FAST National University of Computer and Emerging Sciences Peshawar

OOP Lab # 11

C++ (Classes and Objects)

Computer Instructor: Engr. Khuram Shahzad

DEPARTMENT OF COMPUTER SCIENCE

Programming

الذی علم بالقلم۔ علم الانسان ما لم یعلم۔

OOP Lab-11: Contents

1) Access Specifiers/Modifiers in C++
2) Constructor
3) Types of Constructor (Default Constructor and Parametrized Constructor)

Access Specifiers/Modifiers in C++

- It specifies that member of a class is accessible outside or not.
- Access modifiers are used to implement an important aspect of Object-Oriented Programming known as **Data Hiding.**
- Access Modifiers or Access Specifiers in a class are used to assign the accessibility to the class members. That is, it sets some restrictions on the class members not to get directly accessed by the outside functions.
  There are 3 types of access modifiers available in C++:

1. Public
2. Private
3. Protected

Note: If we do not specify any access modifiers for the members inside the class then by default the access modifier for the members will be **Private**.

1. Public Access Specifier

- All the class members declared under the public specifier will be available to everyone.
- The data members and member functions declared as public can be accessed by other classes and functions too.
- The public members of a class can be accessed from anywhere in the program using the direct member access operator (.) with the object of that class.

Example 1: Public Access Specifier

Program.cpp

```cpp
// C++ program to demonstrate public access modifier
#include<iostream>
using namespace std;
// class definition
class Circle
{
    public:
        double radius;
        double ComputeArea()
        {
            return 3.14*radius*radius;
        }
};
// main function
int main()
{
    Circle obj;
    // accessing public data member outside class
    obj.radius = 5.5;

    cout << "Radius is: " << obj.radius << "\n";
    cout << "Area is: " << obj.ComputeArea();
    return 0;
}
```

Output:
Radius is: 5.5
Area is: 94.985

In the above program the data member *radius* is declared as public so it could be accessed outside the class and thus was allowed access from inside **main().**

2. Private Access Specifier

- The class members declared as *private* can be accessed only by the member functions inside the class.
- They are not allowed to be accessed directly by any object or function outside the class.
- Only the member functions or the **friend functions** are allowed to access the private data members of a class.
- **friend functions** will be discussed later

Example 1: Private Access Specifier…

Program.cpp

```cpp
// C++ program to demonstrate private access modifier
#include<iostream>
using namespace std;
class Circle
{
    // private data member
    private:
        double radius;

    // public member function
    public:
        double ComputeArea()
        { // member function can access private
            // data member radius
            return 3.14*radius*radius;
        }
}; //end of class
// main function
int main()
{
    // creating object of the class
    Circle obj;

    // trying to access private data member
    // directly outside the class
    obj.radius = 1.5;

    cout << "Area is:" << obj.ComputeArea();
    return 0;
}
```

Output:

```
In function 'int main()': 11:16: error:
'double Circle::radius' is private
double radius; ^ 31:9: error: within
this context obj.radius = 1.5; ^
```

Example 1: Private Access Specifier…

The output of above program is a compile time error because we are not allowed to access the private data members of a class directly outside the class. Yet an access to **obj.radius** is attempted, radius being a private data member we obtain a compilation error.

Example 2: Private Access Specifier…

Program.cpp

```cpp
// C++ program to demonstrate private access modifier
#include<iostream>
using namespace std;
class Circle
{
    // private data member
    private:
        double radius;
// public member function
    public:
        void ComputeArea(double r)
        { // member function can access private
            // data member radius
            radius = r;
            double area = 3.14*radius*radius;
            cout << "Radius is: " << radius << endl;
            cout << "Area is: " << area;
        }
};  //end of class
// main function
int main()
{
    // creating object of the class
    Circle obj;
    // trying to access private data member
    // directly outside the class
    obj.ComputeArea(1.5);
    return 0;
}
```

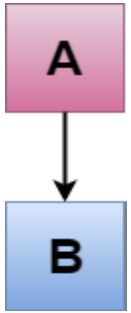
Output:

Radius is: 1.5
Area is: 7.065

However, we can access the private data members of a class indirectly using the public member functions of the class.

