**C# Out Parameter**

C# provides **out** keyword to pass arguments as out-type. It is like reference-type, except that it does not require variable to initialize before passing. We must use **out** keyword to pass argument as out-type. It is useful when we want a function to return multiple values.

**C# Out Parameter Example 1**

1. using System;
2. namespace OutParameter
3. {
4. class Program
5. {
6. // User defined function
7. public void Show(out int val) // Out parameter
8. {
9. int square = 5;
10. val = square;
11. val \*= val; // Manipulating value
12. }
13. // Main function, execution entry point of the program
14. static void Main(string[] args)
15. {
16. int val = 50;
17. Program program = new Program(); // Creating Object
18. Console.WriteLine("Value before passing out variable " + val);
19. program.Show(out val); // Passing out argument
20. Console.WriteLine("Value after recieving the out variable " + val);
21. }
22. }
23. }

**Output:**

Value before passing out variable 50

Value after receiving the out variable 25

The following example demonstrates that how a function can return multiple values.

**C# Out Parameter Example 2**

1. using System;
2. namespace OutParameter
3. {
4. class Program
5. {
6. // User defined function
7. public void Show(out int a, out int b) // Out parameter
8. {
9. int square = 5;
10. a = square;
11. b = square;
12. // Manipulating value
13. a \*= a;
14. b \*= b;
15. }
16. // Main function, execution entry point of the program
17. static void Main(string[] args)
18. {
19. int val1 = 50, val2 = 100;
20. Program program = new Program(); // Creating Object
21. Console.WriteLine("Value before passing \n val1 = " + val1+" \n val2 = "+val2);
22. program.Show(out val1, out val2); // Passing out argument
23. Console.WriteLine("Value after passing \n val1 = " + val1 + " \n val2 = " + val2);
24. }
25. }
26. }

**Output:**

Value before passing

val1 = 50

val2 = 100

Value after passing

val1 = 25

val2 = 25