

Search:

Reference <algorithm> **binary_search**

Not logged in
[register](#) [log in](#)

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

function template

std::binary_search

<algorithm>

```
default (1)  template <class ForwardIterator, class T>
              bool binary_search (ForwardIterator first, ForwardIterator last,
                                const T& val);

custom (2)   template <class ForwardIterator, class T, class Compare>
              bool binary_search (ForwardIterator first, ForwardIterator last,
                                const T& val, Compare comp);
```

Test if value exists in sorted sequence

Returns true if any element in the range `[first, last)` is equivalent to `val`, and false otherwise.

The elements are compared using operator< for the first version, and `comp` for the second. Two elements, `a` and `b` are considered equivalent if `!(a<b) && !(b<a)` or if `!(comp(a,b) && !comp(b,a))`.

The elements in the range shall already be [sorted](#) according to this same criterion (operator< or `comp`), or at least [partitioned](#) with respect to `val`.

The function optimizes the number of comparisons performed by comparing non-consecutive elements of the sorted range, which is specially efficient for [random-access iterators](#).

The behavior of this function template is equivalent to:

```
1 template <class ForwardIterator, class T>
2 bool binary_search (ForwardIterator first, ForwardIterator last, const T& val)
3 {
4     first = std::lower_bound(first, last, val);
5     return (first != last && !(val < *first));
6 }
```

Parameters

`first`, `last`

[Forward iterators](#) to the initial and final positions of a [sorted](#) (or properly [partitioned](#)) 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`.

`val`

Value to search for in the range.

For (1), `T` shall be a type supporting being compared with elements of the range `[first, last)` as either operand of operator<.

`comp`

Binary function that accepts two arguments of the type pointed by `ForwardIterator` (and of type `T`), and returns a value convertible to `bool`. The value returned indicates whether the first argument is considered to go before the second. The function shall not modify any of its arguments. This can either be a function pointer or a function object.

Return value

true if an element equivalent to `val` is found, and false otherwise.

Example

```
1 // binary_search example
2 #include <iostream>      // std::cout
3 #include <algorithm>     // std::binary_search, std::sort
4 #include <vector>        // std::vector
5
6 bool myfunction (int i,int j) { return (i<j); }
7
8 int main () {
9     int myints[] = {1,2,3,4,5,4,3,2,1};
10    std::vector<int> v(myints,myints+9);           // 1 2 3 4 5 4 3 2 1
11
12    // using default comparison:
13    std::sort (v.begin(), v.end());
14
15    std::cout << "looking for a 3... ";
16    if (std::binary_search (v.begin(), v.end(), 3))
17        std::cout << "found!\n"; else std::cout << "not found.\n";
18
19    // using myfunction as comp:
20    std::sort (v.begin(), v.end(), myfunction);
21
22    std::cout << "looking for a 6... ";
23    if (std::binary_search (v.begin(), v.end(), 6, myfunction))
24        std::cout << "found!\n"; else std::cout << "not found.\n";
25
26    return 0;
27 }
```

Output:

```
looking for a 3... found!
looking for a 6... not found.
```

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
shuffle
sort
sort_heap
stable_partition
stable_sort
swap
swap_ranges
transform
unique
unique_copy
upper_bound

Complexity

On average, logarithmic in the [distance](#) between *first* and *last*: Performs approximately $\log_2(N)+2$ element comparisons (where N is this distance).

On *non-random-access iterators*, the iterator [advances](#) produce themselves an additional linear complexity in N on average.

Data races

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

Exceptions

Throws if either an element comparison or an operation on an iterator throws.

Note that invalid arguments cause *undefined behavior*.

See also

find	Find value in range (function template)
lower_bound	Return iterator to lower bound (function template)
upper_bound	Return iterator to upper bound (function template)
equal_range	Get subrange of equal elements (function template)