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::min

<algorithm>

C++98 | C++11 | C++14

```
default (1)          template <class T> const T& min (const T& a, const T& b);
custom (2)           template <class T, class Compare>
                       const T& min (const T& a, const T& b, Compare comp);
initializer list (3) template <class T> T min (initializer_list<T> il);
                     template <class T, class Compare>
                       T min (initializer_list<T> il, Compare comp);
```

Return the smallest

Returns the smallest of *a* and *b*. If both are equivalent, *a* is returned.

The versions for *initializer lists (3)* return the smallest of all the elements in the list. Returning the first of them if these are more than one.

The function uses `operator<` (or *comp*, if provided) to compare the values.

The behavior of this function template (C++98) is equivalent to:

```
template <class T> const T& min (const T& a, const T& b) {
  return !(b<a)?a:b;     // or: return !comp(b,a)?a:b; for version (2)
}
```

Parameters

`a, b`
Values to compare.

`comp`
Binary function that accepts two values of type T as arguments, and returns a value convertible to `bool`. The value returned indicates whether the element passed as first argument is considered less than the second.
The function shall not modify any of its arguments.
This can either be a function pointer or a function object.

`il`
An initializer_list object.
These objects are automatically constructed from *initializer list* declarators.

T shall support being compared with `operator<`.

C++98 | C++11

For *(3)*, T shall be *copy constructible*.

Return value

The lesser of the values passed as arguments.

Example

```
// min example
#include <iostream>     // std::cout
#include <algorithm>    // std::min

int main () {
  std::cout << "min(1,2)==" << std::min(1,2) << '\n';
  std::cout << "min(2,1)==" << std::min(2,1) << '\n';
  std::cout << "min('a','z')==" << std::min('a','z') << '\n';
  std::cout << "min(3.14,2.72)==" << std::min(3.14,2.72) << '\n';
  return 0;
}
```

Output:

```
min(1,2)==1
min(2,1)==1
min('a','z')==a
min(3.14,2.72)==2.72
```

Complexity

Linear in one less than the number of elements compared (constant for *(1)* and *(2)*).

Exceptions

Throws if any comparison throws.
Note that invalid arguments cause *undefined behavior*.

See also

max	Return the largest (function template)

min_element	Return smallest element in range (function template)