

PCP — Lecture 03

Fall 2020 August 25, 2020

Last Time - First Program

- Presentation of the difference between machine, assembly, and high-level languages.
- Some of C#'s specificities.
- The difference between rules and conventions, identifiers and keywords.
- Escape sequences and the difference between `Console.Write` and `Console.WriteLine`.

1 Datatypes Nomenclature

- Value types
 - Numeric
 - * Signed integer (`sbyte`, `short`, *`int`*, `long`)
 - * Unsigned integer (`byte`, `ushort`, `uint`, `ulong`)
 - * Real number (`float`, *`double`*, `decimal`)
 - Logical (*`bool`*)
 - Character (*`char`*)
- Reference types
 - String (*`string`*)
 - Object (`object`)

(In italics, the one we will mainly be using.)

Integers are “whole” numbers ($\mathbb{Z} = \{\dots, -1, 0, 1, 2, 3, \dots\}$), floating point numbers are real numbers (\mathbb{R}), strings are “text messages”, ...

Please refer to the “[Datatypes in C#](#)” [cheatsheet](#) for more information about datatypes.

2 String and Int Variables

Literals are fixed values (“Hi Mom”, 40, 1.2404, ...) in the source code.

```
1 using System;
2 class MyFirstVariables{
3     static void Main(){
4         // Declaration
5         int myAge;
6         string myName;
7         // Assignemnt
8         myAge = 40;
9         myName = "Clément";
```

```

10     // Displaying
11     Console.WriteLine($"I am {myAge} old and my name is {myName}.");
12 }
13 }

```

A variable has a *name* (which must be an identifier), a *type*, a *size*, and a *value*.

Variable Name	Variable Type	Variable Size	Variable Value
myAge	int	32 bit	40
myName	string	Variable ¹	"Clément"

2.1 Variable Initialization

You can declare and assign a variable in one statement using what is called an “initialization statement”.

```

string myMessage = "Hey Mom";
int myValue = 12;

```

There is now one additional rule when it comes to choosing a valid identifier for your variable name: you can not take an identifier that was already used. That is, you can have only one variable named `myMessage`: if you want to re-assign a variable, you can not use an initialization again (that would re-declare the variable), you need to use an assignment statement again.

2.2 Remarks

- The value can change (hence the name!) if you re-assign it. The previously stored value is simply wiped out, and lost.
- You can store one variable’s value into another, but that value in the other variable won’t change when the original variable’s value changes:

```

int a = 12;
int b = a; // b's value is 12
a = 0; // a's value changed to 0, but b's value is still 12.

```

- We can perform basic math operations with numeric datatypes: + (sum), * (multiplication), - (subtraction) but also the modulo operation (%), which corresponds to the remainder. More details will be given in lab #3, in homework #2, and during lecture #4.
- There is a difference between

```

1 int sum = num * 2; // The value of sum is num's value times two
2 Console.WriteLine($"{sum}"); // The value of sum is displayed.

```

and

```

1 Console.WriteLine($"{num * 2}"); // The value of num times two is displayed, but the
  ↳ value of num is still the same.

```

¹It’s actually $20 + (n/2) * 4$ bytes, for n the number of characters in the string, so 7 in that case.

- You can combine multiple declarations, initializations, and even mix both in one statement:

```
1  int a=0, b, c;  // a, b, c are declared as three int variables, and a's value is set  
    ↪ to 0.
```