Using Visual C++ .NET

Editing And Compiling In Microsoft Windows

This explains *in detail* the steps in creating a console program using VisualC++ .NET 2003, in either the *IDE mode* or the *command line mode*.

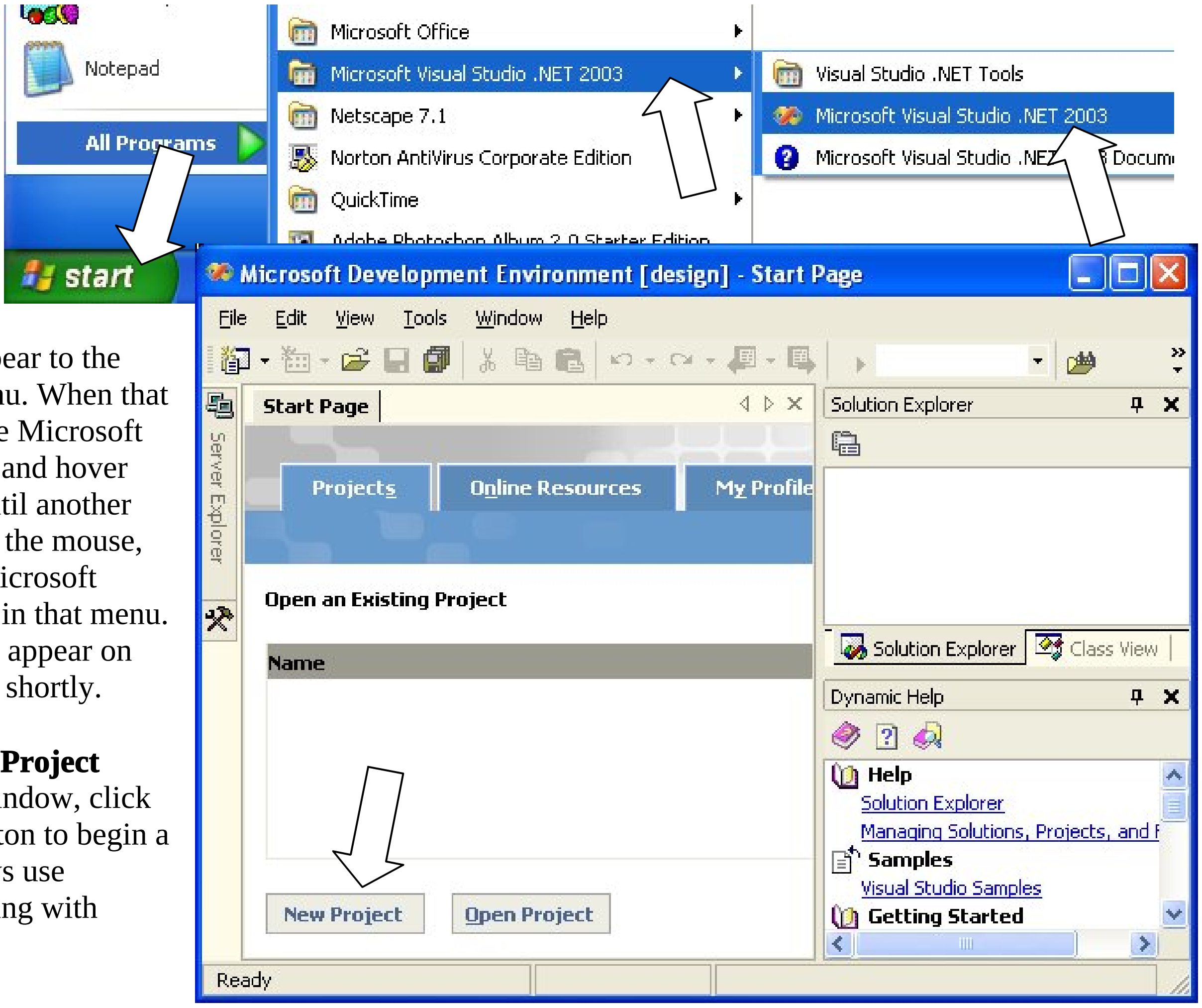
Note that there are some convenient screen control functions in Borland's TurboC++ that are not supported in VisualC++: **randomize** , **random**, **gotoxy**, and **clrscr**, so you will find replacements for these at the end of this writeup.

