Running Visual C++ (Visual Studio 2012)

Revised 8/19/2013

Jim Fawcett

1. *Creating a new empty solution (Figure 1.):*

- Click File\New\Project (ctrl+shift+N) to get a New Project dialog box.
- Select Other Project Types\Visual Studio Solutions in the left pane. Select Blank Solution in the right pane. Add a solution name and browse for a location, then click OK.
- This will bring up Solution Explorer with the new solution.

2. *Creating a new project (Figures 2. and 3.)*

- Right click on the solution in the Solution Explorer and select Add\New Project. This will bring up the Add New Project dialog box.
- Select C++ (or C#) in the left pane and select a project type in the right pane, e.g., Win32 Console Application, or Win32 Project.
- Enter a solution location by typing or browsing (if the path you type does not exist it will be created).
- Enter a name for your project (will reside in a subdirectory in your workspace by default – change by editing the location).
- Click OK. This, for C++ Win32Project, brings up the Application Wizard dialog. Select Application Settings.
- Select Console application and Empty project.
- Click on OK – this creates a directory holding a solution file, intellisense database file, and a user options file
- You can create as many projects as you wish in a given solution. Sometimes you may wish to create two or more projects with the same files but with different properties. Multiple projects let you work on a single package, or the entire program, simply by switching between projects.
- It is a very good idea to make a project for each of your packages. You implement a solution by adding the first packages's project, building, and testing with the package's test stub. Then add another project, and add existing source code from the first project (see item 4., below) if the new package depends on the first package's services. Now you will set this package's properties to compile the second project's test stub, not the first (see Step 8, and Figure 7.). Continue this way until the solution is complete.

3. *Adding a new source code file (Figures 4. and 5.):*

- Right click on project and select Add/Add New Item
- Select Windows Form to add a form, or Header File or C++ File to add a header or implementation file.
- Enter name, and click OK
- Now you are ready to enter code
- Note: adding a new file creates a blank file in the project directory

4. *Adding an existing file to the project:*

- Right click on project and select Add/Add Existing Item
- Browse through directories until you find the file you want.
- Select it and click OK
- Note: adding an existing file to a project does not copy or move any file. It simply adds the path to the selected file into the project settings[1]. Be aware that if you change this added existing file in the IDE editor, it will change the original back in its source folder. This is not the way the C# editor works.
- You can make a copy, using Windows Explorer, and add the copy to the project directory.
- That, of course, means that you now have two versions of the file. Be careful!

5. *Compiling and Linking:*

- Click Build\Build Solution or right click on project or solution and select build or rebuild
  - By default, if you select build it will build all projects in the solution. If you select Project Only/Build, you will build just the selected project.
- Warnings and errors are shown in the bottom Output window. Fix these and rebuild.
- Double click on warnings and errors to go to the problem location in source code

6. *Running the result of a build:*

- Click Debug\Start without Debugging or depress Ctrl F5
- I add the Start without Debugging symbol to my toolbar with Tools\Customize\Debug using drag and drop. A click on that will start an execution

7. *Enter command line parameters before running (Figure 6.):*

- Right click on a project and select Properties to get a tabbed dialog box.
- Click on Debug tab and enter Command arguments for the command line
- Click OK

[1] For C# projects this behavior is very different. The C# project will make a copy of the added file rather than a reference.

8. Enter preprocessor definitions before running (C++ only) (Figure 7.):

- Click Project\Properties to get a tabbed dialog box.
- Expand the C/C++ tab and select Preprocessor
- Add to existing Preprocessor definitions by typing a comma and your definitions
- Click OK
- This is a good way to define the TEST_PACKAGENAME string that determines whether your Package's test stub is compiled or not. Each project, except for the Executive package's has one of these test definitions so that, when you run the project, the package's test stub runs.

9. Debugging:

- Step over: F10 or click on Debug\Step Over
- Step into: F11 or click on Debug\Step Into
- Single step by over, through, out of, or run to cursor, by clicking on buttons on the debug toolbar. You can figure out what each of the icons means by letting the cursor rest for a moment on an icon. A tool tip will pop up with a short description of the resulting action.
- You may have to add the run to cursor option from Tools\Customize\Debug and select and drag the run to cursor icon to the debug part of the menu.
- Add or remove break points by left clicking in the left margin of the text editor at the break line. Step through break points with F5.
- Add watches by clicking in the watch window that appears (by default at the bottom left of the IDE) when you begin stepping. If you don't see watch window, click on Debug\Windows\watch.
- You can click on Tools\Customize to get a tabbed dialog box. Select toolbars and check Debug (you should only need to do this once) and select anything you want on the debug toolbar.

10. Using Help:

- Select any class, function, or namespace names and press F1 for help information. This will appear in a browser window, outside the IDE.
- You can search the online resources by typing in a search query in the search bar at the top of the help window. Be sure to select the filtering you want before you search.
- Help files are hypertext and you can usually find what you want with some link following.

