# CIS7 Lab 3: Sets in C++

In C++ library, a set is an Associative container which ***contains a sorted set of unique objects*** of type Key. Each element may occur only once, so duplicates are not allowed. The value of the elements in a set **cannot** be modified once in the container, i.e., the **elements are always const**. But they can be ***inserted or removed*** from the container. Set containers are generally slower than unordered set containers in accessing individual elements by their key, but they allow the direct iteration on subsets based on their order.

## Insert element

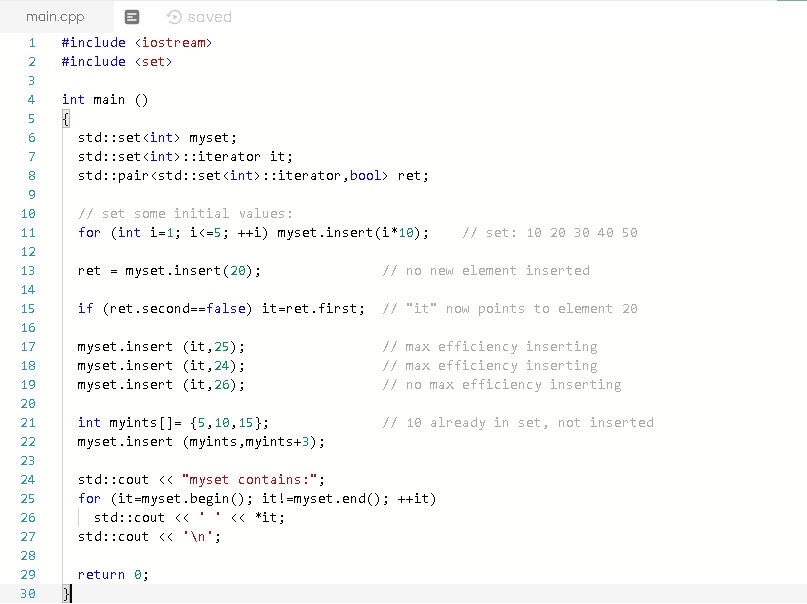
Extends the container by **inserting** new elements, effectively increasing the container size by the number of elements inserted.

Because elements in a set are unique, the ***insertion operation checks whether each inserted element is equivalent to an element already in the container, and if so, the element is not inserted, returning an iterator to this existing element*** (if the function returns a value).

Set containers keep all their elements sorted following the criterion specified by its comparison object. The elements are always inserted in its respective position following this ordering.

The ***parameters*** determine how many elements are inserted and to which values they are initialized

**Example 1:**



1. Follow Example 1 and insert different values (30, 40, 50, 60). What is your observation when inserting these values? Provide a screen capture of the result.

50 and 60 were added but 30 and 40 weren’t because they were already in the set. Also, the set remained sorted after 50 and 60 were added.

A screenshot of a computer

Description automatically generated

1. Refer to Example 1, and create a set that contains numbers that are different from the previous exercise, then attempt to insert numbers. Provide screen capture of your code and output. Briefly describe the result.

Unique values are inserted into the set while values already inserted in the set are not added. The set remains sorted even when values are inserted out of order.

A screenshot of a computer

Description automatically generated

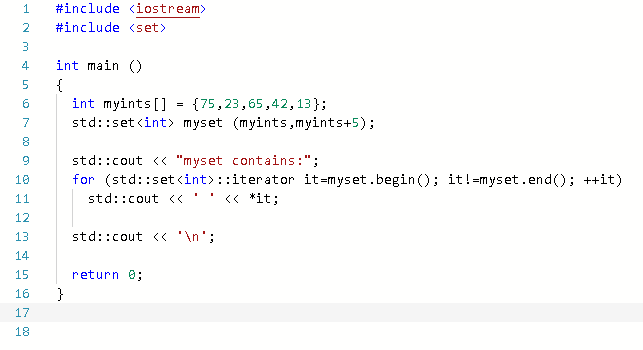
## Iterator - Begin

**Iterator:** provide a means for accessing data stored in container or pointing to an item that is part of a larger container of items.

