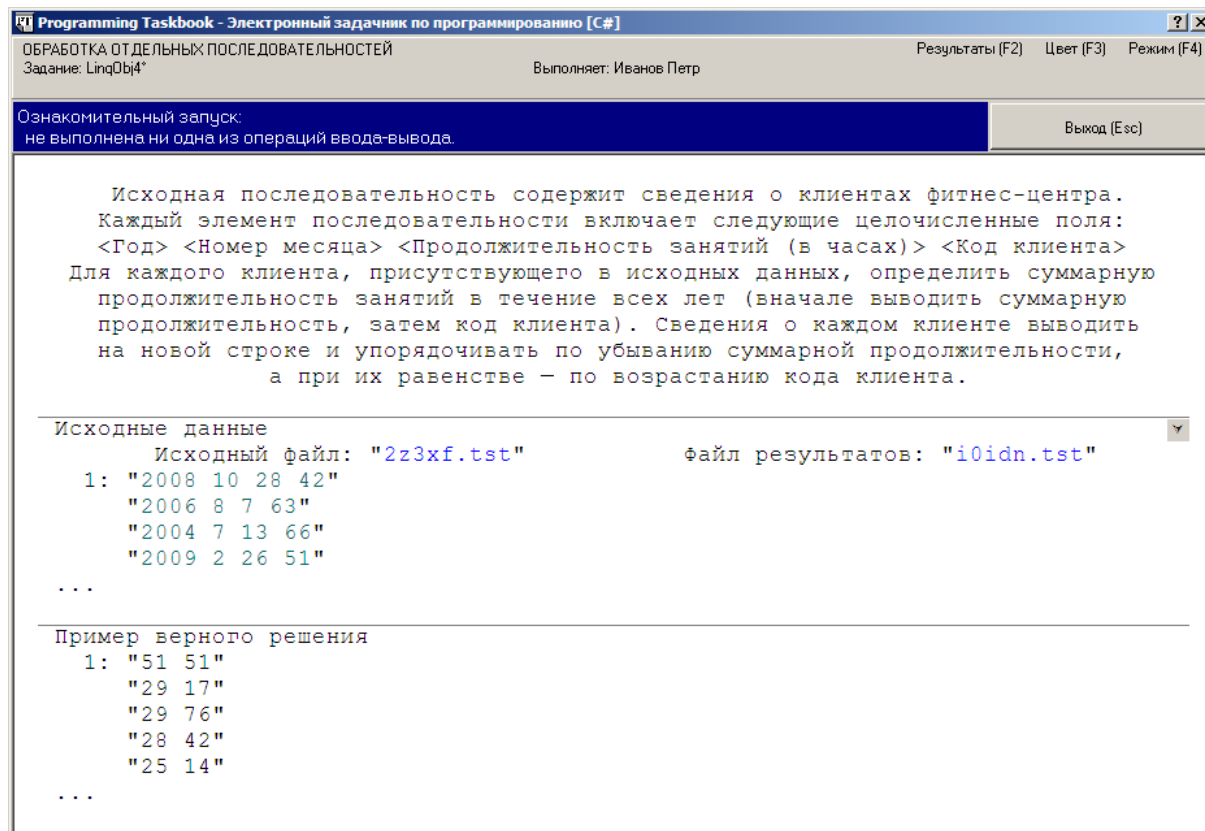


Простое задание на обработку отдельной последовательности: LinqObj4



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO;
using PT4;

namespace PT4Tasks
{
    public class MyTask : PT
    {
        // Для чтения набора строк из исходного текстового файла
        // в массив a типа string[] используйте оператор:
        //
        // a = File.ReadAllLines(GetString(), Encoding.Default);
        //
        // Для записи последовательности r типа IEnumerable<string>
        // в результирующий текстовый файл используйте оператор:
        //
        // File.WriteAllLines(GetString(), r.ToArray(), Encoding.Default);
        //
        // При решении задач группы LinqObj доступны следующие
        // дополнительные методы расширения, определенные в задачнике:
```

```
//  
// Show() и Show(cmt) - отладочная печать последовательности,  
//   cmt - строковый комментарий;  
//  
// Show(e => r) и Show(cmt, e => r) - отладочная печать  
//   значений r, полученных из элементов e последовательности,  
//   cmt - строковый комментарий.  
  
public static void Solve()  
{  
    Task("LinqObj4");  
}  
}
```

Исходная последовательность содержит сведения о клиентах фитнес-центра. Каждый элемент последовательности включает следующие целочисленные поля: <Год> <Номер месяца> <Продолжительность занятий (в часах)> <Код клиента>. Для каждого клиента, присутствующего в исходных данных, определить суммарную продолжительность занятий в течение всех лет (вначале выводить суммарную продолжительность, затем код клиента). Сведения о каждом клиенте выводить на новой строке и упорядочивать по убыванию суммарной продолжительности, а при их равенстве – по возрастанию кода клиента.

Исходные данные

Исходный файл: "2z3xf.tst" файл результатов: "i0idn.tst"

```
1: "2008 10 28 42"  
2: "2006 8 7 63"  
3: "2004 7 13 66"  
4: "2009 2 26 51"  
5: "2007 9 3 80"  
6: "2000 6 29 76"  
7: "2003 6 1 36"  
8: "2007 5 25 91"  
9: "2003 6 29 17"  
10: "2003 2 25 14"  
11: "2010 5 25 51"
```

Пример верного решения