11. Windows

- In the default windows layout, you can toggle between solution explore (view of your solution, its projects, and their files) or class view (view of all the classes in the solution, along with their methods and attributes).
- If you don't see Solution Explorer or Class View, select View\Solution Explorer of View\Class View.
- You can select from a variety of toolbars (Tools\Customize\toolbars), but normally you don't have to do that, as the IDE presents to you toolbars that are appropriate for your selected activities, e.g. editing or debugging.
- You can easily move windows around, and save a particular layout from Tools\Import and Export Settings. You can return to the default IDE window layout by selecting Window\Reset Window Layout.

A note about Solution Layout:

When you create a blank solution, add projects, and add source code items to the projects, each source code file will reside in its own project's folder, which in turn, resides in the solution folder. That means that #include statements have to provide a relative path to header files that need to be included. This works fine for a given project, but when you reuse some of these packages in another project, you will probably place all the files in the same folder, breaking the relative paths. One way to avoid this problem is to move all the files from their project folders to the solution folder. You will have to:

1. Right click on each file and exclude from project.
2. Move the files to the solution folder with Windows Explorer.
3. Then add them back into each project.
4. Now you can change the include paths to be the default current directory.

The result is, all the source code files reside in the solution folder and each project folder contains the project file and the user options file only. You can always create new items in the solution directory by modifying the item path given in the new item dialog. By the way, I don't do this for executive packages. They are seldom reusable, and typically have resource files and other gunk that you don't want in your reusable code area.

