Using MS Visual Studio 2005

Many thanks to Matt Ginzton, Robert Plummer, Erik Neuenschwander, Nick Fang, Justin Manus, Andy Aymeloglu, Pat Burke, and Tom Hurlbutt.

Using a PC

If you wish to do your assignments on a PC, you will need either Windows 2000 or XP, and Microsoft Visual Studio 2005. Note that Windows 98 or Me are **not** able to run Visual Studio 2005. You should be familiar with the Windows operating system and PCs in general, and be willing and able to figure out the Visual Studio 2005 environment. The emphasis in class will be on general programming concepts, not on the workings of Visual Studio (often referred to as just VS), and the code you write will be virtually identical to code written by students using a Macintosh.

Getting started

In order to use your PC to write the programs for the class, you will need to install both a compiler and the special 106 libraries for your system.

Step #1: Getting and installing the Microsoft Visual Studio 2005 compiler

Windows Vista Users:

If you are running Windows Vista, you will also have to download and install two service packs. You should first install Visual Studio 2005 Service Pack 1 (`http://msdn2.microsoft.com/en-us/vstudio/bb265237.aspx`),
and then you should install Visual Studio 2005 Service Pack 1 Update for Windows Vista (`http://www.microsoft.com/downloads/details.aspx?FamilyID=90e2942d-3ad1-4873-a2ee-4acc0aace5b6&displaylang=en`). If you have any further problems, please let the head TA know.

Step #2: Getting the CS106 libraries

Next you will need to get the CS106 libraries that we'll be using throughout the course. Please use the following directions to install these libraries on your PC.

You can obtain the PC CS Libraries by going to the web address http://see.stanford.edu/materials/icspacs106b/PCLibs-VS2005-ADT-Installer.zip

1. Use your web browser to download the PCLibs for VS2005 archive from the location above. You will be asked what to do with the file PCLibs.zip; you should save it to a directory on your computer.
2. Unzip the file PCLibs.zip. Make sure you use the option to expand using original directories. If you do not have an unzip program, a good one is available at http://www.winzip.com/ (the Evaluation Version of WinZip may be sufficient for you for the time being unless you plan on regularly using it). After unzipping PCLibs.zip, you should end up with a few files.
3. Run (and quit) Visual Studio (you can go ahead and register your copy if it asks you). **It is important that you have done this at least once before proceeding to the next step**.
4. Double click the file `Setup.Exe` that was unzipped in step 2. This will install the libraries on your system.

Once you've completed the above steps, you're ready to roll. The rest of the handout deals with doing the actual assignments.

Setting up projects*

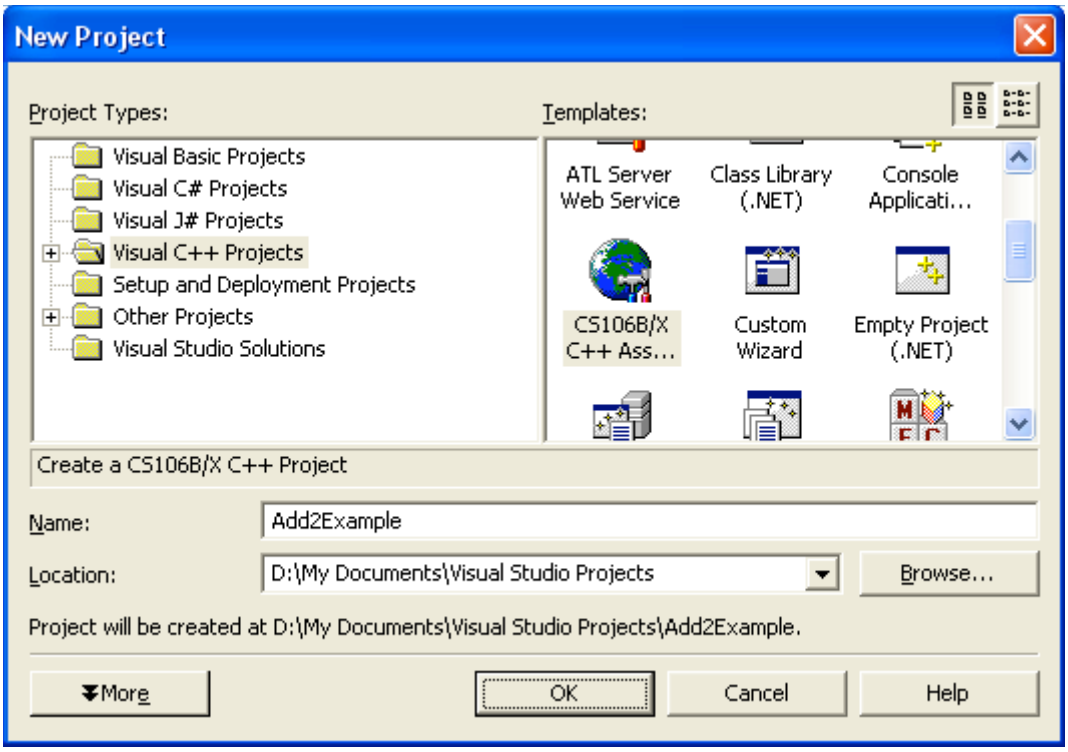
Every program written using Visual Studio has a project that indicates what different program files need to be compiled together in order for the complete program to work. We are going to create a project that uses a simple pre-written program, named `add2.cpp`. You can download the code for `add2.cpp` by using your web browser and visiting the following URL on the CS106B http://see.stanford.edu/materials/icspacs106b/add2.cpp Then, select the "Save As" option from the File menu to save the file to your own computer (if, instead, your web browser brings up a dialog box stating that it wants to open the file, choose the option to save the file to your own computer.) Once you have the .cpp file, you will need to create a Visual Studio project and add the .cpp file we provide. You will go through the basic process described below for each programming assignment that you do in this course, using your own code files.

