CIS 251 • C++ Programming
Laboratory Assignment 3: Branching

In this laboratory assignment you will learn how to use branching structures to determine the amount due from a customer at a fast food restaurant based on the customer's menu choices.

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

From the All Programs list in the Start Menu, open the folder for Visual Studio, and select the icon for the program. Close the start page if you wish to minimize the visual clutter on the screen.

From the **File** menu, select the **New** submenu, and select **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**.

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_Lab03.cpp** (your first initial, your last name, an underscore, and Lab03).
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 2: 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. Include 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. (Review the Laboratory Assignment 2 instructions for an example of these comments.) 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.

It's a good idea to go ahead and set up the basic structure of your program as well:

```
#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;
}
```

Step 3: Declare Your Variables and Constant

This program deals with four numeric quantities: the **subtotal**, **tax amount**, and **total** for the order (variables); and the **sales tax rate** (a constant). Each of these quantities could be a decimal number, so the data type `double` is most appropriate. The program also prompts the user for three choices from a series of menus; you may set up the program such that the user enters a whole number (`int`), a character (`char`), or a word (`string`) for each of these three menu choices (remember that the data type `string` requires an extra `#include` statement).

Return to the blank line between the inner set of curly brackets for the `main` function. Insert **declarations** for all three `double` 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 `0.08` for the sales tax rate). You may want to use all uppercase letters in the name of your constant. Make sure you end each declaration statement with a semicolon, but don't put semicolons between variables in the same statement (use commas instead).

Also add **declarations** for the three menu choice variables. Declare these variables with a data type consistent with how you want the user to respond to each menu.

Be careful with capitalization within variable / constant names and punctuation in the declaration statements (commas, semicolons). Your spelling of identifiers must be consistent throughout the program.

Step 4: Obtain Input from the User

The pricing structure for the restaurant is as follows:

Hamburger	$1.29	Medium Fries	$1.59	Small Drink	$1.59
Cheeseburger	$1.49	Large Fries	$1.79	Large Drink	$1.99

Each customer must order an entrée, but the fries and drink are optional.

Begin your program by displaying a greeting (the restaurant name is up to you) and the first menu in **output** statements:

```
Welcome to Vanilla Ice's Fast Fast Baby Restaurant!
Please choose your entree from the following:
1. Hamburger
2. Cheeseburger
Enter your selection now:
```

Remember to label the choices with the type of value you used for the input variable (a number, a character, or a word). Follow this with an **input** statement for the first menu choice variable.

WARNING: Although the user is choosing from two options, the user's choice is only one value. Don't use more than one variable for each menu.

HINT: Whenever the user is expected to make a selection from a menu, indicate to the user what input value represents each selection. These values can be numbers, letters, or entire words.

NOTE: Usually you want to prevent the program from continuing if the user provides bad input. However, you won't be able to handle this appropriately until you learn about loops.

Next, use **output** statements to describe the options for French fries (include an option for no French fries):

```
Would you like French fries with your entree?
0. None
1. Medium French fries
2. Large French fries
Enter your selection now:
```

Follow this with an **input** statement for the second menu choice variable.

Finally, use output statements to describe the drink options (again giving the user the option of no drink):

```
Would you like a drink with your entree?
0. None
1. Small drink
2. Large drink
Enter your selection now:
```

Follow this with an **input** statement for the third menu choice variable.

Step 5: Evaluate the Input

Depending on the type of variable used for input, you may be able to use `switch` structures, or you may be required to use `if` structures. Review chapter 3 and the outline from lecture 3 for details on each structure.

You'll need three separate sets of **branching structures**.

- For the first set, compare the input variable for the entrée to each of the two entrée options; if the entrée choice matches one of the entrée options, assign the price of that entrée to the subtotal variable for its starting value.
- For the second, compare the input variable for French fries to each of the two size options; if the French fries choice matches one of the size options, add the price for that size to the subtotal variable.
- For the third, compare the input variable for the drink to each of the two size options; if the drink choice matches one of the size options, add the price for that size to the subtotal variable.

WARNING: When using an `if` or `else if` statement, your Boolean expression must be in parentheses and should not end in a semicolon. When using a trailing `else`, don't include a Boolean expression.

WARNING: When using a `switch` structure, remember that you only put the variable, not a complete Boolean expression, in the parentheses at the top, and the header should not end in a semicolon. Also recall that each `case` statement is followed by a colon, and the action for each case must end with a `break`.

Step 6: Perform the Calculations

There are two final calculations to be made in **assignment** statements:

sales tax amount = subtotal * sales tax rate
total = subtotal + sales tax amount

Use the variables and constant you declared at the top of `main` to perform these calculations.

WARNING: The statement to calculate this result must come after the decision structures that determine the subtotal. Never assign a formula to a variable before you've obtained the values for the variables in that formula!

Step 7: Display the Results

When displaying the results using **output** statements, label each value so that the user knows what it represents (words in *italics* represent variables):

```
Subtotal:    $subtotal
Sales Tax:   $sales tax amount
Total:       $total
Please pull forward to the pickup window!
```

HINT: Don't forget the three statements that tell the console to display decimal values in fixed notation, with a required decimal point, and two digits after the decimal point! Place these statements before your output.

Step 8: 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 9: Run the Executable

Once you receive a message in the Output area that the build succeeded, select **Start Without Debugging** from the **Debug** menu, and enter some sample data for each choice in the console. You will need to select the Start Without Debugging command several times to test the program with different data sets.

Example Run: If the user selects a cheeseburger, medium fries, and a large drink,

```
Subtotal:   $5.07
Sales Tax:  $0.41
Total:      $5.48
```

Example Run: If the user selects a hamburger, no fries, and a small drink,

```
Subtotal:   $2.88
Sales Tax:  $0.23
Total:      $3.11
```

Step 10: 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 (especially those that deal with branching). 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.