# CSci 515 Setting Up Visual Studio Express 2010

Spring 2015

## 1 Line Numbers

I prefer to always have line numbers enabled in my source code editor as most compiler errors/warnings refer to specific line numbers in the source file, and thus it is easier to quicly determine the potential location of problems if line numbers are visible. For Visual Studio Express 2010 you can enable line numbers like this:

1. Tools → Options
2. In Options Dialog, select Text Editor → C/C++ → General
3. Enable the Line Numbers checkbox

## 2 Program Indentation and Tab Settings

For this class, you are required to correctly and consistently indent your code according to the Deitel good programming guidelines, and our class coding/formatting guidelines. You can have Visual Studio Express 2010 automatically set your indentation for you by changing the following settings:

1. Tools → Options
2. In Options Dialog, select Text Editor → C/C++ → Tabs
3. Set Indenting to Smart
4. Set Tab size to 2
5. Set indent size to 2
6. Select Insert Spaces (rather than Keep tabs)

The last step will keep you from creating files with hardcoded, embedded tabs, which will not display properly in other editors.

# 3 Display/Undisplay Invisible (space) Characters

If you still have problems with getting hardcoded tabs in your source files, most programming editors have a command that makes normally invisible whitespace characters become visible, using symbols for the different types of characters (space, tab, etc.). To toggle the display of invisible whitespace characters on and off, you can:

## 3.1 Method 1

1. Turn on advanced settings from Tools → Settings → Expert Settings
2. Toggle display of white space on/off from Edit → Advanced → View Whitespace

## 3.2 Method 2

The keyboard shortcut for this toggle is `Ctrl-R Ctrl-W`.

# 4 Cause Terminal/Console to Persist after Program Execution

You are forbidden to include statements in your submitted programs whose sole purpose is to keep the terminal from closing upon program completion, before you can see the output. Statements like `system("pause")` or `getch()` are hacks, and are often not portable to other enviornments or IDE systems. When using Visual Studio Express 2010 you can have your IDE keep the console up in the following 2 ways:

## 4.1 Method 1

Simple use of debugger. You can simply set a debug breakpoint on the closing brace of your main function. Then whenever you run your program (using Debug → Start Debugging or equivalently using the `F5` function key), your program will run till it hits this breakpoint, and you can then look at the terminal output.

### 4.2 Method 2

You can also set a property for console applications, so that if you run without debugging, Visual Studio keeps the terminal open until you press a key. You need to set the following property for each project you create (Visual Studio doesn't enable this by default on new projects):

1. Project → Properties
2. In the Project Property Pages dialog select Configuration Properties → Linker → System
3. In the Linker System options, pull down the SubSystem property and select Console (/SUBSYSTEM:CONSOLE)
4. Hit OK or Apply to have your property setting saved (for this particular Project only, you have to do this for every project you create).

Now if you run your program with debugging (Debug → Start Without Debugging or equivalently `Shift-F5`), your terminal will be paused when the program completes execution.

## 5 Increase/Decrease Editor Font Size

For readability, you can increase/decrease the font size of the Visual Studio programming editor. Use Ctrl-Shift-, to increase font size, and `Ctrl-Shift-.` to decrease text size. You can also directly set the zoom level in the lower left of the editor frame.

## 6 Project Creating Steps for Class Assignments

For this class, you need to submit plain/standard C/C++ code (code that does not use special 3rd party libraries or frameworks. One confusing aspect of Visual Studio is that it supports many types of (Microsoft) specific application and framework targets. For this class, you should always use an Empty Project, which will allow you to create programs using standard C/C++ libraries and no external framework. Always follow these steps when creating a new project for assignment for this course:

1. File → New → Project
2. In the New Project Dialog, select Visual C++ → General

3. Choose Empty Project (Visual C++) as your project type in the center.
4. Enter an appropriate name for the project.
5. Select OK to create the project.
6. Once your IDE and Solution Explorer open with the newly created project, add a source file.
7. In the Solution Explorer, right click on the Source Files folder
8. Select Add → New Item
9. In the Add New Item dialog, select C++ File in the center
10. Choose an appropriate name for you C++ source file, you may be required to name the file with a particular name for an assignment. If so use the required name for the file.
11. Hit the Add button to create the empty source file and add it to your managed Source Files for the project.
12. Copy the boilerplate C++ Hello World program to your empty file, and make sure that it correctly compiles and runs and generates the expected output. You can find the official HelloWorld boilerplate on our eCollege course site.

Here is the boiler plate you should use (but don't type this in by hand every time), including the correct file header and an example function header for the main function.

```
/**
 * @author Joe Programmer
 * @cwid   123 45 678
 * @class  CSci 515, Spring 2015
 * @ide    Visual Studio Express 2010
 * @date   January 1, 2015
 * @assg   ExamplAssg00
 *
 * @description A short descrption of the program and its purpose, and
 *    the approach you took in solving the problem.
 */
#include <iostream>
using namespace std;

```

```
15
16 /** main entry point
17  * The main entry point for this program.  Execution
18  * of this program will beigin with this function.
19  *
20  * @returns An int value.  By default, if we don't specify a return or
21  *           exit value, 0 is returned to indicate successful program
22  *           completion.  A non-zero value indicates an error or
23  *           problem with execution.
24  */
25 int main()
26 {
27   cout << "Hello, world!" << endl;
28
29   // return 0 to indicate successful completion
30   return 0;
31 }
```