MA'LUMOTLARNI TEZ VA SAMARALI QAYTA ISHLASHDA LINQ OPERATORLARI VA SO'ROVLARIDAN FOYDALANISH.

Abdumutalova Husnida Saidahmad qizi
Farg'ona davlat universiteti 2-kurs talabasi
husnidaabdumutalova@gmail.com
Ismoilova Muslimaxon Axmadjon qizi
Farg'ona davlat universiteti 2-kurs talabasi
tursunovamuslimaxon05@gmail.com

Annotatsiya: *LINQ (Language Integrated Query) – .NET platformasida ma'lumotlar bilan ishlashni osonlashtiradigan va samarali qiladigan texnologiya. LINQ yordamida dasturchilar ro'yxatlar, ma'lumotlar bazalari, XML fayllari va boshqa ma'lumotlar manbalaridan ma'lumotlarni osonlik bilan olish, filtrlash, guruhlash, tartiblash va boshqa ko'plab operatsiyalarni bajarishlari mumkin. Ushbu maqolada LINQ so'rovlarining asosiy sintaksisi, operatorlari va ularni dasturlarda qanday samarali ishlatish mumkinligi haqida batafsil tushuntirilgan.*

Kalit So'zlar:*LINQ.NET platformasi, Ma'lumotlarni qayta ishlash, SQL, So'rov sintaksisi, Metod sintaksisi, Ma'lumotlar bazasi, Filtrlash, Guruhlash, Kod optimizatsiyasi, LINQ operatorlari*

Abstract: LINQ (Language Integrated Query) is a technology in the .NET platform that simplifies and enhances data handling. With LINQ, developers can easily retrieve, filter, group, sort, and perform other complex operations on various data sources such as lists, databases, XML files, and more. This article provides an in-depth explanation of the basic syntax, operators, and how to effectively use LINQ queries in programming.

Key words: *LINQ.NET platform, Data Processing, SQL (Structured Query Language), Query Syntax, Method Syntax, Database, Filtering, Grouping, Code Optimization, LINQ Operators.*

Kirish. Dasturlashda ma'lumotlar bilan ishlash har doim murakkab va vaqt talab qiladigan jarayon bo'lib kelgan. Biroq, LINQ (Language Integrated Query) texnologiyasi yordamida bu jarayon sezilarli darajada soddalashtirilgan. LINQ, .NET platformasida ishlab chiqilgan, ma'lumotlarni qayta ishlashni osonlashtiradigan va samarali qiladigan kuchli vositadir. Uning yordamida turli manbalardan (ro'yxatlar, ma'lumotlar bazalari, XML fayllari va boshqalar) ma'lumotlarni osonlik bilan olish, filtrlash, guruhlash va tartiblash mumkin. LINQ o'zining qulay sintaksisi va kuchli imkoniyatlari bilan dasturchilar uchun ma'lumotlar bilan ishlashni yanada osonlashtiradi.

Ma'lumotlarni qayta ishlash va tahlil qilish, ayniqsa katta hajmdagi ma'lumotlar bilan ishlashda, dasturchilar uchun muhim vazifa hisoblanadi. Ko'pincha, ma'lumotlar

bazalaridan, fayllardan yoki xotiradan ma'lumotlarni olib, ular ustida turli amallarni bajarish kerak bo'ladi. **LINQ (Language Integrated Query)** - bu .NET platformasida ma'lumotlar bilan ishlashni soddalashtirish va tezlashtirishga qaratilgan kuchli vositadir. LINQ, ma'lumotlarga so'rovlar yuborishni dasturlash tiliga to'liq integratsiya qilganligi sababli, dasturchilar turli xil ma'lumot manbalaridan (masalan, ma'lumotlar bazasi, xotira kolleksiyalari, XML fayllari) ma'lumotlarni olish, filtrlash, guruhlash, saralash va boshqa tahlillarni oson va samarali tarzda amalga oshirishi mumkin.

Bu maqolada, **LINQ operatorlari** va **so'rovlarining** ma'lumotlarni qayta ishlashdagi o'rni va samaradorligi haqida so'z boradi.

LINQ — bu ma'lumotlar bilan ishlash uchun ishlab chiqilgan dasturlash texnologiyasidir. LINQ, SQL so'rovlariga o'xshash sintaksisga ega bo'lib, u dasturlash tilida ma'lumotlarni qayta ishlashni imkoniyatini beradi. LINQning eng katta afzalligi shundaki, uni turli xil manbalardan ma'lumotlarni olishda, ularni filtrlash, tartiblash, guruhlash va boshqa ko'plab operatsiyalarni bajarishda foydalanish mumkin.

