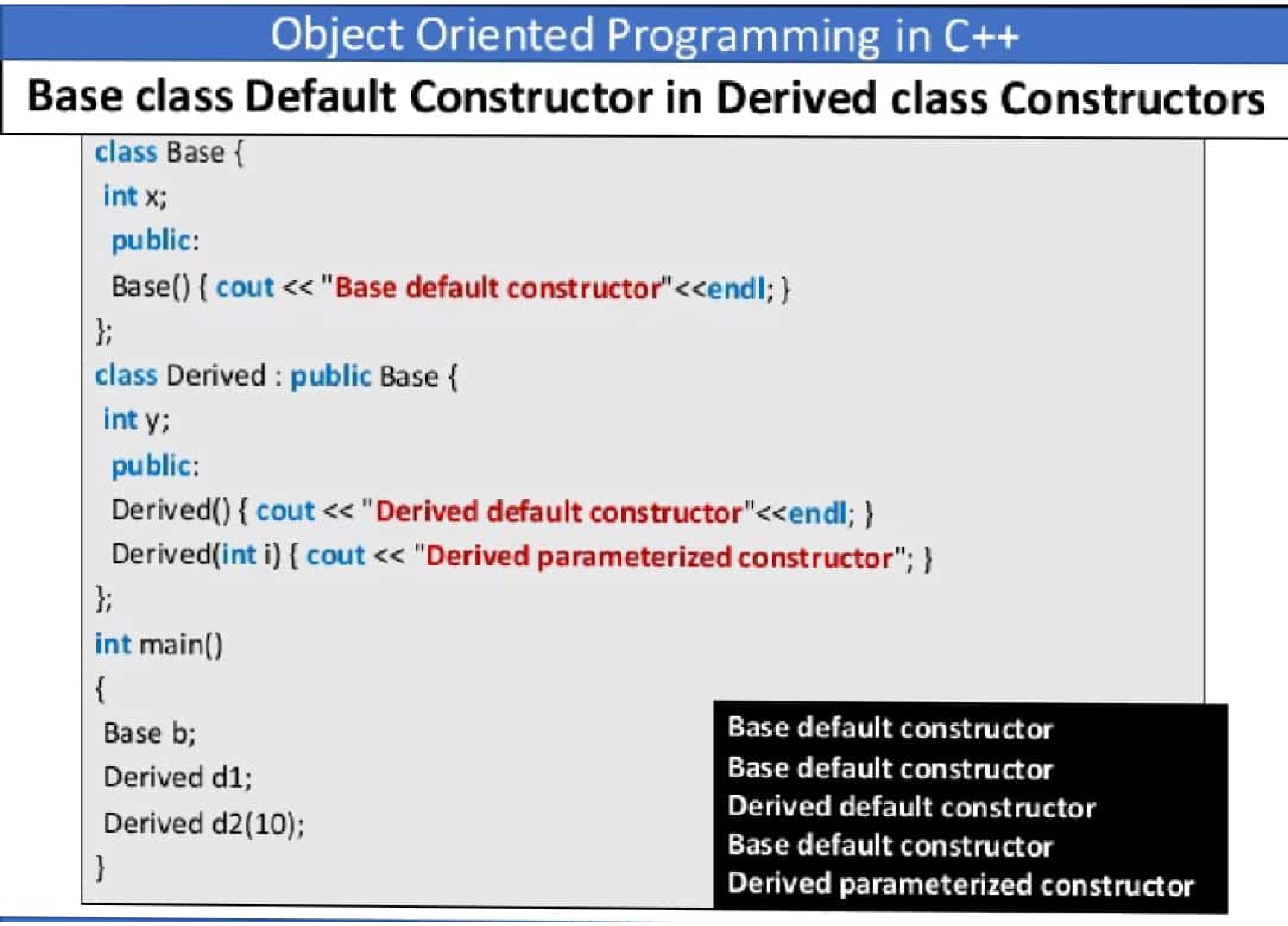
Order of Constructor Call

- Base class constructors are always called in the derived class constructors. Whenever you create derived class object, first the base class default constructor is executed and then the derived class's constructor finishes execution.
- **Points to Remember**
- Whether derived class's default constructor is called or parameterized is called, base class's default constructor is always called inside them.
- To call base class's parameterized constructor inside derived class's parameterized constructor, we must mention it explicitly while declaring derived class's parameterized constructor.

