C++ Templates: Class Templates with Default Parameters | C++ Tutorials for Beginners #66

So far, we have already covered the C++ templates with single parameters. In the last tutorial, we learnt about templates with multiple parameters, when it comes to handling different data types of two or more containers. Today, we'll be learning a very easy yet powerful attribute of templates, its ability to have default parameters. Its ability to have default specifications about the data type, when it receives no arguments from the main.

So, let's start by making a program manifesting the use of default parameters in a C++ template. **Refer to the code snippet below and follow the steps:**

1. We'll start by constructing a class named Harry.
2. We'll then define a template with any number of arguments, let three, T1, T2, and T3. If you remember, we had this feature of specifying default arguments for functions, similarly we'll mention the default parameters, let, int, float and char for T1, T2 and T3 respectively.
3. This ensures that if the user doesn't put any data type in main, default ones get considered.
4. In public, we'll define variables a, b and c of the variable data types T1, T2 and T3. And build their constructors.
5. The constructor accepts the values featured by the main, and assigns them to our class variables a, b and c. If the user specifies the data types along with the values, the compiler assigns them to T1 , T2 and T3, otherwise gives them the default ones, as specified while declaring the template itself.
6. We'll then create a void function display, just to print the values the user inputs.

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

template <class T1=int, class T2=float, class T3=char>
class Harry{
    public:
        T1 a;
        T2 b;
        T3 c;
        Harry(T1 x, T2 y, T3 z) {
            a = x;
            b = y;
            c = z;
        }
        void display(){
```

```
            cout<<"The value of a is "<<a<<endl;
            cout<<"The value of b is "<<b<<endl;
            cout<<"The value of c is "<<c<<endl;
        }
};
```

Since we are done defining the templates and class, we can very easily move to the main where we'll see how these work. **Understanding code snippet 2:**

1. Firstly, we'll create an object, let's name it h, of the class Harry. And we'll pass into it three values, an int, a float and a char, suppose 4, 6.4 and c respectively. Now since we have not specified the data types of the values we have just entered, the default data types, int, float and char would be considered.
2. We'll then display the values, which you'll be seeing when we run the same.
3. And then we'll create another object g, of the class Harry but this time, with the data types of our choice. Let's specify them to be float, char and char.
4. We can then pass some values into it, suppose 1.6, o, and c and call the display function again.
5. These objects are sufficient to give us the main concept behind using a default parameter and the variety of classes we could make via this one template.

```
int main()
{
    Harry<> h(4, 6.4, 'c');
    h.display();
    cout << endl;
    Harry<float, char, char> g(1.6, 'o', 'c');
    g.display();
    return 0;
```

```
}
```

We'll now refer to the output the above codes combinedly gave. As you can see below, it worked all fine. Had we not specified the default parameters; the above program would have thrown an error. Thanks to this feature of C++ templates.

```
The value of a is 4
The value of b is 6.4
The value of c is c

The value of a is 1.6
The value of b is o
The value of c is c
PS D:\MyData\Business\code playground\C++ course>
```

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 all. So far, we were learning about the class templates. See you all in the next tutorial where we'll look after the **function templates**. Till then keep coding.

Previous Next