp >, 1645
 std::modulus
 first_argument_type, 1647
 result_type, 1647
 second_argument_type, 1648
 std::modulus< _Tp >, 1647
 std::money_base, 1648
 std::money_get
 ~money_get, 1651
 char_type, 1651
 do_get, 1652
 get, 1652, 1653
 id, 1653
 iter_type, 1651
 money_get, 1651
 string_type, 1651
 std::money_get< _CharT, _InIter >, 1649
 std::money_put
 ~money_put, 1656
 char_type, 1655
 do_put, 1656
 id, 1658
 iter_type, 1655
 money_put, 1655
 put, 1657
 string_type, 1655

- `std::money_put< _CharT, _OutIter >`, 1654
- `std::moneypunct`
 - `~moneypunct`, 1662
 - `char_type`, 1661
 - `curr_symbol`, 1662
 - `decimal_point`, 1662
 - `do_curr_symbol`, 1662
 - `do_decimal_point`, 1663
 - `do_frac_digits`, 1663
 - `do_grouping`, 1663
 - `do_neg_format`, 1664
 - `do_negative_sign`, 1664
 - `do_pos_format`, 1664
 - `do_positive_sign`, 1665
 - `do_thousands_sep`, 1665
 - `frac_digits`, 1665
 - `grouping`, 1666
 - `id`, 1669
 - `intl`, 1669
 - `moneypunct`, 1661
 - `neg_format`, 1666
 - `negative_sign`, 1667
 - `pos_format`, 1667
 - `positive_sign`, 1668
 - `string_type`, 1661
 - `thousands_sep`, 1668
- `std::moneypunct< _CharT, _Intl >`, 1659
- `std::moneypunct_byname`
 - `curr_symbol`, 1671
 - `decimal_point`, 1671
 - `do_curr_symbol`, 1671
 - `do_decimal_point`, 1672
 - `do_frac_digits`, 1672
 - `do_grouping`, 1672
 - `do_neg_format`, 1673
 - `do_negative_sign`, 1673
 - `do_pos_format`, 1673
 - `do_positive_sign`, 1674
 - `do_thousands_sep`, 1674
 - `frac_digits`, 1674
 - `grouping`, 1675
 - `id`, 1678
 - `neg_format`, 1675
 - `negative_sign`, 1676
 - `pos_format`, 1676
 - `positive_sign`, 1677
 - `thousands_sep`, 1677
- `std::moneypunct_byname< _CharT, _Intl >`, 1669
- `std::move_iterator< _Iterator >`, 1678
- `std::multimap`
 - `begin`, 1685
 - `cbegin`, 1685
 - `cend`, 1685
 - `clear`, 1685
 - `count`, 1685
 - `crbegin`, 1686
 - `crend`, 1686
 - `emplace`, 1686
 - `emplace_hint`, 1686
 - `empty`, 1687
 - `end`, 1687
 - `equal_range`, 1687, 1688
 - `erase`, 1688, 1689
 - `find`, 1690
 - `get_allocator`, 1690
 - `insert`, 1691, 1692
 - `key_comp`, 1692
 - `lower_bound`, 1693
 - `max_size`, 1693
 - `multimap`, 1682–1684
 - `operator=`, 1694
 - `rbegin`, 1695
 - `rend`, 1695
 - `size`, 1695
 - `swap`, 1696
 - `upper_bound`, 1696
 - `value_comp`, 1697
- `std::multimap< _Key, _Tp, _Compare, _Alloc >`, 1679
- `std::multiplies`
 - `first_argument_type`, 1698
 - `result_type`, 1698
 - `second_argument_type`, 1698
- `std::multiplies< _Tp >`, 1697
- `std::multiset`
 - `begin`, 1704
 - `cbegin`, 1704
 - `cend`, 1704
 - `clear`, 1704
 - `count`, 1704
 - `crbegin`, 1705
 - `crend`, 1705
 - `emplace`, 1705
 - `emplace_hint`, 1705
 - `empty`, 1706
 - `end`, 1706
 - `equal_range`, 1706, 1707
 - `erase`, 1707, 1708
 - `find`, 1709
 - `get_allocator`, 1709
 - `insert`, 1710, 1711
 - `key_comp`, 1711
 - `lower_bound`, 1711, 1712
 - `max_size`, 1712
 - `multiset`, 1701–1703
 - `operator=`, 1712, 1713
 - `rbegin`, 1713
 - `rend`, 1714
 - `size`, 1714

- swap, 1714
- upper_bound, 1714, 1715
- value_comp, 1715
- std::multiset< _Key, _Compare, _Alloc >, 1699
- std::negate
 - argument_type, 1716
 - result_type, 1716
- std::negate< _Tp >, 1716
- std::negative_binomial_distribution
 - k, 1718
 - max, 1718
 - min, 1718
 - operator<<, 1719
 - operator>>, 1720
 - operator(), 1719
 - operator==, 1720
 - p, 1719
 - param, 1719
 - reset, 1719
 - result_type, 1718
- std::negative_binomial_distribution< _IntType >, 1717
- std::negative_binomial_distribution< _IntType >::param←_type, 1720
- std::nested_exception, 1721
- std::normal_distribution
 - max, 1723
 - mean, 1723
 - min, 1723
 - normal_distribution, 1723
 - operator<<, 1725
 - operator>>, 1726
 - operator(), 1724
 - operator==, 1726
 - param, 1724
 - reset, 1724
 - result_type, 1723
 - stddev, 1724
- std::normal_distribution< _RealType >, 1722
- std::normal_distribution< _RealType >::param_type, 1726
- std::not_equal_to
 - first_argument_type, 1728
 - result_type, 1728
 - second_argument_type, 1728
- std::not_equal_to< _Tp >, 1727
- std::num_get
 - ~num_get, 1731
 - char_type, 1730
 - do_get, 1731–1736
 - get, 1737–1743
 - id, 1744
 - iter_type, 1730
 - num_get, 1731
- std::num_get< _CharT, _InIter >, 1728
- std::num_put
