Object-Oriented Programming

Lecture No. 6

Static members (cont..), overloaded (unary) operator(cont..)

Static member functions

- By declaring a function member as static, you make it independent of any particular object of the class. A static member function can be called even if no objects of the class exist and the **static** functions are accessed using only the class name and the scope resolution operator **::**.
- A static member function can only access static data member, other static member functions and any other functions from outside the class.
- Static member functions have a class scope and they do not have access to the **this** pointer of the class. You could use a static member function to determine whether some objects of the class have been created or not.

Static members–Example

```
class Box {
 private:
      double length;
      double breadth;
       double height;
public:
      static int objectCount;
Box(double l = 2.0, double b = 2.0,
double h = 2.0) {
 cout <<"Constructor called." << endl;
         length = l;
         breadth = b;
         height = h;
objectCount++;
      }
 double Volume() {
        return length * breadth * height;
      }
      static int getCount() {
        return objectCount;       }
};
int Box::objectCount = 0;
int main(void) {
cout << "Inital Stage Count: " << Box::getCount() << endl;
  Box Box1(3.3, 1.2, 1.5);    // Declare box1
   Box Box2(8.5, 6.0, 2.0);    // Declare box2
cout << "Final Stage Count: " << Box::getCount() << endl;
  return 0;
}
```

Static members (private) –Example

```
class MyClass{
   private:
      static int st_var;
   public:
      MyClass(){
         st_var++; }
      static int getStaticVar() {
         return st_var;
      }
};
```

```
int MyClass::st_var = 0;

 main() {
   MyClass ob1, ob2, ob3; //three
objects are created
   cout << "Number of objects: " <<
MyClass::getStaticVar();
}
```

Overloading Unary Operator

- Let us consider to overload (-) unary operator. In unary operator function, no arguments should be passed. It works only with one class objects. It is a overloading of an operator operating on a single operand.
  Example:
  Assume that class Distance takes two member object i.e. feet and inches, create a function by which Distance object should decrement the value of feet and inches by 1 (having single operand of Distance Type).

Overloading unary operator

```
class Distance {
int feet, inch;
public:
        Distance(int f, int i)
        {       this->feet = f;
                this->inch = i;
        }
        void operator--()
        {       feet--;
                inch--;
cout << "\nFeet &
Inches(Decrement): " << feet << "'"
<< inch; } };
```

```
int main()
{
        Distance d1(8, 9);
        --d1;
        return 0;
}
```

Pre-fix vs post-fix operator overloading

- When increment operator is overloaded in prefix form; Check operator ++ () is called but, when increment operator is overloaded in postfix form; operator ++ (int) is invoked.

- The int inside bracket. This int gives information to the compiler that it is the postfix version of operator.

Exercise

(class, constructor, constructors overloading, static members, operator overloading, array of objects)

Problem statement I

- Write a program that has a class named patient having the patient's name, age, and disease as attributes. The class has a method called dosage that calculates and displays the dosage according to the disease level and age of each patient. If the disease level is 1 and the patient age is less than 10 years then the dosage should be 50mg per day, for patients aged between 10 and less than 19, it should be 100 mg and for patients above 19 and less than 30, the dosage should be 150 mg. Incase of disease level 2, the recommended dosage gets double for all the respective age groups. The program should use an array of objects to get the details and dosage of 10 patients.

Problem statement II

- Write a program that implements a class to multiply and divide two rational numbers using operator overloading. The class should have two data members for denominator and numerator values. It should also have a parameterized constructor to initialize the values of one object and a copy constructor to initialize second object through previously created object. The class should have a method to take inputs of denominator and numerator from user. It should also contain a static member to keep the count of the objects of rational numbers. Display the output after performing multiplication and division. Implement check to ensure that the value of denominator should not be entered as zero.

Reference

- C++ How to Program
  By Deitel & Deitel

- The C++ Programming Language
  By Bjarne Stroustrup

- Object oriented programming using C++ by Tasleem Mustafa, Imran Saeed, Tariq Mehmood, Ahsan Raza

- https://www.tutorialspoint.com

- https://www.geeksforgeeks.org/

- https://www.includehelp.com/cpp-programs/