C++ Templates: Templates with Multiple Parameters | C++ Tutorials for Beginners #65

C++ Templates: Templates with Multiple Parameters | C++ Tutorials for Beginners #65

In the last tutorial, we had ample understanding of a template and its uses. We had created a template which would calculate the Dot Product of two vectors of any data type just by declaring a simple template parameterizing the data type we usually hardcoded in the classes. This already made our task easier but here we are, with our next tutorial focusing on how to handle multiple parameters in a template.

To give you a short overview of how templates work with multiple parameters, you can think of it as a function where you have that power to pass different parameters of the same or different data types. A simple template with two parameters would look something like this. The only effort it demands is the declaration of parameters.

We'll get through it thoroughly by making a real program, so, let's go.

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

/*
template<class T1, class T2>
class nameOfClass{
    //body
}
*/

int main(){
    //body of main
}
```

Code Snippet 1: Syntax of a template with multiple parameter

Suppose we have a class named myClass which has two data in it of data types int and char respectively, and the function embedded just displays the two. Fair enough, no big deal, we'll construct our class something like this. The problem arises when we wish to have both our data types anonymous and to be put from the main itself. You will be surprised to know that very subtle modifications in yesterday's code would do our task. Instead of declaring a single parameter T, we would declare two of them namely T1 And T2.

```
class myClass{
    public:
        int data1;
```

```
        char data2;
    void display(){
        cout<<this->data1<<" "<<this->data2;
    }
};
```

Code Snippet 2: Constructing a class

Refer to changes we have done below to parametrize both our data types using a single template:

1. We have declared data1 and data2 with data types T1 and T2 respectively.
2. We have applied the constructor filling the values we receive from the main into data1 and data2.
3. Finally, we have displayed both of them.

```
template<class T1, class T2>
class myClass{
    public:
        T1 data1;
        T2 data2;
        myClass(T1 a,T2 b){
            data1 = a;
            data2 = b;
        }
    void display(){
        cout<<this->data1<<" "<<this->data2;
    }
};
```

Code Snippet 2: Constructing a template with two parameters.

Let me now show you how this template works for different parameters. I'll pass different data types from the main and see if it's flexible enough.

Firstly, we put an integer and a char,

```
int main()
{
    myClass<int, char> obj(1, 'c');
    obj.display();
}
```

Code Snippet 3: Specifying the data types to be int and char.

And the output received was this, which is correct. Let's feed another one.

```
1   c
PS D:\MyData\Business\code playground\C++ course>
```

Figure 1: Output of code snippet 3.

Now we put an integer and a float,

```
int main()
{
    myClass<int, float> obj(1,1.8 );
    obj.display();
}
```

Code Snippet 4: Specifying the data types to be int and float.

And the output received was this,

```
1   1.8
PS D:\MyData\Business\code playground\C++ course>
```

Figure 1: Output of code snippet 4.

So yes, this is functioning all good.

And this was all about templates with multiple parameters, just don't miss out the commas while defining the parameters in a template. And you can have 2, 3 or more of them according to your needs. Could you believe how luxurious it has become to work with customized data types? It is now you, who'll decide what the data type of some variable in a class should be. It is no longer pre-specified. It has given you some unimaginable power which, if you realise, can save you a lot of energy and time.

Thank you, friends 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. In the next tutorial, we'll be learning about having a default parameter in a template, see you there, till then keep coding.

Previous Next