 - ~num_put, 1746
 - char_type, 1746
 - do_put, 1747–1750
 - id, 1756
 - iter_type, 1746
 - num_put, 1746
 - put, 1751–1756
- std::num_put< _CharT, _OutIter >, 1744
- std::num_punct
 - ~num_punct, 1760
 - char_type, 1759
 - decimal_point, 1760
 - do_decimal_point, 1760
 - do_falsename, 1760
 - do_grouping, 1761
 - do_thousands_sep, 1761
 - do_truename, 1761
 - falsename, 1762
 - grouping, 1762
 - id, 1763
 - num_punct, 1759
 - string_type, 1759
 - thousands_sep, 1762
 - truename, 1763
- std::num_punct< _CharT >, 1757
- std::num_punct_byname
 - decimal_point, 1765
 - do_decimal_point, 1765
 - do_falsename, 1766
 - do_grouping, 1766
 - do_thousands_sep, 1766
 - do_truename, 1767
 - falsename, 1767
 - grouping, 1767
 - id, 1769
 - thousands_sep, 1768
 - truename, 1768
- std::num_punct_byname< _CharT >, 1764
- std::ostream_iterator
 - char_type, 1770
 - difference_type, 1770
 - iterator_category, 1770
 - operator=, 1772
 - ostream_iterator, 1771, 1772
 - ostream_type, 1770
 - pointer, 1771
 - reference, 1771
 - traits_type, 1771
 - value_type, 1771
- std::ostream_iterator< _Tp, _CharT, _Traits >, 1769
- std::ostreambuf_iterator
 - char_type, 1773
 - difference_type, 1773

- failed, [1775](#)
- iterator_category, [1774](#)
- operator*, [1775](#)
- operator++, [1775](#)
- operator=, [1775](#)
- ostream_type, [1774](#)
- ostreambuf_iterator, [1775](#)
- pointer, [1774](#)
- reference, [1774](#)
- streambuf_type, [1774](#)
- traits_type, [1774](#)
- value_type, [1774](#)
- std::ostreambuf_iterator< _CharT, _Traits >, [1772](#)
- std::output_iterator_tag, [1776](#)
- std::owner_less< _Tp >, [1776](#)
- std::owner_less< shared_ptr< _Tp > >, [1776](#)
 - first_argument_type, [1777](#)
 - result_type, [1777](#)
 - second_argument_type, [1777](#)
- std::owner_less< weak_ptr< _Tp > >, [1778](#)
 - first_argument_type, [1778](#)
 - result_type, [1778](#)
 - second_argument_type, [1778](#)
- std::pair
 - first, [1781](#)
 - pair, [1780](#)
 - second, [1781](#)
 - second_type, [1780](#)
- std::pair< _T1, _T2 >, [1779](#)
- std::piecewise_constant_distribution
 - densities, [1783](#)
 - intervals, [1783](#)
 - max, [1783](#)
 - min, [1783](#)
 - operator<<, [1784](#)
 - operator>>, [1784](#)
 - operator(), [1783](#)
 - operator==, [1784](#)
 - param, [1783](#)
 - reset, [1784](#)
 - result_type, [1783](#)
- std::piecewise_constant_distribution< _RealType >, [1781](#)
- std::piecewise_constant_distribution< _RealType >↔
 - ::param_type, [1785](#)
- std::piecewise_construct_t, [1786](#)
- std::piecewise_linear_distribution
 - densities, [1787](#)
 - intervals, [1787](#)
 - max, [1787](#)
 - min, [1788](#)
 - operator<<, [1789](#)
 - operator>>, [1789](#)
 - operator(), [1788](#)
 - operator==, [1789](#)
 - param, [1788](#)
 - reset, [1788](#)
 - result_type, [1787](#)
- std::piecewise_linear_distribution< _RealType >, [1786](#)
- std::piecewise_linear_distribution< _RealType >↔
 - ::param_type, [1790](#)
- std::plus
 - first_argument_type, [1791](#)
 - result_type, [1791](#)
 - second_argument_type, [1792](#)
- std::plus< _Tp >, [1791](#)
- std::pointer_to_binary_function
 - first_argument_type, [1793](#)
 - result_type, [1793](#)
 - second_argument_type, [1793](#)
- std::pointer_to_binary_function< _Arg1, _Arg2, _Result >, [1792](#)
- std::pointer_to_unary_function
 - argument_type, [1795](#)
 - result_type, [1795](#)
- std::pointer_to_unary_function< _Arg, _Result >, [1794](#)
- std::pointer_traits
 - difference_type, [1796](#)
 - element_type, [1796](#)
 - pointer, [1796](#)
- std::pointer_traits< _Ptr >, [1795](#)
- std::pointer_traits< _Tp * >, [1796](#)
 - difference_type, [1797](#)
 - element_type, [1797](#)
 - pointer, [1797](#)
 - pointer_to, [1797](#)
- std::poisson_distribution
 - max, [1799](#)
 - mean, [1799](#)
 - min, [1799](#)
 - operator<<, [1800](#)
 - operator>>, [1800](#)
 - operator(), [1799](#)
 - operator==, [1800](#)
 - param, [1799](#)
 - reset, [1800](#)
 - result_type, [1799](#)
- std::poisson_distribution< _IntType >, [1798](#)
- std::poisson_distribution< _IntType >::param_type, [1801](#)
- std::priority_queue
 - empty, [1804](#)
 - pop, [1804](#)
 - priority_queue, [1803](#)
 - push, [1804](#)
 - size, [1805](#)
 - top, [1805](#)
