Теория и практика программирования

Шпилёв Пётр Валерьевич

Санкт-Петербургский государственный университет
Математико-механический факультет
Кафедра статистического моделирования

Лекция 12

Санкт-Петербург
2016 г.

e-mail: pitshp@hotmail.com

Простой запрос

```
using System.Linq
...
int[ ] nums = { 1, -2, 3, 0, -4, 5 };
// Простой запрос на получение только положительных значений.
var posNums = from n in nums
              where n > 0
              select n;
```

Простой запрос

```
using System.Linq
...
int[ ] nums = { 1, -2, 3, 0, -4, 5 };
// Простой запрос на получение только положительных значений.
var posNums = from n in nums
              where n > 0
              select n;
```

Общая форма оператора from

```
from переменная_диапазона in источник_данных
```

Простой запрос

```
using System.Linq
...
int[ ] nums = { 1, -2, 3, 0, -4, 5 };
// Простой запрос на получение только положительных значений.
var posNums = from n in nums
              where n > 0
              select n;
```

Общая форма оператора from

from переменная_диапазона in источник_данных

Общая форма оператора where

where булево_выражение

Такое выражение иначе называется предикатом. В запросе можно указывать несколько операторов where.

Простой запрос

```
using System.Linq
...
int[ ] nums = { 1, -2, 3, 0, -4, 5 };
// Простой запрос на получение только положительных значений.
var posNums = from n in nums
              where n > 0
              select n;
```

Общая форма оператора from

from переменная_диапазона in источник_данных

Общая форма оператора where

where булево_выражение

Такое выражение иначе называется предикатом. В запросе можно указывать несколько операторов where.

Оператор select

Все запросы оканчиваются оператором select или group. В данном примере используется оператор select, точно определяющий, что именно должно быть получено по запросу.

Простой запрос

```
using System.Linq
...
int[ ] nums = { 1, -2, 3, 0, -4, 5 };
// Простой запрос на получение только положительных значений.
var posNums = from n in nums
              where n > 0
              select n;
```

Общая форма оператора from

from переменная_диапазона in источник_данных

Общая форма оператора where

where булево_выражение

Такое выражение иначе называется предикатом. В запросе можно указывать несколько операторов where.

Упражнение 12.1

Для массива целых чисел сформировать простой запрос и отобразить его результаты. Внести изменения в массив и повторно отобразить результаты запроса.

Таблица 1: Контекстно-зависимые ключевые слова, используемые в запросах.

Ascending	by	descending	equals
from	in	into	group
join	on	let	orderby
select	where		

Таблица 1: Контекстно-зависимые ключевые слова, используемые в запросах.

Ascending	by	descending	equals
from	in	into	group
join	on	let	orderby
select	where		

Таблица 1: Контекстно-зависимые ключевые слова, используемые в запросах.

Ascending	by	descending	equals
from	in	into	group
join	on	let	orderby
select	where		

Замечание

Запрос должен начинаться с ключевого слова from и оканчиваться ключевым словом select или group. Оператор select определяет тип значения, перечисляемого по запросу, а оператор group возвращает данные группами, причем каждая группа может перечисляться по отдельности.

Оператор where

```
int[ ] nums = { 1, -2, 3, -3, 0, -8, 12, 19, 6, 9, 10 };
var posNums = from n in nums
              where n > 0
              where n < 10
              select n;
```

Оператор where

```
int[ ] nums = { 1, -2, 3, -3, 0, -8, 12, 19, 6, 9, 10 };
var posNums = from n in nums
              where n > 0 & n < 10
              select n;
```

Оператор where

```
int[ ] nums = { 1, -2, 3, -3, 0, -8, 12, 19, 6, 9, 10 };
var posNums = from n in nums
              where n > 0 & n < 10
              select n;
```

Оператор orderby

orderby *элемент* порядок

Оператор where

```
int[ ] nums = { 1, -2, 3, -3, 0, -8, 12, 19, 6, 9, 10 };
var posNums = from n in nums
              where n > 0 & n < 10
              select n;
```

