# Лабораторная работа № 14 Методы расширения и LINQ запросы

# LINQ to Objects

**Цель.** Получить практические навыки использования методов расширения коллекций и запросов LINQ to objects.

## Теоретические сведения.

Набор технологий LINQ (Language INtegrated Query — язык интегрированных запросов), появившийся в .NET 3.5, предоставляет удобный способ доступа к различным хранилищам данных.

На самом высоком уровне LINQ можно воспринимать как строго типизированный язык запросов, встроенный непосредственно в грамматику самого языка С#. Используя LINQ, можно строить любое количество выражений, которые выглядят и ведут себя подобно SQL-запросам к базе данных. Однако запрос LINQ может применяться к любому числу хранилищ данных, включая хранилища, которые не имеют ничего общего с реляционными базами данных.

Для того чтобы работать с LINQ to Objects, потребуется обеспечить, чтобы в каждом файле кода С#, содержащем запросы LINQ, импортировалось пространство имен System.Linq;

Рассмотрим ключевые конструкции С#, которые обеспечили возможность существования LINQ.

Язык С# использует следующие связанные с LINQ средства:

* неявно типизированные локальные переменные;
* синтаксис инициализации объектов и коллекций;
* лямбда-выражения;
* расширяющие методы;
* анонимные типы.

В простейшем виде каждый запрос LINQ строится из операций from, in и select. Ниже показан базовый шаблон, которому нужно следовать:

**var** результат = **from** сопоставляемыйЭлемент **in** контейнер

**select** сопоставляемыйЭлемент;

Таблица 1 Операции запросов LINQ

|  |  |
| --- | --- |
| Операции запросов | Назначение |
| from, in | Используется для определения любого запроса |
| where | Используется для определения ограничений о том, т.е. какие  элементы должны извлекаться из контейнера |
| select | Используется для выбора последовательности из контейнера |
| join, on, equals, into | Выполняет соединения на основе указанного ключа. |
| orderby | Позволяет упорядочить результирующий набор в порядке возрастания или убывания |
| group, by | Группирует данные по указанному значению |

Компилятор С# на этапе компиляции транслирует все операции С# LINQ в вызовы методов класса Enumerable.

Класс Enumerable предоставляет набор методов типа static для выполнения запросов к объектам, реализующим интерфейс IEnumerable<T>.

Большинство методов Enumerable принимают в качестве аргументов делегаты. В частности, многие методы требуют обобщенного делегата по имени Func<>. Делегат Func<> представляет шаблон функции с набором аргументов и возвращаемым значением.

Поскольку множество членов System.Linq.Enumerable требуют при вызове в качестве входа делегат, можно либо вручную создать новый тип делегата и разработать для него необходимые целевые методы, воспользоваться анонимным методом С# либо определить подходящее лямбда-выражение.

### Построение выражений запросов с использованием операций запросов

**Запрос на выборку:**

**string[] carNames = {"Opel Corsa","Nissan Juke", "Toyota", "Chevrollet", "Ford Focus", "KIA" };**

**//выбрать все названия машин, содержащие пробелы**

**var subset = from car in carNames where car.Contains(" ") orderby car select car;**

**foreach (string s in subset) Console.WriteLine("Item: {0}", s);**

**Получение счетчика (количества объектов с заданным параметром):**

**//найти количество машин БМВ**

**int numb = (from car in myCars where car.Make == "BMW" select car).Count<Car>()** ;

Console.WriteLine("Number of BMW is "+numb);

**Использование операций над множествами (пересечение, объединение, разность):**

**var carDiff = (from c in list1 select c).Except(from c2 in list2 select c2);**

**Console.WriteLine("Разность множеств:");**

**foreach (var car in carDiff)**

**Console.WriteLine(car);**

**Агрегирование данных:**

**Console.WriteLine("Max Speed={0}",(from t in list1 select t).Max());**

**Console.WriteLine("Min Speed={0}", (from t in list1 select t).Min());**

* 1. **Построение выражений запросов с использованием расширяющих методов и лямбда-выражений**

Используемые операции запросов LINQ — это на самом деле сокращенные версии вызова различных расширяющих методов, определенных в типе Enumerable.

