A LINQ (Language-Integrated Query) a C# egyik legfontosabb eszköze, amely lehetővé teszi az adatok szűrését, rendezését és manipulálását gyűjteményekben. A LINQ kifejezések lambda kifejezésekkel kombinálva rendkívül rugalmasak és hatékonyak.

LINQ alapfogalmak

1. **Forrás**: Egy olyan adatgyűjtemény, mint például lista, tömb vagy adatbázis.
2. **Lekérdezés**: Olyan műveletek halmaza, amelyekkel az adatokat szűrhetjük, rendezhetjük vagy aggregálhatjuk.
3. **Végrehajtás**: A lekérdezés végrehajtása az IEnumerable vagy más típusú eredmény létrehozására.

LINQ példák lambda kifejezésekkel

1. Egyszerű szűrés lista elemeire

```
using System;

using System.Collections.Generic;

using System.Linq;


class Program

{

    static void Main()

    {

        List<int> numbers = new List<int> { 1, 2, 3, 4, 5, 6 };


        // Csak a páros számok szűrése

        var evenNumbers = numbers.Where(n => n % 2 == 0).ToList();


        Console.WriteLine("Páros számok:");

        evenNumbers.ForEach(n => Console.WriteLine(n));

    }

}
```

2. Rendezés

```
List<string> names = new List<string> { "Anna", "Béla", "Cecília", "Attila" };


// Rendezés név szerint
```

```
var sortedNames = names.OrderBy(name => name).ToList();


Console.WriteLine("Rendezett nevek:");
sortedNames.ForEach(name => Console.WriteLine(name));
```

3. Több feltétel szűrésére

```
List<int> numbers = new List<int> { 10, 15, 20, 25, 30 };


// Olyan számok, amelyek oszthatók 5-tel, de nem 10-zel
var filteredNumbers = numbers.Where(n => n % 5 == 0 && n % 10 != 0).ToList();


Console.WriteLine("Számok, amelyek oszthatók 5-tel, de nem 10-zel:");
filteredNumbers.ForEach(n => Console.WriteLine(n));
```

4. Csoportosítás

```
List<string> animals = new List<string> { "kutya", "macska", "kígyó", "kecske", "mókus" };


// Csoportosítás az első betű alapján
var groupedAnimals = animals.GroupBy(animal => animal[0]);


foreach (var group in groupedAnimals)
{
    Console.WriteLine($"Csoport: {group.Key}");
    foreach (var animal in group)
    {
        Console.WriteLine($"  {animal}");
    }
}
```

5. Leképezés (Projection)

```
List<string> words = new List<string> { "alma", "banán", "citrom" };


// Szavak hossza
var wordLengths = words.Select(word => new { Word = word, Length = word.Length }).ToList();
```

```
Console.WriteLine("Szavak és hosszuk:");
wordLengths.ForEach(item => Console.WriteLine($"{item.Word}: {item.Length} karakter"));
```

6. Összegzés

```
List<int> numbers = new List<int> { 10, 20, 30, 40 };

// Az elemek összege
int sum = numbers.Sum();
Console.WriteLine($"Az elemek összege: {sum}");
```

7. Adatok kombinálása (Join)

```
var students = new List<(int Id, string Name)>
{
    (1, "Anna"),
    (2, "Béla"),
    (3, "Cecília")
};

var grades = new List<(int StudentId, string Grade)>
{
    (1, "A"),
    (2, "B"),
    (3, "C")
};

// Diákok és jegyek összekapcsolása
var studentGrades = students.Join(
    grades,
    student => student.Id,
    grade => grade.StudentId,
    (student, grade) => new { student.Name, grade.Grade }
);
```

```
Console.WriteLine("Diákok és jegyeik:");
foreach (var sg in studentGrades)
{
    Console.WriteLine($"{sg.Name}: {sg.Grade}");
}
```

Hasznos LINQ metódusok

- **Where**: Szűrés feltételek alapján.
- **Select**: Adatok transzformálása.
- **OrderBy**, **OrderByDescending**: Rendezés növekvő vagy csökkenő sorrendben.
- **GroupBy**: Elemek csoportosítása.
- **Join**: Gyűjtemények összekapcsolása.
- **Sum**, **Average**, **Count**: Aggregálási műveletek.