Visual C++ Express Tutorial

Creating a new project/ Opening an existing project

Start Visual 2008 Express Edition by clicking the icon on the desktop or from the Start tab:

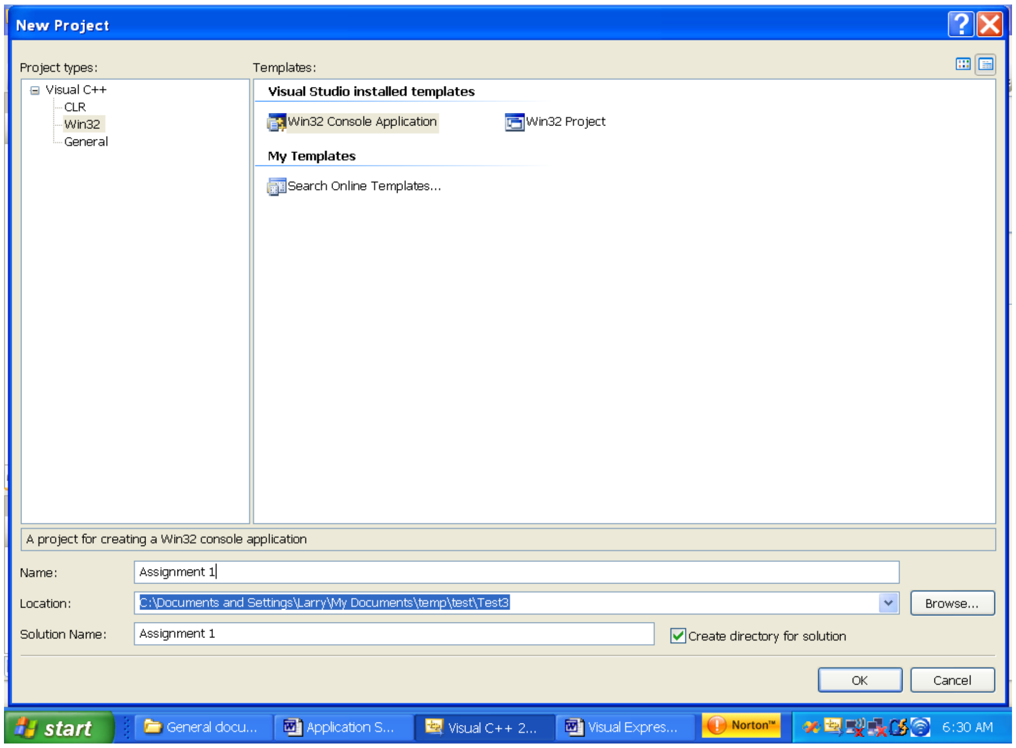
Start → Programs → Microsoft Visual C++ 2008 Express Edition → Microsoft Visual C++ 2008 Express Edition

You may see a tab-page labeled "Start Page" with recent projects and Express News. You may either dismiss this page or select one of the recent projects.

To create a new project:

- Select the menu "File → New → Project…"
- From the menu on the left select "Win32".
- Then from the right panel select **"Win32 Console Application"**.
- In the lower panel select the folder where you want this project to be in the "Location" field. You can click on the button "…" to browse the hard disk. For example, "C:\MyDocuments\EECS211\Program 1"
- In the field "Name" give the project a USEFUL name like Assignment1 or similar.

