Important Things to Consider for E 115 Intro to Programming 2019 edition, Windows Users

updated, June 24, 2020

Ethan Dy Tioco
E 115
Stevens Institute of Technology

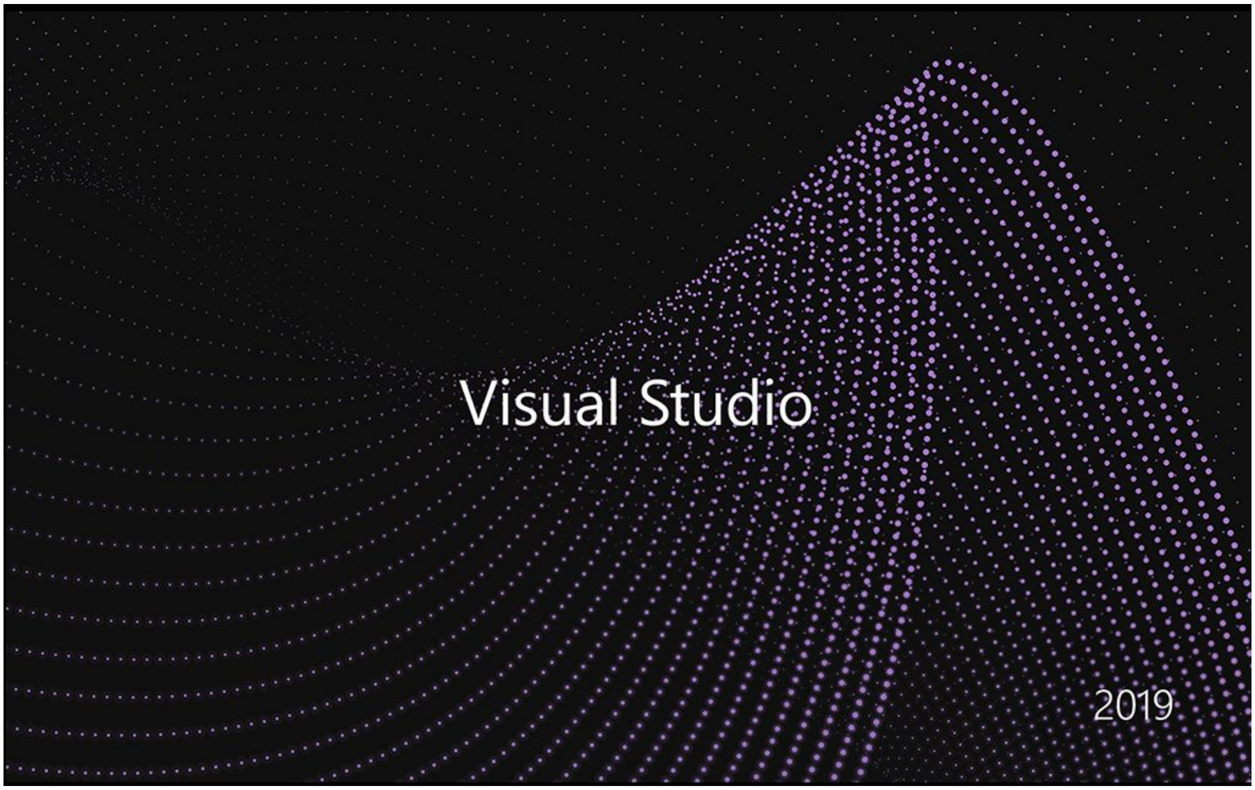


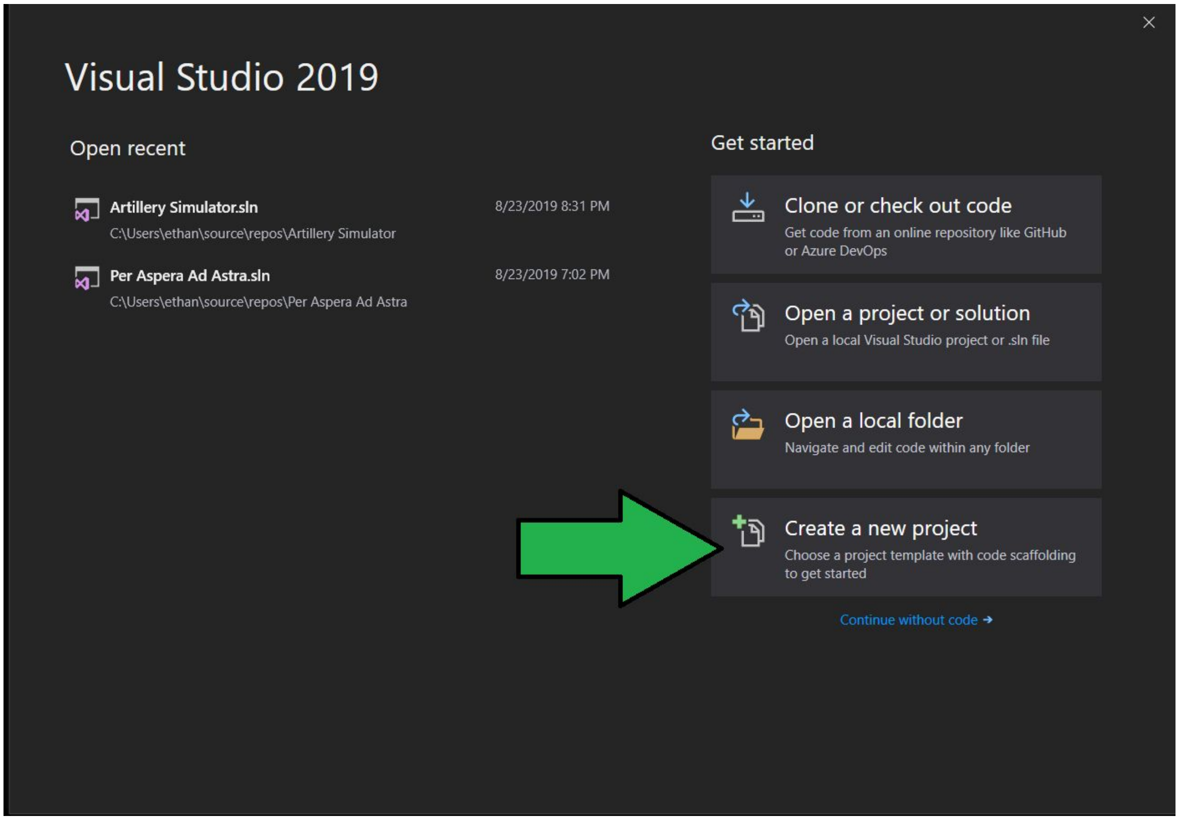
Table of Contents

Visual Studio:

Making a project:

Visual Studio requires that projects are created so that you can write and run C++ programs. It is fairly simple to make them, but there are small nuances in this process that can force a student to follow different rules in Visual Studio, apart from the rest of the class. That means that this student will not be typing out the same code as everyone else, and it may result in a harder time in class. The purpose of this section, and ultimately this paper, is to have everyone on the same page. This will make it much easier for students, TA's, and professors to interact with each other.

Visual Studio 2019 has slightly changed its interface, but it's intuitive enough. For PC users, click on ***Create a New Project***, and then ***Empty Project*** (Fig. 1a, 1b). Fill in the project name with something that makes sense, so that you can quickly find your particular homework or project later on. Finally, check the box that says **Place solution and project in the same directory** (Fig. 1c).



Visual Studio 2019

Open recent

Artillery Simulator.sln 8/23/2019 8:31 PM
C:\Users\ethan\source\repos\Artillery Simulator

Per Aspera Ad Astra.sln 8/23/2019 7:02 PM
C:\Users\ethan\source\repos\Per Aspera Ad Astra

Get started

Clone or check out code
Get code from an online repository like GitHub or Azure DevOps

