STL(string, vector, pair)

Prerequisites:

1. Ability to write a basic C++ program
2. I/O
3. Arrays

String:

INTRODUCTION:

A string is a sequence of characters enclosed by “”. Characters can be numbers, alphabets (lowercase and uppercase) as well as special symbols on the keyboard like $,%,!,@,* etc. Whenever we want to deal with text we have to use a string.

ascii symbols

Let us look at some operations that we can perform on strings. These are provided by C++ STL string class.

DECLARING A STRING:

Syntax:

string <name of string>;

Eg:

string S;

If you create this inside a function (i.e. in stack memory), the

INITIALISING A STRING:

Syntax:

<name of string> = <some text enclosed by “”> ;

Eg:

S = “hello”;

SIMULTANEOUS DECLARATION AND INITIALISATION OF STRINGS:

Syntax:

string <name of string> = <some text enclosed by “”>

Eg:

string S = “hello”;

INPUTTING A STRING:

Eg:

Read the string until space

```
string S;
cin>>S;
```

**This has already been covered in the tutorial on I/O in detail.

LENGTH OF A STRING:

Say you have accepted the string S from the user and want to know the length of the string entered by the user. How do you find the length of the string?

Syntax:

General Syntax of size() function:

<string name>.size()

Eg:

```
int L = S.size();
```

CHECKING WHETHER A STRING IS EMPTY OR NOT:

Syntax:

```
if(<string name>.empty())
{
	cout<<”Empty String”<<endl;
}
else
{
	cout<<”Not an empty String”<<endl;
}
```

Eg:

```
if(S.empty())
{
	cout<<”Empty String”<<endl;
}
else
{
	cout<<”Not an empty String”<<endl;
}
```

ACCESSING A CHARACTER AT A PARTICULAR POSITION IN THE STRING:

Say we want to access the second(index 1) character in string S: Remember we get a character which is of char data type and not string.
General Syntax:
<string name>[index]
Eg:
char x = S[1];

*****Remember second character means index 1! Indexing starts from 0!

Exercise:
Accept a String from the user in upper case and count the number of vowels.
***Note:
How do we check whether char x is equal to 'A' or not?
It is as simple as:
if(x=='A')cout<<"YES\n";
Don't forget the single quotes surrounding A. Remember a char data type stores a single character enclosed by ''(single quotes).
Solution: https://ideone.com/BtGS7v

JOINING STRINGS:

Say we have a string S. We want to concatenate (or join) S to another string P = "hello";
How do we concatenate the two strings?
Syntax:
P = P+S;
OR
P +=S;
If S = "abcd"
then after the above operation we have P = "helloabcd".
***You can join two or more than two strings using the + operator.

If you need to append one character at the end, use .pu

SWAPPING THE VALUES OF TWO STRINGS:

Say we have a string S = "abcd" and another string P = "hello". Now we want to swap the values of S and P i.e we want to have S = "hello" and P = "abcd"
It turns out that we have a function for this as well.
Eg:
S.swap(P);
or
P.swap(S);
**Either will work!

You can swap any containers and integral types,but t

ERASING A CERTAIN PORTION OF THE STRING:

Many a time you will require to erase a portion of the string.
Let's say we have a string S = "Hello World";
And we want to make S = "Hello";
We write:
S.erase(5,6); //"Hello World"
// ^^^^^^

You should understand the COMPLEXITY of this c

The first parameter says that we start erasing from the character at index 5 and second parameter says that we erase 6 characters starting with character at index 5.

General Syntax:
<string name>.erase(starting index(i),number of characters to be erased(x));
The above command will:
Erase x characters starting from index i in <string name>.

CHECK FOR EQUALITY OF STRINGS:

How to check whether two strings S and P are equal or not?
This is simple. You use the == operator.
Eg:

```
if(S==P)
{
        cout<<"Same Strings";
}
else
{
         cout<<"Not same strings";
}
```

CONCLUSION:
These are just the basic functions , enough for you to get started in Competitive Programming. There are tons of other functions which you may check out here:
http://www.cplusplus.com/reference/string/string/

Vector:

INTRODUCTION:
Vectors are resizeable arrays in C++. We can add elements at the end. We can erase elements from the end. We can erase any number of elements from a particular index. We can insert elements after any element etc.

DECLARING A VECTOR:
Syntax:

You should know the complexity of all these operations: - push

vector<data type> <Vector Name>;

You can replace int with any type:vect

Eg: vector <int> S;

This would create a vector S in which we can only insert values of datatype int.

INSERTING AN ELEMENT AT THE END OF THE VECTOR:

Syntax:
<Vector name>.push_back(element);
Eg:
S.push_back(5);
This would push the value 5 at the end of the vector S.

DELETING THE LAST ELEMENT OF THE VECTOR:

Syntax:
<vector name>.pop_back();
Eg:
S.pop_back();

pop_back DOES NOT RETURN ANY VALUE!!!

SIZE OF THE VECTOR:
At some point how do we find out the number of elements present in the vector?
Syntax:
int L = <vector name>.size();
Eg:
int L = S.size();
This will give us the number of elements present in the vector S.