The screen described above looks like:



A project for creating a Win32 console application

Name: Assignment 1

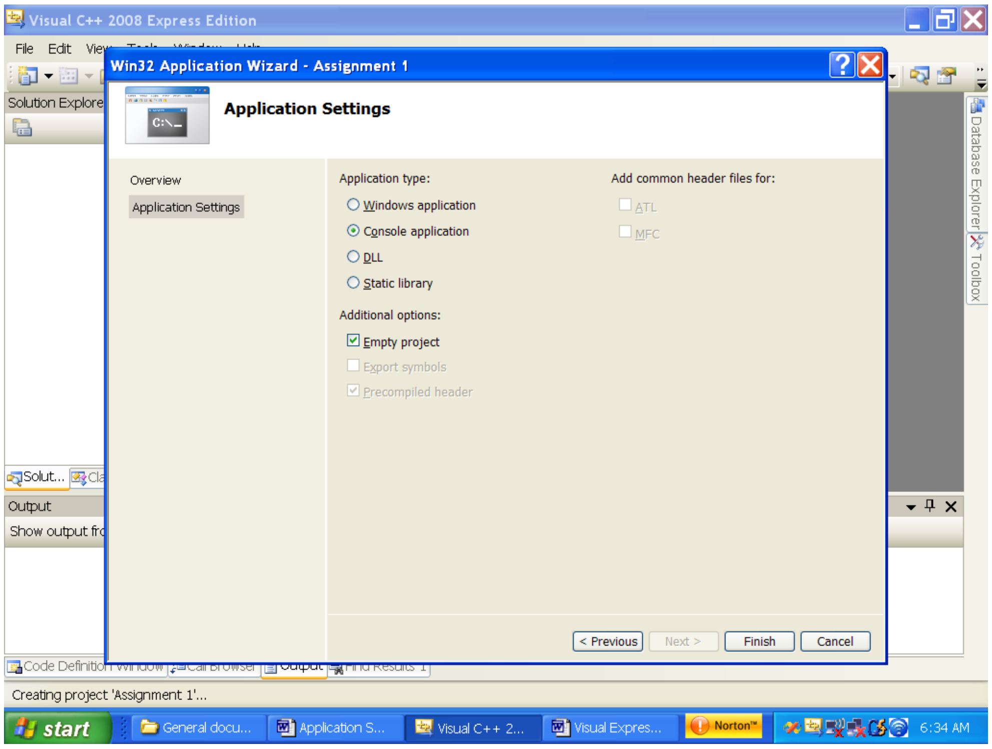
Location: C:\Documents and Settings\Larry\My Documents\temp\test\Test3

Solution Name: Assignment 1

☑ Create directory for solution

- Press OK. You will get a screen that says you have a console application and saying you can click "Finish" at any time to accept defaults. DO NOT CLICK

FINISH YET! Click "Next", and in the following panel select "Empty project", and then click "Finish".



Win32 Application Wizard - Assignment 1

Application Settings

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

Add common header files for:
- ATL
- MFC

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

Creating project 'Assignment 1'...

Now you have an empty project.

If the project you want to work on has already been created (the normal case), you can open it by selecting it from the "Start page" if it appears there or "Open Project/Solution" from the "File" tab:

File -> Open -> Project/Solution…

Note, "Open -> Project/Solution…" is different than "Open -> File". The latter simply opens one file for editing. You must open the entire workspace, which includes all of the files you entered plus all the files C++ created when you created the project and compiled the files in it. If the project you want is one that you worked on recently, you may find it quickly in the "Recent Projects" option under the "File" tab.