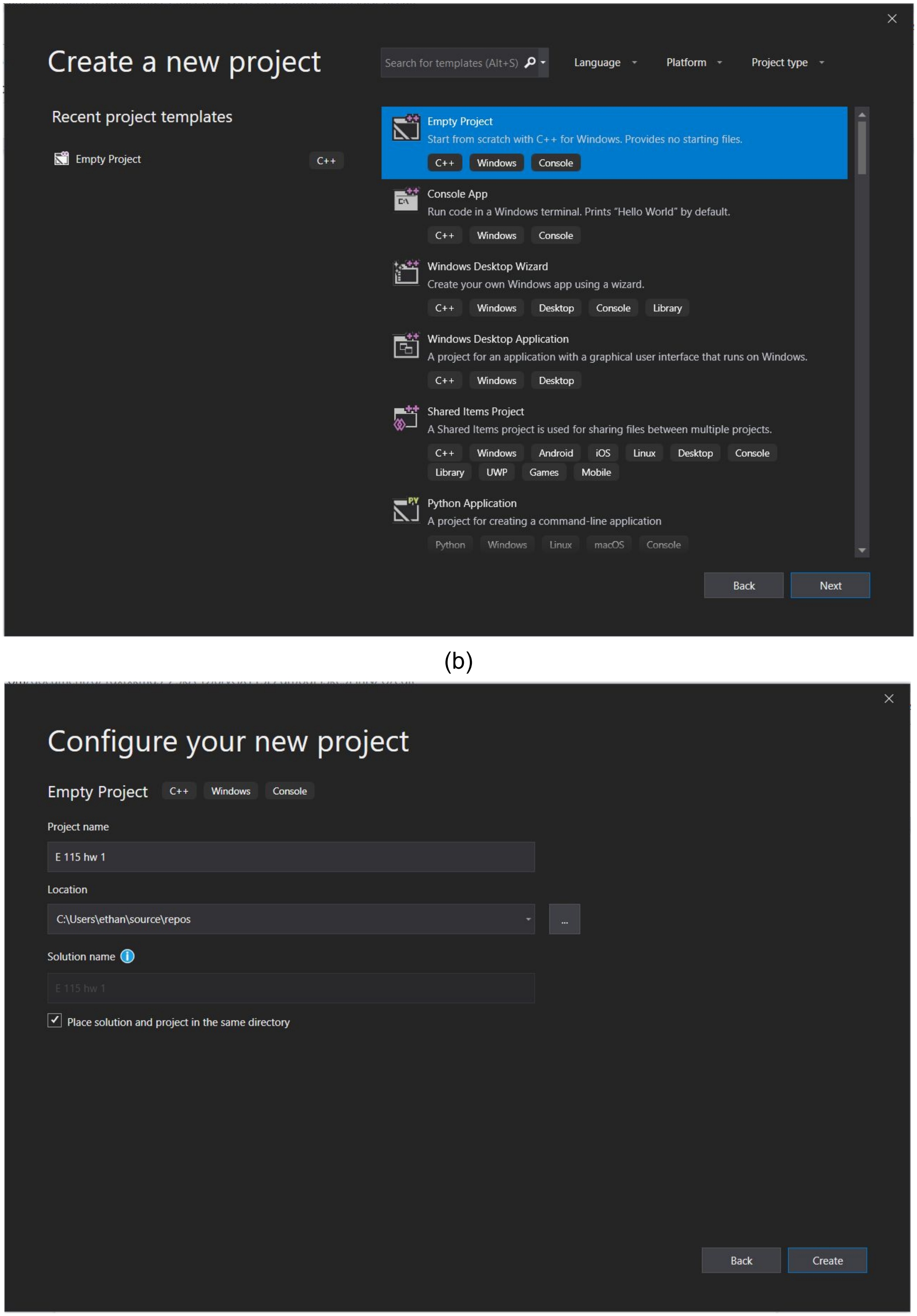
Open a project or solution
Open a local Visual Studio project or .sln file

Open a local folder
Navigate and edit code within any folder

Create a new project
Choose a project template with code scaffolding to get started

Continue without code →

(a)



Create a new project

Search for templates (Alt+S)

Recent project templates

Empty Project C++

Empty Project
Start from scratch with C++ for Windows. Provides no starting files.
C++ Windows Console

