# Изучение технологии [LINQ to Objects](http://habrahabr.ru/post/81592/) с использованием синтаксиса запросов

## 1.1 Основные концепции технологии LINQ to Objects

**LINQ** − это набор расширений языка, поддерживающий формирование запросов данных с контролем типов.

Технология LINQ (Language Integrated Query) предназначена для обработки (для организации запросов и преобразований) практически любого источника данных, начиная от массивов, файлов, строк, коллекций объектов .NET Framework, баз данных SQL Server, наборов данных ADO.NET (DataSet) и XML-документов.

LINQ упрощает ситуацию, предлагая стандартные шаблоны для работы с данными в различных видах источников и различных форматов. Стандартные шаблоны включают в себя основные операции запросов LINQ: фильтрация, упорядочение, группировка, соединение, выбор (проецирование), статистическая обработка. По форме синтаксис языка LINQ очень похож на язык запросов SQL.

LINQ to Objects − простейшая форма LINQ − позволяет формулировать и осуществлять запросы к коллекциям и объектам, находящимся в памяти. LINQ to Objects позволяет заменить итеративную логику (такую как блок foreach) декларативным выражением LINQ.

Базовые единицы данных в LINQ − последовательности.

**Последовательность** − любая коллекция, реализующая интерфейс IEnumerable<> или IQueryable<>.

**Оператор запроса** − метод, преобразующий последовательность.

**Запрос** − выражение, которое преобразует последовательность с помощью операторов запроса.

Оператор запроса никогда не изменяет входную последовательность, вместо этого он возвращает новую.

В LINQ присутствует важная симметрия. Выражения LINQ работают с объектами, реализующими IEnumerable<T>, и возвращают выражения LINQ объекты, реализующие Enumerable<T>. Поэтому можно передать результат одного выражения LINQ другому выражению LINQ и т.д. Такая цепочка выражений LINQ вычисляется в конце, когда выполняется итерация по конечным данным. В зависимости от типа источника данных, LINQ часто может “сплавить” цепочку выражений в одну операцию, и таким образом, выполнить ее наиболее эффективным способом.

Для использования LINQ необходимо подключить пространство имен System.Linq и наличие Framework 3.5 или выше

## 1.2 Синтаксис языка LINQ

LINQ поддерживает два вида синтаксиса:

1. синтаксис запросов и
2. синтаксис методов

В этой лабораторной работе будем использовать синтаксис запросов.  
Синтаксис запросов:

Код C#

int[] msv = { -1, 2, 3, -4 };

var res = from n in msv

          where n > 0

          select n;

// 2, 3

Такой синтаксис всегда должен начинаться с from in и заканчиваться select или

group.

**Ключевые слова синтаксиса запросов**

**from** - Указание источника данных и переменной диапазона (аналогично переменной итерации).

**where** - Фильтрация исходных элементов на основе одного или нескольких логических выражений, разделенных операторами AND и OR ( && или || ).

**select** - Указание типа и формы, которую будут иметь элементы в возвращенной последовательности при выполнении запроса.

