COP 3331
OBJECT ORIENTED DESIGN
SUMMER 2017

WEEK 4 – WEDNESDAY (JUNE 7TH):

- MORE ON OPERATOR OVERLOADING

USF
UNIVERSITY OF SOUTH FLORIDA

RECALL:

- C++ allows you to redefine standard operators <u>when used with class objects</u>

- Why is this necessary?
  - Assignment and member selections are the only built-in operations on classes
  - Therefore, other operators can’t be applied directly to class objects

- Operator overloading provides a way to create more intuitive code

RECALL:

- Syntax:

```
returnType operator operatorSymbol(formal parameter list)
```

- To overload an operator for a class:
  - Include operator function in the class definition
  - Write the definition of the operator function

EXAMPLE:

- Let’s explore the “Student Test Score” and “FeetInches” examples in detail on Canvas

OVERLOADING ++ AND -- OPERATORS

- There are different procedures for overloading prefix and postfix operators

- Recall on prefix vs. postfix:
  - int i = 5;
  - int j = ++i; // the value for j is 6, the new value for i
  - int k = i++;// the value for k is 5, the old value for i

OVERLOADING ++ AND -- OPERATORS

- There are different procedures for overloading prefix and postfix operators

- Prefix syntax:
  - Prototype:

```
className operator++();
```

  - Definition:

```
className className::operator++()
{
     //increment the value of the object by 1
    return *this;
}
```

OVERLOADING ++ AND -- OPERATORS

- Example:

```
FeetInches FeetInches::operator++()
{
   ++inches;
   simplify();
   return *this;
}
```

- The function works as follows:
  - First, the function increments the object’s inches member
  - Then, it calls the simplify function and
  - Then, dereferenced ‘this’ pointer is returned

OVERLOADING ++ AND -- OPERATORS

- The operator function allows the ++ to perform properly in statements like this:

```
distance2 = ++distance1;
```

- Remember, the above statement is equivalent to

```
distance2 = distance1.operator++();
```

OVERLOADING ++ AND -- OPERATORS

- To overload a post fix operator, you the following syntax:
  - Prototype:

```
className operator++(int);
```

  - Function Definition

```
className className::operator++(int u)
{
    className temp = *this;    //use this pointer to copy
                               //the value of the object
      //increment the object

    return temp;  //return the old value of the object
}
```

OVERLOADING ++ AND -- OPERATORS

- Example:

```
FeetInches FeetInches::operator++(int)
{
    FeetInches temp(feet, inches);
    inches++;
    simplify();
    return temp;
}
```

- This function works as follows:
  - The dummy parameter (int) tells the compiler that this function is designed to be used in postfix mode
  - The temporary local variable is a copy of the object being incremented before the increment takes place
  - The contents of temp is returned after inches is incremented and simplify called

OVERLOADING ++ AND -- OPERATORS

- Let’s see the full code for overloading the increment operator for class ‘FeetInches’.
- The code example is posted on Canvas.

ANNOUNCEMENT:

- Programming Assignment 3 will be posted today, June 7th , on Canvas.