C++ Templates:

* What is template in C++ Programming?

A template is believed to escalate the potential of C++ several fold by giving it the ability to define data types as parameters making it useful to reduce repetitions of same declaration of classes for different data types.

Declaring classes for every other data types(which if counted is way too much) in very first place violets the DRY(Don’t Repeat Yourself) rule of programming and on the other doesn’t completely utilise the potential of C++.

It is very analogous to when we said classes are the templates for objects, here templates itself are the templates of the classes. That is, what classes are for objects, templates are for classes.

* Why Templates?

1.DRY Rule:

To understand the reason behind using templates , we will have to understand the effort behind declaring classes for different data types. Suppose we wants to have a vector for each of the three (can be more) data types ,int, float and char. Then we will obviously write the whole things again and again making it awfully difficult. This is where the saviour comes, the templates . it helps parametrizing the data type and declaring it once in the source code suffice . Very similar to what we do in functions .It is because of this , also called ,’parameterized classes'.

2.Generic Programming:

It is called generic, because it is sufficient to declare a template once, it becomes general and it works all along for all the data types.

Class Vector

{

Int \*arr;

Int size;

Public;

}

Class Vector

{

double\*arr;

Int size;

Public;

}

Class Vector

{

float \*arr;

Int size;

Public;

}

Class Vector

{

char \*arr;

Int size;

Public;

}

We had to copy the same thing again and again for different data types, but a template solves it all.

SYNTAX:

Understanding the syntax below:

1. First, we declare a template of class and pass a variable T as its parameter.
2. Define the class of vector and keep the data type of \*arr as T only. Now, the array becomes of the type we supply in the template.

Now we can easily use this template to declare umpteen number of classes in our main scope. Be it int, float, or arr vector .();

#include <iostream>

using namespace std;

template <class T>

class vector

{

T \*arr;

int size;

public:

vector(T\* arr)

[

//code

]

//and many other methods

};

int main()

{

vector<int> myVec1();

vector<float> myVec2();

return 0;

}