# Тема 10. Linq

## 10.1 Основы Linq

LINQ (Language-Integrated Query) — это набор расширений языка C# и Visual Basic, позволяющий писать запросы к различным источникам данных, таким как коллекции объектов, базы данных, XML и другие.

LINQ обеспечивает удобный и выразительный способ выполнения запросов к данным, используя знакомый синтаксис, похожий на SQL, но интегрированный непосредственно в язык программирования.

Пример использования LINQ для выполнения запроса к коллекции объектов:

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

var query = from num in numbers

where num > 2

select num;

foreach (var num in query)

{ Console.WriteLine(num); // результат: 3, 4, 5

}

Этот код выбирает числа больше 2 из коллекции numbers с использованием LINQ-запроса и выводит их

## 10.2 LINQ для множеств

Linq можно использовать для множеств (рис. 28-29).

void Button1Click(object sender, EventArgs e)

{

E=U.Except(B).ToArray();

foreach (int i in E)

textBox6.Text+=i+",";

}

void Button2Click(object sender, EventArgs e)

{

F=E.Intersect(C).ToArray();

foreach (int i in F)

textBox7.Text+=i+",";

}

Изображение выглядит как текст, круг, диаграмма, снимок экрана

Автоматически созданное описание

1. Демонстрация Linq для множеств

Изображение выглядит как текст, диаграмма, круг, Шрифт

Автоматически созданное описание

1. Демонстрация Linq для множеств

## 10.3 Сортировка с Linq

LINQ позволяет выполнять сортировку данных с использованием метода OrderBy или OrderByDescending.

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

var sortedNumbers = numbers.OrderBy(num => num); // сортировка по возрастанию

foreach (var num in sortedNumbers)

{

Console.Write(num + " "); // результат: 1 1 2 3 4 5 5 6 9

}

var reverseSmbers = numbers.OrderByDescending(num => num); // сортировка по убыв.

foreach (var num in reverseSmbers)

{

Console.Write(num + " "); // результат: 9 6 5 5 4 3 2 1 1

}

## 10.4 Методы Linq

Простейшее определение запроса LINQ выглядит следующим образом:

from переменная in набор\_объектов

select переменная;

Методы:

* **Select**: определяет проекцию выбранных значений
* **Where**: определяет фильтр выборки
* **OrderBy**: упорядочивает элементы по возрастанию
* **OrderByDescending**: упорядочивает элементы по убыванию
* **ThenBy**: задает дополнительные критерии для упорядочивания элементов возрастанию
* **ThenByDescending**: задает дополнительные критерии для упорядочивания элементов по убыванию
* **Join**: соединяет две коллекции по определенному признаку
* **Aggregate**: применяет к элементам последовательности агрегатную функцию, которая сводит их к одному объекту
* **GroupBy**: группирует элементы по ключу
* **ToLookup**: группирует элементы по ключу, при этом все элементы добавляются в словарь
* **GroupJoin**: выполняет одновременно соединение коллекций и группировку элементов по ключу
* **Reverse**: располагает элементы в обратном порядке
* **All**: определяет, все ли элементы коллекции удовлятворяют определенному условию
* **Any**: определяет, удовлетворяет хотя бы один элемент коллекции определенному условию
* **Contains**: определяет, содержит ли коллекция определенный элемент
* **Distinct**: удаляет дублирующиеся элементы из коллекции
* **Except**: возвращает разность двух коллекцию, то есть те элементы, которые создаются только в одной коллекции
* **Union**: объединяет две однородные коллекции
* **Intersect**: возвращает пересечение двух коллекций, то есть те элементы, которые встречаются в обоих коллекциях
* **Count**: подсчитывает количество элементов коллекции, которые удовлетворяют определенному условию
* **Sum**: подсчитывает сумму числовых значений в коллекции
* **Average**: подсчитывает cреднее значение числовых значений в коллекции
* **Min**: находит минимальное значение
* **Max**: находит максимальное значение
* **Take**: выбирает определенное количество элементов
* **Skip**: пропускает определенное количество элементов
* **TakeWhile**: возвращает цепочку элементов последовательности, до тех пор, пока условие истинно
* **SkipWhile**: пропускает элементы в последовательности, пока они удовлетворяют заданному условию, и затем возвращает оставшиеся элементы
* **Concat**: объединяет две коллекции
* **Zip**: объединяет две коллекции в соответствии с определенным условием
* **First**: выбирает первый элемент коллекции
* **FirstOrDefault**: выбирает первый элемент коллекции или возвращает значение по умолчанию
* **ElementAt**: выбирает элемент последовательности по определенному индексу
* **Last**: выбирает последний элемент коллекции

