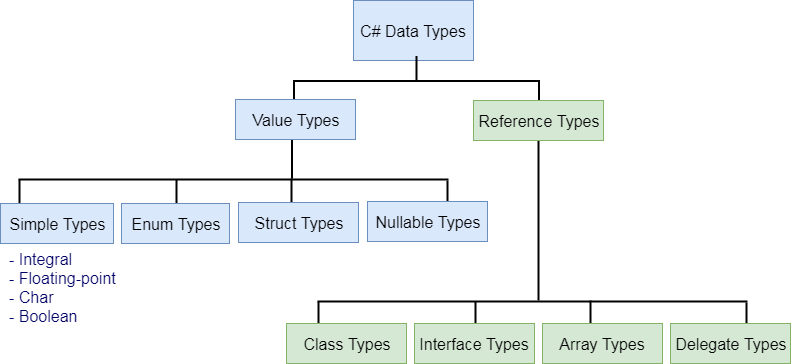
**More on C# Types:**

|  |  |  |
| --- | --- | --- |
| 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 types:**

**Enums**

**Structs**

**Simple types**

**Int**

**Bool**

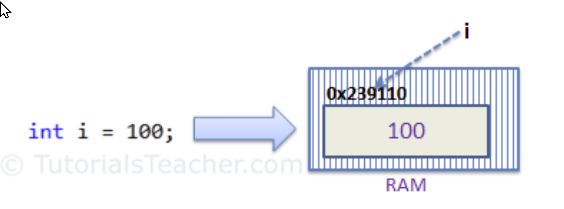
**Char**

**Float**

**Value Types vs Reference Types:**

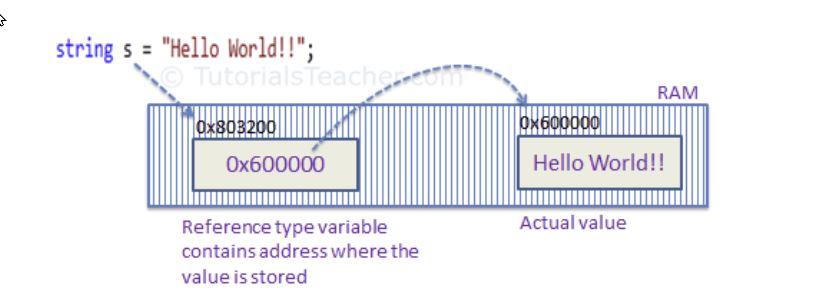
Value Type:

A data type is a value type if it holds a data value within its own memory space. It means variables of these data types directly contain their values.



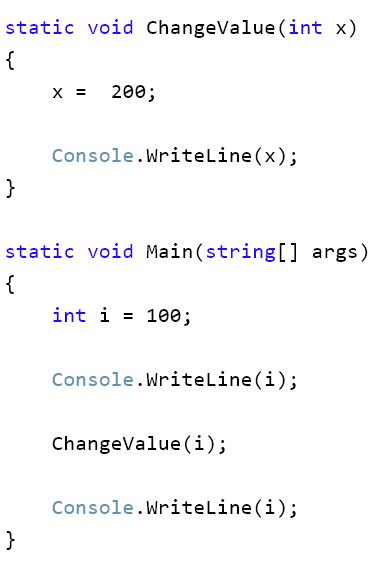
Reference Type:

Unlike value types, a reference type doesn't store its value directly. Instead, it stores the address where the value is being stored. In other words, a reference type contains a pointer to another memory location that holds the data.



**Passing By Reference or Passing By Value**

When you pass a value type variable from one method to another method, the system creates a separate copy of a variable in another method, so that if value got changed in the one method won't affect on the variable in another method.



//100

//200

//100 (see how variable “i” didn’t get changed to 200).

When you pass by reference, you are sending the only copy of that reference (essentially an object pointer) variable.

**Stacks Vs Heaps:**

A Stack