Console App
Run code in a Windows terminal. Prints "Hello World" by default.
C++ Windows Console

Windows Desktop Wizard
Create your own Windows app using a wizard.
C++ Windows Desktop Console Library

Windows Desktop Application
A project for an application with a graphical user interface that runs on Windows.
C++ Windows Desktop

Shared Items Project
A Shared Items project is used for sharing files between multiple projects.
C++ Windows Android iOS Linux Desktop Console Library UWP Games Mobile

Python Application
A project for creating a command-line application
Python Windows Linux macOS Console

(b)

Configure your new project

Empty Project C++ Windows Console

Project name

E 115 hw 1

Location

C:\Users\ethan\source\repos

Solution name

E 115 hw 1

☑ Place solution and project in the same directory

(c)

Fig. 1 Making a new project in Visual Studio. The name given to this project is “E 115 hw 1”.

Making the source file (.cpp):

Sometimes, Visual Studio by default does not show the Solution Explorer. You will need to see it to make your source file in the proper place. For PC users, go to ***View >> Solution Explorer***, or enter ***Ctrl+Alt+L***.

Then, wherever the Solution Explorer pops up in your window, go to ***Source Files >> Add >> New Item…*** (Fig. 2)

A new window will pop up. Hopefully by default, ***C++ file (.cpp)*** is selected. If not, then select it. Then, it is **strongly suggested** that you give the source file a descriptive name (Fig 3). Once you have this, you are ready to code for E 115!