Подробнее можно прочитать:

## 10.5 Лямбда-выражения

Лямбда-выражения в C# — это анонимные функции, которые могут быть использованы для создания краткого и читабельного кода, особенно при работе с делегатами и LINQ.

Лямбда-выражения — это анонимные методы, которые позволяют определить код для обработки события непосредственно в месте регистрации обработчика.

Лямбда-выражение в C# — это удобный способ определения анонимных методов, которые могут использоваться для создания простых функций, передаваемых в качестве аргументов в другие методы или для присваивания им переменных. Лямбда-выражения идеально подходят для работы с LINQ (Language Integrated Query) и являются ключевой частью функционального программирования в C#.

Основные принципы:

Синтаксис: (параметры) => выражение

(параметры) - список параметров, которые принимает лямбда-выражение.

=> - символ, разделяющий параметры от выражения.

выражение - тело лямбда-выражения, которое возвращает результат.

Тип возвращаемого значения: Тип возвращаемого значения определяется типом выражения. Если выражение не возвращает значение, лямбда-выражение имеет тип void.

Сокращенный синтаксис: если лямбда-выражение имеет только один параметр, можно опустить круглые скобки вокруг него.

Примеры:

1. Простой пример:

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

int result = square(5); // result = 25

2. Лямбда-выражение с несколькими параметрами:

Func<int, int, int> sum = (x, y) => x + y;

int result = sum(3, 4); // result = 7

3. Лямбда-выражение с блоком кода:

Func<string, string> greet = name => {

string message = "Привет, " + name + "!";

return message;

};

string greeting = greet("Иван"); // greeting = "Привет, Иван!"

4. Лямбда-выражение с void типом возвращаемого значения:

Action<string> printMessage = message => Console.WriteLine(message);

printMessage("Hello, world!"); // Вывод: Hello, world!

Преимущества использования лямбда-выражений:

-Краткий и чистый код: Лямбда-выражения позволяют писать код более компактно и читаемо.

-Простота использования: Лямбда-выражения легко создавать и использовать.

-Гибкость: Лямбда-выражения могут использоваться в различных контекстах, например, в делегатах, LINQ, асинхронных операциях.

Применение лямбда-выражений:

-Работа с делегатами: Лямбда-выражения идеально подходят для создания анонимных методов, которые могут быть переданы в делегаты.

-LINQ: Лямбда-выражения широко используются в LINQ для фильтрации, сортировки, группировки и других операций с данными.

-Асинхронные операции: Лямбда-выражения можно использовать для создания асинхронных методов.

-Обработка событий: Лямбда-выражения могут использоваться для обработки событий.

Пример использования лямбда-выражения с делегатом:

delegate int MyDelegate(int x);

static void Main(string[] args)

{

MyDelegate myDelegate = x => x 2;

int result = myDelegate(5); // result = 10

}

С помощью лямбда-выражений можно обрабатывать события

// Обработчик события (лямбда-выражение)

myObject.MyEvent += (sender, e) =>

{

Console.WriteLine("Событие MyEvent произошло!");

};

## Пример 10.1. Работа с Linq

Определить класс, в котором в одном поле хранится название товара, а в другом цена. Создать массив объектов этого класса.

a) Отсортировать элементы массива по цене товара.

b) Отобрать из массива элементы, в которых название товара начинается на заданную букву.

c) Выяснить, есть ли товар за указанную цену.

