TATA ELXSI

OBJECT ORIENTED PROGRAMMING USING C++
Module 11

Learning & Development Team

Standard Template Library

Standard Template Library (STL)

- Constructed from template classes. The algorithms and data structures can be applied to any type of data.

- Based on three fundamental items:

  -***Containers***

  template data structures

  -Algorithms

  data manipulation, searching, sorting, etc

  - Iterators

  like pointers, access elements of containers

STL Containers

- There are three types of containers
  - Sequence containers
    - Linear data structures (vectors, linked lists)
  - Associative containers
    - Non-linear, can find elements quickly
    - Key/value pairs
  - Container adapters

- All Containers have some common functions

STL Containers cont..

- Sequence containers
  - Vector , deque, list
- Associative containers
  - Set , multiset, map, multimap.
- Container adapters
  - Stack ,queue, priority_queue.

STL Containers types

Container	Type	Description
deque	Sequential	Double-ended queue
list	Sequential	Linear list
map	Associative	Collection of key/value pairs in which each key is associated with exactly one value
multimap	Associative	Collection of key/value pairs in which each key may be associated with more than one value
multiset	Associative	Collection in which each element is not necessarily unique
priority_queue	Adaptor	Priority queue
queue	Adaptor	Queue
set	Associative	Collection in which each element is unique
stack	Adaptor	Stack
vector	Sequential	Dynamic array

Common STL Member Functions

- Member functions for all containers
  - Default constructor, copy constructor, destructor
  - empty
  - max_size, size
  - = < <= > >= == !=
  - swap
- Functions for first-class containers
  - begin, end
  - erase, clear

Iterators

- Iterators are similar to pointers
  - Point to first element in a container
  - Iterator operators same for all containers
    - `*` dereferences
    - `++` points to next element
    - `begin()` returns iterator to first element
    - `end()` returns iterator to last element

Example : Vector container

- Vector
  - Have to include the following header file **<vector>**
  - Data structure with contiguous memory locations
    - Access elements with **[]**
  - Use when data must be sorted and easily accessible
- When memory exhausted
  - Allocates larger, contiguous area of memory
- Has random access iterators

Vector container operations

- vector class member functions
  - push_back(value)
    - Add element to end (found in all sequence containers).
  - size()
    - Current size of vector
  - capacity()
    - How much vector can hold before reallocating memory
  - insert(*iterator*, *value*)
    - Inserts *value* before location of *iterator*

Vector container operations cont..

- erase(iterator)
  - Remove element from container
- erase(iter1, iter2)
  - Will give a range to delete between iter1 to iter2
- clear()
  - Will erase the entire vector.
- begin()
  - Will return the iterator to the beginning.
- end()
  - Will return the iterator to the end of vector.

STL – Sample Program – 1

```
#include <stack>
int main ()
{  stack<int> mystack;

  for (int i=0; i<5; ++i)
  mystack.push(i);

  cout << "Popping out elements...";
  while (!mystack.empty())
  {     cout << " " << mystack.top();
        mystack.pop();
   }
  return 0;
}
```

- **O/P : ???**

Popping out elements... 4 3 2 1 0

STL – Sample Program – 2

```
#include <queue>
int main ()
{ queue<int> myqueue;
  int sum (0);

  for (int i=1;i<=10;i++)
  myqueue.push(i);

  while (!myqueue.empty())
  {
     sum += myqueue.front();
     myqueue.pop();
  }
  cout << "total: " << sum << endl;
  return 0;
}
```

- **O/P : ???**
- total: 55

STL – Sample Program – 3

```
#include <vector>
int main ()
{
  vector<int> myvector;
  int sum =0;
  myvector.push_back (100);
  myvector.push_back (200);
  myvector.push_back (300);

  while (!myvector.empty())  {
    sum+=myvector.back();
    myvector.pop_back();
  }
  cout << "The elements of   myvector summed "
<< sum;
  return 0;
}
```

- **O/P : ???**

The elements of myvector summed 600

STL – Sample Program – 4

```
#include <algorithm>
#include <vector>
int main()
{ vector<int> v(5);
  bool found;

  for (int i = 0; i < 5; ++i)
     v[i] = i;

  found = binary_search(v.begin(), v.end(), 3);
  cout << found << " ";

  found = binary_search (v.begin(), v.end(), 9);
  cout << found;
  return 0;
}
```

- **O/P : ???**

  1 0

STL – Sample Program – 5

```
#include <algorithm>
int main()
{
    int coll[] = { 5, 6, 2, 4, 1, 3 };

    // sort from beginning
    sort (coll, coll+6);

    // print all elements
        for(int i =0;i<6;i++)
                cout<<coll[i]<<" ";

    return 0;
}
```

- **O/P : ???**

1 2 3 4 5 6

STL – Sample Program – 6

```
#include<iostream>
#include<algorithm>
#include<vector>
using namespace std;

int main (){
   vector<int> vecOb;
  for (int i=1; i<=5; i++) vecOb.push_back(i);

   cout << "vecOb contains:";
  for ( vector<int>::iterator it = vecOb.begin() ; it != vecOb.end(); ++it)
     cout << ' ' << *it;
   cout << '\n';
   return 0;
}
```

- **O/P : ???**

1 2 3 4 5

STL – Sample Program – 7

```
int main()
{
        vector<int> v(5);
        for (int i = 0; i < 5; ++i)
        v[i] = i;

        v.push_back(80);
        v.push_back(10);
        v.insert(v.begin(),90);
        v.insert(v.begin()+3,60);

        vector<int>::iterator it;
        for(it = v.begin(); it != v.end();it++){
                cout<<" -> " << *it;
        }
        sort(v.begin(),v.end());
        cout<<"\n After Sorting : "<<endl;
        for(it = v.begin(); it != v.end();it++){
                cout<<" -> " << *it;
        }

  return 0;
}
```

```
-> 90 -> 0 -> 1 -> 60 -> 2 -> 3 -> 4 -> 80 -> 10
After Sorting :
-> 0 -> 1 -> 2 -> 3 -> 4 -> 10 -> 60 -> 80 -> 90
--------------------------------
```

End of Module 11

Disclaimer

- Some examples and concepts have been sourced from the below links and are open source material
  - **http://cppreference.com**
  - **www.cplusplus.com**

- References:
  - *C++: The Complete Reference* - 4th Edition by Herbert Schildt, Tata McGraw-Hill publications.
  - *The C++ Programming Language*- by Bjarne Stroustrup.
  - *Practical C++ Programming*- by Steve Oualline, O'Reilly publications.

Learning & Development Team

ITPB Road Whitefield
Bangalore 560 048 India
Tel +91 80 2297 9123
Fax +91 80 2841 1474
e-mail info@tataelxsi.com

www.tataelxsi.com