Default Constructors in C++

Difficulty Level : Easy • Last Updated : 16 Mar, 2023

A constructor without any arguments or with the default value for every argument is said to be the **Default constructor**.

A constructor that has zero parameter list or in other sense, a constructor that accept no arguments is called a zero argument constructor or default constructor.

If default constructor is not defined in the source code by the programmer, then the compiler defined the default constructor implicitly during compilation.

If the default constructor is defined explicitly in the program by the programmer, then the compiler will not defined the constructor implicitly, but it calls the constructor implicitly.

What is the significance of the default constructor?

They are used to create objects, which do not have any specific initial value.

Is a default constructor automatically provided?

If no constructors are explicitly declared in the class, a default constructor is provided automatically by the compiler.

Can a default constructor contain a default argument?

Yes, a constructor can contain default argument with default values for an object.

Will there be any code inserted by the compiler to the user implemented default constructor behind the scenes?

The compiler will implicitly declare the default constructor if not provided by the programmer, will define it when in need. The compiler-defined default constructor is required to do certain initialization of class internals. It will not touch the data members or plain old data types (aggregates like an array, structures, etc…). However, the compiler generates code for the default constructor based on the situation.

Consider a class derived from another class with the default constructor, or a class containing another class object with the default constructor. The compiler needs to insert code to call the default constructors of the base class/embedded object.

C++

```cpp
// CPP program to demonstrate Default constructors
#include <iostream>
using namespace std;

class Base {
public:
    // compiler "declares" constructor
};

class A {
public:
    // User defined constructor
    A() { cout << "A Constructor" << endl; }

    // uninitialized
    int size;
};

class B : public A {
    // compiler defines default constructor of B, and
    // inserts stub to call A constructor

    // compiler won't initialize any data of A
};

class C : public A {
public:
    C()
    {
        // User defined default constructor of C
        // Compiler inserts stub to call A's constructor
        cout << "C Constructor" << endl;

        // compiler won't initialize any data of A
    }
};

class D {
public:
    D()
    {
        // User defined default constructor of D
        // a - constructor to be called, compiler inserts
        // stub to call A constructor
        cout << "D Constructor" << endl;

        // compiler won't initialize any data of 'a'
    }

private:
    A a;
};

// Driver Code
int main()
{
    Base base;

    B b;
    C c;
    D d;

    return 0;
}
```

Output

```
A Constructor
A Constructor
C Constructor
A Constructor
D Constructor
```

C++

```
Example:
#include<iostream>
using namespace std;
class student
{
    int rno;
    char name[50];
    double fee;
    public:
    student()                     //  Explicit Default constructor
    {
        cout<<"Enter the RollNo:";
        cin>>rno;
        cout<<"Enter the Name:";
        cin>>name;
        cout<<"Enter the Fee:";
        cin>>fee;
    }

    void display()
    {
        cout<<endl<<rno<<"\t"<<name<<"\t"<<fee;
    }
};

int main()
{
    student s;
    s.display();
    return 0;
}
```

There are different scenarios in which the compiler needs to insert code to ensure some necessary initialization as per language requirements. We will have them in upcoming posts. Our objective is to be aware of C++ internals, not to use them incorrectly.

C++

```
// CPP code to demonstrate constructor can have default
// arguments
#include <iostream>
using namespace std;
class A {
public:
    int sum = 0;
    A(); // default constructor with no argument
    A(int a, int x = 0) // default constructor with one
                        // default argument
    {
        sum = a + x;
    }
    void print() { cout << "Sum =" << sum << endl; }
};
int main()
{
    // This construct has two arguments. Second argument is
    // initialized with a value of 0 Now we can call the
    // constructor in two possible ways.
    A obj1(10, 20);
    A obj2(5);
    obj1.print();
    obj2.print();
    return 0;
}
```

Output

```
Sum =30
Sum =5
```

Explanation : Here, we have a constructor with two parameter- simple parameter and one default parameter. Now, there are two ways of calling this constructor:

1. First, we can assign values to both the arguments and these values will be passed to the constructor and the default argument x with value 0 will be overridden by value passed while calling (in this case 20). Hence, code will give an output of 30 (as, sum= a+x i.e 10+20= 30).
2. Second way is to not pass any value for the default parameter. If you do so, x will take it's default value 0 as it's final value and calculate a sum of 5 (as, sum = a+x i.e 5+0=5).

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

77

Related Articles

1. When Does Compiler Create Default and Copy Constructors in C++?
2. Inheritance and Constructors in Java
3. When are Constructors Called?
4. Constructors in Java
5. C++ | Constructors | Question 2
6. C++ | Constructors | Question 4
7. C++ | Constructors | Question 5