Например

**var subset2 = carNames.Where(car => car.Contains(" ")). OrderBy(car => car).Select(car => car);**

**Console.WriteLine("Вариант 2");**

**foreach (string s in subset2)**

**Console.WriteLine("Item: {0}", s);**

Чтобы было более понятно, разобьем этот запрос на фрагменты:

1. **var carsWithSpaces = carNames.Where(car => car.Contains(" "));**
2. **var orderedGames = carsWithSpaces.OrderBy(car => car);**
3. **var subset2 = orderedGames.Select(car => car);**

1) - вызов расширяющего метода **Where ().**Класс Array получает метод от класса Enumerable.

**public static IEnumerable<TSource> Where<TSource>(this IEnumerable <TSource> source, Func<TSource, bool> predicate**

Метод Enumerable.Where() требует параметра-делегата System.Func<Tl, TResult> Первый параметр делегата – это данные для обработки (массив строк в примере), второй параметр — это результат, который получается от оператора, вставленного в лямбда-выражение. Метод Where () возвращает результат типа OrderedEnumerable.

2) - Для этого результата вызывается обобщенный метод **OrderBy (),** который также принимает параметр — делегат Funс<>.С его помощью производится передача всех элементов по очереди через соответствующее лямбда-выражение. Конечным результатом вызова OrderBy () будет упорядоченная последовательность начальных данных.

3) - Производится вызов метода Select () на последовательности, возвращенной OrderBy (), который в конечном итоге вернет результирующий набор данных.

* 1. **Построение выражений запросов с использованием расширяющих методов и анонимных методов**

Т.к. лямбда-выражения С# — это просто сокращенная нотация вызова анонимных методов, то можно построить запрос с применением анонимных методов. Синтаксис анонимных методов позволяет заключить всю обработку, выполняемую делегатами, в одном определении метода.

**Func<string, bool> searchFilter =delegate(string car) { return car.Contains(" "); };**

**Func<string, string> itemToProcess = delegate(string s) { return s; };**

**var subset = carNames.Where(searchFilter). OrderBy(itemToProcess). Select(itemToProcess);**

Объект **searchFilter** это делегат, который принимает параметр типа string и возвращает результат типа bool.

Объект **itemToProcess** – это делегат, который принимает строку и возвращает строку, в данном случае без изменений.

Используем эти делегаты в методах:

**Where(searchFilter)** – выбирает из carNames элементы для которых результат выполнения метода делегата равен true.

Для результата полученного после применения Where() применяем метод **OrderBy(itemToProcess),** который упорядочивает строки.

Метод **Select(itemToProcess)** возвращает результат.

## Постановка задачи

**Часть 1**

1. Сформировать обобщенную стандартную коллекцию (тип коллекции 1 из таблицы вариантов), содержащую ссылки на другие стандартные обобщенные коллекции (тип коллекции 2 тип коллекции 1 из таблицы вариантов).
2. Заполнить коллекции объектами иерархии классов (лабораторная работа №10).

Выполнить запросы функции (всего должно быть выполнено не менее 5 запросов):

1. На выборку данных (Where).
2. Использование операций над множествами (Union,Except, Intersect).
3. Агрегирование данных (Sum, Max, Min, Average).
4. Группировка данных (Group by).
5. Получение нового типа (Let).
6. Соединение (Join).

Запросы должны быть выполнены двумя способами:

1. С использованием LINQ запросов.
2. С использованием методов расширения.

**Часть 2**

1. К коллекции из лабораторной работы 12 применить запросы:
2. На выборку данных (Where).
3. Получение счетчика (Count).
4. Агрегирование данных (Sum, Max, Min, Average).
5. Группировка данных (Group by).

Запросы должны быть выполнены двумя способами:

1. С использованием LINQ запросов.
2. С использованием методов расширения.