❑ 1. IDE Mode:

The "Hello, World" program is used here as an example. This may look complicated due to its length, but it's not -- it's just very detailed. These instructions are designed for our lab situation, so that the student CPP files are stored on the floppy drive. Otherwise, they risk losing files stored on the hard drive in the event that a reboot is required. Also, VisualC++ uses nearly 1MB of "workspace" files, which are hard to manage on a floppy. So these instructions tell how to put the workspace files on the hard drive, and the source files on the floppy. The same instructions apply for other removable media, such as USB drives, except for the letter designation of the drive.

Start Visual C++ .NET

The PCs in DVC's computer labs have Microsoft VisualC++ .NET installed on them. Using the mouse, click the Windows Start button (located in the lower left corner of the screen display). Then hover the mouse over Programs from the menu that pops up, and wait for another menu to appear to the right of the first menu. When that menu appears, locate Microsoft Visual Studio .NET and hover the mouse over it until another menu appears. With the mouse, point to and click Microsoft Visual Studio .NET in that menu. The program should appear on the computer screen shortly.

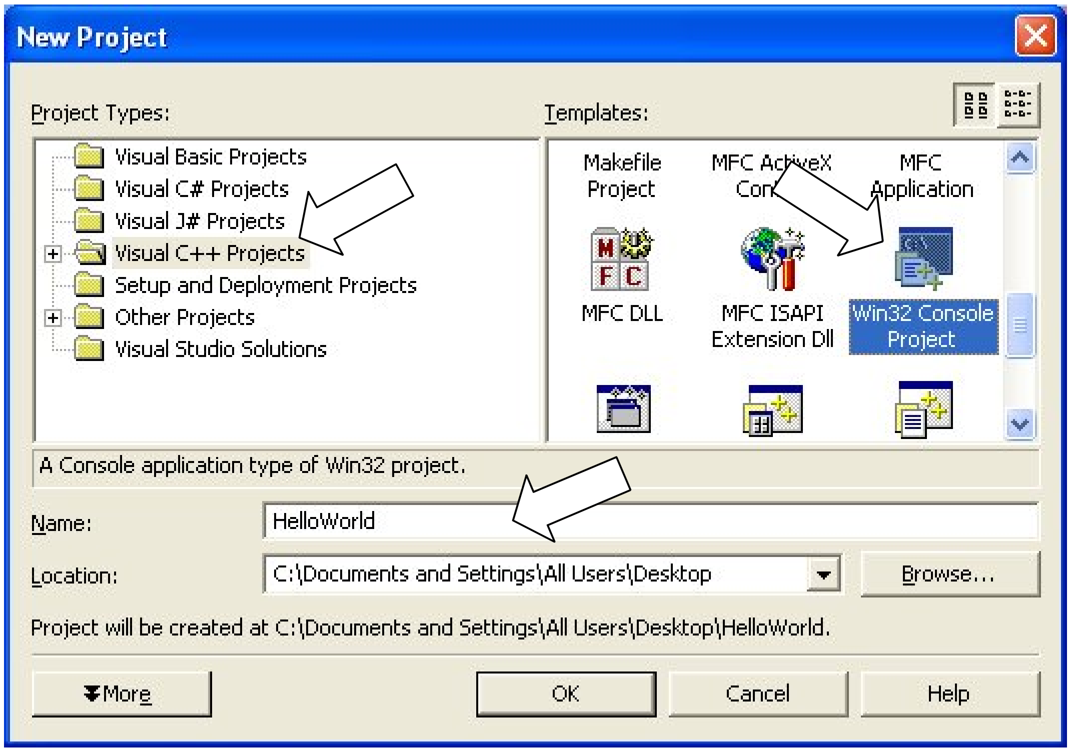