LINQning asosiy afzalliklari:

1. **Kodning o'qilishi osonlashadi**: SQL tiliga o'xshash sintaksis kodni yanada o'qilishi oson va tushunarli qiladi.
2. **Ma'lumot manbalariga keng qo'llanilishi**: LINQ faqat ro'yxatlar bilan cheklanmay, turli ma'lumotlar manbalariga qo'llanilishi mumkin. Masalan, ma'lumotlar bazalari, XML fayllari, massivlar va boshqa tuzilmalar.
3. **Xatolarni kamaytiradi**: LINQ tizimi avtomatik tarzda so'rovlarni bajaradi, bu esa xatolarni kamaytiradi va kodni yaxshilaydi.
4. **Samarali va qulay**: LINQ o'zining integratsiyalashgan tizimi bilan ma'lumotlar bilan ishlashni soddalashtiradi va samaradorlikni oshiradi.

LINQ Operatorlari va So'rovlarining Ahmiyati

LINQ operatorlari va so'rovlarining ma'lumotlarni tez va samarali qayta ishlashda foydalanishning asosiy sababi - ularning intuitiv va to'g'ridan-to'g'ri kod yozish imkoniyatini yaratishidir. **LINQ** orqali ma'lumotlarga so'rov yuborish juda oson, chunki u to'g'ridan-to'g'ri **C#** yoki **VB.NET** kabi dasturlash tillariga integratsiyalashgan. Bu esa, o'z navbatida, kodni o'qish va tushunishni sezilarli darajada soddalashtiradi.

LINQ operatorlari va so'rovlarining asosiy afzalliklari quyidagilardan iborat:

- **Kodni soddalashtirish**: LINQ yordamida kompleks ma'lumotlar tahlilini bir qator oddiy va tushunarli satrlar bilan bajarish mumkin.
- **Samaradorlik**: LINQ tomonidan taqdim etilgan optimizatsiya qilish imkoniyatlari ma'lumotlar bazasi yoki xotiradagi ma'lumotlar bilan ishlashni ancha tezlashtiradi.
- **Tizimga moslashuvchanlik**: LINQ bir nechta ma'lumot manbalariga (SQL, XML, xotira kolleksiyalari va boshqalar) so'rov yuborishni qo'llab-quvvatlaydi, bu esa tizimni kengaytirish va moslashtirish imkoniyatlarini oshiradi.

LINQ Operatorlari Va So'rovlarining Samaradorligi

LINQ operatorlari nafaqat soddaligi bilan, balki samaradorligi bilan ham ajralib turadi. **LINQ** so'rovlarini ishlatish orqali, ma'lumotlarni qayta ishlash jarayonida yuqori samaradorlikka erishish mumkin. **LINQ** ning optimizatsiya qilingan ishlash jarayoni quyidagi afzalliklarni taqdim etadi:

- **Ma'lumotlar bazasidan optimal foydalanish**: LINQ to'liq SQL tilida so'rov yuboradi, bu esa kerakli ma'lumotlarni tezda olishni ta'minlaydi. Shunday qilib, faqat kerakli ma'lumotlar bazasiga so'rov yuboriladi va shartli tahlil amalga oshiriladi.
- **Xotira sarfini kamaytirish**: LINQ yordamida katta ma'lumotlar to'plamlarini xotirada saqlash va qayta ishlash o'rniga, so'rovlarni to'g'ridan-to'g'ri manba orqali bajarish mumkin.
- **Parallel ishlash imkoniyati**: LINQ Parallel (PLINQ) yordamida ma'lumotlar ustida parallel ishlashni qo'llab-quvvatlaydi, bu esa ishlash tezligini yanada oshiradi.