Adding files to the project

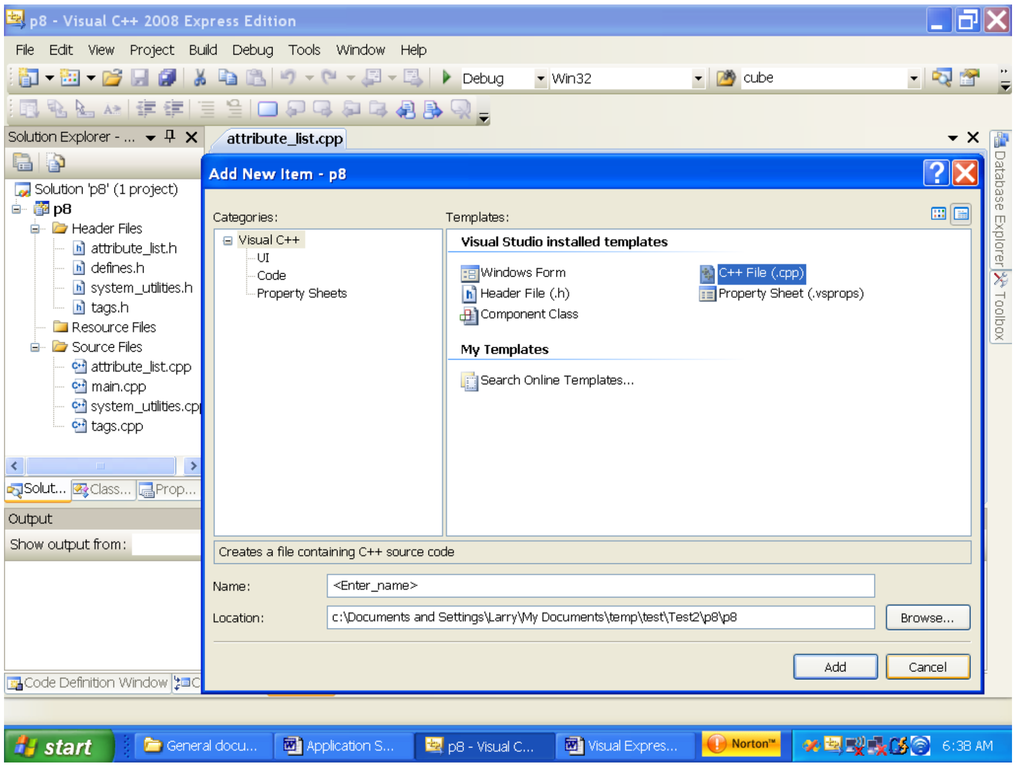
Make sure the "Solution Explorer" tab in the left-hand panel is selected. In this panel you should see a tree-like list with three folders – Header Files, Resource Files, and Source Files.

What is needed next is a file or files where you write your program (the source code and, in later assignments, your header files).

- Highlight either "Header files" or "Source files" and right click.

- In the Add menu you can select "New item …" or "Existing item…" Use the latter if you have already created the file in a different editor and now want to make it part of this project. Select "New item …" to create a new item.
- For a new item select either "Header file" or "C++ file".
- In the lower section of the window give it a USEFUL name and click on "Add".

That dialog box looks as follows:



Creates a file containing C++ source code

Name: <Enter_name>

Location: c:\Documents and Settings\Larry\My Documents\temp\test\Test2\p8\p8

You are ready to write your first program! If you have to create other files for this project then just repeat the above step. That's the easiest way to create and add new, empty files to your project.

You can also add files that you have already created elsewhere or downloaded (for example, from Blackboard). Point the mouse to one of the three folders in the "File view" panel at the left, right click, and select "Add" and then "Existing item …". You can then browse to locate the existing file you want to add.

You can remove a file from the project by right clicking on the file and selecting "Remove".

Typing in the program

The Visual C++ editor has the standard features of text editors plus some special features that help you enter C++ code in a visually effective style. The major additional feature is automatic indentation. As you type statement brackets ('{' and '}') the editor automatically indents or un-indents to make the visual layout of the text match the standard for C++ programs. For example, suppose on one line you type

```
if(x<0) {
```

The editor will indent the following lines until the matching '}' is encountered. For example.

```
if(x<0) {
        cout << "Error: x was negative";
        return;
}
```

This leads to programs that are easier to read because the extent of the block is easy to identify. In addition, you can use this feature to verify that your brackets are balanced. One of the most common errors in typing programs is to forget the closing bracket for a block.

Another useful feature in the C++ editor is the use of color. Tokens that are built in to C++ appear in color, typically blue. These color-coded tokens include the basic data types (int, char, etc.), the main control operators (if, while, do, switch), and others. This feature, again, makes reading the program easier and also allows you to verify that you have spelled these words correctly. Comments are also displayed in color, normally green.