Notepad
All Programs
start
Microsoft Office
Microsoft Visual Studio .NET 2003
Netscape 7.1
Norton AntiVirus Corporate Edition
QuickTime
Adobe Photoshop Album 2.0 Starter Edition
Visual Studio .NET Tools
Microsoft Visual Studio .NET 2003
Microsoft Visual Studio .NET 2003 Documentation
Microsoft Development Environment [design] - Start Page
File Edit View Tools Window Help
Server Explorer
Start Page
Projects
Online Resources
My Profile
Open an Existing Project
Name
New Project
Open Project
Solution Explorer
Class View
Dynamic Help
Help
Solution Explorer
Managing Solutions, Projects, and Files
Samples
Visual Studio Samples
Getting Started
Ready

Create a New C++ Project

In the VisualC++ window, click the New Project button to begin a *new project*. (Always use projects when working with Visual C++.)

In the popup "New Project" window that appears, select Visual C++ Projects, and Win32 Console Project.

Enter the "Location:" **c:\Windows\Desktop**, or in XP/2k, **C:\Documents and Settings\All Users\Desktop**. Enter the project "Name:" as **HelloWorld**. (Note that there are *no spaces* in the word **HelloWorld** -- it is two words run together without a space.) Click the **OK** button to proceed to the "application wizard".



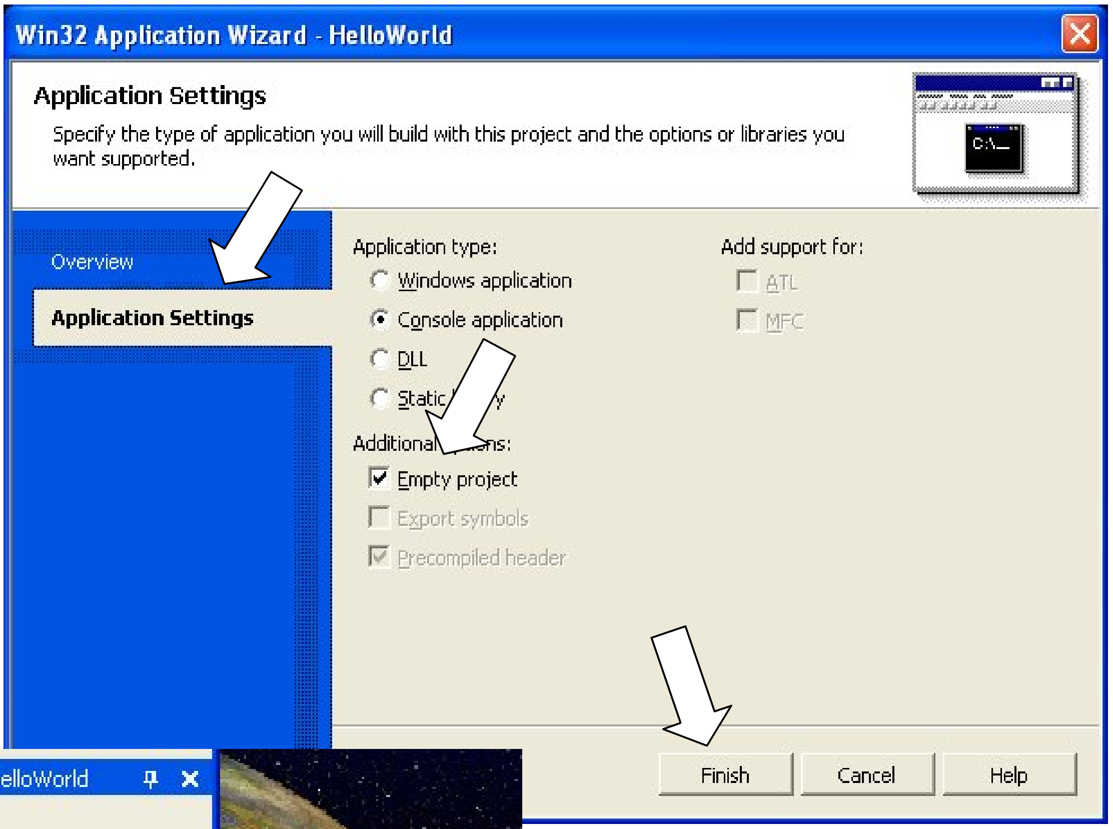
A Console application type of Win32 project.

