# 16 Standard Library Algorithms

## Objectives

In this chapter you'll:

- Understand minimum iterator requirements for working with Standard Library algorithms and containers.
- Create anonymous functions using lambda expressions.
- Capture local variables for use in lambda expressions.
- Use containers and iterators with many of the dozens of Standard Library algorithms.
- Use iterators with algorithms to access and manipulate the elements of Standard Library containers.
- Pass lambda expressions, function pointers and function objects into Standard Library algorithms to help them perform their tasks.

## Self-Review Exercises

**16.1** State whether each of the following is *true* or *false*. If *false*, explain why.

a) Standard Library algorithms can operate on C-like pointer-based arrays.

**ANS:** True.

b) Standard Library algorithms are encapsulated as member functions within each container class.

**ANS:** False. Standard Library algorithms are not member functions. They operate indirectly on containers, through iterators.

c) When using the `remove` algorithm on a container, the algorithm does not decrease the size of the container from which elements are being removed.

**ANS:** True.

d) One disadvantage of using Standard Library algorithms is that they depend on the implementation details of the containers on which they operate.

**ANS:** False. Standard Library algorithms do not depend on the implementation details of the containers on which they operate.

e) The `remove_if` algorithm does not modify the number of elements in the container, but it does move to the beginning of the container all elements that are not removed.

**ANS:** True.

f) The `find_if_not` algorithm locates all the values in the range for which the specified unary predicate function returns `false`.

**ANS:** False. It locates only the first value in the range for which the specified unary predicate function returns `false`.

g) Use the `set_union` algorithm to create a set of all the elements that are in either or both of two sorted sets (both sets of values must be in ascending order).

**ANS:** True.

**16.2** Fill in the blanks in each of the following statements:

a) Standard Library algorithms operate on container elements indirectly, using _____.

**ANS:** Iterators.

b) The `sort` algorithm requires a(n) _____ iterator.

**ANS:** random-access.

c) Algorithms _____ and _____ set every element in a range of container elements to a specific value.

**ANS:** `fill`, `fill_n`.

d) The _____ algorithm compares two sequences of values for equality.

**ANS:** `equal`.

e) The C++11 _____ algorithm locates both the smallest and largest elements in a range.

**ANS:** `minmax_element`.

f) A `back_inserter` calls the container's default _____ function to insert an element at the end of the container. If an element is inserted into a container that has no more space available, the container grows in size.

**ANS:** `push_back`.

g) Any algorithm that can receive a function pointer can also receive an object of a class that overloads the parentheses operator with a function named `operator()`, provided that the overloaded operator meets the requirements of the algorithm. An object of such a class is known as a(n) _____ and can be used syntactically and semantically like a function or function pointer.

**ANS:** function object.

**16.3** Write a statement to peform each of the following tasks:

a) Use the `fill` algorithm to fill the entire `array` of `strings` named `items` with `"hello"`.

**ANS:**
```
fill(items.begin(), items.end(), "hello");
```

b) Function `nextInt` returns the next `int` value in sequence starting with `0` the first time it's called. Use the `generate` algorithm and the `nextInt` function to fill the `array` of `int`s named `integers`.

**ANS:**
```
generate(integers.begin(), integers.end(), nextInt);
```

c) Use the `equal` algorithm to compare two lists (`strings1` and `strings2`) for equality. Store the result in `bool` variable `result`.

**ANS:**
```
bool result{equal(strings1.cbegin(), strings1.cend(), strings2.cbegin())};
```

d) Use the `remove_if` algorithm to remove from the `vector` of `strings` named `colors` all of the strings that start with `"bl"`. Function `startsWithBL` returns `true` if its argument `string` starts with `"bl"`. Store the iterator that the algorithm returns in `newEnd`.

**ANS:**
```
auto newEnd remove_if(colors.begin(), colors.end(), startsWithBL);
```

e) Use the `replace_if` algorithm to replace with `0` all elements with values greater than `100` in the `array` of `int`s named `values`. Function `greaterThan100` returns `true` if its argument is greater than `100`.

**ANS:**
```
replace_if(values.begin(), values.end(), greaterThan100, 0);
```

f) Use the `minmax_element` algorithm to find the smallest and largest values in the `array` of `double`s named `temperatures`. Store the `pair` of iterators that's returned in `result`.

