Технология LINQ. Учебные задания

Указания к выполнению заданий

Структура программы, выполняющей задание из группы LinqBegin, приведена в преамбуле к данной группе.

При выполнении заданий из группы LinqObj требуется считывать данные из исходного текстового файла (или нескольких файлов) и записывать их в файл результатов. Для организации файлового ввода-вывода в данном случае удобно использовать новые методы класса `File`, появившиеся в библиотеке .NET версии 3.0: `ReadAllLines` и `WriteAllLines`. Кроме того, следует учитывать, что данные в исходных файлах хранятся в однобайтной кодировке Windows (кодовая страница 1251), и в этой же кодировке должны сохраняться данные в результирующем файле; поэтому в методах `ReadAllLines` и `WriteAllLines` необходимо указывать дополнительный параметр `Encoding.Default`. В тех заданиях, которые связаны с выводом вещественных числовых данных, необходимо дополнительно обеспечить их форматирование с использованием точки в качестве десятичного разделителя. Для этого достаточно использовать *форматирующий* метод `ToString` с дополнительным параметром-*культурой* (см. приведенный ниже пример решения задачи LinqObj61).

В подавляющем большинстве задач для требуемого преобразования исходной последовательности достаточно использовать единственную цепочку последовательных вызовов методов LINQ To Object.

По поводу выполнения заданий из второй подгруппы группы LinqObj, в которой требуется обработать *несколько взаимосвязанных последовательностей*, можно привести некоторые дополнительные рекомендации. Задачи этой подгруппы делятся на два основных типа. В задачах первого, более простого типа требуется использовать *внутреннее объединение* (т. е. надо обрабатывать только те пары значений, которые могут быть получены из исходных последовательностей). Поэтому для таких задач можно сразу выполнить вызов всех необходимых команд `Join`, обеспечив требуемое объединение всех исходных последовательностей в итоговую последовательность.

В задачах второго типа необходимо выполнить обработку всех возможных комбинаций наборов значений из исходных последовательностей, в том числе тех, которые в этих наборах не представлены. В данной ситуации нельзя использовать метод `Join`, так как этот метод не включает отсутствующие пары в результирующую последовательность; поэтому требуется либо предварительно выполнять *группировку* исходных последовательностей с последующим построением всех возможных пар (т. е. построением *декартова произведения*) с помощью методов `SelectMany` и `Select`, либо (в некоторых специальных случаях) применять команду `GroupJoin`.

В задачах, требующих построения декартова произведения, может оказаться более удобным применять не комбинации методов `SelectMany` и `Select`, а соответствующие *выражения запросов* (конструкции `from`).

Ниже приводится пример решения задачи LinqObj61.

Дополнительные указания по выполнению заданий группы LinqObj приведены в следующей книге:

Абрамян М. Э. Технология LINQ на примерах. Практикум с использованием электронного задачника Programming Taskbook for LINQ. — М.: ДМК Пресс, 2014. — 326 с.

На сайте ptaskbook.com в разделе «PT for LINQ» содержатся примеры программ из данной книги, которые можно немедленно загрузить в веб-среду ProgrammingABC.NET WDE и запустить на выполнение.

Пример решения задачи LinqObj61

LinqObj61. 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.

Решение

```
public static void Solve()
{
  Task("LinqObj61");
  string[] subjects = { "Алгебра", "Геометрия",
    "Информатика" };
  var culture = new System.Globalization
    .CultureInfo("en-US");
// Ввод и обработка
  var r = File.ReadAllLines(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)));
// Вывод
  File.WriteAllLines(GetString(), r.ToArray(),
    Encoding.Default);
}
```

Знакомство с запросами LINQ

При вводе (выводе) последовательности вначале следует ввести (соответственно, вывести) ее размер, а затем ее элементы. Все входные последовательности являются непустыми. Выходные последовательности могут быть пустыми; в этом случае требуется вывести единственное число 0 — размер данной последовательности.

Если в задании идет речь о *порядковых номерах* элементов последовательности, то предполагается, что нумерация ведется от 1 (таким образом, порядковый номер элемента равен *индексу* этого элемента, *увеличенному на* 1).

Для обработки входной последовательности в большинстве заданий достаточно указать *единственный* оператор, содержащий вызовы нужных запросов LINQ to Objects и другие необходимые конструкции, в частности, операцию `??` языка C#.

При выполнении заданий с использованием задачника Programming Taskbook можно использовать дополнительные методы, определенные в задачнике:

• методы `GetEnumerableInt` и `GetEnumerableString` обеспечивают ввод исходных последовательностей с элементами целого и строкового типа соответственно (выполняется ввод размера последовательности и всех ее элементов, возвращается введенная последовательность);

• метод `Put` является методом расширения для последовательности и обеспечивает вывод этой последовательности (выводится размер последовательности и все ее элементы);

• метод `Show` также является методом расширения для последовательности; он обеспечивает печать последовательности в разделе отладки окна задачника и возвращает эту же последовательность (отладочная печать может сопровождаться *комментарием*, который указывается в качестве необязательного строкового параметра метода `Show`).

Использование вспомогательных методов иллюстрируется приведенным ниже фрагментом программы, решающей следующую задачу: извлечь из исходной целочисленной последовательности четные отрицательные числа и заменить порядок их следования на обратный.

```
// Ввод исходных данных
var a = GetEnumerableInt();
// Обработка
var res = a.Where(e => e % 2 == 0 && e < 0)
  .Reverse();
// Вывод результатов
res.Put();
```

Все этапы решения можно объединить в одном операторе, состоящем из цепочки последовательно вызываемых методов:

```
GetEnumerableInt()
  .Where(e => e % 2 == 0 && e < 0)
  .Reverse().Put();
```

Возможен вариант решения, в котором дополнительно выполняется отладочная печать (в данном случае полученная последовательность четных отрицательных чисел печатается перед изменением порядка следования ее элементов и после этого изменения):

```
GetEnumerableInt().Where(e => e % 2 == 0 && e < 0)
  .Show().Reverse().Show().Put();
```

Отладочная печать позволяет увидеть состояние последовательности на различных этапах ее преобразования и тем самым облегчает поиск ошибок.

Поэлементные операции, агрегирование и генерирование последовательностей

Изучаемые методы LINQ:

• First, FirstOrDefault, Last, LastOrDefault, Single, SingleOrDefault (поэлементные операции);

• Count, Sum, Average, Max, Min, Aggregate (агрегирование);

• Range (генерирование последовательностей).

