Static Members of a Class

In the previous sections we have shown examples of classes where each object of a class had its own set of data. Member function could access the object's own version of the data.

In some situations it may be desirable that one or more common data fields should exist, which are accessible to all objects of the class. In C++, a static member is shared by all objects of the class.

Static Data Members

A data member of a class can be declared static; be it in the public or private part of the class definition. Such a data member is created and initialized only once. Static data members which are declared public can be accessed by using class name and the scope resolution operator.

We only include the declaration of static data in the class declaration. Initialization of a static data-member is done outside the class. This is illustrated in the following code fragment:

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

class Circle
{
    private:
        double radius;      // Radius of a circle
    public:
        static int count;
        // Constructor definition
        Circle(double r = 1.0)
        {
            radius = r;
            // Increase every time object is created
            count++;
        }
        double getArea()
        {
            return 3.14 * radius * radius;
        }
};
```

```
// Initialize static member of class Circle
int Circle::count = 0;


int main()
{
    Circle c1(3.3);     // Declare object c1
    Circle c2(4.5);     // Declare object c2

    // Print total number of objects.
    cout << "Total objects: " << Circle::count << endl;

    return 0;
}
```

Output:

Total objects: 2

Static Member Functions

The static functions can access only the static data of a class. Similarly, static functions cannot call non-static functions of the class.

Functions which are static and which are declared in the public section of a class can be called without specifying an object of the class. This is illustrated in the following code fragment:

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

class Circle
{
    private:
        static int count;
        double radius;      // Radius of a circle
    public:
        // Constructor definition
        Circle(double r = 1.0)
        {
```

```
            radius = r;
            // Increase every time object is created
            count++;
        }
        double getArea()
        {
            return 3.14 * radius * radius;
        }
        static int getCount()
        {
            return count;
        }
};

// Initialize static member of class Circle
int Circle::count = 0;

int main()
{
    Circle c1(3.3);    // Declare object c1
    Circle c2(4.5);    // Declare object c2

    // Print total number of objects.
    cout << "Total objects: " << Circle::getCount() << endl;

    return 0;
}
```

Output:

Total objects: 2