1. Start Microsoft Visual Studio 2005, and from the **File** menu choose **New** and then **Project…**
2. Select the Visual C++ Projects folder in the "Project Types" pane.
3. The type of application that you want to create is "CS106B/X C++ Assignment Wizard"; select this from the list on the left, and type a name for the project (avoid using spaces or hyphens). Be sure you have selected "CS106B/X C++ Assignment Wizard" as the type of

* Note: the following instructions are from Visual Studio 2003. Some things may be slightly different in 2005. If there is anything majorly different, we will post information about it on the course website.

the application! Most of the difficulties that students have when setting up projects in this course come from a failure to follow the instructions in this step. If you select another project type, the libraries will not work properly.

4. Here is what you should see after carrying out steps 1-3, opening a project and giving it a name (`Add2Example` in the screenshot shown on the next page).



5. Choose a location for your project. The default location is a good choice, but it is up to you. The name of your project will automatically be added to the end of the location you choose. If you would like to browse around to find the place you want to put the project, click the "Browse..." button. Note that the location name might be too long to show all at once in the Location field. To see all of it, click in the field and use the arrow or Home/End keys to move back and forth. When everything is set, click OK.

6. After clicking OK, the CS106B/X C++ Assignment Wizard might pop-up and ask if you want to create an empty .cpp file to get you started. Since for this example (and for most of your assignments) we'll be providing you with starter files, we can go ahead and uncheck the box and click "Finish" to complete the process.

7. Most assignments will come with a set of starter files. This will usually include a mostly-empty file for the main program, some include (.h) files, libraries used for that particular assignment, and maybe some other C++ files (.cpp). These will usually come in the form of a .zip file that you will download to your computer. At this point you should leave Visual Studio (you needn't shut the program down), go download the appropriate .zip file for a given assignment from the class website (if you haven't already), and unzip the files into the folder (directory) for your project. Note the order here: <u>first</u> you create the new project

(steps 1-6 above), then you unzip the files into the folder created by Visual Studio for the project. For the `Add2Example` above, go ahead and copy the `add2.cpp` file to your project directory.

