🔷 LINQ Kya Hai?

LINQ (Language Integrated Query) ek feature hai C# ka, jiske zariye aap **data ko query** kar saktay ho — chahe wo ho:

- List
- Array
- Database (SQL)
- XML
- ya kisi bhi IEnumerable ya IQueryable cheez se

Ye SQL jaise queries aapko C# ke andar likhne deta hai — aur bohat readable aur short hoti hain.

🔷 LINQ Ke Do Tareeqay

1. Query Syntax (SQL jaisa style)

csharp

CopyEdit

```
var result = from num in numbers
             where num > 5
             orderby num
             select num;
```

Matlab: numbers list mein se wo numbers chuno jo 5 se baray hain, unko order karo, aur result do.

2. Method Syntax (Lambda wali style)

csharp

CopyEdit

```
var result = numbers
             .Where(num => num > 5)
             .OrderBy(num => num)
             .Select(num => num);
```

Ye same kaam karta hai, bas different likhne ka style hai.

🔷 LINQ ke Mashhoor Functions

Function	**Kaam**
Where	Filter karta hai (condition lagata hai)
Select	Har item ko kisi naye form mein nikaalta hai
OrderBy	Choti se bari value tak sort karta hai
OrderByDescending	Bari se choti value tak sort karta hai
First, FirstOrDefault	Pehla matching item nikaalta hai
Any	Check karta hai ke koi item condition puray karta hai ya nahi
All	Check karta hai ke sab item condition puray kartay hain ya nahi
Count	Total matching items ginata hai
GroupBy	Cheezon ko group karta hai
Join	Do lists ya tables ko join karta hai

🔷 **Choti Si Example (Roman Urdu mein samajhaya gaya)**

csharp

CopyEdit

```
List<int> numbers = new List<int> { 1, 2, 3, 4, 5, 6, 7, 8, 9 };

var evenNumbers = numbers
    .Where(n => n % 2 == 0) // Wo numbers jo 2 se divide ho jaate hain
    .OrderBy(n => n)        // Ascending order mein karo
    .ToList();              // List bana lo

foreach (var num in evenNumbers)
{
    Console.WriteLine(num); // Har number print karo
}
```

Output:

CopyEdit

2

4

6

8

Agar aap chaho ke main aapko **LINQ with List of Objects** (jaise Students ya Employees) samjhaoon, to woh bhi kar sakta hoon. Aap bolo sirf.

You said:

IEnumerable vs IQuerable me kya farq huta ha

🔷 Simple Farq (Roman Urdu mein):

Feature	**IEnumerable**	**IQueryable**
Kaha use hota hai?	In-memory collections (jaise List, Array)	Remote data sources (jaise Database)
Execution kab hoti hai?	Jab aap loop chalao (foreach)	Jab query run hoti hai database par
Filtering kahan hoti hai?	**Memory** mein	**Database** mein (SQL ke through)
Performance	Choti data ke liye theek	Badi data ke liye zyada efficient
Supported by	System.Collections	System.Linq
Suitable for	Lists, Arrays, XML	SQL Databases, Entity Framework

---

🔷 Thora Detail mein:

📌 1. IEnumerable:

- Ye **collection ko enumerate** karta hai (yaani item by item read karta hai).
- Filtering/Sorting sab kuch **memory mein hota hai**.
- Example:

csharp

CopyEdit

List<string> names = new List<string> { "Ali", "Ahmed", "Zara" };

IEnumerable<string> result = names.Where(n => n.StartsWith("A"));

📌 **2. IQueryable:**

- Ye **queryable interface** hai jo queries ko **delay** karta hai jab tak aap usay execute nahi karte.
- Ye query ko **SQL** banata hai jab aap Entity Framework ya LINQ to SQL use karte ho.
- Filtering **direct SQL level pe hoti hai**, isliye **faster** hota hai for large datasets.
- Example:

csharp

CopyEdit

IQueryable<Student> students = dbContext.Students.Where(s => s.Age > 18);

🔷 **Real Life Example**

Suppose: 1 Million Students database mein hain

❌ **IEnumerable:**

csharp

CopyEdit

IEnumerable<Student> students = dbContext.Students.ToList(); // 1M records memory mein

var result = students.Where(s => s.Age > 18); // filtering ab ho rahi hai

➡ **Bohat slow** hoga, kyun ke poori data pehle memory mein load ho gayi.