LinqBegin1. Дана целочисленная последовательность, содержащая как положительные, так и отрицательные числа. Вывести ее первый положительный элемент и последний отрицательный элемент.

LinqBegin2. Дана цифра D (однозначное целое число) и целочисленная последовательность A. Вывести первый положительный элемент последовательности A, оканчивающийся цифрой D. Если требуемых элементов в последовательности A нет, то вывести 0.

LinqBegin3. Дано целое число L (> 0) и строковая последовательность A. Вывести последнюю строку из A, начинающуюся с цифры и имеющую длину L. Если требуемых строк в последовательности A нет, то вывести строку «Not found».

Указание. Для обработки ситуации, связанной с отсутствием требуемых строк, использовать операцию ??.

LinqBegin4. Дан символ C и строковая последовательность A. Если A содержит единственный элемент, оканчивающийся символом C, то вывести этот элемент; если требуемых строк в A нет, то вывести пустую строку; если требуемых строк больше одной, то вывести строку «Error».

Указание. Использовать try-блок для перехвата возможного исключения.

LinqBegin5. Дан символ C и строковая последовательность A. Найти количество элементов A, которые содержат более одного символа и при этом начинаются и оканчиваются символом C.

LinqBegin6. Дана строковая последовательность. Найти сумму длин всех строк, входящих в данную последовательность.

LinqBegin7. Дана целочисленная последовательность. Найти количество ее отрицательных элементов, а также их сумму. Если отрицательные элементы отсутствуют, то дважды вывести 0.

LinqBegin8. Дана целочисленная последовательность. Найти количество ее положительных двузначных элементов, а также их среднее арифметическое (как вещественное число). Если требуемые элементы отсутствуют, то дважды вывести 0 (первый раз как целое, второй — как вещественное).

LinqBegin9. Дана целочисленная последовательность. Вывести ее минимальный положительный элемент или число 0, если последовательность не содержит положительных элементов.

LinqBegin10. Дано целое число L (> 0) и строковая последовательность A. Строки последовательности A содержат только заглавные буквы латинского алфавита. Среди всех строк из A, имеющих длину L, найти наибольшую (в смысле лексикографического порядка). Вывести эту строку или пустую строку, если последовательность не содержит строк длины L.

LinqBegin11. Дана последовательность непустых строк. Используя метод Aggregate, получить строку, состоящую из начальных символов всех строк исходной последовательности.

LinqBegin12. Дана целочисленная последовательность. Используя метод Aggregate, найти произведение последних цифр всех элементов последовательности. Чтобы избежать целочисленного переполнения, при вычислении произведения использовать вещественный числовой тип.

LinqBegin13. Дано целое число N (> 0). Используя методы Range и Sum, найти сумму $1 + (1/2) + \ldots + (1/N)$ (как вещественное число).

LinqBegin14. Даны целые числа A и B ($A < B$). Используя методы Range и Average, найти среднее арифметическое квадратов всех целых чисел от A до B включительно: $(A^2 + (A+1)^2 + \ldots + B^2)/(B - A + 1)$ (как вещественное число).

LinqBegin15. Дано целое число N ($0 \le N \le 15$). Используя методы Range и Aggregate, найти *факториал* числа N: $N! = 1 \cdot 2 \cdot \ldots \cdot N$ при $N \ge 1$; $0! = 1$. Чтобы избежать целочисленного переполнения, при вычислении факториала использовать вещественный числовой тип.

Фильтрация, сортировка, теоретико-множественные операции

Изучаемые методы LINQ:
• Where, TakeWhile, SkipWhile, Take, Skip (фильтрация);
• OrderBy, OrderByDescending, ThenBy, ThenByDescending (сортировка);
• Distinct, Reverse (удаление повторяющихся элементов и инвертирование);
• Union, Intersect, Except (теоретико-множественные операции).

LinqBegin16. Дана целочисленная последовательность. Извлечь из нее все положительные числа, сохранив их исходный порядок следования.

LinqBegin17. Дана целочисленная последовательность. Извлечь из нее все нечетные числа, сохранив их исходный порядок следования и удалив все вхождения повторяющихся элементов, кроме первых.

LinqBegin18. Дана целочисленная последовательность. Извлечь из нее все четные отрицательные числа, поменяв порядок извлеченных чисел на обратный.

LinqBegin19. Дана цифра D (целое однозначное число) и целочисленная последовательность A. Извлечь из A все различные положительные числа, оканчивающиеся цифрой D (в исходном порядке). При наличии повторяющихся элементов удалять все их вхождения, кроме последних.
Указание. Последовательно применить методы Reverse, Distinct, Reverse.

LinqBegin20. Дана целочисленная последовательность. Извлечь из нее все положительные двузначные числа, отсортировав их по возрастанию.

LinqBegin21. Дана строковая последовательность. Строки последовательности содержат только заглавные буквы латинского алфавита. Отсортировать последовательность по возрастанию длин строк, а строки одинаковой длины — в лексикографическом порядке по убыванию.

LinqBegin22. Дано целое число K (> 0) и строковая последовательность A. Строки последовательности содержат только цифры и заглавные буквы латинского алфавита. Извлечь из A все строки длины K, оканчивающиеся цифрой, отсортировав их в лексикографическом порядке по возрастанию.

LinqBegin23. Дано целое число K (> 0) и целочисленная последовательность A. Начиная с элемента A с порядковым номером K, извлечь из A все нечетные двузначные числа, отсортировав их по убыванию.

LinqBegin24. Дано целое число K (> 0) и строковая последовательность A. Из элементов A, предшествующих элементу с порядковым номером K, извлечь те строки, которые имеют нечетную длину и начинаются с заглавной латинской буквы, изменив порядок следования извлеченных строк на обратный.

LinqBegin25. Даны целые числа K_1 и K_2 и целочисленная последовательность A; $1 \le K_1 < K_2 \le N$, где N — размер последовательности A. Найти сумму положительных элементов последовательности с порядковыми номерами от K_1 до K_2 включительно.

LinqBegin26. Даны целые числа K_1 и K_2 и последовательность непустых строк A; $1 < K_1 < K_2 \le N$, где N — размер последовательности A. Найти среднее арифметическое длин всех элементов последовательности, кроме элементов с порядковыми номерами от K_1 до K_2 включительно, и вывести его как вещественное число.

LinqBegin27. Дано целое число D и целочисленная последовательность A. Начиная с первого элемента A, большего D, извлечь из A все нечетные положительные числа, поменяв порядок извлеченных чисел на обратный.

