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

Вершинин Д. В.

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

static void StudentsRequest(Student[] students) {

var subset = from s in students where (s.age>19 && s.averageMark > 4.4) select s;

foreach (var s in subset) {

Console.Write(s);

}

}

static void GrantRequest(Grants[] grants)

{

var subset = from g in grants where (g.cash > 300.0) select g;

foreach (var g in subset)

{

Console.Write(g);

}

}

static void CommonRequest(Grants[] grants, Student[] students) {

var subset = grants.Join(

students,

e => e.id,

o => o.id,

(e, o) => new

{

name = string.Format("{0} {1}", o.name, o.secName),

sumOfGrant = e.cash,

DateOfGrant = e.date

}); ;

foreach (var g in subset)

{

Console.WriteLine(g);

}

}

static void Main(string[] args)

{

Student[] students = new Student[5];

students[0] = new Student(0, "Ivan", "Ivanov", 19, 4.1);

students[1] = new Student(1, "Dmitry", "Petrov", 20, 4.8);

students[2] = new Student(2, "Alexander", "Shutov", 18, 3.5);

students[3] = new Student(3, "Gleb", "Gulyaev", 22, 5);

students[4] = new Student(4, "Dmitriy", "Vershinin", 25, 4.4);

Grants[] grants = new Grants[3];

grants[0] = new Grants(0, 456.55, "21.05.18");

grants[1] = new Grants(2, 886.577, "13.06.18");

grants[2] = new Grants(4, 210.123, "17.04.18");

StudentsRequest(students);

Console.WriteLine();

GrantRequest(grants);

Console.WriteLine();

CommonRequest(grants, students);

}