- std::priority_queue< _Tp, _Sequence, _Compare >, [1802](#)
- std::queue
 - back, [1807](#)

- c, 1808
- empty, 1807
- front, 1807
- pop, 1807
- push, 1807
- queue, 1806
- size, 1808
- std::queue< _Tp, _Sequence >, 1805
- std::random_access_iterator_tag, 1809
- std::random_device, 1809
 - result_type, 1810
- std::raw_storage_iterator
 - difference_type, 1811
 - iterator_category, 1811
 - pointer, 1811
 - reference, 1812
 - value_type, 1812
- std::raw_storage_iterator< _OutputIterator, _Tp >, 1810
- std::regex_constants, 447
 - __match_flag, 449
 - __syntax_option, 449
 - awk, 451
 - basic, 451
 - collate, 450
 - ECMAScript, 450
 - egrep, 451
 - error_backref, 451
 - error_badbrace, 451
 - error_badrepeat, 451
 - error_brace, 451
 - error_brack, 451
 - error_collate, 451
 - error_complexity, 452
 - error_ctype, 452
 - error_escape, 452
 - error_paren, 452
 - error_range, 452
 - error_space, 452
 - error_stack, 452
 - error_type, 449
 - extended, 451
 - format_default, 450
 - format_first_only, 450
 - format_no_copy, 450
 - format_sed, 450
 - grep, 451
 - icase, 450
 - match_any, 449
 - match_continuous, 449
 - match_default, 449
 - match_flag_type, 449
 - match_not_bol, 449
 - match_not_bow, 449
 - match_not_eol, 449
 - match_not_eow, 449
 - match_not_null, 449
 - match_prev_avail, 449
 - nosubs, 450
 - operator^, 453
 - operator^=, 453, 454
 - operator&, 452
 - operator&=, 453
 - operator|, 454
 - operator|=, 454, 455
 - operator~, 455
 - optimize, 450
 - syntax_option_type, 450
- std::regex_error, 1812
 - code, 1813
 - regex_error, 1812
- std::regex_iterator
 - operator!=, 1815
 - operator*, 1815
 - operator++, 1815
 - operator->, 1815
 - operator=, 1815
 - operator==, 1815
 - regex_iterator, 1814
- std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >, 1813
- std::regex_token_iterator
 - operator!=, 1819
 - operator*, 1819
 - operator++, 1819
 - operator->, 1819
 - operator=, 1820
 - operator==, 1820
 - regex_token_iterator, 1817–1819
- std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >, 1816
- std::regex_traits
 - getloc, 1821
 - imbue, 1821
 - isctype, 1822
 - length, 1822
 - lookup_classname, 1823
 - lookup_collatename, 1823
 - regex_traits, 1821
 - transform, 1824
 - transform_primary, 1824
 - translate, 1825
 - translate_nocase, 1825
 - value, 1826
- std::regex_traits< _Ch_type >, 1820
- std::rel_ops, 455
 - operator!=, 456
 - operator<=, 456
 - operator>, 456

- operator>=, [457](#)
- std::reverse_iterator
 - base, [1829](#)
 - iterator_category, [1828](#)
 - operator*, [1829](#)
 - operator+, [1829](#)
 - operator++, [1829](#)
 - operator+=, [1830](#)
 - operator-, [1830](#)
 - operator->, [1831](#)
 - operator--, [1830](#)
 - operator-=, [1831](#)
 - operator[], [1831](#)
 - reverse_iterator, [1828](#)
 - value_type, [1828](#)
- std::reverse_iterator<_Iterator>, [1826](#)
- std::seed_seq, [1832](#)
 - result_type, [1832](#)
 - seed_seq, [1832](#)
- std::set
 - allocator_type, [1835](#)
 - begin, [1840](#)
 - cbegin, [1840](#)
 - cend, [1840](#)
 - clear, [1840](#)
 - const_iterator, [1835](#)
 - const_pointer, [1835](#)
 - const_reference, [1835](#)
 - const_reverse_iterator, [1835](#)
 - count, [1841](#)
 - crbegin, [1841](#)
 - crend, [1841](#)
 - difference_type, [1836](#)
 - emplace, [1841](#)
 - emplace_hint, [1842](#)
 - empty, [1842](#)
 - end, [1842](#)
 - equal_range, [1843](#)
 - erase, [1844](#), [1845](#)
 - find, [1845](#), [1846](#)
 - get_allocator, [1846](#)
 - insert, [1846](#), [1847](#)
 - iterator, [1836](#)
 - key_comp, [1848](#)
 - key_compare, [1836](#)
 - key_type, [1836](#)
 - lower_bound, [1848](#)
 - max_size, [1849](#)
 - operator=, [1849](#), [1850](#)
 - pointer, [1836](#)
 - rbegin, [1850](#)
 - reference, [1836](#)
 - rend, [1850](#)
 - reverse_iterator, [1836](#)
 - set, [1837](#)–[1840](#)
 - size, [1850](#)
 - size_type, [1837](#)
 - swap, [1850](#)
 - upper_bound, [1851](#)
 - value_comp, [1851](#)
 - value_compare, [1837](#)
 - value_type, [1837](#)
- std::set<_Key, _Compare, _Alloc>, [1833](#)
- std::shared_ptr
 - allocate_shared, [1859](#)
 - shared_ptr, [1854](#)–[1859](#)
- std::shared_ptr<_Tp>, [1852](#)
- std::shuffle_order_engine
 - base, [1863](#)
 - discard, [1863](#)
 - max, [1863](#)