```
1: "51 51"  
2: "29 17"  
3: "29 76"  
4: "28 42"  
5: "25 14"  
6: "25 91"  
7: "13 66"  
8: "7 63"  
9: "3 80"  
10: "1 36"
```

Programming Taskbook - Электронный задачник по программированию [C#]

ОБРАБОТКА ОТДЕЛЬНЫХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ
Задание: LinqObj4*

Выполняет: Иванов Петр

Результаты (F2) Цвет (F3) Режим (F4)

Ознакомительный запуск:
не выполнена ни одна из операций ввода-вывода. Выход (Esc)

-

+ Исходная последовательность содержит сведения о клиентах фитнес-центра. Каждый элемент последовательности включает следующие целочисленные поля: <Год> <Номер месяца> <Продолжительность занятий (в часах)> <Код клиента> Для каждого клиента, присутствующего в исходных данных, определить суммарную продолжительность занятий в течение всех лет (вначале вывести суммарную продолжительность, затем код клиента). Сведения о каждом клиенте выводить на новой строке и упорядочивать по убыванию суммарной продолжительности, а при их равенстве – по возрастанию кода клиента.

/

Исходные данные

Исходный файл: "2z3xf.tst" файл результатов: "i0idn.tst"

1: "2008 10 28 42"

2: "2006 8 7 63"

Programming Taskbook - Электронный задачник по программированию [C#]

ОБРАБОТКА ОТДЕЛЬНЫХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ
Задание: LinqObj4*

Выполняет: Иванов Петр

Результаты (F2) Цвет (F3) Режим (F4)

Ознакомительный запуск:
не выполнена ни одна из операций ввода-вывода. Выход (Esc)

- Исходные данные

+ Исходный файл: "2z3xf.tst" файл результатов: "i0idn.tst"

/

1: "2008 10 28 42"

2: "2006 8 7 63"

3: "2004 7 13 66"

4: "2009 2 26 51"

5: "2007 9 3 80"

6: "2000 6 29 76"

7: "2003 6 1 36"

8: "2007 5 25 91"

9: "2003 6 29 17"

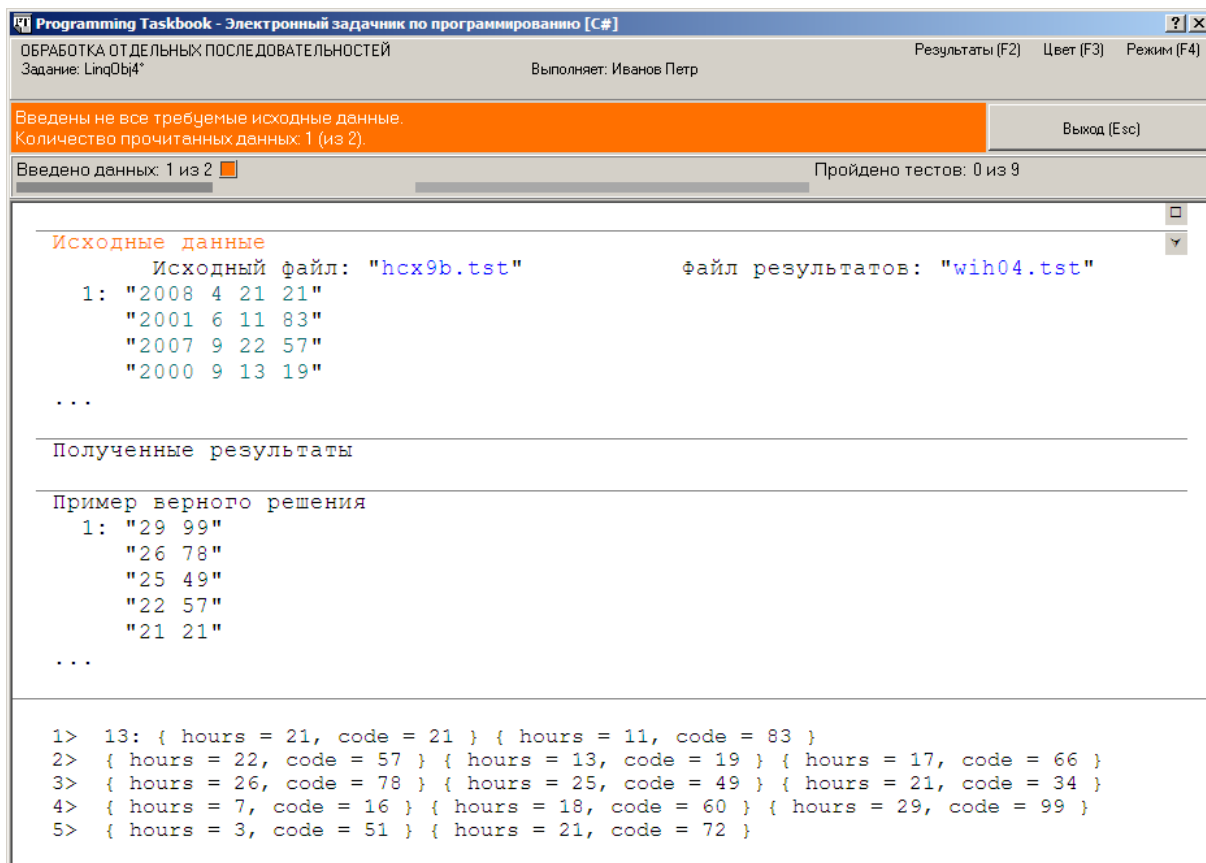
10: "2003 2 25 14"

11: "2010 5 25 51"

