CIS 251 • C++ Programming
Laboratory Assignment 12: Operators

In this laboratory assignment you will learn how to use C++ syntax to add operators to a class definition.

Background

The class definitions presented in chapter ten use member functions (methods) to allow a programmer to create and work with objects declared from the class type. Chapter eleven demonstrates how to overload the operators for arithmetic, comparison, output, and input so that a programmer can use more natural syntax in dealing with objects. These functions are included as friend functions in the class definition; they are not members of the class, but they have access as if they are members.

Step 1: Launch Visual Studio and Set Up an Empty C++ Project

Open Visual Studio from the Start Menu. Start a new project of installed template / type **Visual C++**, subtype **Win32**, template **Win32 Console Application**, project name and location as desired (usually the Visual Studio **Projects** folder), and **Create directory for solution** box **unchecked**. In the **Win32 Application Wizard**, go to **Application Settings**, **check** the box for **Empty project**, and click **Finish**.

Step 2: Download the Starter File and Add Identification Comments

You will not be starting this assignment with a blank code file. Instead, you will work with the class `DayOfYear` as developed in the lectures.

On the Blackboard assignment page, right-click the link for **FLast_Lab12.cpp** and choose the shortcut menu option **Save target as…** (or the equivalent in your browser). Change the file name to include your **first initial** and **last name** in place of **F** and **Last**, respectively. Save this file to your flash drive or, if working on your own computer, to another location you prefer. **If you are working on a campus computer and do not have a flash drive you, you may save the file to campus computer only temporarily; you must delete the file from the computer before you leave!**

Return to Visual Studio. In the **Solution Explorer**, right-click **Source Files**, and select **Add > Existing Item…** from the shortcut menu. Navigate to the location where you saved the starter file, select the file, and add it to your project.

At the top of this starter file, add a single block comment or a set of line comments including your name, the course number, the laboratory assignment number, the date (optional), and a brief description of the program's contents. Place each of these items on a separate line. You may also want to add your own inline comments to the existing code after reviewing the contents; however, these comments will not count toward the requirement for the assignment. As you revise the code, you may go ahead and add comments to the statements or groups of statements you add or change that you believe deserve additional explanation, or you may wait until after you have written your entire program to add comments.

Step 3: Replace the Input Method Prototype

In the `public` section of the class definition, find the prototype for the `input` method. Delete this prototype, and replace it with a `friend` declaration for an overloaded input operator. The return type of this function is a reference to an input stream, the function name is `operator` followed by the characters for the input stream insertion operator, the first parameter is a reference to an input stream, and the second parameter is a reference to an object of this class type.

The return type and first parameter should be written such that the function works for either `cin` or an input file stream variable.

Do not make the object reference parameter in an overloaded input operator constant: the input method needs the ability to change the object for which input is being obtained.

Step 4: Convert the Input Method Definition to an Input Operator Function

Locate the definition of the `input` method after the definition of the `main` function. Delete the header for this method, and replace the header with everything from the friend declaration except the key word `friend` and the semicolon at the end of the statement.

Friend functions do not require a type qualifier and scope resolution operator; adding these could cause compilation or runtime problems.

If you did not include parameter names in the friend declaration, make sure you add these in the function header.

The body of the function requires several modifications as follows:

- Remove any output statements that ask the user for values for the member variables, since the input may be coming from a file instead of from the user.
- Replace any references to `cin` with the name of your input stream reference parameter.
- Precede any references to members of the class (variables or methods) with the name of your object reference parameter and the dot operator.
- Add a statement at the end of the function body that returns the input stream parameter.

In the body of a friend function, a class member name by itself is insufficient. You must have a calling object (which may be a parameter) to refer to any member of the class.

Since the mutator methods in `DayOfYear` validate the values to be assigned to the fields, you should still use these in the overloaded input operator function.

When writing the definition of an overloaded input or output operator, you must return the stream reference parameter so it will be available to the next item in the input or output statement.

Step 5: Replace the Output Method Prototype

Return to the `public` section of the class definition, and find the prototype for the `output` method. Delete this prototype, and replace it with a `friend` declaration for an overloaded output operator. The return type of this function is a reference to an output stream, the function name is `operator` followed by the characters for the output stream insertion operator, the first parameter is a reference to an output stream, and the second parameter is a constant reference to an object of this class type.

If a function should not be allowed to modify the values in an object parameter, it should be a constant reference parameter instead of a regular reference parameter.

Step 6: Convert the Output Method Definition to an Output Operator Function

Locate the definition of the `output` method after the `main` function. Delete the header for this method, and replace the header with everything from the friend declaration except the key word `friend` and the semicolon.

The modifications to the body of this function are similar to those for the input operator function:

- Replace the reference to `cout` with the name of your output stream reference parameter.
- Add the object parameter name as a calling object for any reference to a class member.
- Add a statement that returns the output stream parameter.

Step 7: Move the Comparison Function Prototype

Currently, the program contains an external function named `equal` that compares the contents of two objects using calls to the class's accessor methods. Move the prototype of this function (currently just above the `main` function) to the `public` section of the class. Add the key word `friend` to the beginning of this prototype, and change the function name to the key word `operator` followed by the comparison operator for equality. Finally, change the parameters to be constant reference parameters to the two objects.

Step 8: Convert the Comparison Function to an Equality Operator Function

Locate the definition of the `equal` function after the `main` function. Delete the header for this function, and replace the header with everything from the friend declaration except the key word `friend` and the semicolon.

Now that the function is a friend of the class, it no longer needs to use the accessors to obtain the values of the member variables. Replace the calls to the accessors with the names of the member variables, leaving the calling objects as they are.

Step 9: Create Two Objects and Use the Operators

In the `main` function, declare two objects of the class type. You are not required to use the constructor that accepts arguments, as any arguments will be replaced by input values.

Prompt the user for input for each of the two objects, letting the user know the order in which to enter the values. Then, use the input operator to store the values entered by the user in the object. Use separate prompts before each input statement.

You won't be able to display a prompt for each member variable separately, so make sure the user knows to enter the values in the expected order.

After obtaining the input, use the output operator to display the values of the two objects, labeling each object's output separately.

Finally, use the equality comparison operator in an `if-else` statement to compare the two objects, letting the user know whether the dates entered are the same or different.

Step 10: Build Your Project and Test the Executable

Build your project. If there are syntax errors, double-click each error description to get an idea of where the error is. Once the build succeeds, select **Start Without Debugging** from the Debug menu. Verify that the output appears as expected.

Example Run (user input in **bold**; your formatting, labeling, and spacing may vary):

```
Enter the numeric month and day (separated by a space) of a date:
4 19
Enter the numeric month and day (separated by a space) of another date:
4 23
Date #1: month = 4, day = 19
Date #2: month = 4, day = 23
The dates are different.
```

Step 11: Add Inline Comments and Submit Your Code

Make sure that your code has **three or more inline comments** near the statements you added that you believe warrant the additional description (especially those that pertain to overloading and using operators with objects). These can be simple line comments at the end of a line of code that needs explanation, or block comments where each precedes a series of statements described in the comment.

In Blackboard, return to the Assignments page for the current assignment. Under the **Submission** section, click **Add Attachments**. Click **My Computer**; then, click the first **Browse…** button. Navigate to where you saved the source code file **FLast_Lab12.cpp** (with your name), select it, and then click **OK**. Verify that your file appears in the Submission section, and click the **Submit** button at the bottom of the page.

If you do not click the Submit button, your attachment will not be saved!