Standard Template Library

Programming Club

(2016-17)

Science and Technology Council

IIT Kanpur

Introduction

Competitive Programming doesn’t have any restrictions as to which language you code in - as long as you can get the correct answer to the question without violating the constraints set on the time and memory constraints. So there is no harm in coding in the language you are comfortable with.

However , it is advised to use a language like C++ or Java for Competitive Programming mainly because of the pre-defined data structures offered by both (more so by C++). No one wants to code heavy data-structures like a Red-Black BST,queue,stack or heap in the short period of a contest.

Here we will be teaching you all the power of C++’s STL-one of the most comprehensive collection of data structures and algorithms needed for CP.

Topics to be covered:

1. Hello World! In C++
2. What is STL and Containers
3. Pair
4. Vector
5. Strings
6. Stack/Queue
7. Iterators
8. Sort,upper_bound(),lower_bound()
9. Maps and Set

Hello World!

This is how a standard Hello World! Code in C++ looks like

```
#include <iostream>
using namespace std;
int main()
{
   cout<<"Hello World!"<<endl;
   return 0;
}
```

What is STL? What are Containers?

STL stands for Standard Template Library. It is composed of several pre-defined header files which contain robust implementations of several algorithms and data structures which are very widely used.

STL has a number of containers. What is a container? Basically a container is as the name suggests-something that stores the data. C had one container-the array. C++ on the other hand has several containers depending on our needs with the data:

1. C++ has vectors which are “dynamic” arrays.They can be resized,support insertion and deletion of elements in the beginning and the end.
2. STL set is a container that helps maintaining the functionality of a set(the one in mathematics).We can add elements,remove any elements and then perform various queries.
3. STL map is another container similar to a set. It can be used as an array but with any data type as indices.

Pairs

Often we encounter situations where we need two different values to represent a state.Like (x,y) to define a position in a 2D grid ,(name,roll number) for a student identity, etc. Here the pair object comes in handy.

A pair is what the name suggests:two different value coupled together.We can make a pair of any two values:
pair<int,int> :a pair of two integers
pair<int,char>: a pair of an integer and a character
pair<float,pair<int,int> >:a pair of a float and another pair

How to make a pair:

```
pair<int,int> p=make_pair(72,37); //using make_pair() function
```

Or

```
pair<int,int> p;p.first=5;p.second=3;//using first and second
```

Note:Pairs have prewritten comparisons.It goes like this:

```
bool cmp(pair<int,int> a,pair<int,int> b)
{
if(a.first!=b.first)return a.first<b.first;
return a.second<b.second;
}
```

Vector

The simplest STL container is vector. Vector is just an array with extended functionality. By the way, vector is the only container that is backward-compatible to native C code – this means that vector actually IS the array, but with some additional features.

```
vector<int> v(10); //Creates a vector of size 10
 for(int i = 0; i < 10; i++) {
     v[i] = (i+1)*(i+1);   //once vector is created we can access elements as we did with a normal array
 }
```

1.vector<int> v; creates an empty vector.You will have to resize later like v.resize(n).
v.resize(n,0) creates a vector of n elements with each element set to 0.
2.vector<int> v[10] doesn't create a vector with 10 elements.It creates 10 empty vectors.
3.To add elements to the back you can use v.push_back(value);
To delete elements from the back you can use v.pop_back();
4.Vectors are 0 indexed. To check the length of the vector you use v.size()
5.To empty a vector you can do v.clear();
6.To access the last element you can use v.back(). Similarly you can use v.front()

2D vectors

Basically a 2D vector is a vector of vectors.Its analogous to a list of linked-lists.Now,normally using 2D arrays will do if we know the number of rows and columns(or the maximum limit on the number of rows and columns).
However,when we need to implement some functionality in which we need N lists of different lengths,making a 2D array is not feasible.Here we can use a vector of vectors.For example:

```
vector<vector<int> > mat;//now it is still empty
mat.resize(N);//Now we have got N empty lists
```

