C# - PASSING PARAMETERS BY OUTPUT

A return statement can be used for returning only one value from a function. However, using **output parameters**, you can return two values from a function. Output parameters are similar to reference parameters, except that they transfer data out of the method rather than into it.

The following example illustrates this –

Live Demo

```
using System;

namespace CalculatorApplication {
   class NumberManipulator {
      public void getValue(out int x ) {
         int temp = 5;
         x = temp;
      }
      static void Main(string[] args) {
         NumberManipulator n = new NumberManipulator();

         /* local variable definition */
         int a = 100;

         Console.WriteLine("Before method call, value of a : {0}", a);

         /* calling a function to get the value */
         n.getValue(out a);

         Console.WriteLine("After method call, value of a : {0}", a);
         Console.ReadLine();
      }
   }
}
```

When the above code is compiled and executed, it produces the following result –

```
Before method call, value of a : 100
After method call, value of a : 5
```

The variable supplied for the output parameter need not be assigned a value. Output parameters are particularly useful when you need to return values from a method through the parameters without assigning an initial value to the parameter. Go through the following example, to understand this –

Live Demo

```
using System;

namespace CalculatorApplication {
   class NumberManipulator {
      public void getValues(out int x, out int y ) {
```

```
            Console.WriteLine("Enter the first value: ");
            x = Convert.ToInt32(Console.ReadLine());

            Console.WriteLine("Enter the second value: ");
            y = Convert.ToInt32(Console.ReadLine());
        }
        static void Main(string[] args) {
           NumberManipulator n = new NumberManipulator();

           /* local variable definition */
           int a , b;

           /* calling a function to get the values */
           n.getValues(out a, out b);

           Console.WriteLine("After method call, value of a : {0}", a);
           Console.WriteLine("After method call, value of b : {0}", b);
           Console.ReadLine();
        }
    }
}
```

When the above code is compiled and executed, it produces the following result –

```
Enter the first value:
7
Enter the second value:
8
After method call, value of a : 7
After method call, value of b : 8
```