## Linear Search – Some simple Performance measurements

**The goal for this exercise** is to get a quick orientation to ‘out’ parameters (at least enough to do the further exercises in this lesson)

For this exercise you should examine the following code (which is located in **Main** in the provided starter project):

|  |
| --- |
| QuickOverviewOfOutParameters qooop = new QuickOverviewOfOutParameters();  int num1 = 10;  Console.WriteLine("Normal Parameters: =============");  Console.WriteLine("PRE num1 is: {0}", num1);  qooop.NormalParam(num1);  Console.WriteLine("POST num1 is: {0}", num1); // notice that num1 hasn't changed  Console.WriteLine("Out Parameters: ================");  Console.WriteLine("PRE num1 is: {0}", num1);  qooop.OutParam(out num1);  Console.WriteLine("POST num1 is: {0}", num1); // notice that num1 HAS changed |

Notice that when you run this program that you get the following output:

|  |
| --- |
| Normal Parameters: =============  PRE num1 is: 10  PRE x is: 10  POST x is: 20  POST num1 is: 10  Out Parameters: ================  PRE num1 is: 10  POST x is: 20  POST num1 is: 20 |

I’ve highlighted various lines in order to make it easier to figure out which lines are generating what output.

The main thing to notice about this is that when you use an out parameter it allows the method to change the original parameter, back here in main. You can see this for yourself by examining the yellow-highlit lines that surround a ‘normal’ parameter, then compare that to what happens in the blue-highlit lines.

So how does an out parameter work within the method? In a nutshell you need to (1) add the “out” marker to the parameter declaration, (2) give it a value before you use it, and then (3) get (or change) the value like normal, keeping in mind that any changes will also be applied to the parameter you listed back in main. Let’s look at the C# source code:

|  |
| --- |
| public class QuickOverviewOfOutParameters  {  public void NormalParam(int x)  {  Console.WriteLine("PRE x is: {0}", x);  x = 20;  Console.WriteLine("POST x is: {0}", x);  }  public void OutParam(out int x)  {  // we're not allowed to use an out parameter until we give it a  // starting value.  // Try to uncomment the next line, and notice the compiler error you get  // Console.WriteLine("PRE x is: {0}", x);  x = 0; // starting value assigned here  x = 20; // notice that this changes the param here AND the "num1" back in main!  Console.WriteLine("POST x is: {0}", x);  }  } |

In future exercises within this lesson you will assign the out parameter a starting value (typically zero, since you’re using it as a counter), then increase the parameter whenever a comparison (or swap) occurs exactly as if it was a normal counter. Once back in main you can then print out that value exactly as if you had changed the value in main.

**What you need to do for this exercise:**

1. You do not need to produce anything, but you will need to understand this enough to use this in future exercises.