LinqBegin28. Дано целое число L (> 0) и последовательность непустых строк A. Строки последовательности содержат только цифры и заглавные буквы латинского алфавита. Из элементов A, предшествующих первому элементу, длина которого превышает L, извлечь строки, оканчивающиеся буквой. Полученную последовательность отсортировать по убыванию длин строк, а строки одинаковой длины — в лексикографическом порядке по возрастанию.

LinqBegin29. Даны целые числа D и K ($K > 0$) и целочисленная последовательность A. Найти теоретико-множественное объединение двух фрагментов A: первый содержит все элементы до первого элемента, большего D (не включая его), а второй — все элементы, начиная с элемента с порядковым номером K. Полученную последовательность (не содержащую одинаковых элементов) отсортировать по убыванию.

LinqBegin30. Дано целое число K (> 0) и целочисленная последовательность A. Найти теоретико-множественную разность двух фрагментов A: первый содержит все четные числа, а второй — все числа с порядковыми номерами, большими K. В полученной последовательности (не содержащей одинаковых элементов) поменять порядок элементов на обратный.

LinqBegin31. Дано целое число K (> 0) и последовательность непустых строк A. Строки последовательности содержат только цифры и заглавные буквы латинского алфавита. Найти теоретико-множественное пересечение двух фрагментов A: первый содержит K начальных элементов, а второй — все элементы, расположенные после последнего элемента, оканчивающегося цифрой. Полученную последовательность (не содержащую одинаковых элементов) отсортировать по возрастанию длин строк, а строки одинаковой длины — в лексикографическом порядке по возрастанию.

Проецирование

Изучаемые методы LINQ:
• Select, SelectMany (проецирование).

LinqBegin32. Дана последовательность непустых строк A. Получить последовательность символов, каждый элемент которой является начальным символом соответствующей строки из A. Порядок символов должен быть обратным по отношению к порядку элементов исходной последовательности.

LinqBegin33. Дана целочисленная последовательность. Обрабатывая только положительные числа, получить последовательность их последних цифр и удалить в полученной последовательности все вхождения одинаковых цифр, кроме первого. Порядок полученных цифр должен соответствовать порядку исходных чисел.

LinqBegin34. Дана последовательность положительных целых чисел. Обрабатывая только нечетные числа, получить последовательность их строковых представлений и отсортировать ее в лексикографическом порядке по возрастанию.

LinqBegin35. Дана целочисленная последовательность. Получить последовательность чисел, каждый элемент которой равен произведению соответствующего элемента исходной последовательности на его порядковый номер (1, 2, …). В полученной последовательности удалить все элементы, не являющиеся двузначными, и поменять порядок оставшихся элементов на обратный.

LinqBegin36. Дана последовательность непустых строк. Получить последовательность символов, которая определяется следующим образом: если соответствующая строка исходной последовательности имеет нечетную длину, то в качестве символа берется первый символ этой строки; в противном случае берется последний символ строки. Отсортировать полученные символы по убыванию их кодов.

LinqBegin37. Дана строковая последовательность *A*. Строки последовательности содержат только заглавные буквы латинского алфавита. Получить новую последовательность строк, элементы которой определяются по соответствующим элементам *A* следующим образом: пустые строки в новую последовательность не включаются, а к непустым приписывается порядковый номер данной строки в исходной последовательности (например, если пятый элемент *A* имеет вид «ABC», то в полученной последовательности он будет иметь вид «ABC5»). При нумерации должны учитываться и пустые строки последовательности *A*. Отсортировать полученную последовательность в лексикографическом порядке по возрастанию.

LinqBegin38. Дана целочисленная последовательность *A*. Получить новую последовательность чисел, элементы которой определяются по соответствующим элементам последовательности *A* следующим образом: если порядковый номер элемента *A* делится на 3 (3, 6, …), то этот элемент в новую последовательность не включается; если остаток от деления порядкового номера на 3 равен 1 (1, 4, …), то в новую последовательность добавляется удвоенное значение этого элемента; в противном случае (для элементов *A* с номерами 2, 5, …) элемент добавляется в новую последовательность без изменений. В полученной последовательности сохранить исходный порядок следования элементов.

LinqBegin39. Дана строковая последовательность *A*. Получить последовательность цифровых символов, входящих в строки последовательности *A* (символы могут повторяться). Порядок символов должен соответствовать порядку строк *A* и порядку следования символов в каждой строке.

Указание. Использовать метод SelectMany с учетом того, что строка может интерпретироваться как последовательность символов.

LinqBegin40. Дано число *K* (> 0) и строковая последовательность *A*. Получить последовательность символов, содержащую символы всех строк из *A*, имеющих длину, большую или равную *K* (символы могут повторяться). В полученной последовательности поменять порядок элементов на обратный.

LinqBegin41. Дано целое число *K* (> 0) и строковая последовательность *A*. Каждый элемент последовательности представляет собой несколько слов из заглавных латинских букв, разделенных символами «.» (точка). Получить последовательность строк, содержащую все слова длины *K* из элементов *A* в лексикографическом порядке по возрастанию (слова могут повторяться).

LinqBegin42. Дана последовательность непустых строк. Получить последовательность символов, которая определяется следующим образом: для строк с нечетными порядковыми номерами (1, 3, …) в последовательность символов включаются все прописные латинские буквы, содержащиеся в этих строках, а для строк с четными номерами (2, 4, …) — все их строчные латинские буквы. В полученной последовательности символов сохранить их исходный порядок следования.

LinqBegin43. Дано целое число *K* (> 0) и последовательность непустых строк *A*. Получить последовательность символов, которая определяется следующим образом: для первых *K* элементов последовательности *A* в новую последовательность заносятся символы, стоящие на нечетных позициях данной строки (1, 3, …), а для остальных элементов *A* — символы на четных позициях (2, 4, …). В полученной последовательности поменять порядок элементов на обратный.

Объединение и группировка

Изучаемые методы LINQ:
- Concat (сцепление);
- Join, GroupJoin (объединение);
- DefaultIfEmpty (замена пустой последовательности на одноэлементную);
- GroupBy (группировка).

LinqBegin44. Даны целые числа K_1 и K_2 и целочисленные последовательности *A* и *B*. Получить последовательность, содержащую все числа из *A*, большие K_1, и все числа из *B*, меньшие K_2. Отсортировать полученную последовательность по возрастанию.