3. Protected Access Specifier

- Protected access modifier is similar to private access modifier in the sense that it can't be accessed outside of it's class unless with the help of friend class.
- The difference is that the class members declared as Protected can be accessed by any subclass(derived class) of that class as well.
- **Note**: This access through inheritance can alter the access modifier of the elements of base class in derived class depending on the **modes of Inheritance**.



Where 'A' is the base class, and 'B' is the derived class.

Example 1: Protected Access Specifier

Program.cpp

```cpp
// C++ program to demonstrate protected access modifier
#include <iostream>
using namespace std;
// base class
class Parent
{
    // protected data members
    protected:
    int protectedID;

}; //parent class ends
// sub class or derived class from public base class
class Child : public Parent
{
    public:
    void setId(int id)
    {
        // Child class is able to access the inherited
        // protected data members of base class
        protectedID = id;
    }
    void displayId()
    {
        cout << "protectedID is: " << protectedID << endl;
    }
};  // child class ends
// main function
int main() {
    Child obj1;
    // member function of the derived class can
    // access the protected data members of the base class
    obj1.setId(81);
    obj1.displayId();
    return 0;
}  // end of main() function
```

Output:
protectedID is: 81

Summary: public, private, and protected

public elements can be accessed by all other classes and functions.

private elements cannot be accessed outside the class in which they are declared, except by friend classes and functions.

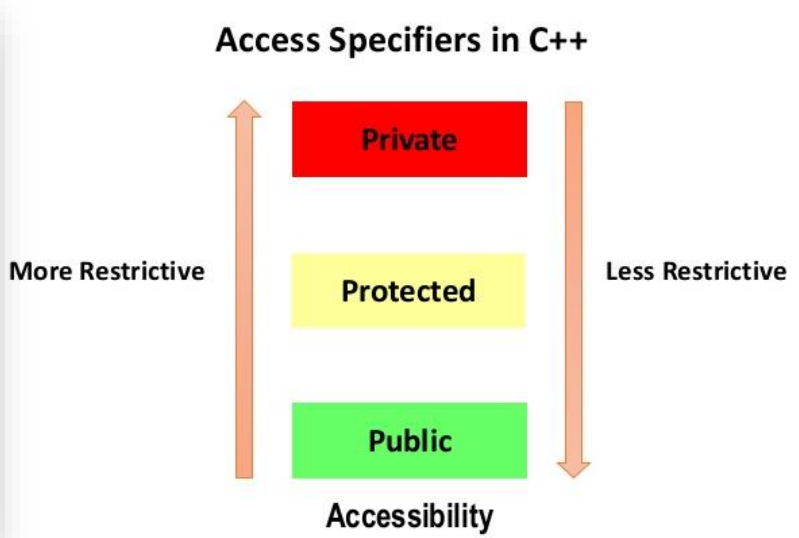
protected elements are just like the private, except they can be accessed by derived classes.

Note: By default, class members in C++ are **private**, unless specified otherwise.

Access Specifiers/Modifiers in C++…

Access Modifiers in C++

Modifiers	Own Class	Derived Class	Main()
Public	Yes	Yes	Yes
Private	Yes	No	No
Protected	Yes	Yes	No



Constructor in C++

- Special method that is implicitly invoked.
- Used to create an object (an instance of the class) and initialize it.
- Every time an object is created, at least one constructor is called.
- It is special member function having same name as class name and is used to initialize object.
- It is invoked/called at the time of object creation.

Constructor in C++…

- It constructs value i.e. provide data for the object that is why it called constructor.
- Can have parameter list or argument list. Can never return any value (no even void).
- Normally declared as public.
- At the time of calling constructor, memory for the object is allocated in the memory.
- It calls a default constructor if there is no constructor available in the class.

