**Pass By Value And Pass By Reference In C#**

**(Ref And Out Keywords)**

PASS BY VALUE

static void Main(string[] args)

{

int value = 5;

PassByValue(value); 15

console.writeline(value);

}

Static void PassByValue (int a)

{

a = a + 10;

console.writeline(“value is: ” + a);

}

5

5

15

abc

PASS BY REFERENCE

Static void PassByRef (ref int a)

{

a = a + 10;

console.writeline(“value is: ” + a);

}

static void Main(string[] args)

{

int value = 5;

PassByRef(ref value);15

console.writeline(value);

}

static void Main(string[] args)

{

int value;

PassByOut(out value);

console.writeline(value);

}

PASS BY OUT

Static void PassByOut (out int a)

{

a = 20;

console.writeline(“value is: ” + a);

}

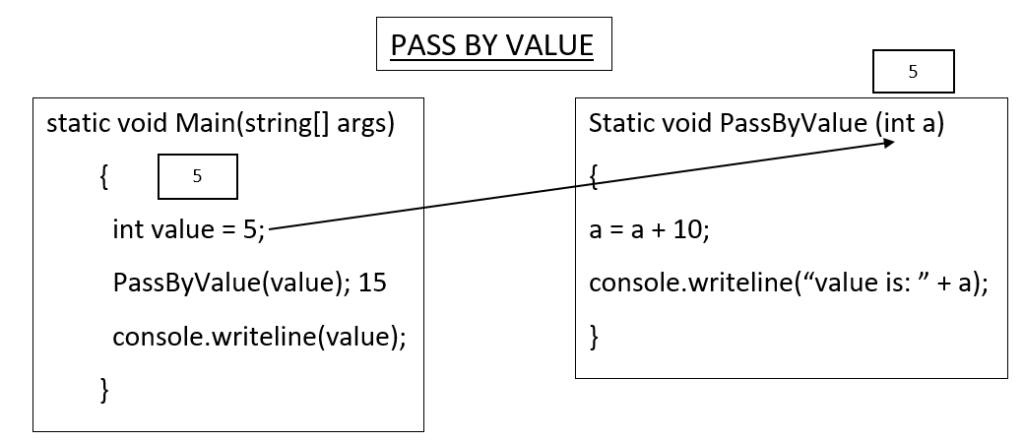
**Pass By Reference (Ref Keyword)**

* The ref keyword causes arguments to be passed in a method by reference.
* In call by reference, the called method changes the value of the parameters passed to it from the calling method.
* Any changes made to the parameters in the called method will be reflected in the parameters passed from the calling method when control passes back to the calling method.
* It is necessary that both the called method and the calling method must explicitly specify the **ref** keyword before the required parameters.
* The variables passed by reference from the calling method must be first initialized.

**Out Keyword**

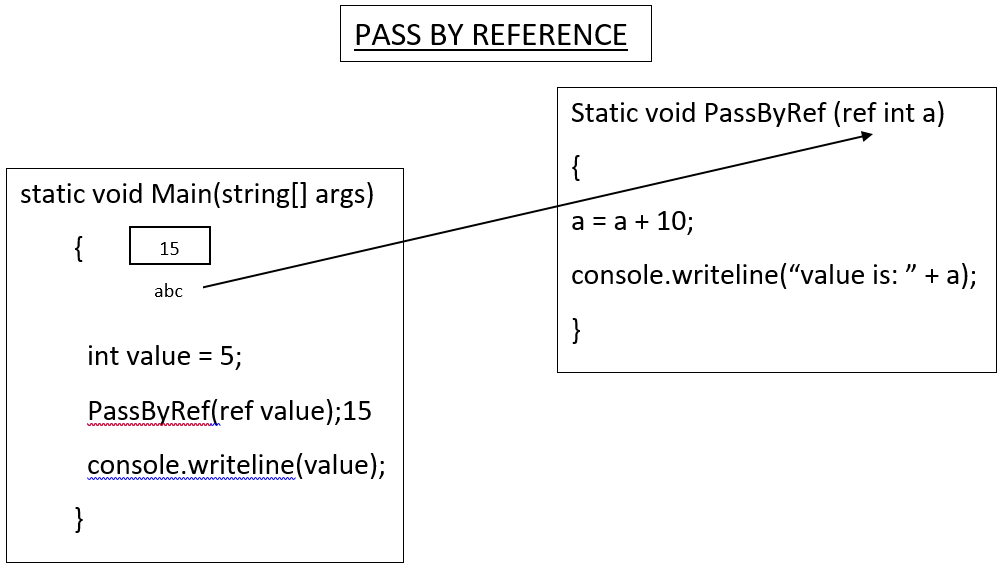
* The out keyword is similar to the ref keyword and causes arguments to be passed by reference.
* The only difference between the two is that the out keyword does not require the variables that are passed by reference to be initialized.
* Both the called method and the calling method must explicitly use the **out** keyword.