Another useful feature is the "Find in Files…" search operation under the "Edit" tab:

Edit -> Find and replace -> Find in Files

Use this to find all occurrences of a search string in your project. By the end of the quarter you will have a project with many files and hundreds of lines of code. This feature makes finding something in all that text quite easy.

You can shrink compound statements by clicking on the small minus sign (-) on the left margin next to the control keyword. This abbreviates the entire structure on a single line. This is useful in two ways. The original intent was to allow the user to shrink sections of code in a large file that were not of immediate interest to the programmer. For example, a section of code near the beginning of the main function that had already been debugged could be shrunk so that the programmer's focus could be directed to newer sections of code that still needed to be tested. The potential danger in doing this is that some section in the shrunk code might be relevant to the lines of code being tested. A second use for this feature is to check the balance of curly brackets. The user can shrink a structure and see the all the lines the user believed to belong to that structure were, indeed, shrunk. Any shrunk structure can be expanded again by clicking on the plus sign that corresponds to the minus sign used to shrink the section.

How do you compile, run and test your program?

If you're done with your program and want to test if it compiles and runs, use the operations under the "Build" and "Debug" tabs.

To check if a cpp file in your project has syntactic errors, such as typing errors, select the file you want (click on its tab if it is open or highlight it in the file panel on the left) and select “Compile …” from the “Build” tab.

Build -> Compile

The C++ compiler will attempt to translate your source code into a so-called “object” file. If there are syntactical errors, these will be reported in the panel at the bottom of the screen. **NOTE: The absence of syntactic errors does not mean your program will produce the correct results, only that you have not misspelled words or left out a bracket or a variety of other typographical mistakes.** If you double-click on an error, the editor will move to the line in your file where the compiler recognized that a problem exists. Note, this is often one or more lines past where the actual error occurs. For example, if you forget to type “;” at the end of one line, C++ will not recognize that until it encounters the next non-blank, non-comment character.

When there are no more compiler errors and you are ready to test your program, select one of the following from the "Debug" tab:

Debug -> Start Without Debugging – to (re)build your project and run without debugging, i.e., normal execution.

Debug -> Start Debugging – to (re)build your project and run to the first breakpoint in your program files.

Debug -> Step Into – to (re)build your project and have the debugger step into the first statement of the main function and then wait for additional commands.

If any of the files in the project have been changed since the last execution C++ will tell you that the exe file does not exist and asks you if you want to create it. Click OK. C++ then compiles any out-of-date files in your project, links the object files together with other libraries on the computer and executes or steps into the resulting application. Any errors in the compilation of any out-of-date files or in the linking process are reported in the panel at the bottom. If there are no errors, your program should begin to execute. A window will normally pop up, and you enter input through this window and see the results of your output in this window. When your program reaches the end and is ready to terminate, the system will display “Press and key to continue” and wait for you to press a key. This allows you to see the output from your program before the window disappears. (Note, for systems that close the window immediately without waiting, you can add a prompt and input at the end of the main function, just before the return statement. For example, input an integer. You program will wait until you actually enter data.)

Using the Debug Features

An important and very helpful feature of modern integrated systems is the debugger. The debugger lets you execute your program one statement or a group of statements at a time and then look at and possibly modify the results before continuing. You can set a breakpoint, a line where the debugger will stop, by clicking in the gray vertical line at the left side of the cpp file; you can remove the breakpoint by clicking on it. You can have the system execute your program until it reaches a breakpoint. For complicated sections of code, you can have the debugger execute one line at a time so you can watch the program's progress very closely. These execution options are available on the "Debug" tab and the corresponding F keys. When debugging, variables in the current scope of execution will be displayed in the panel at the lower left, and you can view and modify their values. Modify the value of a variable by right clicking on the value field and selecting "Edit value".

When you have finished one debugging session and want to make changes to your code, if your program did not terminate in the debugger execution mode you must select "Stop Debugging" from the "Debug" tab.