Оператор orderby

orderby *элемент* порядок

Пример:

```
var posNums = from n in nums
              orderby n
              select n;
```

Оператор where

```
int[ ] nums = { 1, -2, 3, -3, 0, -8, 12, 19, 6, 9, 10 };
var posNums = from n in nums
              where n > 0 & n < 10
              select n;
```

Оператор orderby

orderby *элемент* порядок

Пример:

```
var posNums = from n in nums
              orderby n descending
              select n;
```

Оператор where

```
int[ ] nums = { 1, -2, 3, -3, 0, -8, 12, 19, 6, 9, 10 };
var posNums = from n in nums
              where n > 0 & n < 10
              select n;
```

Оператор orderby

orderby *элемент* порядок

Сортировка по нескольким критериям

orderby элемент_A направление, элемент_B направление, элемент_C направление, ...

Оператор where

```
int[ ] nums = { 1, -2, 3, -3, 0, -8, 12, 19, 6, 9, 10 };
var posNums = from n in nums
              where n > 0 & n < 10
              select n;
```

Оператор orderby

orderby *элемент* порядок

Сортировка по нескольким критериям

orderby элемент_A направление, элемент_B направление, элемент_C направление, ...

Упражнение 12.2

Написать программу в которой, для базы данных клиентов банка, формируется запрос с сортировкой по трем критериям: фамилии клиента, имени и остатку на его счете. Вывести результаты на консоль.

Общая форма оператора select

select выражение

Общая форма оператора select

select выражение

Пример:

```
var sqrRoots = from n in nums
               where n > 0
               select Math.Sqrt(n);
```

Общая форма оператора select

select выражение

Упражнение 12.3

Для объектов класса EmailAddress, содержащего два свойства (имя пользователя и его e-mail) сформировать запрос и вывести по его результатам e-mail-ы пользователей.

Общая форма оператора select

select выражение

Упражнение 12.3

Для объектов класса EmailAddress, содержащего два свойства (имя пользователя и его e-mail) сформировать запрос и вывести по его результатам e-mail-ы пользователей.

Упражнение 12.4

Для объектов класса ContactInfo, содержащего три свойства (имя пользователя, его e-mail и телефон) сформировать запрос и вывести по его результатам объекты класса EmailAddress (см. предыдущий пример).

Общая форма оператора select

select выражение

Упражнение 12.3

Для объектов класса EmailAddress, содержащего два свойства (имя пользователя и его e-mail) сформировать запрос и вывести по его результатам e-mail-ы пользователей.

Упражнение 12.4

Для объектов класса ContactInfo, содержащего три свойства (имя пользователя, его e-mail и телефон) сформировать запрос и вывести по его результатам объекты класса EmailAddress (см. предыдущий пример).

Вложенные операторы from

```
var pairs = from n1 in Array_1
            from n2 in Array_2
            select new MyPairs(n1, n2);
```

Общая форма оператора select

select выражение

Упражнение 12.3

Для объектов класса EmailAddress, содержащего два свойства (имя пользователя и его e-mail) сформировать запрос и вывести по его результатам e-mail-ы пользователей.

Упражнение 12.4

Для объектов класса ContactInfo, содержащего три свойства (имя пользователя, его e-mail и телефон) сформировать запрос и вывести по его результатам объекты класса EmailAddress (см. предыдущий пример).

Вложенные операторы from

```
var pairs = from n1 in Array_1
            from n2 in Array_2
            select new MyPairs(n1, n2);
```

Упражнение 12.5

Использовать два вложенных оператора from для составления списка всех возможных результатов бросков двух игральных костей.

Общая форма оператора group

group переменная_диапазона by ключ

Общая форма оператора group

group переменная_диапазона by ключ

Замечание

Результатом выполнения оператора group является последовательность, состоящая из элементов типа IGrouping<TKey, TElement>, т.е. обобщенного интерфейса, объявляемого в пространстве имен System.Linq.