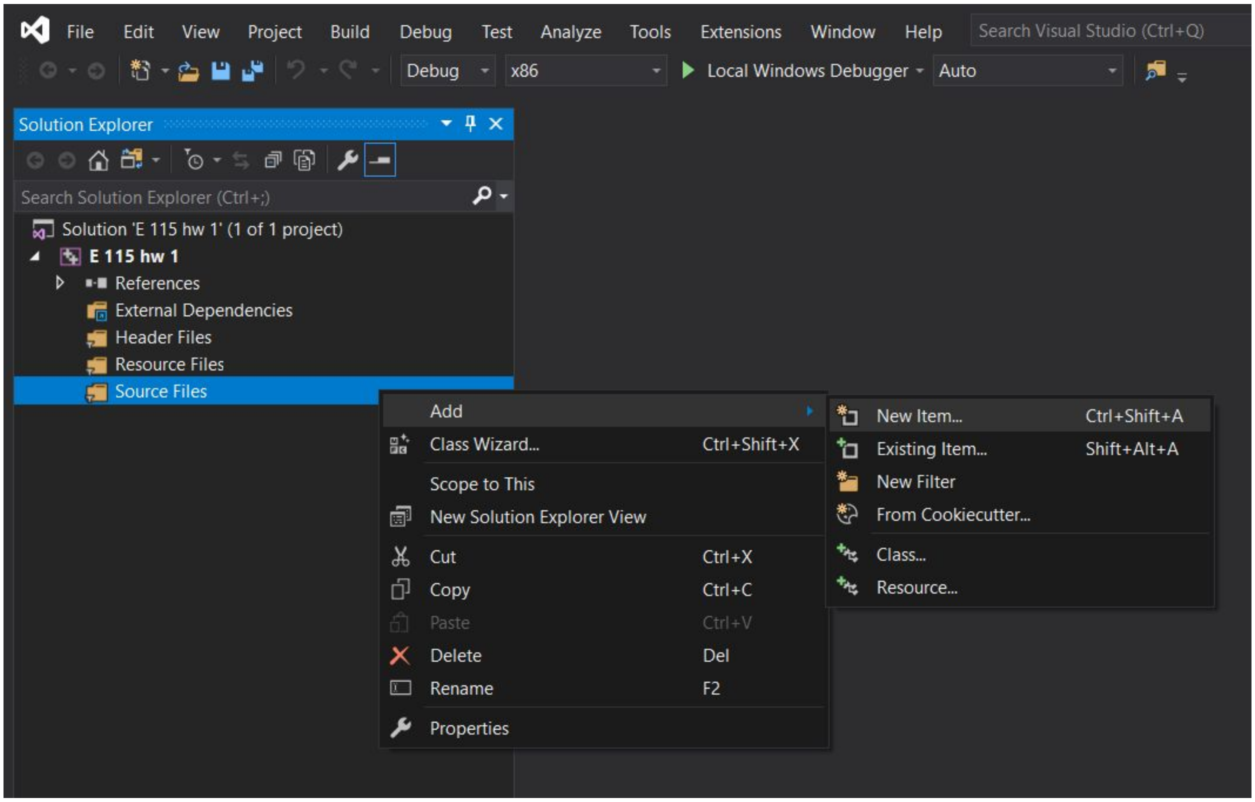
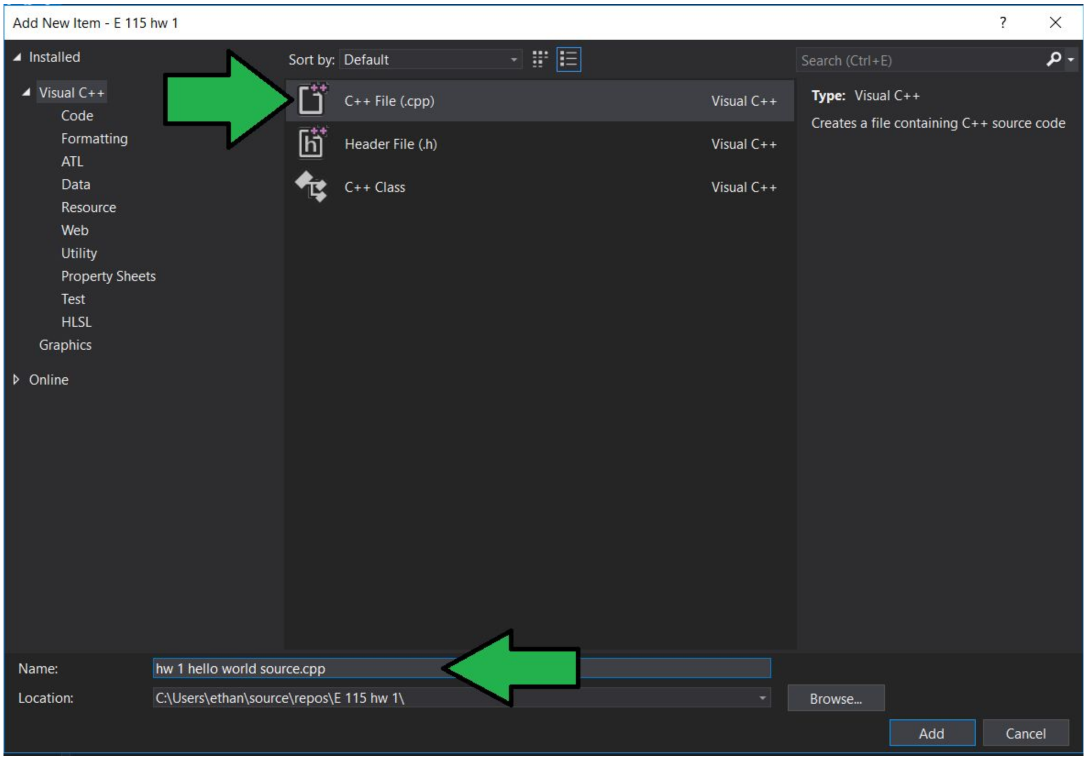


Fig. 2 Creating the source file in its proper folder.



Add New Item - E 115 hw 1

Sort by: Default

C++ File (.cpp)	Visual C++
Header File (.h)	Visual C++
C++ Class	Visual C++

Type: Visual C++

Creates a file containing C++ source code

Name: hw 1 hello world source.cpp

Location: C:\Users\ethan\source\repos\E 115 hw 1\

Fig. 3 Naming the C++ source file. In this example, the file is named “hw 1 hello world source.cpp”

Typing your .cpp program:

Once you've set up your project, you should be met by an empty .cpp file that you can type in. For this class, it's great to always start off with the following code:

```
#include <iostream>
using namespace std;

int main(){
      // Your code goes here!


      return 0;
}
```

This is a good start for your program - go ahead and copy it! You will be fleshing it out as you go, adding or subtracting libraries, functions, classes, and so on. If you're not interested in the technical stuff on these lines of code, then just skip to the next section.

