Oct 01, 2015

COMPUTER PROGRAMMING

LAB 6

ABDUL AZIZ

FAST NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES

Lab Instructor	**Mr. Abdul Aziz**
Course	**Computer Programming Lab**
Duration	**2hrs**

Objectives:

In this lab, following topics will be covered

- ❖ Multilevel Inheritance
- ❖ Function Overriding
- ❖ Class Constructor
- ❖ Types of Constructor
- ❖ Constructor Overloading
- ❖ Destructor

1. Multilevel Inheritance

In C++ programming, a class be can derived from a derived class which is known as multilevel inheritance. For example:

```
#include <iostream>
using namespace std;

class A
{
   public:
     void display()
     {
        cout<<"Base class content.";
     }
};

class B : public A
{

};

class C : public B
{

};

int main()
{
   C c;
   c.display();
   return 0;
}
```

Explanation of Program

In this program, class *B* is derived from *A* and *C* is derived from *B*. An object of class *C* is defined in main( ) function. When the display( ) function is called, display( ) in class *A* is executed because there is no display( ) function in *C* and *B*. The program first looks for display( ) in class C first but, can't find it. Then looks in *B* because *C* is derived from *B*. Again it can't find it. And finally looks it in *A* and executes the codes inside that function.

If there was display( ) function in *C* too, then it would have override display( ) in *A* because of **member function overriding**.

2. Function Overriding

If base class and derived class have member functions with same name and arguments. If you create an object of derived class and write code to access that member function then, the member function in derived class is only invoked, i.e., the member function of derived class overrides the member function of base class. This feature in C++ programming is known as function overriding.

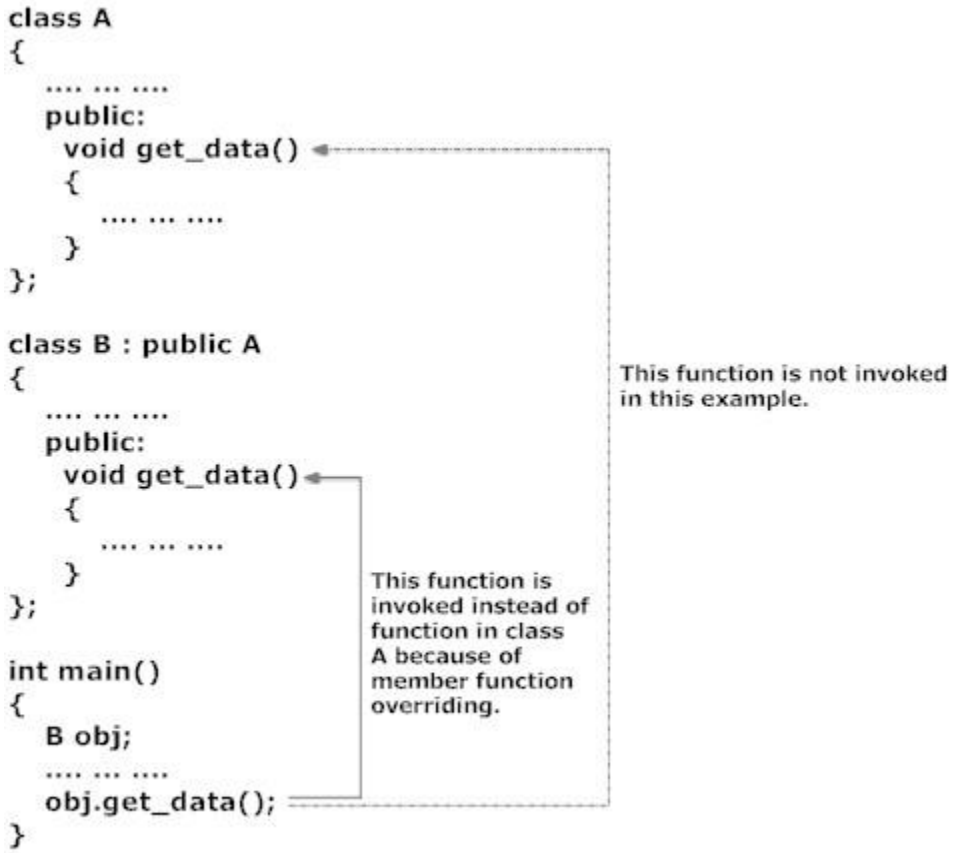

Figure: Member Function Overriding in C++

3. The Class Constructor.

A class **constructor** is a special member function of a class that is executed whenever we create new objects of that class.