**group** - Группировка результатов запроса по заданному ключевому значению.

**into** - Предоставление идентификатора, который может служить ссылкой на результаты предложения join, groupили select.

**orderby** - Сортировка результатов запроса по возрастанию или убыванию в компараторе по умолчанию для типа элемента.

**join** - Соединение двух источников данных на основе вычисления равенства между двумя заданными парными критериями.

**let** - Ввод переменной диапазона для хранения результатов вложенного выражения в выражении запроса.

**in** - Контекстно-зависимое ключевое слово в предложении join.

**on** - Контекстно-зависимое ключевое слово в предложении join.

**equals** - Контекстно-зависимое ключевое слово в предложении join.

**by** - Контекстно-зависимое ключевое слово в предложении group.

**ascending** - Контекстно-зависимое ключевое слово в предложении orderby.

**descending** - Контекстно-зависимое ключевое слово в предложении orderby.

**Ключевые слова синтаксиса методов**

**Объединение**

**Aggregate** - Применяет к последовательности пользовательский метод.

**Average**- Вычисляет среднее для числовой последовательности.

**Count** - Возвращает количество элементов в последовательности (целочисленное значение).

**LongCount**- Возвращает количество элементов в последовательности (значение в диапазоне LongInt).

**Min** - Возвращает наименьшее значение для числовой последовательности.

**Max** - Возвращает наибольшее значение для числовой последовательности.

**Sum** - Складывает члены числовой последовательности.

**Конкатенация**

**Concat** - Соединяет две последовательности в одну.

**Преобразование**

**Cast**- Преобразует элементы последовательности в элемены указанного типа.

**OfType**- Выбирает из элементов последовательности элемены указанного типа.

**ToArray** - Возвращает массив из элементов последовательности.

**ToDictionary** - Возвращает словарь из элементов последовательности.

**ToList** - Возвращает список из элементов последовательности.

**ToLookup** - Возвращает результаты поиска по последовательности.

**ToSequence** - Возвращает последовательность IEnumerable.

**Элемент**

**DefaultIfEmpty** - Создает стандартный элемент для пустой последовательности.

**ElementAt**- Возвращает элемент последовательности по указанному индексу.

**ElementAtOrDefault**- Возвращает элемент по указанному индексу или стандартный элемент (если индекс вышел за пределы диапазона).

**Firs**t - Возвращает первый элемент последовательности.

**FirstOrDefault** - Возвращает первый элемент последовательности или стандартный элемент (если нужный элемент не найден).

**Last**- Возвращает последний элемент последовательности.

**LastOrDefault** - Возвращает последний элемент последовательности или стандартный элемент (если нужный элемент не найден).

**Single** - Возвращает единственный элемент последовательности.  
SingleOrDefault - Возвращает единственный элемент последовательности или стандартный элемент (если нужный элемент не найден).

**Равенство**

**SequenceEqual** - Проверяет эквивалентность двух последовательностей.

**Создание**

**Empty** - Создает пустую последовательность.

**Range** - Создает последовательность в соответствии с заданным диапазоном.

**Repea**t - Создает последовательность, повторяя значение заданное количество раз.

**Группировка**

**GroupBy** - Группирует элементы последовательности указанным образом.

**Присоединение**

**GroupJoin** - Выполняет группированное соединение двух последовательностей.

**Join** - Выполняет внутреннее соединение двух последовательностей.

**Упорядочение**

**OrderBy** - Упорядочивает элементы последовательности по заданным значениям в порядке возрастания.

**OrderByDescending** - Упорядочивает элементы последовательности по заданным значениям в порядке убывания.

**ThenBy** - Упорядочивает элементы уже упорядоченной последовательности в порядке возрастания.

**ThenByDescending** - Упорядочивает элементы уже упорядоченной последовательности в порядке убывания.

**Reverse** - Зеркально отображает порядок расположения элементов в последовательности.

**Разделение на части**

**Skip** - Возвращает последовательность, в которой указанное число элементов пропущено.

**SkipWhile** - Возвращает последовательность, в которой пропущены элементы, не соответствующие указанному условию.

**Take** - Возвращает последовательность, в которую включается указанное число элементов.

**TakeWhile** - Возвращает последовательность, в которую включаются элементы, соответствующие указанному условию.

**Проекция**

**Select** - Создает проекцию части последовательности.

**SelectMany** - Создает проекцию части последовательности по принципу «один ко многим».

**Кванторы**

**All** - Определяет соответствие всех элементов последовательности указанным условиям.

**Any** - Определяет, есть ли в последовательность элементы, удовлетворяющие указанным условиям.

**Contains** - Определяет, есть ли в последовательности указанный элемент.

**Ограничение**

**Where** - Сортирует члены последовательности.

**Множества**

**Distinct** - Возвращает последовательность без повторяющихся элементов.

**Except** - Возвращает последовательность, представляющую собой разность двух других последовательностей.

**Intersect** - Возвращает последовательность, представляющую собой пересечение двух других последовательностей.

**Union** - Возвращает последовательность, представляющую собой объединение двух других последовательностей.

## 1.2 Выполнение LINQ-запроса

Все операции запроса *LINQ* состоят из трех различных действий.

