C++ Function Templates & Function Templates with Parameters | C++ Tutorials for Beginners #67

C++ Function Templates & Function Templates with Parameters | C++ Tutorials for Beginners #67

In this tutorial, we are wishing to learn how a function template works. Prior to this video, we have only talked about a class template and its functionalities. In class template we used to have template parameters which we, very often, addressed as a variable for our data types. We have also declared a class template similar to what shown here below :

```
template <class T1 = int, class T2 = float>
```

Today, we'll be interested in knowing what a function template does. So. let's get ourselves on our editors.

Suppose we want to have a function which calculates the average of two integers. So, this must be very easy for you to formulate. Look for the snippet below.

1. We have declared a float function named funcAverage which will have two integers as its parameters, a and b.
2. We stored its average in a float variable avg and returned the same to the main.
3. Later we called this function by value, and stored the returned float in a float variable a and printed the same.
4. So this was the small effort we had to make to get a function which calculates the average of two integers.

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

float funcAverage(int a, int b){
    float avg= (a+b)/2.0;
    return avg;
}
int main(){
    float a;
    a = funcAverage(5,2);
    printf("The average of these numbers is %f",a);
    return 0;
}
```

The output of the above program is :

```
The average of these numbers is 3.500000
PS D:\MyData\Business\code playground\C++ course>
```

But the effort we made here defining a single function for two integers increases several folds when we demand for a similar function for two floats, or one float and one integer or many more data type combinations. We just cannot repeat the procedure and violate our DRY rule. We'll use function templates very similar to what we did when we had to avoid defining more classes.

See what are the subtle changes we had to make, to make this function generic.

We'll first declare a template with two data type parameters T1 and T2. And replace the data types we mentioned in the function with them. And that's it. Our function has become general for all sorts of data types. Refer to the snippet below.

```
template<class T1, class T2>
float funcAverage(T1 a, T2 b){
    float avg= (a+b)/2.0;
    return avg;
}
```

Let's call this function by passing into it two sorts of data types combination, first, two integers and then one integer and one float. And see if the outputs are correct.

```
int main(){
    float a;
    a = funcAverage(5,2);
    printf("The average of these numbers is %f",a);
    return 0;
}
```

Code snippet: Calling the function by passing two integers

```
The average of these numbers is 3.500000
PS D:\MyData\Business\code playground\C++ course>
```

```
int main(){
    float a;
    a = funcAverage(5,2.8);
    printf("The average of these numbers is %f",a);
    return 0;
}
```

Code snippet: Calling the function by passing one integer and one float

```
The average of these numbers is 3.900000
PS D:\MyData\Business\code playground\C++ course>
```

And a general swap function named swapp for those variety of data types we have, would look something like the one below:

```
template <class T>
void swapp(T &a, T &b)
{
    T temp = a;
    a = b;
    b = temp;
```

```
}
```

So, this is how we utilize this powerful tool to avoid writing such overloaded codes. And this was all about function templates with single or multiple parameters. We covered them all in this tutorial.

Thank you, for being with me throughout, hope you liked the tutorial. If you haven't checked out the whole playlist yet, move on to codewithharry.com or my YouTube channel to access it. I hope you enjoy them all. We are now done with both class and function templates. See you all in the next tutorial where we'll see if a function template can be overloaded. Till then keep coding.

Previous Next

CodeWithHarry