

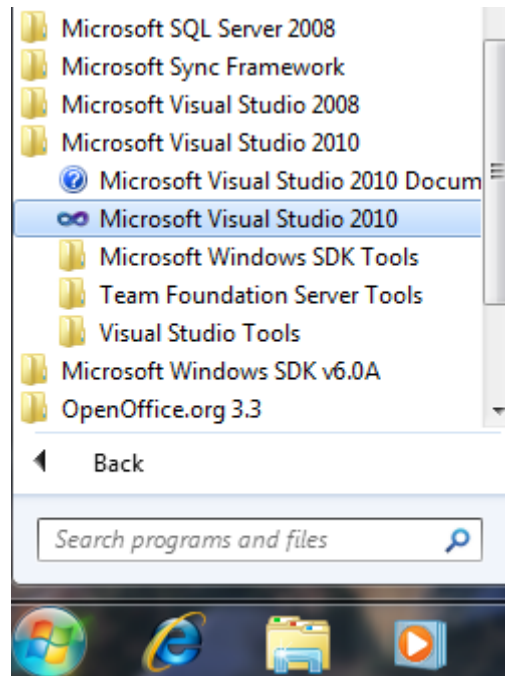
CIS 251 • C++ Programming

Laboratory Assignment 1: Visual Studio

In this laboratory assignment you will learn how to create and set up a C++ project in Visual Studio, how to add a C++ source code file to the project, how to build the project, and how to execute the resulting program. These instructions should suffice for any recent version of Visual Studio (2005, 2008, or 2010), although some of the screens may vary; ask your instructor for assistance should you have difficulties with your installation of Visual Studio. Remember: you are not required to use Visual Studio when coding at home, but the software is available to you for free via the MSDN Academic Alliance web site.

Step 1: Launch Visual Studio

Once you have installed Visual Studio on your computer, it should be accessible via 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).

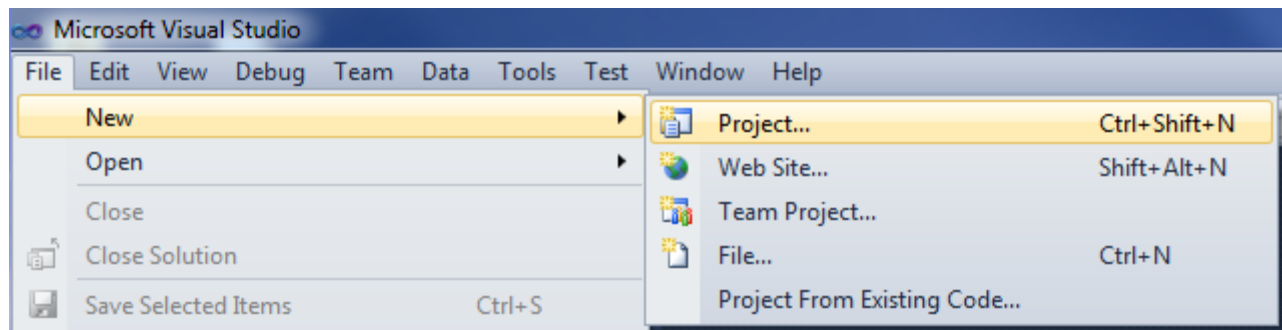


NOTE: Your screen's appearance will vary somewhat depending on the version of Windows you have installed, the theme selected on your computer, etc.

