Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники»
Филиал
«Минский радиотехнический колледж»

Учебный предмет
«Конструирование программ и языки программирования»

Инструкция

по выполнению лабораторной работы №35
«Разработка, отладка и испытание программ на фильтрацию выборки»

Минск 2024

Лабораторная работа №35

Тема работы: «Создание и вывод графический изображений на форму. Анимация»

1 Цель работы

Сформировать умения и навыки разработки программ с использованием фильтрации данных.

2 Задание

Создать два класса по темам, представленным в пункте 5 данной инструкции. Осуществить выборку на основе двух источников данных с использованием LINQ.

3 Оснащение работы

ПК, среда Visual Studio 2019, MSword.

4 Основные 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.

Существует несколько разновидностей LINQ:

- LINQ to Objects: применяется для работы с массивами и коллекциями;
- LINQ to Entities: используется при обращении к базам данных через технологию Entity Framework;
- LINQ to Sql: технология доступа к данным в MS SQL Server;
- LINQ to XML: применяется при работе с файлами XML;
- LINQ to DataSet: применяется при работе с объектом DataSet;
- parallel LINQ (PLINQ): используется для выполнения параллельных запросов.

Чтобы использовать функциональность LINQ, неоходимо убедиться, что в файле подключено пространство имен **System.LINQ**.

Преимущество использования LINQ в коде

Код без LINQ:

```
string[] teams = {"Бавария", "Боруссия", "Реал Мадрид", "Манчестер Сити", "ПСЖ", "Барселона"};

var selectedTeams = new List<string>();
foreach(string s in teams)
{
    if (s.ToUpper().StartsWith("Б"))
```

```
            selectedTeams.Add(s);
    }
    selectedTeams.Sort();

    foreach (string s in selectedTeams)
        Console.WriteLine(s);
```

Код с LINQ:

```
    string[] teams = {"Бавария", "Боруссия", "Реал Мадрид", "Манчестер Сити", "ПСЖ", "Барселона"};

    var selectedTeams = from t in teams // определяем каждый объект из teams как t
                        where t.ToUpper().StartsWith("Б") //фильтрация по критерию
                        orderby t  // упорядочиваем по возрастанию
                        select t; // выбираем объект

    foreach (string s in selectedTeams)
        Console.WriteLine(s);
```

Преимущество использования LINQ заключается в том, что код становится проще и намного меньше.

Несмотря на то, что не указан тип переменной t, выражения LINQ являются строго типизированными. То есть, среда автоматически распознает, что набор teams 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 "Б"), 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.

Список используемых методов расширения LINQ

- select: определяет проекцию выбранных значений;
- where: определяет фильтр выборки;
- orderby: упорядочивает элементы по возрастанию;
- orderbydescending: упорядочивает элементы по убыванию;

- thenby: задает дополнительные критерии для упорядочивания элементов возрастанию;
- thenbydescending: задает дополнительные критерии для упорядочивания элементов 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 среднее 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;
- firstordefault: выбирает первый элемент коллекции или возвращает значение по умолчанию;
- single: выбирает единственный элемент коллекции, если коллекция содердит больше или меньше одного элемента, то генерируется исключение;

- singleordefault: выбирает первый элемент коллекции или возвращает значение по умолчанию;
- elementat: выбирает элемент последовательности по определенному индексу;
- elementatordefault: выбирает элемент коллекции по определенному индексу или возвращает значение по умолчанию, если индекс вне допустимого диапазона;
- last: выбирает последний элемент коллекции;
- lastordefault: выбирает последний элемент коллекции или возвращает значение по умолчанию.

Для выбора элементов из некоторого набора по условию используется метод **Where**.