✅ **IQueryable:**

csharp

CopyEdit

IQueryable<Student> students = dbContext.Students.Where(s => s.Age > 18);

➡ Ye sirf wahi students fetch karega **jo Age > 18** hain — aur directly SQL query chalayega.

🔷 **Summary (1 Line mein)**

🔷 IEnumerable → Memory mein kaam karta hai
🔷 IQueryable → SQL/database level par kaam karta hai (efficient)

💡 LINQ Fast Kyun Hota Hai? — Asaan Zubaan Mein

✅ 1. Kam Kaam Karta Hai

LINQ sirf **wahi data process karta hai jo zaroori hota hai**.

Jaise: Agar aap kehdo "mujhe pehla even number do", to wo **poori list nahi dekhta**, jaise hi pehla milta hai, kaam ruk jaata hai.

csharp

CopyEdit

```
numbers.Where(n => n % 2 == 0).First();
```

➡️ Ye sirf pehla even number dekh ke ruk jaata hai. **Fast!**

✅ 2. Der Se Kaam Karta Hai (Jab zarurat ho)

LINQ kaam **tabhi karta hai jab aap result chaho** — pehle nahi.

Jaise: Aapne order diya "juice lao", magar jab tak aap nahi bolte **"ab do"**, banda wait karta hai. 😄

✅ 3. Database Wala Kaam Database Ko Deta Hai

Agar aap database use kar rahe ho (like SQL), to LINQ query ko **SQL query mein convert kar deta hai** — aur kaam **server pe hota hai**, C# pe nahi.

Matlab: Agar aap kehte ho "wo students do jin ki age 18 se zyada hai", to LINQ yeh kaam **database ko bolta hai**, aur **sirf zaroori log lata hai**.

csharp

CopyEdit

```
var students = db.Students.Where(s => s.Age > 18);
```

➡️ Poora database memory mein nahi aata — sirf filtered data aata hai. **Zyada fast!**

✅ 4. Functions Line Mein Chalte Hain

LINQ ke functions (Where, Select, OrderBy) **pipeline ki tarah** chalte hain — ek ke baad ek, **tezi se**.

Jaise: Factory mein saman banana — har machine ka apna kaam, koi rukawat nahi.

✅ 5. Code Chhota, Fast, Aur Readable Hota Hai

Traditional loop se zyada **short aur samajhne mein asaan hota hai**, aur performance bhi acchi hoti hai (especially with large data).

🔚 Final Summary (1 Line)

LINQ is fast kyunke wo **sirf zaroori data ko, zaroori waqt pe, smart tareeqay se process karta hai** — na zyada, na kam.

🔶 Delegate Kya Hota Hai?

Delegate ek "pointer to function" hota hai.

Matlab:

- Jaise variables data store karte hain
- Waise **delegate kisi method ka address store karta hai**
- Aap method ko **indirectly call karte ho** through delegate

---

🔶 Asaan Lafzon Mein:

"Delegate ek tareeqa hai method ko variable ki tarah treat karne ka."

🔷 Real Life Example:

Jaise aap kehte ho:
"Jo banda message print karta hai, usko bulao"
Aap naam nahi le rahe, sirf kaam define kar rahe ho.

Delegate bhi yehi karta hai — method ko naam se nahi, **signature se represent** karta hai.

