

# CSE 1321L – C#

## Introduction to C# and Visual Studio

### Lab 1

#### Goals:

- Become familiar with the Visual Studio Integrated Development Environment (IDE).
- Understand how to compile and run applications from within an IDE
- Explore how C# applications work and understand the main method.
- Have a basic understanding of I/O – or Input/Output. This means printing things to the screen and reading input from the user.
- Understand how to submit labs in Gradescope
- Understand the basics of the “autograder” in Gradescope, which auto-assigns a grade based on the output of your programs.

For each lab program you develop, make sure to include the following “header” - **replace the dots with your section #, semester, your full name, your instructor’s name, and lab #:**

```
/*  
Class:    CSE 1321L  
Section:  ...  
Term:    ...  
Instructor: ...  
Name:    ...  
Lab#:    ...  
*/
```

*Note: putting **\*anything\*** between a **/\*** and **\*/** (“slash star and star slash”) in most languages makes it a “comment” – which is ignored by the computer. For all your future labs and assignments, you are required to put the header so we know who submitted the file. Thank you ahead of time!*

#### What is an IDE?

An Integrated Development Environment, or IDE, is basically an application that programmers use to make their lives easier when writing and testing code. You can think of it as Word (or Pages if you use a Mac), but instead of using English or some other Human language, it allows programmers to efficiently enter/edit code and run it. More formally, it’s an application that provides almost everything needed by computer programmers for software development. At the core of it is a “compiler” which is responsible for taking (human-readable) source code and converting it into binary code (1s and 0s) to be executed. An IDE normally consists of at least a source code editor, build automation tools (like compilers), and a debugger. You can see a larger overview on Wikipedia:

[https://en.wikipedia.org/wiki/Integrated\\_development\\_environment](https://en.wikipedia.org/wiki/Integrated_development_environment)

*Realize that, technically, an IDE is not needed. You could work directly with the compiler from the command line if you wanted to. This is something you should explore later on, because some companies, like Apple, ask these things in job interviews.*

#### Why do we use an IDE?

IDEs simplify the whole process, allowing programmers to write and test code quicker and with less errors than using the compiler directly. On top of that, many modern IDEs enable multiple developers coordinate their work and also provide fast and intelligent error catching systems. So, all of this together makes any IDE a one-stop place to write, compile, run, and fix code.

## What is Gradescope and the Autograder?

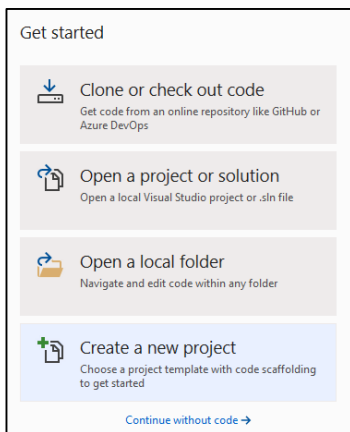
Gradescope is where you're going to submit your labs and assignments. For some of your labs (including this one), you'll submit your source code (which will end in .java for Java, .cs for C-Sharp, or .cpp for C++). When you do that, the autograder in Gradescope will take your source code, compile it, and run some test cases through it to see if you did what you were supposed to do. It then automatically assigns a grade.

However (and pay attention) – it's picky. For example, if the lab requires you to print "Hello, World!" and you print "Hello", you won't receive points for that part of the lab. The same thing would happen if you forgot the "!" at the end – so it's important to have a critical eye! The autograder will point out these errors when you submit. Second, if the lab or assignment says "The filename must be called X", you must do that and the case must match. For example, if you must call your file "Lab1A.java" and you called it "lab1a.java", the autograder may or may not recognize it depending on the mood it is in that day.

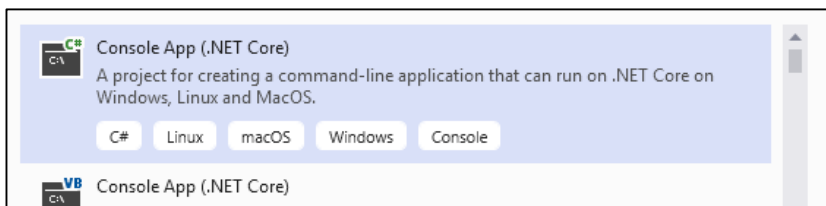
No worries – if it's not working for you, the autograder tries to give you feedback about what it's seeing based on input and output. Also, don't forget you have a lab instructor and lab assistants who are here to help.

