

# CSCI 1301 – Lab 06

August 26, 2019

## 1 Variables Types: From String to Integer

1. Create a new project.
2. Write two statements, one that declares a variable of type `int` named `intVar`, one that declares a variable of type `string` named `stringVar`.
3. Assign the value 3 to `intVar`, and "4" to `stringVar`.
4. Display the values of `intVar` and `stringVar`.
5. Write a statement that assign the value of `stringVar` to `intVar`. Why is the compiler complaining? Comment out the statement you just added (that is, add a `//` in front of it, so that the compiler won't try to execute it).
6. Copy the following statement, to "convert" the string value of `stringVar` into an integer value, and assign it to `intVar`:  
  

```
intVar = int.Parse(stringVar);
```
7. Using <https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/types/how-to-convert-a-string-to-a-number>, try to understand what just happened.
8. Change the value of `stringVar` to be "Train" and assign it to `intVar` using `int.Parse` as previously shown. What happened?

## 2 Reading Other Numeric Datatypes

1. Create a new project.
2. Declare three variables: one of type `int`, one of type `float`, and one of type `double`.
3. We can use the following to read an `int` from the user (assuming your `int` variable is named `intVar`):  
  

```
intVar = int.Parse(Console.ReadLine());
```

  
Similarly, we can read `float` using `float.Parse(Console.ReadLine())`, and `double` using `double.Parse(Console.ReadLine())`.
4. Write statements that ask the user to enter an `int`, a `float` and a `double`, store those values in the appropriate variables, and then display them at the screen.

## 3 Cast Operator

Create a new project, and then do the following.

1. Add in your program the following:

```
float floatVar = 4.3f;  
int intVar = floatVar; // This statement will give you an error
```

You will get an error that reads

```
Cannot implicitly convert type 'float' to 'int'. An explicit conversion  
↪ exists (are you missing a cast?)
```

Can you explain it?

2. VS is suggesting that we use a “cast” to “force” C# to store the value of the variable floatVar into the variable intVar. To do so, replace the previous statement with the following:

```
int intVar = (int)floatVar; // This statement will compile
```

3. Using a Console.WriteLine statement, observe the value stored in intVar. Can you tell if the value stored in floatVar was rounded or truncated before being stored in the variable intVar? Conduct further experiments if needed to answer this question.

## 4 Milestone #1

You accomplished a lot since you started this class. Let’s take a moment to look back at what you learned, and to make sure you understand all those notions and skills. If you have a doubt, feel free to look back at the corresponding lab.

### 4.1 What You Learned

#### 4.1.1 To Use Visual Studio

- What is the structure of a project,
- How to open a pre-existing project,
- How to open a template and edit it,
- How to re-name a project or its files,
- To avoid modifying the files of a project outside of VS,
- How to compile and execute your program,
- How to zip and share a project,
- How to use the “Find and Replace” feature of VS.

### 4.1.2 To Be Methodical

- Not to be afraid of the documentation,
- To use the following shortcuts (remember that there is a cheatsheet at <http://visualstudioshortcuts.com/>):
  - ALT + F4 to close VS (or any Windows program, actually)
  - Ctrl + Shift + b to build the solution
  - Ctrl + F5 to start the program without debugging
  - Ctrl + Shift + n to create a new project
  - Ctrl + Shift + F12 to “jump” to the line where VS thinks there is an error
  - Ctrl + Shift + f to open the “Find and Replace” menu
  - CTRL + k, then CTRL + d to indent your code
- To compile your code frequently,
- To read the error messages,
- To organize your files, to perform regular back-ups, and to make sure that those back-ups were correctly done.

### 4.1.3 To Write Programs!

- What are the important parts of a program,
- To display messages at the screen, using `Write` and `WriteLine`,
- To display messages at the screen that includes escape sequences,
- To declare, assign and display at the screen variables,
- To read a string from the user,
- To convert a string into integer,
- To manipulate various numeric datatypes,
  - To choose the appropriate numeric datatype
  - To determine the “legality” of an operation involving two different datatypes
  - To use the cast operator to explicitly convert between datatypes
  - To be able to identify the datatype of literals
- To read numeric values from the user.

### 4.1.4 Academic Life

- Not to be afraid of your professor (hopefully!).
- How, where, and when to ask for help.
- What are my expectations for this class.
- How to organize your workflow, the importance of planning ahead.
- ... to be continued!

Maybe you decided what your major was going to be. Maybe you changed your mind. Maybe you're not sure. Being confused and uncertain is sometimes part of the process of taking decisions and learning, and that's all right. It is normal to hesitate. This page by a colleague in Computer Science<sup>1</sup> may be a good read for students hesitating between IT, CS and MIS, or wanting to have more information about those majors.

---

<sup>1</sup><http://spots.gru.edu/mdowell/Right%20major.html>

## 5 Working on Problem-Solving

Here is a problem that involves almost all the previously mentioned notions and skills. It is phrased in a more abstract way, closer to the kind of problem you will be facing if you were a software-developer. Think about what you need to do before starting to type your code, and when you start writing your code, make sure you compile it frequently.

Write a program that asks the user for their name, their number of guests, and the number of pizza slices they have. Your program should then display the name of the user, the number of people at the party, the number of slices, and how many slices will each guest have, assuming the slices will be shared equitably.

A couple of additional precisions:

- The user of the program also want some pizza, so the number of people at the party is the number of guests plus one.
- Once you're done, you need to test your program:
  - What happen if the user provide “normal”, plausible data? Does your program work as expected?
  - What happen if the user have 2 guests and 4 slices? Will everyone get  $4 / 3 =$  one slice and a third, as they normally should?
  - What happen if the user have 0 slice?
  - What happen if the user have 0 guest?
  - What happen if the user have -4 slicess?
  - What happen if the user have -1 guest?
- If you want, you can change your program so that it displays the number of slices per guest *without fraction*, and the number of remaining slices. For instance, a user entering 4 guests and 16 slices should read that every member of the party will have  $\lfloor 16 / (4 + 1) \rfloor = 3$  slices, and that  $16 \bmod (4 + 1) = 1$  slice will be left.