Objects and Classes in C Plus Plus

Module 2 - Access Modifiers in C Plus Plus

Access Modifiers

In C++ classes, we can control the access to the members of the class using Access Specifiers. Also known as access modifier, they are the keywords that are specified in the class and all the members of the class under that access specifier will have particular access level.
In C++, there are 3 access specifiers that are as follows:
1. Public: Members declared as public can be accessed from outside the class.
2. Private: Members declared as private can only be accessed within the class itself.
3. Protected: Members declared as protected can be accessed within the class and by derived classes.
If we do not specify the access specifier, the private specifier is applied to every member by default.

1. Public

- Members declared as public can be accessed from **anywhere** in the program.
- These members are accessible outside the class using the object of the class.

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

class Car {
public:
    string brand; // Public member
    void displayBrand() { // Public member function
        cout << "Brand: " << brand << endl;
    }
};

int main() {
    Car myCar;
    myCar.brand = "Toyota"; // Directly accessible
    myCar.displayBrand();   // Directly accessible
    return 0;
}
```

2. Private

- Members declared as private can be accessed **only** within the class where they are declared.
- They are not accessible outside the class, even with an object.

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

class Car {
private:
    string brand; // Private member
public:
    void setBrand(string b) { // Public function to set private member
        brand = b;
    }
    void displayBrand() { // Public function to access private member
        cout << "Brand: " << brand << endl;
    }
};

int main() {
    Car myCar;
    // myCar.brand = "Toyota"; // Error: Cannot access private member directly
    myCar.setBrand("Toyota"); // Accessed using public member function
    myCar.displayBrand();
    return 0;
}
```

3. Protected

- Members declared as protected can be accessed:
  - Within the class itself.
  - By derived classes (subclasses).
- They are not accessible outside the class through objects.

Introduction to 'C++' Language

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

class Vehicle {
protected:
    string brand; // Protected member
public:
    void setBrand(string b) {
        brand = b;
    }
};

class Car : public Vehicle {
public:
    void displayBrand() {
        cout << "Brand: " << brand << endl; // Accessible in derived class
    }
};

int main() {
    Car myCar;
    myCar.setBrand("Honda"); // Set brand using public method of base class
    myCar.displayBrand();    // Access protected member through derived class
    return 0;
}
```

Summary of Access Levels

Modifier	Access in the Same Class	Access in Derived Classes	Access Outside the Class
Public	Yes	Yes	Yes
Private	Yes	No	No
Protected	Yes	Yes	No

Defining member functions (i.e. writing the body of a function):

Member functions can be defined in two ways.
- **Inside the class** (Inline function).
- **Outside the class** using the scope resolution operator (::).

The code for the function body would be identical in both the cases. Irrespective of the place of definition, the function should perform the same task.

Solved Example 1: Inline Member Functions (Defined Inside the Class)

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

class Rectangle {
public:
    int length, breadth;

    // Member function defined inside the class
    int area() {
        return length * breadth;
    }
};

int main() {
    Rectangle rect;
    rect.length = 10;
    rect.breadth = 5;

    cout << "Area: " << rect.area() << endl;
    return 0;
}
```

Explanation of this code:

Class Declaration:

- The class Rectangle is a user-defined type that contains attributes (data members) and functions (member functions).
- int length, breadth; are **public data members** of the class, which represent the dimensions of the rectangle.

Inline Member Function:

- int area() is a **member function** defined inside the class.
- This function calculates the **area of the rectangle** by multiplying the length and breadth.
- Since it's defined inside the class definition, it is considered an **inline function**. The compiler will attempt to insert the code directly at the location where it's called, rather than creating a function call, making it faster for small functions.

Main Function:

Object Creation:

- Rectangle rect; creates an object rect of the Rectangle class.

Assigning Values:

• rect.length = 10; assigns the value 10 to the length of the rectangle.
• rect.breadth = 5; assigns the value 5 to the breadth of the rectangle.

Calling the area() Function:

• rect.area() calls the area() member function of the Rectangle class to calculate the area. It will return the product of length and breadth, which is 10 * 5 = 50.

Displaying the Result:

• cout << "Area: " << rect.area() << endl; displays the result of the area() function (i.e., 50) to the console.

Output:

Area = 50

Solved Example 2: Member Functions Defined Outside the Class

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

class Rectangle {
public:
    int length, breadth;

    // Function declaration
    int area();
    void setDimensions(int l, int b);
};

// Function definitions outside the class
void Rectangle::setDimensions(int l, int b) {
    length = l;
    breadth = b;
}

int Rectangle::area() {
    return length * breadth;
}

int main() {
    Rectangle rect;
    rect.setDimensions(10, 5);

    cout << "Area: " << rect.area() << endl;
    return 0;
}
```

Class Definition:

The Rectangle class has two attributes: length and breadth.
Two member functions are declared inside the class:
• area(): Calculates the area of the rectangle.
• setDimensions(int l, int b): Sets the values for length and breadth.

Function Definitions Outside the Class:

setDimensions(int l, int b):
• Assigns the parameters l and b to the length and breadth attributes of the Rectangle object.
area():
• Calculates and returns the area of the rectangle using the formula: Area= length × breadth

Main Function:
1. An object rect of the Rectangle class is created.
2. The setDimensions() function is called to set length = 10 and breadth = 5.
3. The area() function is called to calculate the area of the rectangle: Area=10 × 5 = 50
4. The result (50) is displayed using cout.

Output

Area = 50

Solved Example 3: Static Member Function

Static Member Functions
• Declared using the static keyword.
• Can only access static data members of the class.

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

class Counter {
private:
    static int count;

public:
    static void increment() { // Static member function
```

```
        count++;
    }

    static int getCount() {
        return count;
    }
};

// Define and initialize static member
int Counter::count = 0;

int main() {
    Counter::increment(); // Call static function without object
    Counter::increment();

    cout << "Count: " << Counter::getCount() << endl;
    return 0;
}
```

Explanation of this code

Static Data Member (count):
- Declared as static int count inside the Counter class.
- A static member is shared among all objects of the class and belongs to the class, not to any individual object.
- It is defined and initialized outside the class as int Counter::count = 0;.

Static Member Functions (increment and getCount):
- Static functions can access only static data members of the class.
- They can be called without creating an object of the class.

Purpose of the Code:
- The Counter class maintains a static count variable.
- The increment function increments the count by 1 each time it is called.
- The getCount function retrieves the current value of count.

Output
Count: 2

Solved Example 4: Const Member Function

- Declared using the const keyword after the function declaration.
- Cannot modify any class data members.

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

class Circle {
private:
    double radius;

public:
    Circle(double r) : radius(r) {}

    // Const member function
    double getRadius() const {
        return radius;
    }
};

int main() {
    Circle c(5.5);

    cout << "Radius: " << c.getRadius() << endl;
    return 0;
}
```

Explanation of this code

const Member Function:

Declared with the keyword const after the function's parameter list (double getRadius() const).
Ensures that the member function does not modify any member variables of the class.
Can be called on const objects of the class or non-const objects.
Constructor:

Circle(double r) is a parameterized constructor.
It initializes the private member radius with the value passed during object creation (5.5 in this case).
Private Member (radius):

Introduction to 'C++' Language

The radius variable stores the radius of the circle.
It is private and can only be accessed via public member functions.
Main Function:

A Circle object c is created with a radius of 5.5.
The getRadius() function is called to retrieve the value of radius.

Output

Radius: 5.5