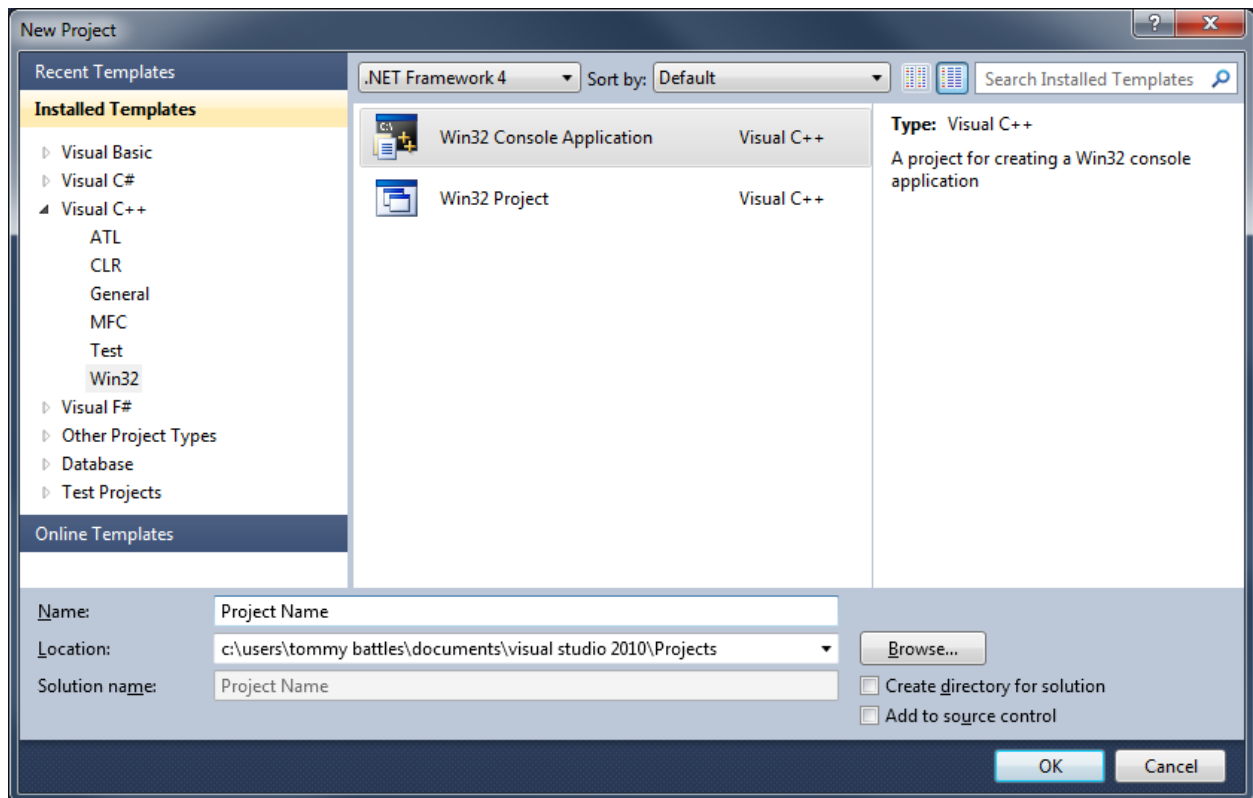
Visual Studio typically launches with a start page that displays developer news; you may close this start page to minimize the visual clutter on the screen.

Step 2: Create a C++ Project

In order for Visual Studio to use a C++ compiler to convert your code to the appropriate executable form, you need to create a C++ project. The first step is to click the **File** menu, select the **New** submenu, and select the item **Project...** from that submenu.



The following dialog appears:

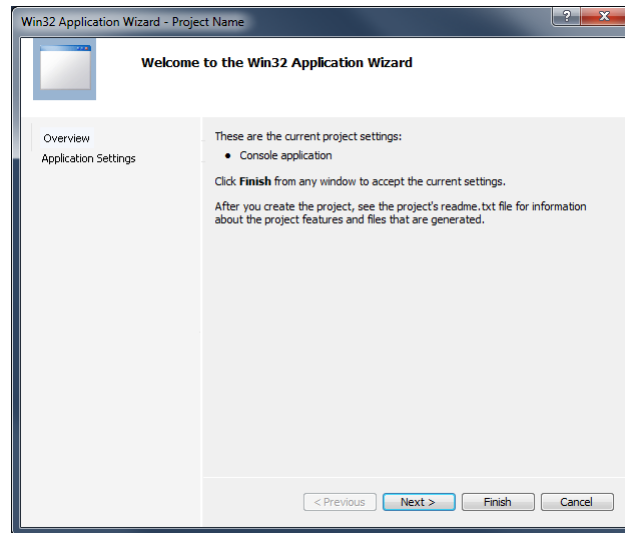


There are several important settings to select in this dialog:

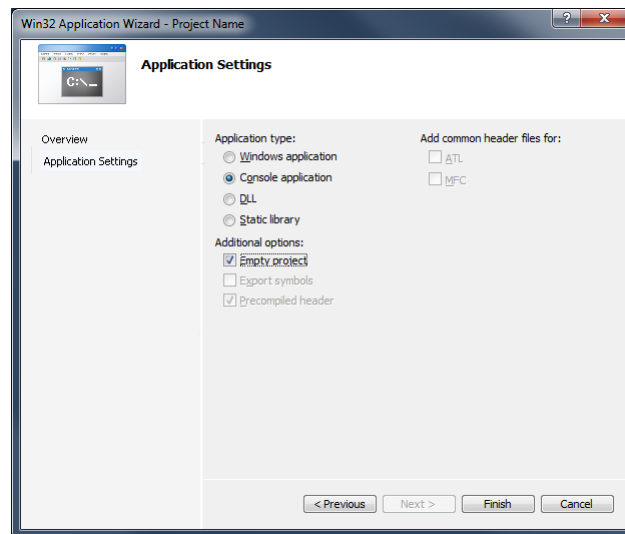
1. Under **Installed Templates** (or **Project Types**), select **Visual C++**. It may appear under **Other Languages**.
2. Under Visual C++, select the subtype **Win32**.
3. In the **Templates** area to the right, select **Win32 Console Application**.
4. At the bottom of the dialog, give the project 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**.

NOTE: Although you may choose to place your project folder on a flash drive, the performance when compiling the program will be sluggish. It's usually best to leave the project folder on the hard disk and add the code from your flash drive separately; you'll see how to do this momentarily.

Once you click the **OK** button, another window, the **Win32 Application Wizard**, appears:



There is one additional setting to be adjusted. Click **Next** to proceed to **Application Settings**.

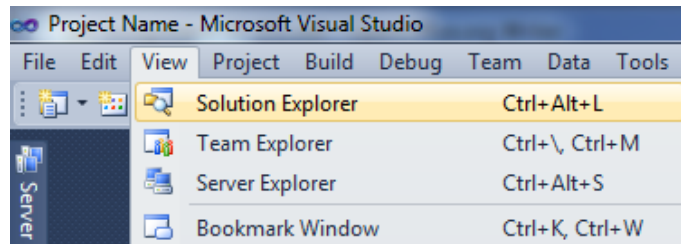


The important setting on this screen is to select the checkbox labeled **Empty project**. Then click **Finish**.

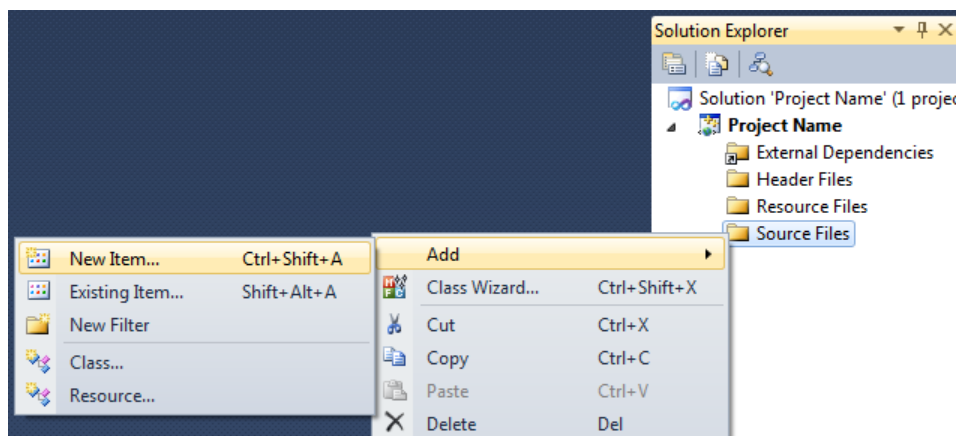
WARNING: If you do not select **Empty project**, your project will be created with additional files that will conflict with the nature of the projects we create in this class. Don't miss this important step!

