# 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

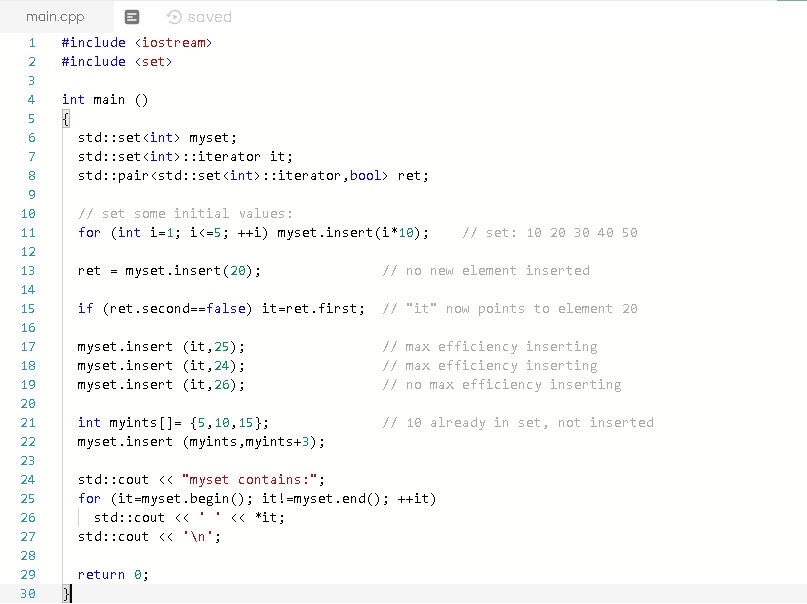
Extends the container by **inserting** new elements, effectively increasing the container size by the number of elements inserted.

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.
2. 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.

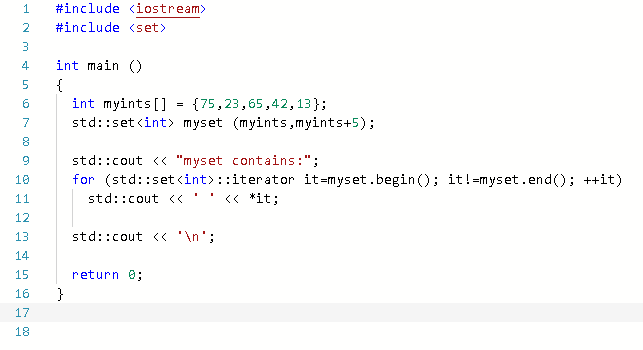
## Iterator - Begin

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

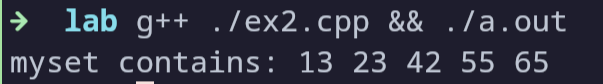
**Returns an iterator** referring to the ***first element in the set container***. Because set containers keep their ***elements ordered at all times***, ***begin points*** to the element that ***goes first*** following the container's sorting criterion. If the ***container is empty, the returned iterator 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.

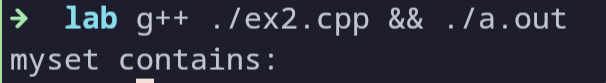
**Example 2**



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.



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



## Clear Content

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

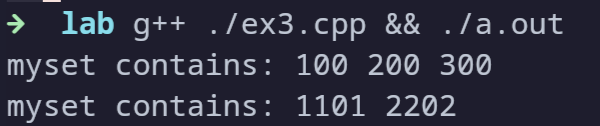
**Example 3**



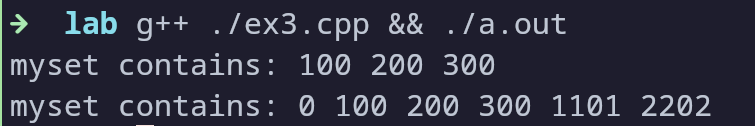
All iterators, pointers and references related to this container are invalidated. The container is modified.

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.



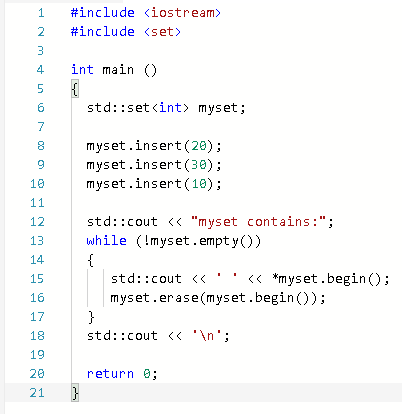
1. Replace **myset.clear ();** with **myset.insert(0);** and rerun the program. Provide screen capture and explain the result.



## 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.
2. Save document and submit Lab 3 in Canvas.