[C# and dotnet 2](#_Toc15601)

[Delegates 2](#_Toc27510)

[Lambda Expression 2](#_Toc32264)

[Events 2](#_Toc13602)

[Extension Methods 2](#_Toc10199)

[Keyword 2](#_Toc6745)

[Sealed 2](#_Toc27361)

[Where 2](#_Toc10578)

[Dynamic 2](#_Toc9552)

[Var 3](#_Toc29978)

[Operator 3](#_Toc15710)

[Nameof 3](#_Toc29910)

[Typeof 3](#_Toc14356)

[Struct 4](#_Toc8732)

[Bigstruct 4](#_Toc13996)

[Generic Collections 4](#_Toc30517)

[Important interface 4](#_Toc6238)

[IEnumerable 4](#_Toc13377)

[Dictionary<TKey,TValue> Class 4](#_Toc11120)

[List<T> Class 4](#_Toc19084)

[Dependency Injection 4](#_Toc28587)

[Covariance and Contravariance 5](#_Toc1955)

[Generics 5](#_Toc15792)

[LINQ - Language Integrated Query 5](#_Toc9695)

[Aggregate functions 7](#_Toc447)

[Restriction Function 8](#_Toc22149)

[Projection Function 8](#_Toc28144)

[Method to suppress code analysis warning 8](#_Toc14765)

[Memory 8](#_Toc26346)

[Stack vs Heap Memory 8](#_Toc6793)

[Dotmemory 8](#_Toc18331)

[Unit testing 9](#_Toc15047)

[Arrange, Act, Assert 9](#_Toc15241)

[Attributes 10](#_Toc3619)

[xUnit 10](#_Toc11429)

[NUnit 10](#_Toc3938)

[MSTest 10](#_Toc26831)

[Attributes 11](#_Toc25165)

[Web Driver Test 11](#_Toc1135)

[Runsettings file 11](#_Toc21098)

[Unit test misc 11](#_Toc27334)

[How to write unit test for private methods and class? 11](#_Toc31868)

[Misc and questions 11](#_Toc20971)

[Mutable vs Immutable? 11](#_Toc8473)

[Can we have multiple awaits in asynchronous methods? 11](#_Toc14224)

[Value Types and Reference Types 11](#_Toc20691)

[Access Modifiers in C#. Explain each 12](#_Toc29952)

[Difference between ‘==’ and ‘.Equals()’ 12](#_Toc26313)

[Parallel class in C# 12](#_Toc11251)

[LINQ 12](#_Toc11415)

[What are Lambda Expressions in LINQ? Can you provide an example? 12](#_Toc6960)

# C# and dotnet

## Delegates

## Lambda Expression

## Events

## Extension Methods

Enables user to add new methods to existing type without creating new derived type or modifying the existing type. Static method, but used as instance method. While using it the call looks same similar to the usage of instance method.

## Keyword

### Sealed

### Where

Specifies constraints on the types that are used as arguments for type parameters in a generic type, method, delegate, or local function.

Sample code in Github

### Dynamic

Type is defined at run time. For variable which involves late binding. Use cases where reflection will be used to get the instance.

### Var

For anonymous types. Once a type is assigned, cannot cast a a new type again. Type is defined at compile time.

Sample code in Github

## Operator

**Msdn reference:** https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/operators/

### Nameof

Converts name of member to string

**nameof(List<int>.Count) - Output is “Count”**

### Typeof

*Type t =* ***typeof****(String);*

***MethodInfo*** *substr =* ***t.GetMethod****("Substring", new Type[] { typeof(int), typeof(int) });*

*Object result =* ***substr.Invoke****("Hello, World!", new Object[] { 7, 5 });*

*Console.WriteLine("{0} returned \"{1}\".", substr, result);*

//Output: System.String Substring(Int32, Int32) returned "World".