LinqBegin45. Даны целые положительные числа L_1 и L_2 и строковые последовательности *A* и *B*. Строки последовательностей содержат только цифры и заглавные буквы латинского алфавита. Получить последовательность, содержащую все строки из *A* длины L_1 и все строки из *B* длины L_2. Отсортировать полученную последовательность в лексикографическом порядке по убыванию.

LinqBegin46. Даны последовательности положительных целых чисел *A* и *B*; все числа в каждой последовательности различны. Найти последовательность всех пар чисел, удовлетворяющих следующим условиям: первый элемент пары принадлежит последовательности *A*, второй принадлежит *B*, и оба элемента оканчиваются одной и той же цифрой. Результирующая последовательность называется *внутренним объединением* последовательностей *A* и *B* по *ключу*, определяемому последними цифрами исходных чисел. Представить найденное объединение в виде последовательности строк, содержащих первый и второй элементы пары, разделенные дефисом, например, «49-129». Порядок следования пар должен определяться исходным порядком элементов последовательности *A*, а для равных первых элементов — порядком элементов последовательности *B*.

LinqBegin47. Даны последовательности положительных целых чисел *A* и *B*; все числа в каждой последовательности различны. Найти внутреннее объединение *A* и *B* (см. LinqBegin46), пары в котором должны удовлетворять следующему условию: последняя цифра первого элемента пары (из *A*) должна совпадать с первой цифрой второго элемента пары (из *B*). Представить найденное объединение в виде последовательности строк, содержащих первый и второй элементы пары, разделенные двоеточием, например, «49:921». Порядок следования пар должен определяться исходным порядком элементов последовательности *A*, а для равных первых элементов пар — лексикографическим порядком строковых представлений вторых элементов (по возрастанию).

LinqBegin48. Даны строковые последовательности *A* и *B*; все строки в каждой последовательности различны, имеют ненулевую длину и содержат только цифры и заглавные буквы латинского алфавита. Найти внутреннее объединение *A* и *B* (см. LinqBegin46), каждая пара которого должна содержать строки одинаковой длины. Представить найденное объединение в виде последовательности строк, содержащих первый и второй элементы пары, разделенные двоеточием, например, «AB:CD». Порядок следования пар должен определяться лексикографическим порядком первых элементов пар (по возрастанию), а для равных первых элементов — лексикографическим порядком вторых элементов пар (по убыванию).

LinqBegin49. Даны строковые последовательности *A*, *B* и *C*; все строки в каждой последовательности различны, имеют

ненулевую длину и содержат только цифры и заглавные буквы латинского алфавита. Найти внутреннее объединение *A, B* и *C* (см. LinqBegin46), каждая тройка которого должна содержать строки, начинающиеся с одного и того же символа. Представить найденное объединение в виде последовательности строк вида «E_A=E_B=E_C», где E_A, E_B, E_C — элементы из *A, B, C* соответственно. Для различных элементов E_A сохраняется исходный порядок их следования, для равных элементов E_A порядок троек определяется лексикографическим порядком элементов E_B (по возрастанию), а для равных элементов E_A и E_B — лексикографическим порядком элементов E_C (по убыванию).

LinqBegin50. Даны строковые последовательности *A* и *B*; все строки в каждой последовательности различны и имеют ненулевую длину. Получить последовательность строк вида «*E*:*N*», где *E* обозначает один из элементов последовательности *A*, а *N* — количество элементов из *B*, начинающихся с того же символа, что и элемент *E* (например, «abc:4»); количество *N* может быть равно 0. Порядок элементов полученной последовательности должен определяться исходным порядком элементов последовательности *A*.

Указание. Использовать метод GroupJoin.

LinqBegin51. Даны последовательности положительных целых чисел *A* и *B*; все числа в последовательности *A* различны. Получить последовательность строк вида «*S*:*E*», где *S* обозначает сумму тех чисел из *B*, которые оканчиваются на ту же цифру, что и число *E* — один из элементов последовательности *A* (например, «74:23»); если для числа *E* не найдено ни одного подходящего числа из последовательности *B*, то в качестве *S* указать 0. Расположить элементы полученной последовательности по возрастанию значений найденных сумм, а при равных суммах — по убыванию значений элементов *A*.

LinqBegin52. Даны строковые последовательности *A* и *B*; все строки в каждой последовательности различны, имеют ненулевую длину и содержат только цифры и заглавные буквы латинского алфавита. Получить последовательность всевозможных комбинаций вида «E_A=E_B», где E_A — некоторый элемент из *A*, E_B — некоторый элемент из *B*, причем оба элемента оканчиваются цифрой (например, «AF3=D78»). Упорядочить полученную последовательность в лексикографическом порядке по возрастанию элементов E_A, а при одинаковых элементах E_A — в лексикографическом порядке по убыванию элементов E_B.

Указание. Для перебора комбинаций использовать методы SelectMany и Select.

LinqBegin53. Даны целочисленные последовательности *A* и *B*. Получить последовательность всех *различных* сумм, в которых первое слагаемое берется из *A*, а второе из *B*. Упорядочить полученную последовательность по возрастанию.

LinqBegin54. Даны строковые последовательности *A* и *B*; все строки в каждой последовательности различны, имеют ненулевую длину и содержат только цифры и заглавные буквы латинского алфавита. Найти последовательность всех пар строк, удовлетворяющих следующим условиям: первый элемент пары принадлежит последовательности *A*, а второй либо является одним из элементов последовательности *B*, начинающихся с того же символа, что и первый элемент пары, либо является пустой строкой (если *B* не содержит ни одной подходящей строки). Результирующая последовательность называется *левым внешним объединением* последовательностей *A* и *B* по *ключу*, определяемому первыми символами исходных строк. Представить найденное объединение в виде последовательности строк вида «E_A.E_B», где E_A — элемент из *A*, а E_B — либо один из соответствующих ему элементов из *B*, либо пустая строка. Расположить элементы полученной строковой последовательности в лексикографическом порядке по возрастанию.

Указание. Использовать методы GroupJoin, DefaultIfEmpty, Select и SelectMany.

LinqBegin55. Даны последовательности положительных целых чисел *A* и *B*; все числа в каждой последовательности различны. Найти левое внешнее объединение *A* и *B* (см. LinqBegin54), пары в котором должны удовлетворять следующему условию: оба элемента пары оканчиваются одной и той же цифрой. Представить найденное объединение в виде последовательности строк вида «E_A:E_B», где E_A — число из *A*, а E_B — либо одно из соответствующих ему чисел из *B*, либо 0 (если в *B* не содержится чисел, соответствующих E_A). Расположить элементы полученной последовательности по убыванию чисел E_A, а при одинаковых числах E_A — по возрастанию чисел E_B.

