Variables

A variable can be compared to a storage room, and is essential for the programmer. In C#, a variable is declared like this:

<data type> <name>;

An example could look like this:

string name;

That's the most basic version, but the variable doesn't yet have a value. You can assign one at a later point or at the same time as declaring it, like this:

<data type> <name> = <value>;

If this variable is not local to the method you're currently working in (e.g. a class member variable), you might want to assign a visibility to the variable:

<visibility> <data type> <name> = <value>;

And a complete example:

private string name = "John Doe";

The visibility part is related to classes.

\*An object is an instance (creation) of a class (classes are blueprints for objects).

\*\*Objects will be discussed in more detail later.

Data types (value and reference types)

Data types are used everywhere in a programming language like C#. Because it's a strongly typed language, you are required to inform the compiler about which data types you wish to use every time you declare a variable, as you will see in the chapter about variables. In this chapter we will take a look at some of the most used data types and how they work.

**bool** is one of the simplest data types. It can contain only 2 values - false or true. The bool type is important to understand when using logical operators like the if statement.

**int** is short for integer, a data type for storing numbers without decimals. When working with numbers, int is the most commonly used data type. Integers have several data types within C#, depending on the size of the number they are supposed to store.

**string** is used for storing text, that is, a number of chars. In C#, strings are immutable, which means that strings are never changed after they have been created. When using methods which changes a string, the actual string is not changed - a new string is returned instead.

A string simply contains text, as you can see, since we give them a value straight away. Next, we output a line of text to the console, where we use the two variables. The string is made up by using the + characters to "collect" the different parts.

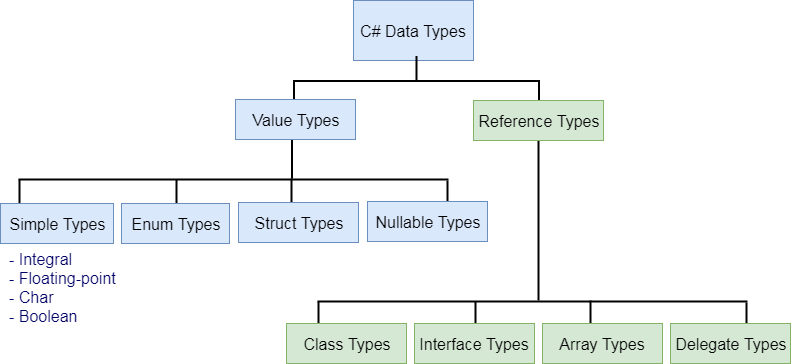
**char** is used for storing a single character.

**float** is one of the data types used to store numbers which can contain decimals.

**More on C# Types:**

|  |  |  |
| --- | --- | --- |
| byte | 8-bit unsigned integer | 0 to 255 |
| sbyte | 8-bit signed integer | -128 to 127 |
| short | 16-bit signed integer | -32,768 to 32,767 |
| ushort | 16-bit unsigned integer | 0 to 65,535 |
| int | 32-bit signed integer | -2,147,483,648 to 2,147,483,647 |
| uint | 32-bit unsigned integer | 0 to 4,294,967,295 |
| long | 64-bit signed integer | -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 |
| ulong | 64-bit unsigned integer | 0 to 18,446,744,073,709,551,615 |
| float | 32-bit Single-precision floating point type | -3.402823e38 to 3.402823e38 |
| double | 64-bit double-precision floating point type | -1.79769313486232e308 to 1.79769313486232e308 |
| decimal | 128-bit decimal type for financial and monetary calculations | (+or-)1.0 x 10e-28 to 7.9 x 10e28 |
| char | 16-bit single Unicode character | Any valid character, e.g. a,\*, \x0058 (hex), or\u0058 (Unicode) |
| bool | 8-bit logical true/false value | True or False |
| object | Base type of all other types. |  |
| string | A sequence of Unicode characters |  |
| DateTime | Represents date and time | 0:00:00am 1/1/01 to 11:59:59pm 12/31/9999 |

**Value Vs Reference Types:**



**Value types:**

**Enums**

**Structs**

**Simple types**

**Int**

**Bool**

**Char**

**Float**

**\*\*More on reference types (objects) later.**

**Variables & scope**

So far, we have only used local variables, which are variables defined and used within the same method. In C#, a variable defined inside a method can't be used outside of this method - that's why it's called local. If you're familiar with other programming languages, you may also know about global variables, which can be accessed from more places, but C# doesn't support the concept of global variables. Instead, you can define a field on a class, which can be accessed from all the methods of this class.

The concept of differentiating between where a variable has been declared is called **scoping** and it prevents your code from becoming a huge mess of variables which can be changed from too many places. Another technique that helps us with this is called member *visibility* (in this case illustrated with the private keyword), which we'll discuss in the chapter about classes.