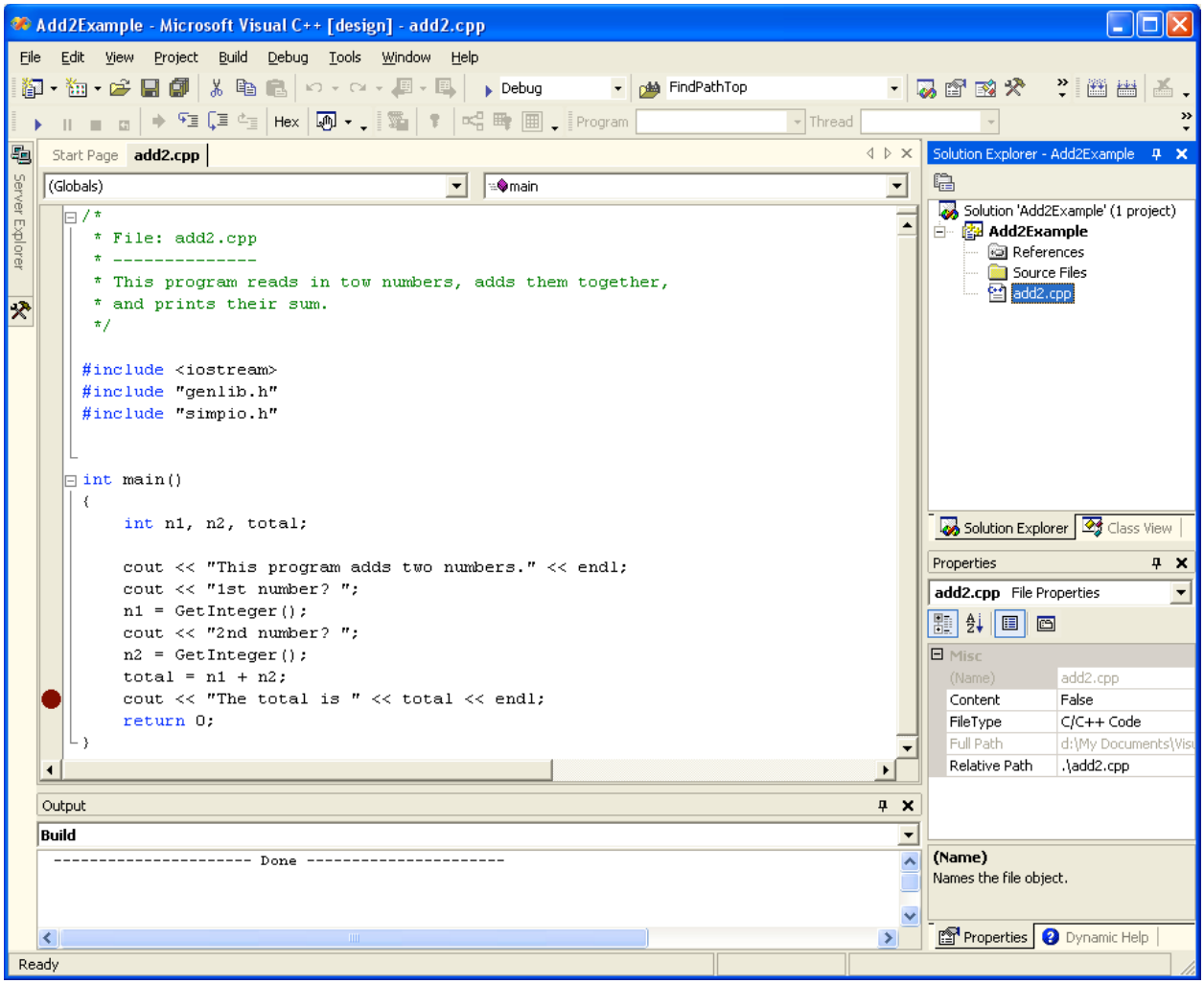
8. Now return to Visual Studio, because it is time to add the file(s) to your project (just having them in the directory is not sufficient). If you previously closed your project (or Visual Studio), make sure to open your project. From the **Project** menu, select the **Add Existing Item…** command. This brings up a dialog box that will let you add individual or groups of files to the project. Add the provided starter files with the extensions .cpp and .lib.

9. If you need to create .cpp files of your own, select **Add New Item…** from the **Project** menu. Select **C++ File** from the Template pane, and type the name of the file name (such as mycode.cpp), and click OK. The file will be added and you will be taken to an empty editor window, ready to type in your code. Be sure you provide the .cpp extension to the file name, so that everything works. Since the `add2.cpp` file is the only source code file we'll be using for the `Add2Example` project, we don't need to worry about this now, but be aware that you'll be creating new .cpp files in future assignments.