LinqBegin56. Дана целочисленная последовательность *A*. *Сгруппировать* элементы последовательности *A*, оканчивающиеся одной и той же цифрой, и на основе этой группировки получить последовательность строк вида «*D*:*S*», где *D* — *ключ группировки* (т. е. некоторая цифра, которой оканчивается хотя бы одно из чисел последовательности *A*), а *S* — сумма всех чисел из *A*, которые оканчиваются цифрой *D*. Полученную последовательность упорядочить по возрастанию ключей.

Указание. Использовать метод GroupBy.

LinqBegin57. Дана целочисленная последовательность. Среди всех элементов последовательности, оканчивающихся одной и той же цифрой, выбрать максимальный. Полученную последовательность максимальных элементов упорядочить по возрастанию их последних цифр.

LinqBegin58. Дана последовательность непустых строк. Среди всех строк, начинающихся с одного и того же символа, выбрать наиболее длинную. Если таких строк несколько, то выбрать первую по порядку их следования в исходной последовательности. Полученную последовательность строк упорядочить по возрастанию кодов их начальных символов.

LinqBegin59. Дана последовательность непустых строк, содержащих только заглавные буквы латинского алфавита. Среди всех строк одинаковой длины выбрать первую в лексикографическом порядке (по возрастанию). Полученную последовательность строк упорядочить по убыванию их длин.

LinqBegin60. Дана последовательность непустых строк *A*, содержащих только заглавные буквы латинского алфавита. Для всех строк, начинающихся с одной и той же буквы, определить их суммарную длину и получить последовательность строк вида «*S*-*C*», где *S* — суммарная длина всех строк из *A*, которые начинаются с буквы *C*. Полученную последовательность упорядочить по убыванию числовых значений сумм, а при равных значениях сумм — по возрастанию кодов символов *C*.

Технология LINQ to Objects

В каждом задании даются имена одного или нескольких текстовых файлов, содержащих исходные последовательности, а также имя текстового файла, в который требуется записать результаты обработки исходных последовательностей (имя результирующего файла указывается последним). Каждая исходная последовательность содержится в отдельном файле. Все исходные файлы содержат текст в кодировке «windows-1251»; эта же кодировка должна использоваться при записи полученных данных в результирующий файл.

Каждый элемент последовательности размещается в отдельной строке файла, в начале и конце строки пробелы отсутствуют, поля элемента не содержат пробелов и разделяются ровно

одним пробелом. Все исходные числовые данные являются положительными. В качестве десятичного разделителя используется *точка*.

Если в задание входят дополнительные числовые или строковые исходные данные, то они указываются в начале набора исходных данных (перед именами файлов).

Обработка отдельных последовательностей

LinqObj1. Исходная последовательность содержит сведения о клиентах фитнес-центра. Каждый элемент последовательности включает следующие целочисленные поля:

```
<Код клиента> <Год> <Номер месяца>
<Продолжительность занятий (в часах)>
```

Найти элемент последовательности с минимальной продолжительностью занятий. Вывести эту продолжительность, а также соответствующие ей год и номер месяца (в указанном порядке на той же строке). Если имеется несколько элементов с минимальной продолжительностью, то вывести данные того из них, который является последним в исходной последовательности.

LinqObj2. Исходная последовательность содержит сведения о клиентах фитнес-центра. Каждый элемент последовательности включает следующие целочисленные поля:

```
<Номер месяца> <Год> <Код клиента>
<Продолжительность занятий (в часах)>
```

Найти элемент последовательности с максимальной продолжительностью занятий. Вывести эту продолжительность, а также соответствующие ей год и номер месяца (в указанном порядке на той же строке). Если имеется несколько элементов с максимальной продолжительностью, то вывести данные, соответствующие самой поздней дате.

LinqObj3. Исходная последовательность содержит сведения о клиентах фитнес-центра. Каждый элемент последовательности включает следующие целочисленные поля:

```
<Год> <Номер месяца> <Продолжительность
занятий (в часах)> <Код клиента>
```

Определить год, в котором суммарная продолжительность занятий всех клиентов была наибольшей, и вывести этот год и наибольшую суммарную продолжительность. Если таких годов было несколько, то вывести наименьший из них.

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.

LinqObj5. Исходная последовательность содержит сведения о клиентах фитнес-центра. Каждый элемент последовательности включает следующие целочисленные поля:

```
<Код клиента> <Продолжительность занятий (в
часах)> <Год> <Номер месяца>
```

Для каждого клиента, присутствующего в исходных данных, определить общее количество месяцев, в течение которых он посещал занятия (вначале выводить количество месяцев, затем код клиента). Сведения о каждом клиенте выводить на новой строке и упорядочивать по возрастанию количества месяцев, а при их равенстве — по возрастанию кода клиента.

LinqObj6. Исходная последовательность содержит сведения о клиентах фитнес-центра. Каждый элемент последовательности включает следующие целочисленные поля:

```
<Код клиента> <Продолжительность занятий (в
часах)> <Год> <Номер месяца>
```

Для каждого месяца определить суммарную продолжительность занятий всех клиентов за все годы (вначале выводить суммарную продолжительность, затем номер месяца). Если данные о некотором месяце отсутствуют, то для этого месяца вывести 0. Сведения о каждом месяце выводить на новой строке и упорядочивать по убыванию суммарной продолжительности, а при равной продолжительности — по возрастанию номера месяца.

LinqObj7. Дано целое число *K* — код одного из клиентов фитнес-центра. Исходная последовательность содержит сведения о клиентах этого фитнес-центра. Каждый элемент последовательности включает следующие целочисленные поля:

```
<Продолжительность занятий (в часах)> <Год>
<Номер месяца> <Код клиента>
```

Для каждого года, в котором клиент с кодом *K* посещал центр, определить месяц, в котором продолжительность занятий данного клиента была наибольшей для данного года (если таких месяцев несколько, то выбирать месяц с наименьшим номером). Сведения о каждом годе выводить на новой строке в следующем порядке: год, номер месяца, продолжительность занятий в этом месяце. Упорядочивать сведения по убыванию номера года. Если данные о клиенте с кодом *K* отсутствуют, то записать в результирующий файл строку «Нет данных».