Общая форма оператора group

group переменная_диапазона by ключ

Замечание

Результатом выполнения оператора group является последовательность, состоящая из элементов типа IGrouping<TKey, TElement>, т.е. обобщенного интерфейса, объявляемого в пространстве имен System.Linq.

Замечание

Типом переменной запроса, возвращающего группу, является IEnumerable<IGrouping<TKey, TElement>>. В интерфейсе IGrouping определено также доступное только для чтения свойство Key, возвращающее ключ, связанный с каждой коллекцией.

Общая форма оператора group

group переменная_диапазона by ключ

Пример:
Garage - массив объектов Car с тремя свойствами:
имя владельца, марка машины и ее цвет

```
var MyCars = from SomeCar in Garage
                where SomeCar.Owner == "Имя владельца"
                group SomeCar by SomeCar.Model;
foreach (var Cars in MyCars)
{
    Console.WriteLine("Машины по маркам: " + Cars.Key);
    foreach (var Car in Cars)
        Console.WriteLine("Марка: " + Car.Model + "цвет: " + Car.Color);
    Console.WriteLine();
}
```

Общая форма оператора group

group переменная_диапазона by ключ

Пример:
Garage - массив объектов Car с тремя свойствами:
имя владельца, марка машины и ее цвет

```
var MyCars = from SomeCar in Garage
                where SomeCar.Owner == "Имя владельца"
                group SomeCar by SomeCar.Model;
foreach (var Cars in MyCars)
{
    Console.WriteLine("Машины по маркам: " + Cars.Key);
    foreach (var Car in Cars)
        Console.WriteLine("Марка: " + Car.Model + "цвет: " + Car.Color);
    Console.WriteLine();
}
```

Упражнение 12.6

Для массива содержащего список веб-сайтов (в формате string), сформировать запрос, в котором этот список группируется по имени домена самого верхнего уровня, например .org или .com.

Общая форма оператора into

into имя тело_запроса

где имя обозначает конкретное имя переменной диапазона, используемой для циклического обращения к временному результату в продолжении запроса, на которое указывает тело_запроса.

Общая форма оператора into

into имя тело_запроса
где имя обозначает конкретное имя переменной диапазона, используемой для циклического обращения к временному результату в продолжении запроса, на которое указывает тело_запроса.

Замечание

Оператор into используется вместе с оператором select или group и называется продолжением запроса, поскольку он продолжает запрос для получения окончательного результата

Общая форма оператора into

into имя тело_запроса
где имя обозначает конкретное имя переменной диапазона, используемой для циклического обращения к временному результату в продолжении запроса, на которое указывает тело_запроса.

Замечание

Оператор into используется вместе с оператором select или group и называется продолжением запроса, поскольку он продолжает запрос для получения окончательного результата

Пример:

```
var MyCars = from Car in Garage
             where Car.Owner == "Имя владельца"
             group Car by Car.Model;
             into ws
             where ws.Count() > 2
             select ws;
```

Общая форма оператора let

let имя = выражение
где имя обозначает идентификатор, получающий значение, которое дает выражение.

Общая форма оператора let

let имя = выражение
где имя обозначает идентификатор, получающий значение, которое дает выражение.

Пример:
Garages - массив объектов Garage с двумя свойствами:
Адрес владельца и массив объектов Car

```
var MyCars = from MyGarage in Garages
             where MyGarage.Address == "Адрес владельца "
             let Garage = MyGarage.Cars
             from SomeCar in Garage
             where SomeCar.Owner == "Имя владельца"
             group SomeCar by SomeCar.Model;
```

Общая форма оператора let

let имя = выражение
где имя обозначает идентификатор, получающий значение, которое дает выражение.