Пример фильтрации выборки:

```
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);
```

Пример выборки сложных объектов:

```
List<User> users = new List<User>
{
    new User {Name="Том", Age=23, Languages = new List<string> {"английский", "немецкий" }},
    new User {Name="Боб", Age=27, Languages = new List<string> {"английский", "французский" }},
    new User {Name="Джон", Age=29, Languages = new List<string> {"английский", "испанский" }},
    new User {Name="Элис", Age=24, Languages = new List<string> {"испанский", "немецкий" }}
};

var selectedUsers = from user in users
                    where user.Age > 25
                    select user;
foreach (User user in selectedUsers)
    Console.WriteLine($"{user.Name} - {user.Age}");
```

Пример сложных фильтров:

В классе пользователя есть список языков, которыми владеет пользователь. Необходимо отфильтровать пользователей по языку:

```
var selectedUsers = users.SelectMany(u => u.Languages,
                                (u, l) => new { User = u, Lang = l })
```

```
                            .Where(u    =>    u.Lang    ==
"английский" && u.User.Age < 28)
                            .Select(u=>u.User);
```

Метод SelectMany() в качестве первого параметра принимает последовательность, которую надо проецировать, а в качестве второго параметра - функцию преобразования, которая применяется к каждому элементу. На выходе она возвращает 8 пар "пользователь - язык" (new { User = u, Lang = l }), к которым потом применяетс фильтр с помощью Where.

Проекция позволяет спроектировать из текущего типа выборки какой-то другой тип. Для проекции используется оператор select. Допустим, имеется набор объектов следующего класса, представляющего пользователя. Необходимо выбрать только свойство Name.

Для этого необходимо выполнить следующий код:

```
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.

В LINQ можно выбирать объекты не только из одного, но и из большего количества источников:

Например, существуют следующие классы:

```
class Phone
{
    public string Name { get; set; }
    public string Company { get; set; }
}
class User
{
    public string Name { get; set; }
    public int Age { get; set; }
}
```

На основе этих классов создается выборка из двух источников данных:

```
List<User> users = new List<User>()
{
    new User { Name = "Sam", Age = 43 },
    new User { Name = "Tom", Age = 33 }
};
List<Phone> phones = new List<Phone>()
```

```
    {
        new Phone {Name="Lumia 630", Company="Microsoft"
},
        new Phone {Name="iPhone 6", Company="Apple"},
    };

    var people = from user in users
                 from phone in phones
                 select new { Name = user.Name, Phone =
phone.Name };

    foreach (var p in people)
        Console.WriteLine($"{p.Name} - {p.Phone}");
```

5 Порядок выполнения работы

Создать два класса по темам, представленным ниже. Осуществить выборку на основе двух источников данных с использованием LINQ.

1. Больница.
2. Компьютерные сети.
3. Библиотека.
4. Бытовая техника.
5. Образование.
6. Телекоммуникации. Аудио- и видеотехника.
7. Информационные технологии.
8. Офисное оборудование.
9. Образование (высшее).
10. Автомобилестроение.
11. Образование (базовое).
12. Банковская система.
13. Поликлиника.
14. Сельское хозяйство.
15. Производство продуктов.

6 Форма отчета о работе

Лабораторная работа № ___

Номер учебной группы _____

Фамилия, инициалы учащегося _____

Дата выполнения работы _____

Тема работы: _____

Цель работы: _____

Оснащение работы:

_____

Результат выполнения работы:

_____

_____

__

_____

__

7 Контрольные вопросы и задания

1. Что такое LINQ?
2. Какие виды фильтрации выборки вы знаете?

8 Рекомендуемая литература

1. Рихтер, Дж. CLR via C#. Программирование на платформе Microsoft .NET Framework 4.5 на языке C# / Дж. Рихтер. СПб. : Изд-во Питер, 2021. 896 с.

2. Прайс, М. Дж. C# 10 и .NET 6. Современная кросс-платформенная разработка / М. Дж. Прайс. СПб : Изд-во Питер, 2023. 848 с.

3. Васильев, А.Н. Программирование на C# для начинающих. Особенности языка / А.Н. Васильев. М. : Эксмо, 2022. 528 с.

4. Фримен, А. ASP.NET Core 3 с примерами на C# для профессионалов / А. Фримен. СПб. : Изд-во Вильямс, 2021. 1184 с.