**Pass By Value**

**Pass By Value**

**Pass By Reference (Ref Keyword)**

* The ref keyword causes arguments to be passed in a method by reference.
* In call by reference, the called method changes the value of the parameters passed to it from the calling method.
* Any changes made to the parameters in the called method will be reflected in the parameters passed from the calling method when control passes back to the calling method.
* It is necessary that both the called method and the calling method must explicitly specify the ref keyword before the required parameters.
* The variables passed by reference from the calling method must be first initialized.

**Pass By Reference**

**The following syntax is used to pass values by reference using the ref keyword.**

<access\_modifier><return\_type><MethodName> (ref parameter1, ref parameter2, parameter3, parameter4, ...parameterN)

{

// actions to be performed

}

**where,**  
**parameter 1…parameterN:** Specifies that there can be any number of parameters and it is not necessary for all the parameters to be ref parameters.

using System;

class RefParameters

{

static void Calculate(ref intnumValueOne, ref int

numValueTwo)

{

numValueOne = numValueOne \* 2;

numValueTwo = numValueTwo / 2;

]

}

static void Main(string[] args)

{

intnumOne = 10;

intnumTwo = 20;

Console.WriteLine(“Value of Num1 and Num2 before calling method “ +numOne + “, “ + numTwo);

Calculate(ref numOne, ref numTwo);

Console.WriteLine(“Value of Num1 and Num2 after calling method “ +numOne + “, “ + numTwo);

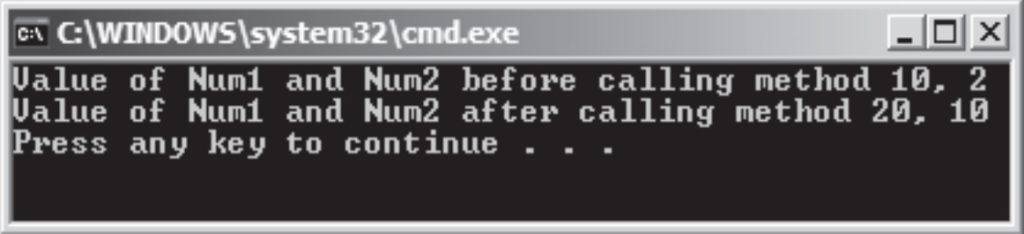
}

}

**In Above Code,**

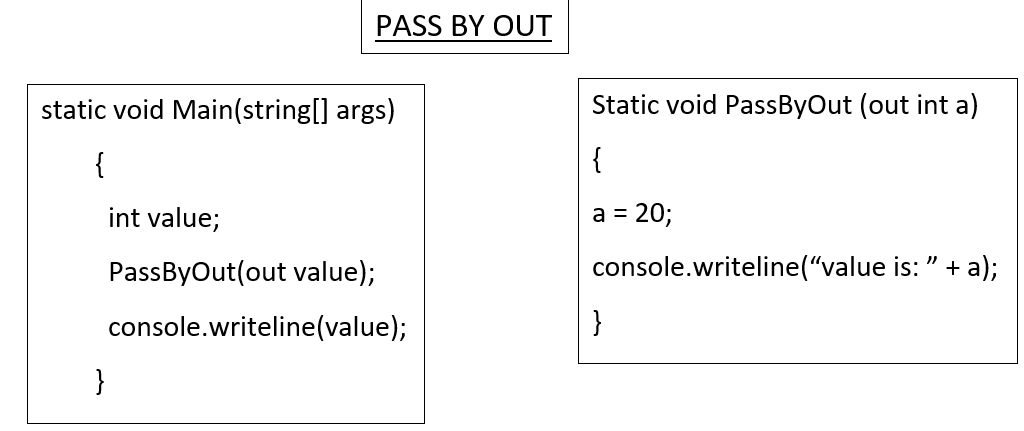
* The Calculate() method is called from the Main() method, which takes the parameters prefixed with the ref keyword.
* The same keyword is also used in the Calculate() method before the variables numValueOne and numValueTwo.
* In the Calculate() method, the multiplication and division operations are performed on the values passed as parameters and the results are stored in the numValueOne and numValueTwovariables respectively.
* The resultant values stored in these variables are also reflected in the numOne and numTwo variables respectively as the values are passed by reference to the method Calculate().