**Returns an iterator** referring to the ***first element in the set container***. Because set containers keep their ***elements ordered at all times***, ***begin points*** to the element that ***goes first*** following the container's sorting criterion. If the ***container is empty, the returned iterator value shall not be dereferenced***.

An iterator to the first element in the container. If the set object is const-qualified, the function returns a const\_iterator. Otherwise, it returns an iterator. Member type iterator and const\_iterator are bidirectional iterator types pointing to elements.

**Example 2**



1. Input Example 2 code into IDE and run the program. Edit the program by changing “75” element to “55,” or “13” to “33”. Rerun the program and describe the output AND provide screen capture.

No issue changing value because value has not been inserted into set. 75 / 13 is no longer in the set and the set now contains 55 / 33 respectively in the correct sorted order.

A screenshot of a computer

Description automatically generated

1. Edit the code and remove all the elements. Provide screen capture and explain why the output is as shown on screen capture.

Random Garbage Values are displayed because we are trying to dereference memory in the set which has not been assigned to any value since our input data was empty. When inputting into our set we specified that our set was 5 values long, so we get back 5 garbage values. The values that are shown are left over values from whatever values that memory held before being allocated to the set. Since C/C++ does not initialize variables to a default value like other programming languages such as Java we get garbage values.

A screenshot of a computer

Description automatically generated

## Clear Content

Removes all elements from the set container (which are **destroyed**), leaving the container with a size of 0.

**Example 3**



All iterators, pointers and references related to this container are invalidated. The container is modified.

All contained elements are modified.

1. Input Example 3 code in IDE and run the program. Which set is cleared? Provide screen capture and explain.

The set containing 100, 200, 300 was cleared. The empty set left over after clearing then had 1101 and 2202 inserted leaving a new set with 2 elements.

A screenshot of a computer

Description automatically generated

1. Replace **myset.clear ();** with **myset.insert(0);** and rerun the program. Provide screen capture and explain the result.

We end up with a result of 0, 100, 200, 300, 1101, 2202 because our set is never cleared. We are simply inserting another value into the set which in this case is 0 hence the first value being 0. We end up with a set of length 6 because we insert 6 times and none of the values are ever removed.

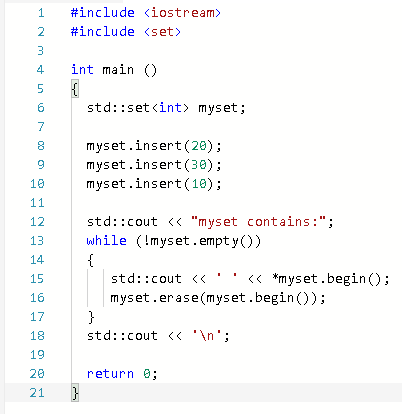
A screenshot of a computer

Description automatically generated

## Empty Set

**set::empty** returns whether the set container is empty (i.e. whether its size is 0). This function does not modify the container in any way. To clear the content of a set container, see set::clear. Return value is true if the container size is 0, false otherwise.

**Example 4:**



1. Test Example 4 in IDE. Notice that WHILE loop is used with set::empty function. Provide screen capture and explain the purpose of WHILE loop in relation to the output.

This code essentially performs the same function as a for loop using an iterator to print the contents of the set. We constantly test to see if the set is empty in the while loop. Each iteration we print out the value stored in the first element of the set with std::cout << “ ” << \*myset.begin() and then erase the first element of the set using myset.erase(myset.begin());. This process repeats until myset.empty() evaluates to false so that when the Logical Not applies to it the while loop stops execution since our set no longer contains any more elements to print.

A screenshot of a computer

Description automatically generated

1. Save document and submit Lab 3 in Canvas.