Derived Class Constructors-Overriding Member Functions

Derived Class Constructors:

If the class "A" is written before class "B" then the constructor of class "A" will be executed first. But if the class "B" is written before class "A" then the constructor of class "B" will be executed first.

- A constructor plays a vital role in initializing an object. An important note, while using constructors during inheritance, is that, as long as a base class constructor does not take any arguments, the derived class need not have a constructor function.
- However, if a base class contains a constructor with one or more arguments, then it is mandatory for the derived class to have a constructor and pass the arguments to the base class constructor.
- Remember, while applying inheritance, we usually create objects using derived class. Thus, it makes sense for the derived class to pass arguments to the base class constructor.
- When both the derived and base class contains constructors, the base constructor is executed first and then the constructor in the derived class is executed.
- In case of multiple inheritance, the base class is constructed in the same order in which they appear in the declaration of the derived class.
- Similarly, in a multilevel inheritance, the constructor will be executed in the order of inheritance.
- The derived class takes the responsibility of supplying the initial values to its base class.
- The constructor of the derived class receives the entire list of required values as its argument and passes them on to the base constructor in the order in which they are declared in the derived class.
- A base class constructor is called and executed before executing the statements in the body of the derived class.
- The header line of the derived-constructor function contains two parts separated by a colon (:).
- The first part provides the declaration of the arguments that are passed to the derived class constructor and the second part lists the function calls to the base class.

Example:

```
D(int a1, int a2, int b1, int b2, int d): A(a1, a2), B(b1, b2)
{
   d = d1;
}
```

A(a1, a2) invokes the base constructor A() and B(b1, b2) invokes another base class constructor B(). The constructor D() supply the values i.e. a1, a2, b1, b2 (to A() and B()) and to one of its own variables d1.

Hence, the assignment of the values will be as follows:

When an object of D is created, D object-name(5, 12, 7, 8, 30);

a1 <- 5 a2 <- 12 b1 <- 7 b2 <- 8 d1 <- 30

The constructors for a virtual base class are invoked before any non-virtual base classes. If there are multiple virtual base classes, then they are invoked in the order in which they are declared.

Example:

```
// program to show how constructors are invoked in derived class
#include <iostream.h>
class alpha
(
   private:
      int x;
   public:
      alpha(int i)
      {
         x = i;
         cout << "\n alpha initialized \n";
      }
      void show_x()
      {
         cout << "\n x = "<<x;
      }
);
class beta
(
   private:
      float y;
   public:
      beta(float j)
      {
         y = j;
         cout << "\n beta initialized \n";
      }
      void show_y()
      {
         cout << "\n y = "<<y;
      }
);
class gamma : public beta, public alpha
(
   private:
      int n,m;
   public:
      gamma(int a, float b, int c, int d):: alpha(a), beta(b)
      {
         m = c;
         n = d;
         cout << "\n gamma initialized \n";
      }
      void show_mn()
```

```
    {
      cout << "\n m = "<<m;
      cout << "\n n = "<<n;
    }
);

void main()
{
   gamma g(5, 7.65, 30, 100);
   cout << "\n";
   g.show_x();
   g.show_y();
   g.show_mn();
}
```

Output:

```
beta initialized

alpha initialized

gamma initialized


x = 5
y = 7.65
m = 30
n = 100
```

Overriding member functions in c++:

Function overriding in C++ is **a feature that allows us to use a function in the child class that is already present in its parent class**. The child class inherits all the data members, and the member functions present in the parent class.

Example 1: C++ Function Overriding

```
// C++ program to demonstrate function overriding

#include <iostream>
using namespace std;

class Base {
  public:
   void print() {
      cout << "Base Function" << endl;
   }
};
```

```
class Derived : public Base {
   public:
    void print() {
        cout << "Derived Function" << endl;
    }
};

int main() {
    Derived derived1;
    derived1.print();
    return 0;
}
```

Output

Derived Function