Lambda Expression Assignment Tutorial for C#

Jia Then

September 15, 2020

Introduction

This tutorial will cover the concept of Lambda Expression using anonymous function expressions in C#. Additionally, the various segments of the expression will be covered in this assignment. The repository for this tutorial is available in this link (TECHCareers, 2020). Codes and code blocks will be highlighted in grey.

Requirements

This tutorial will require Visual Studio 2019 installed in the computer. Visual Studio 2019 can be acquired through this link.

Background

Lambda expression is a form of an anonymous function that uses Lambda syntax => which is a combination of an assignment operator, = and a larger than operator, >. An anonymous function is defined as an inline statement or expression that can be used wherever a delegate type is expected (Microsoft, 2015a). Therefore, the anonymous function is a method without a name (TutorialsTeacher, 2020a). Delegate is defined as a reference type that represents references to methods with a particular parameter list and return type (Microsoft, 2015b).

In 2005, C# 2.0 introduced the concept of anonymous methods as a way to write unnamed inline statement blocks that can be executed in a delegate invocation (Microsoft, 2015a). The delegate is initialized with the keyword delegate as shown below as Example 1:

```
delegate void TestDelegate(string s);
TestDelegate testDelB = delegate(string s) { Console.WriteLine(s);
};
testDelB("Print testDelB");
```

Example 1: Initialization of an anonymous method with the use of delegate.

However, the introduction of Lambda syntax => in C# 3.0 made anonymous methods a much more concise code as shown below as Example 2 (Microsoft, 2015a):

```
delegate void TestDelegate(string s);
TestDelegate testDelC = (x) => { Console.WriteLine(x); };
testDelC("Print testDelC");
```

Example 2: Initialization of an anonymous method with the use of an lambda expression

Currently, as of September 2020, lambda expression has become the standard for creating anonymous functions. Figure 1 below illustrates anonymous methods evolution from C# 1.0 to C# 3.0 (Microsoft, 2015a).

```
class Test
{
    delegate void TestDelegate(string s);
    static void M(string s)
    {
        Console.WriteLine(s);
    }

    static void Main(string[] args)
    {
        // Original delegate syntax required
        // initialization with a named method.
        TestDelegate testDelA = new TestDelegate(M);

        // C# 2.0: A delegate can be initialized with
        // inline code, called an "anonymous method." This
        // method takes a string as an input parameter.
        TestDelegate testDelB = delegate(string s) { Console.WriteLine(s); };

        // C# 3.0. A delegate can be initialized with
        // a lambda expression. The lambda also takes a string
        // as an input parameter (x). The type of x is inferred by the compiler.
        TestDelegate testDelC = (x) => { Console.WriteLine(x); };

        // Invoke the delegates.
        testDelA("Hello. My name is M and I write lines.");
        testDelB("That's nothing. I'm anonymous and ");
        testDelC("I'm a famous author.");

        // Keep console window open in debug mode.
        Console.WriteLine("Press any key to exit.");
        Console.ReadKey();
    }
}
/* Output:
    Hello. My name is M and I write lines.
    That's nothing. I'm anonymous and
    I'm a famous author.
    Press any key to exit.
 */
```

Figure 1: The evolution of Lambda Expression from Anonymous Method (Microsoft, 2015a).

Methodology

The lambda operator, `=>` is used to separate the input parameters on the left side from the lambda body on the right side (Microsoft, 2019b). Lambda expression is created by specifying the input parameters (if any) on the left side of `=>` and expression/statement block on the right.

Implementation Example

Let's make our own Anonymous function using Lambda expression, `=>`. The goal of this example is to solidify the concepts of Anonymous function, delegates, and lambda expressions.

Step 1: Make a delegate

```
delegate int Example3(int x, int y);
```

Delegate with name `Example3` was declared. `Example3` accepts two parameters, x and y. Both parameters are integer type. The completed code within Visual Studio 2019 should look like this:

```
// Example 3:
// Step 1: Make a delegate.
delegate int Example3(int x, int y);
```

Figure 2.

Step 2: Initialize anonymous function

```
Example3 test = (x, y) => x + y;
```

Anonymous function initialized of type `Example3` with the variable name `test` declared. The lambda expression in Step 2 initializes the delegate implicitly that accepts input parameters of x and y as (x , y). To the right of the lambda expression, `=>`, the values are returned as the sum of x+y. The completed code within Visual Studio 2019 should look like this:

```
// Example 3:
// Step 2: Initialize Anonymous function()
Example3 test = (x, y) => x + y;
```

Figure 3.

