Hello World Lab

https://csci-1301.github.io/about#authors

May 19, 2021 (09:06:46 PM)

Contents

1 Your First Program **1**
1.1 Opening Your First Program 1
1.2 Compiling and Executing Your First Program 2
1.3 Configuring Visual Studio 2

2 Backups **4**
2.1 Finding The Right Tool 4
2.2 Making Sure You Have the Right Files 5
2.3 How Was the Backup? 5

3 Orientation **6**

4 Breaking Your Program **6**

1 Your First Program

1.1 Opening Your First Program

1. Download Welcome.zip from the same place where you downloaded this instructions file and save it on your computer.
2. Extract (Right-click on the file, then click on "Extract all") this archive. **Do not simply double-click on it, as that would only give you a preview of the archive without actually extracting it.**
3. Go in the "Welcome" folder that was created.
4. Double click on the "Welcome.sln" file.
5. If you are prompted with a screen like this one:

Pick "Visual Studio 2019" (and **not** Visual Studio Code or Blend for Visual Studio).

1. Visual Studio (VS) should start. You don't have to register to the "Visual Studio Team Services Organizations" (but you can, using your @augusta.edu account, if you want), discard the security warning if there is one.
2. In the "Solution Explorer" to the right, expand all the items that can be expanded by clicking on the (right-triangle) symbol.

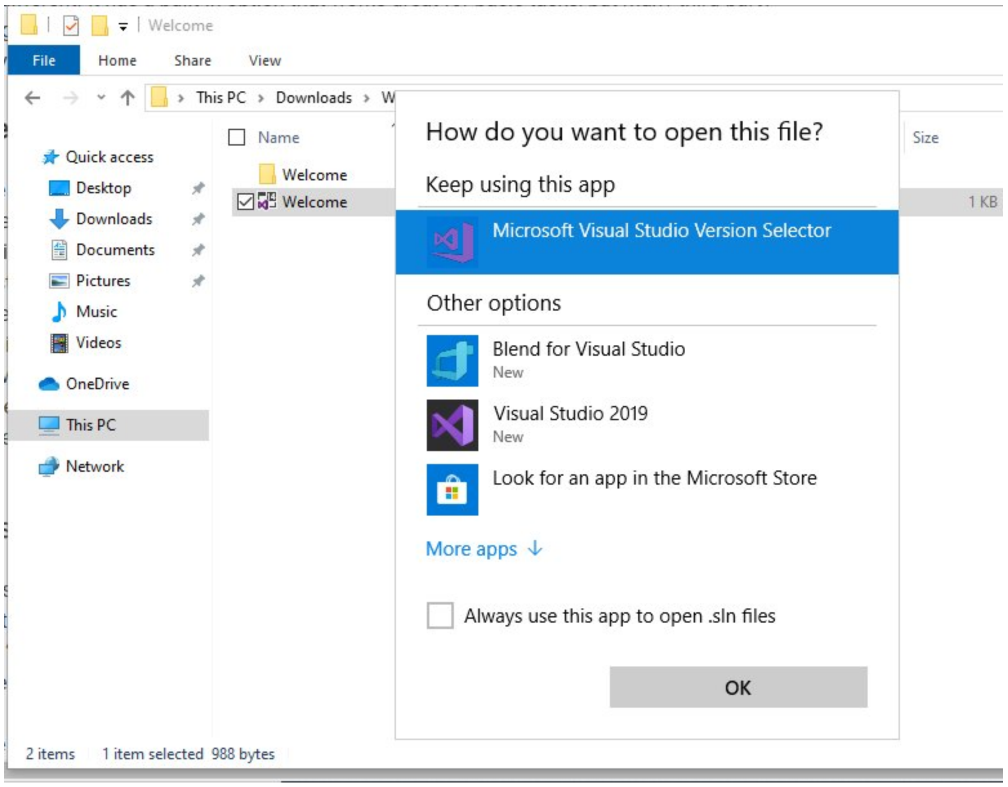


Figure 1: "windows file picker asks how do you want to open this file?"

1.2 Compiling and Executing Your First Program

1. In the Solution Explorer, double-click on `Program.cs`. This is the *source code* of the application you are actually considering.
2. Let's compile this program, by clicking `Build` → `Build solution`. What happened?
3. Let's run this program, by clicking `Debug` → `Start without Debugging`. What happened?

You will **extensively** compile and run programs in this class. Instead of having to click twice, I highly recommend that you start now memorizing shortcuts:

- Use Ctrl + Shift + B to build the solution.
- Use Ctrl + F5 to start the program without debugging.