```
File.ReadLines(GetString(), Encoding.Default)
    .Select(e =>
    {
        string[] s = e.Split(' ');
        return new
        {
            hours = int.Parse(s[2]),
            code = int.Parse(s[3])
        };
    })
    .Show();
```

Или:

```
.Select(e => new
{
    hours = int.Parse(e.Split[2]),
    code = int.Parse(e.Split[3])
})
```



```
.GroupBy(e => e.code)
.OrderByDescending(e => e.Sum(c => c.hours))
.ThenBy(e => e.Key)
.Select(e => e.Sum(c => c.hours) + " " + e.Key)
```

```
.GroupBy(e => e.code,
    (k, ee) => new {k, sum = ee.Sum(c => c.hours)})
.OrderByDescending(e => e.sum).ThenBy(e => e.k)
.Select(e => e.sum + " " + e.k)
```

```
File.WriteAllLines(GetString(), r.ToArray(), Encoding.Default);
```

```
public static void Solve()
{
    Task("LinqObj4");
    var r = File.ReadLines(GetString(), Encoding.Default)
        .Select(e =>
        {
            string[] s = e.Split(' ');
            return new
            {
                hours = int.Parse(s[2]),
                code = int.Parse(s[3])
            };
        })
}
```

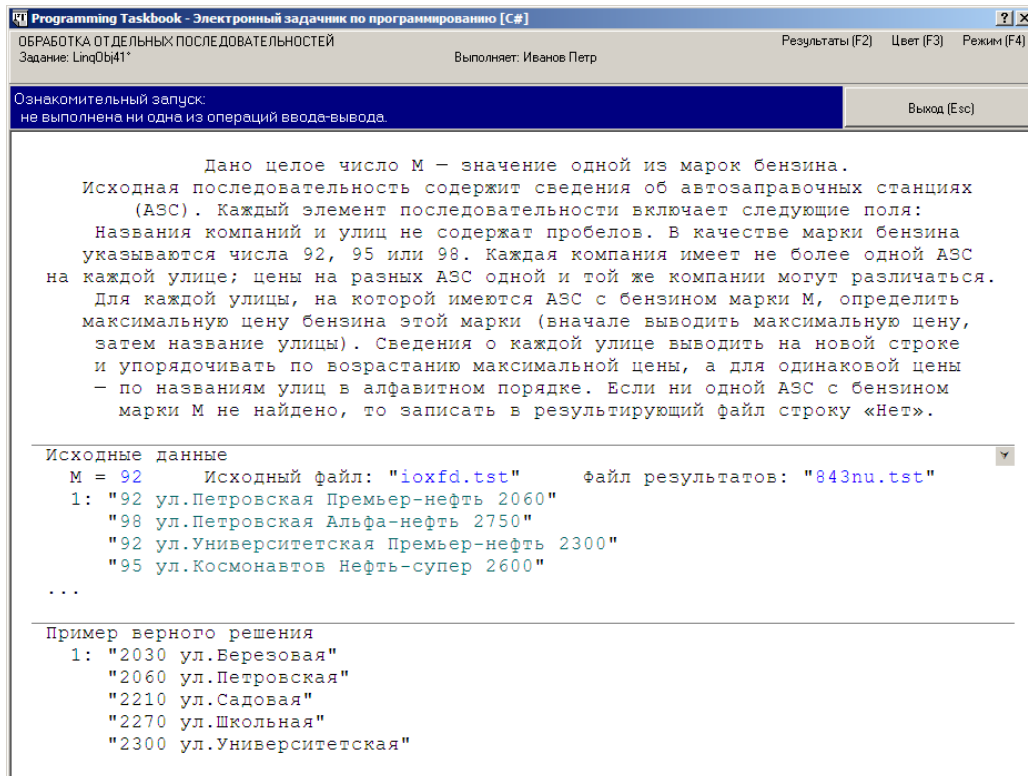
```
.GroupBy(e => e.code,
    (k, ee) => new { k, sum = ee.Sum(c => c.hours) })
.OrderByDescending(e => e.sum).ThenBy(e => e.k)
.Select(e => e.sum + " " + e.k);
File.WriteAllLines(GetString(), r.ToArray(), Encoding.Default);
}
```

Примечание. В некоторых задачах группы `LinqObj` в начало файла требуется записать *отдельные* строки (не входящие в последовательность). Чтобы преобразовать строку `s` в одноэлементный массив, достаточно использовать выражение `new string[] { s }`; напомним, что для объединения последовательностей можно использовать метод `Concat`.

Вариант решения, использующий выражение запроса:

```
public static void Solve()
{
    Task("LinqObj4");
    var r =
        from e in File.ReadLines(GetString(), Encoding.Default)
        let s = e.Split(' ')
        select new
        {
            hours = int.Parse(s[2]),
            code = int.Parse(s[3])
        }
        into e
        group e.hours by e.code
        into e
        let sum = e.Sum()
        orderby sum descending, e.Key
        select sum + " " + e.Key;
    File.WriteAllLines(GetString(), r.ToArray(), Encoding.Default);
}
```

Более сложные задания на обработку отдельных последовательностей: *LinqObj41*, *LinqObj61*