## Directions:

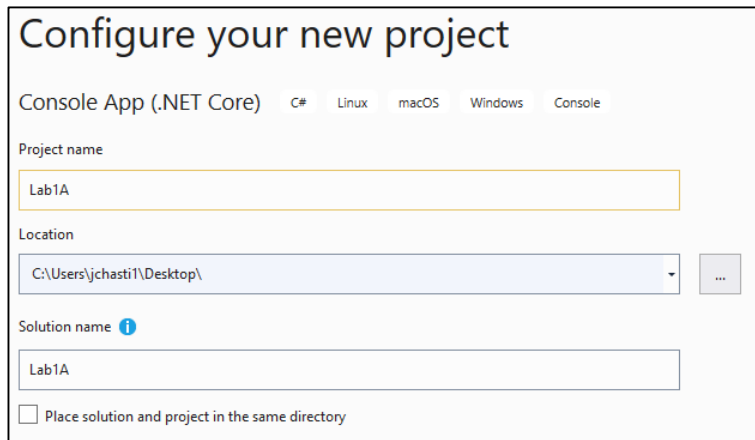
- 1) If you are working from home and have not installed Visual Studio, please visit "Integrated Development Environments" at <https://ccse.kennesaw.edu/fye/resources.php>
- 2) Open Visual Studio 2019 on your computer. It will prompt you with the following dialog box. Click "Create a new Project"



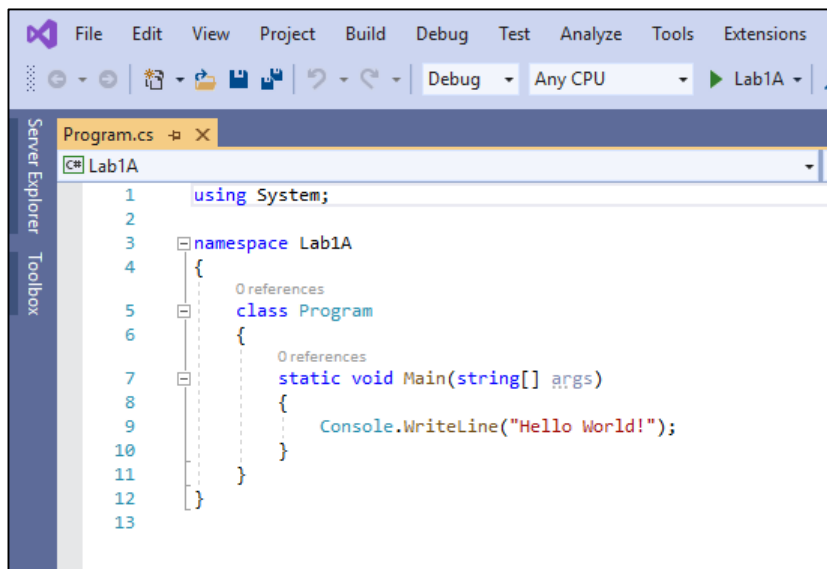
- 3) Next, it will want to know what kind of project to create. You will want to create a Console App, but specifically for C# (there are other options that look similar, so be careful).



- 4) Next, you will name the project/solution. In this case, call it Lab1A. Please make sure you capitalize correctly, because languages are case-sensitive (e.g. don't call it "lab1a").



- 5) Click Create Project. You should finally see some C# code.  
6) You can compile and run the code by clicking the green play button (where it says "Lab1A").



- 7) Note where it says "class Program". For the labs below, you will need to change this to "class Lab1A", "class Lab1B" and "class Lab1C"  
8) Note the "Console.WriteLine". This is how you print to the screen. Also note that it's WriteLine, not Writeline or writeline. That is, C# is a case-sensitive language.  
9) Remember you must save and recompiled the code **every** time you change it before being able to run it.