10. To compile and link all the code in your project, select **Rebuild Solution** from the **Build** menu.

11. To run the program with the debugger, set a breakpoint where you would first like the program to stop by placing the cursor in the desired line of code and pressing **F9**. You should see a red dot appear in the column to the left of the line of code indicating that line now contains a breakpoint. (Note that pressing **F9** on a line that already contains a breakpoint will remove the breakpoint.) After inserting the breakpoint, select **Start** from the **Debug** menu, or press **F5**. Of course, you can also just run your program without having it stop anywhere by picking the **Start** command from the **Debug** menu without previously setting any breakpoints.

    In the screenshot below, I've placed a breakpoint on line 23, the last `cout` statement of the program in my `Add2Example` project:



Running your program

Writing a program and typing it in is hardly the end of the programming process. You need to test your program and make sure that it works. When you pulled down the **Debug** menu and selected **Start**, the system translated the source program into the internal language of the machine. This process is called *compiling*. If you've done everything right, the compilation process will finish in a few seconds, and you can test your program to see if it works. If it does, awesome! If the program is for an assignment, print out a copy of the program and go on to the next problem.

Chances are, however, that on your real assignments you won't be quite so lucky. Somewhere in typing in your program, you will have entered something incorrectly or made some other sort of mistake. Perhaps your program does not work as intended, or, worse yet, does not even make it through the compilation process. Making such errors seems to be an unavoidable part of programming. When errors do occur, it is time to sit down, find them, and fix them. This is the process of *debugging*.

Debugging syntax errors

The errors that you are likely to encounter fall into two principal classes. The first class of errors is *syntax errors*, which encompasses those cases in which you violate some linguistic rule of the programming language. As the C++ compiler goes through your program to translate it into its machine-language form, it has encountered some piece of the program that it can't understand. Such errors may be typographical (if, for example, you spell some word incorrectly) or they may reflect a lack of understanding of the language rules.

When the compiler encounters a syntax error when you are trying build your program, it lists the error in bottom pane of the VS application. You can then go back to your program, find the error, and make the necessary correction. Save the file and try rebuilding it.

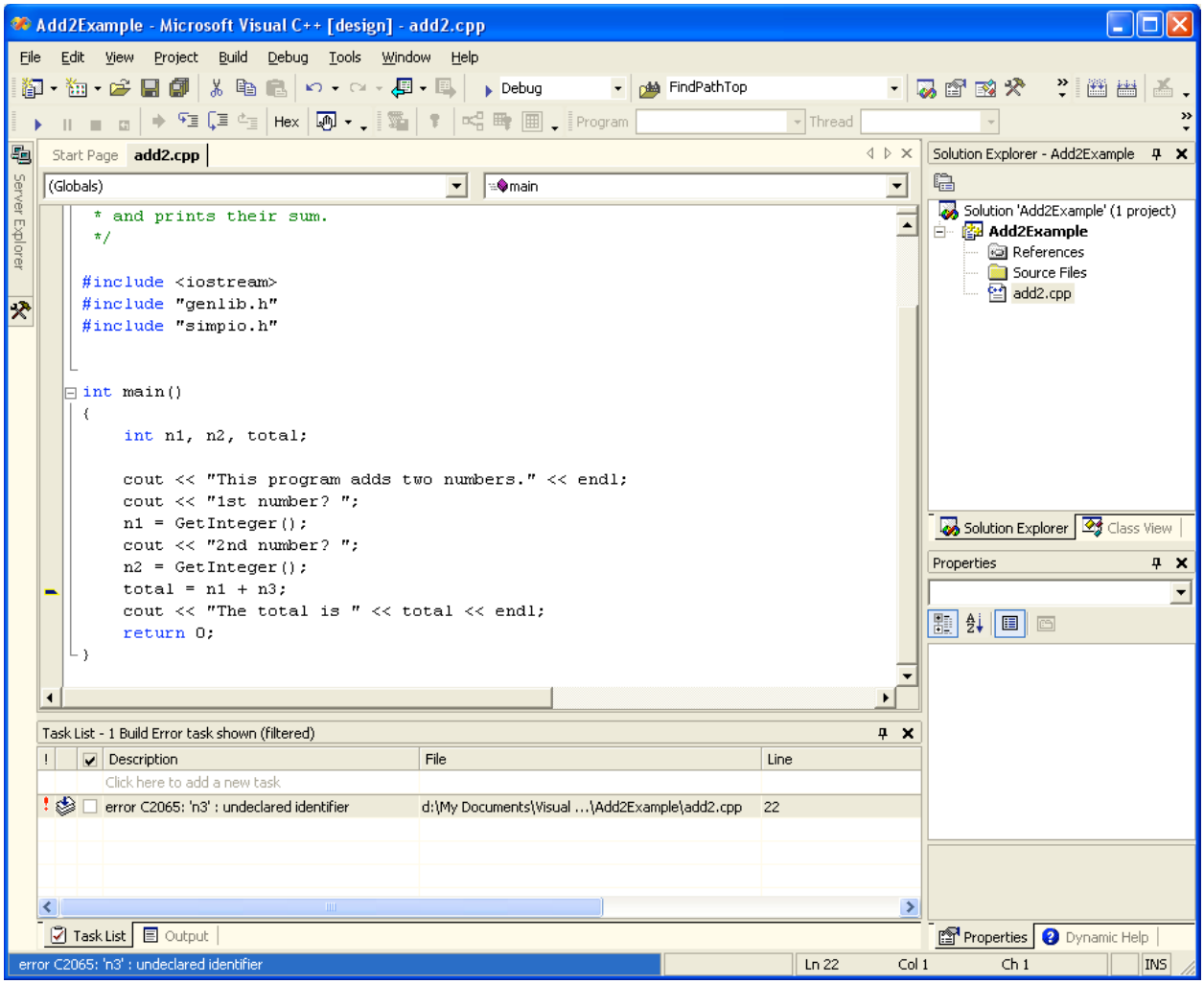
For example, let's say that I mistype line 22 in my `add2.cpp` file and issue the following statement:

```
total = n1 + n3;
```