## Struct

### Bigstruct

## Generic Collections

### Important interface

#### IEnumerable

### Dictionary<TKey,TValue> Class

**Implements,**

**ICollection<KeyValuePair<TKey,TValue>>**, IDictionary<TKey,TValue>, **IEnumerable<KeyValuePair<TKey,TValue>>**, IReadOnlyCollection<KeyValuePair<TKey,TValue>>, IReadOnlyDictionary<TKey,TValue>,

System.Collections.IDictionary, IDeserializationCallback, ISerializable

### List<T> Class

**ICollection<T>**, **IEnumerable<T>**, IList<T>, IReadOnlyCollection<T>, IReadOnlyList<T>, IList

## Dependency Injection

Software design pattern and a technique used in object-oriented programming to achieve loose coupling between components and manage dependencies between classes. The process of obtaining the dependencies required by a class is moved out of the class itself, and the dependencies are provided from the outside (typically by a DI container) when the class is instantiated.

Example in Github: DependencyInjection folder

## Covariance and Contravariance

Msdn ref: <https://learn.microsoft.com/en-us/dotnet/standard/generics/covariance-and-contravariance>

## Generics

Introduces concept of type paramters.

To create reusable, type-safe code by defining classes, interfaces, and methods that can work with different data types. Avoids the need for unnecessary type casting.

Signature:

***public class GenericList<T>***

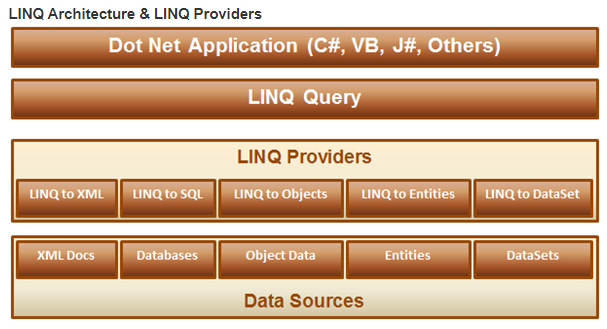
***public interface IRepository<T, TKey>***

Applicable for interface and struct as well

Opposite of this is heterogeneous collection one of the example is array list. Array list is not recommended.

## LINQ - Language Integrated Query

Name for a set of technologies based on the integration of query capabilities directly into the C# language. Queries any type of data (SQL server, XML documents, objects in memory).

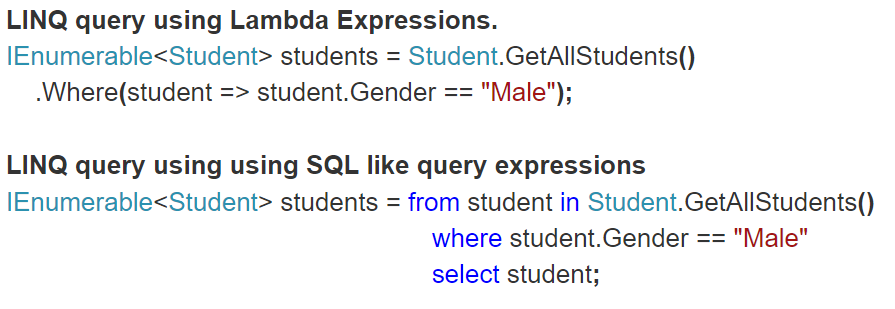


LINQ Provider: Component to convert LINQ Query to format that the underlying dataset can understand. LINQ to SQL converts LINQ to T-SQL.

2 Ways to write LINQ Queries. Performance remains same for both. But SQL like queries will be converted to lambda expression, before execution.