The first line, **`#include <iostream>`**, lets the program call a library that lets you use input and output to the console.

The second line, **`using namespace std;`**, lets you use the "standard" namespace, where a lot of the functions like `cin` and `cout` are defined in. The full name for `cout` is actually `std::cout`, and for `cin` it's `std::cin`. Without this line of code, you will have to write the prefix `std::` for every time you call `cin` and `cout`, and every other function that requires it. So, save yourself the trouble and make your code look cleaner by writing out the second line.

It's important to make the **`main()`** function an `int` type and finish the function with the line **`return 0;`**. By convention, returning a zero from `main()` means that your program successfully completed, and returning a non-zero value means that it did not successfully complete[1, 2]. For this class, and especially if you think that you'll be working extensively with the command prompt as you code, **do not forget the line *`return 0;`* at the end of your *`main()`***. But, if you do forget to write it in, some systems will make `main()` return a zero by default. So, why even bother? Because good coding practices like adhering to a function's return type is important to be learned sooner than later.

In Visual Studio 2019, you no longer need the line **`system("pause");`**. In the off-chance that you *do* need this line of code, then just put it in before `return 0;`. No big deal.

tl;dr just copy the code above to start off your program :)

Handing in Class Material

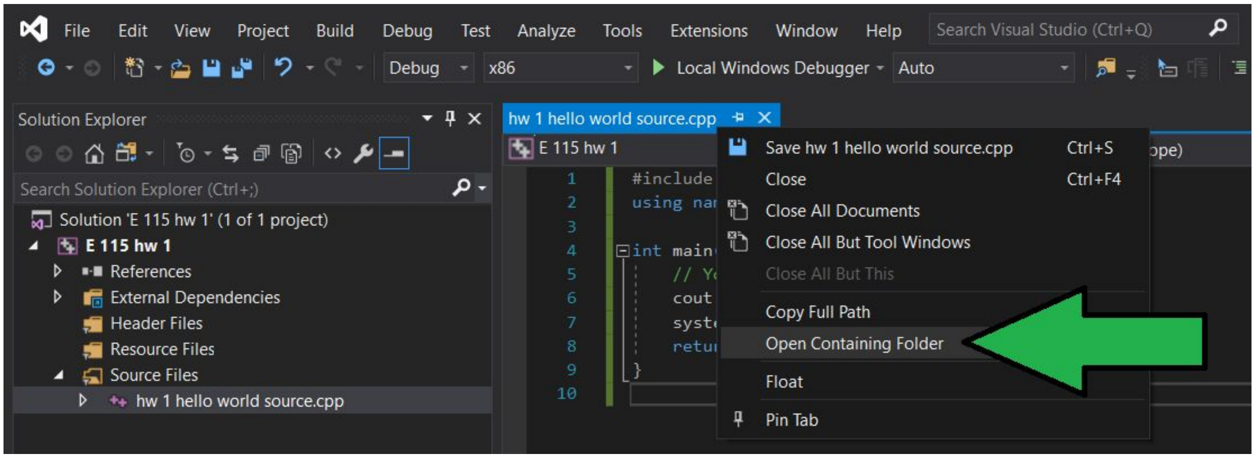
Handing in the program:

This paper assumes that Stevens is using Canvas as its learning management system, where all your classes are. There are a few *important* points you need to consider when handing in materials for E 115:

NEVER hand in the .sln file >:(This does not contain the actual code. That shows the Visual Studio diagnostics, and is only good for opening a project *while already in Visual Studio.* If you *do* hand in the .sln file, you're only going to bring wrath upon yourself.

DO hand in the .cpp source file :) This is where your actual written code is. By default, it's called "Source.cpp". If you've followed the suggestion above, it will be whatever you've named it.