🔷 Basic Syntax:

csharp

CopyEdit

```
// 1. Delegate define karo

public delegate void MyDelegate(string msg);


// 2. Method banao jiska signature match karta ho
```

```
public void ShowMessage(string message)
{
   Console.WriteLine("Message: " + message);
}

// 3. Use delegate
MyDelegate del = ShowMessage;
del("Hello from delegate!");
```

🔷 Output:

vbnet

CopyEdit

```
Message: Hello from delegate!
```

🔷 Types of Delegates

Type	Description
Single-cast	Sirf 1 method hold karta hai
Multi-cast	Ek se zyada methods call karta hai
Func<>	Method return karta ho
Action<>	Method kuch return na kare
Predicate<>	Method true/false return kare

🔷 Func, Action, Predicate (Built-in Delegates)

✅ Action (kuch return nahi karta):

csharp

CopyEdit

```
Action<string> greet = name => Console.WriteLine("Hello, " + name);
greet("Moaz");
```

✅ **Func (return karta hai):**

csharp

CopyEdit

```
Func<int, int, int> add = (a, b) => a + b;
int result = add(5, 3);  // result = 8
```

✅ **Predicate (bool return karta hai):**

csharp

CopyEdit

```
Predicate<int> isEven = n => n % 2 == 0;
bool check = isEven(4);  // true
```

🔷 Delegate Kyun Use Karte Hain?

Wajah	**Explanation**
🔄 **Runtime par method select karna**	Flexibility milti hai kis method ko run karna hai
🧩 **Loosely coupled code**	Code modules independent rehtay hain
⚙️ **Event handling**	Events aur delegates ka strong connection hota hai
📦 **Callbacks**	Kisi kaam ke baad method ko wapis call karna

🔚 Final Summary (1 line mein):

Delegate C# ka tareeqa hai method ko variable ki tarah treat karne ka — jisse aap flexible aur smart code likh sako.

🔷 **Short Answer (Simple Roman Urdu):**

Delegate khud stack mein ho sakta hai,
lekin uske andar jo method ka reference hota hai, wo heap ya method area mein hota hai.

Yani:

🔶 **Delegate ek variable hota hai (stack)**
🔶 **Woh method ka reference store karta hai (jo heap/method area mein hota hai)**

Lambda Expression

🔷 **Lambda Expression Kya Hota Hai?**

Lambda expression ek **short tareeqa** hai **anonymous method** (yaani naam ke bina function) likhne ka.

Simple Lafzon Mein:

"Lambda = Choti function wali line"
Jo aap kisi variable, delegate, ya LINQ mein use karte ho.

🔶 **Syntax:**

csharp

CopyEdit

```
(parameters) => expression_or_statements
```

Example:

csharp

CopyEdit

```
x => x * x
```

Yani: **Ek function jo x ko square karta hai**.

🔷 **Example with Explanation**

✅ **Traditional Method:**

csharp

CopyEdit

```
int Square(int x)
{
   return x * x;
}
```

✅ **Lambda Expression:**

csharp

CopyEdit

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

Same kaam — **lekin chhoti line mein**.

🔷 **Lambda in Real Use (jaise LINQ mein):**

csharp

CopyEdit

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


var even = numbers.Where(n => n % 2 == 0);
```

- n => n % 2 == 0
  Matlab: **Aise numbers chuno jo even hoon**

🔷 **Components of Lambda**

Part	Example	Explanation
Parameters	x	Input (jaise function ka parameter)
Arrow	=>	"Goes to" (ye batata hai ke function kya karega)
Body	x * x	Jo kaam karna hai (return value ya statement)

🔷 **Different Forms**

✅ **1. One-line expression**

csharp

CopyEdit

