算法库

算法库提供大量用途的函数（例如查找、排序、计数、操作），它们在元素范围上操作。注意，范围定义为 [`first`, `last`)，其中 `last` 指代要查询或修改的最后元素的*后一个*元素。

受约束算法 (C++20 起)

C++20 在命名空间 `std::ranges` 中提供大多数算法的受约束版本，在这些算法中，范围既可以由迭代器-哨位对，也可以由单个 range 实参指定，还支持投影和成员指针可调用对象。另外，还更改了大多数算法的返回类型，以返回算法执行过程中计算的所有潜在有用信息。

```
std::vector<int> v{7, 1, 4, 0, -1};
std::ranges::sort(v); // 受约束算法
```

执行策略 (C++17 起)

大多数算法拥有接受执行策略的重载。标准库的算法支持几种执行策略，库中提供了相应的执行策略类型和对象。用户可以通过以对应类型的执行策略对象为参数调用并行算法，静态地选择执行策略。

标准库实现（但不是用户）可以定义额外的执行策略作为扩展。以实现定义类型的执行策略对象调用并行算法的语义是由实现定义的。

允许算法的并行版本（除了 `std::for_each` 与 `std::for_each_n`）从范围进行任意的元素复制，只要 `std::is_trivially_copy_constructible_v<T>` 与 `std::is_trivially_destructible_v<T>` 都是 `true`，其中 T 是元素的类型。

在标头 <execution> 定义 在命名空间 std::execution 定义		
sequenced_policy	(C++17)	执行策略类型 (类)
parallel_policy	(C++17)	
parallel_unsequenced_policy	(C++17)	
unsequenced_policy	(C++20)	
seq	(C++17)	全局执行策略对象 (常量)
par	(C++17)	
par_unseq	(C++17)	
unseq	(C++20)	
在命名空间 std 定义		
is_execution_policy	(C++17)	测试类是否表示某种执行策略 (类模板)

功能特性测试宏	值	标准	功能特性
`__cpp_lib_parallel_algorithm`	`201603L`	(C++17)	并行算法
`__cpp_lib_execution`	`201603L`	(C++17)	执行策略
	`201902L`	(C++20)	`std::execution::unsequenced_policy`

不修改序列的操作

批量操作

在标头 <algorithm> 定义	
for_each	应用一元函数对象到范围中元素 (函数模板)
ranges::for_each (C++20)	应用一元函数对象到范围中元素 (算法函数对象)
for_each_n (C++17)	应用函数对象到序列的前 N 个元素 (函数模板)
ranges::for_each_n (C++20)	应用函数对象到序列的前 N 个元素 (算法函数对象)

搜索操作

在标头 <algorithm> 定义

all_of (C++11) **any_of** (C++11) **none_of** (C++11)	检查谓词是否对范围中所有、任一或无元素为 `true` (函数模板)
ranges::all_of (C++20) **ranges::any_of** (C++20) **ranges::none_of** (C++20)	检查谓词是否对范围中所有、任一或无元素为 `true` (算法函数对象)
ranges::contains (C++23) **ranges::contains_subrange** (C++23)	检查范围是否包含给定元素或子范围 (算法函数对象)
find **find_if** **find_if_not** (C++11)	查找首个满足特定条件的元素 (函数模板)
ranges::find (C++20) **ranges::find_if** (C++20) **ranges::find_if_not** (C++20)	查找首个满足特定条件的元素 (算法函数对象)
ranges::find_last (C++23) **ranges::find_last_if** (C++23) **ranges::find_last_if_not** (C++23)	查找最后一个满足特定条件的元素 (算法函数对象)
find_end	查找元素序列在特定范围中最后一次出现 (函数模板)
ranges::find_end (C++20)	查找元素序列在特定范围中最后一次出现 (算法函数对象)
find_first_of	搜索一组元素中任一元素 (函数模板)
ranges::find_first_of (C++20)	搜索一组元素中任一元素 (算法函数对象)
adjacent_find	查找首对相同（或满足给定谓词）的相邻元素 (函数模板)
ranges::adjacent_find (C++20)	查找首对相同（或满足给定谓词）的相邻元素 (算法函数对象)
count **count_if**	返回满足特定条件的元素数目 (函数模板)
ranges::count (C++20) **ranges::count_if** (C++20)	返回满足特定条件的元素数目 (算法函数对象)
mismatch	查找两个范围的首个不同之处 (函数模板)
ranges::mismatch (C++20)	查找两个范围的首个不同之处 (算法函数对象)
equal	判断两组元素是否相同 (函数模板)
ranges::equal (C++20)	判断两组元素是否相同 (算法函数对象)
search	搜索元素范围的首次出现 (函数模板)
ranges::search (C++20)	搜索元素范围的首次出现 (算法函数对象)
search_n	搜索元素在范围中首次连续若干次出现 (函数模板)
ranges::search_n (C++20)	搜索元素在范围中首次连续若干次出现 (算法函数对象)
ranges::starts_with (C++23)	检查一个范围是否始于另一范围 (算法函数对象)
ranges::ends_with (C++23)	检查一个范围是否终于另一范围 (算法函数对象)

