# Search and Sort

This topic teaches the search and sort algorithms by demonstrating the usage including find, find_if, sort, binary_search, all_of, any_of, none_of and find_if_not  and explaining the algorithms

- Figure 16.6 demonstrates some basic searching and sorting capabilities of the Standard Library, including `find`, `find_if`, `sort`, `binary_search`, `all_of`, `any_of`, `none_of` and `find_if_not`.

```
auto location =
    find(a.cbegin(), a.cend(), 16);
```

- Uses the `find` algorithm to locate the value `16` in the range from `a.cbegin()` up to, but not including, `a.cend()`
- Requires its two iterator arguments to be at least *input iterators* and returns an *input iterator* that
  - Either is positioned at the first element containing the value
  - Or indicates the end of the sequence

```
location =
    find_if(a.cbegin(), a.cend(), greater10);
```

- Uses the `find_if` algorithm (a linear search) to locate the first value in the range from `a.cbegin()` up to, but *not* including, `a.cend()` for which the *unary predicate function* `greater10` returns `true`
- Requires its two iterator arguments to be at least *input iterators*
- Returns an *input iterator* that either is
  - Positioned at the first element containing a value for which the predicate function returns `true`
  - Or indicates the end of the sequence

```
sort(a.begin(), a.end());
```

- Uses the `sort` algorithm to arrange the elements in the range from `a.begin()` up to, but *not* including, `a.end()` in *ascending order*
- Requires its two iterator arguments to be *random-access iterators*
- A second version of this algorithm
  - Takes a third argument that is a *binary predicate function*
    - Taking two arguments that are values in the sequence
    - Returning a `bool` indicating the *sorting order*
      - If the return value is `true`
        - The two elements being compared are in *sorted order*

```
if (binary_search(a.cbegin(), a.cend(), 13))
    cout << "\n\n13 was found in a";
else
    cout << "\n\n13 was not found in a";
```

- Uses the `binary_search` algorithm to determine whether the value 13 is in the range from `a.cbegin()` up to, but *not* including, `a.cend()`
- The values must be sorted in *ascending* order
- Requires its two iterator arguments to be at least *forward iterators*
- Returns a `bool` indicating whether the value was found in the sequence

- A second version of this algorithm takes a fourth argument that is a *binary predicate function* taking two arguments that are values in the sequence and returning a `bool`
- The predicate function returns `true` if the two elements being compared are in *sorted order*
- To obtain the *location* of the search key in the container, use the `lower_bound` or `find` algorithms

```
if ( all_of( a.cbegin(), a.cend(), greater10))
   cout << "All are greater than 10";
else
   cout << "Some are not greater than 10";
```

- Uses the `all_of` algorithm to determine whether the *unary predicate function* `greater10` returns `true` for all of the elements in the range from `a.cbegin()` up to, but *not* including, `a.cend()`
- Requires its two iterator arguments to be at least *input iterators*

```
if ( any_of( a.cbegin(), a.cend(), greater10 ) )
   cout << "Some elements are greater than 10";
else
   cout << "No elements are greater than 10";
```

- Uses the `any_of` algorithm to determine whether the *unary predicate function* `greater10` returns `true` for at least one of the elements in the range from `a.cbegin()` up to, but *not* including, `a.cend()`
- Requires its two iterator arguments to be at least *input iterators*

```
if (none_of( a.cbegin(), a.cend(), greater10 ))
   cout << "None are greater than 10";
else
   cout << "Some are greater than 10";
```

- Uses the `none_of` algorithm to determine whether the *unary predicate function* `greater10` returns `false` for *all* of the elements in the range from `a.cbegin()` up to, but not including, `a.cend()`
- Requires its two iterator arguments to be at least *input iterators*

```
location = find_if_not
    (a.cbegin(), a.cend(), greater10);
```

- Uses the `find_if_not` algorithm to locate the first value in the range from `a.cbegin()` up to, but *not* including, `a.cend()` for which the *unary predicate function* `greater10` returns `false`
- Requires its two iterator arguments to be at least *input iterators*
- Returns an *input iterator* that
  - Either is positioned at the first element containing a value for which the predicate function returns `false`
  - Or indicates the end of the sequence

This topic taught the search and sort algorithms by demonstrating the usage including find, find_if, sort, binary_search, all_of, any_of, none_of and find_if_not  and explaining the algorithms