1. получение источника данных;
2. создание запроса;
3. выполнение запроса.

В первую очередь в запросе *LINQ* нужно указать источник данных. В C#, как и в большинстве языков программирования, переменная должна быть объявлена до ее использования. В запросе *LINQ* первым идет предложение from для указания источника данных ( customers ) и переменная диапазона ( cust ):

//queryAllCustomers – это IEnumerable<Customer>

var queryAllCustomers = from cust in customers

select cust;

Переменная диапазона схожа с переменной итерации в цикле foreach за исключением того, что в выражении запроса не происходит фактической итерации. При выполнении запроса переменная диапазона будет использоваться как ссылка на каждый последующий элемент в customers. Поскольку компилятор может определить тип cust, нет необходимости указывать его в явном виде.

Выполнение запроса разделяют на отложенное и принудительное (немедленное).

В запросе, возвращающем последовательность значений, переменная запроса никогда не содержит результаты запроса, а только хранит его команды. Выполнение запроса откладывается, пока переменная запроса используется в циклах foreach или For Each. Это называется **отложенным выполнением**, то есть выполнение запроса происходит спустя некоторое время после его создания. Это означает, что запрос можно выполнять настолько часто, насколько это необходимо.

Запрос с отложенным выполнением выполняется повторно при повторном переборе элементов.

В отличие от отложенного выполнения запросов, возвращающих последовательность значений, запросы, возвращающие одноэлементное значение, выполняются немедленно.

Примерами одноэлементных запросов являются Average, Count, First и Max. Они выполняются немедленно, потому что запрос должен создать последовательность, чтобы вычислить одноэлементный результат. Можно также принудительно вызвать немедленное выполнение. Это может оказаться полезным в тех случаях, когда результаты запроса необходимо поместить в кэш. Чтобы немедленно выполнить запрос, который не возвращает одноэлементное значение, можно вызвать метод ToList<TSource>, ToDictionary или ToArray<TSource>запроса или переменной запроса.

## 1.3 Примеры

Несколько примеров запросов LINQ to Objects.

1. Найти сотрудников, фамилии которых начинаются с определенной буквы:

IEnumerable<EmployeeDetails> matches;

matches = from employee in employees

where employee.LastName.StartsWith("D")

select employee;

2. Отфильтровать продукты дороже определенной пороговой цены:

IEnumerable<Product> matches;

matches = from product in products

where product.UnitsInStock > 0 && product.UnitPrice > 3.00M

select product;

3. Пример сортировки:

IEnumerable<EmployeeDetails> matches;

matches = from employee in employees

orderby employee.LastName, employee.FirstName

select employee;

4. В следующем примере возвращается новый анонимный тип, включающий значение ключа группы и количество объектов в группе. При этом используется встроенный вызов метода по имени Count().

var matches = from employee in employees

group employee by employee.TitleOfCourtesy into g

select new {Title = g.Key, Employees = g.Count()};

5. Пример немедленного выполнения.

using (AdventureWorksEntities context = new AdventureWorksEntities())

{

ObjectSet<Product> products = context.Products;

Product[] prodArray = (

from product in products

orderby product.ListPrice descending

select product).ToArray();

Console.WriteLine("Every price from highest to lowest:");

foreach (Product product in prodArray)

{

Console.WriteLine(product.ListPrice);

}

}

6. Частотный анализ текста

string s = "LINQ is a set of extensions to the .NET Framework that

encompass language-integrated query, set, and transform

operations. It extends C# and Visual Basic with native language

syntax for queries and provides class libraries to take advantage

of these capabilities.";

var res = from n in s.ToLower().ToCharArray()

          group n by n into g

          let count = g.Count()

          orderby count descending

          select new

          {

              Letter = g.Key,

              Count = count

          };

foreach (var item in res)