## Directions

For this lab, you're going to work with your IDE to compile and run some programs we have provided you. By the end of this lab, you will have three files that you need to submit to the autograder at the same time – Lab1A.cs, Lab1B.cs and Lab1C.cs. Note: For each one of these, you will create a New Project in Visual Studio, then create a new .cs file within that project. Don't forget to change the name of the classes.

Before you begin the next part of this lab, you might start feeling overwhelmed when you first look at the source code below. There's a lot of strange symbols there! You'll get comfortable with these, but here's a quick explanation:

- "main" is the starting point of the program. The program execution then goes line by line.
- Semicolons ; are used at the end of a single statement. It's similar to a period at the end of a sentence. See below for examples.
- Opening and closing curly braces { } contain zero or more single statements inside of them. They technically mean "begin" and "end" a block of code that should be considered as "one thing". That's a sloppy definition, but it will make more sense later.
- Comments are denoted one of a few ways. The /\* and \*/ have comments in between. This is good when you have multiple lines of comments. However, if you have only one or two lines of comments, you can use the double-slash // notation. You can see this below as well.

Three programs have three problems with them:

- One of the programs below has a logic error in it – meaning that, while it compiles and runs, it doesn't produce the correct output. You must find the error and fix it.
- One of the programs below doesn't compile, so you can't even run it! Based on what we just told you, you must figure out why, fix the error, save the file, and re-run the code.
- One of the programs compiles and runs fine – but has a typo in the output. If you were to submit this file, the autograder won't like it (because the output does not match). You may try submitting the lab before fixing the error so you can see what the autograder does.

When creating the files for the programs below, they must be called Lab1A, Lab1B, and Lab1C (and have the appropriate .java, .cs or .cpp file extension).

Fix, compile and run the programs below, then submit them in Gradescope.

```
===== Program Lab1A.cs =====
using System;

// Program Lab1A.cs
// Demonstrate the difference between Write and WriteLine methods.

public class Lab1A
{
    // Prints two lines of output representing a rocket countdown.
    public static void Main (String[] args)
    {
        Console.Write ("Three... ");
        Console.Write ("Two... ");
        Console.Write ("One... ")
        Console.Write ("Zero... ");
        Console.WriteLine ("Liftoff!"); // appears on first line
        Console.WriteLine ("Houston, we have a problem.");
    }
}
```

```

===== Program Lab1B.cs =====
using System;

// Program Lab1B.cs
// Demonstrate reading a string from the user.

public class Lab1B
{
    // Reads a character string from the user and prints it.
    public static void Main (String[] args)
    {
        String message;
        Console.Write("Enter a line of text: ");
        message = Console.ReadLine();
        Console.WriteLine("You wrote'" + message + "'");
    }
}

```

```

===== Program Lab1C.cs =====
using System;

// Program Lab1C.cs
public class Lab1C
{
    // Calculates fuel efficiency based on values entered by the user.
    public static void Main (String[] args)
    {
        int miles;
        double gallons, mpg;
        Console.Write ("Enter the number of miles: ");
        miles = Convert.ToInt32(Console.ReadLine());

        Console.Write ("Enter the gallons of fuel used: ");
        gallons = Convert.ToInt32(Console.ReadLine());

        mpg = miles * gallons;
        Console.WriteLine ("Miles Per Gallon: " + mpg);
    }
}

```

**Save all programs in one folder (call it Lab1) for future reference. You may also save this on a USB device or OneDrive.**

**Instructions for submitting:**

1. All practice programs must compile and work correctly.
2. The programs must be checked by the end of the designated lab session by Lab Instructor or TA. Raise your hand and someone will help you. If you are online, you will submit your files in Gradescope under Lab1.
3. All three source code files (files with a .cs extension) must be uploaded to Gradescope by due date. Do NOT upload other files like the .sln file.
4. If you have done things correctly, you will receive a grade of 105 for this lab, since the autograder likes it when things work correctly/cleaning. We've tried asking the autograder not to do this, but she will not change her mind.