Code Example: Constructors in Derived Class in Cpp | C++ Tutorials for Beginners #48

In this tutorial, we will discuss constructors in derived class with code example in C++

Constructors in Derived Class in C++

As we have discussed before about the constructors in derived class in a code snippet below three cases are given to clarify the execution of constructors.

```
/*
Case1:
class B: public A{
   // Order of execution of constructor -> first A() then B()
};

Case2:
class A: public B, public C{
    // Order of execution of constructor -> B() then C() and A()
};

Case3:
class A: public B, virtual public C{
    // Order of execution of constructor -> C() then B() and A()
};

*/
```

Copy

Code Snippet 1: Constructors Execution Example Cases

As shown in Code Snippet 1,

1. In case 1, class “B” is inheriting class “A”, so the order of execution will be that first the constructor of class “A” will be executed and then the constructor of class “B” will be executed.
2. In case 2, class “A” is inheriting two classes “B” and “C”, so the order of execution will be that first constructor of class “B” will be executed and then the constructor of class “C” will be executed and at the end constructor of class “A” will be executed.
3. In case 3, class “A” is inheriting two classes “B” and virtual class “C”, so the order of execution will be that first constructor of class “C” will be executed because it is a virtual class and it is given more preference and then the constructor of class “B” will be executed and at the end constructor of class “A” will be executed.

To demonstrate the concept of constructors in derived classes an example program is shown below.

```
class Base1{
    int data1;
    public:
        Base1(int i){
            data1 = i;
            cout<<"Base1 class constructor called"<<endl;
        }
        void printDataBase1(void){
            cout<<"The value of data1 is "<<data1<<endl;
        }
};

class Base2{
    int data2;
```

```
    public:
        Base2(int i){
            data2 = i;
            cout << "Base2 class constructor called" << endl;
        }
        void printDataBase2(void){
            cout << "The value of data2 is " << data2 << endl;
        }
};

class Derived: public Base2, public Base1{
    int derived1, derived2;
    public:
        Derived(int a, int b, int c, int d) : Base2(b), Base1(a)
        {
            derived1 = c;
            derived2 = d;
            cout<< "Derived class constructor called"<<endl;
        }
        void printDataDerived(void)
        {
            cout << "The value of derived1 is " << derived1 << endl;
            cout << "The value of derived2 is " << derived2 << endl;
        }
};
```

Copy

Code Snippet 2: Constructors in Derived Class Example Program

As shown in code snippet 2,

1. We have created a “Base1” class which consists of private data member “data1” and parameterized constructor which takes only one argument and set the value of data member “data1”. The “Base1” class also contains the member function “printDataBase1” which will print the value of data member “data1”.
2. We have created a “Base2” class which consists of private data member “data2” and parameterized constructor which takes only one argument and set the value of data member “data2”. The “Base2” class also contains the member function “printDataBase2” which will print the value of data member “data2”.
3. We have created a “Derived” class that is inheriting base classes “Base1” and “Base2”. The “Derived” class consists of private data members “derived1” and “derived2”. The “Derived” class contains parameterized constructor which calls the “Base1” and “Base2” class constructors to pass the values, it also assigns the values to the data members “derived1” and “derived2”. The “Derived” class also contains member functions “printDataDerived”. The function “printDataDerived” will print the values of the data member “derived1” and “derived2”.

The main thing to note here is that the constructors will be executed in the order in which the classes are being inherited. As in the example program above the “Base2” class is being inherited first and then “Base1” class is being inherited, so the constructor of “Base2” class will be executed first. The main program of the following example code is shown below.

```
int main(){
    Derived harry(1, 2, 3, 4);
    harry.printDataBase1();
    harry.printDataBase2();
    harry.printDataDerived();
    return 0;
}
```

Code Snippet 3: Main Program
As shown in code snippet 3,

1. Object “harry” is created of the “Derived” data type and the values (1, 2, 3, 4) are passed.
2. The function “printDataBase1” is called by the object “harry”.
3. The function “printDataBase2” is called by the object “harry”.
4. The function “printDataDerived” is called by the object “harry”.

The output of the following program is shown below,