CSCI 1301 - Lab 03

Clément Aubert

January 29, 2018

Deadlines: All three labs need to be completed before taking Lab 04.

Dependencies: The recommended order is Part 0 ⟶ lab 02 (if needed) ⟶ Part I ⟶ Part II (optional).

Part 0 - Getting Organized (Bis)

I apologize for the unfortunate experience of discovering the files you set up during lab 01 being removed. On top of that, I uploaded for some time the wrong example file `Welcome1.zip`. Fortunately, we can fix that easily.

Find a place to do your backup

Since you can not store files permanently on the lab's computer, you will have to store them either

- On an external / removable data storage: USB flash drive, External hard disk drive, or any kind of USB mass storage device, or
- On a server: the University has a partnership with box.com, and you can follow this tutorial to get started, but any service (Google Drive, Dropbox, OneDrive, etc.) would do.

If you chose the "remote" option (i.e., using a server), **do not** install a synchronization software (like Google Drive and Sync, Box's app, etc.) on the lab computer: it will likely not work, due to University rules. 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 unloging from the computer.

If you can't / don't want to find a solution during the lab, you can skip to Part I, send yourself the files over email at the end of this lab, and come back when you can.

Make 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 each of the first three labs. Re-download `Welcome1.zip`. You organization should look like the following:

```
└──csci1301
    ├── 01_lab
    ├── 02_lab
    │   └── Welcome1.zip
    └── 03_lab
```

Extract the archive `Welcome1.zip` and suppress it.

Resume

Complete lab 02 if you did not already. Don't forget to save your lab on your USB key or on a server once you're done.

Part I - Your First Own Project

Starting from a template

We will first create a new project for Visual C# using the template for "Console App (.NET Framework)".

a. Create a new project, using Ctrl + Shift + N or "File" → "New" → "Project"
b. Find the "Console Application Visual C#" (a.k.a. "Console App (.NET Framework)") template, by using Ctrl + E and then typing "Console", or by navigating using the menu on the left panel: "Templates" → "Installed" → "Visual C#" → "Windows" → "Classic Desktop" (or "Installed" → "Visual C#" → "Classic Desktop"). Effectively create the project only after you completed the next step.
c. Enter "MyFirstProject" as the name of the project, select the right location (for instance in the folder you created for lab 03), and enter "MyFirstSolution" as the name of the solution.
d. Now, answer the following:

- A source code appeared in the main window of VS. Compare this code with the code you studied in lab 02 (`Welcome1.cs`): how are they different? How are they the same?
- Open a file explorer and navigate to the place where you stored your project. Compare the files with the files from the `Welcome1` project: how are they different? How are they the same?
- Try to compile this code, using Ctrl + Shift + B or `Build` → `Build solution`. Did the compilation succeed?
- Execute the code, using Ctrl + F5 or `Debug` → `Start without Debugging`. What happened? Compare with what was happening with the `Welcome1.cs` project.

Editing the template

Now, you will start writing your own code. We'll start by writing a very familiar instruction to print a message.

a. Place the cursor inside the `Main` method (i.e., after the brace after `static void Main(string[] args)`).

b. Type `Console`. The (at first sight annoying) auto-completion feature that display suggestions and messages as soon as you start typing is called Intellisense. You can read about it at https://msdn.microsoft.com/en-us/library/hcw1s69b.aspx or https://docs.microsoft.com/en-us/visualstudio/ide/using-intellisense, you'll probably ending up using it a lot, but let's not worry about that for now.
c. Type in `.Wri` after `Console` (don't forget the period!) and notice that Intellisense is already making good suggestions: you actually want to write `WriteLine`! Either finish writing `WriteLine` or select it from the menu that appeared.
d. Now, open a parenthesis, i.e., type `(`, and notice that Intellisense closed it for you, and is already displaying another message.
e. Type the string of your choice between those two parenthesis, i.e., something like `"This is my first message"` (and don't forget the quotes).

At this point, your `Main` method should look like this:

```
static void Main(string[] args)
{
   Console.WriteLine("This is my first message!")
}
```

f. Compile (= "build") your file. Oh, no, something went wrong! Can you fix this problem?
g. Once you can compile your program without error, execute (= "run without debugging") it.

Discovering methods and escape sequences

a. Add the following line after the one you just typed in:

```
Console.Write("This is second first message!");
```

b. Compile and run your program. Can you notice the difference between `WriteLine` and `Write`?
c. We can also use *escape sequences* (cf. https://docs.microsoft.com/en-us/cpp/c-language/escape-sequences) to represent various character. For instance, add the following to your program:

```
Console.Write("\n Here \n \t is \n \t \t another message.");
```

d. Compile and run your program. What is the purpose of `\n`? What is the purpose of `\t`?
e. Look at the link mentioned in c., and write a statement that prints the `\` and the `"` characters, as well as two different forms of spacing.

Part II (Optional) - Pushing Further

The following are two independent tasks, to widen your understanding of this class, and to prepare you for the next labs.

a. You may have noticed (depending on the VS version you're using, and its configuration), during Part I, the existence of a template named "Console App (.NET Core)". Using the following two ressources,

sketch what the differences between these two frameworks are: https://docs.microsoft.com/en-us/dotnet/standard/choosing-core-framework-server, https://stackoverflow.com/q/38063837/.

b. At https://docs.microsoft.com/en-us/cpp/c-language/escape-sequences, it is mentionned that you can also print Unicode characters using escape sequences. Try to print some: look for codes at https://en.wikipedia.org/wiki/List_of_Unicode_characters (you need to keep only the part after `U+`). You may have to add

```
Console.OutputEncoding = System.Text.Encoding.UTF8;
```

in your main method for the printing to take place. I recommend *typing* this statement instead of copy / pasting it, to have a chance to read Intellisense's descriptions.