C++
Information
Documentation
Reference
Articles
Forum

Reference
C Library
IOstream Library
Strings library
STL Containers
STL Algorithms
Miscellaneous

STL Algorithms
algorithm:
adjacent_find
binary_search
copy
copy_backward
count
count_if
equal
equal_range
fill
fill_n
find
find_end
find_first_of
find_if
for_each
generate
generate_n
includes
inplace_merge
iter_swap
lexicographical_compare
lower_bound
make_heap
max
max_element
merge
min
min_element
mismatch
next_permutation
nth_element
partial_sort
partial_sort_copy
partition
pop_heap
prev_permutation
push_heap
random_shuffle
remove
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
sort
sort_heap
stable_partition
stable_sort
swap
swap_ranges
transform
unique
unique_copy

function template

max_element

<algorithm>

```
template <class ForwardIterator>
  ForwardIterator max_element ( ForwardIterator first, ForwardIterator last );

template <class ForwardIterator, class Compare>
  ForwardIterator max_element ( ForwardIterator first, ForwardIterator last,
                                Compare comp );
```

Return largest element in range

Returns an iterator pointing to the element with the largest value in the range *[first,last)*. The comparisons are performed using either `operator<` for the first version, or *comp* for the second; An element is the largest if it does not compare less than any other element (it may compare equal, though).

The behavior of this function template is equivalent to:

```
template <class ForwardIterator>
  ForwardIterator max_element ( ForwardIterator first, ForwardIterator last )
{
  ForwardIterator largest = first;
  if (first==last) return last;
  while (++first!=last)
    if (*largest<*first)    // or: if (comp(*largest,*first)) for the comp version
      largest=first;
  return largest;
}
```

Parameters

first, last
: Input iterators to the initial and final positions of the sequence to use. 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*.

comp
: Comparison function object that, taking two values of the same type than those contained in the range, returns `true` if the first argument is to be considered less than the second argument, and false otherwise.

Return value

Iterator to largest value in the range.

Example

```
// min_element/max_element
#include <iostream>
#include <algorithm>
using namespace std;

bool myfn(int i, int j) { return i<j; }

struct myclass {
  bool operator() (int i,int j) { return i<j; }
} myobj;

int main () {
  int myints[] = {3,7,2,5,6,4,9};

  // using default comparison:
  cout << "The smallest element is " << *min_element(myints,myints+7) << endl;
  cout << "The largest element is " << *max_element(myints,myints+7) << endl;

  // using function myfn as comp:
  cout << "The smallest element is " << *min_element(myints,myints+7,myfn) << endl;
  cout << "The largest element is " << *max_element(myints,myints+7,myfn) << endl;

  // using object myobj as comp:
  cout << "The smallest element is " << *min_element(myints,myints+7,myobj) << endl;
  cout << "The largest element is " << *max_element(myints,myints+7,myobj) << endl;

  return 0;
}
```

upper_bound

The example illustrates how the comparison object can be either a function or an object whose class defines the `operator()` member. In this case both have been defined to perform a standard less-than comparison.
Output:

```
The smallest element is 2
The largest element is 9
The smallest element is 2
The largest element is 9
The smallest element is 2
The largest element is 9
```

Complexity

Linear: Performs as many comparisons as the number of elements in *[first,last)*, except for *first*.

See also

min_element	Return smallest element in range (function template)
upper_bound	Return iterator to upper bound (function template)
max	Return the greater of two arguments (function template)