Constructor in C++…

Note: It is called constructor because it constructs the values at the time of object creation.

There can be two types of constructors in C++.

1. Default constructor (no-argument constructor)
2. Parameterized constructor.

1. Default constructor

- A constructor is called "Default Constructor" when it doesn't have any parameter.
- It is also called non-parameterized constructor.
- A constructor which has no argument is known as default constructor. It is invoked at the time of creating object.
- **Note:** If we have not defined a constructor in our class, then the C++ compiler will automatically create a default constructor with an empty code and no parameters.
- Let's see the simple example of C++ default Constructor.

1. Default constructor…

```
#include <iostream>
using namespace std;
class Employee
 {
   public:
        Employee()
        {
            cout<<"Default Constructor Invoked/Called"<<endl;
        }
};
int main(void)
{
    Employee e1; //creating an object of Employee
    Employee e2;
    return 0;
}
```

Output:

Default Constructor Invoked/Called
Default Constructor Invoked/Called

1. Default constructor…

Program.cpp

```cpp
#include<iostream>
#include<string.h>
using namespace std;
class Student
    {
        int Roll;
        char Name[25];
        float Marks;
        public:
        Student()           //Default Constructor
            {
                Roll = 1;
                strcpy(Name,"Kumar");
                Marks = 78.42;
            }
 void Display()
       {
            cout<<"\n\tRoll : "<<Roll;
            cout<<"\n\tName : "<<Name;
            cout<<"\n\tMarks : "<<Marks;
       }
    };  // end of class
int main()
    {
        Student S;          //Creating Object
        S.Display();        //Displaying Student Details
        return 0;
    }
```

Output:
Roll : 1
Name : Kumar
Marks : 78.42

2. Parameterized Constructor

- A constructor which has a specific number of parameters is called a parameterized constructor.

Why use the parameterized constructor?

- The parameterized constructor is used to provide different values to distinct objects. However, you can provide the same values also.
- Used to initialize objects with different values.
- This is the preferred method to initialize member data.

- Let's see the simple example of C++ Parameterized Constructor.

2. Parameterized Constructor…

Program.cpp

```cpp
#include <iostream>
using namespace std;
class Employee {
   public:
       int id; //data member (also instance variable)
       string name; //data member(also instance variable)
       float salary;

    Employee(int i, string n, float s)
        {
            id = i;
            name = n;
            salary = s;
        }
void display()
        {
            cout<<id<<"  "<<name<<"  "<<salary<<endl;
        }
};
int main(void) {
  Employee e1 =Employee(101, "Ali", 890000); //creating an object of Employee
    Employee e2=Employee(102, "Saad", 59000);
    e1.display();
    e2.display();
    return 0;
}
```

Output:

101 Ali 890000
102 Saad 59000

2. Parameterized Constructor…

Program.cpp

```cpp
#include<iostream>
#include<string.h>
using namespace std;
class Student
    {
        int Roll;
        char Name[25];
        float Marks;
        public:
        Student(int r,char nm[],float m)           //Parameterize Constructor
        {
            Roll = r;
            strcpy(Name,nm);
            Marks = m;
        }
void Display()
        {
            cout<<"\n\tRoll : "<<Roll;
            cout<<"\n\tName : "<<Name;
            cout<<"\n\tMarks : "<<Marks;
        }
    };
int main()
    {    Student S(2,"Sumit",89.63);
        //Creating Object and passing values to Constructor
        S.Display();
        //Displaying Student Details
        return 0;
    }
```

OUTPUT
Roll : 2
Name : Ali
Marks : 89.63

References

- https://beginnersbook.com/2017/08/cpp-data-types/
- http://www.cplusplus.com/doc/tutorial/basic_io/
- https://www.w3schools.com/cpp/default.asp
- https://www.javatpoint.com/cpp-tutorial
- https://www.geeksforgeeks.org/object-oriented-programming-in-cpp/?ref=lbp
- https://www.programiz.com/
- https://ecomputernotes.com/cpp/

THANK YOU