LINQ So'rovlarini Yaratish-LINQ so'rovlarini yaratishning ikkita asosiy usuli mavjud: Query Syntax (so'rov sintaksisi) va Method Syntax (metodlar orqali)

LINQning **query syntax** usuli SQL tiliga o'xshash bo'lib, so'rovlar from, where, select, orderby kabi operatorlar yordamida yoziladi. Quyidagi misolda, faqat juft sonlarni olish uchun so'rov yozilgan:

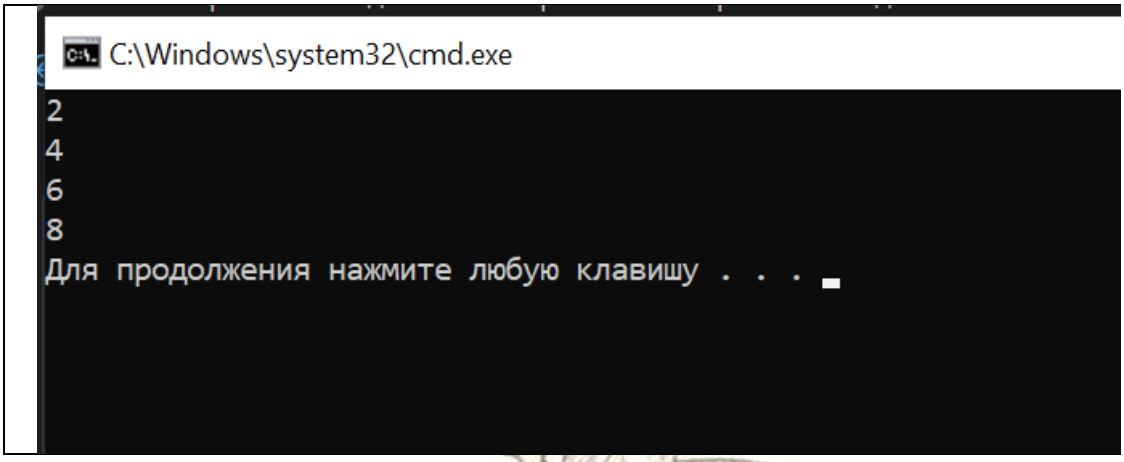
C#dagi kodi

```
using System;
using System.Linq;
using System.Collections.Generic;
class Program
{
   static void Main()
   {
      List<int> numbers = new List<int> { 1, 2, 3, 4, 5, 6, 7, 8, 9 };

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

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

Dastur natijasi:



Method Syntax (Metodlar Orqali)

Method syntax usulida LINQ metodlari yordamida so'rovlar yaratiladi. Bu usul ko'proq metodlar va lambda ifodalari ishlatiladi, bu esa kodni qisqa va aniq qiladi. Quyidagi misolda juft sonlar Where() metodi yordamida olinadi:

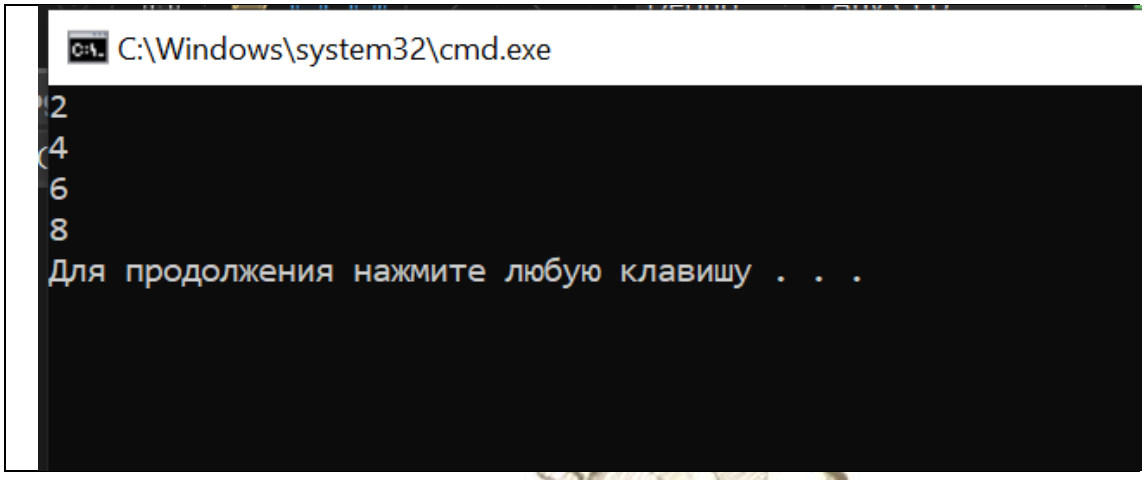
C# dagi kodi:

```
using System;
using System.Linq;
using System.Collections.Generic;
class Program
{
   static void Main()
   {
      List<int> numbers = new List<int> { 1, 2, 3, 4, 5, 6, 7, 8, 9 };

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

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

Dastur natijasi:



LINQ Operatorlari

LINQ so'rovlarida ma'lumotlar ustida turli operatsiyalarni amalga oshiruvchi ko'plab operatorlar mavjud. Ba'zi eng mashhur operatorlar quyidagilardir:

1. **Where**: Ma'lumotlar to'plamidan ma'lum shartlarga mos keladigan elementlarni tanlash uchun ishlatiladi.
2. **Select**: Ma'lumotlar to'plamidan tanlangan elementlarni qaytaradi.
3. **Orderby**: Ma'lumotlarni tartiblash uchun ishlatiladi.
4. **Group by**: Elementlarni ma'lum mezonlar bo'yicha guruhlash.
5. **Join**: Ikki yoki undan ortiq ma'lumotlar to'plamini birlashtirish.
6. **Distinct**: Takroriy elementlarni olib tashlash.
7. **Count**: Elementlar sonini hisoblash.
8. **Sum, Average**: Ma'lumotlar to'plamining yig'indisi yoki o'rtacha qiymatini hisoblash.

Quyida shu operatorlar yordamida bir nechta masalalarni yechimini C# dasturlash tilida korib o’tamiz:

1-masala:Bizda talabalar ro'yxati mavjud. Har bir talabada nomi va yoshi bor. Talabalar ro'yxatidan quyidagi amallarni bajarish talab etiladi:

1. 18 yoshdan katta bo'lgan talabalarni aniqlang.
2. Talabalarni ismlari bo'yicha saralang.
3. Har bir sinf (A, B, C kabi) uchun talabalarni guruhlang.
4. Faqat talabalar ismlarini ko'rsating.
5. Har bir talabani o'qiyotgan kursi bilan bog'lang. Kurslar ro'yxatiga qarab talabalar va kurslar o'rtasidagi bog'lanishni aniqlang.
6. Talabalardan hech bo'lmaganda biror biri 18 yoshdan katta ekanligini tekshirib ko'ring.
7. Birinchi 2 talabani va keyingi 2 talabani oling.

C# dagi kodi:

```
using System;
using System.Collections.Generic;
```

```
using System.Linq;
class Student
{
    public string Name { get; set; }
    public int Age { get; set; }
    public int CourseID { get; set; }
    public string Grade { get; set; }
}
class Course
{
    public int CourseID { get; set; }
    public string CourseName { get; set; }
}
class Program
{
    static void Main()
    {
        // Talabalar ro'yxati
        var students = new List<Student>
        {
            new Student { Name = "Ali", Age = 20, CourseID = 1, Grade = "A" },
            new Student { Name = "Vera", Age = 17, CourseID = 2, Grade = "B" },
            new Student { Name = "John", Age = 22, CourseID = 1, Grade = "A" },
            new Student { Name = "Sara", Age = 19, CourseID = 3, Grade = "C" }
        };
        // Kurslar ro'yxati
        var courses = new List<Course>
        {
            new Course { CourseID = 1, CourseName = "Mathematics" },
            new Course { CourseID = 2, CourseName = "Physics" },
            new Course { CourseID = 3, CourseName = "Chemistry" }
        };
        // 1. 18 yoshdan katta bo'lgan talabalarni aniqlash
        var adultStudents = students.Where(s => s.Age > 18);
        Console.WriteLine("18 yoshdan katta bo'lgan talabalar:");
        foreach (var student in adultStudents)
        {
            Console.WriteLine(student.Name);  // Ali, John
        }
        // 2. Talabalarni ismlariga ko'ra saralash
```

```
    var sortedStudents = students.OrderBy(s => s.Name);
    Console.WriteLine("\nTalabalar ismlariga ko'ra saralangan:");
    foreach (var student in sortedStudents)
    {
        Console.WriteLine(student.Name);  // Ali, John, Sara, Vera
    }
    // 3. Talabalarni sinflariga ko'ra guruhlash
    var groupedByGrade = students.GroupBy(s => s.Grade);
    Console.WriteLine("\nTalabalar sinf darajasiga ko'ra guruhlangan:");
    foreach (var group in groupedByGrade)
    {
        Console.WriteLine($"Grade: {group.Key}");
        foreach (var student in group)
        {
            Console.WriteLine(student.Name);
        }
    }
    // 4. Faqat talabalar ismlarini olish
    var studentNames = students.Select(s => s.Name);
    Console.WriteLine("\nFqat talabalar ismlarini ko'rsatish:");
    foreach (var name in studentNames)
    {
        Console.WriteLine(name);  // Ali, Vera, John, Sara
    }
    // 5. Talabalar va kurslarni bog'lash
    var studentCourses = from student in students
                         join course in courses on student.CourseID equals course.CourseID
                         select new { student.Name, course.CourseName };

    Console.WriteLine("\nTalabalar va ularning kurslari:");
    foreach (var sc in studentCourses)
    {
        Console.WriteLine($"{sc.Name} is enrolled in {sc.CourseName}");
    }

    // 6. Talabalar orasida 18 yoshdan katta bo'lgan bo'lsa, tekshirish
    bool hasAdult = students.Any(s => s.Age > 18);
    Console.WriteLine($"\nTalabalardan hech bo'lmaganda birortasi 18 yoshdan katta:
{hasAdult}");
```

```
        // 7. Birinchi 2 talabani va keyingi 2 talabani olish
        var firstTwo = students.Take(2);
        Console.WriteLine("\nBirinchi 2 talaba:");
        foreach (var student in firstTwo)
        {
            Console.WriteLine(student.Name);  // Ali, Vera
        }

        var nextTwo = students.Skip(2).Take(2);
        Console.WriteLine("\nKeyingi 2 talaba:");
        foreach (var student in nextTwo)
        {
            Console.WriteLine(student.Name);  // John, Sara
        }
    }
}
```

