Value Type and Reference Type

We have learned about the data types in the previous section. In C#, these data types are categorized based on how they store their value in the memory. C# includes following categories of data types:

1. Value type
2. Reference type

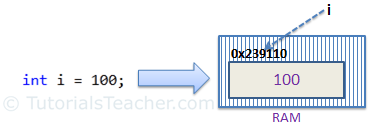
Value Type:

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.

tipAll the value types derive from *System.ValueType*, which in-turn, derives from *System.Object*.

For example, consider integer variable int i = 100;

The system stores 100 in the memory space allocated for the variable 'i'. The following image illustrates how 100 is stored at some hypothetical location in the memory (0x239110) for 'i':

[](https://www.tutorialsteacher.com/Content/images/csharp/value-type-memory-allocation.png)Memory allocation for Value Type

The following data types are all of value type:

* bool
* byte
* char
* decimal
* double
* enum
* float
* int
* long
* sbyte
* short
* struct
* uint
* ulong
* ushort

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.

Example: Value Type

static void ChangeValue(int x)

{

x = 200;

Console.WriteLine(x);

}

static void Main(string[] args)

{

int i = 100;

Console.WriteLine(i);

ChangeValue(i);

Console.WriteLine(i);

}

[Try it](https://www.tutorialsteacher.com/codeeditor?cid=cs-ba76t1)

Output:

100   
200   
100

In the above example, variable i in Main() method remains unchanged even after we pass it to the ChangeValue() method and change it's value there.

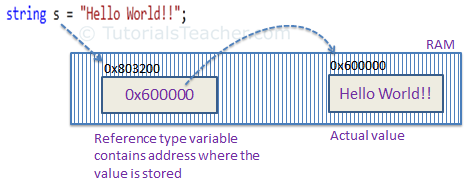
Reference Type

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.

For example, consider following string variable:

string s = "Hello World!!";

The following image shows how the system allocates the memory for the above string variable.

[](https://www.tutorialsteacher.com/Content/images/csharp/raference-type-memory-allocation.png)Memory allocation for Reference type

As you can see in the above image, the system selects a random location in memory (0x803200) for the variable 's'. The value of a variable s is 0x600000 which is the memory address of the actual data value. Thus, reference type stores the address of the location where the actual value is stored instead of value itself.

The following data types are of reference type:

* String
* All arrays, even if their elements are value types
* Class
* Delegates

Pass by Reference

When you pass a reference type variable from one method to another, it doesn't create a new copy; instead, it passes the address of the variable. If we now change the value of the variable in a method, it will also be reflected in the calling method.

Example: Reference Type Variable

static void ChangeReferenceType(Student std2)

{

std2.StudentName = "Steve";

}

static void Main(string[] args)

{

Student std1 = new Student();

std1.StudentName = "Bill";

ChangeReferenceType(std1);

Console.WriteLine(std1.StudentName);

}

[Try it](https://www.tutorialsteacher.com/codeeditor?cid=cs-WL8Xgk)

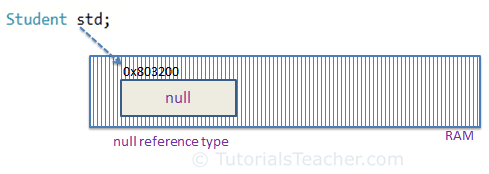
Output:

Steve

In the above example, since Student is an object, when we send the Student object std1 to the ChangeReferenceType() method, what is actually sent is the memory address of std1. Thus, when the ChangeReferenceType() method changes StudentName, it is actually changing StudentName of std1, because std1 and std2 are both pointing to the same address in memory. Therefore, the output is Steve.

Null

Reference types have null value by default, when they are not initialized. For example, a string variable (or any other variable of reference type datatype) without a value assigned to it. In this case, it has a null value, meaning it doesn't point to any other memory location, because it has no value yet.

[](https://www.tutorialsteacher.com/Content/images/csharp/null.png)Null Reference type

A value type variable cannot be null because it holds a value not a memory address. However, value type variables must be assigned some value before use. The compiler will give an error if you try to use a local value type variable without assigning a value to it.

Example: Compile Time Error

void someFunction()

{

int i;

Console.WriteLine(i);

}

tipC# 2.0 introduced nullable types for value types so that you can assign null to a value type variable or declare a value type variable without assigning a value to it.

However, value type field in a class can be declared without initialization (field not a local variable in the function) . It will have a default value if not assigned any value, e.g., int will have 0, boolean will have false and so on.

Example: Value Type Field

class myClass

{

public int i;

}

myClass mcls = new myClass();

Console.WriteLine(mcls.i);

[Try it](https://www.tutorialsteacher.com/codeeditor?cid=cs-S9ne6B)

Output:

0

https://www.tutorialsteacher.com/Content/images/bulb-glow.png Points to Remember :

1. Value type stores the value in its memory space, whereas reference type stores the address of the value where it is stored.
2. Primitive data types and struct are of the 'Value' type. Class objects, string, array, delegates are reference types.
3. Value type passes byval by default. Reference type passes byref by default.
4. Value types and reference types stored in Stack and Heap in the memory depends on the scope of the variable.