 - min, [1863](#)
 - operator<<, [1864](#)
 - operator>>, [1864](#)
 - operator(), [1863](#)
 - operator==, [1864](#)
 - result_type, [1861](#)
 - seed, [1863](#)
 - shuffle_order_engine, [1861](#), [1862](#)
- std::shuffle_order_engine<_RandomNumberEngine, _k>, [1860](#)
- std::slice, [1865](#)
- std::slice_array<_Tp>, [1866](#)
- std::stack
 - empty, [1869](#)
 - pop, [1869](#)
 - push, [1869](#)
 - size, [1869](#)
 - stack, [1868](#)
 - top, [1869](#)
- std::stack<_Tp, _Sequence>, [1867](#)
- std::student_t_distribution
 - max, [1871](#)
 - min, [1871](#)
 - operator<<, [1872](#)
 - operator>>, [1873](#)
 - operator(), [1872](#)
 - operator==, [1873](#)
 - param, [1872](#)
 - reset, [1872](#)
 - result_type, [1871](#)
- std::student_t_distribution<_RealType>, [1870](#)
- std::student_t_distribution<_RealType>::param_type, [1873](#)
- std::sub_match
 - compare, [1875](#), [1876](#)
 - first, [1877](#)
 - length, [1877](#)

- operator string_type, 1877
- second, 1877
- second_type, 1875
- str, 1877
- std::sub_match<_Bilter >, 1874
- std::time_base, 1878
- std::time_get
 - ~time_get, 1881
 - char_type, 1881
 - date_order, 1881
 - do_date_order, 1882
 - do_get_date, 1882
 - do_get_monthname, 1882
 - do_get_time, 1883
 - do_get_weekday, 1883
 - do_get_year, 1884
 - get_date, 1885
 - get_monthname, 1885
 - get_time, 1886
 - get_weekday, 1886
 - get_year, 1887
 - id, 1887
 - iter_type, 1881
 - time_get, 1881
- std::time_get<_CharT, _InIter >, 1879
- std::time_get_byname
 - date_order, 1890
 - do_date_order, 1890
 - do_get_date, 1890
 - do_get_monthname, 1891
 - do_get_time, 1891
 - do_get_weekday, 1892
 - do_get_year, 1892
 - get_date, 1893
 - get_monthname, 1893
 - get_time, 1894
 - get_weekday, 1894
 - get_year, 1895
 - id, 1895
- std::time_get_byname<_CharT, _InIter >, 1888
- std::time_put
 - ~time_put, 1898
 - char_type, 1897
 - do_put, 1898
 - id, 1900
 - iter_type, 1897
 - put, 1898, 1900
 - time_put, 1897
- std::time_put<_CharT, _OutIter >, 1896
- std::time_put_byname
 - do_put, 1902
 - id, 1904
 - put, 1902, 1903
- std::time_put_byname<_CharT, _OutIter >, 1901
- std::tr1, 457
- std::tr1::__detail, 458
- std::tr2, 458
- std::tr2::__detail, 458
- std::unary_function
 - argument_type, 1905
 - result_type, 1905
- std::unary_function<_Arg, _Result >, 1904
- std::unary_negate
 - argument_type, 1906
 - result_type, 1906
- std::unary_negate<_Predicate >, 1905
- std::uniform_int_distribution
 - max, 1908
 - min, 1908
 - operator(), 1908
 - operator==, 1909
 - param, 1908
 - reset, 1909
 - result_type, 1907
 - uniform_int_distribution, 1908
- std::uniform_int_distribution<_IntType >, 1906
- std::uniform_int_distribution<_IntType >::param_type, 1909
- std::uniform_real_distribution
 - max, 1911
 - min, 1912
 - operator(), 1912
 - operator==, 1913
 - param, 1912
 - reset, 1912
 - result_type, 1911
 - uniform_real_distribution, 1911
- std::uniform_real_distribution<_RealType >, 1910
- std::uniform_real_distribution<_RealType >::param_type, 1913
- std::unique_ptr
 - ~unique_ptr, 1916
 - get, 1917
 - get_deleter, 1917
 - operator bool, 1917
 - operator*, 1917
 - operator->, 1917
 - operator=, 1917, 1918
 - release, 1918
 - reset, 1918
 - swap, 1919
 - unique_ptr, 1915, 1916
- std::unique_ptr<_Tp, _Dp >, 1914
- std::unique_ptr<_Tp[], _Dp >, 1919
 - ~unique_ptr, 1921
 - get, 1922
 - get_deleter, 1922
 - operator bool, 1922

- operator=, 1922, 1923
- operator[], 1923
- release, 1923
- reset, 1923
- swap, 1924
- unique_ptr, 1920, 1921
- std::unordered_map
 - allocator_type, 1927
 - at, 1931, 1932
 - begin, 1932, 1933
 - bucket_count, 1933
 - cbegin, 1933
 - cend, 1934
 - clear, 1934
 - const_iterator, 1927
 - const_local_iterator, 1927
 - const_pointer, 1927
 - const_reference, 1927
 - count, 1934
 - difference_type, 1928
 - emplace, 1935
 - emplace_hint, 1935
 - empty, 1936
 - end, 1936, 1937
 - equal_range, 1937
 - erase, 1938, 1939
 - find, 1940
 - get_allocator, 1940
 - hash_function, 1941
 - hasher, 1928
 - insert, 1941–1943
 - iterator, 1928
 - key_eq, 1944
 - key_equal, 1928
 - key_type, 1928
 - load_factor, 1944
 - local_iterator, 1928
 - mapped_type, 1929
 - max_bucket_count, 1944
 - max_load_factor, 1944
 - max_size, 1944
 - operator=, 1945