折叠操作 (C++23 起)

在标头 <algorithm> 定义

ranges::fold_left (C++23)	左折叠范围中元素 (算法函数对象)
ranges::fold_left_first (C++23)	以首元素为初值左折叠范围中元素 (算法函数对象)
ranges::fold_right (C++23)	右折叠范围中元素 (算法函数对象)
ranges::fold_right_last (C++23)	以末元素为初值右折叠范围中元素 (算法函数对象)

ranges::fold_left_with_iter (C++23)	左折叠范围中元素，并返回 pair（迭代器，值） (算法函数对象)
ranges::fold_left_first_with_iter (C++23)	以首元素为初值左折叠范围中元素，并返回 pair（迭代器，optional） (算法函数对象)

修改序列的操作

复制操作

在标头 <algorithm> 定义

copy **copy_if** (C++11)	复制范围中元素到新位置 (函数模板)
ranges::copy (C++20) **ranges::copy_if** (C++20)	复制范围中元素到新位置 (算法函数对象)
copy_n (C++11)	复制若干元素到新位置 (函数模板)
ranges::copy_n (C++20)	复制若干元素到新位置 (算法函数对象)
copy_backward	从后往前复制范围中元素 (函数模板)
ranges::copy_backward (C++20)	从后往前复制范围中元素 (算法函数对象)
move (C++11)	将范围中元素移到新位置 (函数模板)
ranges::move (C++20)	将范围中元素移到新位置 (算法函数对象)
move_backward (C++11)	从后往前将范围中元素移到新位置 (函数模板)
ranges::move_backward (C++20)	从后往前将范围中元素移到新位置 (算法函数对象)

交换操作

在标头 <algorithm> 定义 (C++11 前)
在标头 <utility> 定义 (C++11 起)
在标头 <string_view> 定义

swap	交换两个对象的值 (函数模板)

在标头 <algorithm> 定义

swap_ranges	交换两个范围的元素 (函数模板)
ranges::swap_ranges (C++20)	交换两个范围的元素 (算法函数对象)
iter_swap	交换两个迭代器所指向的元素 (函数模板)

变换操作

在标头 <algorithm> 定义

transform	应用函数到元素范围，并在目标范围存储结果 (函数模板)
ranges::transform (C++20)	应用函数到元素范围 (算法函数对象)
replace **replace_if**	替换所有满足特定条件的值为另一个值 (函数模板)
ranges::replace (C++20) **ranges::replace_if** (C++20)	替换所有满足特定条件的值为另一个值 (算法函数对象)
replace_copy **replace_copy_if**	复制范围，并将满足特定条件的元素替换为另一个值 (函数模板)
ranges::replace_copy (C++20) **ranges::replace_copy_if** (C++20)	复制范围，并将满足特定条件的元素替换为另一个值 (算法函数对象)

生成操作

在标头 <algorithm> 定义

fill	以复制的方式赋给定值到范围中所有元素 (函数模板)

ranges::fill (C++20)	赋给定值到范围中元素 (算法函数对象)
fill_n	以复制的方式赋给定值到范围中 N 个元素 (函数模板)
ranges::fill_n (C++20)	赋给定值到若干元素 (算法函数对象)
generate	赋连续函数调用结果到范围中所有元素 (函数模板)
ranges::generate (C++20)	将函数结果保存到范围中 (算法函数对象)
generate_n	赋连续函数调用结果到范围中 N 个元素 (函数模板)
ranges::generate_n (C++20)	保存 N 次函数应用的结果 (算法函数对象)