Step 3: Invoke the delegate

```
Console.WriteLine(test(5, 5));
```

To invoke the delegate, example test(5,5) was used. The output of the Anonymous function test should be 10. The completed code within Visual Studio 2019 should look like this:

```
// Step 3: Invoke the delegate
Console.WriteLine(test(5, 5));
```

FIgure 4.

Step 4: Output within Microsoft Visual Studio Debug Console

The output of the test within the Microsoft Visual Studio Debug Console should have a value of 10, underlined in red in Figure 4 below.

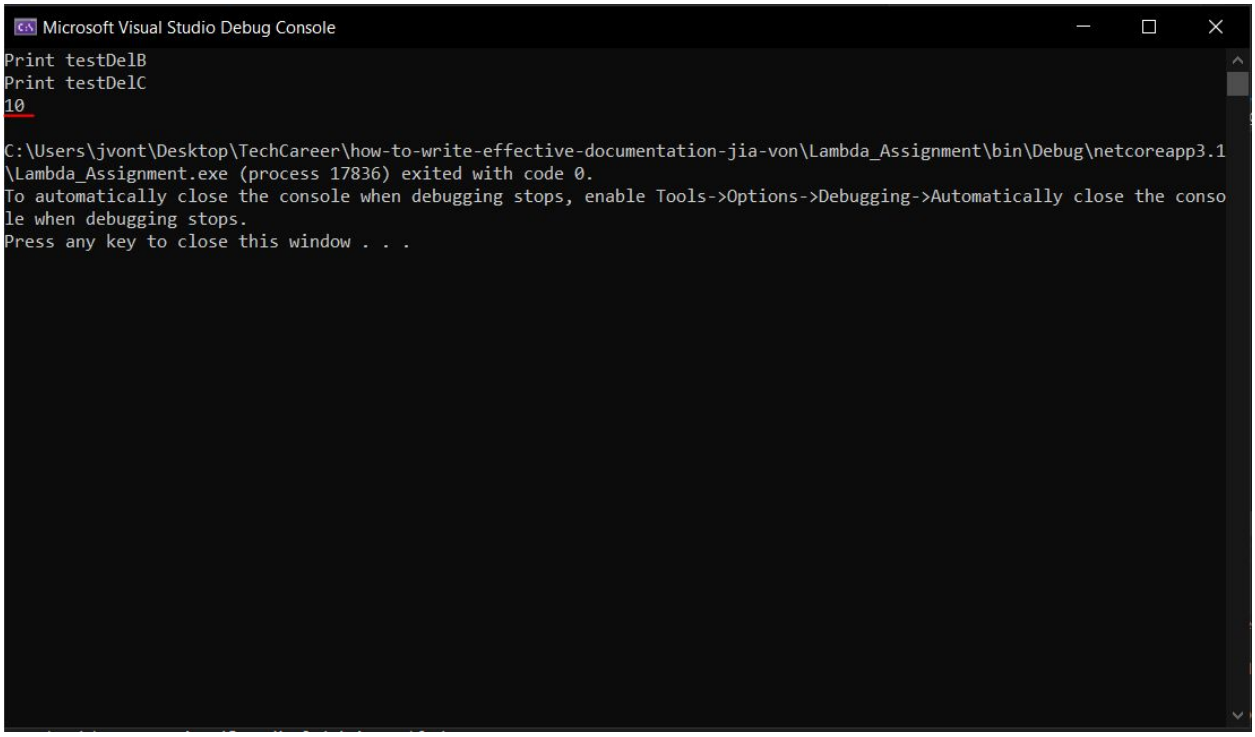

Figure 5.

References:

TECHCareers. 2020. Lambda Expression Assignment (C# reference). Accessed September 15, 2020. URL: https://github.com/TECHCareers-by-Manpower/how-to-write-effective-documentation-jia-von

TutorialsTeacher. 2020a. C# - Anonymous Method. Accessed September 15, 2020. URL: https://www.tutorialsteacher.com/csharp/csharp-anonymous-method

TutorialsTeacher. 2020b. Anatomy of the Lambda Expression. URL: https://www.tutorialsteacher.com/linq/linq-lambda-expression

Microsoft. 2015a. Anonymous functions (C# Programming Guide): The Evolution of Delegates in C#. Accessed September 15, 2020. URL: https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/statements-expressions-operators/anonymous-functions

Microsoft. 2015b. Delegates (C# Programming Guide). Accessed September 15, 2020. URL: https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/delegates/

Microsoft. 2019a. Lambda expressions (C# reference). Accessed September 15, 2020. URL: https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/operators/lambda-expressions.

Microsoft. 2019b. => operator (C# reference). Accessed September 15, 2020. URL: https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/operators/lambda-operator