Containers in STL

STL = Containers + Algorithm + Iterators

**Object Which Points to an Element of a Containers**

**Procedure to Process Data**

**Objects Which Stores Data**

Containers

Sequence Containers 🡪 Stores Data in Linear Fashion

Derived Containers 🡪 Real World Modelling

Associative Containers 🡪 Direct Access to Some Elements

* Sequence Containers :

A **Sequence container** stores that data in a linear fashion .

8

1

5

3

4

Elements are added here or in the beginning

* Sequence Containers include **Vector, List,**

**Dequeue** etc. These are most important

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 elements .Some of these containers are **Sets ,Multisets ,Map ,Multimaps etc.** They store there data in tree like structure.

4

1

* 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 **Stacks, Queue, Priority Queue , etc.**

NOTES:- ->A Queue works on First in first out [FIFO] method.

->A Stacks works on Last in first out [LIFO] method.

* **The following illustration give you the idea of how a stacks works.**

Element are added as well as removed from here only

2

76

11

45

* **Vectors**

1. Faster random access to element in comparison to array
2. Slower insertion and deletion at some random position ,except at the end .
3. Faster insertion/deletion at the end.

* **Lists**

1. **Random accessing element is too slow**, because every elements is transferred using Pointers.
2. **Insertion and Deletion at any position is relatively faster**, because they only use Pointers, which can easily be manipulated.
3. **Deletion/Insertion is Faster at the end.**

>> In associative containers, every operation except random access is faster in comparison to any other containers.

>>In Derived containers, we cannot specifically tell which operation is faster or slower, all depends upon DATA STRUCTURE.