➢ Class Template Declaration

A class template starts with the keyword template followed by template parameter(s) inside <> which is followed by the class declaration.

```
template <class T>
class className
{
  private:
    T var;
    ... .. ...
  public:
    T functionName(T arg);
    ... .. ...
};
```

In the above declaration, T is the template argument which is a placeholder for the data type used, and class is a keyword.
Inside the class body, a member variable var and a member function functionName() is both of type T.

➢ Creating a Class Template Object

Once we've declared and defined a class template, we can create its objects in other classes or functions (such as the main() function) with the following syntax:

```
className<dataType> classObject;
```

For example,

```
className<int> classObject;
className<float> classObject;
className<string> classObject;
```

Example 1: C++ Class Templates

```
#include<iostream>
using namespace std;
template<class T>
class A
{
   public:
            T num1 = 5;
            T num2 = 6;
   void add()
   {
      std::cout<<"Addition of num1 and num2 : "<<num1+num2<<std::endl;
   }

};

int main()
{
   A<int> d;
   d.add();
   return 0;
}
```

Ex2: // C++ program to demonstrate the use of class templates

```
#include <iostream>
using namespace std;

// Class template
template <class T>
class Number {
  private:
          // Variable of type T
          T num;

  public:
   Number(T n) : num(n) {}   // constructor

   T getNum() {
      return num;
```

```
    }
};

int main() {

    // create object with int type
    Number<int> numberInt(7);

    // create object with double type
    Number<double> numberDouble(7.7);

    cout << "int Number = " << numberInt.getNum() << endl;
    cout << "double Number = " << numberDouble.getNum() << endl;

    return 0;
}
```

➢ Function Templates with Multiple Parameters

You can also use multiple parameters in your function template.

Syntax:-

```
template<class T1, class T2,.....>

return_type functionName (arguments of type T1, T2....)  {

    // body

}
```

Ex. // CPP program to illustrate Class template with multiple parameters

```
#include<iostream>
using namespace std;

// Class template with two parameters
template<class T1, class T2>
class Test
{
        T1 a;
        T2 b;
    public:
        Test(T1 x, T2 y)
        {
            a = x;
            b = y;
        }
        void show()
        {
            cout << a << " and " << b << endl;
        }
};




// Main Function
int main()
{
    // instantiation with float and int type
    Test <float, int> test1 (1.23, 123);

    // instantiation with float and char type
    Test <int, char> test2 (100, 'W');

    test1.show();
    test2.show();

    return 0;
}
```

Example : - Class template with multiple parameters

```
#include <iostream>
using namespace std;
template<class T1, class T2>
class Adder
{
        private:
                T1 x;
                T2 y;
        public:
                Adder(T1 x, T2 y)
                {       this->x = x;
                        this->y = y;
                }
                void add()
                {
                cout<<"Sum is: "<<(x+y)<<endl;
                }
};
int main()
{
        Adder<int,int> a1(3, 5);
        a1.add();
        Adder<int,double> a2(2, 5.3);
        a2.add();
        return 0;    }
```