Name: HelloWorld

Location: C:\Documents and Settings\All Users\Desktop

Project will be created at C:\Documents and Settings\All Users\Desktop\HelloWorld.

In the application wizard window, click the **Application Settings** link at the left. Check the "Empty project" option, and click the Finish button.



Application Settings

Specify the type of application you will build with this project and the options or libraries you want supported.

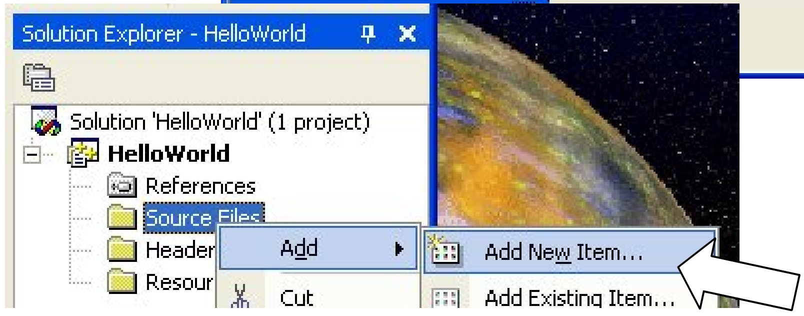
Application type:
- Windows application
- [✓ Console application]
- DLL
- Static library

Additional options:
- [✓] Empty project
- [ ] Export symbols
- [✓] Precompiled header