 - operator[], 1945, 1946
 - pointer, 1929
 - reference, 1929
 - rehash, 1946
 - reserve, 1947
 - size, 1947
 - size_type, 1929
 - swap, 1947
 - unordered_map, 1929–1931
 - value_type, 1929
- std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 1924
- std::unordered_multimap
 - allocator_type, 1950
 - begin, 1955
 - bucket_count, 1956
 - cbegin, 1956
 - cend, 1956, 1957
 - clear, 1957
 - const_iterator, 1950
 - const_local_iterator, 1951
 - const_pointer, 1951
 - const_reference, 1951
 - count, 1957
 - difference_type, 1951
 - emplace, 1958
 - emplace_hint, 1958
 - empty, 1959
 - end, 1959
 - equal_range, 1960
 - erase, 1960–1962
 - find, 1962, 1963
 - get_allocator, 1963
 - hash_function, 1963
 - hasher, 1951
 - insert, 1963–1966
 - iterator, 1951
 - key_eq, 1966
 - key_equal, 1952
 - key_type, 1952
 - load_factor, 1966
 - local_iterator, 1952
 - mapped_type, 1952
 - max_bucket_count, 1966
 - max_load_factor, 1966
 - max_size, 1968
 - operator=, 1968
 - pointer, 1952
 - reference, 1952
 - rehash, 1968
 - reserve, 1969
 - size, 1969
 - size_type, 1953
 - swap, 1969
 - unordered_multimap, 1953, 1954
 - value_type, 1953
- std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 1948
- std::unordered_multiset
 - allocator_type, 1973
 - begin, 1977, 1978
 - bucket_count, 1978
 - cbegin, 1978
 - cend, 1979
 - clear, 1979
 - const_iterator, 1973

- const_local_iterator, 1973
- const_pointer, 1973
- const_reference, 1973
- count, 1979
- difference_type, 1973
- emplace, 1980
- emplace_hint, 1980
- empty, 1981
- end, 1981
- equal_range, 1982
- erase, 1983, 1984
- find, 1984, 1985
- get_allocator, 1985
- hash_function, 1985
- hasher, 1974
- insert, 1985, 1987–1989
- iterator, 1974
- key_eq, 1989
- key_equal, 1974
- key_type, 1974
- load_factor, 1989
- local_iterator, 1974
- max_bucket_count, 1989
- max_load_factor, 1989
- max_size, 1991
- operator=, 1991
- pointer, 1974
- reference, 1975
- rehash, 1991
- reserve, 1992
- size, 1992
- size_type, 1975
- swap, 1992
- unordered_multiset, 1975–1977
- value_type, 1975
- std::unordered_multiset< _Value, _Hash, _Pred, _Alloc
>, 1970
- std::unordered_set
 - allocator_type, 1996
 - begin, 2000, 2001
 - bucket_count, 2001
 - cbegin, 2001
 - cend, 2002
 - clear, 2002
 - const_iterator, 1996
 - const_local_iterator, 1996
 - const_pointer, 1996
 - const_reference, 1996
 - count, 2002
 - difference_type, 1996
 - emplace, 2003
 - emplace_hint, 2003
 - empty, 2004
 - end, 2004, 2005
 - equal_range, 2005
 - erase, 2006, 2007
 - find, 2008
 - get_allocator, 2008
 - hash_function, 2009
 - hasher, 1997
 - insert, 2009–2011
 - iterator, 1997
 - key_eq, 2011
 - key_equal, 1997
 - key_type, 1997
 - load_factor, 2012
 - local_iterator, 1997
 - max_bucket_count, 2012
 - max_load_factor, 2012
 - max_size, 2012
 - operator=, 2013
 - pointer, 1997
 - reference, 1998
 - rehash, 2013
 - reserve, 2014
 - size, 2014
 - size_type, 1998
 - swap, 2014
 - unordered_set, 1998–2000
 - value_type, 1998
- std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 1993
- std::uses_allocator< _Tp, _Alloc >, 2015
- std::vector
 - _M_allocate_and_copy, 2021
 - _M_range_check, 2021
 - ~vector, 2021
 - assign, 2022
 - at, 2023
 - back, 2024
 - begin, 2024
 - capacity, 2024
 - cbegin, 2025
 - cend, 2025
 - clear, 2025
 - crbegin, 2025
 - crend, 2025
 - data, 2025
 - emplace, 2025
 - empty, 2026
 - end, 2026
 - erase, 2026, 2027
 - front, 2027
 - insert, 2028, 2029
 - max_size, 2030
 - operator=, 2030
 - operator[], 2031
 - pop_back, 2032

- push_back, 2032
- rbegin, 2032
- rend, 2032, 2033
- reserve, 2033
- resize, 2033, 2034
- shrink_to_fit, 2034
- size, 2034
- swap, 2034
- vector, 2018–2021
- std::vector< _Tp, _Alloc >, 2015
- std::vector< bool, _Alloc >, 2035
- std::weak_ptr< _Tp >, 2038
- std::weibull_distribution
 - a, 2041
 - b, 2041
 - max, 2041
 - min, 2041
 - operator(), 2041
 - operator==, 2042
 - param, 2041
 - reset, 2042
 - result_type, 2040
- std::weibull_distribution< _RealType >, 2039
- std::weibull_distribution< _RealType >::param_type, 2042
- stdc++.h, 2253
- stddev
 - std::normal_distribution, 1724
- stdio_filebuf
 - __gnu_cxx::stdio_filebuf, 566
- stdio_filebuf.h, 2253
- stdio_sync_filebuf.h, 2253
- stdtr1c++.h, 2254
- stl_algo.h, 2254