{

    Console.WriteLine(item);

}

Результат:

{ Letter =  , Count = 39 }

{ Letter = a, Count = 26 }

{ Letter = e, Count = 26 }

{ Letter = t, Count = 22 }

{ Letter = s, Count = 20 }

{ Letter = i, Count = 17 }

{ Letter = n, Count = 17 }

{ Letter = o, Count = 12 }

{ Letter = r, Count = 12 }

{ Letter = l, Count = 7 }

{ Letter = d, Count = 7 }

{ Letter = g, Count = 6 }

{ Letter = f, Count = 5 }

{ Letter = c, Count = 5 }

{ Letter = u, Count = 5 }

{ Letter = h, Count = 4 }

{ Letter = p, Count = 4 }

{ Letter = v, Count = 4 }

{ Letter = q, Count = 3 }

{ Letter = x, Count = 3 }

{ Letter = ., Count = 3 }

{ Letter = m, Count = 3 }

{ Letter = b, Count = 3 }

{ Letter = w, Count = 2 }

{ Letter = k, Count = 2 }

{ Letter = y, Count = 2 }

{ Letter = ,, Count = 2 }

{ Letter = -, Count = 1 }

{ Letter = #, Count = 1 }

7. Запрос к массиву объектов класса с проекцией на массив

В следующем примере показан простой запрос к [ArrayList](http://msdn.microsoft.com/ru-ru/library/system.collections.arraylist(v=vs.90).aspx). Обратите внимание, что в этом примере при вызове кодом метода [Add](http://msdn.microsoft.com/ru-ru/library/system.collections.arraylist.add(v=vs.90).aspx) используются инициализаторы объектов, но это не является обязательным.

using System;

using System.Collections;

using System.Linq;

namespace NonGenericLINQ

{

public class Student

{

public string FirstName { get; set; }

public string LastName { get; set; }

public int[] Scores { get; set; }

}

class Program

{

static void Main(string[] args)

{

ArrayList arrList = new ArrayList();

arrList.Add(new Student{FirstName="Svetlana", LastName ="Omelchenko", Scores = new int[] { 98, 92, 81, 60 }});

arrList.Add(new Student{FirstName = "Claire", LastName = "O’Donnell",

Scores = new int[] { 75, 84, 91, 39 }});

arrList.Add(new Student{FirstName = "Sven", LastName = "Mortensen",

Scores = new int[] { 88, 94, 65, 91 }});

arrList.Add(new Student{FirstName = "Cesar", LastName = "Garcia",

Scores = new int[] { 97, 89, 85, 82 }});

var query = from Student student in arrList

where student.Scores[0] > 95

select student;

foreach (Student s in query)

Console.WriteLine(s.LastName + ": " + s.Scores[0]);

// Keep the console window open in debug mode.

Console.WriteLine("Press any key to exit.");

Console.ReadKey();

}

}

}

/\* Output:

Omelchenko: 98

Garcia: 97

\*/

Литература

<http://www.intuit.ru/department/internet/mwebtech/10/3.html>

<http://msdn.microsoft.com/ru-ru/library/bb397926(v=vs.100).aspx>

<http://www.k-press.ru/cs/2007/2/linq/linq.asp>

## САМОСТОЯТЕЛЬНАЯ РАБОТА

## Задания

**ОБЩИЕ ДАННЫЕ ДЛЯ ВСЕХ ВАРИАНТОВ:**

**Даны:**

**1. Массивы:**

**A={ “Falluot 3”, “Daxter 2”, “System Shok 2”, “Morrowind”, “Pes 2013”}.**

**B={ 2, -7, -10, 6, 7, 9,3}**

**C={“Light Green”, “Red”, “Green”, “Yellow”, “Purple”, “Dark Green”, “Light Red”, “Dark Red”, “Dark Yellow”, “Light Yellow”}**