Пример:
Garages - массив объектов Garage с двумя свойствами:
Адрес владельца и массив объектов Car

```
var MyCars = from MyGarage in Garages
             where MyGarage.Address == "Адрес владельца "
             let Garage = MyGarage.Cars
             from SomeCar in Garage
             where SomeCar.Owner == "Имя владельца"
             group SomeCar by SomeCar.Model;
```

Упражнение 12.7

Сформировать запрос на получение отсортированной последовательности символов, возвращаемых из массива строк.

Общая форма оператора join

```
from переменная_диапазона_A in источник_данных_A
join переменная_диапазона_B in источник_данных_B
on переменная_диапазона_A.свойство equals
переменная_диапазона_B.свойство.
```

Общая форма оператора join

from переменная_диапазона_A in источник_данных_A
join переменная_диапазона_B in источник_данных_B
on переменная_диапазона_A.свойство equals
переменная_диапазона_B.свойство.

Упражнение 12.8

Написать программу для учета товаров на складе. Реализовать класс с двумя свойствами: наименование и инвентарный номер товара и класс со свойствами: инвентарный номер товара и количество единиц товара на складе. Сформировать запрос по результатам которого будет создан массив объектов содержащих информацию о наименование товара и количестве единиц на складе.

Общая форма оператора join

from переменная_диапазона_A in источник_данных_A
join переменная_диапазона_B in источник_данных_B
on переменная_диапазона_A.свойство equals
переменная_диапазона_B.свойство.

Упражнение 12.8

Написать программу для учета товаров на складе. Реализовать класс с двумя свойствами: наименование и инвентарный номер товара и класс со свойствами: инвентарный номер товара и количество единиц товара на складе. Сформировать запрос по результатам которого будет создан массив объектов содержащих информацию о наименование товара и количестве единиц на складе.

Общая форма объявления анонимного типа

new { имя_A = значение_A, имя_B = значение_B, ... }

Общая форма оператора join

from переменная_диапазона_A in источник_данных_A
join переменная_диапазона_B in источник_данных_B
on переменная_диапазона_A.свойство equals
переменная_диапазона_B.свойство.

Упражнение 12.8

Написать программу для учета товаров на складе. Реализовать класс с двумя свойствами: наименование и инвентарный номер товара и класс со свойствами: инвентарный номер товара и количество единиц товара на складе. Сформировать запрос по результатам которого будет создан массив объектов содержащих информацию о наименование товара и количестве единиц на складе.

Общая форма объявления анонимного типа

new { имя_A = значение_A, имя_B = значение_B, ... }

Пример:
var myOb = new { Count = 10, Max = 100, Min = 0 }

Общая форма оператора join

from переменная_диапазона_A in источник_данных_A
join переменная_диапазона_B in источник_данных_B
on переменная_диапазона_A.свойство equals
переменная_диапазона_B.свойство.

Упражнение 12.8

Написать программу для учета товаров на складе. Реализовать класс с двумя свойствами: наименование и инвентарный номер товара и класс со свойствами: инвентарный номер товара и количество единиц товара на складе. Сформировать запрос по результатам которого будет создан массив объектов содержащих информацию о наименование товара и количестве единиц на складе.

Общая форма объявления анонимного типа

new { имя_A = значение_A, имя_B = значение_B, ... }

Упражнение 12.9

Использовать анонимный тип для инкапсуляции результата, возвращаемого оператором join в предыдущем упражнении.

Оператор into

Оператор into можно использовать вместе с оператором join для создания группового объединения, образующего последовательность, в которой каждый результат состоит из элементов данных из первой последовательности и группы всех совпадающих элементов из второй последовательности.

Группа А	
А	X
Б	Y
А	Z

Группа Б	
А	1
Б	2
В	3
А	4
Б	5

join(by name)

А	X	А	1
А	Z	А	1
А	X	А	4
А	Z	А	4
Б	Y	Б	2
Б	Y	Б	5

into

А	X

А	1
А	4

А	Z

А	1
А	4

Б	Y

Б	2
Б	5

Оператор into

Оператор into можно использовать вместе с оператором join для создания группового объединения, образующего последовательность, в которой каждый результат состоит из элементов данных из первой последовательности и группы всех совпадающих элементов из второй последовательности.

