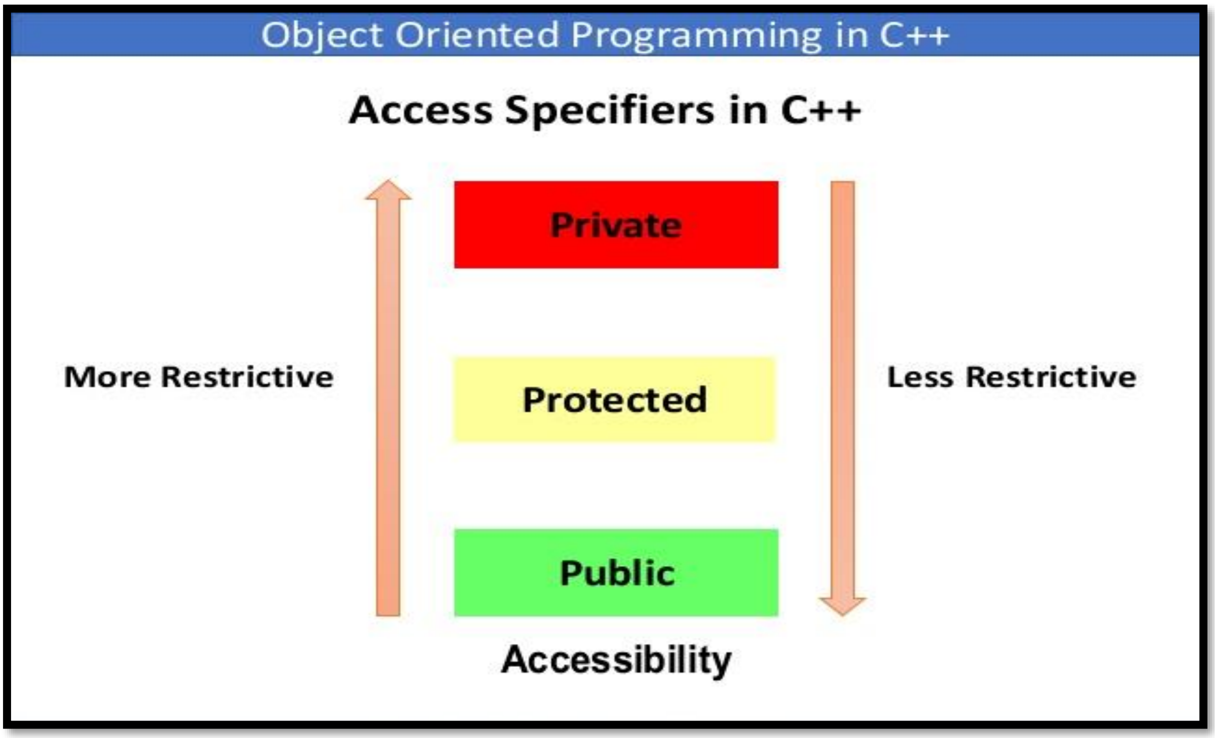
National University of Computer & Emerging Sciences, Karachi
Computer Science Department
Spring 2022, Lab Manual - 05

Course Code: CL-1004	**Course : Object Oriented Programming Lab**
Instructor(s) :	**Shoaib Rauf**

Contents:

Access Modifiers
Static Keyword
Constant Keyword

Introduction to Access Modifiers in C++:



Access modifiers is the techniques that is applied to members of class to restrict their access beyond the class.