LinqObj8. Дано целое число *K* — код одного из клиентов фитнес-центра. Исходная последовательность содержит сведения о клиентах этого фитнес-центра. Каждый элемент последовательности включает следующие целочисленные поля:

```
<Продолжительность занятий (в часах)> <Код
клиента> <Год> <Номер месяца>
```

Для каждого года, в котором клиент с кодом *K* посещал центр, определить месяц, в котором продолжительность занятий данного клиента была наименьшей для данного года (если таких месяцев несколько, то выбирать первый из этих месяцев в исходном наборе; месяцы с нулевой продолжительностью занятий не учитывать). Сведения о каждом годе выводить на новой строке в следующем порядке: наименьшая продолжительность занятий, год, номер месяца. Упорядочивать сведения по возрастанию продолжительности занятий, а при равной продолжительности — по возрастанию номера года. Если данные о клиенте с кодом *K* отсутствуют, то записать в результирующий файл строку «Нет данных».

LinqObj9. Дано целое число *K* — код одного из клиентов фитнес-центра. Исходная последовательность содержит сведения о клиентах этого фитнес-центра. Каждый элемент последовательности включает следующие целочисленные поля:

```
<Код клиента> <Продолжительность занятий (в
часах)> <Номер месяца> <Год>
```

Для каждого года, в котором клиент с кодом *K* посещал центр, определить число месяцев, для которых продолжительность занятий данного клиента превосходила 15 часов (вначале выводить число месяцев, затем год). Если для некоторого года требуемые месяцы отсутствуют, то вывести для него 0. Сведения о каждом годе выводить на новой строке; данные упорядочивать по убыванию числа месяцев, а при равном числе месяцев — по возрастанию номера года. Если данные об указанном клиенте отсутствуют, то записать в результирующий файл строку «Нет данных».

LinqObj10. Исходная последовательность содержит сведения о клиентах фитнес-центра. Каждый элемент последовательности включает следующие целочисленные поля:

`<Год> <Номер месяца> <Код клиента> <Продолжительность занятий (в часах)>`

Для каждой пары «год–месяц», присутствующей в исходных данных, определить количество клиентов, которые посещали центр в указанное время (вначале выводится год, затем месяц, затем количество клиентов). Сведения о каждой паре «год–месяц» выводить на новой строке и упорядочивать по убыванию номера года, а для одинакового номера года — по возрастанию номера месяца.

LinqObj11. Исходная последовательность содержит сведения о клиентах фитнес-центра. Каждый элемент последовательности включает следующие целочисленные поля:

`<Код клиента> <Год> <Номер месяца> <Продолжительность занятий (в часах)>`

Для каждой пары «год–месяц», присутствующей в исходных данных, определить общую продолжительность занятий всех клиентов в указанное время (вначале выводится общая продолжительность, затем год, затем месяц). Сведения о каждой паре «год–месяц» выводить на новой строке и упорядочивать по возрастанию общей продолжительности занятий, для одинаковой продолжительности — по убыванию номера года, а для одинакового номера года — по возрастанию номера месяца.

LinqObj12. Дано целое число P ($10 < P < 50$). Исходная последовательность содержит сведения о клиентах фитнес-центра. Каждый элемент последовательности включает следующие целочисленные поля:

`<Продолжительность занятий (в часах)> <Код клиента> <Номер месяца> <Год>`

Для каждого года, присутствующего в исходных данных, определить количество месяцев, в которых суммарная длительность занятий всех клиентов составляла более P процентов от суммарной длительности за этот год (вначале выводить количество месяцев, затем год). Если в некотором году ни для одного месяца не выполнялось требуемое условие, то вывести для него 0. Сведения о каждом годе выводить на новой строке и упорядочивать по убыванию количества месяцев, а для одинакового количества — по возрастанию номера года.

LinqObj13. Исходная последовательность содержит сведения об абитуриентах. Каждый элемент последовательности включает следующие поля:

`<Номер школы> <Год поступления> <Фамилия>`

Для каждого года, присутствующего в исходных данных, найти школу с наибольшим номером среди школ, которые окончили абитуриенты, поступившие в этом году, и вывести год и найденный номер школы. Сведения о каждом годе выводить на новой строке и упорядочивать по возрастанию номера года.

LinqObj14. Исходная последовательность содержит сведения об абитуриентах. Каждый элемент последовательности включает следующие поля:

`<Год поступления> <Номер школы> <Фамилия>`

Определить, в какие годы общее число абитуриентов для всех школ было наибольшим, и вывести это число, а также количество таких лет. Каждое число выводить на новой строке.

LinqObj15. Исходная последовательность содержит сведения об абитуриентах. Каждый элемент последовательности включает следующие поля:

`<Номер школы> <Год поступления> <Фамилия>`

Определить, в какие годы общее число абитуриентов для всех школ было наибольшим, и вывести это число, а также годы, в которые оно было достигнуто (годы упорядочивать по возрастанию, каждое число выводить на новой строке).

LinqObj16. Исходная последовательность содержит сведения об абитуриентах. Каждый элемент последовательности включает следующие поля:

`<Год поступления> <Номер школы> <Фамилия>`

Для каждого года, присутствующего в исходных данных, вывести общее число абитуриентов, поступивших в этом году (вначале указывать число абитуриентов, затем год). Сведения о каждом годе выводить на новой строке и упорядочивать по убыванию числа поступивших, а для совпадающих чисел — по возрастанию номера года.

LinqObj17. Исходная последовательность содержит сведения об абитуриентах. Каждый элемент последовательности включает следующие поля:

`<Номер школы> <Год поступления> <Фамилия>`

Для каждого года, присутствующего в исходных данных, вывести число различных школ, которые окончили абитуриенты, поступившие в этом году (вначале указывать число школ, затем год). Сведения о каждом годе выводить на новой строке и упорядочивать по возрастанию числа школ, а для совпадающих чисел — по возрастанию номера года.

LinqObj18. Исходная последовательность содержит сведения об абитуриентах. Каждый элемент последовательности включает следующие поля:

`<Год поступления> <Номер школы> <Фамилия>`

Найти годы, для которых число абитуриентов было не меньше среднего значения по всем годам (вначале указывать число абитуриентов для данного года, затем год). Сведения о каждом годе выводить на новой строке и упорядочивать по убыванию числа абитуриентов, а для совпадающих чисел — по возрастанию номера года.

LinqObj19. Исходная последовательность содержит сведения об абитуриентах. Каждый элемент последовательности включает следующие поля:

`<Фамилия> <Год поступления> <Номер школы>`

Для каждой школы вывести общее число абитуриентов за все годы и фамилию первого из абитуриентов этой школы, содержащихся в исходном наборе данных (вначале указывать номер школы, затем число абитуриентов, затем фамилию). Сведения о каждой школе выводить на новой строке и упорядочивать по возрастанию номеров школ.

