U10787
—
LAB 4: Understanding Parameters in Methods

Author	**Dr. Alexios Louridas**
Last Update	**2023-02-24**
Number of pages	**5**

1 Parameters passed By Value

Parameters in a method that are declared without the keywords: in, ref or out, are passed to the called method by value. That value can be changed in the method, but the changed value will not be retained when control passes back to the calling procedure.

1.1 Example

```
public void PrintName(string name)
{
    name += " added string";

    Console.WriteLine(name);
}

string firstName = "Alex";

PrintName(firstName);

Console.WriteLine(firstName);
```

1.2 Example Explanation

The firstName variable is created and passed by value in the method PrintName. This means that we only pass the value of the variable not the variable itself. What the program does is that it creates a copy of firstName and uses that copy within the method. It does not pass back anything else. So any changes in the variable passed within the method only happen in the copy not in the actual variable passed.

1.3 Task 1

Write a method Power that take two integer parameters base and exponent. The method should return the value of $base^{exponent}$.Do not use any Math library method. Create a simple application that reads integer values for base and exponent and displays the result.

2 Parameters passed By Reference

To pass the variable as a memory location and not only its value. You can use three keywords in front of variable type:

in specifies that this parameter is passed by reference but is only read by the called method.

2.1 Example

```
public void InArgExample(in int number)
{
    // Uncomment the following line to see error CS8331
    // number = 19;
}

int readonlyArgument = 42;

InArgExample(readonlyArgument);

Console.WriteLine(readonlyArgument);     // value is still 44
```

2.2 Example Explanation

Variables passed as "in" must be initialized before being passed in a method call. However, the called method may not assign a value or modify the variable.

ref specifies that this parameter is passed by reference and may be read or written by the called method.

2.3 Example

```
int number = 1;
void MethodByRef(ref int refArgument)
{
    Console.WriteLine("Variable \"refArgument\" value as received in the method:
      " + refArgument);

    // Change the value of number
    refArgument = refArgument + 44;

    Console.WriteLine("Variable \"refArgument\" value as changed and called
    within the method: " + refArgument);

    // Feedback
    Console.WriteLine();
    Console.WriteLine("Both refArgument and number are refering to the same
    location in memory.");
    Console.WriteLine();

}

MethodByRef(ref number);

Console.WriteLine("Variable value after calling the method: " + number);
```

2.4 Example Explanation

Variables passed as "ref" must be initialized before being passed in a method call. To use a ref parameter, both the method definition and the calling method must explicitly use the ref keyword, as shown in the example

out specifies that this parameter is passed by reference and is written by the called method.

2.5 Example

```
void MethodByRef(out int refArgument)
{
    // Initialise the variable within the method
    refArgument = 0;

    Console.WriteLine("Variable \"refArgument\" value as assinged in the method:
     " + refArgument);

    // Change the value of number
    refArgument = refArgument + 49;

    Console.WriteLine("Variable \"refArgument\" value as changed and called
    within the method: " + refArgument);
}

MethodByRef(out number);

Console.WriteLine("Variable value after calling the method: " + number);
```

2.6 Example Explanation

Variables passed as "out" must be initialized inside the method call. Declaring a method with out parameters is a classic workaround to return multiple values from a method.

2.7 Task 2

Try and complete the methods within the MovieCollection class seen below. Once completed test your code by creating an object of MovieCollection and calling all methods to test.

```
public class Movie
{
    public string Title;
    public string Director;
}

public class MovieCollection
{
    List<Movie> MyCollection = new List<Movie>();

    public void AddMovie(string title, string director)
    {
        // Add code
    }

    public void AddMovie(ref Movie movieToAdd)
    {
        // Add code
    }

    // Return null if not found
    public Movie GetMovie(string title)
    {
        // Add code
    }

    // True if movie has been found, False if not found
    public bool GetMovieByTitle(string title, out Movie movieFound)
    {
        // Add code
    }

    // True if movie has been found, False if not found
    public bool DoesMovieExist(in Movie checkMovie)
    {
        // Add code
    }
}


// Instntiate an object called myMovies


// Add at least 2 movies in myMovies using both overloaded methods AddMovie


// Try and use GetMovieByTitle and GetMovieByDirector

// Create a movie class and check if it exists inside your collection using the
    DoesMovieExist method
```

3 Variable Length Parameter List

The ***params*** keyword allow you to create methods with variable length parameter list. The ***params*** keyword can only be used for the last parameter in a method.

3.1 Example

```
double Average(params double[] numbers)
{
    double total = 0.0;

    for (int i = 0; i < numbers.Length; i++)
    {
        total += numbers[i];
    }

    return total / numbers.Length;
}

double d1 = 10.0;
double d2 = 20.0;
double d3 = 30.0;
double d4 = 40.0;

Console.WriteLine ($"Average of d1 and d2 is {Average(d1,d2):F1}");
Console.WriteLine ($"Average d1, d2 and d3 is {Average(d1,d2,d3):F1}");
Console.WriteLine ($"Average d1, d2, d3 and d4 is {Average(d1,d2,d3,d4):F1}");
```

3.2 Example Explanation

The method Average receives a variable-length sequence of doubles. The variable-length parameter list is treated as a one dimensional array of the same type declared, in this instance double.You then can call the method multiple times with a number of parameters.

3.3 Task 3

Create a console application that calculates the product of a series of integers that are passed to a method called Product using a variable-length parameter list. Make sure you test your method with several calls, each with different number of parameters.

4 Sieve of Eratosthenes

A prime number is defined as any whole number greater than 1 that can only be divided by itself and 1. The Sieve of Eratosthenes is a method for identifying prime numbers. The process involves creating a boolean array with all elements initially set to true. The array elements that correspond to prime numbers will remain true, while all other elements will eventually be set to false.

Starting from index 2 of the array, check if the element is true. If it is, then iterate through the rest of the array and set every element whose index is a multiple of the index of the true element to false. Then, repeat this process with the next element that has a value of true.

Example 1: for the element at index 2, all elements in the array with indices that are multiples of 2 (such as 4, 6, 8, 10, etc.) will be set to false.

Example 2: for the element at index 3, all elements in the array with indices that are multiples of 3 (such as 6, 9, 12, 15, etc.) will be set to false, and so on.