ACCESSING AN ELEMENT IN A VECTOR:
Let us now see how to access element at a particular index.
Suppose we want to access the 0th indexed element in vector S., we write:
General Syntax:
int var = <Name of vector>[index of the element to be accessed];
Eg:
int value = S[0];

ERASING AN ELEMENT AT A PARTICULAR INDEX:
Let's now see how to erase an element at a particular index.
Say we want to erase the 5th indexed element in vector S.
Syntax:
S.erase(S.begin()+5);

This has linear complexity and is NOT used quite often in comtetitive

Note: Now what is S.begin()? It is a pointer pointing to the address of the first element in vector S in the computer's memory. The +5 means from that address pointed by the pointer move 5 elements to the right(in the memory) and this is the element to be erased.
Refer to this tutorial for an in depth understanding of pointers/iterators:
https://drive.google.com/open?id=0B4AmxgIIrh_SUmxtTW5qVUdvaUU

General Syntax:
If you want to erase the ith indexed element in a vector V, we write:
V.erase(V.begin()+i);

ERASING A RANGE OF ELEMENTS FROM A VECTOR: Same as previous section. Skip it for now

Let's see how do we erase a range of elements in a vector S.
If you want to erase elements indexed L to R, we write:

S.erase(S.begin()+L,S.begin()+R+1);

Notice that we add 1 to the second parameter. This is because the function erases elements starting from the address pointed by the first parameter but not including the address pointed by the second parameter.

Exercise:
Accept 10 numbers from the user and erase the first 5 elements.
Hint: What are the indices of the first 5 elements. Plug it into our formula.
Solution: https://ideone.com/Y00TvP

INSERTING AN ELEMENT AT A PARTICULAR INDEX: Same as previous section. Skip it for now
Say we have the following vector:
1, 2, 3, 4 , 6.
And we want to insert 5, at the 4th index(i.e after the 3rd index).

Our syntax is:
S.insert(S.begin()+4, 500);

Again S.begin() is the reference to the memory location of the first element of the vector. +i indicates move right i addresses and insert the element there.
Remember: -i would indicate move left i addresses and insert the element there.

Exercise:
Accept 10 numbers from the user. After that accept 2 numbers from the user and insert these two numbers at indices 1 and 2 respectively.
Solution: https://ideone.com/7mDZAm

DELETING THE ELEMENTS IN A VECTOR:

Many a time you need to clear the contents of a vector.
Syntax:
<vector name>.clear();

Eg:
S.clear();  now S is empty
This will clear the contents of S and make it empty

CHECKING WHETHER A VECTOR IS EMPTY OR NOT:

Syntax:

```
if(<vector name>.empty()==true)
{
        cout<<"Vector Empty"<<endl;
}
else
{
        cout<<"Not empty!";
}
```

Eg:

```
if(S.empty()==true)
{
        cout<<"Vector Empty"<<endl;
}
else
{
        cout<<"Not empty!";
}
```

 or just if(S.empty())

CHANGING THE VALUE AT A PARTICULAR INDEX:

How do we change value at a particular index(say ith index)?
<vector name>[i]=<changed value>;
Eg:
S[1]=5;
***This will work only if the vector has at least 2 elements. In other words if you try changing the value of an index which is out of range, you will get an error.

CONCLUSION:

This is all you need to know about vectors to get started. There are many more functions. You may refer here to learn these functions as and when required:
http://www.cplusplus.com/reference/vector/vector/

Pair:

INTRODUCTION:
Many a time you will find that you have to keep track of two values. In such a case you may use two variables but that would lead to your data being scattered(just because the two values are related to each other, it would be useful to have them tied together).

MOTIVATION PROBLEM:
Let us consider the following problem:
You have a class of 10 students. You have to input the details of the students. Details include roll number of the student(in the range of 1 to 10) and marks in Computer Science.
How will you store them?
Well you can create two separate arrays, one array to store the roll number and the other array to store the marks but that doesn't looks good. The data seems to be scattered. Wouldn't it be better if we had an array containing elements and each element of this array could hold the two details of each student together? Think of it this way. You have an array and each element of the array is a container. I can put the two details of each student in a container. This way the data is much better organised. Each container belongs to a unique student. We seem to have an option to tie the data together like this by using pairs provided by C++ STL.
Pairs helps us to store two values together. You can think of it as it ties the two values together.

DECLARING/CREATING A PAIR:
Syntax:
pair <datatype of first element , datatype of second element> name;
Eg:
pair<int,int> P;
This will create a pair P which will be able to store an int value in the first place as well as in the second place.

INITIALIZING A PAIR:
Syntax:
<name of pair> = make_pair(value 1, value 2);
**Make sure value 1 and value 2 are of the same datatype as {datatype of first element} and {datatype of second element}
Eg:
P = make_pair(13,2);

Or just declare and initialize in 1 row:#include <utility>...pair<in

ACCESSING THE VALUE OF THE FIRST ELEMENT IN THE PAIR:

It is as simple as writing:

int X = <name of pair>.first;

Eg:

int X = P.first;

This should print 13.(Continuing our example from above)

ACCESSING THE VALUE OF THE SECOND ELEMENT IN THE PAIR:

Syntax:

int X = <name of pair>.second;

int X = P.second;

CONCLUSION:

Pairs are really useful with maps which you will learn later. These help us to keep track of data uniquely identified by two variables.

You can also return pairs from functions;Useful when

Tricks like pair of pairs is also possible:pair<int, pair<i