- stl_algobase.h, 2263
- stl_bvector.h, 2266
- stl_construct.h, 2266
- stl_deque.h, 2267
 - _GLIBCXX_DEQUE_BUF_SIZE, 2269
- stl_function.h, 2270
- stl_heap.h, 2271
- stl_iterator.h, 2272
- stl_iterator_base_funcs.h, 2275
- stl_iterator_base_types.h, 2276
- stl_list.h, 2277
- stl_map.h, 2278
- stl_multimap.h, 2278
- stl_multiset.h, 2279
- stl_numeric.h, 2280
- stl_pair.h, 2281
- stl_queue.h, 2282
- stl_raw_storage_iter.h, 2283
- stl_relops.h, 2283
- stl_set.h, 2284
- stl_stack.h, 2284
- stl_tempbuf.h, 2285
- stl_tree.h, 2286
- stl_uninitialized.h, 2287
- stl_vector.h, 2288
- str
 - std::match_results, 1626
 - std::sub_match, 1877
- stream_iterator.h, 2289
- streambuf_iterator.h, 2290
- streambuf_type
 - std::istreambuf_iterator, 1543
 - std::ostreambuf_iterator, 1774
- streamoff
 - std, 369
- streampos
 - std, 369
- streamsize
 - std, 369
- stride
 - Numeric_arrays, 121
- string
 - Strings, 203
- string_conversions.h, 2290
- string_type
 - std::collate, 1315
 - std::collate_byname, 1321
 - std::messages, 1641
 - std::money_get, 1651
 - std::money_put, 1655
 - std::moneypunct, 1661
 - std::numpunct, 1759
- stringfwd.h, 2291
- Strings, 203
 - string, 203
 - u16string, 203
 - u32string, 203
 - wstring, 203
- substr
 - __gnu_cxx::__versa_string, 535
 - std::basic_string, 1262
- suffix
 - std::match_results, 1628
- swap
 - __gnu_cxx, 235
 - __gnu_cxx::__versa_string, 535
 - __gnu_pbds::sample_probe_fn, 966
 - __gnu_pbds::sample_range_hashing, 967
 - __gnu_pbds::sample_ranged_hash_fn, 968
 - __gnu_pbds::sample_resize_policy, 971
 - __gnu_pbds::sample_resize_trigger, 974
 - __gnu_pbds::sample_size_policy, 976
 - __gnu_pbds::sample_update_policy, 979
 - Regular Expressions, 174

- std, [420–422](#)
- std::basic_regex, [1212](#)
- std::basic_string, [1263](#)
- std::deque, [1426](#)
- std::forward_list, [1470](#)
- std::list, [1574](#)
- std::map, [1614](#)
- std::match_results, [1628](#)
- std::multimap, [1696](#)
- std::multiset, [1714](#)
- std::set, [1850](#)
- std::unique_ptr, [1919](#)
- std::unique_ptr< _Tp[], _Dp >, [1924](#)
- std::unordered_map, [1947](#)
- std::unordered_multimap, [1969](#)
- std::unordered_multiset, [1992](#)
- std::unordered_set, [2014](#)
- std::vector, [2034](#)
- Utilities, [216](#), [217](#)
- swap_ranges
 - Mutating, [86](#)
- sync_with_stdio
 - std::basic_ios, [1194](#)
 - std::ios_base, [1531](#)
- syntax_option_type
 - std::regex_constants, [450](#)
- synth_access_traits
 - __gnu_pbds::detail::trie_traits< Key, Mapped, _←
ATraits, Node_Update, pat_trie_tag, _Alloc >, [925](#)
 - __gnu_pbds::detail::trie_traits< Key, null_type, _←
ATraits, Node_Update, pat_trie_tag, _Alloc >, [926](#)
- synth_access_traits.hpp, [2291](#)
- t
 - std::binomial_distribution, [1276](#)
- TLB_size
 - __gnu_parallel::_Settings, [723](#)
- table
 - std::ctype< char >, [1353](#)
 - std::ctype_byname< char >, [1398](#)
- table_size
 - std::ctype< char >, [1356](#)
 - std::ctype_byname< char >, [1401](#)
- tag_and_trait.hpp, [2292](#)
- Tags, [204](#)
 - trivial_iterator_difference_type, [204](#)
- tags.h, [2293](#)
- tgmath.h, [2294](#)
- thin_heap.hpp, [2294](#)
- thousands_sep
 - std::moneypunct, [1668](#)
 - std::moneypunct_byname, [1677](#)
 - std::numpunct, [1762](#)
 - std::numpunct_byname, [1768](#)
- throw_allocator.h, [2295](#)
- throw_with_nested
 - Exceptions, [46](#)
- tie
 - std::basic_ios, [1195](#)
- time
 - std::locale, [1582](#)
- time_get
 - std::time_get, [1881](#)
- time_members.h, [2296](#)
- time_put
 - std::time_put, [1897](#)
- tolower
 - std, [422](#)
 - std::__ctype_abstract_base, [1053](#), [1054](#)
 - std::ctype, [1340](#)
 - std::ctype< char >, [1353](#), [1354](#)
 - std::ctype< wchar_t >, [1369](#), [1370](#)
 - std::ctype_byname, [1384](#), [1386](#)
 - std::ctype_byname< char >, [1398](#)
- top
 - std::priority_queue, [1805](#)
 - std::stack, [1869](#)
- toupper
 - std, [422](#)
 - std::__ctype_abstract_base, [1054](#)
 - std::ctype, [1341](#)
 - std::ctype< char >, [1354](#), [1355](#)
 - std::ctype< wchar_t >, [1370](#), [1371](#)
 - std::ctype_byname, [1386](#), [1387](#)
 - std::ctype_byname< char >, [1399](#)
- trace_fn_imps.hpp, [2297](#), [2298](#)
- Traits, [205](#)
- traits.hpp, [2298–2300](#)
