LINQ

(Language-Integrated Query)

.NET

Language-Integrated Query (LINQ) is the name for a set of technologies based on the integration of query capabilities directly into the C# language.

https://docs.microsoft.com/en-us/dotnet/csharp/linq/

LINQ - Overview

https://docs.microsoft.com/en-us/dotnet/csharp/linq/
https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/linq/

Traditionally, queries against data (in a DB or file) have been expressed as simple strings without type checking at compile time or IntelliSense support.

You'd have to learn a different query language for each type of data source:

- SQL databases,
- XML documents,
- various Web services, etc.

◦ With ***LINQ***, a query is a language construct, just like ***classes***, ***methods***, ***events***.

◦ Query expressions are written in a declarative query syntax.

◦ You can perform filtering, ordering, and grouping operations on data sources with minimum code.

◦ You use the same basic query expression patterns to query and transform data in SQL databases, ADO .NET Datasets, XML documents and streams, and .NET collections.

LINQ - Overview

https://docs.microsoft.com/en-us/dotnet/csharp/linq/

A complete LINQ query operation includes

- creating a data source,
- defining the query expression, and
- executing the query in a ***foreach*** statement.

There are two ***Query Expression*** syntaxes

- Query Syntax
- Method-Based Syntax

```
class LINQQueryExpressions
{
    static void Main()
    {

        // Specify the data source.
        int[] scores = new int[] { 97, 92, 81, 60 };

        // Define the query expression.
        IEnumerable<int> scoreQuery =
            from score in scores
            where score > 80
            select score;

        // Execute the query.
        foreach (int i in scoreQuery)
        {
            Console.Write(i + " ");
        }
    }
}
// Output: 97 92 81
```

LINQ – Query Expression Basics

A ***query*** is a set of instructions that describes what data to retrieve from a given data source (or sources) and what type and organization the returned data should have. A ***query expression*** is a query expressed in ***query*** syntax.

The source data is organized logically as a sequence of elements of the same kind. For example:

- a SQL database table contains a sequence of rows.
- In an XML file, there is a "sequence" of XML elements
- An ‘in-memory’ collection contains a sequence of objects.

A ***query expression*** must begin with a ***from*** clause and must end with a ***select*** or ***group*** clause. Between the first ***from*** clause and the last ***select*** or ***group*** clause, it can contain one or more of: ***where, orderby, join, let*** and even more ***from*** clauses. You can also use the ***into*** keyword to enable the result of a ***join*** or ***group*** clause to serve as the source for additional ***query*** clauses in the same query expression.

From an application's viewpoint, the specific ***type*** and structure of the original source data is not important.

The application always sees the source data as an *IEnumerable<T>* or *IQueryable<T>* collection.

LINQ – Query Expression Examples

https://docs.microsoft.com/en-us/dotnet/csharp/linq/query-expression-basics
https://docs.microsoft.com/en-us/dotnet/csharp/linq/query-expression-basics#what-is-a-query-expression

---

A query can:

1. Retrieve a subset of the elements to produce a new sequence without modifying the individual elements. The query may then sort or group the returned sequence in various ways
2. Retrieve a sequence of elements but transform them to a new type of object.
3. Retrieve a singleton value about the source data, such as:
   - The number of elements that match a certain condition.
   - The element that has the greatest or least value.
   - The first element that matches a condition, or the sum of particular values in a specified set of elements.

```
IEnumerable<int> highScoresQuery =
    from score in scores
    where score > 80
    orderby score descending
    select score;
```

```
IEnumerable<string> highScoresQuery2 =
    from score in scores
    where score > 80
    orderby score descending
    select $"The score is {score}";
```

```
int highScoreCount =
    (from score in scores
     where score > 80
     select score)
     .Count();
```

LING – Query Variables

https://docs.microsoft.com/en-us/dotnet/csharp/linq/query-expression-basics#query-variable

A ***query variable*** is any variable that stores a ***query*** instead of the result of a ***query***.

A ***query variable*** is always an ***enumerable type*** that will produce a sequence of elements when it is iterated over in a ***foreach*** statement or a direct call to its ***IEnumerator.MoveNext*** method.

```
static void Main()
{
    // Data source.
    int[] scores = { 90, 71, 82, 93, 75, 82 };

    // Query Expression.
    IEnumerable<int> scoreQuery = //query variable
        from score in scores //required
        where score > 80 // optional
        orderby score descending // optional
        select score; //must end with select or group

    // Execute the query to produce the results
    foreach (int testScore in scoreQuery)
    {
        Console.WriteLine(testScore);
    }
}
// Outputs: 93 90 82 82
```

LINQ – Additional Practice

Starting a query expression

select clause

Filtering, ordering, and joining

orderby clause

let clause

Ending a query expression

*Continuations with "**into**“*

where clause

join clause

Subqueries in a query expression

LINQ – Method Expressions

https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/linq/query-syntax-and-method-syntax-in-linq

As a rule when you write ***LINQ*** queries, it is recommended to use ***query syntax*** whenever possible and ***method syntax*** whenever necessary.

Some queries must be expressed as method calls. Such as:

- to retrieve the number of elements that match a specified condition.
- to retrieve the element that has the maximum value in a source sequence.

```
class QueryVMethodSyntax
{
    static void Main()
    {
        int[] numbers = { 5, 10, 8, 3, 6, 12};

        //Query syntax:
        IEnumerable<int> numQuery1 =
            from num in numbers
            where num % 2 == 0
            orderby num
            select num;

        //Method syntax:
        IEnumerable<int> numQuery2 = numbers.Where(num => num % 2 == 0).OrderBy(n => n);

        foreach (int i in numQuery1)
        {
            Console.Write(i + " ");
        }
        Console.WriteLine(System.Environment.NewLine);
        foreach (int i in numQuery2)
        {
            Console.Write(i + " ");
        }

        // Keep the console open in debug mode.
        Console.WriteLine(System.Environment.NewLine);
        Console.WriteLine("Press any key to exit");
        Console.ReadKey();
    }
}
/*
    Output:
    6 8 10 12
    6 8 10 12
 */
```

LINQ – Activity

https://docs.microsoft.com/en-us/dotnet/csharp/tutorials/working-with-linq

---

1. Complete the tutorial at the above link.
2. Then change all queries in Method Syntax.