Constructors in Derived Classes

- We can use constructors in derived classes in C++.
- If the base class constructor does not have any arguments, there is no need for any constructor in the derived class.
- But if there are one or more arguments in the base class constructor, derived class need to pass argument to the base class constructor.
- If both base and derived classes have constructors, base class constructor is executed first.

Constructors in Multiple Inheritances

- In multiple inheritances, base classes are constructed in the order in which they appear in the class declaration. For example if there are three classes “A”, “B”, and “C”, and the class “C” is inheriting classes “A” and “B”. 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.

Constructors in Multilevel Inheritances

- In multilevel inheritance, the constructors are executed in the order of inheritance. For example if there are three classes “A”, “B”, and “C”, and the class “B” is inheriting classes “A” and the class “C” is inheriting classes “B”. Then the constructor will run according to the order of inheritance such as the constructor of class “A” will be called first then the constructor of class “B” will be called and at the end constructor of class “C” will be called.

Special Syntax

- C++ supports a special syntax for passing arguments to multiple base classes
- The constructor of the derived class receives all the arguments at once and then will pass the call to the respective base classes
- The body is called after the constructors finish executing.

```
Derived_Constructor(arg1,arg2,….):Base1_Constructor(args),Base2_Constructor(args)
{
   Body of derived Constructor
}
```

Example Program

```
#include<iostream>
using namespace std;
class alpha
{
 int x;
 public:
  alpha(int i)
  {
    x = i;
    cout<<“Alpha initialized\n”;
  }
  void show_x(void)
  {
   cout<<“x = “<<x<<“\n”;
  }
};
```

```
class beta
{
  float y;
  public:
     beta(float j)
     {
          y = j;
          cout<<“ Beta initialzed\n”;
     }
      void show_y(void)
     {
       cout<<“y = “<<y<<“\n”;
     }
};
```

```
class gamma : public beta, public alpha
{
  int m,n;
  public:
    gamma(int a, float b, int c, int d):
          alpha(a), beta(b)
    {
      m = c;
      n = d;
      cout<<“gamma initialized\n”;
    }
    void show_mn(void)
    {
      cout<<“m = “ <<m<<“\n”;
      cout<<“n = “ <<n<<“\n”;
    }
};
```

```
int main()
{
  gamma g(5,10.75,20,30);
  cout<<"\n";
  g.show_x();
  g.show_y();
  g.show_mn();
  return 0 ;
}
```

OUTPUT

```
beta initialized
alpha initialized
gamma initialized
x = 5
y = 10.75
m = 20
n = 30
```