CIS 251 • C++ Programming
Laboratory Assignment 2: Input and Output

In this laboratory assignment you will learn how to use variables, constants, and input and output statements to prompt the user for the radius of a circle and display its area and circumference.

Step 1: Launch Visual Studio

In the All Programs list in the Start Menu, open the folder for the version of Visual Studio you have installed, and select the icon for the program (it usually takes the form of an infinity sign). You may close the Visual Studio start page to minimize the visual clutter on the screen.

Step 2: Create a C++ Project

Click the **File** menu, select the **New** submenu, and select the item **Project…** from that submenu. In the dialog that appears, make the following selections:

1. Under **Installed Templates** (or **Project Types**), select **Visual C++**. It may appear under **Other Languages**.
2. Under Visual C++, select the subtype **Win32**.
3. In the **Templates** area to the right, select **Win32 Console Application**.
4. At the bottom of the dialog, give the project a name. The name you use for the project is not important: you will not submit the files that store the project settings for any assignment.
5. Note the location where the project will be created: it is usually placed in the **Documents** folder for the currently logged in user, within the **Visual Studio 20xx** subfolder, and within the **Projects** sub-subfolder. A folder with your project name will be created in the Projects folder.
6. To prevent the creation of an additional folder within the project folder, make sure that the checkbox marked **Create directory for solution** is **unchecked**.

Once you click the **OK** button, a second window, the **Win32 Application Wizard**, appears. Click **Next** to proceed to **Application Settings**. The important setting on this screen is to select the checkbox labeled **Empty project**. Finally, click **Finish**.

Step 3: Add a C++ Code File to Your Project

In the **Solution Explorer** (select it from the **View** menu if you don't see it), right-click the **Source Files** folder, select the **Add** submenu, and select **New Item…** from the submenu. In the **Add New Item** dialog:

1. Select the template or category **Code**.
2. Select the file type **C++ File (.cpp)**.
3. Enter the file name **FLast_Lab02.cpp** (your first initial, your last name, an underscore, and Lab02).
4. Use the **Browse…** button to navigate to your flash drive and the subfolder in which you want to save your code. (If you don't have your flash drive with you, you may place the file in the project directory or elsewhere on the hard disk, but be sure to delete the file once you've turned in your submission.)

You should now see an empty code window on your screen.

Step 4: Begin Your Code

The first element you should provide in your code is a set of comments identifying yourself and the purpose of your program. You may enter this as a single block comment or as a set of line comments. (See section 2.5 in your textbook, or the outline from Lecture 2, for more details on the syntax of comments.)

Comments for the code you submit for this class should include your name, the course number, the laboratory assignment or project number, and a brief description of the program's contents. You may also wish to include the date (perhaps after the laboratory assignment number). Place each of these items on a separate line, either as individual line comments or in a block comment:

```
// Student Name
// CIS 251 (Summer 2012)
// Laboratory Assignment 2
// Thursday, May 31, 2012
// In this assignment, I am practicing basic C++ statements such as
// declaring variables and constants, input and output, and assignment.
```

As you continue your code, you may go ahead and add comments to the statements or groups of statements that you believe deserve additional explanation. You may also wait until after you have written your entire program to add comments. On each assignment you submit, you will be required to provide a minimum number of comments (usually listed at the end of the assignment description).

It's a good idea to go ahead and set up the basic structure of your code as well, including `include` statements, the standard namespace, the `main` function header and brackets, and the `return` statement. The following portions will be standard across the majority of your programs this semester:

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

int main()
{
	(Leave a blank line here for the space where you'll enter the statements for your program)
	return 0;
}
```

HINT: Keep the indentation in your program consistent. The opening and closing curly brackets for the main function are usually flush with the left side of the document window, and the contents of `main` are indented one level, usually with a tab.

Step 5: Declare Your Variables and Constant

This program deals with four numeric quantities: the radius, area, and circumference of a circle; and the value of pi. Each of these quantities could be a decimal number, so the data type `double` is most appropriate.

Return to the blank line between the inner set of curly brackets for the `main` function. Insert **declarations** for all three variables and the constant. You may declare the variables individually (one variable per statement) or all at once (all three variables in a single declaration statement). The constant should be declared similarly, except that you add the key word `const` before the data type, and you must initialize the constant in the declaration statement (use the value `3.14` for this program). You may want to use all uppercase letters in the name of your constant. Make sure you end each declaration with a semicolon.

Step 6: Obtain Input from the User

Since the input in this program will come from the user, you should display a prompt to let the user know what to enter. Add an **output** statement to ask the user for the radius in a manner similar to this:

```
Please enter the radius of your circle in inches:
```

Follow this with an **input** statement that reads the user's input into the variable you declared for the radius.

WARNING: Watch your angle brackets! The direction of the brackets differs between input and output statements. If you receive over twenty errors when compiling, you may have bracket issues.

Step 7: Perform the Calculations

There are two values that the program will determine from the user's input based on these formulas:

area = π * radius^2 (or π * radius * radius)
circumference = π * diameter (or π * 2 * radius)

Write two **assignment** statements to calculate these values and store them in their respective variables.

Step 8: Display the Results

Without output statements, the user is unaware of the values stored in the program's variables. It would be nice to ensure that the program displays these numbers in fixed (non-scientific) format with a required decimal point and a specific number of digits shown after the decimal (three at the most). Insert the three **output formatting** statements you saw in Lecture 2 (also in section 2.2 of your textbook).

Then, use a set of **output** statements to display the results in the following format (words in *italics* represent variables):

```
If a circle has a radius of radius inches,
its area is area square inches, and its
circumference is circumference inches.
```

Step 9: Build Your Project

To determine if there are any syntax errors in your code, and to generate the executable once your code is free of syntax errors, you'll need to select the item **Build Solution** from the **Build** menu. The results of the build process will appear in the **Output** area below your code. If the Output area indicates that there are syntax errors in your code, you may double-click the line that describes the syntax error to see the line that contains the error (or, in some cases, the line after the error).

HINT: Don't try to correct all of your syntax errors at once. Sometimes a single error in your code will generate several different syntax error messages. Correcting that error and compiling your code again will reveal what errors remain.

Step 10: Run the Executable

Once you receive a message in the Output area that the build succeeded, you're ready to test your program. Remember: the **Start Without Debugging** option in the **Debug** menu pauses at the end of the program to let you see the final output, whereas the **Start Debugging** option does not. (See Laboratory Assignment 1 for details on how to add this command to the Debug menu if you don't see it.)

Once you select Start Without Debugging from the Debug menu, a console window appears. The program will pause to wait for user input. Enter some sample data for the radius, and examine the results. You will need to select the Start Without Debugging command several times to test the program with different data sets. (You will learn in a later chapter how to code a program such that it will allow you to enter several sets of data in one program run.)

Example Run (assume the user enters the value at the end of the first line):

```
Please enter the radius of your circle in inches: 5
If a circle has a radius of 5.000 inches,
its area is 78.500 square inches, and its
circumference is 31.400 inches.
```

Another Example Run:

```
Please enter the radius of your circle in inches: 2.5
If a circle has a radius of 2.500 inches,
its area is 19.625 square inches, and its
circumference is 15.700 inches.
```

Step 11: Add Inline Comments and Submit Your Code

Make sure that your code has **three or more inline comments** near the statements that you believe warrant the additional description. 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.

You are not required to attach your entire project folder for any assignment, as each student may be using a different version of Visual Studio or a different environment altogether. Only your source code file (the file that ends in .cpp) is required.

In Blackboard, return to the Assignments page for the current assignment. Under the **Submission** section, click the **Add Attachments** button. On the left side of the resulting window, click **My Computer**; then, click the first **Browse…** button on the right. Navigate to where you saved the source code file, 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.

WARNING: If you do not click the Submit button, your attachment will not be saved! Once you click the Submit button, your assignment should move from the Inbox to the Submitted tab.