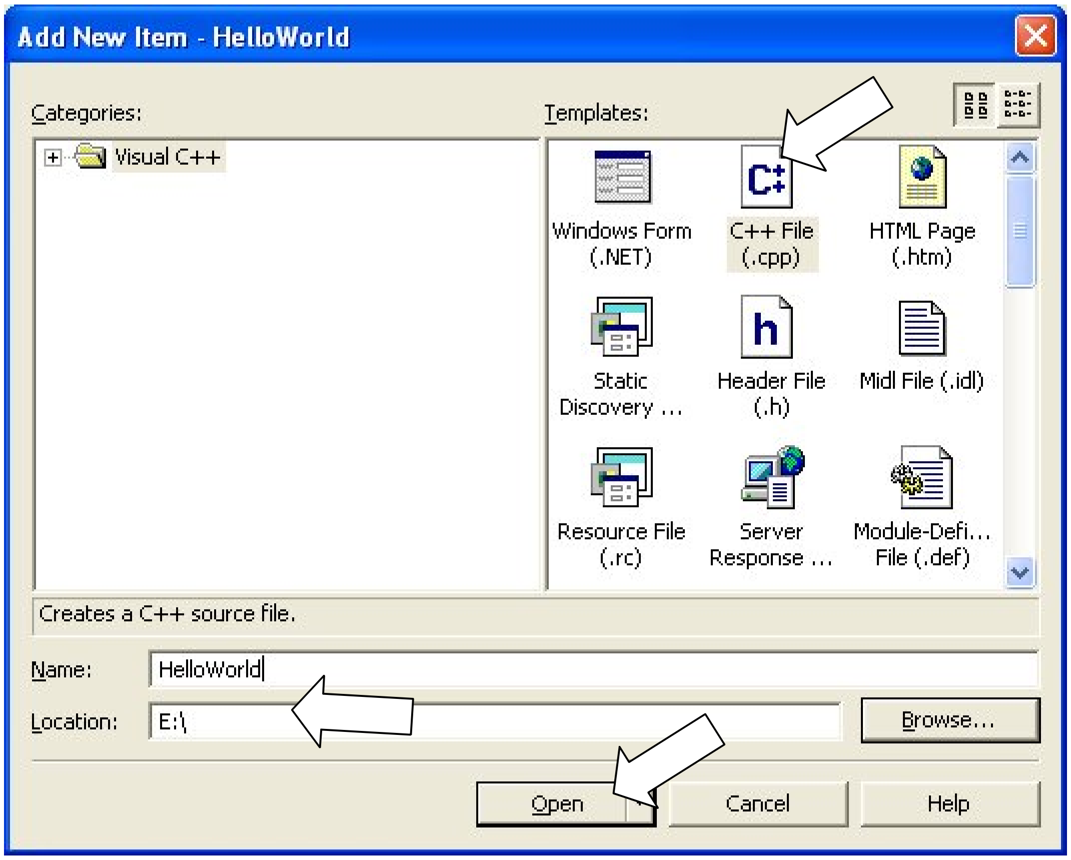
Add support for:
- [ ] ATL
- [ ] MFC

The "Solution Explorer" panel should not appear on the Visual C++ display. Right-click over "Source Files", and select Add->Add New Item from the menu.

(For adding a CPP that already exists, use Add Existing Item instead.)



Select the floppy drive (or other removable media, such as a USB drive E: as shown in the example.) Choose the C++ File template, and type the name **HelloWorld**. Click the Open button to complete the process.



Creates a C++ source file.

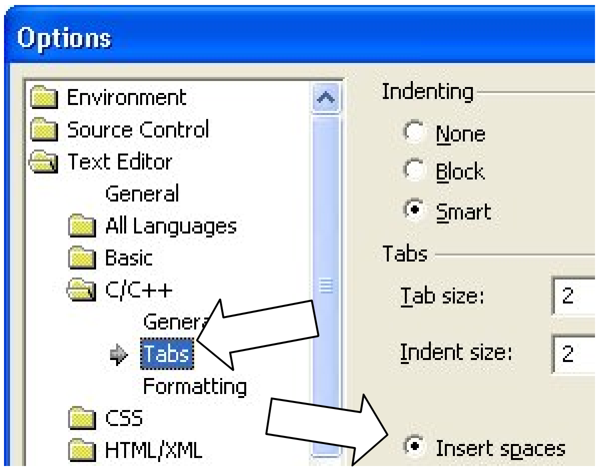
Name: HelloWorld

Location: E:\

(Note: this places the CPP file on your floppy diskette, making it easy to save your work, and protecting your files from loss due to unexpected reboot of the computer.)

Configure the Editor

It is better to use spaces (as opposed to tabs) to ident and align code, so that your code looks the same in any editor. To configure the IDE's editor to insert spaces when you press the TAB key, use Tools->Options, and click the Tabs tab. Choose Text Eidtor->C/C++->Tabs. Set the tab size and indent size both to 2 (or another number of your liking), and select "Insert spaces", instead of "Keep tabs". (To convert a file that already has tabs, Use Edit->Select All and Edit->Advanced->Untabify Selection).



Indenting: None [✓ Smart] Block

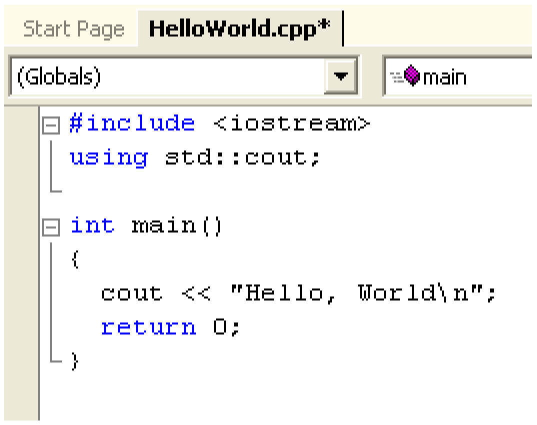
Tab size: 2

Indent size: 2

[✓ Insert spaces]