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

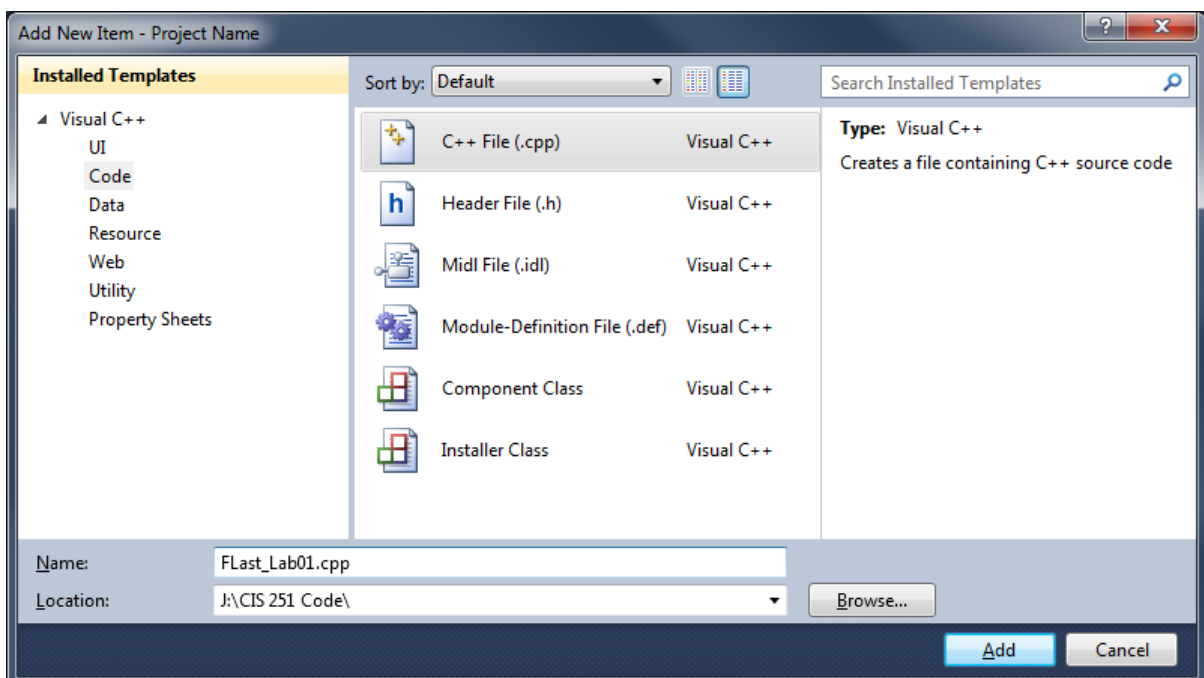
After you create a project, you should see the **Solution Explorer**, a hierarchical view of files associated with your project, docked to either side of the Visual Studio window. If you do not see this task pane, you need to select it from the **View** menu:



Once you see the Solution Explorer, right-click the **Source Files** folder, select the **Add** submenu, and select **New Item...** from the submenu:



The **Add New Item** dialog appears:



Be sure to select the following settings:

1. Select the template or category **Code**.
2. Select the file type **C++ File (.cpp)**.
3. Enter a file name appropriate for the code you are about to enter. If this is a lab assignment or a project, you will be instructed as to what to name the file. For this lab assignment, use the file name form **FLast_Lab01.cpp** (your first initial, your last name, an underscore, and Lab01).
4. It's best to keep your code on your flash drive for portability. 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.)

