standard template library

STL is a set of classes to provide a common programming data and functions such as list, stacks, array, etc.

STL has 4 compnents

1. Algorithms
2. Containers
3. Function
4. Iterators

**Container:** container classes store objects and data

**Types of container**

**1.sequenctial container:** implement data structure which can be accessed in a sequential

* List
* Vector
* Dequeue
* Forward list

2.**container adaptor:** provide a different interface for sequential containers

* Queue
* Priority queue
* Stack

3**.Associative container**: implement sorted data structure that can be quickly searched

* Set
* Multiset
* Map
* Multimap

4.**Unordered associative container**: implement unsorted data structure that can be quickly searched

* Unsorted multiset
* Unsorted map
* Unsorted multimap
* Unsorted set

**Algorithm**: the header algorithm define a collection of function especially design to act on container

* Sorting
* partition operator
* searching
* Array algorithm(max/min)

**Functions**: the STL includes classes that overload the function call operator instances of such classes are called function object or functors.

Eg. Functors

**Iterators**: iterators are used for working upon a sequence of they ae the major feature that allow generality In STL.

eg. Iterators

**vector**

vector are same as dynamic arrays with the ability to resize itself automatically when an element is inserted or deleted, with their storage being handled automatically by the container.it is stored in contiguous memory location.

Some important function:

1. begin()- returns the iterator (or pointer) pointing to the first element in the vector. Eg. Vrr.begin()
2. end()- returns the iterator pointing to the theoretical element that follows the last element in the vector.eg. vrr.end()

example code for loop:

for(auto it=vrr.begin();it!=vrr.end();it++)

{

Cout<<\*it<< “ “;

}

1. rbegin()- returns the iterators pointing to the last element in

the vector(reverse beginning).

1. rend()- returns a reverse iterators pointing to the theoretical element preceding the first element in the vector
2. cbegin()- returns a constant iterator pointing to the first element in the vector.
3. cend()- returns a constant iterator pointing to the theoretical element that follows the last element in the vector
4. size()- returns the maximum number of elements in the vector.

Eg. Vrr.size()

1. max\_size()- returns the maximum number of the elements that the vector can store. Eg max\_size().
2. empty()- returns whether the container is empty or not.

Eg if(vrr.empty())

Cout<<”empty”;

Else

Cout<<”not empty”;

1. at(g)- returns the reference to the element at the position ‘g’ in the vector. Eg. Vrr.at(4);
2. front()- returns a reference to the first element in the vector. Eg. Vrr.front()
3. end()- returns the reference to the last element in the vector. Eg. Vrr.end()
4. push\_back()- it push the element into the vector from the back. Eg. Vrr.push\_back()
5. pop\_back()- it is used to pop or rmove elements from a vector from the back. Eg. Vrr.pop\_back().
6. swap()- it is used to swap the contents of one vector with another vector of same type.size may differ.

V1.swap(v2).

1. clear()- it is used to remove all the elements of the vector.

Eg. Vrr.clear().

#include<iostream>

#include<vector>

using namespace std;

int main()

{

  //vector <datatype> vetorname

    vector <int> vrr;

  //function1. size()=>returns the number of elements in a vector

    cout<<vrr.size()<<endl;

  //function2. push\_back()=> it puches the element at the back at the end of vector

  vrr.push\_back(10);

  cout<<vrr.size()<<endl;

  cout<<vrr[0];

  cout<<vrr.max\_size();

      return 0;

}