CS106 - Using Microsoft Visual C++

This handout was written by Matt Ginzton and Robert Plummer.

Using a PC

If you wish to do your assignments on a PC, you will need either Windows 95/98/2000 or NT 4.0, and Microsoft Visual C / C++ 6.0. You should be familiar with the Windows operating system and PCs in general, and to be willing and able to figure out the VC++ environment. The emphasis in class will be on general programming concepts, not on the workings of Visual C++, and the code you write will be identical to code written by students using a Macintosh. (Note: even though the compiler is normally called VC++, you will be coding in C.)

What Have I Got To Lose?

While being able to run the same code on PC's and Macs might seem to be an end-all to the platform struggle, there is one downside to such an approach, and anyone interested in using PCs in this course should be fully aware of it.

Many of our section leaders are more familiar with Macs than PCs, and some, since they come from Stanford's Mac-intensive environment, have no PC experience at all. This means that for PC-specific issues, there are fewer people who can help you. The section leaders working at Tresidder Lair come from all three 106's: A, B and X, and all are qualified to answer questions about these courses. These helper hours are very useful, but for the reason stated above, you may not be able to get help on PC-specific questions. Don't let this discourage you from using helper hours as a resource for questions about your assignment, C syntax, etc.

Getting help

To compensate for the slight lack of staff experience with PCs, we provide a special PC email "help line". You can get prompt, accurate answers to your PC-specific questions by sending them to cs106-pchelp@cs . Please note that this service is for questions that are truly PC specific. If you have a question that is platform-independent (e.g., how to tackle an assignment, or a general C question that would pertain to any compiler), then you should send it to your section leader, TA, or instructor. Only questions about VC++ itself or the PC version of the libraries should be sent to the help line.

You will also find much useful PC information at the following web site: http://cs106-pclibs.stanford.edu . If you are using a PC, you should definitely check it out.

What's the bottom line?

Using a PC means that you have a smaller support net, but on the other hand, you get to stay with the machine you like, you probably get faster compiling and execution of your programs, and you can switch to the Mac without having to rewrite any of your code. It's your choice.

Getting started

If you've read the above and you're still interested in using a PC, then read on. As we noted before, to use the PC version of the libraries, you must:

- Be familiar with your PC and with the Windows.
- Own a copy of Microsoft Visual C++ version 6, Learning Edition or Professional Edition (other 32-bit Windows compilers might possibly work, but if they don't there will be *no one* to help you).

Your first job will be to install the compiler according to the instructions that come with it. The development environment is called Visual Studio (aka Developer Studio). You will probably want to place a shortcut to it on your desktop. If you can't find it, search for the file msdev.exe.

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. If you have questions about this process, please email cs106-pchelp@cs.stanford.edu.

You can obtain the CS Libraries from *http://cs106-pclibs.stanford.edu/pclibs.zip* This file is a ZIP archive containing the library itself (CSLib.lib) and the .h files from the include directory. Here are the steps to follow:

1. Use your web browser to download PCLibs.zip 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/*. This will leave you with a file called CS106Lib.lib and a directory called CS106inc (which contains a bunch of .h files).
3. Place the library file, CS106Lib.lib, in your Visual Studio "Lib" directory. If you followed the default VS installation, this will be at: C:\Program Files\Microsoft Visual Studio\VC98\Lib
4. Place the assignment starter file, CS106Wiz.awx, in your Visual Studio "Template" directory. If you followed the default VS installation, this will be at: C:\Program Files\Microsoft Visual Studio\Common\MSDev98\Template
5. Move the CS106inc directory somewhere that makes sense to you.

Now it's time to tell Visual C where to find the .h files:

1. Start Developer Studio and choose Options from the Tools menu.
2. Select the "Directories" tab, make sure that Show Directories For is set to "Include files", and click inside the dotted gray rectangle in the list below.
3. Type the name of the directory containing the .h files (from step 5 above) and press Enter.

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 a project

Sooner or later, you'll have to do an assignment or two. The following instructions should aid in getting started. (For some assignments, your instructor may provide a starter project that simplifies a few of these steps. Either way, you should know how to do the following:)

