Editing and Compiling in Microsoft Windows

This explains *in detail* the steps in creating a console program using Visual C++ 2005 and later, in *command line mode.*

Choose Your "Working Folder"

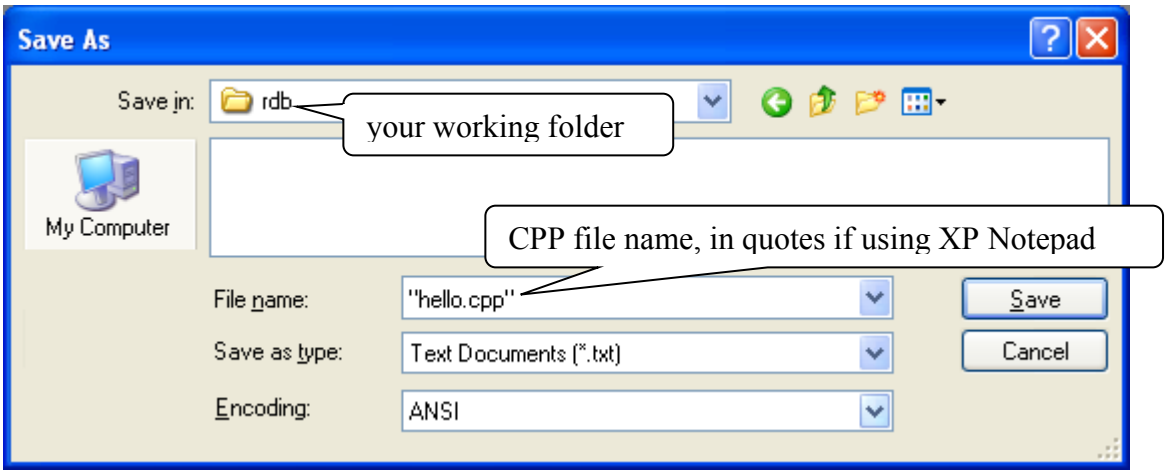
Decide where you want to store your files on your computer – either in a folder of you hard drive or on removable media such as a flash drive or SD card. This is called the "working folder" – also known as the "working directory". Use this in the steps below, where `e:\rdb` is used as the example working folder.

Editing

Use any text editor of your choosing, such as Windows' Notepad, Apple TextEdit, Adobe Brackets, etc, or any other text or code editor that you prefer. Editors are interchangeable, so you can start editing in one, save, and continue editing in another.

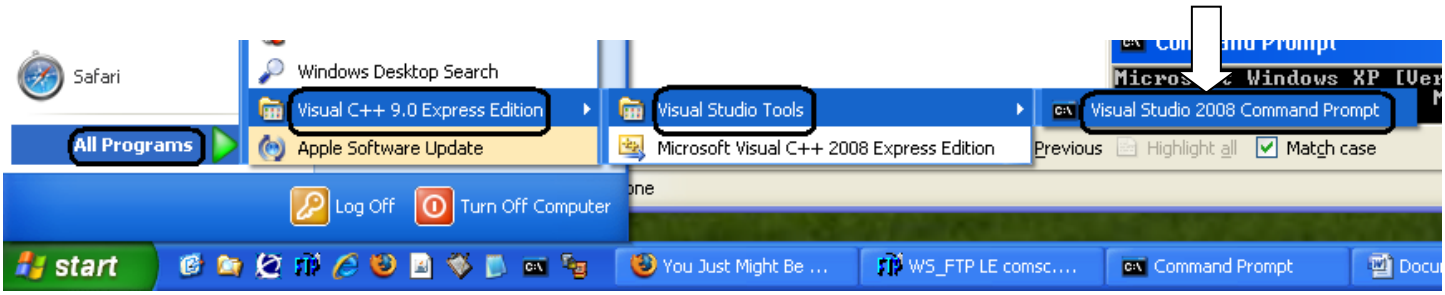
When writing code, use 2 spaces instead of a tab to indent, because the number of spaces that a tab represents varies with the editor or viewer of your code. For consistency and readability, avoid tabs in source code.

When saving CPP files from Windows Notepad, put the filename in quotes to prevent Windows from adding **.txt** to the filename. Save to the "working folder":



Compiling

Access the command line directly from the Start menu:



If you do not see this sequence in your Start menu, you may need to download and install Visual C++. Go to this URL: http://www.microsoft.com/express/vc/ to get a free Microsoft compiler.

