

C++
Information
Tutorials
Reference
Articles
Forum
Reference
C library:
Containers:
Input/Output:
Multi-threading:
Other:
<algorithm>
<bitset>
<chrono>
<codecvt>
<complex>
<exception>
<functional>
<initializer_list>
<iterator>
<limits>
<locale>
<memory>
<new>
<numeric>
<random>
<ratio>
<regex>
<stdexcept>
<string>
<system_error>
<tuple>
<typeindex>
<typeinfo>
<type_traits>
<utility>
<valarray>

<algorithm>
adjacent_find
all_of
any_of
binary_search
copy
copy_backward
copy_if
copy_n
count
count_if
equal
equal_range
fill
fill_n
find
find_end
find_first_of
find_if
find_if_not
for_each
generate
generate_n
includes
inplace_merge
is_heap
is_heap_until
is_partitioned
is_permutation
is_sorted
is_sorted_until
iter_swap
lexicographical_compare
lower_bound
make_heap
max
max_element
merge
min
minmax
minmax_element
min_element
mismatch
move
move_backward
next_permutation
none_of
nth_element
partial_sort
partial_sort_copy
partition
partition_copy
partition_point
pop_heap
prev_permutation
push_heap
random_shuffle
remove

function template

std::sort

<algorithm>

default (1)

template <class RandomAccessIterator>  
void sort (RandomAccessIterator first, RandomAccessIterator last);

custom (2)

template <class RandomAccessIterator, class Compare>  
void sort (RandomAccessIterator first, RandomAccessIterator last, Compare comp);

Sort elements in range

Sorts the elements in the range [first, last) into ascending order.

The elements are compared using operator< for the first version, and comp for the second.

Equivalent elements are not guaranteed to keep their original relative order (see [stable\\_sort](#)).

Parameters

first, last

Random-access iterators to the initial and final positions of the sequence to be sorted. The range used is [first, last), which contains all the elements between first and last, including the element pointed by first but not the element pointed by last. RandomAccessIterator shall point to a type for which swap is properly defined and which is both move-constructible and move-assignable.

comp

Binary function that accepts two elements in the range as arguments, and returns a value convertible to bool. The value returned indicates whether the element passed as first argument is considered to go before the second in the specific strict weak ordering it defines. The function shall not modify any of its arguments. This can either be a function pointer or a function object.

Return value

none

Example

```
1 // sort algorithm example
2 #include <iostream> // std::cout
3 #include <algorithm> // std::sort
4 #include <vector> // std::vector
5
6 bool myfunction (int i,int j) { return (i<j); }
7
8 struct myclass {
9     bool operator() (int i,int j) { return (i<j);}
10 } myobject;
11
12 int main () {
13     int myints[] = {32,71,12,45,26,80,53,33};
14     std::vector<int> myvector (myints, myints+8); // 32 71 12 45 26 80 53 33
15
16     // using default comparison (operator <):
17     std::sort (myvector.begin(), myvector.begin()+4); // (12 32 45 71)26 80 53 33
18
19     // using function as comp
20     std::sort (myvector.begin()+4, myvector.end(), myfunction); // 12 32 45 71(26 33 53 80)
21
22     // using object as comp
23     std::sort (myvector.begin(), myvector.end(), myobject); // (12 26 32 33 45 53 71 80)
24
25     // print out content:
26     std::cout << "myvector contains:";
27     for (std::vector<int>::iterator it=myvector.begin(); it!=myvector.end(); ++it)
28         std::cout << ' ' << *it;
29     std::cout << '\n';
30
31     return 0;
32 }
```

Output:

myvector contains: 12 26 32 33 45 53 71 80

Complexity

On average, linearithmic in the distance between first and last: Performs approximately  $N \cdot \log_2(N)$  (where  $N$  is this distance) comparisons of elements, and up to that many element swaps (or moves).

Data races

The objects in the range [first, last) are modified.

Exceptions

Throws if any of the element comparisons, the element swaps (or moves) or the operations on iterators throws. Note that invalid arguments cause undefined behavior.

See also

<a href="#">stable_sort</a>	Sort elements preserving order of equivalents (function template)
<a href="#">partial_sort</a>	Partially sort elements in range (function template)
<a href="#">search</a>	Search range for subsequence (function template)

- [remove\\_copy](#)
- [remove\\_copy\\_if](#)
- [remove\\_if](#)
- [replace](#)
- [replace\\_copy](#)
- [replace\\_copy\\_if](#)
- [replace\\_if](#)
- [reverse](#)
- [reverse\\_copy](#)
- [rotate](#)
- [rotate\\_copy](#)
- [search](#)
- [search\\_n](#)
- [set\\_difference](#)
- [set\\_intersection](#)
- [set\\_symmetric\\_difference](#)
- [set\\_union](#)
- [shuffle](#)
- [sort](#)
- [sort\\_heap](#)
- [stable\\_partition](#)
- [stable\\_sort](#)
- [swap](#)
- [swap\\_ranges](#)
- [transform](#)
- [unique](#)
- [unique\\_copy](#)
- [upper\\_bound](#)

<b>reverse</b>	Reverse range (function template)
----------------	-----------------------------------