VARIABLES

THEORY

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

* **Name:** the name of a variables will never change and you’ll use it to recall a variable. You can think of it as a label stuck on the outside of the box;
* **Type:** the type defines which kind of value the variable will store (e.g. numbers, words, lists, etc.). In our box metaphor the type is the shape of the box, you can't fit a square in a triangular box, right? Same goes for the type: you can't fit a word in a numeric variable;
* **Value:** it's the value inside the box, you can change it whenever you want, given that you respect the type limitations. It’s like a piece of paper inside the box that you can read and substitute whenever you want.

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 | A |
| **String** | It's a sequence of character as long as you like | Hello World |
| **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.

To better explain this I’ll use an example: you want to create a program that given 2 initial numbers sums them and writes the result.

In pseudocode that would look like this:

**What does “pseudocode” mean?**

It’s a codewithout being bound to a specific language.

int Cypher0 = 2

int Cypher1 = 3

int Result = Cypher0 + Cypher1

write Result

As you can see, the program treats the variables as they were the number they represented.  
As a rule of thumb the pc will replace the variable’s name with his value in every part of the code, but initialization and declaration of that variable (when the variable is on the left of the =).

So when the pc reads the third row of our code it reasons like this:

int Result = Cypher0 + Cypher1

int Result = 2 + 3

int Result = 5

Also, if you were wondering why I start counting from 0, don’t worry, it’s a common thing in programming, you’ll see this frequently.

**PRACTICE**

To use a variable, you have to declare it (create the box) and initialize it (put the first piece of paper into the box).

Declaring syntax: type name;

e.g. int x;

int y;

char character;

bool raining;

Initializing syntax: name = value;

e.g. x = -5;

y = 3;

character = "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 between 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 let’s 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
* the first 2 lines are declarations
* the 3-4 lines are initialization
* the line 5 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 \*/

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”