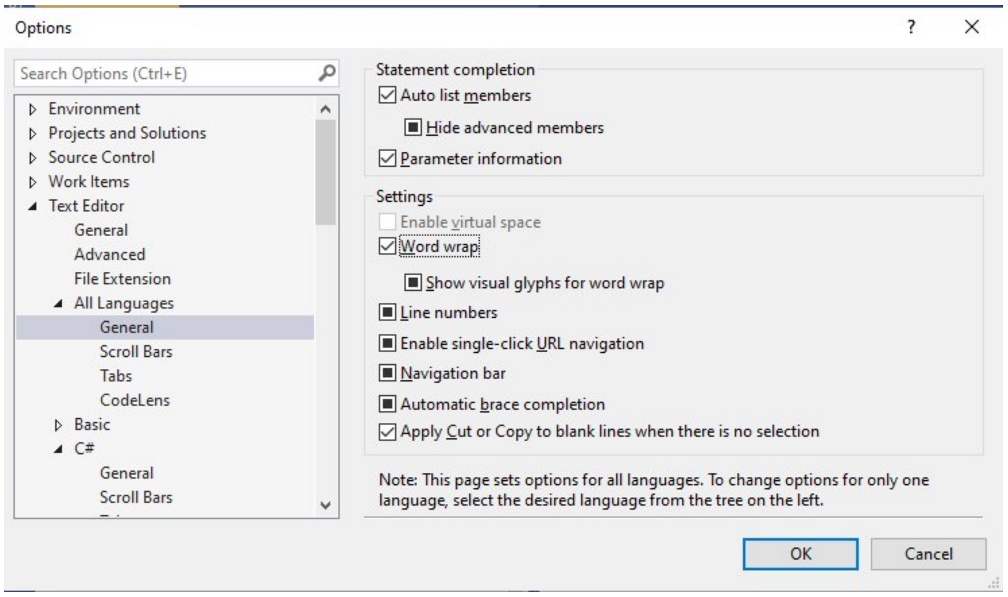
With Alt + F4 (to exit any program), that makes 3 shortcuts already! You can find a complete list at http://visualstudioshortcuts.com/ and on this page[1] of the Microsoft docs. I will try to introduce more useful shortcuts as we progress.

1.3 Configuring Visual Studio

Now that you have practiced starting and using Visual Studio, there are a few changes you should make to ensure your installation is set up in an easy-to-use way:

[1] https://docs.microsoft.com/en-us/visualstudio/ide/default-keyboard-shortcuts-in-visual-studio?view=vs-2019

1. Make Windows Explorer show file extensions. Follow the instructions described in Microsoft's help forum[2], on this website[3], or consult the textbook's "Before You Begin" chapter.
2. In Visual Studio, make sure the line numbers are shown: Go to "Tools" → "Options" → "Text Editor" → "All Languages" → "General" → "Line Numbers" (VS 15.5.2), or "Text Editor" → "Options" → "All languages" → "Line Numbers" (newer versions).
3. Activate word-wrap in VS. Refer to this page[4] for VS 2015, or this page[5] for VS 2017 and 2019). The item to be clicked should look like this:



Statement completion

- ☑ Auto list members
  - ■ Hide advanced members
- ☑ Parameter information

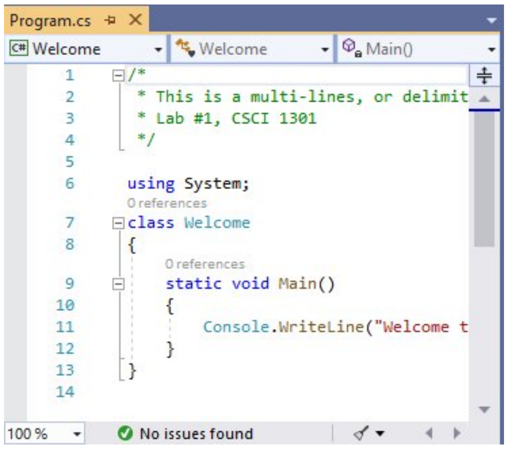
Settings

- ☐ Enable virtual space
- ☑ Word wrap
  - ■ Show visual glyphs for word wrap
- ■ Line numbers
- ■ Enable single-click URL navigation
- ■ Navigation bar
- ■ Automatic brace completion
- ☑ Apply Cut or Copy to blank lines when there is no selection

Note: This page sets options for all languages. To change options for only one language, select the desired language from the tree on the left.