A constructor will have exact same name as the class and it does not have any return type at all, not even void.

```
class A
{
 int x;
 public:
 A();  //Constructor
};
```

Constructors can be defined either inside the class definition or outside class definition using class name and scope resolution `::` operator.

3.1 Types of Constructor

1) Default Constructor
2) Parameterized Constructor

1. Default Constructor

Default constructor is the constructor which doesn't take any argument or you can say it has no any parameter.

```
class Cube
{
public:
int side;
public:
Cube()
 {
  side=10;
 }
};
int main()
{
Cube c;
cout << c.side;
}
```

```
Output: 10
```

In this case, as soon as the object is created the constructor is called which initializes its data members.

2. **Parameterized Constructor**

   These are the constructors with parameters. Using this Constructor you can provide different values to data members of different objects, by passing the appropriate values as argument.

```
class Cube
{
public:
 int side;
 public:
 Cube(int x)
  {
   side=x;
  }
};

int main()
{
 Cube c1(10);
 Cube c2(20);
 Cube c3(30);
 cout << c1.side;
 cout << c2.side;
 cout << c3.side;
}
```

`OUTPUT: 10 20 30`

By using parameterized construcor in above case, we have initialized 3 objects with user defin ed values. We can have any number of parameters in a constructor.

4. Destructor

Destructor is a special class function which destroys the object as soon as the scope of object ends. The destructor is called automatically by the compiler when the object goes out of scope.

The syntax for destructor is same as that for the constructor, the class name is used for the name of destructor; with a **tilde** `~` sign as prefix to it.

```
class A
{
 public:
 ~A();
};
```

Destructors will never have any arguments.

Example to see how Constructor and Destructor are called

```
class A
{
public: A()
 {
  cout << "Constructor called";
 }
~A()
 {
  cout << "Destructor called";
 }
};
int main()
{
 A obj1;   // Constructor Called
 int x=1
 if(x)
  {
   A obj2;                    // Constructor Called
  }                            // Destructor Called for obj2
}                              // Destructor called for obj1
```

EXERCISE

1.

a) Define a class named Payment that contains:
   - a member variable of type double that stores the amount of the payment and a method named paymentDetails that outputs an English sentence that describes the amount of the payment.

b) Define a class named CashPayment that is derived from Payment.
   - This class should redefine the paymentDetails method to indicate that the payment is in cash. Include appropriate constructor(s).

c) Define a class named CreditCardPayment that is derived from Payment.
   - This class should contain member variables for the name on the card, expiration date, and credit card number. Include appropriate constructor(s).
   - Redefine the paymentDetails method to include all credit card information in the printout.

d) Define a test class having a main method that creates at least two CashPayment and two CreditCardPayment objects with different values and calls paymentDetails for each.

2. Write a class named Car that has the following member variables:

- **year**. An int that holds the car's model year.
- **make**. A string that holds the make of the car.
- **speed**. An int that holds the car's current speed.

In addition, the class should have the following member functions.

- **Constructor**. The constructor should accept the car's year and make as arguments and assign these values to the object's year and make member variables. The constructor should initialize the speed member variable to 0.
- **accelerate**. The accelerate function should add 5 to the speed member variable each time it is called.
- **brake**. The brake function should subtract 5 from the speed member variable each time it is called.

Demonstrate the class in a program that creates a Car object, and then calls the accelerate function five times. After each call to the accelerate function, get the current speed of the car and display it. Then, call the brake function two times. After each call to the brake function, get the current speed of the car and display it.

Home Work

Suppose you are a programmer for the SILK Bank and you are assigned to develop a class that models the basic workings of a bank account. The class should perform the following tasks:

- Save the account balance.
- Save the number of transactions performed on the account.
- Allow deposits to be made to the account.
- Allow with drawls to be taken from the account.
- Report the current account balance at any time.
- Report the current number of transactions at any time.

Program Output

Menu

a) Display the account balance
b) Display the number of transactions
c) Make a deposit
d) Make a withdrawal
e) Exit the program