Лабораторная работа № 14

LINQ to Objects

Цель. Получить практические навыки использования запросов LINQ to objects.

1. 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 C#. 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 C#, содержащем запросы LINQ, импортировалось пространство имен System.Linq;

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

Язык C# 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 C# на этапе компиляции транслирует все операции C# 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 C# либо определить подходящее лямбда-выражение.

1.1. 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());
```

1.2. 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 Func<>.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.

1.3. Построение выражений запросов с использованием расширяющих методов и анонимных методов

Т.к. лямбда-выражения C# — 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.

1.4. Примеры использования метода Aggregate()

Метод **Aggregate**() применяет к последовательности агрегатную функцию.

Синтаксис :

```
public static TSource Aggregate<TSource>( this IEnumerable
<TSource> source, Func<TSource, TSource, Source> func)
```

где **TSource** – тип элементов обрабатываемой последовательности **source**,

Параметры:

IEnumerable<T> source – объект к которому будет применяться агрегатная функция,

Func<TSource, TSource, TSource> func – агрегатная функция, вызываемая для каждого элемента последовательности.

Возвращаемое значение типа **TSource** – конечное агрегатное значение.

Агрегатная функция (может предоставляться в виде лямбда-выражения) будет применяться для каждой пары элементов в коллекции от начала до конца, причем

результат каждой операции будет являться входными данными следующей операции вычисления.

Задача: Найти сумму элементов массива, состоящего из целых чисел.

```
//последовательность
int[] arr1 = { 1, 2, 3, 4, 5, 6, 7, 8, 9 };
// метод расширения и лямбда выражение
int sum1 = arr1.Aggregate<int>((a, b) => a + b);
Console.WriteLine("sum1=" + sum1);
//метод расширения и анонимный делегат
Func<int,int,int> summa=delegate(int a, int b) {return a+b;};
int sum2 = arr1.Aggregate<int>(summa);
Console.WriteLine("sum2=" + sum2);
//LINQ-запрос
int sum3=(from num in arr1 select num).Sum();
Console.WriteLine("sum3=" + sum3);
```

2. Постановка задачи

1. Сформировать обобщенную коллекцию (лабораторная работа № 11), содержащую ссылки на другие коллекции.
2. Заполнить коллекции объектами иерархии классов (лабораторная работа №10).

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

a) На выборку данных.
b) Получение счетчика (количества объектов с заданным параметром).
c) Использование операций над множествами (пересечение, объединение, разность).
d) Агрегирование данных.

Запросы должны быть выполнены двумя способами:

a) С использованием LINQ запросов.
b) С использованием методов расширения.

Каждый запрос выполняется в отдельной функции.
Примеры запросов (лабораторная работа №10).

3. Варианты

№	Коллекция_1	Коллекция_2	Иерархия классов
1	Вуз	Факультет	студент, преподаватель, **персона**, сотрудник;
2	Завод	Цеха	служащий, **персона**, рабочий, инженер;
3	Предприятие	Отдел	рабочий, **персона**, инженер, администрация;
4	Город	Район	**организация,** страховая компания, судостроительная компания, завод, библиотека;
5	Зачетная книжка	Семестр	тест, экзамен, выпускной экзамен, **испытание;**
6	Континент	Страна	**место,** область, город, мегаполис, адрес;

7	Магазин	Отдел	игрушка, продукт, **товар**, молочный продукт;
8	Документы	Папка	квитанция, накладная, **документ**, чек;
9	Корпорация	Филиал	цех, мастерская, фабрика, **производство**;
10	Город	Образовательное учреждение	**персона**, студент, школьник, студент-заочник;
11	Город	Вокзал	автобус, поезд, **транспортное средство**, экспресс;
12	Земля	Континент	республика, монархия, королевство, **государство**;
13	Зоопарк	Секция	млекопитающие, парнокопытные, птицы, **животное**;
14	Море	Порт	**корабль**, пароход, парусник, корвет;
15	Мастерская	Машина	**двигатель**, двигатель внутреннего сгорания, дизель, турбореактивный двигатель;
16	Библиотека	Отдел	журнал, книга, **печатное издание**, учебник

Запросы на выборку

1. Имена всех лиц мужского (женского) пола.
2. Имена студентов указанного курса.
3. Имена и должность преподавателей указанной кафедры.
4. Имена служащих со стажем не менее заданного.
5. Имена служащих заданной профессии.
6. Имена рабочих заданного цеха.
7. Имена рабочих заданной профессии.
8. Имена студентов, сдавших все (заданный) экзамены на отлично (хорошо и отлично).
9. Имена всех монархов на заданном континенте.
10. Наименование всех деталей (узлов), входящих в заданный узел (механизм).
11. Наименование всех книг в библиотеке (магазине), вышедших не ранее указанного года.
12. Названия всех городов заданной области.
13. Наименование всех товаров в заданном отделе магазина.
14. Наименование всех цехов на данном заводе.
15. Наименование птиц в зоопарке.
16. Имена пароходов, приписанных к данному порту.
17. Наименование журналов, выписываемых библиотекой.

Получение счетчика

18. Количество инженеров на заводе.
19. Количество мужчин (женщин).
20. Количество студентов на указанном курсе.
21. Количество рабочих со стажем не менее заданного.
22. Количество рабочих заданной профессии.
23. Количество инженеров в заданном подразделении.
24. Количество товара заданного наименования.

25. Количество студентов, сдавших все экзамены на отлично.
26. Количество студентов, не сдавших хотя бы один экзамен.
27. Количество деталей (узлов), входящих в заданный узел (механизм).
28. Количество указанного транспортного средства в автопарке (на автостоянке).
29. Количество пассажиров во всех вагонах экспресса.
30. Количество библиотек в городе.
31. Количество рабочих в заданном цехе.
32. Количество жителей данного континента.
33. Количество различных типов ДВС, обслуживаемых автомастерской.
34. Количество книг во всех библиотеках города.
35. Количество чеков на сумму превышающую заданную.

Агрегирование данных

36. Суммарная стоимость товара заданного наименования.
37. Средний балл за сессию заданного студента.
38. Суммарное количество учебников в библиотеке (магазине).
39. Суммарное количество жителей всех городов в области.
40. Суммарная стоимость продукции заданного наименования по всем накладным.
41. Средняя мощность всех (заданного типа) транспортных средств в организации.
42. Средняя мощность всех дизелей, обслуживаемых заданной фирмой.
43. Средний вес животных заданного вида в зоопарке.
44. Среднее водоизмещение всех парусников на верфи (в порту).
45. Суммарный вес всех деталей в заданном узле.
46. Суммарная стоимость всех деталей в механизме.
47. Суммарный страховой фонд всех страховых компаний региона.
48. Самый мощный автомобиль в данной организации.
49. Общая сумма по всем чекам, выписанным в организации.
50. Самая дорогая и самая дешевая игрушка в магазине(наименование и стоимость).