Figure 1. – Add Blank Solution

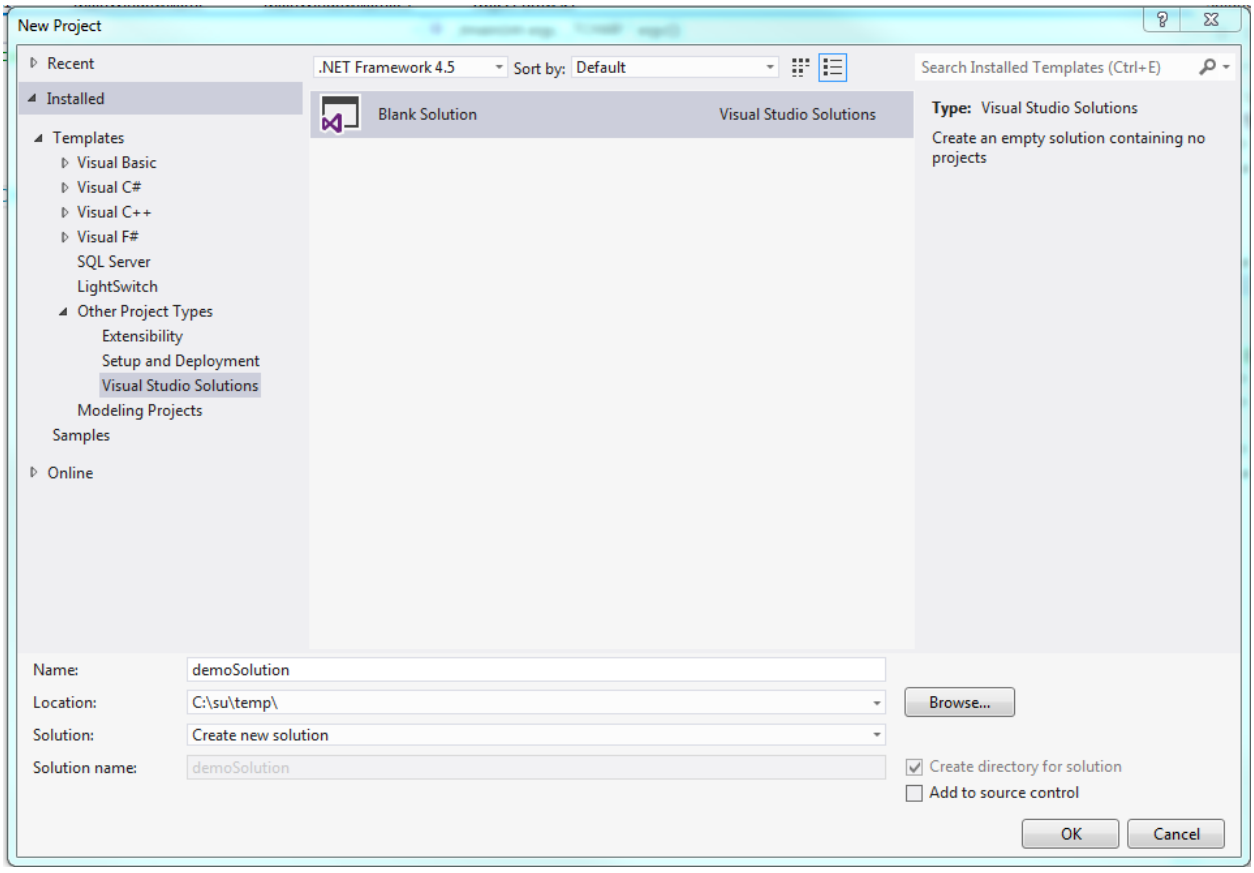

Figure 2. – Add a Win32 Project

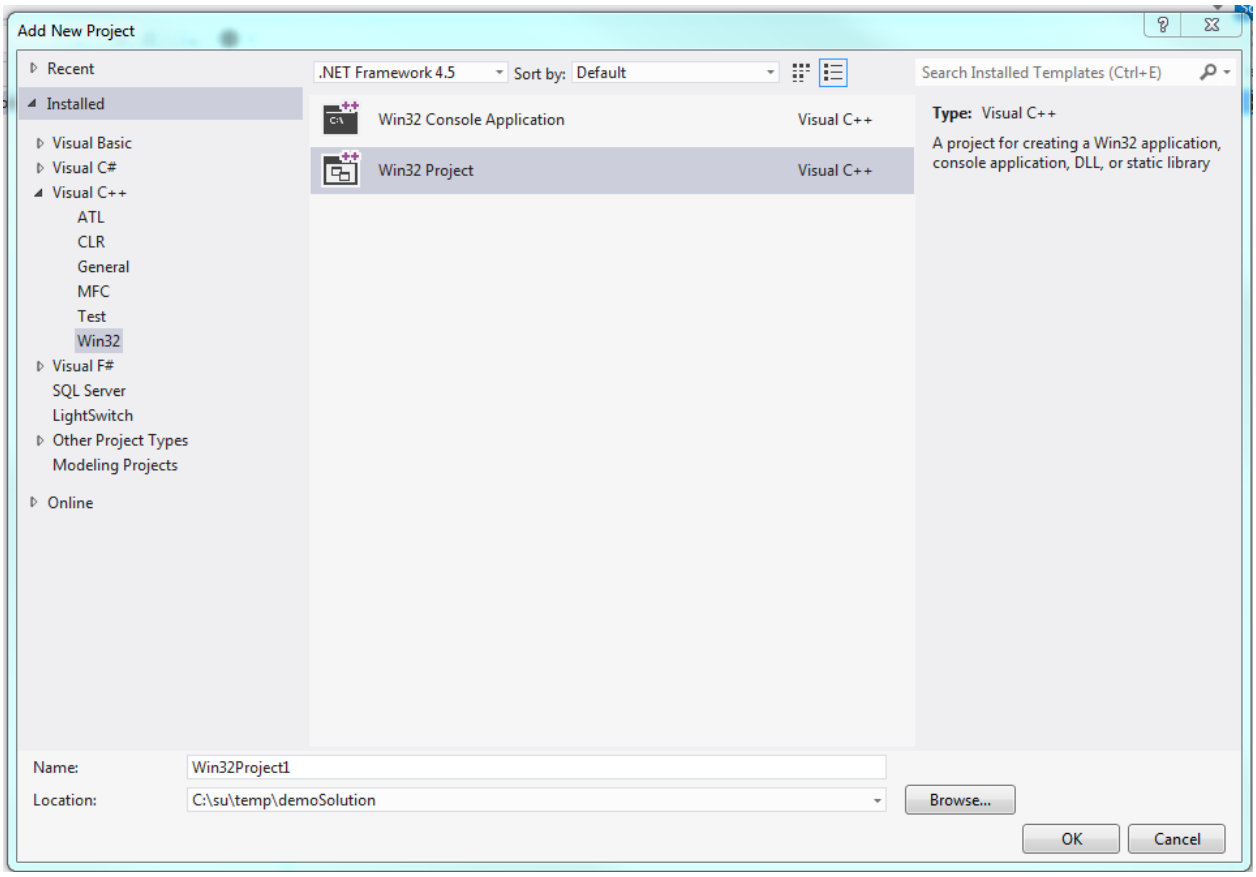

Figure 3. – Select Application Settings

