# Standard Template Library

# (STL)

# STL INTRODUCTION

- Standard Template Library (STL) is a collection of generic software components (**generic containers**) and **generic algorithms**, glued by objects called **iterators**.
- STL functions are known as ***generic algorithms***.
- STL contains so many useful algorithms such as find(), replace(), merge(), and sort().

- Being generic, these algorithms are non-member functions and can be used with all the containers.

# STL

- Basically STL Contains :
  - **generic containers**
  - **generic algorithms**
  - **Iterators**

- **GENERIC PROGRAMMING :** STL is not a normal library. It is designed on the basis of a few very important principles independent of C++ itself.

- **Iterators** can be refer as generic pointers.

# GENERIC SOFTWARE COMPONENTS(Containers)

- STL has generic software components (containers). These containers are classes that can contain other objects.

1) **Vector** (implementation is like an array), **list** (implementation of doubly linked list), and **deque** (implementation of deque) are called ***sequence containers.***
2) **Set**, **multiset**, **map**, and **multimap** are known as ***associative sorted containers***. Unlike sequence containers, these containers keep the contents in a sorted form.
3) There are adapted containers such as **queue** and **stack** that are not true containers but are implemented using sequence containers.

- STL provides 'reusable' components such as vector, list, and deque.
- **Advantages:**
  - Small in number
  - Generic
  - Efficient, tested, debugged, and standardized
  - Portability and reusability

# GENERIC ALGORITHMS

- As STL is designed for speed, the algorithms that operate on the software components are designed in such a way that they depend very little on the data structure of the component (e.g., an algorithm called find() does not vary much in speed if it is finding in the vector, stack, or queue).
- Advantages :
  - readymade sort(),merge()
  - Efficient
  - Standardized
  - Semantics

```
//A Vector using STL
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;
class Student
{
private:
int RollNumber;
float TotalMarks;
public:
Student(){}
Student(int TempRollNumber, float
TempTotalMarks)
{
RollNumber = TempRollNumber;
TotalMarks = TempTotalMarks;
}
void operator = (Student TempStud)
{
RollNumber = TempStud.RollNumber;
TotalMarks = TempStud.TotalMarks;
}
bool operator < (Student TempStud)
{
return(TotalMarks < TempStud.TotalMarks);
}
friend ostream & operator <<(ostream &
TempOut, Student & TempStud);
};
ostream & operator <<(ostream & TempOut, Student &
TempStud)
{
TempOut << "The mark of roll number " <<
TempStud.RollNumber << " is " << TempStud.
TotalMarks;
return TempOut;
}

int main()
{
vector  <Student>  StudMarks;
float TempMarks;
int i = 0;
for(;;)
{
cout << "Enter the mark for roll number " << i + 1 << " Enter
−1 to stop: ";
cin >> TempMarks;
if(TempMarks== -1) break;
StudMarks.push_back(Student(i + 1, TempMarks));
++i;
}
cout << "The size of StudMarks is " << StudMarks.size()<<
endl;
vector <Student>::iterator index;
sort(StudMarks.begin(), StudMarks.end());
for(index = StudMarks.begin(); index < StudMarks.end();
++index)
cout << *index << endl;
return 0;
}
```