Object Oriented Programming in C++

Base class Default Constructor in Derived class Constructors

```
class Base {
 int x;
  public:
  Base() { cout << "Base default constructor"<<endl; }
};
class Derived : public Base {
 int y;
  public:
  Derived() { cout << "Derived default constructor"<<endl; }
  Derived(int i) { cout << "Derived parameterized constructor"; }
};
int main()
{
 Base b;
 Derived d1;
 Derived d2(10);
}
```

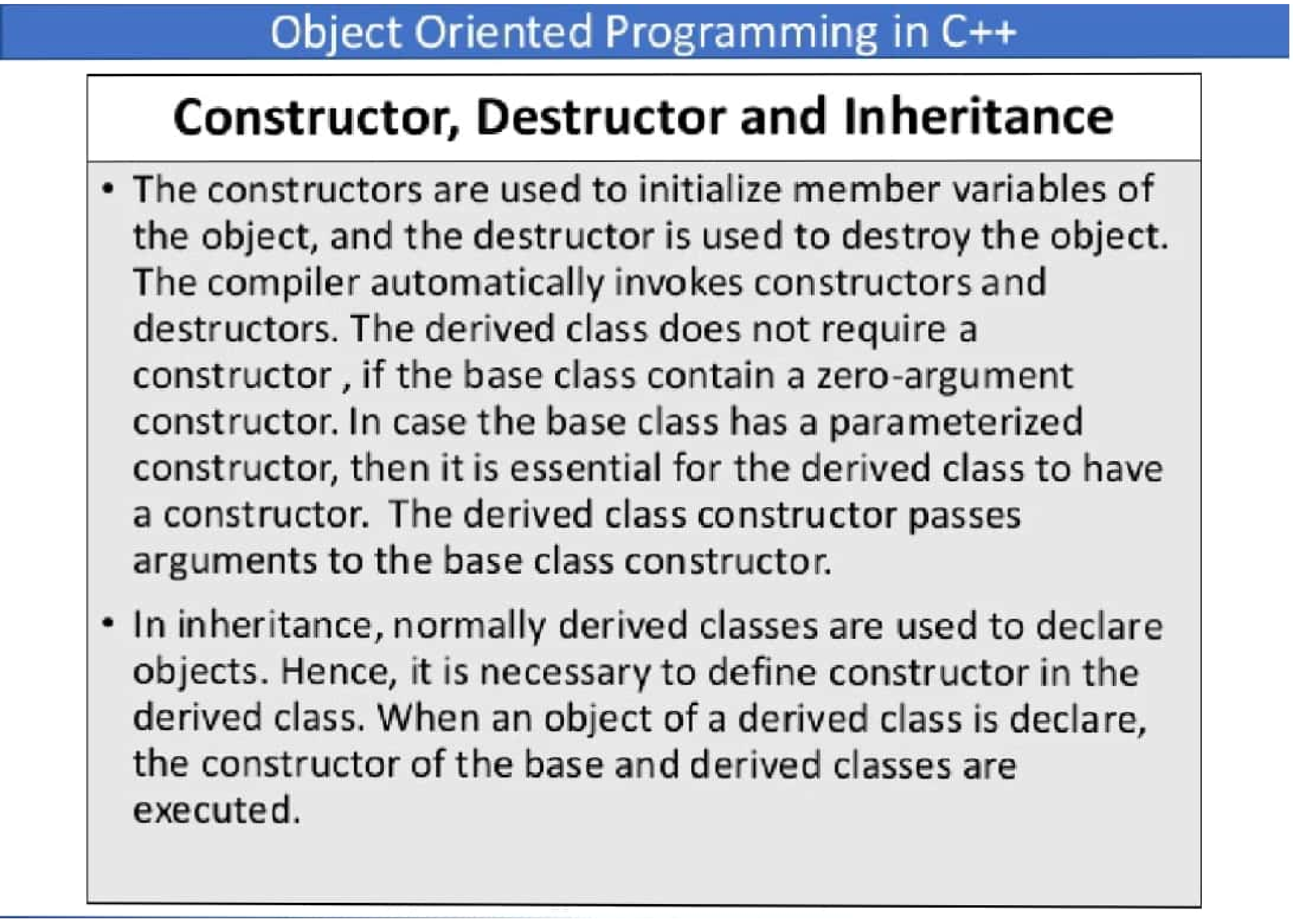


Object Oriented Programming in C++

Base class Default Constructor in Derived class Constructors

```
class Base {
 int x;
  public:
  Base() { cout << "Base default constructor"<<endl; }
};
class Derived : public Base {
 int y;
  public:
  Derived() { cout << "Derived default constructor"<<endl; }
  Derived(int i) { cout << "Derived parameterized constructor"; }
};
int main()
{
 Base b;
 Derived d1;
 Derived d2(10);
}
```

```
Base default constructor
Base default constructor
Derived default constructor
Base default constructor
Derived parameterized constructor
```



