**C++ Class Templates**

Class templates are one application of templates. Class templates could be used according to the requirements of the programmer. Templates are nothing but shortened generic declarations of similar entities. They are expanded at the compile time.

**Examples of some of the in-built class templates are,**

Vector, Set, Linked List etc.

Class templates are helpful for classes that are independent of the data type. And like any other function, there could be more than one parameter to a template. On the basis of the number of parameters of a template, they could be single parameters as well as multiple parameters.

**A. Single parameter**

Syntax for declaring a single parametrized template is,

#include <iostream>

using namespace std;

template <class T>

class nameOfClass

{

//body

};

int main()

{

//body of main

}

The example of the vector easily demonstrates how templates are used with a single parameter.

#include <iostream>

using namespace std;

template <class T>

class vector

{

T \*arr;

int size;

};

int main()

{

vector<int> v1();

vector<float> v2();

}

**B. Multiple Parameter**

Syntax for declaring a multiple parametrized template is,

#include <iostream>

using namespace std;

template <class T1, class T2>

class nameOfClass

{

//body

};

int main()

{

//body of main

}

The difference lies only in the number of parameters we declare inside the template. Consider an example that demonstrates the use of multiple parameters in a class template.

#include <iostream>

using namespace std;

template <class T1, class T2>

class myClass

{

public:

T1 data1;

T2 data2;

myClass(T1 a, T2 b)

{

data1 = a;

data2 = b;

}

void display()

{

cout << this->data1 << " " << this->data2;

}

};

int main()

{

myClass<char, int> obj('C', 1);

obj.display();

}