```
public static void Solve()
{
    Task("LinqObj41");
    int m = GetInt();
    var r = File.ReadLines(GetString(), Encoding.Default)
        .Select(e =>
        {
            string[] s = e.Split(' ');
            return new
            {
                brand = int.Parse(s[0]),
                street = s[1],
                price = int.Parse(s[3])
            };
        })
        .Where(e => e.brand == m)
        .GroupBy(e => e.street,
            (k, ee) => new { street = k, max = ee.Max(e => e.price) })
        .OrderBy(e => e.max).ThenBy(e => e.street)
        .Select(e => e.max + " " + e.street)
        .DefaultIfEmpty("Нет");
    File.WriteAllLines(GetString(), r.ToArray(), Encoding.Default);
}
```

```
public static void Solve()
{
    Task("LinqObj41");
    int m = GetInt();
    var r =
        (from e in File.ReadLines(GetString(), Encoding.Default)
         let s = e.Split(' ')
         select new
         {
             brand = int.Parse(s[0]),
             street = s[1],
             price = int.Parse(s[3])
         }
         into e
         where e.brand == m
         group e.price by e.street
         into e
         let max = e.Max()
         orderby max, e.Key
         select max + " " + e.Key)
        .DefaultIfEmpty("Her");
    File.WriteAllLines(GetString(), r.ToArray(), Encoding.Default);
}
```

ОБРАБОТКА ОТДЕЛЬНЫХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ
Задание: LinqObj61

Выполняет: Иванов Петр

Результаты (F2) Цвет (F3) Режим (F4)

Ознакомительный запуск:
не выполнена ни одна из операций ввода-вывода. Выход (Esc)

Исходная последовательность содержит сведения об оценках учащихся по трем предметам: алгебре, геометрии и информатике. Каждый элемент последовательности содержит данные об одной оценке и включает следующие поля: <фамилия> <Инициалы> <Класс> <Название предмета> <Оценка> Полных однофамильцев (с совпадающей фамилией и инициалами) среди учащихся нет. Класс задается целым числом, оценка – целое число в диапазоне 2-5. Название предмета указывается с заглавной буквы. Для каждого учащегося определить среднюю оценку по каждому предмету и вывести ее с двумя дробными знаками (если по какому-либо предмету учащийся не получил ни одной оценки, то вывести для этого предмета 0.00). Сведения о каждом учащемся выводить на отдельной строке, указывая фамилию, инициалы и средние оценки по алгебре, геометрии и информатике. Данные располагать в алфавитном порядке фамилий и инициалов.

Исходные данные

Исходный файл: "m6o7v.tst" файл результатов: "md11q.tst"

1: "Александров Я.Ф. 9 Информатика 2"
"Юрьева И.А. 9 Геометрия 2"
"Юрьева И.А. 9 Алгебра 4"
"Кузнецов Ю.Л. 11 Алгебра 4"
...

Пример верного решения

1: "Александров Я.Ф. 3.20 3.67 3.50"
"Белкин Д.Н. 4.00 4.33 3.75"
"Белкин О.В. 3.67 3.00 3.33"
"Веляева Р.О. 3.67 3.50 4.50"
"Греков С.Ф. 3.67 4.00 4.25"
...

```
public static void Solve()
{
    Task("LinqObj61");
    string[] subjects = { "Алгебра", "Геометрия", "Информатика" };
    var culture = new System.Globalization.CultureInfo("en-US");
    var r = File.ReadLines(GetString(), Encoding.Default)
        .Select(e =>
        {
            string[] s = e.Split(' ');
            return new
            {
                name = s[0] + " " + s[1],
                subj = s[3],
                mark = int.Parse(s[4])
            };
        })
        .GroupBy(e => e.name, (k, ee) => new
        {
            name = k,
            avrs = subjects
                .GroupJoin(ee, s => s, e => e.subj,
                    (s1, ee1) => ee1.Select(e1 => e1.mark)
                    .DefaultIfEmpty().Average())
        })
        .OrderBy(e => e.name)
        .Select(e => e.name + e.avrs.Aggregate("",
            (a, d) => a + " " + d.ToString("f2", culture)));
    File.WriteAllLines(GetString(), r.ToArray(), Encoding.Default);
}
```

```
public static void Solve()
{
    Task("LinqObj61");
    string[] subjects = { "Алгебра", "Геометрия", "Информатика" };
    var culture = new System.Globalization.CultureInfo("en-US");
    var r =
        from e in File.ReadLines(GetString(), Encoding.Default)
        let s = e.Split(' ')
        select new
        {
            name = s[0] + " " + s[1],
            subj = s[3],
            mark = int.Parse(s[4])
        }
        into e
        group e by e.name
```



