How to Create a New Visual Studio 2015 Console Project

The following will assume you have Visual Studio 2015 (Enterprise or Community) installed on your machine.

1. Open Visual Studio 2015. Signing in is not required.
2. On the left-hand navigation menu, click on "New Project...". A popup window should appear.
3. Select C++ in the Templates menu, then Win32 Console Application.
4. Name your project and, if required, change the location it will be saved to.
5. Press OK. Another popup window should appear.
6. Click the "Next >" button.
7. Under "Additional options", check the box marked "Empty project".
8. Press "Finish". Your project should now be created.
9. In the Solution Explorer on the right of the screen, right-click on "Source Files", then choose "Add", then "New Item...".
10. Ensure the file type you have selected is .cpp.
11. Name the file "Main.cpp".
12. Click "Add". Your new file should now open.
13. Create a C++ "Hello World" program.
14. In the toolbar just below the dropdown menu bar, click the button with the green Play icon that says "Local Windows Debugger".
15. If a box pops up asking if you want to compile changes you've made, check the box that says "Don't ask me again" and hit "Yes".
16. Your program should now be running.

How to Create a Breakpoint and Use Debugging

The following will assume you have completed the steps in the first section.

1. Find the line of code you want to add a breakpoint on.
2. On the far left of the window, click on the portion of the off-colour bar that aligns with the line you wish to add the breakpoint on.
   a. Alternatively, right-click on the line, then choose "Breakpoint", then "Insert Breakpoint".
3. Click the "Local Windows Debugger" button to compile and run the program.
4. Program execution should now halt at the designated breakpoint.

In the space where the "Local Windows Debugger" button used to be, additional controls should have appeared.

- The "Continue" button will resume normal program execution.
- The red square button will stop debugging and close the program.
- The button where a blue arrow is circling over a circle will step over to the next line, execute it, and then pause execution again.
- The button where a blue arrow is pointing towards a circle will step into the next function called on that line and keep execution paused - in other cases, it acts the same as the above button.

- The button where a blue arrow is pointing away from a circle will execute everything remaining in the current function, go to the line that called that function, and then pause execution again.

To view the value of a variable during debugging, look at the bottom-left corner of the window. This will list the name of a variable, the value it currently stores, and its type, in tabular format.

How to Utilize Build Configurations

The following will assume you have completed the steps in the first section.

1. To the left of the "Local Windows Debugger" button, there are two drop-down menus. By default, these should read "Debug" and "x86".
2. The right dropdown controls the CPU architecture being targeted. x86 applications are 32-bit and come with all the restrictions thereof, although they can run on x64 systems. x64 applications are natively 64-bit, which also means they cannot run on x86 systems.
3. The left dropdown controls the build configuration mode. "Debug" mode compiles code-based debugging directives, like "System.Diagnostics.Debug.WriteLine" in C#. "Release" mode does away with code-based debugging directives, and as such it is what should be used when compiling a project for release.
   These modes exist so that source code can contain things like detailed logging that only appears in debug mode, meaning all the I/O doesn't impact performance in the release build.
   a. Note that, even if "Release" mode is chosen, breakpoints will still pause execution, although variable inspecting will be disabled.