移除操作

在标头 <algorithm> 定义

remove **remove_if**	移除满足特定条件的元素 (函数模板)
ranges::remove (C++20) **ranges::remove_if** (C++20)	移除满足特定条件的元素 (算法函数对象)
remove_copy **remove_copy_if**	复制范围并忽略满足特定条件的元素 (函数模板)
ranges::remove_copy (C++20) **ranges::remove_copy_if** (C++20)	复制范围并忽略满足特定条件的元素 (算法函数对象)
unique	移除范围中连续重复元素 (函数模板)
ranges::unique (C++20)	移除范围中连续重复元素 (算法函数对象)
unique_copy	创建某范围的不含连续重复元素的副本 (函数模板)
ranges::unique_copy (C++20)	创建某范围的不含连续重复元素的副本 (算法函数对象)

顺序变更操作

在标头 <algorithm> 定义

reverse	逆转范围中的元素顺序 (函数模板)
ranges::reverse (C++20)	逆转范围中的元素顺序 (算法函数对象)
reverse_copy	创建范围的逆向副本 (函数模板)
ranges::reverse_copy (C++20)	创建范围的逆向副本 (算法函数对象)
rotate	旋转范围中的元素顺序 (函数模板)
ranges::rotate (C++20)	旋转范围中的元素顺序 (算法函数对象)
rotate_copy	复制并旋转元素范围 (函数模板)
ranges::rotate_copy (C++20)	复制并旋转元素范围 (算法函数对象)
shift_left **shift_right** (C++20)	迁移范围中元素 (函数模板)
ranges::shift_left **ranges::shift_right** (C++23)	迁移范围中元素 (算法函数对象)
random_shuffle (C++17 前) **shuffle** (C++11)	随机重排范围中元素 (函数模板)
ranges::shuffle (C++20)	随机重排范围中元素 (算法函数对象)

采样操作

在标头 <algorithm> 定义

sample (C++17)	从序列中随机选择 N 个元素 (函数模板)
ranges::sample (C++20)	从序列中随机选择 N 个元素 (算法函数对象)

排序和相关操作

要求

部分算法要求由实参表示的序列“已排序”或“已划分”。未满足要求时行为未定义。

序列在满足以下条件时*已按比较器 `comp` 排序*：对于指向序列中的每个迭代器 `iter` 和每个非负整数 `n`，如果 `iter + n`[1] 是一个指向序列中的某个元素的有效迭代器，那么 `comp(*(iter + n), *iter) == false`[1]。 (C++20 前)
序列在满足以下条件时*已按比较器 `comp` 和投影 `proj` 排序*：对于指向序列中的每个迭代器 `iter` 和每个非负整数 `n`，如果 `iter + n`[1] 是一个指向序列中的某个元素的有效迭代器，那么 `bool(std::invoke(comp, std::invoke(proj, *(iter + n)), std::invoke(proj, *iter)))`[1] 是 `false`。 (C++20 起) 序列在满足以下条件时*已按比较器 `comp` 排序*：该序列已按比较器 `comp` 和投影 `std::identity{}`（即投影到自身）排序。

序列 [`start`, `finish`) 在满足以下条件时*已按表达式 `f(e)` 划分*：存在一个整数 `n`，使得对于 [`0`, `std::distance(start, finish)`) 中的所有整数 `i`， `f(*(start + i))`[1] 当且仅当 `i < n` 时是 `true`。

1. ↑ 1.0 1.1 1.2 1.3 1.4 在这里 `iter + n` 只表示 “`iter` 自增 `n` 次后的结果”，而 `iter` 本身不需要是随机访问迭代器。

划分操作

在标头 <algorithm> 定义

is_partitioned (C++11)	判断范围是否已按给定谓词划分 (函数模板)
ranges::is_partitioned (C++20)	判断范围是否已按给定谓词划分 (算法函数对象)
partition	将范围中元素分为两组 (函数模板)
ranges::partition (C++20)	将范围中元素分为两组 (算法函数对象)
partition_copy (C++11)	复制范围并将元素分为两组 (函数模板)
ranges::partition_copy (C++20)	复制范围并将元素分为两组 (算法函数对象)
stable_partition	将元素分为两组，同时保留其相对顺序 (函数模板)
ranges::stable_partition (C++20)	将元素分为两组，同时保留其相对顺序 (算法函数对象)
partition_point (C++11)	定位已划分范围的划分点 (函数模板)
ranges::partition_point (C++20)	定位已划分范围的划分点 (算法函数对象)