```
into e
let avrs =
    from e1 in subjects
    join e2 in e
    on e1 equals e2.subj
    into ee
    select
        (from ee1 in ee
         select ee1.mark)
        .DefaultIfEmpty().Average()
orderby e.Key
select e.Key + avrs.Aggregate("",
    (a, d) => a + " " + d.ToString("f2", culture));
File.WriteAllLines(GetString(), r.ToArray(), Encoding.Default);
}
```

Обработка взаимосвязанных последовательностей: *LinqObj98*

LinqObj98. Даны последовательности *B*, *C*, *D* и *E*, включающие следующие поля:

B: <Страна-производитель> <Артикул товара> <Категория>

C: <Скидка (в процентах)> <Код потребителя> <Название магазина>

D: <Артикул товара> <Цена (в рублях)> <Название магазина>

E: <Код потребителя> <Артикул товара> <Название магазина>

Свойства последовательностей описаны в преамбуле к данной подгруппе заданий. Для каждой категории товаров и каждого магазина, указанного в *E*, определить суммарный размер скидки на все товары данной категории, проданные в данном магазине (вначале выводится название категории, затем название магазина, затем суммарная скидка). При вычислении размера скидки на товар копейки отбрасываются. Если на проданный товар скидка отсутствует, то ее размер полагается равным 0. Если для некоторой категории товаров в каком-либо магазине не было продаж, то суммарная скидка для этой пары «категория–магазин» полагается равной –1. Сведения о каждой паре «категория–магазин» выводить на новой строке и упорядочивать по названиям категорий в алфавитном порядке, а для одинаковых названий категорий — по названиям магазинов (также в алфавитном порядке).

```
Исходные данные
Исходные файлы: "i7.tst" "lz.tst" "ou.tst" "tu.tst" файл результатов: "pc.tst"
1: "Германия ML100-0037 Авто"
1: "20 176 Альфа"
1: "XV197-0617 5980 Титул"
1: "235 XV197-0617 Полет"
...

Пример верного решения
1: "Авто Альфа 1647"
   "Авто Весна 2916"
   "Авто Гинея 0"
   "Авто Лидер -1"
   "Авто Полет 175"
...
```

```
- 37: "20 112 Лидер"
+ 38: "10 179 Весна"
/ 39: "25 105 Лидер"
40: "40 151 Супер"
41: "5 271 Лидер"
42: "5 115 Титул"
1: "XV197-0617 5980 Титул"
2: "IS190-0065 550 Полет"
3: "PT030-0169 1870 Гинея"
4: "MR090-0055 500 Полет"
5: "HN103-0058 490 Весна"
6: "HI172-1225 10290 Лидер"
7: "PH146-0579 5840 Полет"
8: "MW145-0328 2880 Альфа"
9: "RB122-0048 570 Весна"
10: "TR087-0094 870 Супер"
11: "MS142-0062 510 Гинея"
12: "MS142-0062 630 Альфа"
13: "XS023-0096 840 Альфа"
14: "IJ054-0044 420 Весна"
15: "XV197-0617 5240 Полет"
16: "IJ054-0044 470 Лидер"
17: "LX118-1226 13110 Весна"
18: "MW145-0328 3240 Весна"
19: "LN187-0054 550 Альфа"
20: "TR087-0094 820 Полет"
21: "IJ054-0044 510 Полет"
22: "RB122-0048 480 Титул"
23: "TR087-0094 940 Титул"
1: "235 XV197-0617 Полет"
2: "176 MW145-0328 Альфа"
3: "125 XS023-0096 Альфа"
4: "179 HN103-0058 Весна"
5: "118 LN187-0054 Альфа"
6: "151 MS142-0062 Альфа"
7: "228 XV197-0617 Полет"
```

Первый этап решения состоит в формировании последовательностей анонимных типов на основе исходных строковых последовательностей. Для большей наглядности присвоим этим последовательностям имена, которые используются в условии задачи:

```
var B = File.ReadLines(GetString(), Encoding.Default)
    .Select(e =>
    {
        string[] s = e.Split(' ');
        return new
        {
            art = s[1],
            cat = s[2]
        };
    });
var C = File.ReadLines(GetString(), Encoding.Default)
    .Select(e =>
    {
        string[] s = e.Split(' ');
        return new
        {
            discount = int.Parse(s[0]),
            code = s[1],
            shop = s[2]
        };
    });
var D = File.ReadLines(GetString(), Encoding.Default)
    .Select(e =>
    {
        string[] s = e.Split(' ');
        return new
        {
            art = s[0],
            price = int.Parse(s[1]),
            shop = s[2]
        };
    });
var E = File.ReadLines(GetString(), Encoding.Default)
    .Select(e =>
    {
        string[] s = e.Split(' ');
        return new
        {
            code = s[0],
            art = s[1],
            shop = s[2]
        };
    });
```

- art — «Артикул товара» (строка),
- cat — «Категория» (строка),
- discount — «Скидка (в процентах)» (целое число),
- code — «Код потребителя» (строка),
- shop — «Название магазина» (строка),
- price — «Цена (в рублях)» (целое число).

```

Programming Taskbook - Электронный задачник по программированию [C#]
56> { discount = 20, code = 233, shop = Лидер }
57> { discount = 10, code = 104, shop = Люкс }
58> D 18: { art = MH115-0569, price = 5630, shop = Весна }
59> { art = IP019-0090, price = 910, shop = Титул }
60> { art = CL072-0025, price = 230, shop = Титул }
61> { art = MH115-0569, price = 6250, shop = Лукат }
62> { art = RS054-1024, price = 11570, shop = Весна }
63> { art = QY139-0025, price = 260, shop = Альфа }
64> { art = GI138-0161, price = 1910, shop = Титул }
65> { art = RD184-1857, price = 18570, shop = Альфа }
66> { art = RS071-0297, price = 2820, shop = Альфа }
67> { art = RD184-1857, price = 18750, shop = Титул }
68> { art = PG033-0072, price = 620, shop = Альфа }
69> { art = GU075-0089, price = 830, shop = Титул }
70> { art = CU173-0036, price = 410, shop = Люкс }
71> { art = QZ159-1072, price = 10180, shop = Лукат }
72> { art = CU173-0036, price = 400, shop = Весна }
73> { art = RS054-1024, price = 10240, shop = Люкс }
74> { art = CL072-0025, price = 260, shop = Весна }
75> { art = QG129-0692, price = 7260, shop = Альфа }
76> E 47: { code = 197, art = GU075-0089, shop = Титул }
77> { code = 287, art = RS054-1024, shop = Весна }
78> { code = 117, art = QG129-0692, shop = Альфа }

```

На втором этапе решения преобразуем последовательность *E*, добавив в нее дополнительные данные из последовательностей *B*, *C* и *D*.

