# N ЛАБОРАТОРНАЯ РАБОТА №N

**ЯЗЫК ЗАПРОСОВ LINQ**

## N.1 Цель работы

Приобретение практических навыков использования языка запросов LINQ.

## N.2 Краткие теоретические сведения

LINQ (Language-Integrated Query) представляет простой и удобный язык запросов к источнику данных. В качестве источника данных может выступать объект, реализующий интерфейс IEnumerable (например, стандартные коллекции, массивы), набор данных DataSet, документ XML. Но вне зависимости от типа источника LINQ позволяет применить ко всем один и тот же подход для выборки данных.

**N.2.1. Простейшие запросы**

Чтобы использовать функциональность LINQ, убедимся, что в файле подключено пространство имен System.LINQ.

Простейшее определение запроса LINQ выглядит следующим образом:

from переменная in набор\_объектов

select переменная;

Рассмотрим пример: без и с использованием LINQ.

У нас есть список имён, и нам надо выбрать элементы начинающиеся на букву ‘Д’.

Без LINQ программа будет выглядеть так:

string[] names = {"Дмитрий", "Игорь", "Иван", "Денис", "Дианисий", "Андрей"};

var selectedNames = new List<string>();

foreach(string s in names)

{

    if (s.ToUpper().StartsWith("Д"))

        selectedNames.Add(s);

}

selectedTeams.Sort();

foreach (string s in selectedNames)

    Console.WriteLine(s);

C LINQ:

string[] names = {"Дмитрий", "Игорь", "Иван", "Денис", "Дианисий", "Андрей"};

var selectedNames = from n in names // определяем каждый объект из namess как n

                    where n.ToUpper().StartsWith("Д") //фильтрация по критерию

                    orderby t  // упорядочиваем по возрастанию

                    select t; // выбираем объект

foreach (string s in selectedNames)

    Console.WriteLine(s);

Выражение **from n in names** проходит по всем элементам массива names и определяет каждый элемент как n. Используя переменную n мы можем проводить над ней разные операции.

Несмотря на то, что мы не указываем тип переменной n, выражения LINQ являются строго типизированными. То есть среда автоматически распознает, что набор names состоит из объектов string, поэтому переменная t будет рассматриваться в качестве строки.

Далее с помощью оператора where проводится фильтрация объектов, и если объект соответствует критерию (в данном случае начальная буква должна быть "Д"), то этот объект передается дальше.

Оператор **orderby** упорядочивает по возрастанию, то есть сортирует выбранные объекты.

Оператор **select** передает выбранные значения в результирующую выборку, которая возвращается LINQ-выражением.

В данном случае результатом выражения LINQ является объект **IEnumerable<T>**. Нередко результирующая выборка определяется с помощью ключевого слова **var**, тогда компилятор на этапе компиляции сам выводит тип.

Преимуществом подобных запросов также является и то, что они интуитивно похожи на запросы языка SQL, хотя и имеют некоторые отличия.

**N.2.2. Методы расширения LINQ**

Кроме стандартного синтаксиса **from .. in .. select** для создания запроса LINQ мы можем применять специальные методы расширения, которые определены для интерфейса **IEnumerable**. Как правило, эти методы реализуют ту же функциональность, что и операторы LINQ типа where или orderby.

Например:

string[] names = {"Дмитрий", "Игорь", "Иван", "Денис", "Дианисий", "Андрей"};

var selectedNames = names.Where(n=>n.ToUpper().StartsWith("Д")).OrderBy(n => n);

foreach (string s in selectedNames)

    Console.WriteLine(s);

Запрос будет аналогичен предыдущему. Он состоит из цепочки методов **Where** и **OrderBy**. В качестве аргумента эти методы принимают делегат или лямбда-выражение.

Не каждый метод расширения имеет аналог среди операторов LINQ, но в этом случае можно сочетать оба подхода. Например, используем стандартный синтаксис linq и метод расширения **Count(),** возвращающий количество элементов в выборке:

int number = (from n in names where n.ToUpper().StartsWith("Д") select t).Count();

## N.2.3. Фильтрация выборки и проекция

**Фильтрация**

Для выбора элементов из некоторого набора по условию используется метод Where. Например, выберем все четные элементы, которые больше 10.

Фильтрация с помощью операторов LINQ:

int[] numbers = { 1, 2, 3, 4, 10, 34, 55, 66, 77, 88 };

IEnumerable<int> evens = from i in numbers

where i%2==0 && i>10

select i;

foreach (int i in evens)

Console.WriteLine(i);

Здесь используется конструкция **from**: from i in numbers

Тот же запрос с помощью метода расширения:

int[] numbers = { 1, 2, 3, 4, 10, 34, 55, 66, 77, 88 };

IEnumerable<int> evens = numbers.Where(i => i % 2 == 0 && i > 10);

Если выражение в методе Where для определенного элемента будет равно true (в данном случае выражение i % 2 == 0 && i > 10), то данный элемент попадает в результирующую выборку.

**Проекция**

Проекция позволяет спроектировать из текущего типа выборки какой-то другой тип. Для проекции используется оператор select. Допустим, у нас есть набор объектов следующего класса, представляющего пользователя:

class User

{

public string Name { get;set; }

public int Age { get; set; }

}

Но нам нужен не весь объект, а только его свойство Name:

List<User> users = new List<User>();

users.Add(new User { Name = "Sam", Age = 43 });

users.Add(new User { Name = "Tom", Age = 33 });

var names = from u in users select u.Name;

foreach (string n in names)

Console.WriteLine(n);

Результат выражения LINQ будет представлять набор строк, поскольку выражение select u.Name выбирают в результирующую выборку только значения свойства Name.