排序操作

在标头 <algorithm> 定义

sort	将范围按升序排序 (函数模板)
ranges::sort (C++20)	将范围按升序排序 (算法函数对象)
stable_sort	将范围中元素排序，同时保持相等元之间的顺序 (函数模板)
ranges::stable_sort (C++20)	将范围中元素排序，同时保持相等元之间的顺序 (算法函数对象)
partial_sort	将范围中前 N 个元素排序 (函数模板)
ranges::partial_sort (C++20)	将范围中前 N 个元素排序 (算法函数对象)
partial_sort_copy	复制范围中元素并部分排序 (函数模板)

ranges::partial_sort_copy (C++20)	复制范围中元素并部分排序 (算法函数对象)
is_sorted (C++11)	检查范围是否已按升序排列 (函数模板)
ranges::is_sorted (C++20)	检查范围是否已按升序排列 (算法函数对象)
is_sorted_until (C++11)	找出最大的有序子范围 (函数模板)
ranges::is_sorted_until (C++20)	找出最大的有序子范围 (算法函数对象)
nth_element	将给定范围部分排序，确保其按给定元素划分 (函数模板)
ranges::nth_element (C++20)	将给定范围部分排序，确保其按给定元素划分 (算法函数对象)

二分搜索操作（在已划分范围上）

在标头 <algorithm> 定义

lower_bound	返回首个*不小于*给定值的元素的迭代器 (函数模板)
ranges::lower_bound (C++20)	返回首个*不小于*给定值的元素的迭代器 (算法函数对象)
upper_bound	返回首个*大于*给定值的元素的迭代器 (函数模板)
ranges::upper_bound (C++20)	返回首个*大于*给定值的元素的迭代器 (算法函数对象)
equal_range	返回匹配特定键值的元素范围 (函数模板)
ranges::equal_range (C++20)	返回匹配特定键值的元素范围 (算法函数对象)
binary_search	判断元素是否在偏序范围中 (函数模板)
ranges::binary_search (C++20)	判断元素是否在偏序范围中 (算法函数对象)

集合操作（在已排序范围上）

在标头 <algorithm> 定义

includes	当一个序列是另一个的子序列时返回 `true` (函数模板)
ranges::includes (C++20)	当一个序列是另一个的子序列时返回 `true` (算法函数对象)
set_union	计算两个集合的并集 (函数模板)
ranges::set_union (C++20)	计算两个集合的并集 (算法函数对象)
set_intersection	计算两个集合的交集 (函数模板)
ranges::set_intersection (C++20)	计算两个集合的交集 (算法函数对象)
set_difference	计算两个集合的差集 (函数模板)
ranges::set_difference (C++20)	计算两个集合的差集 (算法函数对象)
set_symmetric_difference	计算两个集合的对称差 (函数模板)
ranges::set_symmetric_difference (C++20)	计算两个集合的对称差 (算法函数对象)

归并操作（在已排序范围上）

在标头 <algorithm> 定义

merge	合并两个有序范围 (函数模板)
ranges::merge (C++20)	合并两个有序范围 (算法函数对象)
inplace_merge	就地合并两个有序范围 (函数模板)

ranges::inplace_merge (C++20)	就地合并两个有序范围 (算法函数对象)

堆操作

随机访问范围 [`first` , `last`) 在满足以下条件时是一个*关于比较器 `comp` 的堆*：对于 (`0` , `last - first`) 中的所有整数 `i` , `bool(comp(first[(i - 1) / 2], first[i]))` 都是 `false` 。 (C++20 前)
随机访问范围 [`first` , `last`) 在满足以下条件时是一个*关于比较器 `comp` 和投影 `proj` 的堆*：对于 (`0` , `last - first`) 中的所有整数 `i` , `bool(std::invoke(comp, std::invoke(proj, first[(i - 1) / 2]),` `std::invoke(proj, first[i]))` 都是 `false` 。 (C++20 起) 随机访问范围 [`first` , `last`) 在满足以下条件时是一个*关于比较器 `comp` 的堆*：该范围是一个关于 `comp` 和 `std::identity{}` （即投影到自身）的堆。