Figure 2: "how to enable word wrap on Windows version of VS"

If you were successful, you should go from

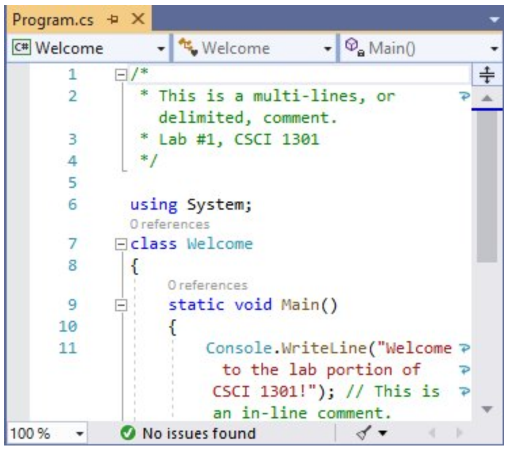


```
/*
 * This is a multi-lines, or delimit
 * Lab #1, CSCI 1301
 */

using System;
0 references
class Welcome
{
    0 references
    static void Main()
    {
        Console.WriteLine("Welcome t
    }
}

```

Figure 3: "VS editor before enabling word wrap"

to

[2]https://answers.microsoft.com/en-us/windows/forum/all/in-win10-how-to-show-the-file-extension-for/ed21ff20-cdb3-4263-9c7d-fc6ed125fc82
[3]http://kb.winzip.com/kb/entry/26/
[4]https://msdn.microsoft.com/en-us/library/ms165339.aspx
[5]https://docs.microsoft.com/en-us/visualstudio/ide/reference/how-to-manage-word-wrap-in-the-editor



```
/*
 * This is a multi-lines, or
   delimited, comment.
 * Lab #1, CSCI 1301
 */

using System;
0 references
class Welcome
{
    0 references
    static void Main()
    {
        Console.WriteLine("Welcome
         to the lab portion of
        CSCI 1301!"); // This is
        an in-line comment.
```

Figure 4: "VS editor after enabling word wrap"

```
See the difference?
Once word-wrap is enabled, the horizontal scrolling will disappear, and every line that is too long is
```

2 Backups

Now we need to make sure you know how to save your work and access it. This is especially important if you are using the computer lab rooms, as **you can not store files permanently on the lab's computer, you will have to store them either online in your cloud storage or on a USB drive**.

2.1 Finding The Right Tool

You can save your project:

- On your hard drive, if you are using your own computer.
- On an external/removable data storage: USB flash drive, external hard disk drive, or any kind of USB mass storage device.
- On a server: the University has a partnership[6] with box.com[7], and you can follow this tutorial[8] to get started, but any service (Google Drive[9], Dropbox[10], OneDrive[11], etc.) would do.

Having *two* backups is generally recommended.

If you chose the "virtual" option (i.e., using a server) and you are in a computer lab, **do not** try to install a synchronization program (like Google Drive and Sync[12], Box's app[13], etc.) on the lab computer: it will likely not work, due to University rules[14]. Instead, create the structure/project/files on the computer during the lab and upload them (using the web-interface) at the end of the lab. Make sure to always upload your files before logging out of the computer.

[6] https://www.augusta.edu/its/box/
[7] https://box.com/
[8] https://www.augusta.edu/its/box/quickstart.php
[9] https://www.google.com/drive/
[10] https://www.dropbox.com/
[11] https://onedrive.live.com/
[12] https://www.google.com/drive/download/
[13] https://app.box.com/services/browse/official
[14] https://augusta.policytech.com/dotNet/documents/?docid=5702

2.2 Making Sure You Have the Right Files

Now that you know where to store your files, create a folder for this class, and a subfolder for the first lab. Put all of the files related to the "Welcome" solution in this folder. You organization should look like the following:

```
csci1301
    01_lab
        Welcome.zip
        Welcome
            Welcome.sln
            Welcome
                Welcome.csproj
                Properties
                    AssemblyInfo.cs
                Program.cs
                obj
                    Debug
                        Welcome.pdb
                        Welcome.exe
                        Welcome.csprojResolveAssemblyReference.cache
                        Welcome.csproj.FileListAbsolute.txt
                        TempPE
                        TemporaryGeneratedFile_E7A71F73-0F8D-4B9B-B56E-8E70B10BC5D3.cs
                        TemporaryGeneratedFile_5937a670-0e60-4077-877b-f7221da3dda1.cs
                        TemporaryGeneratedFile_036C0B5B-1481-4323-8D20-8F5ADCB23D92.cs
                        DesignTimeResolveAssemblyReferencesInput.cache
                bin
                    Debug
                        Welcome.vshost.exe.manifest
                        Welcome.vshost.exe.config
                        Welcome.vshost.exe
                        Welcome.pdb
                        Welcome.exe.config
                        Welcome.exe
                App.config
```

where `csci1301` and `01_lab` are folders, and `Welcome.zip` is a file.

You do not need to check that every single file and folder is here, just note that you have multiple folders, and that there are many files in the `Welcome` folder, not only the `.sln` and the `.cs`: make sure you copy the entire structure when you want to backup or share (i.e. turn in) your program! In this case, copying the `Welcome` folder is enough.