To get the actual .cpp source file, right click on the tab you're working on, and click **Open Containing Folder** (Fig. 4). You will see that your project folder is inside another folder called **repos**. It would be a good idea to pin this to Quick Access, so that you can find all of your projects quickly (Fig. 5).



```
#include
using nar

int main
    // Yo
    cout
    syst
    retur
}
```

Fig. 4 Finding where the source file is located on your computer.

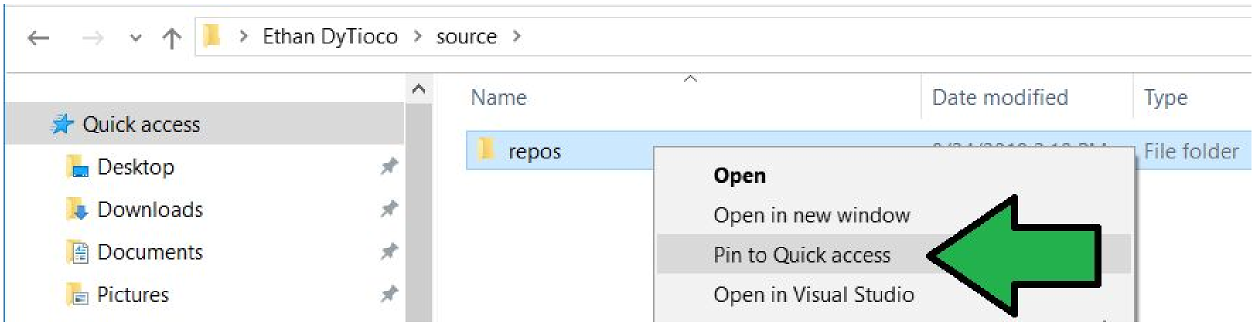


Fig. 5 Pinning the repos folder to Quick access. It will be listed along with the rest of the folders on the left side of the file explorer window.

Homework:

For homework, submit the following:

1. **.cpp file**
2. **screenshots of the command window running the code (i.e. the output)**

The TA's ultimately want to see the code, and proof that it runs. We may ask for additional things, such as screenshots of the code itself, for the purpose of fulfilling the TA's end goal.

If you've missed the deadline, don't worry! Always strive to hand in your homework, even if it's late. Homework contributes to a *huge* part of your grade, so it's best to have a healthy partial credited homework, than no homework at all. Make sure to communicate with your TA's, and don't wait until the last week of the semester to hand everything in.

Projects:

Hand in all essential files that are included in your project on canvas. Many great projects can be condensed into one .cpp file, just like your homework. Some may require a .txt file for file streaming. Other projects can be full-blown games made in Unity, which would require either an installation file, or the .exe game file and a folder with all the game data. Whatever files you made, include them all. If you're handing in a folder, it's a good idea to make it a .zip file.

Each group member should individually hand in their group project on Canvas. Groups can be made across different sections, which is good! However, if only one member hands it in for the whole group, it will be hard for TA's to track down which student did what project.

HAND IN THE PROJECTS ON TIME. You'll read later that TA's have to follow strict grading deadlines. Please spare them the heartache of giving you a zero or a heavily-penalized grade because you did not submit your project on time.

Teaching Assistants (TA's)

Contact Information:

Make sure that you have your TA's email addresses written down somewhere! Some of them are comfortable with sharing their phone numbers as well. TA-student communication is very important, and in many cases, *very time sensitive*. Keep a close eye out for emails or announcements from them.

Don't be afraid to ask the TA's for help! The average student for E 115 more likely does not have prior coding experience, so the class material can seem intimidating. The TA's are there to help and guide you to understand the fundamental programming concepts that are covered in this class. They *want* to help you out. So, whether it's a homework question, or a lecture topic question, or even a technical question (like navigating through Visual Studios), ask away and ask quickly!

If you have any grading questions as well, make sure to contact your TA's immediately. They have grading deadlines to follow, and it would be easier to sort things out sooner than later.

References

[1] https://docs.microsoft.com/en-us/cpp/cpp/program-termination?view=vs-2019

[2] http://www.cplusplus.com/reference/cstdlib/exit/