可以通过 `std::make_heap` 和 `ranges::make_heap`(C++20 起) 创建堆。

关于堆的更多性质，可以参考最大堆 。

在标头 <algorithm> 定义

push_heap	添加元素到最大堆 (函数模板)
ranges::push_heap (C++20)	添加元素到最大堆 (算法函数对象)
pop_heap	移除最大堆中最大元 (函数模板)
ranges::pop_heap (C++20)	移除最大堆中最大元 (算法函数对象)
make_heap	从元素范围创建最大堆 (函数模板)
ranges::make_heap (C++20)	从元素范围创建最大堆 (算法函数对象)
sort_heap	将最大堆变成按升序排序的元素范围 (函数模板)
ranges::sort_heap (C++20)	将最大堆变成按升序排序的元素范围 (算法函数对象)
is_heap	检查给定范围是否为最大堆 (函数模板)
ranges::is_heap (C++20)	检查给定范围是否为最大堆 (算法函数对象)
is_heap_until (C++11)	查找能成为最大堆的最大子范围 (函数模板)
ranges::is_heap_until (C++20)	查找能成为最大堆的最大子范围 (算法函数对象)

最小/最大操作

在标头 <algorithm> 定义

max	返回给定值中较大者 (函数模板)
ranges::max (C++20)	返回给定值中较大者 (算法函数对象)
max_element	返回范围中最大元 (函数模板)
ranges::max_element (C++20)	返回范围中最大元 (算法函数对象)
min	返回给定值中较小者 (函数模板)
ranges::min (C++20)	返回给定值中较小者 (算法函数对象)
min_element	返回范围中最小元 (函数模板)
ranges::min_element (C++20)	返回范围中最小元 (算法函数对象)
minmax (C++11)	返回两个元素间的较小者和较大者 (函数模板)

ranges::minmax (C++20)	返回两个元素间的较小者和较大者 (算法函数对象)
minmax_element (C++11)	返回范围中的最小元和最大元 (函数模板)
ranges::minmax_element (C++20)	返回范围中的最小元和最大元 (算法函数对象)
clamp (C++17)	在一对边界值下夹逼一个值 (函数模板)
ranges::clamp (C++20)	在一对边界值下夹逼一个值 (算法函数对象)

字典序比较操作

在标头 <algorithm> 定义

lexicographical_compare	当一个范围字典序小于另一个时返回 `true` (函数模板)
ranges::lexicographical_compare (C++20)	当一个范围字典序小于另一个时返回 `true` (算法函数对象)
lexicographical_compare_three_way (C++20)	三路比较两个范围 (函数模板)

排列操作

在标头 <algorithm> 定义

next_permutation	生成元素范围的下一个字典序更大的排列 (函数模板)
ranges::next_permutation (C++20)	生成元素范围的下一个字典序更大的排列 (算法函数对象)
prev_permutation	生成元素范围的下一个字典序更小的排列 (函数模板)
ranges::prev_permutation (C++20)	生成元素范围的下一个字典序更小的排列 (算法函数对象)
is_permutation (C++11)	判断一个序列是否为另一个序列的排列 (函数模板)
ranges::is_permutation (C++20)	判断一个序列是否为另一个序列的排列 (算法函数对象)

数值运算

在标头 <numeric> 定义

iota (C++11)	从初始值开始连续递增填充范围 (函数模板)
ranges::iota (C++23)	从初始值开始连续递增填充范围 (算法函数对象)
accumulate	求和或折叠范围中元素 (函数模板)
inner_product	计算两个范围中元素的内积 (函数模板)
adjacent_difference	计算范围中相邻元素的差 (函数模板)
partial_sum	计算范围中元素的部分和 (函数模板)
reduce (C++17)	类似 std::accumulate，但不依序执行 (函数模板)
exclusive_scan (C++17)	类似 std::partial_sum，第 i 个和中排除第 i 个输入 (函数模板)
inclusive_scan (C++17)	类似 std::partial_sum，第 i 个和中包含第 i 个输入 (函数模板)
transform_reduce (C++17)	应用可调用对象，然后乱序规约 (函数模板)
transform_exclusive_scan (C++17)	应用可调用对象，然后计算排除扫描 (函数模板)
transform_inclusive_scan (C++17)	应用可调用对象，然后计算包含扫描 (函数模板)