## Варианты

|  |  |  |  |
| --- | --- | --- | --- |
| № | Коллекция\_1 | Коллекция\_2 | Иерархия классов |
| 1 | Сеть магазинов по продаже часов (List) | Магазин по продаже часов (Dictionary) | **Часы**, «умные» часы, электронные часы, аналоговые часы, |
| 2 | Завод по изготовлению автомобилей (Queue) | Цех (List) | **Автомобиль**, легковой автомобиль, внедорожник, грузовой автомобиль; |
| 3 | Тетрадь (Dictionary) | Страница тетради (Stack) | **Геометрическая фигура**, окружность, параллелепипед, прямоугольник; |
| 4 | Сеть магазинов по продаже инструментов (SortedDictionary) | Магазин по продаже инструментов (Queue) | **Инструмент**, ручной инструмент, электрический инструмент, измерительный инструмент; |
| 5 | Чат (Stack) | Сообщение (Dictionary) | **Эмодзи**, лицевые эмодзи, улыбающееся эмодзи, животные эмодзи, |
| 6 | Банк (List) | Отделение банка по обслуживанию клиентов (SortedDictionary) | **Банковская карта**, дебетовая карта, молодёжная карта, кредитная карта; |
| 7 | Дом (Queue) | Квартира (Dictionary) | **ДомашниеЖивотные**, млекопитающие, кошки, рыбы; |
| 8 | Станция (Dictionary) | Поезд (Stack) | **Вагон**, пассажирский вагон, грузовой вагон, вагон-ресторан |
| 9 | Участники концерта (SortedDictionary) | Музыкальная группа (List) | **Музыкальный инструмент**, гитара, электрогитара, фортепиано; |
| 10 | Приложение (Stack) | Окно (Dictionary) | **Элемент управления**, кнопка, кнопка множественного выбора, текстовое поле; |
| 11 | Город (List) | Аэропорт (Stack) | **Воздушное судно**, самолёт, вертолёт, истребитель; |
| 12 | Вселенная (Queue) | Галактика (Dictionary) | **Небесное тело**, планета, газовый гигант, звезда; |
| 13 | Поселок (Dictionary) | Сад (Queue) | **Растение**, дерево, цветок, роза; |
| 14 | Сеть магазинов по продаже игр (SortedDictionary) | Магазин по продаже игр (List) | **Игра**, настольная игра, видео игра, VR игра. |
| 15 | Сеть кондитерских (List) | Кондитерская (Stack) | **Кондитерское изделие**, торт, печенье, шоколадное печенье; |

## Методические указания

1. Интерфейс реализовать в виде текстового меню.
2. Коллекции для выполнения запросов сформировать с помощью ДСЧ.
3. Каждый запрос выполнить в отдельной функции.
4. При выполнении группировки подсчитать количество элементов в группе.
5. Для выполнения соединения добавить новый тип данных в иерархию классов из лаб. работы №10.

## Содержание отчета

1. Постановка задачи (общая и конкретного варианта)
2. Диаграмма классов для каждой части работы (из VS или Visio)
3. Код программы (или ссылка на гитхаб)
4. Код unit-тестов (или тесты в Excel)
5. Анализ покрытия кода тестами (из VS или в Excel)

## Вопросы для защиты лабораторной работы

1. Чем отличаются отложенное и неотложенное выполнение LINQ запросов?
2. Чем методы расширения отличаются от LINQ запросов?
3. Что является параметром методов расширения?
4. Как выполняется настройка методов расширения на задачу пользователя?
5. Привести примеры отложенных и неотложенных методов расширения.

## Критерии оценивания работы

|  |  |  |
| --- | --- | --- |
| № | Критерий | Баллы |
| 1 | Решены задачи 1-ой части работы:  продемонстрирована работа со стандартными универсальными коллекциями | 3 |
| 2. | Решены задачи 2-ой части работы:  продемонстрирована работа со стандартными обобщенными коллекциями | 1 |
| 3 | Написаны тесты ко всем функциям из 1 и 2 части | 2 |
| 5 | Даны полные ответы на теоретические вопросы (п.5) | 1 |
| 6 | Написанный код соответствует стиль-гайду | 1 |
| 7 | Дополнительные баллы:  - за описание предметной области (правильный выбор полей, ключей коллекций, осмысленные запросы, добавленные классы и т.п.)  - за реализацию своих методов расширения для коллекции из лаб. работы 12 | 1-2 |