C ++ has some built-in libraries, called standard template libraries. Many problems can be easily solved by using these STL’s. If you need to use a stack, queue in such a problem, you don't need to write new code for them later. The problem can be solved by using stack, queue STL.

**STL** 🡪 Standard template library

**Library of what?**

Generic classes and functions.

**Why use STL?**

We can use well-tested components for time Savings.

STL is used because it’s a good idea not to reinvent invented things.

**Components of STL ->**

I > Containers (Stores data) -> Use template classes

II > Algorithms (Sorting/Searching) -> Use template functions

II > Iterator (Object that points to an element is a container + connect algorithms with containers)

#Suppose this is a **container** because it stores some values.



#Now we want to **sort** the elements.

#For sorting we use an **iterator** to point some elements in the container.



#Using an iterator we have pointed to our first element.



#Now we need to **move the iterator** for sorting the elements.

# Iterator moves as instructed by the **algorithms**.

**## STL worked like this way.**

**STL** 🡪 Containers + Algo + Iterators

**Containers** 🡪 Object which stores data

**Algo** 🡪 Procedure to process data

**Iterator** 🡪 Object which points to an element of a container

#Three types of containers.

**Sequence Containers**



🡪 Stores data in a linear fashion.Like :



I > Vector

II > Deque

III > List

**Associative Containers**

🡪Direct Access. Stores data as a tree-like structure. Like :



I > Set



II > Multiset



III > Map



IV> Multimap

**Derived Containers**

🡪Real-world Modeling.

I > Stack

II > Queue

III > Priority Queue

**#When to use which :**

**Sequence containers :**

I > vector

Random access fast. Because we can use [] operator to direct access.

Insertion, deletion slow.

II > list

Random access is slow. Because we can’t use [] and we have to use an iterator for random access.

Insertion, deletion fast.

**Associative containers:**

All operations are fast except for random access.

**Derived containers:**

Depends on the data structure.