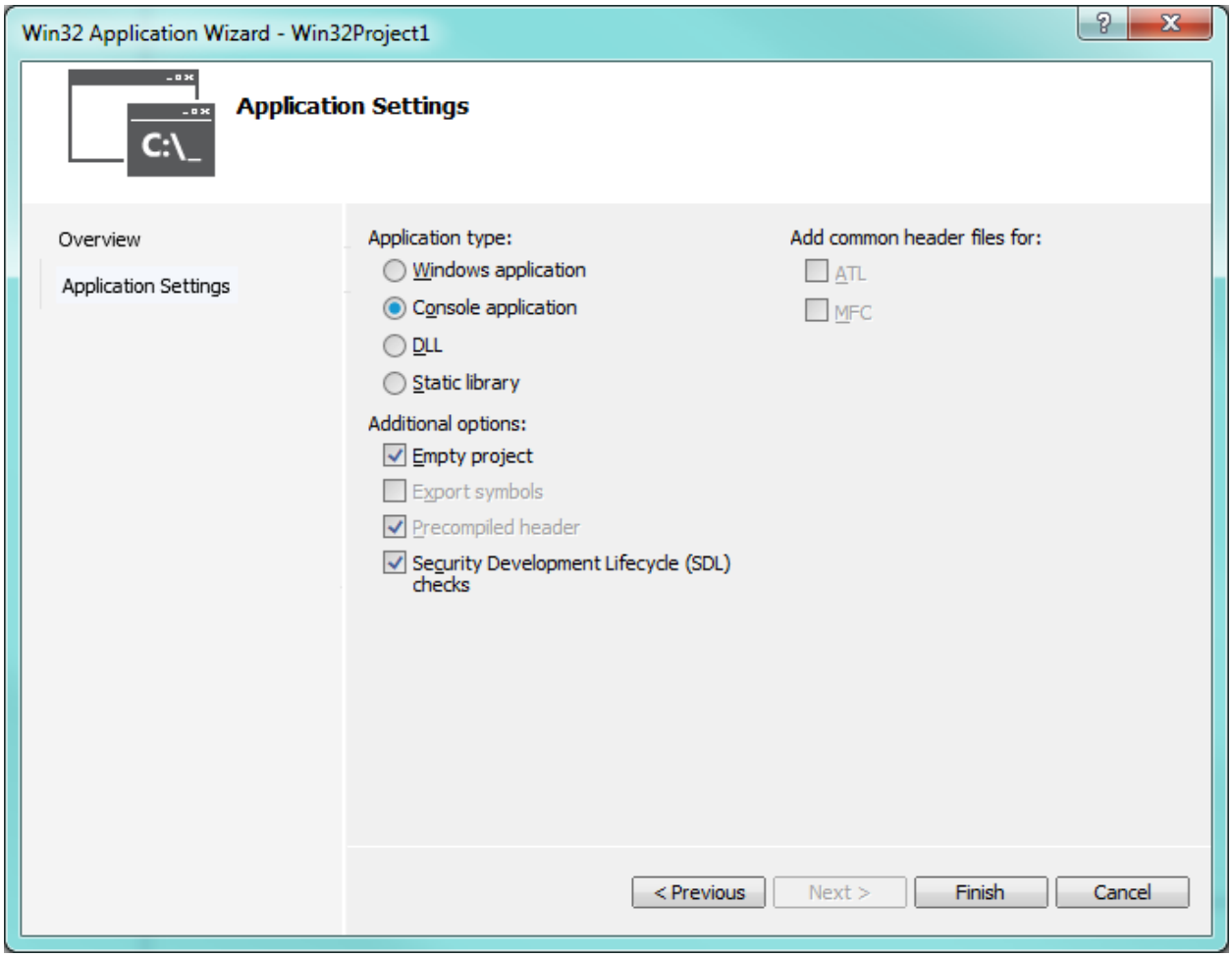

Figure 4. – Add C++ Header File

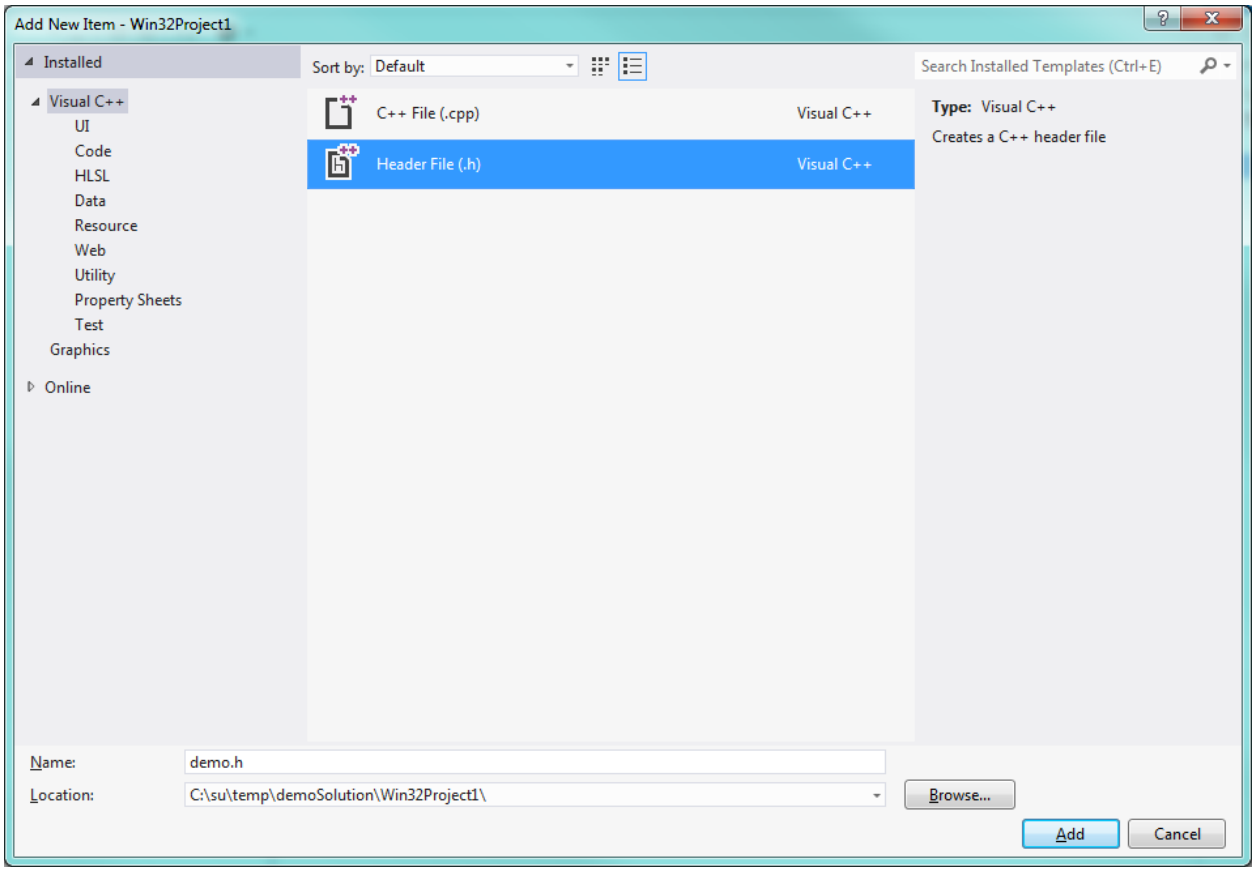

Figure 5. Add C++ Implementation File

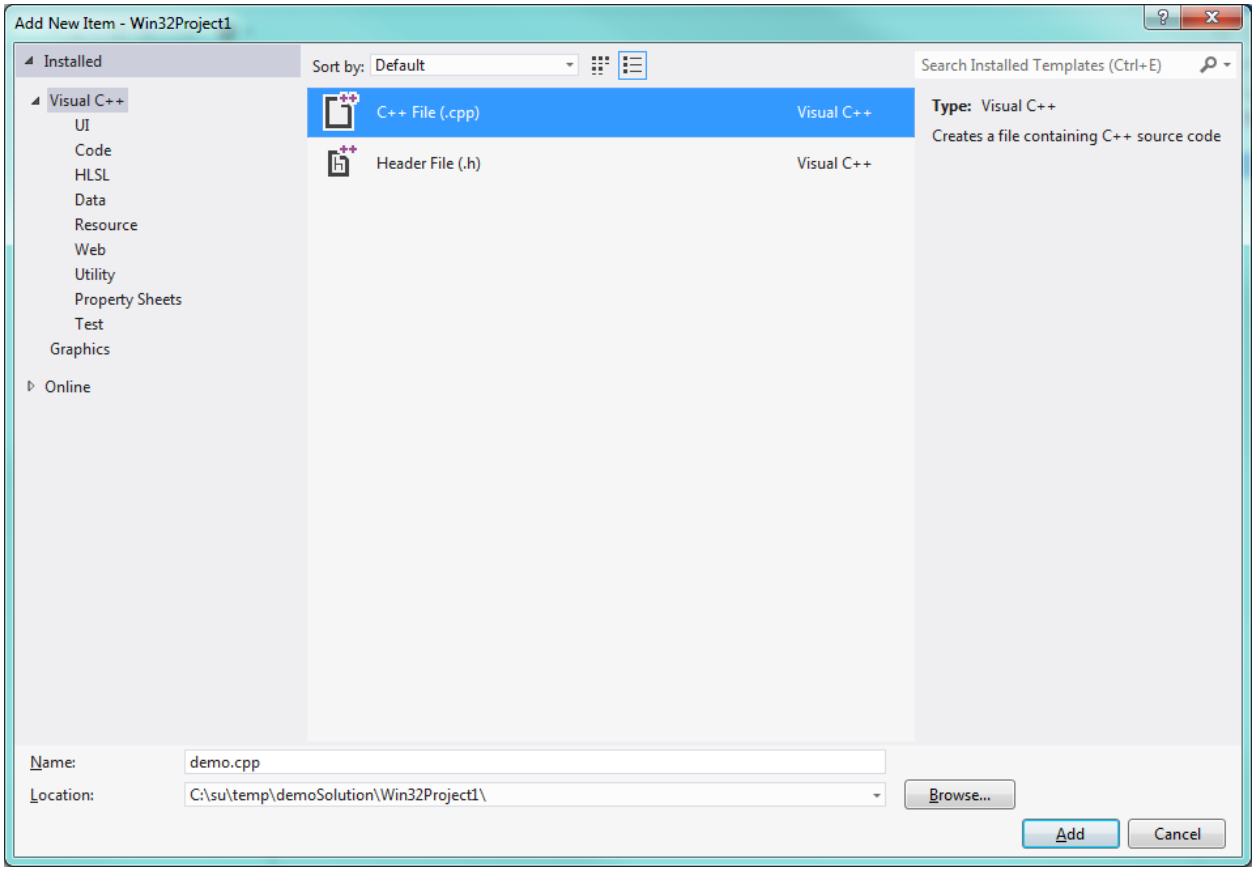

Figure 6. – Command Line Parameters