We don't have a variable named `n3`, so this should not compile. If you go ahead and rebuild the project, the Output window should output something to this effect:

```
------ Rebuild All started: Project: Add2Example, Configuration: Debug Win32 ------

Deleting intermediate files and output files for project 'Add2Example', configuration
'Debug|Win32'.
Compiling...
add2.cpp
d:\My Documents\Visual Studio Projects\Add2Example\add2.cpp(22) : error C2065: 'n3' :
undeclared identifier

Build log was saved at "file://d:\My Documents\Visual Studio
Projects\Add2Example\Debug\BuildLog.htm"
Add2Example - 1 error(s), 0 warning(s)


---------------------- Done ----------------------

    Rebuild All: 0 succeeded, 1 failed, 0 skipped
```

There's seems to be a lot parse here, but we can see that there's one error on line 22 of `add2.cpp`, specifically that `n3` is an undeclared identifier. While this example only has one syntax error, your programs will probably have more (especially as they become larger and more complex…it's just the nature of the beast,) and reading through the Output dialog will become tedious. This is where the Task List window comes in. When you try to build a project and it fails due to syntax errors, Visual Studio will list each error in the Task List window. Each line in the Task List window represents one error, and each entry lists the description of the error, the file in which the error is located, and on what line the error occurs in the file. By double-clicking on the error in the Task List window, Visual Studio will open the corresponding file and jump to the line with the error, placing, as shown in the screenshot below:



Once the error is fixed and the project is rebuilt, it should compile without any errors. If, for some reason, the Output and Task List windows are not displayed by default, you can bring them up by going to the **View** menu, expanding the **Other Windows** option, and choosing either **Output** or **Task List** (you can also hit Ctrl+Alt+O or Ctrl+Alt+K to bring up the respective windows.)

Debugging logic errors

Eventually, you will get the last syntax errors out of your program, and you'll be up and running. This is when the fun begins. The most serious errors in a program are the ones the compiler doesn't catch—the errors that occur when your program is doing exactly what you told it to do, only that what you told it to do wasn't at all what you really wanted it to do. These are the second class of errors, known as *logic errors* or, more commonly, *bugs*.

Since this is a larger topic, we have another handout devoted to debugging logic errors in Visual Studio that you should read and become familiar with.

Source: http://see.stanford.edu/materials/icspacs106b/H05P-UsingVisualStudio.pdf