WARNING: This is the file you will submit when you are finished! Make sure you know where the file is saved!

Once you click the **Add** button, the dialog disappears, and an empty code window appears in the main area of the program window.

Step 4: Enter Your Code

Enter the code from Display 1.8 in your textbook. The code also appears below:

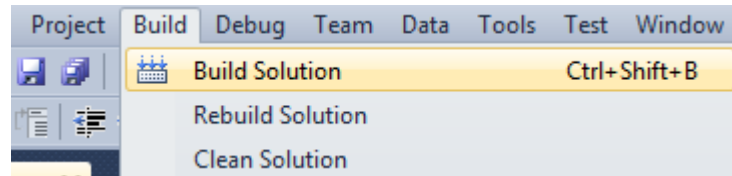
```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      int number_of_pods, peas_per_pod, total_peas;
6      cout << "Press return after entering a number.\n";
7      cout << "Enter the number of pods:\n";
8      cin >> number_of_pods;
9      cout << "Enter the number of peas in a pod:\n";
10     cin >> peas_per_pod;
11     total_peas = number_of_pods * peas_per_pod;
12     cout << "If you have ";
13     cout << number_of_pods;
14     cout << " pea pods\n";
15     cout << "and ";
16     cout << peas_per_pod;
17     cout << " peas in each pod, then\n";
18     cout << "you have ";
19     cout << total_peas;
20     cout << " peas in all the pods.\n";
21     return 0;
22 }
```

Be careful to enter the code exactly as it appears in the textbook.

HINT: Many code editors highlight C++ syntax in various ways. In Visual Studio, keywords appear in blue, and text within certain special character pairs (angle brackets, quotation marks) appears in red. If the highlighting on your screen does not match what is shown above, you may have misspelled a keyword, or you may have forgotten the closing character of a character pair.

