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

function template

std::next_permutation

<algorithm>

```
                template <class BidirectionalIterator>
    default (1)   bool next_permutation (BidirectionalIterator first,
                                        BidirectionalIterator last);
                template <class BidirectionalIterator, class Compare>
    custom (2)    bool next_permutation (BidirectionalIterator first,
                                        BidirectionalIterator last, Compare comp);
```

Transform range to next permutation

Rearranges the elements in the range [first,last) into the next *lexicographically greater* permutation.

A *permutation* is each one of the N! possible arrangements the elements can take (where *N* is the number of elements in the range). Different permutations can be ordered according to how they compare *lexicographicaly* to each other; The first such-sorted possible permutation (the one that would compare *lexicographically smaller* to all other permutations) is the one which has all its elements sorted in ascending order, and the largest has all its elements sorted in descending order.

The comparisons of individual elements are performed using either operator< for the first version, or *comp* for the second.

If the function can determine the next higher permutation, it rearranges the elements as such and returns true. If that was not possible (because it is already at the largest possible permutation), it rearranges the elements according to the first permutation (sorted in ascending order) and returns false.

Parameters

first, last
: Bidirectional iterators to the initial and final positions of the sequence. 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*.
BidirectionalIterator shall point to a type for which swap is properly defined.

comp
: Binary function that accepts two arguments of the type pointed by BidirectionalIterator, and returns a value convertible to bool. The value returned indicates whether the 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

true if the function could rearrange the object as a lexicographicaly greater permutation.
Otherwise, the function returns false to indicate that the arrangement is not greater than the previous, but the lowest possible (sorted in ascending order).

Example

```
1  // next_permutation example
2  #include <iostream>     // std::cout
3  #include <algorithm>    // std::next_permutation, std::sort
4
5  int main () {
6    int myints[] = {1,2,3};
7
8    std::sort (myints,myints+3);
9
10   std::cout << "The 3! possible permutations with 3 elements:\n";
11   do {
12     std::cout << myints[0] << ' ' << myints[1] << ' ' << myints[2] << '\n';
13   } while ( std::next_permutation(myints,myints+3) );
14
15   std::cout << "After loop: " << myints[0] << ' ' << myints[1] << ' ' << myints[2] << '\n';
16
17   return 0;
18 }
```

Output:

```
The 3! possible permutations with 3 elements:
1 2 3
1 3 2
2 1 3
2 3 1
3 1 2
3 2 1
After loop: 1 2 3
```

Complexity

Up to linear in half the distance between *first* and *last* (in terms of actual swaps).

Data races

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

Exceptions

Throws if any element swap throws or if any operation on an iterator throws.
Note that invalid arguments cause *undefined behavior*.

See also

prev_permutation	Transform range to previous permutation (function template)
lexicographical_compare	Lexicographical less-than comparison (function template)