- traits_type
 - std::basic_ios, [1183](#)
 - std::istreambuf_iterator, [1543](#)
 - std::ostream_iterator, [1771](#)
 - std::ostreambuf_iterator, [1774](#)
- transform
 - Mutating, [86](#), [87](#)
 - std::collate, [1318](#)
 - std::collate_byname, [1323](#)
 - std::regex_traits, [1824](#)
- transform_minimal_n
 - __gnu_parallel::_Settings, [723](#)
- transform_primary
 - std::regex_traits, [1824](#)
- translate
 - std::regex_traits, [1825](#)
- translate_nocase
 - std::regex_traits, [1825](#)

- tree
 - `__gnu_pbds::tree`, [985](#)
- `tree_policy.hpp`, [2300](#)
- `tree_trace_base.hpp`, [2301](#)
- trie
 - `__gnu_pbds::trie`, [990](#)
- `trie_policy.hpp`, [2301](#)
- `trie_policy_base.hpp`, [2301](#)
- `trie_string_access_traits_imp.hpp`, [2302](#)
- trivial_iterator_difference_type
 - Tags, [204](#)
- truename
 - `std::numpunct`, [1763](#)
 - `std::numpunct_byname`, [1768](#)
- trunc
 - `std::basic_ios`, [1202](#)
 - `std::ios_base`, [1537](#)
- type
 - `__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, cc_hash_tag, Policy_TI >`, [809](#)
 - `__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, gp_hash_tag, Policy_TI >`, [810](#)
 - `__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, list_update_tag, Policy_TI >`, [811](#)
 - `__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, ov_tree_tag, Policy_TI >`, [811](#)
 - `__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, rb_tree_tag, Policy_TI >`, [812](#)
 - `__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, splay_tree_tag, Policy_TI >`, [813](#)
 - `__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, cc_hash_tag, Policy_TI >`, [814](#)
 - `__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, gp_hash_tag, Policy_TI >`, [814](#)
 - `__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, list_update_tag, Policy_TI >`, [815](#)
 - `__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, ov_tree_tag, Policy_TI >`, [816](#)
 - `__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, pat_trie_tag, Policy_TI >`, [816](#)
 - `__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, splay_tree_tag, Policy_TI >`, [817](#)
 - `__gnu_pbds::detail::container_base_dispatch< _V, Tp, Cmp_Fn, _Alloc, binary_heap_tag, null_type >`, [806](#)
 - `__gnu_pbds::detail::container_base_dispatch< _V, Tp, Cmp_Fn, _Alloc, binomial_heap_tag, null_type >`, [807](#)
 - `__gnu_pbds::detail::container_base_dispatch< _V, Tp, Cmp_Fn, _Alloc, rc_binomial_heap_tag, null_type >`, [808](#)
 - `__gnu_pbds::detail::container_base_dispatch< _V, Tp, Cmp_Fn, _Alloc, thin_heap_tag, null_type >`, [809](#)
 - `__gnu_pbds::detail::default_comb_hash_fn`, [818](#)
 - `__gnu_pbds::detail::default_eq_fn`, [819](#)
 - `__gnu_pbds::detail::default_hash_fn`, [819](#)
 - `__gnu_pbds::detail::default_probe_fn`, [820](#)
 - `__gnu_pbds::detail::default_resize_policy`, [820](#)
 - `__gnu_pbds::detail::default_trie_access_traits< std::basic_string< Char, Char_Traits, std::allocator< char > > >`, [821](#)
 - `__gnu_pbds::detail::default_update_policy`, [822](#)
 - `__gnu_pbds::detail::entry_cmp< _V, Tp, Cmp_Fn, _Alloc, true >`, [824](#)
- `type_traits.h`, [2302](#)
- `type_utils.hpp`, [2302](#)
- `typelist.h`, [2303](#)
- `types.h`, [2304](#)
- `types_traits.hpp`, [2305](#)
- u16streampos
 - std, [369](#)
- u16string
 - Strings, [203](#)
- u32streampos
 - std, [369](#)
- u32string
 - Strings, [203](#)
- Uniform Distributions, [207](#)
 - operator!=, [207](#)
 - operator<<, [208](#)
 - operator>>, [208](#), [209](#)
- uniform_int_distribution
 - std::uniform_int_distribution, [1908](#)
- uniform_real_distribution
 - std::uniform_real_distribution, [1911](#)
- uninitialized_copy
 - std, [422](#)
- uninitialized_copy_n
 - std, [423](#)
- uninitialized_fill
 - std, [423](#)
- uninitialized_fill_n
 - std, [424](#)
- unique
 - Mutating, [87](#), [88](#)
 - std::forward_list, [1470](#), [1471](#)
 - std::list, [1574](#)
- unique_copy

- Mutating, [88](#), [89](#)
- unique_copy.h, [2305](#)
- unique_copy_minimal_n
 - __gnu_parallel::_Settings, [723](#)
- unique_ptr
 - std::unique_ptr, [1915](#), [1916](#)
 - std::unique_ptr< _Tp[], _Dp >, [1920](#), [1921](#)
- unique_ptr.h, [2306](#)
- unitbuf
 - std, [424](#)