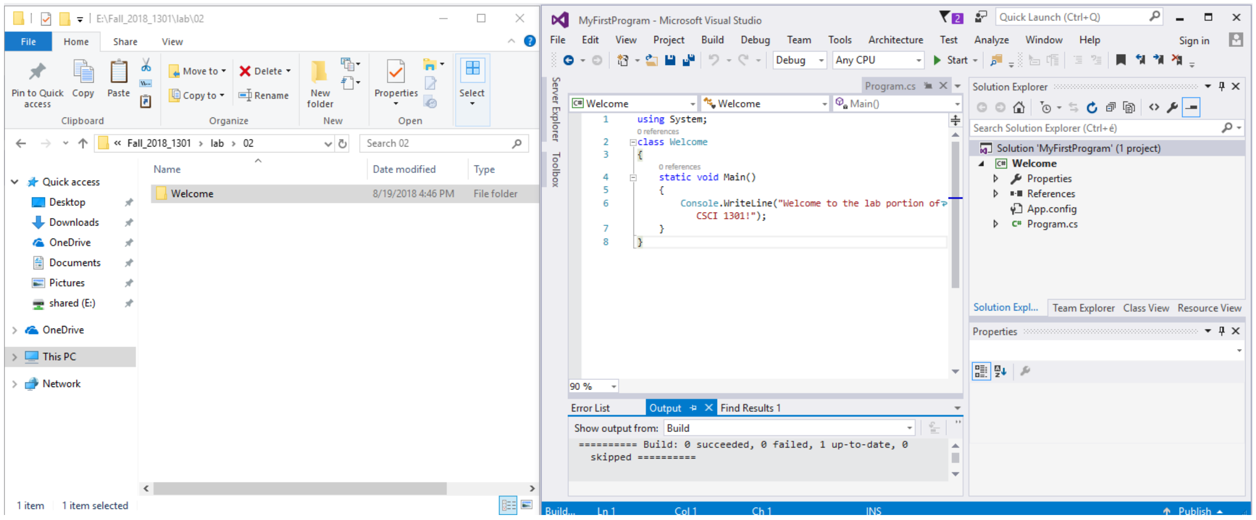
2.3 How Was the Backup?

Once you are done, test that you performed the backup properly. Re-download or transfer the files you just saved (the whole `Welcome` folder) on the computer and make sure you can still open the project with Visual Studio (VS). Do you remember how to build the solution and start the program without debugging? Using the shortcuts? If not, go have another look back at the "Compiling and Executing Your First Program" section.

If your backup went wrong (you can't open the project, it won't compile, ...), try to understand what happened. Then, re-download the "Welcome" archive (Welcome.zip), extract it, and make sure you can build the solution and start the program without debugging.

3 Orientation

Have a look at the following screenshot:



```
using System;
class Welcome
{
    static void Main()
    {
        Console.WriteLine("Welcome to the lab portion of
            CSCI 1301!");
    }
}
```

```
========== Build: 0 succeeded, 0 failed, 1 up-to-date, 0
  skipped ==========
```

Figure 5: “image shows windows file system with VS IDE instance”

You should have roughly the same on your computer. If this image appears too small, you can zoom to have a clearer look.

Answer the following:

1. Where is the file explorer?
2. How many folders are in the folder opened in the file explorer?
3. How many files are in the folder opened in the file explorer?
4. Where is Visual Studio (VS)?
5. Where is the solution explorer?
6. Where is the window where you can edit the source code?
7. What is the output of the program you just built?

4 Breaking Your Program

If you followed the instructions carefully, your were able to build the solution and start the program without debugging after each step. As you know, C# has precise rules and not respecting them can prevent your solution from being built by VS.

In this exercise, you are asked to do the following:

1. Change the program so that it violates one of the syntax rules of `C#`.
2. Build the solution and note that an error is reported. In the build output (cf. the screenshot presented earlier, you may need to click “View” → “Output” or switch tabs in VS to see it), you will see a message like

```
========== Build: 0 succeeded, 1 failed, 0 up-to-date, 0 skipped ==========
```

3. Click on the “Error list” tab next to the build output, and make sure you understand the error message.
4. Undo your change, using Ctrl + z.

5. Make sure you can build the solution without a new error message.

Do it three times, in order to identify three different error messages, and three ways of breaking C#'s rules.

If you have time or need ideas, you can try with the following, and see which one(s) make the building impossible (do not forget to undo your change after):

- Remove the semicolon after `using System`
- Replace `class Welcome` with `class TestOne`
- Remove the brace (or "curly bracket", i.e., the `}` symbol) at the last line.
- Add three new lines at the end of the file
- Replace `Console.WriteLine` with `CONSOLE.WriteLine`
- Replace `Console.WriteLine` with `Console.WRITELINE`
- Add a new line between `Console.` and `WriteLine`
- Add a new line between `WriteLine` and `(`
- Add a new line between `Write` and `Line`
- Replace `Main()` with `Method()`
- Remove the indentation (i.e., the space between the beginning of the line and the first character of the instruction) on all lines