```
x => x + 10
```

✅ **2. With multiple parameters**

csharp

CopyEdit

```
(x, y) => x + y
```

✅ **3. With statement block**

csharp

CopyEdit

```
(x, y) =>
{
   int sum = x + y;
   return sum;
}
```

🔷 **Where Lambda Expressions Use Hote Hain?**

Use Case	**Example**
LINQ	.Where(n => n > 5)
Delegates	Action<string> greet = msg => Console.WriteLine(msg);
Events	button.Click += (s, e) => { /* code */ };
Anonymous methods	Without defining named functions

🔚 **Final Summary (1 Line):**

Lambda expression ek short tareeqa hai function likhne ka — bina naam ke, ek line mein.

✅ **Q1: Kya Lambda Delegate mein aa sakta hai? Aur kya Delegate Lambda ke jaisa kaam kar sakta hai?**

🔷 **Answer:**

Jee haan! Lambda aur Delegate ek dusray ke sath kaam kar saktay hain.

📌 **Lambda = Delegate ka short form hai.**

Example 1: Lambda passed into delegate

csharp

CopyEdit

```
Action<string> show = msg => Console.WriteLine(msg);
show("Hello");  // ✅ Lambda expression delegate me use hua
```

Example 2: Delegate assigned to Lambda

csharp

CopyEdit

```
delegate int MyDelegate(int x);


MyDelegate square = x => x * x;
Console.WriteLine(square(5));  // Output: 25
```

💡 **Conclusion:**

- **Lambda = Anonymous Function = Delegate**
- Lambda ko aap kisi delegate, Action, Func, Predicate mein use kar saktay ho

---

✅ **Q2: Reference ayega to kya iska matlab delegate hai?**

🔷 **Answer:**

Har reference delegate nahi hota.

- Reference ka matlab sirf itna hai: **kisi cheez ka address ya pointer**
- Lekin agar **reference ek method ka ho**, to haan — **wo delegate ho sakta hai**

Example:

csharp

CopyEdit

void ShowMessage(string msg) => Console.WriteLine(msg);

Action<string> del = ShowMessage; // ✅ Yeh method ka reference hai => delegate

💡 Jab aap = lagate ho kisi method ko assign karte waqt, aur uska signature match karta hai, to **C# usay delegate mein wrap kar deta hai**.

So:

✔️ **Method ka reference = Delegate**
❌ **Object ya value ka reference ≠ Delegate**

✅ **Q3: Mera pointer kahan hai? Kaise pata chalega?**

🔷 **Answer:**

C# mein **direct pointers** nahi hote by default (jaise C/C++ mein), lekin **reference** (yaani pointer-like behavior) hota hai.

Jab aap delegate ya lambda use karte ho, tab:

- Aapka **pointer us method/function ki taraf hota hai**
- Aur wo pointer **delegate object ke andar hota hai**

🔶 **Example Code:**

csharp

CopyEdit

```
delegate void MyDelegate(string msg);


void Print(string message)

{

   Console.WriteLine(message);

}


MyDelegate del = Print;  // yahan pointer method "Print" ki taraf hai
```

👉 **Yahan kya ho raha hai?**

Variable	Kya hai?	Memory Location
del	Delegate object ka reference	Stack
Print	Method ka pointer (code memory)	Code area
del ke andar	Print() ka pointer	Heap (delegate object)

🔷 Kaise pata chalega pointer kahan hai?

✅ 1. Debugger use karo:

- Breakpoint lagao del = Print; line pe
- Watch window mein dekho del.Method kya show kar raha hai
- Ye batayega ke **delegate kis function ko point kar raha hai**

csharp

CopyEdit

```
Console.WriteLine(del.Method.Name);    // Output: Print
Console.WriteLine(del.Target);         // null, agar static method ho
```

✅ 2. Reflection se bhi dekh sakte ho:

csharp

CopyEdit

```
Console.WriteLine(del.Method);  // Prints method info (pointer jesa)
```

🔚 Summary:

Question	Short Answer
Q1. Lambda in Delegate?	✅ Yes, Lambda is a delegate
Q2. Reference = Delegate?	❗ Sirf method reference = delegate
Q3. Pointer kahan hai?	🧠 Pointer delegate.Method ke andar hota hai (runtime pe check karo)