Аналогично можно создать объекты другого типа, в том числе анонимного.

Так же любой полученный объект можно преобразовать в нужную коллекцию, например с помощью методоoList(), ToArray(), ToDictionary() и т.д.

## N.2.4. Сортировка

Для сортировки набора данных по возрастанию используется оператор **orderby**:

int[] numbers = { 3, 12, 4, 10, 34, 20, 55, -66, 77, 88, 4 };

var orderedNumbers = from i in numbers

orderby i

select i;

foreach (int i in orderedNumbers)

Console.WriteLine(i);

Оператор **orderby** принимает критерий сортировки. В данном случае в качестве критерия выступает само число.

По умолчанию оператор **orderby** производит сортировку по возрастанию. Однако с помощью ключевых слов **ascending** (сортировка по возрастанию) и **descending** (сортировка по убыванию) можно явным образом указать направление сортировки.

Вместо оператора **orderby** можно использовать методы расширения **OrderBy.**

## N.2.5. Объединение

Для объединения двух последовательностей используется метод Union. Его результатом является новый набор, в котором имеются элементы, как из первой, так и из второй последовательности. Повторяющиеся элементы добавляются в результат только один раз:

string[] soft = { "Microsoft", "Google", "Apple"};

string[] hard = { "Apple", "IBM", "Samsung"};

var result = soft.Union(hard);

foreach (string s in result)

Console.WriteLine(s);

Результатом операции будет следующий набор:

Microsoft

Google

Apple

IBM

Samsung

Если же нам нужно простое объединение двух наборов, то мы можем использовать метод **Concat**:

var result = soft.Concat(hard);

Те элементы, которые встречаются в обоих наборах, дублируются.

**N.2.3. Список используемых методов расширения LINQ**

* Select: определяет проекцию выбранных значений
* Where: определяет фильтр выборки
* OrderBy: упорядочивает элементы по возрастанию
* OrderByDescending: упорядочивает элементы по убыванию
* Join: соединяет две коллекции по определенному признаку
* 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
* Single: выбирает единственный элемент коллекции, если коллекция содердит больше или меньше одного элемента, то генерируется исключение
* ElementAt: выбирает элемент последовательности по определенному индексу
* Last: выбирает последний элемент коллекции

## N.3. Порядок выполнения лабораторной работы

### Задание

Разработать структуру по варианту задания а так же выполнить запрос LINQ

N.3.3 Разработать тестовые примеры;

N.3.4 Выполнить отладку программы;

N.3.5 Оформить отчет по проделанной работе.

## N.3 Варианты заданий

Вариант определяется согласно формуле: №варианта = №студента-в-группе mod 6.

### Вариант 1

У нас есть список всех преступников.

В преступнике есть поля: ФИО, заключен ли он под стражу, рост, вес, национальность.

Вашей программой будут пользоваться детективы.

У детектива запрашиваются данные (рост, вес, национальность), и детективу выводятся все преступники, которые подходят под эти параметры, но уже заключенные под стражу выводиться не должны.

### Вариант 2

У вас есть список больных

Класс больного состоит из полей: ФИО, возраст, заболевание.

Требуется написать программу больницы, в которой перед пользователем будет меню со следующими пунктами:

1)Отсортировать всех больных по фио

2)Отсортировать всех больных по возрасту

3)Вывести больных с определенным заболеванием

(название заболевания вводится пользователем с клавиатуры)

### Вариант 3

В нашей великой стране Арстоцка произошла амнистия!

Всех людей, заключенных за преступление "Антиправительственное", следует исключить из списка заключенных.

Есть список заключенных, каждый заключенный состоит из полей: ФИО, преступление.

Вывести список до амнистии и после.

### Вариант 4

У нас есть список всех игроков. У каждого игрока есть поля: имя, уровень, сила. Требуется написать запрос для определения топ 3 игроков по уровню и топ 3 игроков по силе, после чего вывести каждый топ.

2 запроса получится.

### Вариант 5

Есть набор тушенки. У тушенки есть название, год производства и срок годности.

Написать запрос для получения всех просроченных банок тушенки.

Можно считать только года без месяцев.

### Вариант 6

Есть 2 списка с солдатами.

Всех бойцов из отряда 1, у которых фамилия начинается на букву Б, требуется перевести в отряд 2.

### Вариант 7

Есть группа студентов. Каждый студент имеет ФИО, средний балл и количество пропусков.

Условие получение зачёта: средний балл больше 2.5, количество пропусков меньше 4. Вывести студентов получивших зачёт и их средний балл.

### Вариант 8

Есть 2 группы студентов.

После репрессивного отчисления студентов в группах осталось мало студентов, поэтому их следует объединить. Создать запрос для объединения двух групп студентов.

### Вариант 9

Имеется рота бойцов. Каждый солдат имеет ФИО, звание, рост.

Требуется создать запрос который выстраивает бойцов по росту. Вывести ФИО и звания отсортированные по росту.

### Вариант 10

Есть список футболистов. Каждый футболист имеет ФИО и количество забитых голов.

Создать запрос для нахождения 3х лучших и 3х худших футболистов. Вывести топы на экран.

## N.5 Содержание отчета о выполнении лабораторной работы

Титульный лист, цель работы, вариант задания, текст программы с комментариями, описание тестовых примеров и выводы по проделанной работе.

## N.6 Контрольные вопросы

1. Что такое LINQ?
2. Для чего нужен LINQ?
3. Какие преимущества LINQ перед стандартными средствами языка?
4. Какие способы задания запроса LINQ бывают?
5. Назовите методы расширения LINQ которые знаете.