**2. Классы:**

**Car, содержащий максимальную скорость, цвет и производителя, число пассажиров.**

**Product, содержащий название продукта, количество упаковок на складе, описание.**

**3. Списки(контейнеры):**

**List<string> myCars = new List<String> {"Yugo", "Aztec", "BMW"};**

**List<string> yourCars = new List<String>{"BMW", "Saab", "Aztec" };**

**Варианты**

**1.**

1. **Написать запрос для нахождения єлементов массива А включающих пробел.**
2. **Написать запрос для нахождения отрицательных єлементов массива В.**
3. **Написать запрос для нахождения всех оттенков зеленого цвета в массиве С.**
4. **Создать список автомобилей класса Car. Выбрать из списка автомобили BMW и вывести всю информацию о них.**
5. **Создать список продуктов класса Product. Выбрать из списка продукты, которые закончились и вывести всю информацию о них.**
6. **Вывести результирующий набор списков myCars и yourCars, содержащий общие элементы.**

**2.**

1. **Написать запрос для нахождения єлементов массива А включающих цифры.**
2. **Написать запрос для нахождения положительных єлементов массива В.**
3. **Написать запрос для нахождения всех оттенков красного цвета в массиве С.**
4. **Создать список автомобилей класса Car. Выбрать из списка автомобили максимальная скорость которых превышает 200 км/час и вывести всю информацию о них.**
5. **Создать список продуктов класса Product. Выбрать из списка продукты, ассортимент которых превышает 50 единиц и вывести всю информацию о них.**
6. **Вывести результирующий набор списков myCars и yourCars, содержащий все элементы без повторений.**

**3.**

1. **Написать запрос для нахождения єлементов массива А включающих двойку.**
2. **Написать запрос для немедленного нахождения положительных єлементов массива В как int[].**
3. **Написать запрос для нахождения всех оттенков желтого цвета в массиве С.**
4. **Создать список автомобилей класса Car. Выбрать из списка автомобили количество пассажиров которых равно четырем и вывести всю информацию о них.**
5. **Создать список продуктов класса Product. Выбрать из списка продукты, ассортимент которых меньше 5 единиц и вывести всю информацию о них.**
6. **Вывести результирующий набор списков myCars и yourCars, содержащий все элементы с повторениями.**

**4.**

1. **Написать запрос для нахождения єлементов массива А включающих пробел.**
2. **Написать запрос для немедленного нахождения отрицательных єлементов массива D как List[].**
3. **Написать запрос для нахождения всех оттенков зеленого цвета в массиве С.**
4. **Создать список автомобилей класса Car. Выбрать из списка автомобили Audi и вывести всю информацию о них.**
5. **Создать список продуктов класса Product. Выбрать из списка продукты, которые закончились и вывести всю информацию о них.**
6. **Вывести результирующий набор списков myCars и yourCars, содержащий общие элементы.**

**5.**

1. **Написать запрос для нахождения єлементов массива А включающих пробел.**
2. **Написать запрос для нахождения положительных єлементов массива В.**
3. **Написать запрос для нахождения всех оттенков красного цвета в массиве С.**
4. **Создать список автомобилей класса Car. Выбрать из списка автомобили максимальная скорость которых превышает 200 км/час и вывести всю информацию о них.**
5. **Создать список продуктов класса Product. Выбрать из списка продукты, ассортимент которых меньше 5 единиц и вывести всю информацию о них.**
6. **Вывести результирующий набор списков myCars и yourCars, содержащий все элементы с повторениями.**

**6.**

1. **Написать запрос для нахождения єлементов массива А включающих цифры.**
2. **Написать запрос для нахождения отрицательных єлементов массива В.**
3. **Написать запрос для нахождения всех оттенков желтого цвета в массиве С.**
4. **Создать список автомобилей класса Car. Выбрать из списка автомобили Audi и вывести всю информацию о них.**
5. **Создать список продуктов класса Product. Выбрать из списка продукты, которые закончились и вывести всю информацию о них.**
6. **Вывести результирующий набор списков myCars и yourCars, содержащий общие элементы.**

