Templates

Definition

- Templates in C++ is an interesting feature that is used for generic programming. Generic Programming is an approach of programming where generic types are used as parameters in algorithms to work for a variety of data types.
- It is defined as a blueprint or formula for creating a generic class or a function. To simply put, you can create a single function or single class to work with different data types using templates.
- It works in such a way that it gets expanded at compiler time, just like macros and allows a function or class to work on different data types without being rewritten.

Types of Template

There are two types of templates in C++

- **Function template**
- **Class template**

Function Template

- Function template in c++ is a single function template that works with multiple data types simultaneously, but a standard function works only with one set of data types.

C++ Function Template Syntax -

```
template<class type>ret-type func-name(parameter list)
{
//body of the function
}
```

Here, type is a placeholder name for a data type used by the function. It is used within the function definition.

The class keyword is used to specify a generic type in a template declaration.

Example

```
#include<iostream.h>
using namespace std;
template<classX>        //can replace 'class" keyword by
                        "typename" keyword
X func( Xa,Xb) {
 return a;
}
int main()
{
count<<func(15,8),,endl;                //func(int,int);
    count,,func('p','q'),,endl;         //func(char,char);
    count<<func(7.5,9.2),,endl;         //func(double,double)
    return();
}
```

Class Template

The class template in c++ is like function templates. They are known as generic templates. They define a family of classes in C++.

Syntax of Class Template

```
template<class Ttype>
class class_name
{
//class body;
}
```

Here Ttype is a placeholder type name, which will be specified when a class instantiated.

The Ttype can be used inside the body of the class.

Example

```
#include<iostream.h>
using namespace std;
template <class C> class A {
 private:
 C a,b;
public:
A(Cx,Cy){
a=x; b=y;
}
void show() {
count<<"The Addition of"<<a<<"and"<<b<<"is"<<add()<<endl;
 }
 C add() {
C c=a+b;
return c;
 }
};
 int main(){
A addint(8,6);
A addfloat(3.5,2.6);
A aaddouble(2.156,5.234);
A addint.show();
cout<<endl;
adddouble.show();
 count<<endl;
return 0;
 }
```