Исходный код можно найти в папке

<https://disk.yandex.ru/d/Hs0QHM1uXWs6yg>

Текст программы:

Класс «Tovar»

using System;

using **System**.Windows.Forms;

namespace TRPO1

{

public class Tovar

{

public string name;

public int price;

public Tovar (string xx, int yy) { name=xx; price=yy;}

}

}

Класс «MainForm»

**using System;**

**using System.Collections.Generic;**

**using System.Linq;**

**using System.Windows.Forms;**

**namespace TRPO1**

**{**

**public partial class MainForm : Form**

**{**

**public MainForm()**

**{**

**InitializeComponent();**

**for (int i=0; i<mas.Length;i++)**

**{**

**string s = mas[i].name.ToString()+", цена: "+mas[i].price.ToString();**

**listBox1.Items.Add (s);**

**}**

**}**

**Tovar [] mas={ new Tovar("Яблоки", 60),new Tovar("Оливки",90),new Tovar("Садовое кресло",4000),**

**new Tovar("Пачка бумаги",244), new Tovar ("Масло",44), new Tovar ("Полотенце",600),**

**new Tovar ("Спички",13), new Tovar ("Шампунь",333), new Tovar ("Шампунь",313)};**

**void Button1Click(object sender, EventArgs e)**

**{ //Сортировка**

**IEnumerable<Tovar> sequence = mas.OrderBy(n => n.price);**

**int p=0;**

**foreach (Tovar d in sequence)**

**{**

**mas[p]=d;**

**p++;**

**}**

**listBox1.Items.Clear();**

**for (int i=0; i<mas.Length;i++)**

**{**

**string s = mas[i].name.ToString()+", цена: "+mas[i].price.ToString();**

**listBox1.Items.Add (s);**

**}**

**}**

**void Button2Click(object sender, EventArgs e)**

**{ //Поиск начинающегося на букву**

**listBox2.Items.Clear();**

**String t=textBox4.Text;**

**IEnumerable<Tovar> sequence = mas.Where(n => (n.name.Substring(0,1).Equals(t)));**

**string s="";**

**// MessageBox.Show(sequence.Count().ToString());**

**foreach (Tovar h in sequence) {**

**s = h.name.ToString()+", цена: "+h.price.ToString();**

**listBox2.Items.Add (s);**

**}**

**}**

**void Button3Click(object sender, EventArgs e)**

**{ //Изменение**

**try {**

**mas[listBox1.SelectedIndex]=new Tovar((textBox1.Text),Convert.ToInt16(textBox2.Text));**

**listBox1.Items.Clear();**

for (int i=0; i<mas.Length;i++)

{

string s = mas[i].name.ToString()+", цена: "+mas[i].price.ToString();

listBox1.Items.Add (s);

}

textBox1.Text="";

textBox2.Text="";

}

catch ( System.FormatException)

{

MessageBox.Show ("Неверный ввод");

}

catch (System.I**ndexOutOfRangeException)**

**{**

**MessageBox.Show ("Выберите элемент");**

**}**

**}**

**void ListBox1SelectedIndexChanged(object sender, EventArgs e)**

**{**

**int n=listBox1.SelectedIndex;**

**textBox1.Text=Convert.ToString(mas[n].name);**

**textBox2.Text=Convert.ToString(mas[n].price);**

**}**

**void Button4Click(object sender, EventArgs e)**

**{ //Поиск товара с ценой**

**label1.Text = "";**

**int nt;**

**try {**

**nt=Convert.ToInt16(textBox3.Text);**

**IEnumerable<Tovar> seq = mas.Where(x => x.price==nt);**

**if (seq.Any())**

**foreach (Tovar t in seq)**

**label1.Text = t.name;**

**else**

**label1.Text="Нет такого товара";**

**} catch (Exception) {**

**MessageBox.Show("Неверный ввод");**

**} }**

**} }**

Результаты выполнения программы представлены на рис. 30-34.

Изображение выглядит как текст, программное обеспечение, Значок на компьютере, Мультимедийное программное обеспечение

Автоматически созданное описание

1. Форма после запуска

Изображение выглядит как текст, программное обеспечение, Значок на компьютере, Мультимедийное программное обеспечение

Автоматически созданное описание

1. Сортировка

Изображение выглядит как текст, программное обеспечение, Значок на компьютере, Мультимедийное программное обеспечение

Автоматически созданное описание

1. Отбор элементов

Изображение выглядит как текст, программное обеспечение, Значок на компьютере, Мультимедийное программное обеспечение

Автоматически созданное описание

1. Поиск товара

## Задания по теме

ЛР 6.

