**Value Type and Reference Type:**

In C#, data types are categorized based on how they store their value in the memory. C# includes following categories of data types:

* Value type
* Reference type

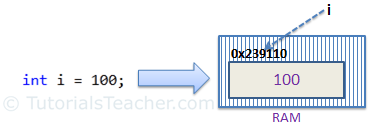
**Value Type:**

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.

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':

[](http://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

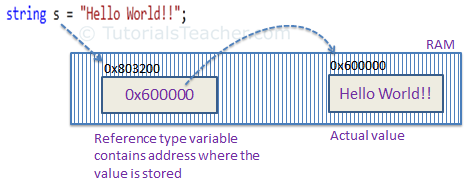
**Reference type:**

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.

[](http://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

**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 passes by value

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);

}

**Passing 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 passes by reference

static void ChangeReferenceType(Student std2)

{

std2.StudentName = "Steve";

}

static void Main(string[] args)

{

Student std1 = new Student();

std1.StudentName = "Bill";

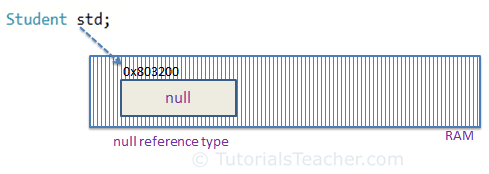
ChangeReferenceType(std1);

Console.WriteLine(std1.StudentName);

}

**null value:**

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.

[](http://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.

**Points to Remember :**

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