Step 5: Build Your Project

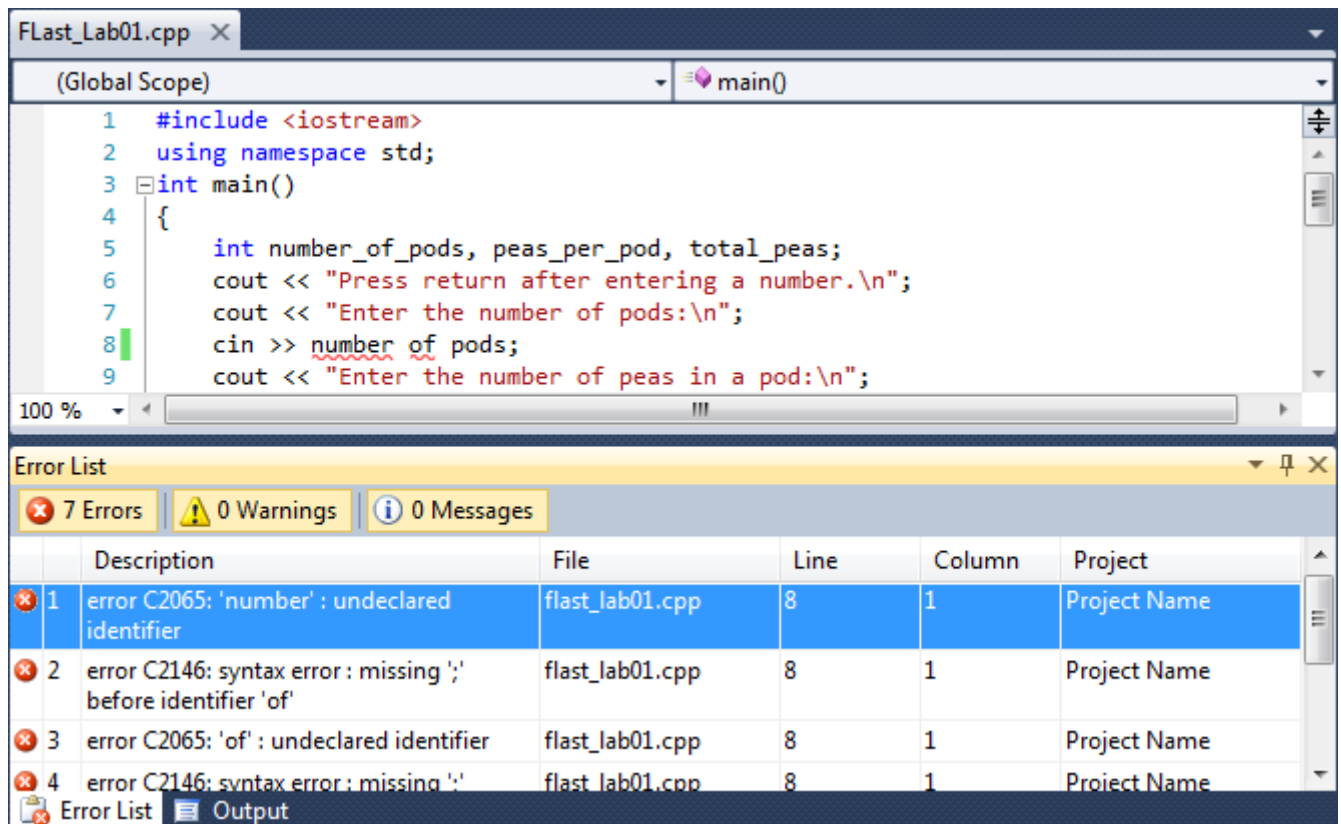
As we discussed in the lecture, the process of converting our code into an executable program includes compiling (generating object code from our source code) and linking (combining our object code with that of predefined functionality). However, many environments combine these two steps into a single step, typically named **Build**. 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:

```
Output
Show output from: Build
1>----- Build started: Project: Project Name, Configuration: Debug Win32 -----
1> Flast_Lab01.cpp
1> Project Name.vcxproj -> c:\users\tommy battles\documents\visual studio 2010\Projects\Project Name\Debug\Project Name.exe
===== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped =====
```

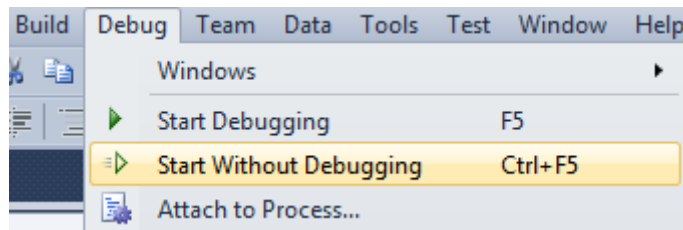
If the Output area indicates that there are syntax errors in your code, you may double-click the line that describes the syntax error to see the line that contains the error (or, in some cases, the line after the error). In the following example, I used spaces in place of underscores in a variable name, and seven syntax errors are listed from that one mistake:



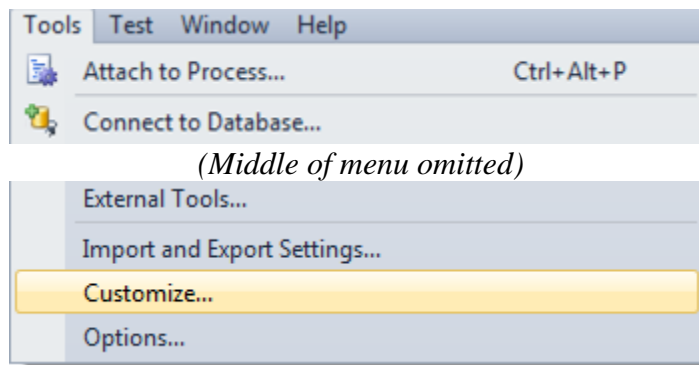
Step 6: Run the Executable

The process of removing syntax errors from a program may take some time. You may want to correct a single error in the code and run the Build command again to see how many syntax errors remain in the list. Once you receive a message in the Output area that the build succeeded, you're ready to test your program.

