Object-oriented Programming

Lecture 7

Static Keyword

- **As local variable in global function**
  - *"Remembers" its previous value between function calls*

- **As a class member variable**
  - *Associated with class rather than an object. Is shared between all instances of class*

- **As a class member function**
  - *Associated with class rather than an object. Is shared between all instances of class*
  - *Cannot use instance members of class directly*

A simple function call

```cpp
void staticDemo()
{
    int val = 0;
    ++val;
    cout << "val = "  << val << endl;
}

int main()
{
    staticDemo();        // prints val = 1
    staticDemo();        // prints  val = 1
    staticDemo();        // prints val  = 1
}
```

Static Local Variables

When static local variables in a global function are used, they "*remember*" their values from previous function calls

Static Local Variables

```cpp
void staticDemo()
{
    static int val = 0;
    ++val;
    cout << "val = "  << val << endl;
}

int main()
{
    staticDemo();        // prints val = 1
    staticDemo();        // prints  val = 2
    staticDemo();        // prints val  = 3
}
```

Static Class Variables

- Static member variables are shared between all instances of a class

- Value of a static variable modified through one object will be reflected for all other objects

```
class xyz {
    static var_type var_name;
};
var_type xyz::var_name;
```

Example

```
class A
{
    static int val;

    public:
    A(int x)
    {
        val = x;
    }

    void setVal(int y)
    {
        val = y;
    }

    void show( )
    {
        cout << "Static Variable" << x;
    }
}; int A:: val;
```

```
int main()
{
        A ob1(10);
        ob1.show( );        // val becomes 10

        A ob2(20);
        ob2.show( );        // becomes 20

        ob2.setVal(30);
        ob1.show( );        //becomes 30
}
```

Static Member Functions

- Just like static variables, static member functions are shared between all instances of a class

- A non-static (instance) function can call other static functions (and use static variables)

- A static function cannot <u>directly</u> use instance members of the class

- However, they can do it by either making a local object or taking object as an argument

Example

```
class A
{
    public:
    A( )
    {  }

    static void f( )
    {
        f2( ); // will cause error
    }

    void f2( )
    {
        cout << "Instance function";
    }
};
```

```
int main()
{
        A::f( );
// we use class scope to call static functions
}
```

Example

```cpp
class A
{
    public:
    A( )
    {  }

    static void f( )
    {
        A temp(10);
        temp.f2( );          // OK
    }

    void f2( )
    {
        cout << "Instance function";
    }
};
```

```cpp
int main()
{
        A::f( );
// we use class scope to call static functions
}
```

Example

```
class A
{
    public:
    A( )
    { }

    static void f(A myOb)
    {
        myOb.f2( );        // OK
    }

    void f2( )
    {
        cout << "Instance function";
    }
};
```

```
int main()
{
        A ob(10);
        A::f(ob);
// passing object as an argument
}
```