Написать программу с использованием Linq-запросов по инд. заданию. На оценку 4 и 5 нужен графический интерфейс.

Варианты заданий

1. Определить класс, предназначенный для работы с деньгами. В этом классе должно быть два поля. В первом поле хранятся рубли, во втором копейки. Создать массив объектов этого класса.

a) Отсортировать элементы массива по полю рубль;

b) Отобрать из массива элементы, в которых хранится сумма денег в два раза больше заданного пользователем значения. Обеспечить пользователю ввод указанной суммы.

c) Выяснить, есть ли в массиве элементы, у которых значения полей совпадают (например, 5 руб. 5 коп.)

2) Определить класс, предназначенный для работы с деньгами. В этом классе должно быть два поля. В первом поле хранятся рубли, во втором копейки. Создать массив объектов этого класса.

a) Отсортировать элементы массива по возрастанию;

b) Отобрать из массива элементы, в которых хранится сумма денег меньше заданного пользователем значения. Обеспечить пользователю ввод указанной суммы.

c) Найти сумму всех денег, хранящихся в массиве.

3) Определить класс, предназначенный для работы с деньгами. В этом классе должно быть два поля. В первом поле хранятся рубли, во втором копейки. Создать массив объектов этого класса.

a) Отсортировать элементы массива по поля копейки;

b) Отобрать из массива элементы, в которых нет копеек, т.е. значение соответствующего поля равно нулю.

c) Найти среднеарифметическое значение всех денег, хранящихся в массиве.

4) Определить класс, предназначенный для работы с двумерными векторами. Создать массив объектов этого класса.

a) Отсортировать элементы массива по длине векторов.

b) Отобрать из массива элементы, в которых значение первого поля больше чем второго.

c) Выяснить, есть ли в массиве вектора с двумя отрицательными координатами.

5) Определить класс, в котором хранится адрес (достаточно трех полей, в которых хранятся названии населенного пункта, улицы, и номер дома). Создать массив объектов этого класса.

a) Сгруппировать элементы, по населенным пунктам.

b) Выяснить принадлежит ли массиву адрес, введенный пользователем.

c) Отобрать элементы массива, у которых номер дома четное число.

6) Определить класс, в котором хранится адрес (достаточно трех полей, в которых хранятся названии населенного пункта, улицы, и номер дома)Создать массив объектов этого класса.

a) Сгруппировать элементы, по улицам.

b) Выяснить есть ли повторяющиеся элементы в массиве.

c) Отобрать элементы массива, у которых номер дома нечетное число.

7) Определить класс, в котором хранится адрес (достаточно трех полей, в которых хранятся названии населенного пункта, улицы, и номер дома)Создать массив объектов этого класса.

a) Упорядочить элементы массива по населенным пунктам.

b) Выяснить принадлежит ли массиву адрес, введенный пользователем.

c) Отобрать элементы массива, у которых номер дома больше заданного числа.

8) Определить класс, в котором в одном поле хранится название товара, а в другом цена. Создать массив объектов этого класса.

a) Отсортировать элементы массива по названию товара.

b) Сгруппировать элементы массива по ценовым промежуткам. Должно быть, хотя бы 3 промежутка (дорогие товары, товары, имеющие среднюю стоимость и дешевые товары).

c) Выяснить, есть ли, введенный пользователем товар и указать его стоимость.

9) Определить класс, в котором в одном поле хранится название товара, а в другом цена. Создать массив объектов этого класса.

a) Отсортировать элементы массива по цене товара.

b) Отобрать из массива элементы, в которых название товара заканчинается на заданную букву.

c) Выяснить, есть ли товар за указанную цену.

10) Определить класс, в котором в одном поле хранится название товара, а в другом цена. Создать массив объектов этого класса.

a) Отсортировать элементы массива по названию товара.

b) Сгруппировать элементы массива по ценовым промежуткам. Должно быть, хотя бы 3 промежутка (дорогие товары, товары, имеющие среднюю стоимость и дешевые товары).

c) Выяснить, сколько товаров имеет цену больше, заданного пользователем числа.