- Lambda expression

- SQL like queries



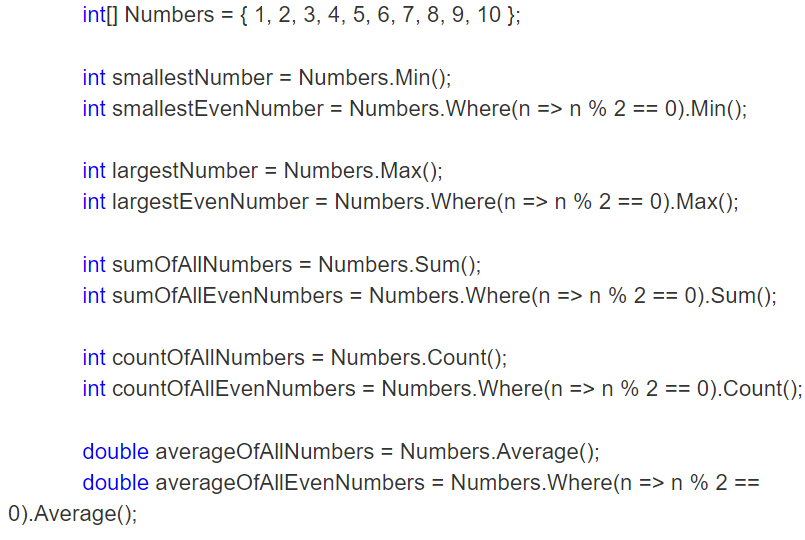
| **Clause** | **Description** |
| --- | --- |
| **[from](https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/from-clause)** | Specifies a **data source** and a range variable (similar to an iteration variable). |
| **[where](https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/where-clause)** | **Filters** source elements **based on** one or more **Boolean expressions** separated by logical AND and OR operators ( && or || ). |
| **[select](https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/select-clause)** | **Specifies** the **type and shape** that the elements in the returned sequence will have when the query is executed. |
| **[group](https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/group-clause)** | **Groups** query results **according to a specified key value**. |
| **[into](https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/into)** | Provides an identifier that can serve as a reference to the results of a join, group or select clause. |
| **[orderby](https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/orderby-clause)** | Sorts query results in ascending or descending order based on the default comparer for the element type. |
| **[join](https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/join-clause)** | **Joins two data sources** based on an **equality comparison** between two specified matching criteria. |
| **[let](https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/let-clause)** | Introduces a range variable to store sub-expression results in a query expression. |
| **[in](https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/in)** | Contextual keyword in a [join](https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/join-clause) clause. |
| **[on](https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/on)** | Contextual keyword in a [join](https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/join-clause) clause. |
| **[equals](https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/equals)** | Contextual keyword in a [join](https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/join-clause) clause. |
| **[by](https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/by)** | Contextual keyword in a [group](https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/group-clause) clause. |
| **[ascending](https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/ascending)** | Contextual keyword in an [orderby](https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/orderby-clause) clause. |
| **[descending](https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/descending)** | Contextual keyword in an [orderby](https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/orderby-clause) clause. |

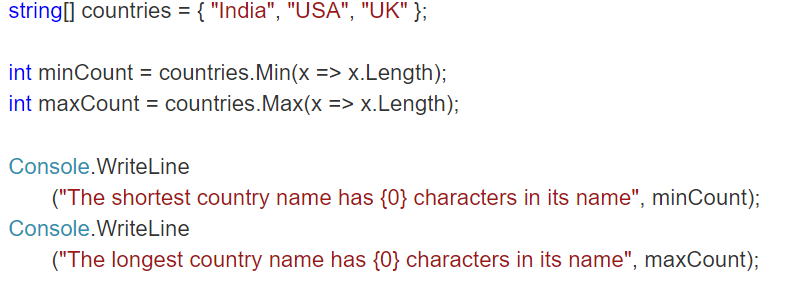
Benefits:

1. Provides compile time error checking
2. Same syntax for various data source

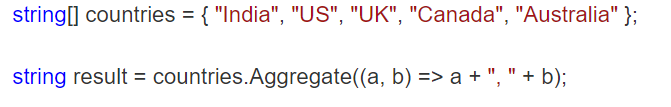
### Aggregate functions

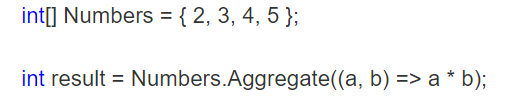
Min, Max, Sum, Count, Average, Aggregate





