МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение

высшего образования

**«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Кафедра «Вычислительная техника»

Дисциплина «Технология программирования»

**Лабораторная работа №1.**

**LINQ to Objects**

Выполнил:

студент группы ИВТАПбд-21

Кондратьев П. С.

Проверил:

Фролов В. А.

Ульяновск, 2018

По большей части LINQ ориентирован на запросы - будь то запросы, возвращающие набор подходящих объектов, единственный объект или подмножество полей из объекта либо набора объектов. В LINQ этот возвращенный набор называется последовательностью (sequence). Большинство последовательностей LINQ имеют тип IEnumerable<T>, где T — тип данных объектов, находящихся в последовательности. Например, если есть последовательность целых чисел, они должны храниться в переменной типа IEnumerable<int>. Вы увидите, что IEnumerable<T> буквально господствует в LINQ и очень многие методы LINQ возвращают IEnumerable<T>.

**Грамматика выражений запросов**

* Выражения запросов должны подчиняться перечисленным ниже правилам:
* Выражение должно начинаться с конструкции from.
* Остальная часть выражения может содержать ноль или более конструкций from, let или where. Конструкция from — это генератор, который объявляет одну или более переменных диапазона, перечисляющих последовательность или соединение нескольких последовательностей. Конструкция let представляет переменную диапазона и присваивает ей значение. Конструкция where фильтрует элементы из входной последовательности или соединения несколько входных последовательностей в выходную последовательность.
* Остальная часть выражения запроса может затем включать конструкцию orderby, содержащую одно или более полей сортировки с необязательным направлением упорядочивания. Направлением может быть ascending (по возрастанию) или descending (по убыванию).
* Затем в оставшейся части выражения может идти конструкция select или group.
* Наконец в оставшейся части выражения может следовать необязательная конструкция продолжения. Такой конструкцией может быть либо into, ноль или более конструкций join, или же другая повторяющаяся последовательность перечисленных элементов, начиная с конструкций из правила 2. Конструкция into направляет результаты запроса в воображаемую выходную последовательность, которая служит конструкцией from для последующих выражения запросов, начиная с конструкций из правила 2.

**Техническое задание**

* + создать 2 связанные коллекции и выполнить к ним 3 LINQ запроса:
  + к первой коллекции
  + ко второй
  + к их объединению

Исходный код

namespace ConsoleApplication1

{

public class Actor

{

public int id;

public int cost;

public string firstName;

public string lastName;

public Actor(int id, int cost, string firstName, string lastName) {

this.id = id;

this.cost = cost;

this.firstName = firstName;

this.lastName = lastName;

}

}

public class Test

{

public int id; public int yearPopular;

public Test(int id, int yearPopular)

{

this.id = id;

this.yearPopular = yearPopular;

}

}

class Program

{

static void CommonRequest(Action[] people, Test[] test)

{ }

static void Main(string[] args)

{

Actor p1 = new Actor(1, 1030, "Keanu", "Wilson");Actor p2 = new Actor(2, 1030, "Jam", "Wilson");

Actor p3 = new Actor(3, 3100, "Sandra", "Reeves");Actor p4 = new Actor(4, 700, "Gim", "Vollen");

Actor[] people = new Actor[] { p1, p2, p3, p4 };

Test t1 = new Test(1, 1987);

Test t2 = new Test(2, 1998);

Test t3 = new Test(3, 2007);

Test t4 = new Test(4, 2002);

Test[] test = new Test[] { t1, t2, t3, t4 };

var subset = people.TakeWhile(s => s.cost < 1500); var subset = from s in people where (s.cost > 1000) select s;

foreach (var s in subset)

{

Console.WriteLine("{0} - {1} - {2} - {3}", s.id, s.cost, s.firstName, s.lastName);

}

var subset = people.Min();

Console.WriteLine(subset);

|преобразовать в лист или массив|

ToArray() / ToList()

|проверка всей коллекции|

bool all = people.All(s => s.cost < 3000);

Console.WriteLine("Правда ли, что все разценки коллекции people меньше 3000: " + all);

IEnumerable<Actor> auto = people.OrderBy(s => s.cost);

foreach (Actor str in auto)

Console.WriteLine("{0} - {1} - {2} - {3}", s.id, s.cost, s.firstName, s.lastName);

IEnumerable<Actor> auto = people.Take(4).Concat(people.Skip(4));

foreach (Actor str in auto)

Console.WriteLine(str);

IEnumerable<Actor> auto = people.OrderBy(s => s.cost);

foreach (Actor str in auto)

Console.WriteLine("{0} - {1} - {2} - {3}", str.id, str.cost, str.firstName, str.lastName);

var item = people.Join(

test,

e => e.id, o => o.id, (e, o) => new

{

id = e.id, name = string.Format("{0} - {1}", e.firstName, e.lastName), year = o.yearPopular, cost = e.cost

});

foreach (var g in item)

Console.WriteLine(g);

}

}

}