**The following figure displays the use of ref keyword: Output**



**Out Keyword**

* The out keyword is similar to the ref keyword and causes arguments to be passed by reference.
* The only difference between the two is that the out keyword does not require the variables that are passed by reference to be initialized.
* Both the called method and the calling method must explicitly use the out keyword.

**Out Keyword C#**

**The following syntax is used to pass values by reference using the out keyword:**

<access\_modifier><return\_type><MethodName> (out parameter1, out parameter2, ...parameterN)

{

// actions to be performed

}

**where,**  
parameter 1…parameterN: Specifies that there can be any  
number of parameters and it is not necessary for all the parameters to be  
out parameters.

**The following code uses the out keyword to pass the parameters by reference:**

using System;

class OutParameters

{

static void Depreciation(out intval)

{

{

val = 20000;

intdep = val \* 5/100;

intamt = val - dep;

Console.WriteLine(“Depreciation Amount: “ + dep);

Console.WriteLine(“Reduced value after depreciation: “ +

amt);

}

}

static void Main(string[] args)

{

int value;

Depreciation(out value);

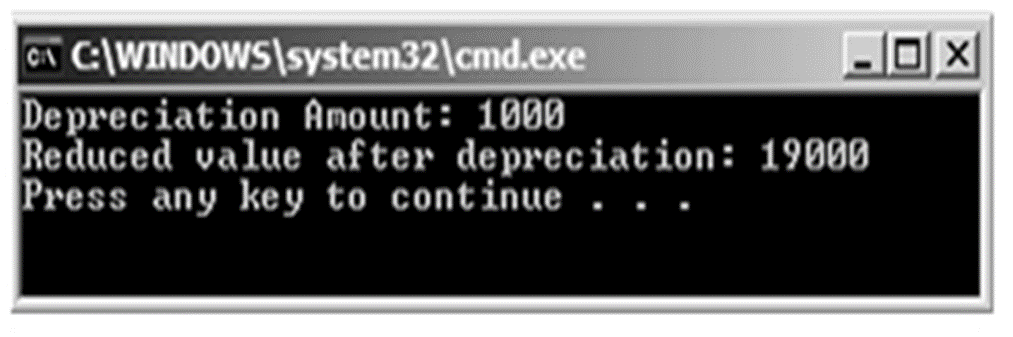
}

}

**In Above Code,**

* the Depreciation() method is invoked from the Main() method passing the val parameter using the out keyword. In the Depreciation() method, the depreciation is calculated and the resultant depreciated amount is deducted from the val variable. The final value in the amt variable is displayed as the output.

**The following figure shows the use of out keyword:**



**Source Code Of Pass By Value, Pass By Reference & Out Keywords**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace REF\_AND\_OUT\_KEYWORDS

{

class Program

{

//public static void PassByValue(int a)

//{

// a = a + 10;

// Console.WriteLine("Value is: {0}", a);

//}

//public static void PassByRef(ref int a)

//{

// a = a + 10;

// Console.WriteLine("Value is: {0}", a);

//}

public static void PassByOut(out int a)

{

a = 20;

Console.WriteLine("Value is: {0}", a);

}

static void Main(string[] args)

{

int value;

PassByOut(out value); // 20

Console.WriteLine(value); // 20

Console.ReadLine();

}

}

}