CS217 – Object Oriented Programming (OOP)

Week – 04

Mar 1-5, 2021

Instructor: **Basit Ali**

Static Members of a C++ Class

- We can define class members static using **static** keyword.
- When we declare a member of a class as static it means no matter how many objects of the class are created, there is only one copy of the static member.
- A static member is shared by all objects of the class.

Static Members of a C++ Class

- It can be initialized outside the class using the scope resolution operator :: to identify which class it belongs to.
- All static data is initialized to zero when the first object is created, if no other initialization is present.

Concept: Static Members can be initialized outside the class

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

class Student {
    public:
        int rollnumber, age;
        static int m = 1; //Error
};
int main() {
    Student S1;
    return 1;
}
```

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

class Student {
    public:
        int rollnumber, age;
        static int m;
};

int Student::m=1; //Correct Initialization

int main() {
    Student S1;
    return 1;
}
```

Concept: A static member is shared by all objects of the class

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

class Student {
    public:
        int rollnumber, age;
        static int m;
};

int Student::m=1; //Correct Initialization
```

```cpp
int main() {
    Student S1;
    S1.m = 5;
    Student S2;
    cout<<S1.m<<endl<<S2.m<<endl;
    S2.m = 10;
    cout<<S1.m<<endl<<S2.m<<endl;
    Student S3;
    cout<<S3.m;
    return 1;
}
```

Static Member Functions of a C++ Class

- 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 **::**.

Static Member Functions of a C++ Class

- A static member function can only access static data member, other static member functions and any other functions from outside the class.

Concept: A static member function can only access static data member

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

class Student {
    public:
        int rollnumber, age;
        static int m;

    static modifyAge () //Error
    {
        age = 10;
    }
};
```

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

class Student {
    public:
        int rollnumber, age;
        static int m;

    static modifyAge () //Not an Error
    {
        m = 10;
    }
};
```

Concept: A static member function can only access other static member functions

```
1  #include <iostream>
2  using namespace std;
3
4  class Student {
5      public:
6          int rollnumber, age;
7          static int m;
8      void print()
9      {
10         cout<<"Hello"<<endl;
11     }
12     static modifyAge () |
13     {
14         m = 10;
15         print();   //Error
16     }
17 };
```

```
1  #include <iostream>
2  using namespace std;
3
4  class Student {
5      public:
6          int rollnumber, age;
7          static int m;
8      static void print()
9      {
10         cout<<"Hello"<<endl;
11     }
12     static modifyAge ()
13     {
14         m = 10;
15         print();   //Not an Error
16     }
17 };
```

Concept: A static member function can be called even if no objects of the class exist

```
21  int main() {
22      //Student S1;
23      Student::print();
24  }
```

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) Pointers 3) Function arguments 4) Class Data members 5) Class Member functions 6) Objects

1) Constant Variables in C++

- 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.

```
21  int main() {
22      const int a=1;
23      a=3; //Error
24  }
```

```
21  int main() {
22      const int a; //Error - Must be Initialized
23  }
```

2) Pointers with **const** keyword in C++

- Pointers can be declared using **const** keyword too.

- When we use **const** with pointers, we can do it in two ways, either we can apply **const** to what the pointer is pointing to, or we can make the pointer itself a constant.

2) Pointers with **const** keyword in C++

```
21  int main() {
22      const int* u; //pointer is pointing to a const variable.
23                    //Here, u is a pointer that can point to a const int type variable
24
25      int x = 1;          //Here, w is a pointer, which is const, that points to an int.
26      int* const w = &x;  //Now we can't change the pointer, which means it will always
27                          //point to the variable x but can change the value that it points to,
28                          //by changing the value of x
29  }
```

3) Function with constant Arguments

- We can make the arguments of a function as const. Then we cannot change any of them.

```
19  void f(const int i)
20  {
21      i++;     // error
22  }
```

4) Defining Class Data members as const

- These are data variables in class which are defined using const keyword. They are not initialized during declaration. Their initialization is done in the constructor.