在未初始化内存上的操作

在标头 <memory> 定义

uninitialized_copy	复制范围中对象到未初始化内存 (函数模板)
ranges::uninitialized_copy (C++20)	复制范围中对象到未初始化内存 (算法函数对象)
uninitialized_copy_n (C++11)	复制若干对象到未初始化内存 (函数模板)
ranges::uninitialized_copy_n (C++20)	复制若干对象到未初始化内存 (算法函数对象)
uninitialized_fill	复制一个对象到范围所定义的未初始化内存 (函数模板)
ranges::uninitialized_fill (C++20)	复制一个对象到范围所定义的未初始化内存 (算法函数对象)
uninitialized_fill_n	复制一个对象到起点和数量所定义的未初始化内存 (函数模板)
ranges::uninitialized_fill_n (C++20)	复制一个对象到起点和数量所定义的未初始化内存 (算法函数对象)
uninitialized_move (C++17)	移动范围中对象到未初始化内存 (函数模板)
ranges::uninitialized_move (C++20)	移动范围中对象到未初始化内存 (算法函数对象)
uninitialized_move_n (C++17)	移动若干对象到未初始化内存 (函数模板)
ranges::uninitialized_move_n (C++20)	移动若干对象到未初始化内存 (算法函数对象)
uninitialized_default_construct (C++17)	在范围所定义的未初始化内存中用默认初始化构造对象 (函数模板)
ranges::uninitialized_default_construct (C++20)	在范围所定义的未初始化内存中用默认初始化构造对象 (算法函数对象)
uninitialized_default_construct_n (C++17)	在起点和数量所定义的未初始化内存中用默认初始化构造对象 (函数模板)
ranges::uninitialized_default_construct_n (C++20)	在起点和数量所定义的未初始化内存中用默认初始化构造对象 (算法函数对象)
uninitialized_value_construct (C++17)	在范围所定义的未初始化内存中用值初始化构造对象 (函数模板)
ranges::uninitialized_value_construct (C++20)	在范围所定义的未初始化内存中用值初始化构造对象 (算法函数对象)
uninitialized_value_construct_n (C++17)	在起点和数量所定义的未初始化内存中用值初始化构造对象 (函数模板)
ranges::uninitialized_value_construct_n (C++20)	在起点和数量所定义的未初始化内存中用值初始化构造对象 (算法函数对象)
destroy (C++17)	销毁范围中的对象 (函数模板)
ranges::destroy (C++20)	销毁范围中的对象 (算法函数对象)
destroy_n (C++17)	销毁范围中若干对象 (函数模板)
ranges::destroy_n (C++20)	销毁范围中若干对象 (算法函数对象)
destroy_at (C++17)	销毁给定地址的对象 (函数模板)
ranges::destroy_at (C++20)	销毁给定地址的对象 (算法函数对象)
construct_at (C++20)	在给定地址创建对象 (函数模板)
ranges::construct_at (C++20)	在给定地址创建对象 (算法函数对象)

随机数生成 (C++26 起)

在标头 <random> 定义

ranges::generate_random (C++26)	用来自均匀随机位发生器的随机数填充范围 (算法函数对象)

注解

功能特性测试宏	值	标准	功能特性
__cpp_lib_algorithm_iterator_requirements	202207L	(C++23)	对非范围算法使用范围迭代器
__cpp_lib_clamp	201603L	(C++17)	std::clamp
__cpp_lib_constexpr_algorithms	201806L	(C++20)	constexpr 算法
	202306L	(C++26)	constexpr 稳定排序
__cpp_lib_algorithm_default_value_type	202403L	(C++26)	算法的列表初始化
__cpp_lib_freestanding_algorithm	202311L	(C++26)	<algorithm> 中的独立设施
__cpp_lib_robust_nonmodifying_seq_ops	201304L	(C++14)	使不修改序列的操作更健壮（std::mismatch、std::equal 和 std::is_permutation 的两个范围重载）
__cpp_lib_sample	201603L	(C++17)	std::sample
__cpp_lib_shift	201806L	(C++20)	std::shift_left 与 std::shift_right

C 库

在标头 <cstdlib> 定义

qsort	对未指定类型的元素范围排序 (函数)
bsearch	在未指定类型的数组中搜索元素 (函数)

缺陷报告

下列更改行为的缺陷报告追溯地应用于以前出版的 C++ 标准。

缺陷报告	应用于	出版时的行为	正确行为
LWG 193 (https://cplusplus.github.io/LWG/issue193)	C++98	堆要求 `*first` 是最大的元素	可以有等于 `*first` 的元素
LWG 2150 (https://cplusplus.github.io/LWG/issue2150)	C++98	已排序序列的定义不正确	已改正
LWG 2166 (https://cplusplus.github.io/LWG/issue2166)	C++98	对堆的要求与最大堆　的定义不够接近	改进要求

参阅

算法的 **C 文档**

来自“https://zh.cppreference.com/mwiki/index.php?title=cpp/algorithm&oldid=100887”