Constructor, Destructor and Inheritance

- The constructors are used to initialize member variables of the object, and the destructor is used to destroy the object. The compiler automatically invokes constructors and destructors. The derived class does not require a constructor , if the base class contain a zero-argument constructor. In case the base class has a parameterized constructor, then it is essential for the derived class to have a constructor. The derived class constructor passes arguments to the base class constructor.
- In inheritance, normally derived classes are used to declare objects. Hence, it is necessary to define constructor in the derived class. When an object of a derived class is declare, the constructor of the base and derived classes are executed.

Constructor, Destructor and Inheritance

- In heritance, destructors are executed in reverse order of the constructor execution. The destructors are executed when an object goes out of scope.
- To know the execution of constructors and destructors, let us study the following program:

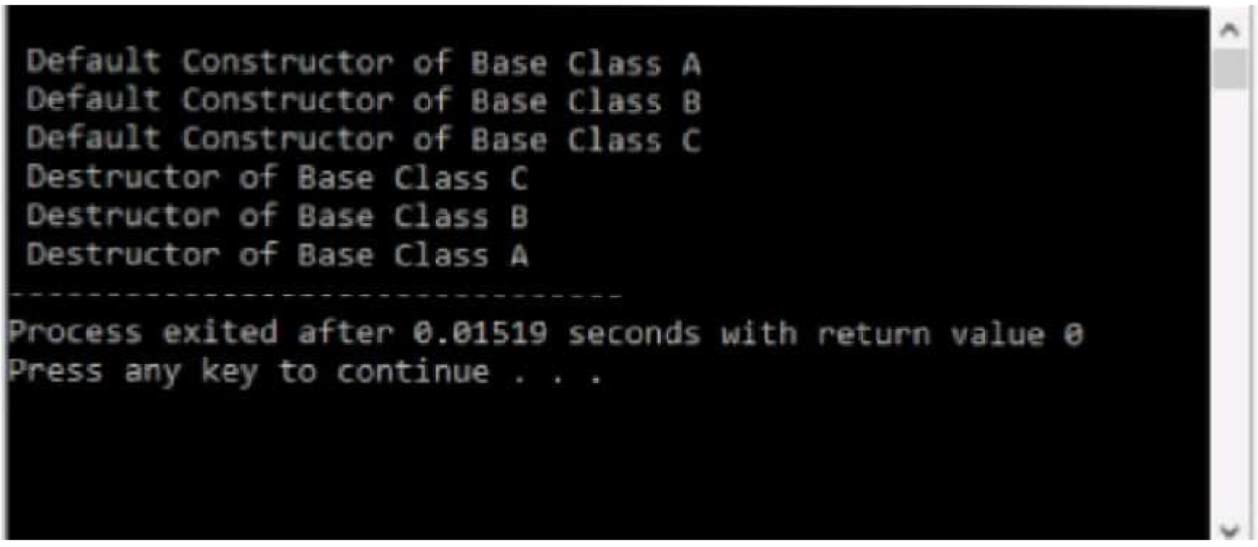
Write a program to show sequence of execution of constructor and destructor in multiple inheritance.

Constructor, Destructor and Inheritance-1

```
class A {   //Base Class
        public:
        A() {
                cout<<"\n Default Constructor of Base Class A";  }
        ~A() {
                cout<<"\n Destructor of Base Class A";  }
};
class B { //Base Class
        public:
        B() {
                cout<<"\n Default Constructor of Base Class B";  }
        ~B() {
                cout<<"\n Destructor of Base Class B"; }
};
```

Constructor, Destructor and Inheritance-2

```
class C: public A, public B    //Derivation of Class
{
          public:
          C() {
                    cout<<"\n Default Constructor of Base Class C";
                    }
          ~C() {
                    cout<<"\n Destructor of Base Class C";
          }
};
int main()
{
          C object;  //Creating an Object of Class C
          return 0;
}
```



```
 Default Constructor of Base Class A
 Default Constructor of Base Class B
 Default Constructor of Base Class C
 Destructor of Base Class C
 Destructor of Base Class B
 Destructor of Base Class A
--------------------------------
Process exited after 0.01519 seconds with return value 0
Press any key to continue . . .
```