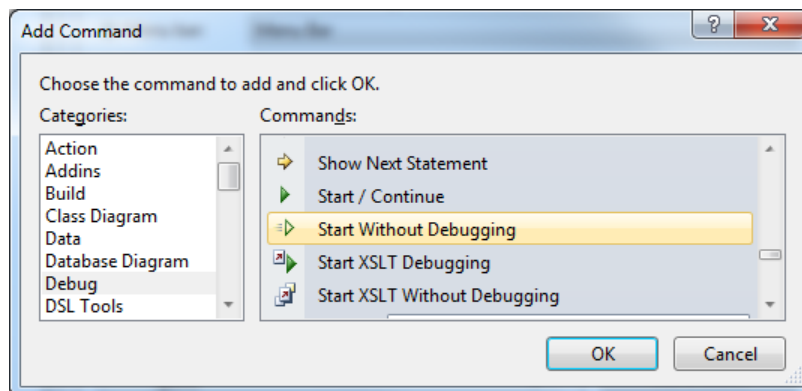
There is a command in the **Debug** menu named **Start Debugging** that will allow you to interact with the program to a degree. However, this command will not pause once the program is complete to allow you to see the final output to the screen. Another option, **Start Without Debugging**, will allow you to see the end result:



Visual Studio may not be configured to display this menu item by default, so you may need to add the item to the menu. If so, go to the **Tools** menu and select **Customize...**:



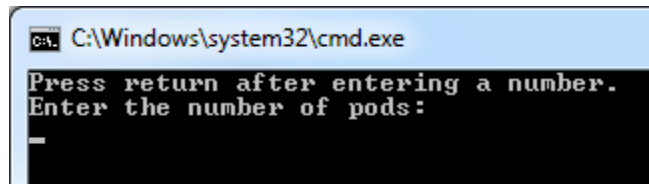
In the Customize dialog, click the **Commands** tab. Under **Controls**, select the **Debug** menu, and then click the **Add Command** button. In the Add Command dialog, select the category **Debug**. The Commands list on the right is sorted alphabetically; scroll down until you encounter the item **Start Without Debugging**:



Click **OK** to add the item to the Debug menu, and click **Close** to close the Customize dialog.

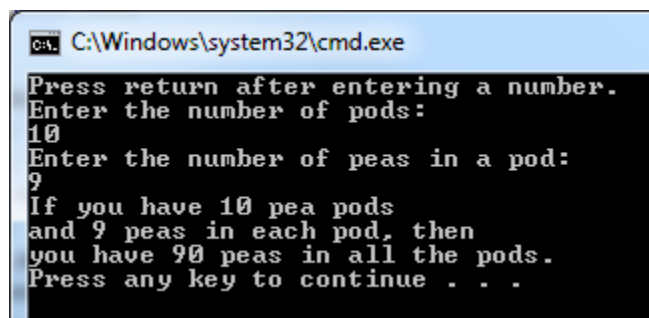
NOTE: The process of adding an item to a menu varies among versions of Visual Studio. You may have to drag the item to the menu where you want the item to appear.

Once you select Start Without Debugging from the Debug menu, a console window appears. We will use the console interface (a text-driven window) for the entire semester; we will not be advancing into GUI programming, as the details of this are platform-specific. The program will pause as it waits for user input (the first `cin` statement):



```
C:\Windows\system32\cmd.exe
Press return after entering a number.
Enter the number of pods:
_
```

Enter some sample data for each prompt, 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.)



```
C:\Windows\system32\cmd.exe
Press return after entering a number.
Enter the number of pods:
10
Enter the number of peas in a pod:
9
If you have 10 pea pods
and 9 peas in each pod, then
you have 90 peas in all the pods.
Press any key to continue . . .
```

Step 7: Submit Your Code

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.