In C++, access modifiers can be achieved by using three keywords:

Public

Private

Protected

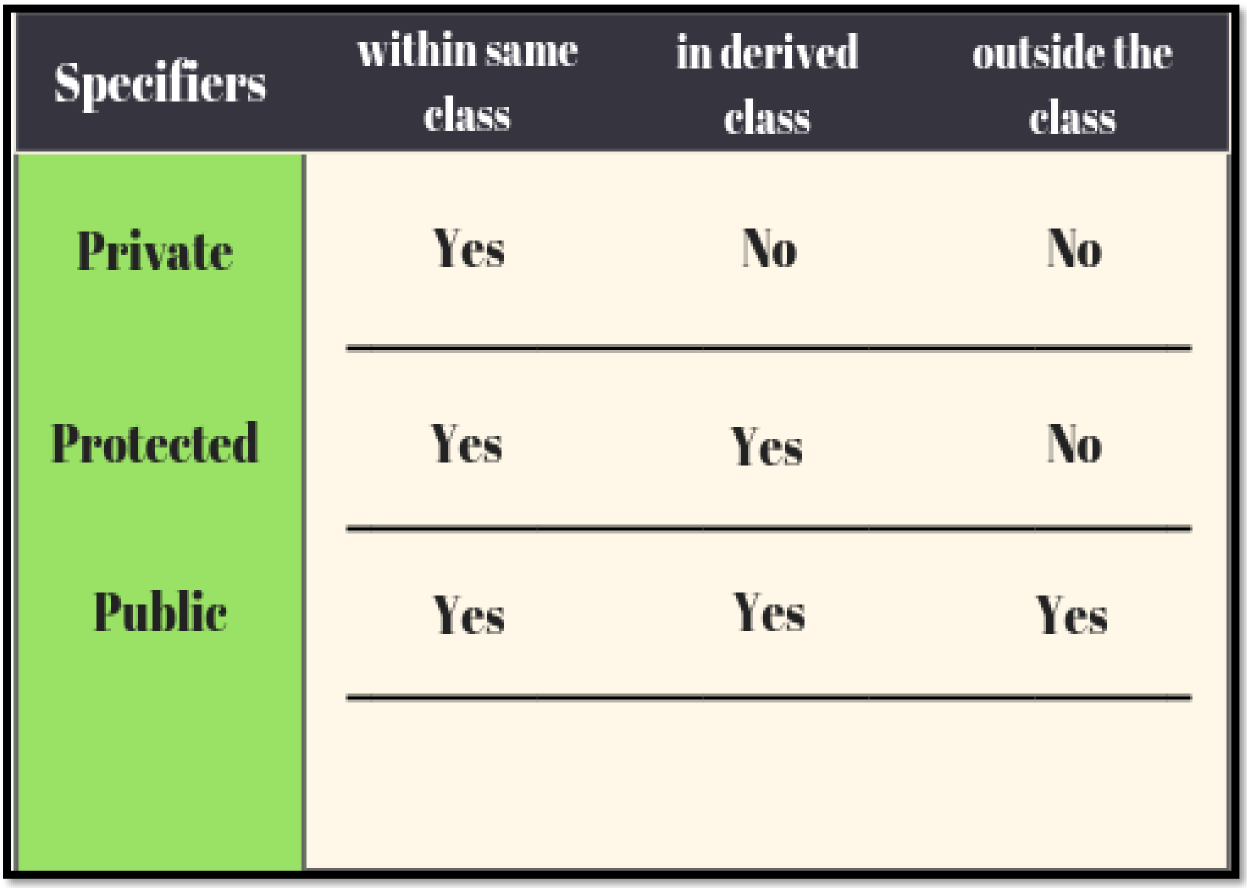
Types of Access Modifiers in C++

There are 3 types of Access Modifiers in C++

1. Public
2. Private
3. Protected

Public members can be accessed anywhere i.e. inside or outside the class but within the program only,

Private members can be accessed inside the class only,



Specifiers	within same class	in derived class	outside the class
Private	Yes	No	No
Protected	Yes	Yes	No
Public	Yes	Yes	Yes

Protected members are like the private they can be accessed in the child class/derived class.