To compile, use a command like the following (the command is "see-el", *not* "see-one") to create an EXE file:

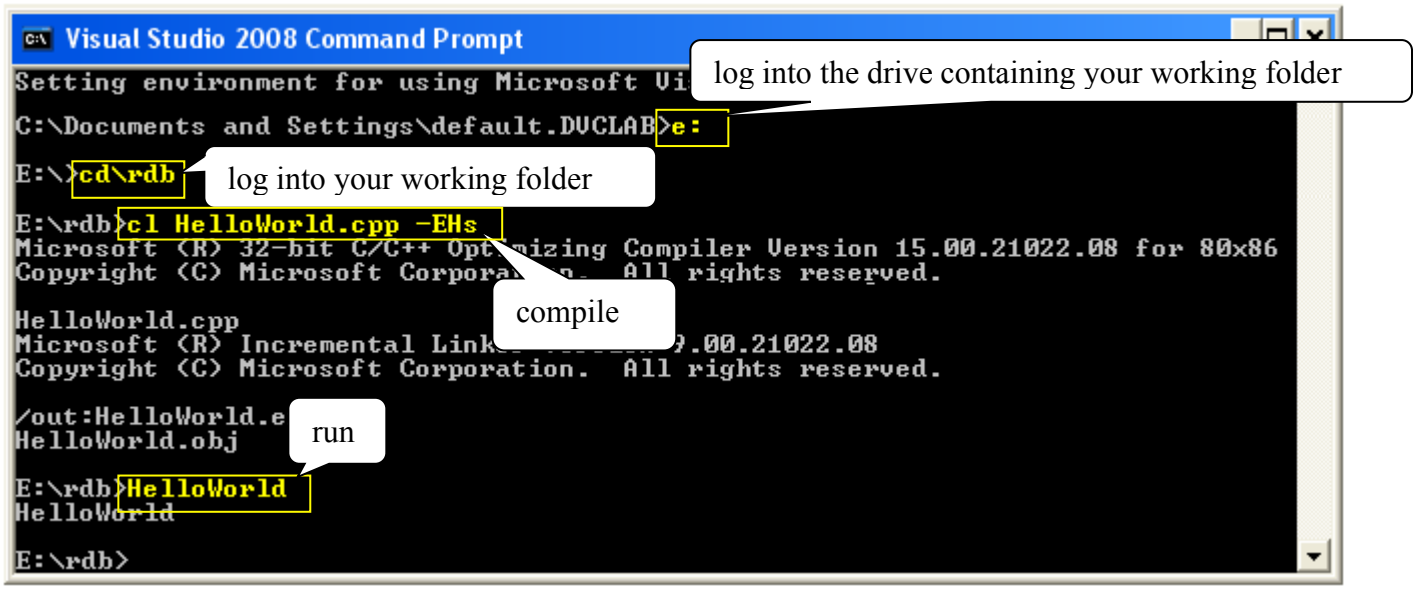
```
cl HelloWorld.cpp -EHs
```

Running

After successfully compiling the CPP file, you can run it as many time as you want.

To run, type the command:

```
HelloWorld
```



In the above example, the file `HelloWorld.exe` was created in the working folder, `e:\rdb`.

Using Windows' Command Line Buffer

So that you do not have to retype the compile and run commands, use the up and down arrow keys to navigate through previously-typed commands. Use the **F7** key to popup a list of commands in the buffer.

The usual sequence is to type the compile and build command, followed by the run command. After that , *up-up* returns to the compile and build command, and *down* goes from there to the run command.

Working with Projects

"Projects" consist of more than one CPP file to make a single EXE. To compile and build projects consisting of more than one CPP, list the CPPs separated by spaces, like this:

```
cl main.cpp Time.cpp -EHs
```

The EXE has the same name as the first (or only) listed CPP, but with the extension "exe". In the example above, the EXE will be `main.exe`. ***To run*** the program, enter the name of the EXE on the command line -- you may leave off the trailing ".exe".

To compile a CPP ***without building***, include the `-c` flag -- this produces an OBJ file with the same name as the listed CPP, but with the extension "obj":

```
cl Time.cpp -c -EHs
```

To build an EXE from already-compiled OBJs, list the OBJs and do not use the `-EHs` flag, like this (creating **main.exe**):

```
cl main.obj Time.obj
```

When working with multiple CPP files in a single project, it is recommended to compile each CPP separately, using the `-c` flag during development. This makes debugging easier. Once the program is working, and you are making small code adjustments, then you should go back to compiling and building all in one command.

About H Files
Do *not* compile H files, and do *not* put their filenames in the compile command. The #include statements in the CPP files is sufficient for their use in projects.