Variables - theory

You are finally ready to start programming, exicted eh? Well, the whole programming thing is based on variables, variables are like boxes in which you can put datas; in c# a variables has 3 important properties:

- a name, the name of a variables will never change and you will use it to recall a variables. You can think of it as a label sticked on the box
- a type, the type defines which kind of value the variable will store, ie. numbers, words, lists,... In our box metaphor the type is the shape of the box, you can't fir a square in a triangular box, right? Same goes for type you can't fit a word in a numeric variable
- a value, it's the value inside the box, you can change it whenever you want, given that you respect the type limitations

The types we are going to need the most in this first part of the guide are:

Type	Meaning	example
bool	It's a data type with only two possible values: *true* or *false*	true
char	It can contain only a single character	I
string	It's a sequence of character as long as u like	This is a string value
int	Whole numbers	-2
float	Real numbers	0.355f
double	It's a float that allows for bigger numbers	

At this point you may argue "Why should I use a variable?", well the cool part of variables is that once you created them you can just write the name of the variable and the program will understand that you want him to replace that name with the actual value of the variable.
ie. you want to create a program that given 2 initial numbers sums them and writes the result.
In pseudocode that would look like this:

```
int x = input1
int y = input 2
int result = x + y
write result
```

Pseudocode mean that it's not following the correct syntax of the language, programmers often use it to explain how to do something in a not-language-related way.

Variables – practice

To use a variable you have to declare it (create the box) and initialize it (put the first value into the box).
Declaring syntax: type name;
ie. `int x;`

```
int y;
char carattere;
bool raining;
```

Initializing syntax: name = value ;
ie.
```
x = -5;
y = 3;
carattere = "t";
raining = false;
```

We can do both the declaring and initializing in just one line like this:
```
int x = -5;
```
or we can use a variable as value for the initializing
```
int x = y;
```

After this first 2 steps the variable is ready. So we will write our first program: you should have already created a new project from the previous chapters, if not go back and read them.
Visual studio already prepared for you many lines of code, at the moment you don't need to know what those lines do, the important part is that all the code we are going to write must be placed beetween the { } after *static void Main(string[] args)*

Our program will sum 2 integers and write the result, the code is this:
```
int x;
int y;
x = 5;
y = 8;
int result = x + y;
Console.WriteLine(result);
Console.ReadLine();
// the end
```
Now lets try it, by clicking F5 or the green Start button. The program will start, a console will pop up and we'll see that the program wrote 13.

Now let's dig into the code and understand how it works:

- every line end with a semicolon, this is a rule of C# that you have to follow always
- the first 2 lines are declarations
- the 3-4 lines are initialization
- the line 5 is declares and initialize result
- in the line 5 as value we use x + y, c# can handle this, in fact we can use any basic mathematic expression as value: x * y, x – y – 5, y / 8 are all acceptable values
- in the line 6 there is this new syntax we never saw, it's called a method and we'll talk about that later, but basically it takes some input and does something, in

this case it writes the input in the console

- line 7 is another method, it doesn't require input and it asks the user to type a value. We used this, because when the program reaches the end of the code it shuts itself down, so we added this method that doesn't end until the user press enter
- the last line is a comment, it's good practice to use those a lot, so that you and other people can understand the code when reading it in the future. To comment something just write `// comment text 123` for a single line comment, you can also comment after a statement `x=5 // x is now 5` or make multiline comments

```
x=5 /* x
is
now
5 */
```

Variables advanced

Here are a bunch of useful syntaxes for numeric types:

x=y + 3;
x=3*6;
y=8/x;
x=(y+5)*(8-3)
	As we already said basic mathematical expressions are allowed

x++; x--;	It's a quick way to add or subtract 1

x = 0;
x+=3;
x-=4;
x*=-2;
	The same as:
	x=x+3;
	x=x-4;
	x=x*(-2);

x = 22 % 8	The % is the modulus/remainder sign

It's useful to know that you can sum strings, they will just snap together:

```
string a = "abc";
string b = "def";
string c = a+b;   // c will evaluate to "abcdef"
```