Let’s look at these modifiers with examples:

Public:

As there are no restrictions in public modifier, we can use the (.)dot operator directly accesses member functions and data.

```
// C++ program to demonstrate public access modifier

#include<iostream>

using namespace std;

// class definition

class Circle

{    public:

        double radius;

        double  compute_area()

        {           return 3.14*radius*radius;       }

    };

int main ()

{   Circle obj;
```

```
    // accessing public data member outside class

     obj.radius = 5.5;

     cout << "Radius is: " << obj.radius << "\n";

     cout << "Area is: " << obj.compute_area();

     return 0;  }
```

Output:

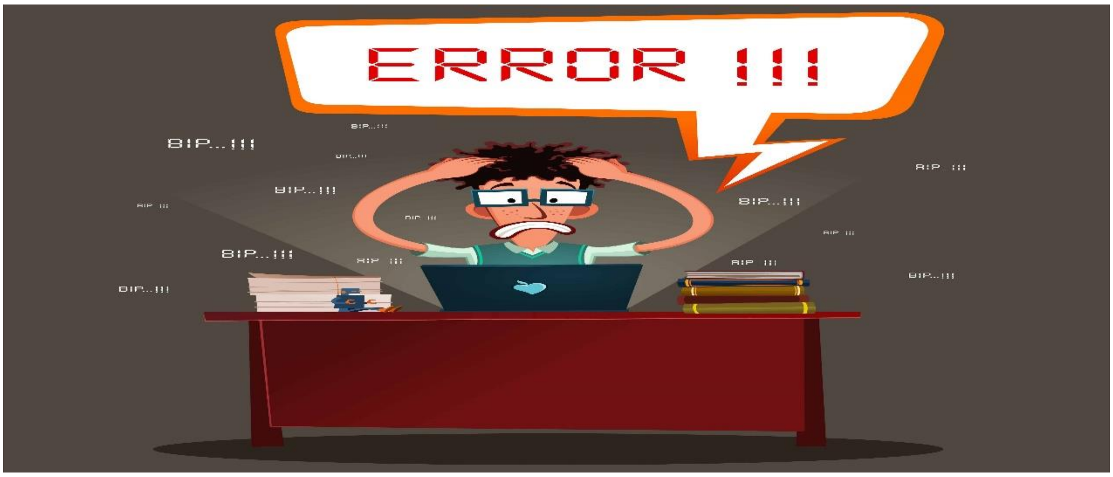
```
Radius is: 5.5

Area is: 94.985
```

Private:

Only the member functions or the friend functions are allowed to access the private data members of a class.

The example below has an error let's find out:



```
// 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 compute_area()
            { // member function can access private
                    // data member radius
                    return 3.14*radius*radius;
            }
};
// 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.compute_area();
    return 0;
}
```

```
// 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 compute_area(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;
            }
    };
// main function
int main()
{   // creating object of the class
    Circle obj;
    // trying to access private data member
    // directly outside the class
    obj.compute_area(1.5);
    return 0;  }
```

Output:

```
Radius is: 1.5   Area is: 7.065
```

Protected:

```
#include <bits/stdc++.h>
using namespace std;
// base class
class Parent
{   // protected data members
    protected:
    int id_protected;
};
// 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
            id_protected = id;
    }
    void displayId()
    {
            cout << "id_protected is: " << id_protected << endl;
    }
};
// 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;
}
```

Output:

```
id_protected is: 81
```

const **Keyword in C++**

Constant is something that doesn't change. In C language and C++ we use the keyword const to make program elements constant. **const** keyword can be used in many contexts in a C++ program. It can be used with:

1. Variables
2. Function arguments and return types
3. Class Data members
4. Class Member functions
5. Objects

Constant Variables in C++

```
int main
{
   const int i = 10;
   const int j = i + 10;     // works fine
   i++;    // this leads to Compile time error
}
```

If you make any variable as constant, using const keyword, you cannot change its value. Also, the constant variables must be initialized while they are declared.

In the above code we have made i as constant, hence if we try to change its value, we will get compile time error. Though we can use it for substitution for other variables.

Static Keyword in C++

Static is a keyword in C++ used to give special characteristics to an element. Static elements are allocated storage only once in a program lifetime in static storage area. And they have a scope till the program lifetime. Static Keyword can be used with following,

1. Static variable in functions
2. Static Class Objects
3. Static member Variable in class
4. Static Methods in class