Containers in C++ STL | C++ Tutorials for Beginners #70

In the last tutorial, we had briefed about the three components of STL, namely,
Containers, objects which store data, **Algorithms,** set of procedures to process data, and **Iterators,** objects which point to some element in a container. Today, in this tutorial, we will be interested in discussing more about containers.
Containers are themselves of three types:
1. Sequence Containers
2. Associative Containers
3. Derived Containers

When we talked about containers, we said containers are objects which store data, but what are its three types all about? We'll discuss that too.

- **Sequence Containers**

A **sequence container** stores that data in a linear fashion. Refer to the illustration below to understand what storing something in a linear fashion means.

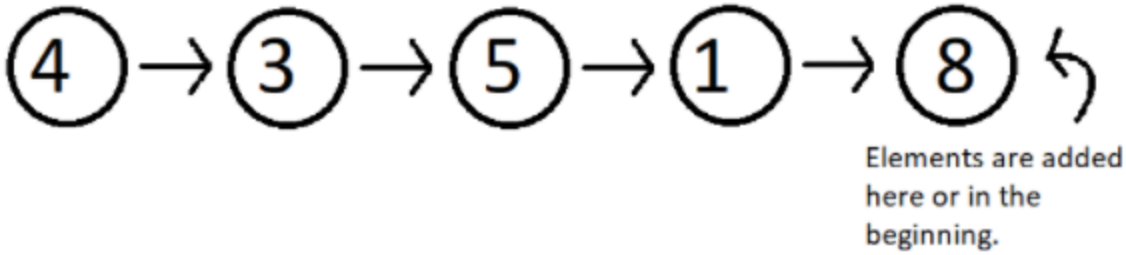

Figure 1: Elements stored in a linear fashion

Sequence containers include **Vector, List, Dequeue etc.** These are some of the most used sequence containers.

- **Associative Containers**

An **associative container** is designed in such a way that enhances the accessing of some element in that container. It is very much used when the user wants to fastly reach some element. Some of these containers are, **Set, Multiset, Map, Multimap etc.** They store their data in a tree-like structure.

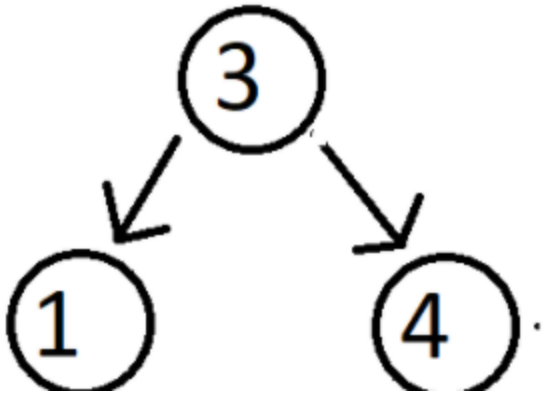

Figure 2: A tree-like structure

- **Derived Containers**

As the name suggests, these containers are derived from either the sequence or the associative containers. They often provide you with some better methods to deal with your data. They deal with real life modelling. Some examples of derived containers are **Stack, Queue, Priority Queue, etc.** The following illustration give you the idea of how a stack works.

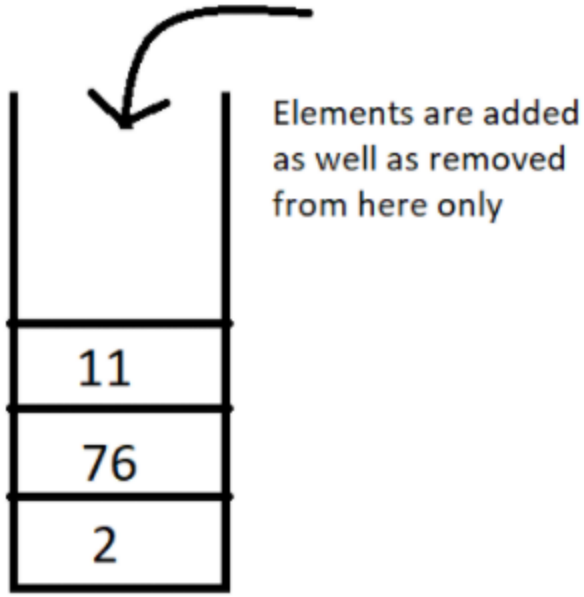

Figure 3: A stack, works on the first in first out [FIFO] method

Now since we have got the basic idea of all the three types of containers, a question which might arise is **when to use which**. So, let's deal with that,

In sequence containers, we have **Vectors**, which has following properties:

1. Faster random access to elements in comparison to array

2. Slower insertion and deletion at some random position, except at the end.
3. Faster insertion at the end.

In **Lists,** we have,

1. Random accessing elements is too slow, because every element is traversed using pointers.
2. Insertion and deletion at any position is relatively faster, because they only use pointers, which can easily be manipulated.

In associative containers, every operation except random access is faster in comparison to any other containers, be it inserting or deleting any element.

In associative containers, we cannot specifically tell which operation is faster or slower, we'll have to inspect every data structure separately, and to get a clearer picture of all of these, you can access my Data Structure course : Data Structures and Algorithms Course in Hindi

For now, I'd like to hold on to our topic STL, and get you a strong hold on this too. In the coming videos, we'll deal with our vectors, list, dequeues, set, multiset, maps, stack and much more. Just bear with me.

Thank you, for being with me throughout, hope you liked the tutorial. If you haven't checked out the whole playlist yet, move on to codewithharry.com or my YouTube channel to access it. I hope you enjoy them all. See you all in the next tutorial where we'll talk about Vectors in C++ STL in detail. Till then keep coding.

Previous Next

CodeWithHarry