

# C# Lambdas and LINQ

Rasmus Lystrøm  
Associate Professor  
ITU  
[rne@itu.dk](mailto:rne@itu.dk)

```
File Edit Selection View Go Debug Terminal Help
ProgramTests.cs x
HelloWorld.Tests > ProgramTests.cs > {} HelloWorld.Tests > HelloWorld
1 using System;
2 using System.IO;
3 using Xunit;
4
5 namespace HelloWorld.Tests
6 {
7     0 references | Run All Tests | Debug All Tests
8     public class ProgramTests
9     {
10         [Fact]
11         0 references | Run Test | Debug Test
12         public void Main_given_no_args_p
13         {
14             // Arrange
15             using var writer = new StringWriter();
16             Console.SetOut(writer);
17
18             // Act
19             Program.Main(new string[0]);
20
21             // Assert
22             var output = writer.GetStringBuilder().ToString();
23             Assert.Equal("Hello World!", output);
24         }
25     }

```

```
Program.cs x
HelloWorld > Program.cs > ...
1 using System;
2
3 namespace HelloWorld
4 {
5     1 reference
6     public class Program
7     {
8         1 reference
9         public static void Main(string[] args)
10         {
11             Console.WriteLine("Hello World!");
12         }
13     }

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

1: pwsh

```
Loading personal and system profiles took 1008ms.
C:\HelloWorld> dotnet test
Test run for C:\HelloWorld\HelloWorld.Tests\bin\Debug\netcoreapp3.0\HelloWorld.Tests.dll (.NETCoreApp, Version=v3.0)
Microsoft (R) Test Execution Command Line Tool Version 16.3.0
Copyright (c) Microsoft Corporation. All rights reserved.

Starting test execution, please wait...

A total of 1 test files matched the specified pattern.

Test Run Successful.
Total tests: 1
Passed: 1
Total time: 3,4618 Seconds
C:\HelloWorld>

```

1 tests 0 0 Azure: rasmusl@microsoft.com HelloWorld.sln Ln 22, Col 50 Spaces: 4 UTF-8 CRLF C# 1

# Agenda

Properties

Anonymous methods

Delegates

Lambda expressions

Local functions

Anonymous types

Tuples

Extension methods

LINQ

# Properties

# Properties 1/3

```
public class City
{
    public int Id { get; set; }
    public string Name { get; private set; }
}
```



# Properties 2/3

```
public class City
{
    private int _id;
    public int Id { get => _id; set => _id = value; }

    private string _name;
    public string Name
    {
        get { return _name; }
        set
        {
            _name = value;
        }
    }
}
```

# Properties 3/3

```
public class City
{
    private int _id;
    public int Id
    {
        get => _id;
        set
        {
            // Place setter validation logic here if required
            _id = value;
        }
    }
}
```

# Delegates

# Delegates – Building block for Higher-order functions

```
public delegate int BinaryOperation(int x, int y);
```

```
static void Main(string[] args)
{
    var add = new BinaryOperation(
        delegate (int x, int y)
        {
            return x + y;
        }
    );
}
```



# Delegates demo

# Lambda Expressions

# Lambda Expressions

```
Action<string> write = s => Console.WriteLine(s);
```

```
Predicate<City> b = c => c.Name.StartsWith("B");
```

```
Func<int, int> square = a => a * a;
```

(Local functions)

```
static void Main(string[] args)
{
    int square(int a) { return a * a; };

    Console.WriteLine(square(16));
}
```

# Anonymous types

```
var question = new
{
    Title = "The answer...",
    Answer = 42
};
```

# (Tuples)

```
var s = Tuple.Create("Clark Kent", "Superman");
```

```
var b = ("Bruce Wayne", "Batman");
```

```
var f = (name: "Barry Allen", alterEgo: "The Flash");
```

```
IEnumerable<(float x, float y)> GenerateCoordinates()  
{  
    yield return (1.3f, 23.45f);  
}
```

# Extension Methods

# Data: Collection\_INITIALIZER

```
IEnumerable<City> cities = new[]  
{  
    new City(1, "Berlin"),  
    new City(2, "Hamburg"),  
    new City(3, "Frankfurt")  
};
```



# Data: Collection + Object Initializer

```
IEnumerable<City> cities = new[]  
{  
    new City { Id = 1, Name = "Berlin" },  
    new City { Id = 2, Name = "Hamburg" },  
    new City { Id = 3, Name = "Frankfurt" }  
};
```

# Extension methods 1/2

```
var count = cities.Count();
```

```
var sorted = cities.OrderBy(c => c.Name);
```

```
var filtered = cities.Where(c => c.Name.Contains("i"));
```

```
var pick = cities.FirstOrDefault(c => c.Id == 2);
```

```
var all = cities.All(c => c.Name.Length < 10);
```

```
var any = cities.Any(c => c.Name.StartsWith("B"));
```

```
var select = cities.Select(c => c.Name);
```

# Extension methods 2/2

```
public static class Extensions
{
    public static int WordCount(this string str)
    {
        return str.Split(
            new char[] { ' ', '.', '?' },
            StringSplitOptions.RemoveEmptyEntries).Length;
    }
}
```

# LINQ – Language INtegrated Query

# LINQ

```
var sorted = from c in cities
              where c.Name.Contains("i")
              orderby c.Name descending
              select new { Name = c.Name };
```

# Extension Methods Version

```
var sorted = cities.Where(c => c.Name.Contains("i"))  
                    .OrderByDescending(c => c.Name)  
                    .Select(c => new { Name = c.Name });
```

# LINQ demo