



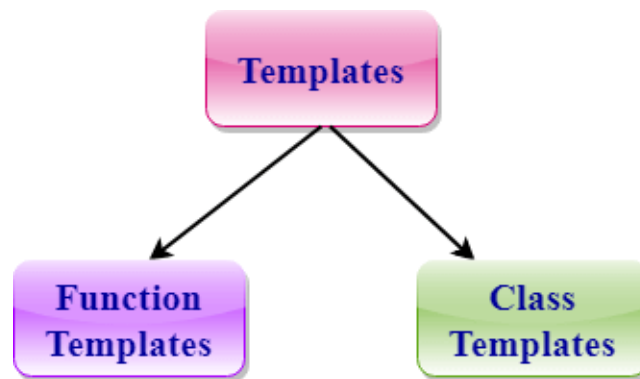
Templates: Introduction

C++ Templates

A C++ template is a powerful feature added to C++. It allows you to define the generic classes and generic functions and thus provides support for generic programming. Generic programming is a technique where generic types are used as parameters in algorithms so that they can work for a variety of data types.

Templates can be represented in two ways:

- Function templates
- Class templates



Function Templates:

We can define a template for a function. For example, if we have an add() function, we can create versions of the add function for adding the int, float or double type values.

Class Template:

We can define a template for a class. For example, a class template can be created for the array class that can accept the array of various types such as int array, float array or double array.

Function Template

- Generic functions use the concept of a function template. Generic functions define a set of operations that can be applied to the various types of data.
- The type of the data that the function will operate on depends on the type of the data passed as a parameter.
- For example, Quick sorting algorithm is implemented using a generic function, it can be implemented to an array of integers or array of floats.
- A Generic function is created by using the keyword template. The template defines what function will do.



```
template < class Ttype> ret_type func_name(parameter_list)
```

1. {
2. // body of function.
3. }

Where **Ttype**: It is a placeholder name for a data type used by the function. It is used within the function definition. It is only a placeholder that the compiler will automatically replace this placeholder with the actual data type.

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

Let's see a simple example of a function template:

```
#include <iostream>
using namespace std;
template<class T> T add(T &a,T &b)
{
    T result = a+b;
    return result;
}
int main()
{
    int i =2;
    int j =3;
    float m = 2.3;
    float n = 1.2;
    cout<<"Addition of i and j is :"<<add(i,j);
    cout<<"\n";
    cout<<"Addition of m and n is :"<<add(m,n);
    return 0;
}
```

Output:

Addition of i and j is :5
Addition of m and n is :3.5