```
 7   class test
 8   {
 9       const int i; //i is a constant data member in every object
10
11       public:
12           test (int x) : i(x) {} //i can only be set through member initializer
13   };
14
15   int main()
16   {
17       test t1(1);
18
19   }
```

5) Defining Class Object as const

- When an object is declared or created using the const keyword, its data members can never be changed, during the object's lifetime.

- Syntax:

```
const class_name object;
```

6) Defining Class's Member function as const

- A const member function never modifies data members in an object.

- Syntax:

```
return_type function_name() const;
```

Example for const Object and const Member function

```
1  class StarWars
2  {
3      public:
4      int i;
5      StarWars(int x)    // constructor
6      {
7          i = x;
8      }
9
10     int falcon() const  // constant function
11     {
12         /*
13             can do anything but will not
14             modify any data members
15         */
16         cout << "Falcon has left the Base";
17     }
18
19     int gamma()
20     {
21         i++;
22     }
23 };
```

Example for const Object and const Member function

```
25  int main()
26  {
27      StarWars objOne(10);          // non const object
28      const StarWars objTwo(20);      // const object
29
30      objOne.falcon();      // No error
31      objTwo.falcon();      // No error
32
33      cout << objOne.i << objTwo.i;
34
35      objOne.gamma();      // No error
36      objTwo.gamma();      // Compile time error
37  }
```

mutable Keyword

- mutable keyword is used with member variables of class, which we want to change even if the object is of const type. Hence, mutable data members of a const objects can be modified.

Example: Mutable

```
1  class Zee
2  {
3      int i;
4      mutable int j;
5      public:
6      Zee()
7      {
8          i = 0;
9          j = 0;
10     }
11
12     void fool() const
13     {
14         i++;    // will give error
15         j++;    // works, because j is mutable
16     }
17 };
18
19 int main()
20 {
21     const Zee obj;
22     obj.fool();
23 }
```

Reading Assignment!

- **Dietel, 7th Edition**
- **Classes: A Deeper Look, Part 2**
- **10.2 const (Constant) Objects and const Member Functions**
- **10.6 static Class Members**

Exercise!

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

class Point
{
    int x, y;

    public:
    Point(int i = 0, int j =0)
    { x = i; y = j;  }

    int getX() const { return x; }

    int getY() {return y;}
};
```

```
int main()
{
    const Point t;
    cout << t.getX() << " ";
    cout << t.gety();
    getchar();
    return 0;
}
```

Exercise!

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

class Test
{
    static int x;

    public:

    Test() { x++; }

    static int getX() {return x;}
};

int Test::x = 0;
```

```
int main()
{
    cout << Test::getX() << " ";
    Test t[5];
    cout << Test::getX();
    getchar();
    return 0;
}
```

Exercise!

```
class A;

void print(A a);

class A
{
     int x;

     public:
          A()
          {
               x = 1;
          }

           get() const
          {
          A a;
               print(a);
               //x = x + 3;
               //get_c();
          }

          get_c()
          {
               x = x + 4;
               cout<<"C";
          }
};
```

```
void print(A a)
{
     cout<<"A";
     //A a;
     a.get_c();
}


int main()
{
//   A::get();
A a1;
a1.get();

     getchar();
     return 0;
}
```

Saving Account Class

Create a SavingsAccount class. Use a static data member annualInterestRate to store the annual interest rate for each of the savers. Each member of the class contains a private data member savingsBalance indicating the amount the saver currently has on deposit. Provide member function calculateMonthlyInterest that calculates the monthly interest by multiplying the balance by annualInterestRate divided by 12; this interest should be added to savingsBalance. Provide a static member function modifyInterestRate that sets the static annualInterestRate to a new value. Write a driver program to test class SavingsAccount. Instantiate two different objects of class SavingsAccount, saver1 and saver2, with balances of $2000.00 and $3000.00, respectively. Set the annualInterestRate to 3 percent. Then calculate the monthly interest and print the new balances for each of the savers. Then set the annualInterestRate to 4 percent, calculate the next month's interest and print the new balances for each of the savers.