```

E.Join(B, e1 => e1.art, e2 => e2.art,
      (e1, e2) => new { e1.code, e1.art, e1.shop, e2.cat })
.Join(D, e1 => e1.art + e1.shop, e2 => e2.art + e2.shop,
      (e1, e2) => new { e1.code, e1.art, e1.shop, e1.cat, e2.price })
.Show();

```

```

Programming Taskbook - Электронный задачник по программированию [C#]
1> 73: { code = 132, art = TC173-1507, shop = Весна, cat = Игрушки, price = 15070 }
2> { code = 277, art = TM168-0226, shop = Весна, cat = Игрушки, price = 2070 }
3> { code = 122, art = SS016-0316, shop = Гинея, cat = Быт.техника, price = 2970 }
4> { code = 133, art = CW197-0117, shop = Титул, cat = Компьютеры, price = 1280 }
5> { code = 122, art = SS016-0316, shop = Гинея, cat = Быт.техника, price = 2970 }
6> { code = 290, art = TQ057-0589, shop = Гинея, cat = Игрушки, price = 6120 }
7> { code = 272, art = CW197-0117, shop = Титул, cat = Компьютеры, price = 1280 }
8> { code = 156, art = UB161-0263, shop = Гинея, cat = Мебель, price = 2650 }
9> { code = 132, art = NR190-0335, shop = Титул, cat = Компьютеры, price = 3910 }
10> { code = 149, art = TQ057-0589, shop = Гинея, cat = Игрушки, price = 6120 }
11> { code = 185, art = CW197-0117, shop = Титул, cat = Компьютеры, price = 1280 }
12> { code = 130, art = SQ200-0071, shop = Титул, cat = Быт.техника, price = 830 }
13> { code = 103, art = TM168-0226, shop = Весна, cat = Игрушки, price = 2070 }

```

```

Programming Taskbook - Электронный задачник по программированию [C#]
ОБРАБОТКА НЕСКОЛЬКИХ ВЗАИМОСВЯЗАННЫХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ
Задача: LinqObj98*
Выполняет: Иванов Петр
Результаты (F2) Цвет (F3) Режим (F4)
Введены не все требуемые исходные данные.
Количество прочитанных данных: 4 (из 5).
Выход (Esc)
Введено данных: 4 из 5
Пройдено тестов: 0 из 9
- 1: "132 TC173-1507 Весна"
+ 2: "277 TM168-0226 Весна"
/ 3: "122 SS016-0316 Гинея"
4: "133 CW197-0117 Титул"
5: "122 SS016-0316 Гинея"
6: "290 TQ057-0589 Гинея"
7: "272 CW197-0117 Титул"

```

Осталось выполнить объединение полученной последовательности и последовательности *C*, в результате которого мы найдем *размер скидки* для каждого проданного товара. Скидка определяется парой «потребитель–магазин»: магазин может предоставлять некоторым потребителям фиксированную скидку *d* (в процентах) на все товары.

В данном случае нельзя применить метод `Join`, обеспечивающий получение внутреннего объединения, так как для некоторых возможных пар «потребитель–магазин» скидка не определена (т. е. некоторые потребители не имеют скидки в некоторых магазинах). Вместо этого используем метод `GroupJoin`, возвращающий *левое внешнее объединение*, в котором каждому элементу внешней последовательности (полученной ранее в результате выполненных объединений) сопоставляется *набор* элементов внутренней последовательности *C*. Объединение выполняется по составному ключу — коду потребителя и названию магазина. В результирующем объединении достаточно сохранять три поля: категорию товара, название магазина и размер скидки для данного товара (прочие поля были нужны для построения объединений и вычисления скидки, и в дальнейшем не потребуются):

```
.GroupJoin(C, e1 => e1.code + e1.shop, e2 => e2.code + e2.shop,
    (e1, ee2) => new { e1.shop, e1.cat,
        discount = ee2.Select(e => e.discount)
            .FirstOrDefault() * e1.price / 100 })
```

Представим в одном выражении все описанные выше операции объединения и сохраним полученную последовательность под именем `discounts`:

