Contents

Variable Manipulation

This lab serves multiple goals:

- To help you develop good lab habits,
- To declare, assign and display variables of different datatypes,
- To understand how to use arithmetic operators,
- To experience the differences in numerical datatypes,
- To use combined assignment operators.

Variable Manipulation

Developing good lab habits

While working through these labs, you should be putting previously learned material to use and developing good habits. You should know how to:

1. Manipulate projects (creating them, opening them, editing them, saving them, making backups of them, etc.).
2. Compile and execute your program frequently.

You have probably already started to read "compile and execute" as:

Windows/Linux:
- Compile (build solution): Ctrl + Shift + B
- Execute: Ctrl + F5

MacOS:
- Compile (build solution): Command + B
- Execute: F5 -or- Command + return

From now on, read "Create a new project" as

- Create a new project using the "Console Application" template for C# (remember that this is explained in more details in the "Your first program" lab[1].

[1] https:/princomp.github.io/labs/FirstProgram#creating-your-first-new-project

- Pick simple and valid names for your project and solution, and save it in a suitable place (ex. a folder with a name that matches the name of the lab).
- Compile and execute your program frequently (ex. upon completion of every lab task).
- *Do not hesitate to change your program to answer questions; you're in a lab, you're supposed to conduct experiments!*
- If you get an error when trying to compile your program, go to the line where your IDE thinks there is an error.

And, *even if it is not explicitly stated*, you should save your work once you are done and reopen it at the beginning of the next session to verify that you saved it properly.

If you want a UCA or your instructor to check your answer to a previous lab, just ask!

First Variable Manipulations

1. Create a new project as indicated above.
2. In the `Main` method, add three statements:
   - One that declares a `string` variable named `lastName`,
   - One that declares a `string` variable named `firstName`, and
   - One that declares an `int` variable named `classOf`.
3. Below those three statements, write statements that assign your last name to the first variable, your first name to the second variable, and your anticipated graduation year (ex. 2025) to the third variable.
4. Display the values of the three variables on the screen using the following statement:

   ```
   Console.WriteLine($"My name is {firstName}
    ↪  {lastName}, and I expect to graduate in
    ↪  {classOf}.");
   ```

5. Compile and execute your program. It should display a message like:

   ```
   My name is Evi Nemeth, and I expect to graduate in
    ↪  2025.
   ```

6. Answer the following questions by first thinking of what may happen and then by editing your program to check your hypotheses. Would the compilation still be successful if we were to try to:

   (a) Assign a value to a variable before declaring it?

(b) Display the content of a variable before assigning a value to it?
(c) Assign a `string` value to an `int` variable?
(d) Assign an `int` value to a `string` variable?

7. At the end of your `Main` method:

   (a) Add three statements that change the values of the three variables.
   (b) Copy the `Console.WriteLine` statement that was previously given.
   (c) Compile and execute the program.

   Notice that the very same statement will now display a different message on the screen!

Experimenting With Numeric Datatypes

Mentally compute the result of the following operation: `1000000.0 + 1.2 - 1000000.0`.

Now, implement it

> (Read "implement it" as: Create a new project and, in the `Main` method, add the code below to display the result of this computation, as computed by C#, on the screen.)

using the `float`, `double`, and `decimal` datatypes:

```
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);
```

Compile and execute your program. Can you explain what you just observed?

Now, add and execute the following code:

```
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?

Variable Assignments

For this problem, it is recommended that you attempt to manually compute this with a pen and a sheet of paper for some time before opening your IDE. Consider the following code:

```
int a = 0;
a = 12 - 3;
a = 10 % 3;
a = 12 * 2;
a = 9 / 3;
a = -3;
a -= 3; // This is the same as a = a - 3;
a += 5; // This is the same as a = a + 5;
a *=12; // This is the same as a = a * 12;
a /= 2; // This is the same as a = a / 2;
a = a + a;
```

1. In the previous code, what is the value of `a` after each line is executed?
2. Check your answers. Create a new project, copy the previous statements in the `Main` method, and use `Console.WriteLine(a);` to display the value of `a` after each statement.