Упражнение 12.10

Написать программу иллюстрирующую, как групповое объединение может использоваться для составления списка, в котором различные транспортные средства (автомашины, суда и самолеты) организованы по общим для них категориям транспорта: наземного, морского, воздушного и речного. Использовать анонимный тип.

Таблица 2: Основные методы запроса

Оператор запроса	**Эквивалентный метод запроса**
select	Select(selector)
where	Where(predicate)
orderby	OrderBy(keySelector) или OrderByDescending(keySelector)
join	Join(inner, outerKeySelector, innerKeySelector, resultSelector)
group	GroupBy(keySelector)

Таблица 2: Основные методы запроса

Оператор запроса	Эквивалентный метод запроса
select	Select(selector)
where	Where(predicate)
orderby	OrderBy(keySelector) или OrderByDescending(keySelector)
join	Join(inner, outerKeySelector, innerKeySelector, resultSelector)
group	GroupBy(keySelector)

Замечание

За исключением метода Join(), остальные методы запроса принимают единственный аргумент, который представляет собой объект некоторой разновидности обобщенного типа Func<T, TResult>. Аргумент метода запроса представляет собой метод, совместимый с формой встроенного делегата, объявляемый следующим образом:
delegate TResult Func<in T, out TResult>(T arg)
где TResult обозначает тип результата, который дает делегат, а T — тип элемента.

Простой запрос

```
int[ ] nums = { 1, -2, 3, 0, -4, 5 };
// Простой запрос на получение только положительных значений.
var posNums = from n in nums
              where n > 0 & n < 5
              select n;
```

Формирование запроса с помощью методов расширения

```
int[ ] nums = { 1, -2, 3, 0, -4, 5 };
// Простой запрос на получение только положительных значений.
var posNums = nums.Where(n => n>0 & n<5).Select(r => r);
```

Формирование запроса с помощью методов расширения

```
int[ ] nums = { 1, -2, 3, 0, -4, 5 };
// Простой запрос на получение только положительных значений.
var posNums = nums.Where(n => n>0 & n<5).OrderByDescending(j => j);
```

Формирование запроса с помощью методов расширения

```
int[ ] nums = { 1, -2, 3, 0, -4, 5 };
// Простой запрос на получение только положительных значений.
var posNums = nums.Where(n => n>0 & n<5).OrderByDescending(j => j);
```

Оператор group

```
Garage - массив объектов Car с тремя свойствами:
имя владельца, марка машины и ее цвет
var MyCars = from SomeCar in Garage
                where SomeCar.Owner == "Имя владельца"
                group SomeCar by SomeCar.Model;
```

Формирование запроса с помощью методов расширения

```
int[ ] nums = { 1, -2, 3, 0, -4, 5 };
// Простой запрос на получение только положительных значений.
var posNums = nums.Where(n => n>0 & n<5).OrderByDescending(j => j);
```

Метод расширения GroupBy

```
Garage - массив объектов Car с тремя свойствами:
имя владельца, марка машины и ее цвет
var MyCars = Garage.Where(SomeCar => SomeCar.Owner == "Имя
владельца") .GroupBy(SomeCar => SomeCar.Model);
```

Формирование запроса с помощью методов расширения

```
int[ ] nums = { 1, -2, 3, 0, -4, 5 };
// Простой запрос на получение только положительных значений.
var posNums = nums.Where(n => n>0 & n<5).OrderByDescending(j => j);
```

Метод расширения GroupBy

Garage - массив объектов Car с тремя свойствами:
имя владельца, марка машины и ее цвет

```
var MyCars = Garage.Where(SomeCar => SomeCar.Owner == "Имя
владельца") .GroupBy(SomeCar => SomeCar.Model);
```

