Default Arguments in C++

A default argument is a value provided in a function declaration that is automatically assigned by the compiler if the calling function doesn't provide a value for the argument. In case any value is passed, the default value is overridden.

1) The following is a simple C++ example to demonstrate the use of default arguments. Here, we don't have to write 3 sum functions; only one function works by using the default values for 3rd and 4th arguments.

```
// CPP Program to demonstrate Default Arguments
#include <iostream>
using namespace std;

// A function with default arguments,
// it can be called with
// 2 arguments or 3 arguments or 4 arguments.
int sum(int x, int y, int z = 0, int w = 0) //assigning defau
{
    return (x + y + z + w);
}

// Driver Codeint main()
{
    // Statement 1
    cout << sum(10, 15) << endl;

    // Statement 2
    cout << sum(10, 15, 25) << endl;

    // Statement 3
    cout << sum(10, 15, 25, 30) << endl;
    return 0;
}
```

Output

```
25
```

```
50
80
```

Explanation: In statement 1, only two values are passed, hence the variables z and w take the default values of 0. In statement 2, three values are passed, so the value of z is overridden with 25. In statement 3, four values are passed, so the values of z and w are overridden with 25 and 30 respectively.

2) If function overloading is done containing the default arguments, then we need to make sure it is not ambiguous to the compiler, otherwise it will throw an error. The following is the modified version of the above program:

```
// CPP Program to demonstrate Function overloading in
// Default Arguments
#include <iostream>
using namespace std;

// A function with default arguments, it can be called with
// 2 arguments or 3 arguments or 4 arguments.
int sum(int x, int y, int z = 0, int w = 0)
{
    return (x + y + z + w);
}
int sum(int x, int y, float z = 0, float w = 0)
{
    return (x + y + z + w);
}
// Driver Codeint main()
{
    cout << sum(10, 15) << endl;
    cout << sum(10, 15, 25) << endl;
    cout << sum(10, 15, 25, 30) << endl;
    return 0;
}
```

Error:

```
prog.cpp: In function 'int main()':
prog.cpp:17:20: error: call of overloaded
'sum(int, int)' is ambiguous
  cout << sum(10, 15) << endl;
                   ^
prog.cpp:6:5: note: candidate:
int sum(int, int, int, int)
 int sum(int x, int y, int z=0, int w=0)
     ^
prog.cpp:10:5: note: candidate:
int sum(int, int, float, float)
 int sum(int x, int y, float z=0, float w=0)
```

3) A constructor can contain default parameters as well. A default constructor can either have no parameters or parameters with default arguments.

```
// CPP code to demonstrate use of default arguments in
// Constructors
#include <iostream>
using namespace std;

class A {
    private:
        int var = 0;
    public:
        A(int x = 0): var(x){}; // default constructor with o
            var = s; // OR => this->var = s;
            return;
        }
        int getVar(){
            return var; // OR => return this->var;
        }
};
int main(){
    A a(1);
```

```
    a.setVar(2);

    cout << "var = " << a.getVar() << endl;

    /* ANOTHER APPROACH:
    A *a = new A(1);

    a->setVar(2);

    cout << "var = " << a->getVar() << endl;
    */
}
```

Explanation: Here, we see a default constructor with no arguments and a default constructor with one default argument. The default constructor with the argument has a default parameter x, which has been assigned a value of 0.

Key Points:

- Default arguments are different from constant arguments as constant arguments can’t be changed whereas default arguments can be overwritten if required.
- Default arguments are overwritten when the calling function provides values for them. For example, calling the function sum(10, 15, 25, 30) overwrites the values of z and w to 25 and 30 respectively.
- When a function is called, the arguments are copied from the calling function to the called function in the order left to right. Therefore, sum(10, 15, 25) will assign 10, 15, and 25 to x, y, and z respectively, which means that only the default value of w is used.
- Once a default value is used for an argument in the function definition, all subsequent arguments must also have a default value. It can also be stated that the default arguments are assigned from right to left. For example, the following function definition is invalid as the subsequent argument of the default variable z is not default.

```
// Invalid because z has default value, but w after it does
n't have a default value
int sum(int x, int y, int z = 0, int w).
```

Advantages of Default Arguments:

- Default arguments are useful when we want to increase the capabilities of an existing function as we can do it just by adding another default argument to the function.
- It helps in reducing the size of a program.
- It provides a simple and effective programming approach.
- Default arguments improve the consistency of a program.

Disadvantages of Default Arguments:

- It increases the execution time as the compiler needs to replace the omitted arguments by their default values in the function call.