**7.**

1. **Написать запрос для нахождения єлементов массива А включающих двойку.**
2. **Написать запрос для немедленного нахождения положительных єлементов массива В как int[].**
3. **Написать запрос для нахождения всех оттенков зеленого цвета в массиве С.**
4. **Создать список автомобилей класса Car. Выбрать из списка автомобили максимальная скорость которых превышает 200 км/час и вывести всю информацию о них.**
5. **Создать список продуктов класса Product. Выбрать из списка продукты, ассортимент которых превышает 50 единиц и вывести всю информацию о них.**
6. **Вывести результирующий набор списков myCars и yourCars, содержащий все элементы без повторений.**

**8.**

1. **Написать запрос для нахождения єлементов массива А включающих цифры.**
2. **Написать запрос для нахождения отрицательных єлементов массива В.**
3. **Написать запрос для нахождения всех оттенков желтого цвета в массиве С.**
4. **Создать список автомобилей класса Car. Выбрать из списка автомобили BMW и вывести всю информацию о них.**
5. **Создать список продуктов класса Product. Выбрать из списка продукты, ассортимент которых меньше 5 единиц и вывести всю информацию о них.**
6. **Вывести результирующий набор списков myCars и yourCars, содержащий все элементы с повторениями.**

**9.**

1. **Написать запрос для нахождения єлементов массива А включающих двойку.**
2. **Написать запрос для немедленного нахождения положительных єлементов массива В как int[].**
3. **Написать запрос для нахождения всех оттенков желтого цвета в массиве С.**
4. **Создать список автомобилей класса Car. Выбрать из списка автомобили Audi и вывести всю информацию о них.**
5. **Создать список продуктов класса Product. Выбрать из списка продукты, которые закончились и вывести всю информацию о них.**
6. **Вывести результирующий набор списков myCars и yourCars, содержащий все элементы без повторений.**

**10.**

1. **Написать запрос для нахождения єлементов массива А включающих двойку.**
2. **Написать запрос для немедленного нахождения положительных єлементов массива В как int[].**
3. **Написать запрос для нахождения всех оттенков желтого цвета в массиве С.**
4. **Создать список автомобилей класса Car. Выбрать из списка автомобили количество пассажиров которых равно четырем и вывести всю информацию о них.**
5. **Создать список продуктов класса Product. Выбрать из списка продукты, ассортимент которых меньше 5 единиц и вывести всю информацию о них.**
6. **Вывести результирующий набор списков myCars и yourCars, содержащий общие элементы.**

**11.**

1. **Написать запрос для нахождения єлементов массива А включающих пробел.**
2. **Написать запрос для нахождения положительных єлементов массива В.**
3. **Написать запрос для нахождения всех оттенков красного цвета в массиве С.**
4. **Создать список автомобилей класса Car. Выбрать из списка автомобили максимальная скорость которых превышает 200 км/час и вывести всю информацию о них.**
5. **Создать список продуктов класса Product. Выбрать из списка продукты, ассортимент которых меньше 5 единиц и вывести всю информацию о них.**
6. **Вывести результирующий набор списков myCars и yourCars, содержащий все элементы с повторениями.**

**12.**

1. **Написать запрос для нахождения єлементов массива А включающих пробел.**
2. **Написать запрос для немедленного нахождения отрицательных єлементов массива D как List[].**
3. **Написать запрос для нахождения всех оттенков зеленого цвета в массиве С.**
4. **Создать список автомобилей класса Car. Выбрать из списка автомобили Audi и вывести всю информацию о них.**
5. **Создать список продуктов класса Product. Выбрать из списка продукты, которые закончились и вывести всю информацию о них.**
6. **Вывести результирующий набор списков myCars и yourCars, содержащий общие элементы.**