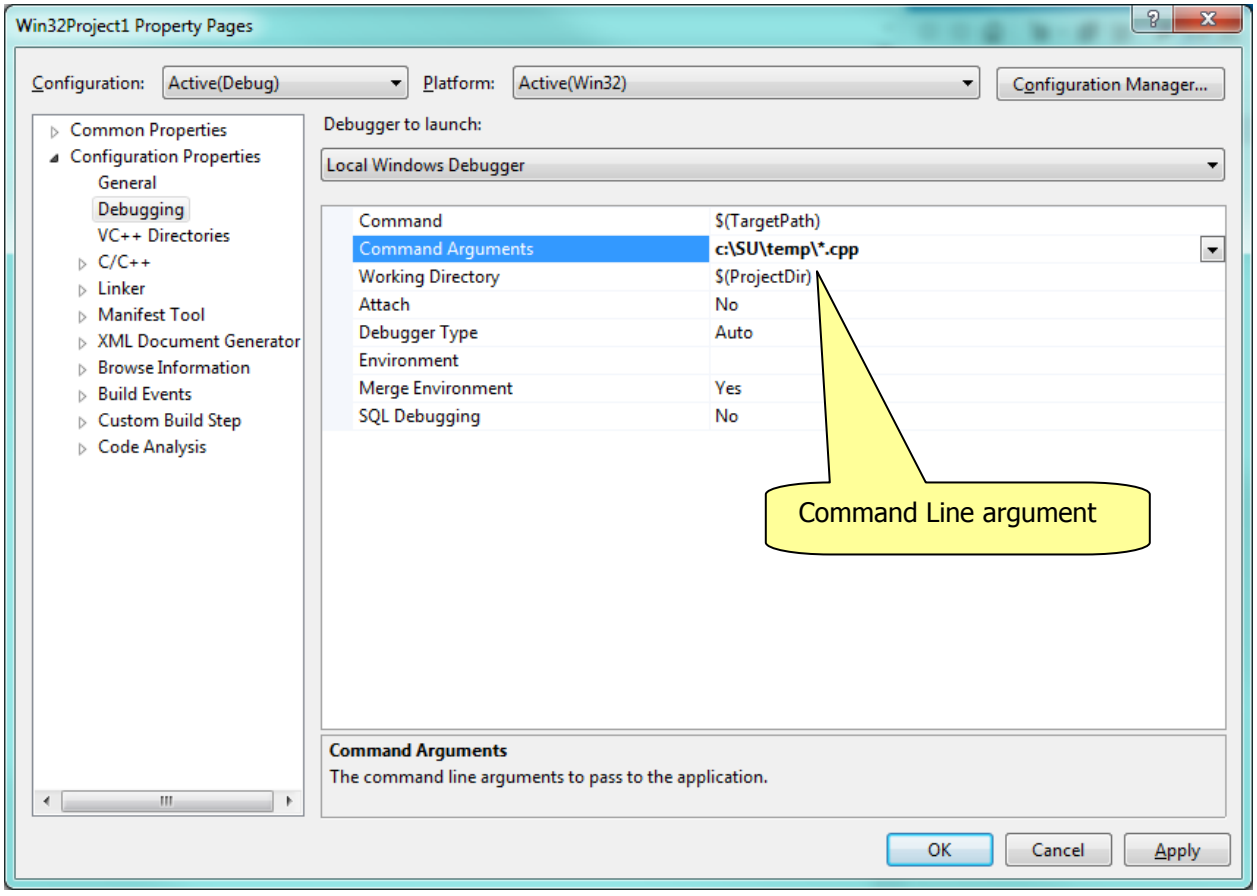

Figure 7. – Add Preprocessor Definitions

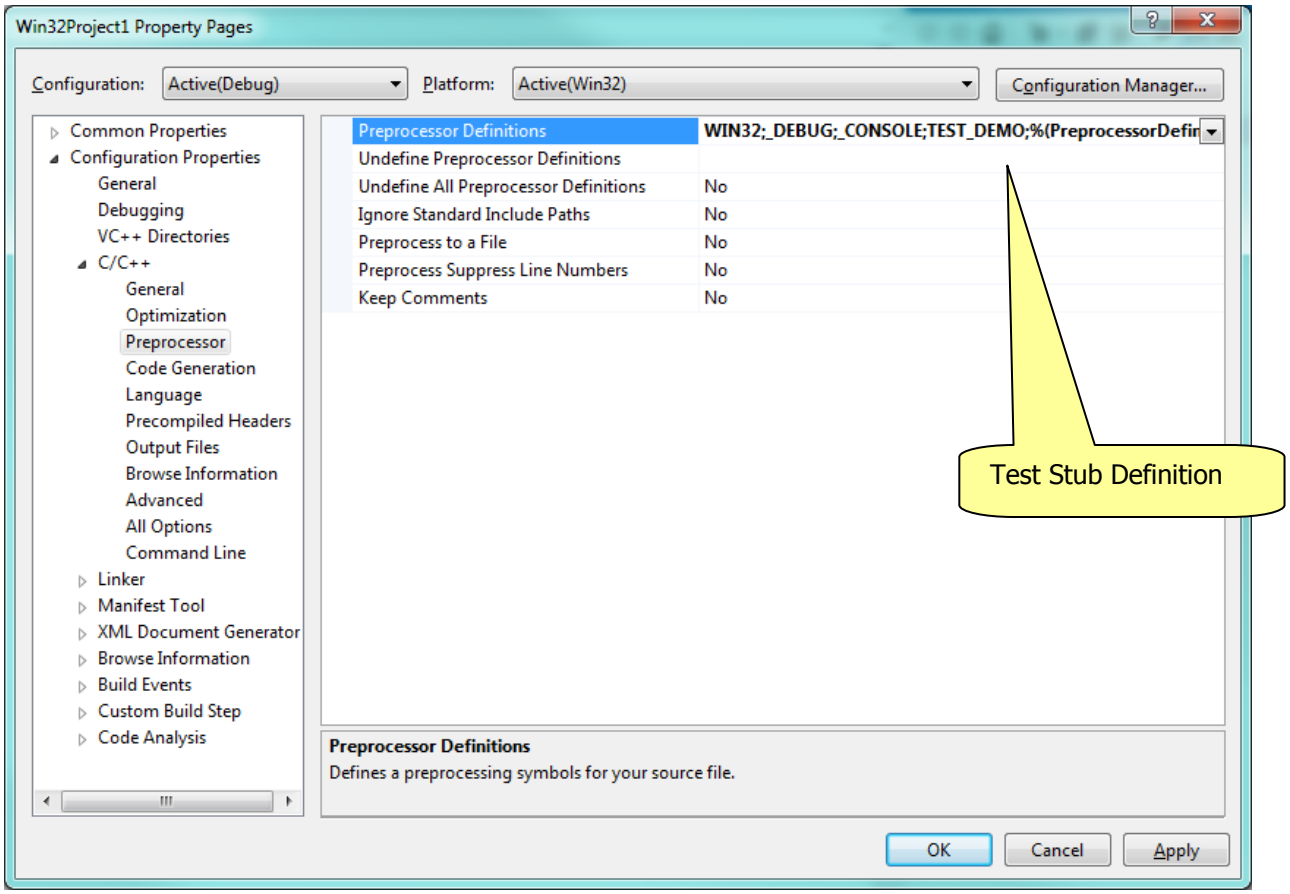

Figure 8. – Add Linker Input to Expand Wild Cards on Command Line