Type the Code for the Program

Using C++ language syntax, type your program into the window titled **HelloWorld.cpp** . Type the following, with no indenting on the first line of coding. Use 2-space indenting on the first indented line. Skip single lines where indicated. There are three different sets of enclosing symbols used -- the less-than and greater-than symbols around **`iostream`**, the parentheses after **`main`**, and the curly braces where indicated. Be sure to use upper-case and lower-case lettering where shown.



```
#include <iostream>
using std::cout;

int main()
{
  cout << "Hello, World\n";
  return 0;
}
```

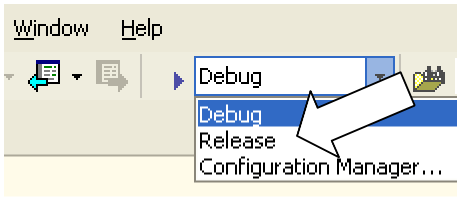
```
#include <iostream>
using std::cout;

int main()
{
  cout << "Hello, World\n";
  return 0;
}
```

Save your work to the computer's floppy drive using the File->Save menu command.

Compile and Run the Program
Press **Ctrl-F7** to *compile*. If not successful, make corrections and repeat until successful. Press **Ctrl--Shift- B** for a "full build", which includes linking, and if successful press **F5** to run. You can also add **`cin.get();`** to suspend your program until a key is pressed -- otherwise, the DOS window disappears right away.

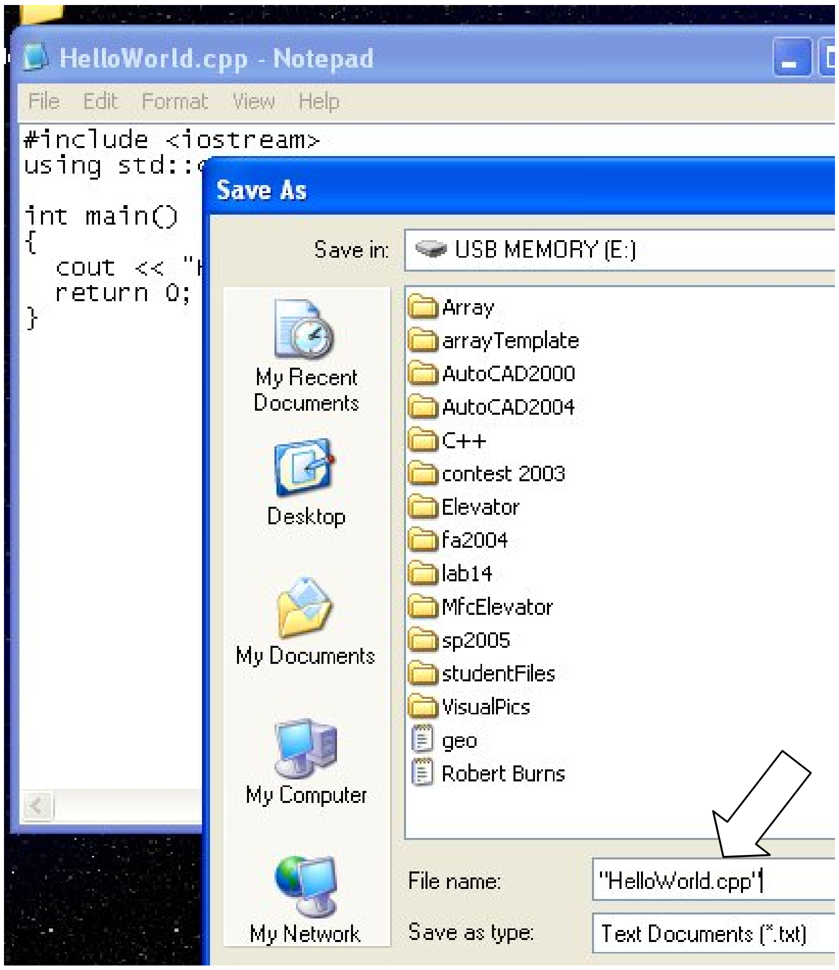
Distributing the Program
The compilation process creates an "executable" file named **HelloWorld.exe**. Your program in its compiled form will run on any Windows 95/98/ME/NT/2000/XP computer, whether or not that computer has VisualC++ or any other C++ compiler.