Now if we wanted to add 3 to the 5 th list and 4 to the 2nd list we would do:

```
mat[5].push_back(3);
mat[2].push_back(2);
```

And if we wanted to delete the last element of the 3rd list

```
mat[3].pop_back();
```

Strings

We have all implemented strings as char arrays in C.
In C++ you can do all that you did in C. However a much better alternative exists in the form of the string class.
Strings are quite easily used in C++:

```
string s="Hello"; //creates a string of length 4
s+=" World";      //Appends " World" to s
s="ab"+s;         //Adds "ab" to the beginning of s
```

In C++ ,a string is implemented as a vector. So you can resize it,clear it,sort its elements,etc.
Note:
1.You can make an array of strings : `string a[100];//array of 100 strings`
2.You can use cin to read strings: `cin>>s;//can't use scanf("%s",s);`
3.Like pair,strings also have predefined comparisons:
In C++,strings are compared lexicographically,i.e,in dictionary order.

Stack/Queue

These are popular applications of a standard linked list.These are pre-written in STL.

STACK: LIFO

```
stack<int> st;//Stack having integer values
st.push(10);//pushes value 10
st.top();//returns top of stack
st.pop();//pops top off stack
st.empty();//returns whether stack is empty or not
```

QUEUE:FIFO

```
queue<int> q;//Queue having integer values
q.push(10);//pushes value 10 to back of queue
q.front();//returns value at top of queue
q.pop();//pops front off queue
q.empty();//returns if  queue is empty or not
```

Iterators

We have studied various containers now and how to add/remove elements from them. However we don't yet know how to access elements within the containers according to our needs . Here , a particular STL object : iterators come in handy . Iterators can basically be visualised as pointers to elements in a container but with much more functionality and are much easier to use (they aren't as bad as the ESC101 pointers). Any STL container , be it a vector/set/map can be traversed using a pointer . The functions of iterators are:

- get value of an iterator, int x = *it;
- increment and decrement iterators it1++, it2−-; Note that it1++ not always same as it1 = it1 + 1
- compare iterators by '!=' and by '<'
- add an immediate to iterator it += 20; <=> shift 20 elements forward
- get the distance between iterators, int n = it2-it1;

Every STL container has pointers begin() and end() , which denote the beginning of the container and the end of the container. Also , one important point to remember while using iterators is that we can't use < or > operators , only == and != operators for comparison.

Iterators(continued)

Here we can examine some code using iterators to reverse a vector of integers.

```
vector<int> v;
//take input for v
//Actual Reversing starts here
vector<int>::iterator l=v.begin();//iterator to the beginning of  the vector
vector<int>::iterator r=v.end();//iterator to end of vector
if(l!=r)//if l==r it means that vector is empty
{       r--;
        while(true)
        {       swap(*l,*r);
                l++;
                if(l==r)break;//Even length of array case
                r--;
                if(l==r)break;//Odd length of array case
        }
}
```

Conclusion

We have by far not covered the vast uses of STL. It is advised that soon after this lecture you review this topcoder tutorial:
https://www.topcoder.com/community/data-science/data-science-tutorials/power-up-c-with-the-standard-template-library-part-1/
Also , we have not covered std::set and std::map in this lecture but it is comprehensively covered in the link above . STL is used a lot elsewhere , but is used a lot in Graph Theory which we will be seeing next .
Some STL centric problems are:
1.https://www.codechef.com/problems/CHEFSQUA/ (uses maps)
2.https://www.codechef.com/LTIME17/problems/CSS2/ (uses maps)
3.http://codeforces.com/contest/527/problem/C/ (uses sets)
4.https://www.codechef.com/COOK51/problems/ANUMLA/ (uses priority_queue/set)
Using STL can help simplify many problems , and many containers can be used as subroutines as a part of a larger more complex algorithm . Now , we will examine graphs , wherein STL is used abundatly.