# Standard Template Library

26 Stl CodeMonk

**C++ Templates**
Templates are a feature of the C++ programming language that allows functions and classes to operate with generic types. This allows a function or class to work on many different data types without being rewritten for each one.

What are **generic functions or classes?**
Many times while programming, there is a need for creating functions which perform the same operations but work with different data types. So C++ provides a feature to create a single generic function instead of many functions which can work with different data type by using the **template parameter**.

What is the **template parameter?**

The way we use normal parameters to pass as a value to function, in the same manner template parameters can be used to pass type as argument to function. Basically, it tells what type of data is being passed to the function.

The syntax for creating a generic function:

```
template <class  type> return-type function-name (parameter-list)
```

Here, ‘type’ is just a placeholder used to store the data type when this function is used you can use any other name instead class is used to specify the generic type of template, alternatively typename can be used instead of it.

Let’s try to understand it with an example:

Assume we have to swap two variables of int type and two of float type. Then, we will have to make two functions where one can swap int type variables and the other one can swap float type variables. But here if we use a generic function, then we can simply make one function and can swap both type of variables by passing their different type in the arguments. Let’s implement this:

```
#include <iostream>
using namespace std ;
// creating a generic function ‘swap (parameter-list)’ using templa
template <class X>
void swap( X &a, X &b) {
    X tp;
```