**ANS:**
```
auto result = minmax_element(temperatures.cbegin(), temperatures.cend());
```

g) Use the `sort` algorithm to sort the `array` of `strings` named `colors`.

**ANS:**
```
sort(colors.begin(), colors.end());
```

h) Use the `reverse` algorithm to reverse order of the elements in the `array` of `strings` named `colors`.

**ANS:**
```
reverse(colors.begin(), colors.end());
```

i) Use the `merge` algorithm to merge the contents of the two sorted `array`s named `values1` and `values2` into a third `array` named `results`.

**ANS:**
```
merge(values1.cbegin(), values1.cend(), values2.cbegin(), values2.cend(),
   results.begin());
```

j) Write a lambda expression that returns the square of its `int` argument and assign the lambda expression to variable `squareInt`.

**ANS:**
```
auto squareInt = [](int i) {return i * i;};
```

## Exercises

*NOTE: Solutions to the programming exercises are located in the* **`ch16solutions`** *folder.*

**16.4** State whether each of the following is *true* or *false*. If *false*, explain why.

a) Because Standard Library algorithms process containers directly, one algorithm can often be used with many different containers.

**ANS:** False. Because Standard Library algorithms process containers *only indirectly* through iterators, one algorithm can often be used with many different containers.

b) Use the `for_each` algorithm to apply a general function to every element in a range; `for_each` does not modify the sequence.

**ANS:** False. It is possible to modify the sequence if the general function receives the elements by reference.

c) By default, the `sort` algorithm arranges the elements in a range in ascending order.

**ANS:** True.

d) Use the `merge` algorithm to form a new sequence by placing the second sequence after the first.

**ANS:** False. The `merge` algorithm combine two sorted ascending sequences of values into a third sorted ascending sequence.

e) Use the `set_intersection` algorithm to find the elements from a first set of sorted values that are not in a second set of sorted values (both sets of values must be in ascending order).

**ANS:** False. This algorithm is used to find the elements from the first set of sorted values that *are* in the second set of sorted values.

f) Algorithms `lower_bound`, `upper_bound` and `equal_range` are often used to locate insertion points in sorted sequences.

**ANS:** True.

g) Lambda expressions can also be used where function pointers and function objects are used in algorithms.

**ANS:** True.

h) C++11's lambda expressions are defined locally inside functions and can "capture" (by value or by reference) the local variables of the enclosing function then manipulate these variables in the lambda's body.

**16.5** Fill in the blanks in each of the following statements:

a) As long as a container's (or built-in array's) _____ satisfy the requirements of an algorithm, the algorithm can work on the container.

**ANS:** iterators.

b) Algorithms `generate` and `generate_n` use a(n) _____ function to create values for every element in a *range* of container elements. That type of function takes no arguments and returns a value that can be placed in an element of the container.

**ANS:** generator function.

c) Pointers into built-in arrays are _____ iterators.

**ANS:** random-access.

d) Use the _____ algorithm (the template of which is in header `<numeric>`) to sum the values in a range.

**ANS:** `accumulate`.

e) Use the _____ algorithm to apply a general function to every element in a range when you need to modify those elements.

**ANS:** `for_each`.

f) In order to work properly, the `binary_search` algorithm requires that the sequence of values must be _____.

**ANS:** sorted.

g) Use the function `iter_swap` to exchange the elements that are pointed to by two _____ iterators and exchanges the values in those elements.

**ANS:** forward.

h) C++11's `minmax` algorithm receives two items and returns a(n) _____ in which the smaller item is stored in `first` and the larger item is stored in `second`.

**ANS:** `pair`.

i) _____ algorithms modify the containers they operate on.

**ANS:** mutating sequence algorithms.

**16.6** List several advantages function objects provide over function pointers.

**ANS:** The compiler can inline a function object's overloaded `operator()` to improve performance. Since they're objects of classes, function objects can have data members that `operator()` can use to perform its task.

**16.7** What happens when you apply the `unique` algorithm to a sorted sequence of elements in a range?

**ANS:** All of the duplicated elements in that range are removed.

**16.14** Explain why using the "weakest iterator" that yields acceptable performance helps produce maximally reusable components.

**ANS:** More containter types can be used by the code with the specified iterators.