***Aggregate***





Iterates untill end of list

A,b - 2, 3

A,b - 6, 4

A,b - 24, 5

### Restriction Function

**Where** keyword - used to represent constraint

Refer Github RestrictionOperator

### Projection Function

**Select and SelectMany**

Used to transform the results of a query

Retrieves single or multiple properties into an anonymous type

To perform calculations

## Method to suppress code analysis warning

try { ... }

catch (Exception e)

{

**#pragma warning disable** CA2200 // Rethrow to preserve stack details

throw e;

**#pragma warning restore** CA2200 // Rethrow to preserve stack details

}

## Memory

### Stack vs Heap Memory

### Dotmemory

Memory profiler tool by jet brains.



Unmanaged memory

Heap generation 0,1,2

Large object heap

## Unit testing

**Msdn reference:** https://learn.microsoft.com/en-us/dotnet/core/testing/

### Arrange, Act, Assert

Its a common pattern when unit testing. As the name implies, it consists of three main actions:

* Arrange your objects, create and set them up as necessary.
* Act on an object.
* Assert that something is as expected.

***[Fact]***

*public void Add\_EmptyString\_ReturnsZero()*

*{*

*// Arrange*

*var stringCalculator = new StringCalculator();*

*// Act*

*var actual = stringCalculator.Add("");*

*// Assert*

*Assert.Equal(0, actual);*

*}*

### Attributes

**[Fact]:** Used to mark a method that contains a test

**[Theory]:** Used in conjunction with [InlineData], [ClassData], or [MemberData] to run a test multiple times with different input values.

**[InlineData]:** Specify the parameters of a test.

**[ClassData]:** Specify a class that returns the parameters of a test.

**[MemberData]:** Specify a property, field, or method that returns the parameters of a test.

**[Trait]:** Used to categorize tests, allowing you to filter which tests are run.

**[Collection]:** Used to indicate that a test class is part of a collection that shares a test context.

**[CollectionDefinition]:** Used to define a collection of tests that can share a context.

**[Output]:** Used to capture and output additional information during test execution. This requires an instance of ITestOutputHelper to be passed into the test class's constructor.

**[Skip]:** An argument you can add to the [Fact] and [Theory] attributes to skip the execution of a particular test, usually with a reason.

**[BeforeAfterTest]:** An attribute that can be applied to a class, allowing actions to be performed before and after each test within the class.

Sample code in github

### xUnit

### NUnit

### MSTest

Integrated with visual studio

#### Attributes

TestClass: Decorator for class that contains test methods

TestMethod: Decorator for test methods

### Web Driver Test

Project type in visual studio to automate unit testing of web sites within edge browser

### Runsettings file

Refer this link: <https://learn.microsoft.com/en-us/visualstudio/test/configure-unit-tests-by-using-a-dot-runsettings-file?view=vs-2022>

TODO:: Explore this file

### Unit test misc

#### How to write unit test for private methods and class?

## Misc and questions

### Mutable vs Immutable?

### Can we have multiple awaits in asynchronous methods?

Yes

### Value Types and Reference Types

Value types are stored on the stack, while reference types are stored on the heap

### Access Modifiers in C#. Explain each

Public, private, protected, internal, protected internal

### Difference between ‘==’ and ‘.Equals()’

Reference code in github Equals\_EqualToOperator. Always recommended to use .Equals or is for null check, since == can be overlaoded.

### Parallel class in C#

<https://learn.microsoft.com/en-us/dotnet/api/system.threading.tasks.parallel?view=net-7.0>

Provides support for parallel loops and regions.

### LINQ

#### What are Lambda Expressions in LINQ? Can you provide an example?

Lambda Expressions are anonymous functions used to encapsulate a set of instructions. They are commonly used in LINQ queries. Example: list.Where(x => x > 5), where x => x > 5 is the lambda expression.