CSCI 1301 – Lab 05

August 30, 2018

1 Reading From the User

1. Download the PersonalizedWelcomeMessage[1] project.
2. Extract it, and open it in VS.
3. Compile and execute it.
4. You will be prompted with the message

   > Please, enter your first name, followed by "Enter":

   Enter your first name, followed by Enter . You just witnessed an interaction between a program and an user!
5. Read the source code, and make sure you understand all of it.
6. Change the code, so that the program would also ask the user's last name, and print both their first and last name.

2 Numeric Datatypes

For this part, I recommend opening the web page[2], printable version[3] or editable version[4] of the document we just studied in class, if you don't have its physical version. Note that it contains numerous references at its end.

2.1 Experimenting

Compute in your head the result of the following operation: $1000000.0 + 1.2 - 1000000.0$.

Now, implement it (read as "Create a new project and copy that code in the `Main` method") using `float`, `double`, and `decimal`:

```
Console.Write("With floats:\n\t");
Console.WriteLine(1000000.0f + 1.2f - 1000000.0f);
Console.Write("With double:\n\t");
Console.WriteLine(1000000.0 + 1.2 - 1000000.0);
Console.Write("With decimal:\n\t");
Console.WriteLine(1000000.0m + 1.2m - 1000000.0m);
```

Can you explain what you just observed?

Now, execute the following code:

[1] PersonalizedWelcomeMessage.zip
[2] ../../shared/datatypes/
[3] ../../shared/datatypes/index.pdf
[4] ../../shared/datatypes/index.docx

```
decimal decVar = 12344321.4999999991M;
double douVar = (double)decVar;
float floVar = (float)douVar;
Console.WriteLine($"With decimal: {decVar} \nWith double: {douVar} \nWith float:
 ↪  {floVar}");
```

Can you explain the gradual loss of precision?

2.2 Making Simple Calculations

This part should be first carried out without using VS.

Assume we have the following statements:

```
int a = 21, b = 4;
float f = 2.5000000f;
double d = -1.3;
decimal m = 2.5m;
```

Answer the following:

- How many variables are declared?
- What are their datatype?
- What are their values?
- What are their names?
- Consider the following expressions. For each of them, tell if they are legal, and if so, give the result and its corresponding datatype. The first two are given as examples.

Operation	Legal?	Result	Datatype
`a + d`	Yes	19.7	`double`
`m + f`	No	_	_
`a / b`	_	_	_
`b * f`	_	_	_
`d + f`	_	_	_
`d + b`	_	_	_
`a + m`	_	_	_
`f / m`	_	_	_
`d * m`	_	_	_

You can check your answers using VS: create a new project, copy the variables declarations and assignments, and write your own statements to perform the calculations in the `Main` method. For instance, if you want to check that the result of `a + d` is of type `double`, write something like

```
double tempVariable1 = a + d;
Console.WriteLine($"The value of d+f is {tempVariable1}");
int tempVariable2 = a + d; // This line should give you an error.
```

2.3 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.