Dastur natijasi:

```
18 yoshdan katta bo'lgan talabalar:
Ali
John
Sara
Talabalar ismlariga ko'ra saralangan:
Ali
John
Sara
Vera
Talabalar sinf darajasiga ko'ra guruhlangan:
Grade: A
Ali
John
Grade: B
Vera
Grade: C
Sara
Faqat talabalar ismlarini ko'rsatish:
Ali
Vera
John
Sara
Talabalar va ularning kurslari:
```

```
Ali is enrolled in Mathematics
Vera is enrolled in Physics
John is enrolled in Mathematics
Sara is enrolled in Chemistry
Talabalardan hech bo'lmaganda birortasi 18 yoshdan katta: True
Birinchi 2 talaba:
Ali
Vera
Keyingi 2 talaba:
John
Sara
```

Xulosa. LINQ – bu .NET platformasida ma'lumotlar bilan ishlashda sezilarli darajada samaradorlikni oshiradigan kuchli vosita. LINQ yordamida ma'lumotlarni olish, tartiblash, filtrlash va guruhlashni oson va samarali tarzda amalga oshirish mumkin. LINQning intuitiv sintaksisi, yuqori darajadagi operatsiyalarni amalga oshirish imkoniyati, va turli ma'lumot manbalariga moslashuvchanligi uni dasturchilar orasida keng tarqalgan va mashhur vositaga aylantirgan.

Dasturchilar LINQ yordamida yuqori samarali va o'qilishi oson kodlar yozishlari mumkin. Bu esa dastur ishlab chiqishda tezlik va samaradorlikni oshiradi. LINQni o'z dasturlarida qo'llash, ayniqsa katta hajmdagi ma'lumotlar bilan ishlashda, juda foydali bo'lishi mumkin.

FOYDALANILGAN ADABIYOTLAR:

1. "C# 9.0 and .NET 5 – Modern Cross-Platform Development" – Mark J. Price
2. "Pro LINQ: Language Integrated Query in C# 2008" – Scott Mitchell
3. "LINQ Programming: Querying the Data" – Rafał Kuczynski
4. "C# in Depth" – Jon Skeet
5. "C# 7.0 in a Nutshell" – Joseph Albahari, Ben Albahari
6. "C# 8.0 and .NET Core 3.0 – Modern Cross-Platform Development" – Mark J. Price
7. "Mastering LINQ" – Ian Griffiths
8. "Beginning C# 7 Programming with Visual Studio 2017" – Benjamin Perkins, Jacob Vibe Hammer
9. "C# Programming: From Problem Analysis to Program Design" – Barbara Doyle
10. "Professional C# 7 and .NET Core 2.0" – Christian Nagel
11. "C# Dasturlash Tili Asoslari" – O'zbekcha qo'llanma
12. "C# va .NET Asosida Dasturlash" – Shavkatbek Tohirov
13. "C# va LINQ" – Rustamov Akram (O'zbek tilidagi LINQ operatorlari va amaliyotlariga oid kitob)
14. "C# Dasturlash Asoslari" – Jamshidbek Khamraev
15. "C# va .NET Asoslari: O'qituvchilar uchun qo'llanma" – Saida Toshpulatova
16. "Dasturlash Asoslari: C# va LINQ" – Firdavs Ahrorov