```
var discounts = E.Join(B, e1 => e1.art, e2 => e2.art,
    (e1, e2) => new { e1.code, e1.art, e1.shop, e2.cat })
    .Join(D, e1 => e1.art + e1.shop, e2 => e2.art + e2.shop,
    (e1, e2) => new { e1.code, e1.art, e1.shop, e1.cat, e2.price })
    .GroupJoin(C, e1 => e1.code + e1.shop, e2 => e2.code + e2.shop,
    (e1, ee2) => new
    {
        e1.shop,
        e1.cat,
        discount = ee2.Select(e => e.discount)
            .FirstOrDefault() * e1.price / 100
    });
```

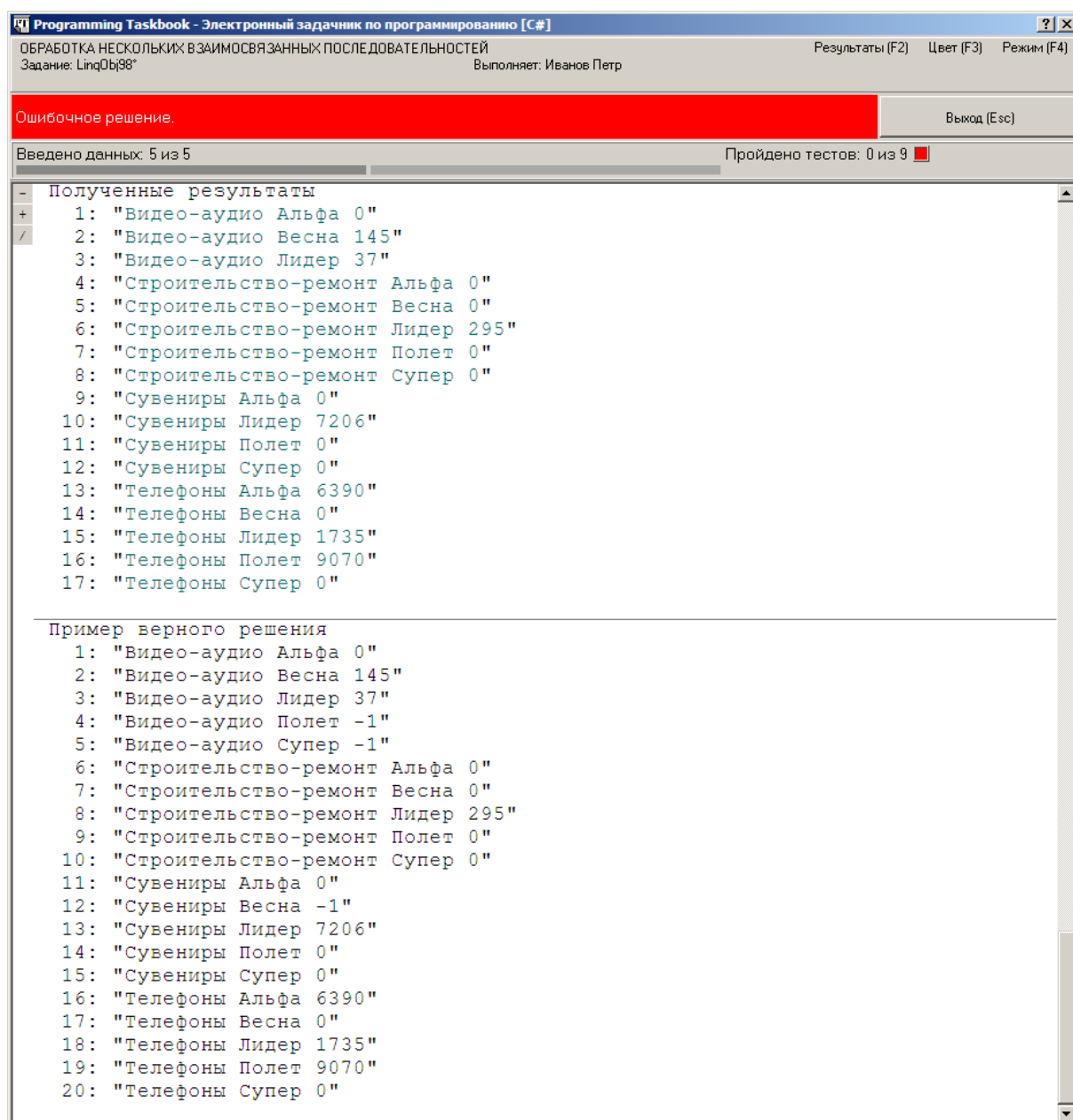
Перейдем к третьему, завершающему этапу решения задачи: обработке полученной последовательности.

Если бы в задаче требовалось вывести данные только о тех парах «категория–магазин», которые встречаются в последовательности `discounts`, то было бы достаточно выполнить группировку этой последовательности по составному ключу, включающему поля `cat` и `shop`, отсортировать полученную последовательность и найти сумму скидок по всем товарам, имеющим общий ключ:

```
var r = discounts
    .GroupBy(e => e.cat + " " + e.shop, e => e.discount)
    .OrderBy(e => e.Key)
    .Select(e => e.Key + " " + e.Sum());
```

Добавив к решению указанный фрагмент вместе с завершающим оператором `File.WriteAllLines(GetString(), r.ToArray(), Encoding.Default);`

обеспечивающим сохранение последовательности в текстовом файле, и запустив программу, мы получим результат, подобный приведенному на рис. 38. Отличие от правильного решения состоит в том, что в полученной последовательности отсутствует информация о парах «категория–магазин», которым *не соответствует ни одного проданного товара*.



Для того чтобы учесть отсутствующие пары, поступим следующим образом. Вначале сформируем набор *cats* всех категорий и набор *shops* всех магазинов, используя группировку последовательности *B* по ключу *cat* и группировку последовательности *E* по ключу *shop* (в сгруппированных последовательностях достаточно сохранить только набор ключей группировки):

```
var cats = B.GroupBy(e => e.cat, (k, ee) => k);  
var shops = E.GroupBy(e => e.shop, (k, ee) => k);
```

Затем получим перекрестное объединение наборов *cats* и *shops* в виде набора строк, используя комбинацию методов *SelectMany* и *Select* (см. п. 5.5.2) и отсортируем результат:

```
var cats_shops =  
    cats.SelectMany(e1 => shops.Select(e2 => e1 + " " + e2))  
        .OrderBy(e => e);
```

И наконец, выполним левое внешнее объединение последовательностей *cats_shops* и *discounts* методом *GroupJoin*, вернув результат в виде набора строк (дополнительная сортировка этого набора уже не потребуется):

```
var r = cats_shops.GroupJoin(discounts, e1 => e1,  
    e2 => e2.cat + " " + e2.shop, (e1, ee2) => e1 + " " +  
    ee2.Select(e => e.discount).DefaultIfEmpty(-1).Sum());
```

При обработке последовательности *ee2* мы учли, что она может оказаться пустой, причем в этом случае следует вернуть значение -1 .

Данное решение допускает две модификации: во-первых, при выполнении объединения промежуточной последовательности с последовательностью *D* в результирующей последовательности можно не сохранять поле «Артикул», так как в дальнейшем оно не потребуется; во-вторых, при последующем объединении этой последовательности с последовательностью *C* (методом *GroupJoin*) можно объединить поля «Категория» и «Магазин», поскольку в дальнейшем они всегда будут использоваться совместно.

Анализируя решение, нетрудно заметить, что все именованные вспомогательные последовательности (*discounts*, *cats*, *shops*, *cats_shops*) в дальнейшем коде используются по одному разу, и поэтому обращение к ним можно заменить их инициализирующими выражениями. Однако к подобной возможности следует относиться с осторожностью, чтобы не получить излишне громоздкое и поэтому сложное для восприятия выражение. Можно, например, сохранить имена *discounts*, *cats*, *shops* и отказаться от именованного перекрестного объединения *cats_shops*.

```
public static void Solve()
{
    Task("LinqObj98");

    // здесь должен находиться фрагмент, обеспечивающий
    // ввод исходных последовательностей B, C, D и E

    var discounts = E.Join(B, e1 => e1.art, e2 => e2.art,
        (e1, e2) => new { e1.code, e1.art, e1.shop, e2.cat })
        .Join(D, e1 => e1.art + e1.shop, e2 => e2.art + e2.shop,
            (e1, e2) => new { e1.code, e1.shop, e1.cat, e2.price })
        .GroupJoin(C, e1 => e1.code + e1.shop, e2 => e2.code + e2.shop,
            (e1, ee2) => new
            {
                cat_shop = e1.cat + " " + e1.shop,
                discount = ee2.Select(e => e.discount)
                    .FirstOrDefault() * e1.price / 100
            });
    var cats = B.GroupBy(e => e.cat, (k, ee) => k);
    var shops = E.GroupBy(e => e.shop, (k, ee) => k);
    var r = cats
        .SelectMany(e1 => shops.Select(e2 => e1 + " " + e2))
        .OrderBy(e => e)
        .GroupJoin(discounts, e1 => e1, e2 => e2.cat_shop,
            (e1, ee2) => e1 + " " +
            ee2.Select(e => e.discount).DefaultIfEmpty(-1).Sum());
    File.WriteAllLines(GetString(), r.ToArray(), Encoding.Default);
}
```



```
public static void Solve()
{
    Task("LinqObj98");

    // здесь должен находиться фрагмент, обеспечивающий
    // ввод исходных последовательностей B, C, D и E

    var r =
        from e1 in
            // нахождение последовательности cats
            from e in B
            group e by e.cat
            // конец вложенного запроса
        from e2 in
            // нахождение последовательности shops
            from e in E
            group e by e.shop
            // конец вложенного запроса
        let s = e1.Key + " " + e2.Key
        orderby s
        select s
        into e1
        join e2 in
            // нахождение последовательности discounts
            from e1 in E
            join e2 in B
            on e1.art equals e2.art
            join e3 in D
            on e1.art + e1.shop equals e3.art + e3.shop
            join e4 in C
            on e1.code + e1.shop equals e4.code + e4.shop
            into ee4
        select new
        {
            cat_shop = e2.cat + " " + e1.shop,
            discount = (from e in ee4 select e.discount)
                .FirstOrDefault() * e3.price / 100
        }
        // конец вложенного запроса
        on e1 equals e2.cat_shop
        into ee2
        select e1 + " " +
            (from e in ee2 select e.discount)
                .DefaultIfEmpty(-1).Sum();
    File.WriteAllLines(GetString(), r.ToArray(), Encoding.Default);
}
```

Завершая обсуждение задачи LinqObj98, приведем фрагмент кода, содержащий выражение запроса для более простой задачи, в которой требуется вывести информацию только о тех парах «категория–магазин», которые присутствуют в последовательности проданных товаров (ранее мы приводили соответствующий фрагмент, использующий вызовы методов LINQ):

```
var r =
    from e1 in E
    join e2 in B
    on e1.art equals e2.art
    join e3 in D
    on e1.art + e1.shop equals e3.art + e3.shop
    join e4 in C
    on e1.code + e1.shop equals e4.code + e4.shop
    into ee4
    select new
    {
        cat_shop = e2.cat + " " + e1.shop,
        discount = (from e in ee4 select e.discount)
            .FirstOrDefault() * e3.price / 100
    }
    into e
    orderby e.cat_shop
    group e.discount by e.cat_shop
    into e
    select e.Key + " " + e.Sum();
```