LINQ Programming in C#

Introduction to LINQ

Language Integrated Query (LINQ) is a powerful feature in C# that provides a unified way to query collections, databases, XML, and other data sources. It simplifies data manipulation and retrieval using a consistent query syntax.

Why Use LINQ?

- **Improved Readability**: Query syntax resembles SQL, making it easier to understand.
- **Type Safety**: LINQ queries are checked at compile time.
- **Less Code**: Reduces the amount of boilerplate code compared to traditional loops and data processing methods.
- **Flexibility**: Works with various data sources like collections, databases (Entity Framework), XML, and JSON.

Basic LINQ Syntax

LINQ queries can be written in two styles:

1. **Query Syntax** (similar to SQL)
2. **Method Syntax** (uses extension methods)

1. Query Syntax

```
int[] numbers = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };

var evenNumbers = from num in numbers
                  where num % 2 == 0
                  select num;

foreach (var num in evenNumbers)
{
    Console.WriteLine(num);
}
```

2. Method Syntax

```
var evenNumbers = numbers.Where(num => num % 2 == 0);
foreach (var num in evenNumbers)
{
    Console.WriteLine(num);
}
```

LINQ Query Operations

LINQ provides several query operations to manipulate data:

1. Filtering (`Where`)

Filters elements based on a condition.

```
var filtered = numbers.Where(n => n > 5);
```

2. Ordering (`OrderBy`, `OrderByDescending`)

Sorts elements in ascending or descending order.

```
var sorted = numbers.OrderBy(n => n);
var sortedDesc = numbers.OrderByDescending(n => n);
```

3. Projection (`Select`)

Transforms each element into a new form.

```
var squares = numbers.Select(n => n * n);
```

4. Aggregation (`Sum`, `Count`, `Average`, `Max`, `Min`)

Performs calculations on data.

```
int sum = numbers.Sum();
double avg = numbers.Average();
int count = numbers.Count();
```

5. Grouping (`GroupBy`)

Groups elements based on a key.

```
var grouped = numbers.GroupBy(n => n % 2 == 0 ? "Even" : "Odd");
```

LINQ with Collections

LINQ can be applied to different data structures like arrays, lists, and dictionaries.

Example with **List**:

```
List<string> names = new List<string> { "Alice", "Bob", "Charlie", "David" };
var shortNames = names.Where(n => n.Length <= 4).ToList();
```

LINQ with Entity Framework (Database)

LINQ is commonly used with databases through Entity Framework.

```
using (var context = new SchoolContext())
{
    var students = from s in context.Students
                   where s.Age > 18
                   select s;

    foreach (var student in students)
    {
        Console.WriteLine(student.Name);
    }
}
```

Conclusion

- LINQ simplifies data manipulation in C#.
- It supports both **Query Syntax** and **Method Syntax**.
- Can be used with collections, databases, XML, and other data sources.
- Provides powerful operations like filtering, ordering, grouping, and joining.