Оператор join

```
var Массив_группы_B =
        from a in Группа_A
        join b in Группа_Б
        on a.Свойство_1 equals b.Свойство_1
        select new Группа_B(a.Свойство_2, b.Свойство_2);
```

Формирование запроса с помощью методов расширения

```
int[ ] nums = { 1, -2, 3, 0, -4, 5 };
// Простой запрос на получение только положительных значений.
var posNums = nums.Where(n => n>0 & n<5).OrderByDescending(j => j);
```

Метод расширения GroupBy

Garage - массив объектов Car с тремя свойствами:
имя владельца, марка машины и ее цвет

```
var MyCars = Garage.Where(SomeCar => SomeCar.Owner == "Имя
владельца") .GroupBy(SomeCar => SomeCar.Model);
```

Метод расширения Join

```
var Массив_группы_B =
Группа_A.Join(Группа_Б, a => a.Свойство_1, b => b.Свойство_1,
(a,b) => select new Группа_B(a.Свойство_2, b.Свойство_2));
```

Таблица 3: Дополнительные методы расширения

Метод	Описание
All(predicate)	Возвращает логическое значение true, если все элементы в последовательности удовлетворяют условию, задаваемому параметром predicate
Any(predicate)	Возвращает логическое значение true, если любой элемент в последовательности удовлетворяет условию, задаваемому параметром predicate
Average()	Возвращает среднее всех значений в числовой последовательности
Contains(value)	Возвращает логическое значение true, если в последовательности содержится указанный объект
Count()	Возвращает длину последовательности, т.е. количество составляющих ее элементов
First()	Возвращает первый элемент в последовательности
Last()	Возвращает последний элемент в последовательности
Max()	Возвращает максимальное значение в последовательности
Min()	Возвращает минимальное значение в последовательности
Sum()	Возвращает сумму значений в числовой последовательности

Таблица 3: Дополнительные методы расширения

Метод	Описание
All(predicate)	Возвращает логическое значение true, если все элементы в последовательности удовлетворяют условию, задаваемому параметром predicate
Any(predicate)	Возвращает логическое значение true, если любой элемент в последовательности удовлетворяет условию, задаваемому параметром predicate
Average()	Возвращает среднее всех значений в числовой последовательности
Contains(value)	Возвращает логическое значение true, если в последовательности со-
Cou	...их
First()	Возвращает первый элемент в последовательности
Last()	Возвращает последний элемент в последовательности
Max()	Возвращает максимальное значение в последовательности
Min()	Возвращает минимальное значение в последовательности
Sum()	Возвращает сумму значений в числовой последовательности

Упражнение 12.11

Написать программу для демонстрации методов из таблицы 3.

Общая форма метода расширения

static возвращаемый_тип имя (this тип_вызывающего_объекта ob, список_параметров)

Общая форма метода расширения

static возвращаемый_тип имя (this тип_вызывающего_объекта ob, список_параметров)

Пример:

```
static class MyExtMeths
{
    // Возвратить обратную величину числового значения типа double.
    public static double Reciprocal(this double v)
    {
        return 1.0 / v;
    }
}
class ExtDemo
{
   static void Main()
   {
      double val = 3.0;
       // Вызвать метод расширения Reciprocal().
      Console.WriteLine("Обратная величина {0} равна {1}",val,val.Reciprocal());
   }
}
```

Общая форма метода расширения

static возращаемый_тип имя (this тип_вызывающего_объекта ob, список_параметров)

Пример:

```
static class MyExtMeths
{
    // Возвратить обратную величину числового значения типа double.
    public static double Reciprocal(this double v)
```

Упражнение 12.12

Реализовать несколько методов расширения и использовать их в одном из ранее созданных проектов.

```
}
class ExtDemo
{
   static void Main()
   {
      double val = 3.0;
       // Вызвать метод расширения Reciprocal().
      Console.WriteLine("Обратная величина {0} равна {1}",val,val.Reciprocal());
   }
}
```