But before distributing **HelloWorld.exe**, you should create a "Release" version of the program -- because the EXE file is smaller. To do so, use the pulldown menu on the toolbar, and change the selection from **Debug** to **Release**. Then press **Ctrl-Shift-B** to rebuild -- this should result in a new version of your EXE, located in a subfolder named **Release** .



To run the executable, go to a command prompt (Start->All Programs->Accessories->Command Prompt) and log into the drive and folder that contain the EXE file. Enter the name of the file, without the **.exe** filename extension.

Managing Project Files
For backup purposes, all you need for console programs are the source files that you write – that is, the CPP (and H, if any) files. When returning to the lab for another editing session, create a *new* project as you did originally, and add the files to the project from your floppy diskette.



```
#include <iostream>
using std::c
int main()
{
   cout << "H
   return 0;
}
```

File name: "HelloWorld.cpp"

To continue editing a project that remains on your hard drive from a previous editing session, use the Open Project button on the opening screen of Visual C++, instead of the New Project button.

❑ 2. Command Line Mode:
In order to use command line compiling, first copy this file:

C:\Program Files\Microsoft Visual Studio .NET 2003\Vc7\bin\vcvars32.bat

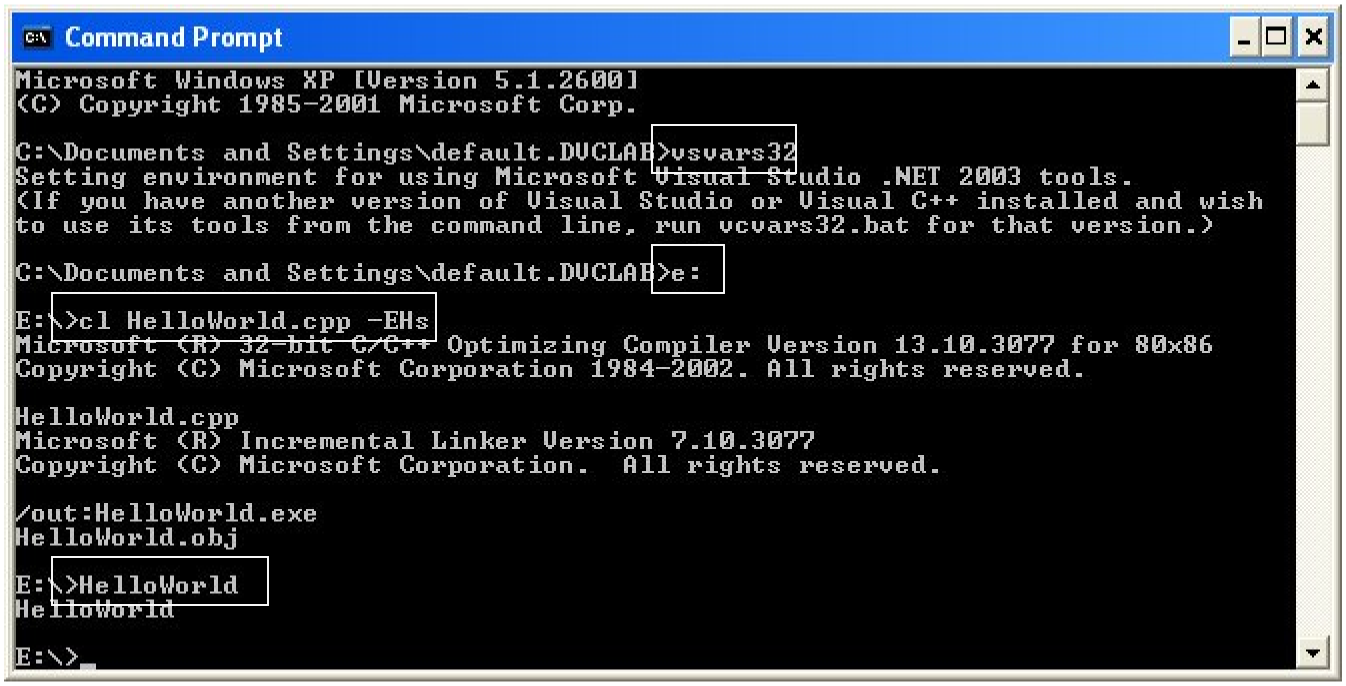
...to your C:\WINDOWS folder. This has already been done on the computer lab PCs, but it is renamed as **vsvars32.bat**, to distinguish it from the Visual C++ 6.0 version of this file.