1. Start Developer Studio, and from the File menu choose New
2. Select the Project tab.
3. The Type of application that you want to create is "CS106 Assignment Wizard"; select this from the list on the left, type a name for the project (avoid using spaces or hyphens).
4. Choose a location. The typical default will be: C:\Program Files\DevStudio\MyProjects\projectname
   where the last part is the project name you entered. This location is a good choice, but it is up to you. When everything is set, press OK.
5. 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 (.c). These will come in the form of a ZIP file, and at this point you should leave Developer Studio (you needn't shut the program down) and unzip the files into the directory for your project.
6. Now return to Developer Studio, because it is time to add some files to your project. Pull down the Project menu, select Add to Project, then select Files. 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 following extensions: .C, .LIB, and .RC. Note that you never add .H files to the project.
7. You will probably need to create some new .C files of your own. Select New from the File menu and click the Files tab of the dialog. Select Text File as the type, type the file name (such as mycode.c), be sure Add To Project is checked, and click OK. The file will be added and you will be taken to an empty editor window, ready to type in your code. (Note: you will repeat this step for each module (.C file) in your project.)
8. When you are finally ready to compile your code, one thing that you need to realize is that VC++ creates two "configurations" of your project, known as Debug and Release. You will almost always want to deal with the Debug configuration, so that you can use the debugger to test and debug your program. To be sure that you are working with the Debug configuration, select Set Active Configuration from the Build menu, then choose Debug.
9. To compile and link all the code in your project, select Rebuild All from the Build menu, or click the Rebuild icon on the toolbar (the menu shows what it looks like). If you select Build projectname.exe from the Build menu (or press F7), just the files you have changed since the last build will be recompiled.
10. To run your program, select Execute from the Build menu, or click its icon on the toolbar, or press Ctrl+F5. However, unless you're already sure the program works correctly, we recommend running it with the debugger. To use 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. Then pull down the Build menu, select Start Debug, and select Go from the submenu, or press F5.

Bug Hunting

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. As you know, 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 *logic errors* or, more commonly, *bugs*.

Debugging logic errors is a skill that comes only with practice. Here is the most important rule:

> *In trying to find a program bug, it is far more important to understand what your program is doing than to understand what it isn't doing.*

Most people who come upon a problem in their code go back to the original problem and try to figure out why their program isn't doing what they wanted. Such an approach can, in some cases, be helpful, but it is more often the case that this kind of thinking can make you blind to the real problem. If you make an unwarranted assumption the first time around, you may make it again, and be left in the position that you just can't see why your program isn't doing the right thing.

When you reach this point, it often helps to try a different approach. Your program is doing something. Forget entirely for the moment what it was supposed to be doing, and figure out exactly what is happening. Figuring out what a wayward program is doing tends to be a relatively easy task, mostly because you have that computer right there in front of you. If you can't understand what is happening in your program, go back into the editor and add `printf` calls in those areas that seem to be implicated in the problems. These statements may be as simple as

```
printf("I got to here\n");
```

or they may be used to display the values of variables, as in

```
printf("The value of total is %d\n", total);
```

Run your program again and watch what happens. Often, the text generated by the `printf` calls gives you a significant insight into what is going on. And, once you understand what is going on, it is usually not too hard to figure out how this deviates from the intended purpose of the program.

Another useful debugging tool in C is the "assert" macro. We will provide a separate handout that discusses this tool.

Fortunately for us, MSVC's debugger is very, very good. You can set breakpoints, view the values of variables, step through code one line or one function at a time, and all the normal debugging features, all with a great interface. Try holding the mouse cursor over any variable in your source code for a few seconds -- up pops the value. Read the manual and play around. Just remember to set at least your first breakpoint before starting the program.

Declaring "main"

In your textbook, the `main` function is declared as follows:

```
main()
{
```

```
    ...code goes here...
}
```

VC++ prefers it to be done this way:

```
void main(void)
{
    ...code goes here...
}
```

The VC++ way is actually closer to the letter of the law in C. The first `void` states that `main` does not return a value. Leaving it off implies that it returns a value of type `int`, and since we normally don't return anything from `main`, the compiler gives a warning: "main : no return value".

Some compilers don't bother with this, treating `main` as a special case. This is why the book leaves the first `void` off, and why VC++ only gives a warning rather than a compile error.
The `void` inside the parentheses means that main takes no arguments. Many compilers, including VC++, will let you have empty parentheses after main, but we recommend that you go all the way and write `void main(void)`.

Helpline notice

Some of the CS106 staff are not familiar with Windows or the VC++ environment, so if you have questions that are PC-specific (installation questions, VC++ questions, reports of bugs in the libraries, etc.) please refer them to: cs106-pchelp@cs.stanford.edu

To maximize the efficiency of the helpline, please observe the following:

1. Please read the list of frequently asked questions (FAQ), available at http://cs106-pclibs.stanford.edu/pclib-faq/, to see if your question has already been asked and answered.
2. Please refer questions on the starter files provided with some assignments to your TA.
3. Please refer general programming questions and questions that are not specific to PCs to your section leader, TA, or instructor.

Closing Reminder

The code you write is not dependent on whether you are using a PC or a Mac. The interfaces in the .h's are the same -- that's why this handout doesn't mention them. Good luck, and happy coding.