LinqObj20. Исходная последовательность содержит сведения об абитуриентах. Каждый элемент последовательности включает следующие поля:

`<Фамилия> <Номер школы> <Год поступления>`

Определить, для каких школ общее число абитуриентов за все годы было наибольшим, и вывести данные об абитуриентах из этих школ (вначале указывать номер школы, затем фамилию абитуриента). Сведения о каждом абитуриенте выводить на новой строке и упорядочивать по возрастанию номеров школ, а для одинаковых номеров — в порядке следования абитуриентов в исходном наборе данных.

LinqObj21. Исходная последовательность содержит сведения об абитуриентах. Каждый элемент последовательности включает следующие поля:

`<Фамилия> <Год поступления> <Номер школы>`

Определить, для каких школ общее число абитуриентов за все годы было наибольшим, и вывести для каждой из этих школ данные о первом абитуриенте в алфавитном порядке (вначале указывать фамилию абитуриента, затем номер шко-

лы). Сведения о каждом абитуриенте выводить на новой строке и упорядочивать в алфавитном порядке, а для одинаковых фамилий — по возрастанию номеров школ.

LinqObj22. Исходная последовательность содержит сведения об абитуриентах. Каждый элемент последовательности включает следующие поля:

`<Фамилия> <Номер школы> <Год поступления>`

Для каждой школы найти годы поступления абитуриентов из этой школы и вывести номер школы и найденные для нее годы (годы располагаются на той же строке, что и номер школы, и упорядочиваются по возрастанию). Сведения о каждой школе выводить на новой строке и упорядочивать по возрастанию номеров школ.

LinqObj23. Исходная последовательность содержит сведения об абитуриентах. Каждый элемент последовательности включает следующие поля:

`<Фамилия> <Год поступления> <Номер школы>`

Для каждой пары «год–школа», присутствующей в исходных данных, найти число абитуриентов, относящихся к этому году и школе, и вывести год, номер школы и найденное число абитуриентов. Сведения о каждой паре «год–школа» выводить на новой строке и упорядочивать по убыванию года, а для совпадающих годов — по возрастанию номера школы.

LinqObj24. Исходная последовательность содержит сведения об абитуриентах. Каждый элемент последовательности включает следующие поля:

`<Фамилия> <Номер школы> <Год поступления>`

Для каждой пары «школа–год», присутствующей в исходных данных, найти трех первых абитуриентов, относящихся к этой школе и году, и вывести номер школы, год и найденные фамилии (в порядке их следования в исходном наборе данных). Если для некоторой пары «школа–год» имеется менее трех абитуриентов, то вывести информацию обо всех абитуриентах, относящихся к этой паре. Сведения о каждой паре «школа–год» выводить на новой строке и упорядочивать по возрастанию номера школы, а для совпадающих номеров — по убыванию года.

LinqObj25. Исходная последовательность содержит сведения о задолжниках по оплате коммунальных услуг, живущих в 144-квартирном 9-этажном доме. Каждый элемент последовательности включает следующие поля:

`<Фамилия> <Задолженность> <Номер квартиры>`

Задолженность указывается в виде дробного числа (целая часть — рубли, дробная часть — копейки). В каждом подъезде на каждом этаже располагаются по 4 квартиры. Найти номер подъезда, жильцы которого имеют наибольшую суммарную задолженность, и вывести этот номер вместе с размером суммарной задолженности (выводится с двумя дробными знаками). Считать, что суммарные задолженности для всех подъездов имеют различные значения.

LinqObj26. Исходная последовательность содержит сведения о задолжниках по оплате коммунальных услуг, живущих в 144-квартирном 9-этажном доме. Каждый элемент последовательности включает следующие поля:

`<Номер квартиры> <Фамилия> <Задолженность>`

Задолженность указывается в виде дробного числа (целая часть — рубли, дробная часть — копейки). В каждом подъезде на каждом этаже располагаются по 4 квартиры. Для каждого из 4 подъездов дома вывести сведения о задолжниках, живущих в этом подъезде: номер подъезда, число задолжников, средняя задолженность для жильцов этого подъезда (выводится с двумя дробными знаками). Жильцы, не имеющие долга, при вычислении средней задолженности не учитываются. Сведения о каждом подъезде выводить на отдельной строке и упорядочивать по возрастанию номера подъезда. Если в каком-либо подъезде задолжники отсутствуют, то данные об этом подъезде не выводить.

LinqObj27. Исходная последовательность содержит сведения о задолжниках по оплате коммунальных услуг, живущих в 144-квартирном 9-этажном доме. Каждый элемент последовательности включает следующие поля:

`<Фамилия> <Номер квартиры> <Задолженность>`

Задолженность указывается в виде дробного числа (целая часть — рубли, дробная часть — копейки). В каждом подъезде на каждом этаже располагаются по 4 квартиры. Для каждого из 9 этажей дома вывести сведения о задолжниках, живущих на этом этаже: число задолжников, номер этажа, суммарная задолженность для жильцов этого этажа (выводится с двумя дробными знаками). Сведения о каждом этаже выводить на отдельной строке и упорядочивать по возрастанию числа задолжников, а для совпадающих чисел — по возрастанию этажа. Если на каком-либо этаже задолжники отсутствуют, то данные об этом этаже не выводить.

LinqObj28. Исходная последовательность содержит сведения о задолжниках по оплате коммунальных услуг, живущих в 144-квартирном 9-этажном доме. Каждый элемент последовательности включает следующие поля:

`<Задолженность> <Фамилия> <Номер квартиры>`

Задолженность указывается в виде дробного числа (целая часть — рубли, дробная часть — копейки). В каждом подъезде на каждом этаже располагаются по 4 квартиры. Для каждого из 9 этажей дома вывести сведения о задолжниках, живущих на этом этаже: номер этажа, суммарная задолженность для жильцов этого этажа (выводится с двумя дробными знаками), число задолжников. Сведения о каждом этаже выводить на отдельной строке и упорядочивать по убыванию номера этажа. Если на каком-либо этаже задолжники отсутствуют, то вывести для этого этажа нулевые данные.