Use any text editor of your choosing, such as XP's Notepad, JNotePad, or Hesky-Data Pad. Then when you go to a command prompt to compile and build, enter the command **vcvars32** once to prepare for using the command line editor.

*When saving CPP files from Notepad, put the filename in quotes to prevent Windows from adding **.txt** to the filename.*

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
```



```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\default.DVCLAB>vsvars32
Setting environment for using Microsoft Visual Studio .NET 2003 tools.
(If you have another version of Visual Studio or Visual C++ installed and wish
to use its tools from the command line, run vcvars32.bat for that version.)

C:\Documents and Settings\default.DVCLAB>e:

E:\>cl HelloWorld.cpp -EHs
Microsoft (R) 32-bit C/C++ Optimizing Compiler Version 13.10.3077 for 80x86
Copyright (C) Microsoft Corporation 1984-2002. All rights reserved.

HelloWorld.cpp
Microsoft (R) Incremental Linker Version 7.10.3077
Copyright (C) Microsoft Corporation.  All rights reserved.

/out:HelloWorld.exe
HelloWorld.obj

E:\>HelloWorld
HelloWorld

E:\>_
```

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". ***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.

Using XP's 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.

How To: Use randomize And random In Visual C++

Add the following code below the **`include`** statements and above **`int main()`**, so that the **`randomize()`** and **`random()`** functions referenced in the text's code listing work in VisualC++.

```
#include <cstdlib>
#include <ctime>
#define randomize() (srand(time(0)))
#define random(x) (rand() % x)
```

For example, call `random(10)` to get a randomly selected integer between 0 and 9, inclusive. The expression `(1 + random(6))` simulates the rolling of a 6-sided die. The expression `(2 + random(6) + random(6))` simulates the rolling of two 6-sided dice. Every time you run your program you will get the same sequence of random numbers, which is useful for debugging, but not so useful for the final product. Call `randomize()` *once* in your program in order to get a new sequence every time your program runs. Usually this is the first statement in the `main` function.

How To: Use gotoxy And clrscr In Visual C++

TurboC++ has **gotoxy** and **clrscr** functions, which enable you to position the cursor on the screen (so that you canput the contents of the next **cout** statement at a specific location) and to clear the screen. These functions do not exist in VisualC++, but you can write them yourself. Here's a sample:

```
#include <iostream>
using namespace std;

#include <windows.h>
#include <process.h>

void gotoxy(int, int); // prototype
void clrscr(); // prototype

int main()
{
  // write text in 4 corners of the screen
  clrscr(); // function call
  gotoxy(10,10); // function call
  cout << "at 10,10";
  gotoxy(10,20); // function call
  cout << "at 10,20";
  gotoxy(20,10); // function call
  cout << "at 20,10";
  gotoxy(20,20); // function call
  cout << "at 20,20";
  return 0;
}

// function definition -- requires windows.h
void gotoxy(int x, int y)
{
  HANDLE hConsoleOutput;
  COORD dwCursorPosition;

  cout.flush();
  dwCursorPosition.X = x;
  dwCursorPosition.Y = y;
  hConsoleOutput = GetStdHandle(STD_OUTPUT_HANDLE);
  SetConsoleCursorPosition(hConsoleOutput,dwCursorPosition);
}

// function definition -- requires process.h
void clrscr()
{
  system("cls");
}
```