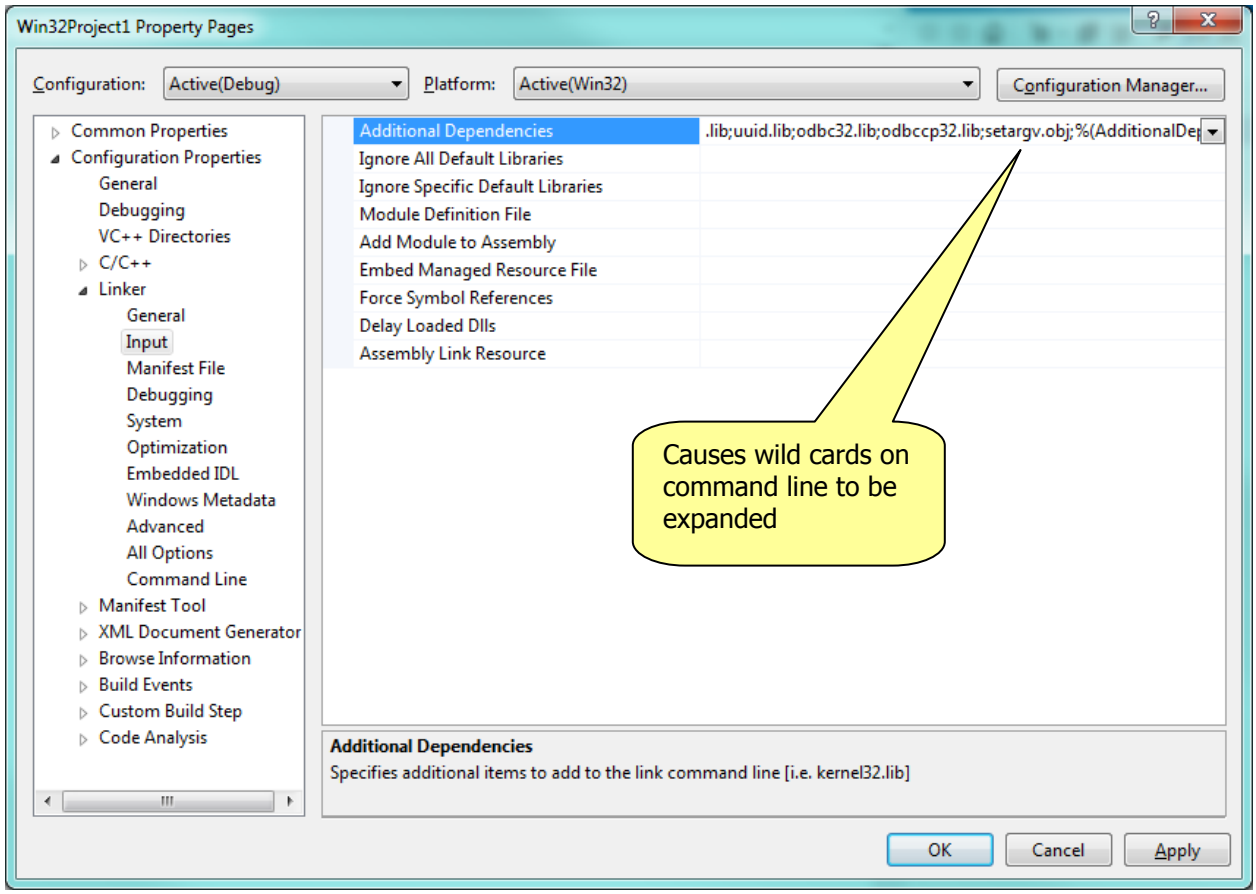

Figure 9. – Runtime Library and Exceptions

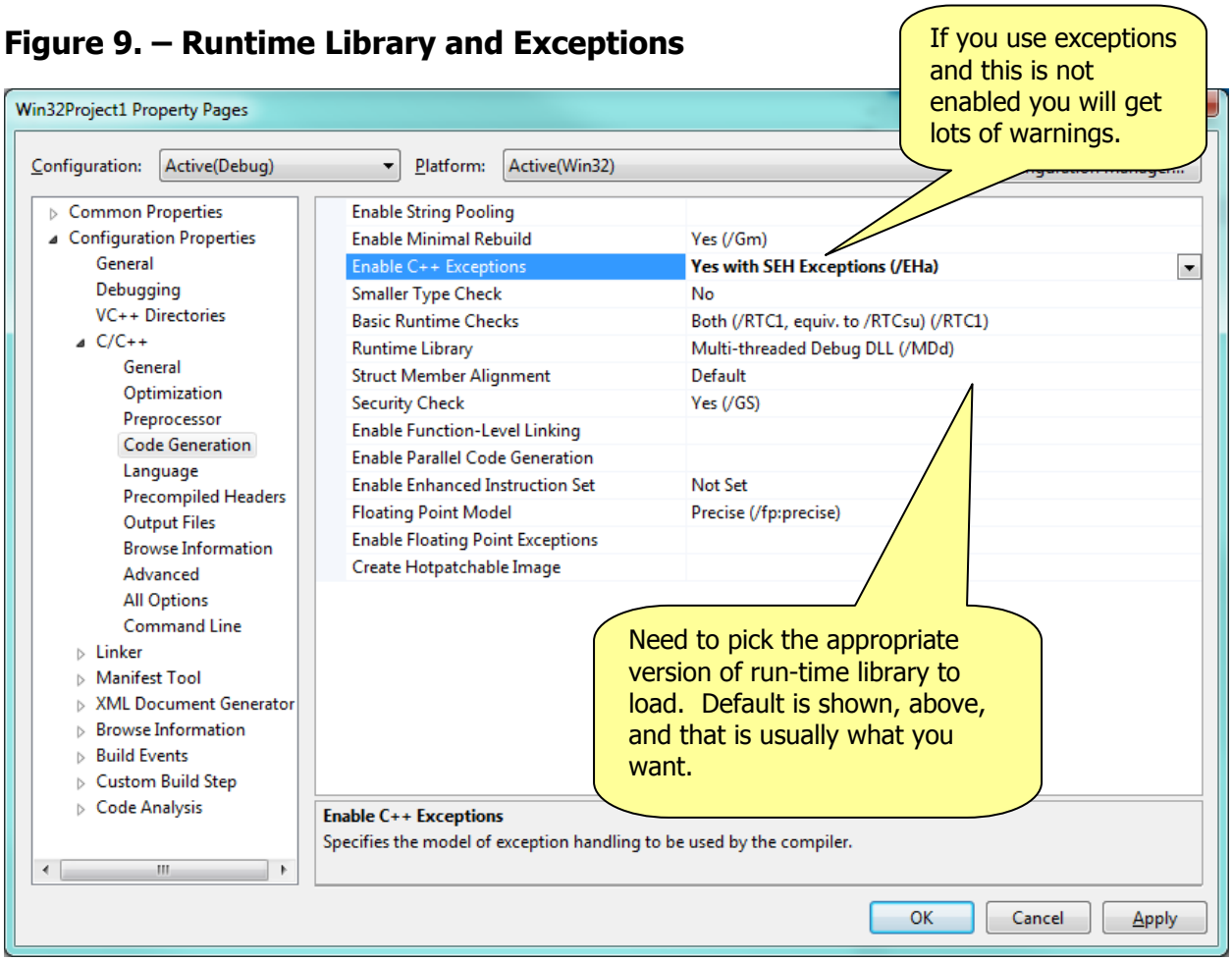

Figure 10. – Tabs and Spaces

Select a tab size to suit your preferences for indenting sizes. It is a good idea to select Insert spaces. The reason for that is, almost every printer on earth will, by default, print tabs as 8 spaces, which will make your printed listings look peculiar unless you use spaces instead of tabs.

Note that, if you print from the idea, the IDE commands the printer to use correct spacing, but if you print with any other program, or from the command line, you will get the printer defaults.