 - std::basic_ios, [1202](#)
 - std::ios_base, [1537](#)
- Unordered Associative, [210](#)
- unordered_base.h, [2307](#)
- unordered_map
 - std::unordered_map, [1929–1931](#)
- unordered_map.h, [2308](#)
- unordered_multimap
 - std::unordered_multimap, [1953](#), [1954](#)
- unordered_multiset
 - std::unordered_multiset, [1975–1977](#)
- unordered_set
 - std::unordered_set, [1998–2000](#)
- unordered_set.h, [2309](#)
- unsetf
 - std::basic_ios, [1195](#)
 - std::ios_base, [1531](#)
- unshift
 - std::__codecvt_abstract_base, [1041](#)
 - std::codecvt, [1293](#)
 - std::codecvt< _InternT, _ExternT, encoding_state >, [1298](#)
 - std::codecvt< char, char, mbstate_t >, [1302](#)
 - std::codecvt< wchar_t, char, mbstate_t >, [1307](#)
 - std::codecvt_byname, [1312](#)
- update_fn_imps.hpp, [2310](#)
- upper_bound
 - Binary Search, [32](#), [33](#)
 - std::map, [1614](#), [1615](#)
 - std::multimap, [1696](#)
 - std::multiset, [1714](#), [1715](#)
 - std::set, [1851](#)
- uppercase
 - std, [424](#)
 - std::basic_ios, [1202](#)
 - std::ios_base, [1537](#)
- Utilities, [211](#)
 - __addressof, [212](#)
 - addressof, [212](#)
 - forward, [213](#)
 - make_pair, [213](#)
 - move, [214](#)
 - move_if_noexcept, [215](#)
 - operator!=, [215](#)
 - operator<, [215](#)
 - operator<=, [215](#)
 - operator>, [216](#)
 - operator>=, [216](#)
 - operator==, [215](#)
 - piecewise_construct, [217](#)
 - swap, [216](#), [217](#)
- valarray_after.h, [2310](#)
- valarray_array.h, [2320](#)
- valarray_before.h, [2328](#)
- valid_prefix
 - __gnu_pbds::detail::pat_trie_base::_Node_citer, [877](#)
 - __gnu_pbds::detail::pat_trie_base::_Node_iter, [880](#)
- value
 - std::regex_traits, [1826](#)
- value_comp
 - std::map, [1615](#)
 - std::multimap, [1697](#)
 - std::multiset, [1715](#)
 - std::set, [1851](#)
- value_compare
 - std::set, [1837](#)
- value_type
 - __gnu_pbds::detail::bin_search_tree_const_node_↵ it_, [775](#)
 - __gnu_pbds::detail::bin_search_tree_node_it_, [780](#)
 - __gnu_pbds::detail::binary_heap_const_iterator_, [788](#)
 - __gnu_pbds::detail::binary_heap_point_const_↵ iterator_, [792](#)
 - __gnu_pbds::detail::left_child_next_sibling_heap_↵ const_iterator_, [838](#)
 - __gnu_pbds::detail::left_child_next_sibling_heap_↵ node_point_const_iterator_, [842](#)
 - const_iterator_, [1022](#)
 - iterator_, [1026](#)
 - point_const_iterator_, [1030](#)
 - point_iterator_, [1033](#)
 - std::allocator_traits, [1158](#)
 - std::allocator_traits< allocator< _Tp > >, [1163](#)
 - std::back_insert_iterator, [1174](#)
 - std::front_insert_iterator, [1476](#)
 - std::insert_iterator, [1520](#)
 - std::istream_iterator, [1540](#)
 - std::istreambuf_iterator, [1543](#)
 - std::iterator, [1546](#)
 - std::ostream_iterator, [1771](#)
 - std::ostreambuf_iterator, [1774](#)
 - std::raw_storage_iterator, [1812](#)
 - std::reverse_iterator, [1828](#)
 - std::set, [1837](#)
 - std::unordered_map, [1929](#)
 - std::unordered_multimap, [1953](#)

- [std::unordered_multiset](#), [1975](#)
 - [std::unordered_set](#), [1998](#)
- [vector](#)
 - [std::vector](#), [2018–2021](#)
- [void_pointer](#)
 - [__gnu_cxx::__alloc_traits](#), [461](#)
 - [std::allocator_traits](#), [1158](#)
 - [std::allocator_traits< allocator< _Tp > >](#), [1163](#)
- [vstring.h](#), [2328](#)
- [vstring_fwd.h](#), [2330](#)
- [vstring_util.h](#), [2331](#)
- [wregex_token_iterator](#)
 - [Regular Expressions](#), [145](#)
- [wsub_match](#)
 - [Regular Expressions](#), [145](#)
- [widen](#)
 - [std::__ctype_abstract_base](#), [1055](#)
 - [std::basic_ios](#), [1196](#)
 - [std::ctype](#), [1342](#)
 - [std::ctype< char >](#), [1355](#), [1356](#)
 - [std::ctype< wchar_t >](#), [1371](#)
 - [std::ctype_byname](#), [1387](#)
 - [std::ctype_byname< char >](#), [1400](#)
- [width](#)
 - [std::basic_ios](#), [1196](#)
 - [std::ios_base](#), [1532](#)
- [workstealing.h](#), [2331](#)
- [wregex](#)
 - [Regular Expressions](#), [146](#)
- [wsregex_token_iterator](#)
 - [Regular Expressions](#), [146](#)
- [wsub_match](#)
 - [Regular Expressions](#), [146](#)
- [wstreampos](#)
 - [std](#), [369](#)
- [wstring](#)
 - [Strings](#), [203](#)
- [xalloc](#)
 - [std::basic_ios](#), [1197](#)
 - [std::ios_base](#), [1532](#)