LinqObj29. Исходная последовательность содержит сведения о задолжниках по оплате коммунальных услуг, живущих в 144-квартирном 9-этажном доме. Каждый элемент последовательности включает следующие поля:

`<Номер квартиры> <Фамилия> <Задолженность>`

Задолженность указывается в виде дробного числа (целая часть — рубли, дробная часть — копейки). В каждом подъезде на каждом этаже располагаются по 4 квартиры. Для каждого из 4 подъездов дома найти жильца с наибольшей задолженностью и вывести сведения о нем: номер подъезда, номер квартиры, фамилия жильца, задолженность (выводится с двумя дробными знаками). Считать, что в наборе исходных данных все задолженности имеют различные значения. Сведения о каждом задолжнике выводить на отдельной строке и упорядочивать по возрастанию номера подъезда. Если в каком-либо подъезде задолжники отсутствуют, то данные об этом подъезде не выводить.

LinqObj30. Исходная последовательность содержит сведения о задолжниках по оплате коммунальных услуг, живущих в 144-квартирном 9-этажном доме. Каждый элемент последовательности включает следующие поля:

`<Фамилия> <Задолженность> <Номер квартиры>`

Задолженность указывается в виде дробного числа (целая часть — рубли, дробная часть — копейки). В каждом подъезде на каждом этаже располагаются по 4 квартиры. Для каждого из 9 этажей дома найти жильца с наименьшей задолженностью и вывести сведения о нем: номер квартиры, номер этажа, фамилия жильца, задолженность (выводится с двумя дробными знаками). Считать, что в наборе исходных данных все задолженности имеют различные значения. Сведения о каждом задолжнике выводить на отдельной строке и

упорядочивать по возрастанию номера квартиры. Если на каком-либо этаже задолжники отсутствуют, то данные об этом этаже не выводить.

LinqObj31. Исходная последовательность содержит сведения о задолжниках по оплате коммунальных услуг, живущих в 144-квартирном 9-этажном доме. Каждый элемент последовательности включает следующие поля:

```
<Задолженность> <Фамилия> <Номер квартиры>
```

Задолженность указывается в виде дробного числа (целая часть — рубли, дробная часть — копейки). В каждом подъезде на каждом этаже располагаются по 4 квартиры. Для каждого из 4 подъездов дома найти трех жильцов с наибольшей задолженностью и вывести сведения о них: задолженность (выводится с двумя дробными знаками), номер подъезда, номер квартиры, фамилия жильца. Считать, что в наборе исходных данных все задолженности имеют различные значения. Сведения о каждом задолжнике выводить на отдельной строке и упорядочивать по убыванию размера задолженности (номер подъезда при сортировке не учитывать). Если в каком-либо подъезде число задолжников меньше трех, то включить в полученный набор всех задолжников этого подъезда.

LinqObj32. Исходная последовательность содержит сведения о задолжниках по оплате коммунальных услуг, живущих в 144-квартирном 9-этажном доме. Каждый элемент последовательности включает следующие поля:

```
<Фамилия> <Номер квартиры> <Задолженность>
```

Задолженность указывается в виде дробного числа (целая часть — рубли, дробная часть — копейки). В каждом подъезде на каждом этаже располагаются по 4 квартиры. Для каждого из 9 этажей дома найти жильца с наименьшей задолженностью и вывести сведения о нем: номер этажа и задолженность (выводится с двумя дробными знаками). Считать, что в наборе исходных данных все задолженности имеют различные значения. Сведения о каждом этаже выводить на отдельной строке и упорядочивать по возрастанию номера этажа. Если на каком-либо этаже задолжники отсутствуют, то для этого этажа вывести задолженность, равную 0.00.

LinqObj33. Исходная последовательность содержит сведения о задолжниках по оплате коммунальных услуг, живущих в 144-квартирном 9-этажном доме. Каждый элемент последовательности включает следующие поля:

```
<Номер квартиры> <Задолженность> <Фамилия>
```

Задолженность указывается в виде дробного числа (целая часть — рубли, дробная часть — копейки). В каждом подъезде на каждом этаже располагаются по 4 квартиры. Найти жильцов, долг которых не меньше величины средней задолженности по дому, и вывести сведения о них: номер квартиры, фамилия, задолженность (выводится с двумя дробными знаками). Жильцы, не имеющие долга, при вычислении средней задолженности не учитываются. Сведения о каждом задолжнике выводить на отдельной строке и упорядочивать по возрастанию номеров квартир.

LinqObj34. Исходная последовательность содержит сведения о задолжниках по оплате коммунальных услуг, живущих в 144-квартирном 9-этажном доме. Каждый элемент последовательности включает следующие поля:

```
<Задолженность> <Номер квартиры> <Фамилия>
```

Задолженность указывается в виде дробного числа (целая часть — рубли, дробная часть — копейки). В каждом подъезде на каждом этаже располагаются по 4 квартиры. Найти жильцов, долг которых не больше величины средней задолженности по дому, и вывести сведения о них: номер этажа, номер квартиры, фамилия, задолженность (выводится с двумя дробными знаками). Жильцы, не имеющие долга, при вычислении средней задолженности не учитываются. Сведения о каждом задолжнике выводить на отдельной строке и упорядочивать по убыванию номеров этажей, а для одинаковых этажей — по возрастанию номеров квартир.

LinqObj35. Исходная последовательность содержит сведения о задолжниках по оплате коммунальных услуг, живущих в 144-квартирном 9-этажном доме. Каждый элемент последовательности включает следующие поля:

```
<Задолженность> <Фамилия> <Номер квартиры>
```

Задолженность указывается в виде дробного числа (целая часть — рубли, дробная часть — копейки). В каждом подъезде на каждом этаже располагаются по 4 квартиры. Для каждого из 4 подъездов дома найти задолжников, долг которых не меньше величины средней задолженности по данному подъезду, и вывести сведения о них: номер подъезда, задолженность (выводится с двумя дробными знаками), фамилия, номер квартиры. Жильцы, не имеющие долга, при вычислении средней задолженности не учитываются. Сведения о каждом задолжнике выводить на отдельной строке и упорядочивать по возрастанию номеров подъездов, а для одинаковых подъездов — по убыванию размера задолженности. Считать, что в наборе исходных данных все задолженности имеют различные значения.

LinqObj36. Исходная последовательность содержит сведения о задолжниках по оплате коммунальных услуг, живущих в 144-квартирном 9-этажном доме. Каждый элемент последовательности включает следующие поля:

```
<Номер квартиры> <Фамилия> <Задолженность>
```

Задолженность указывается в виде дробного числа (целая часть — рубли, дробная часть — копейки). В каждом подъезде на каждом этаже располагаются по 4 квартиры. Для каждого из 9 этажей дома найти задолжников, долг которых не больше величины средней задолженности по данному этажу, и вывести сведения о них: номер этажа, задолженность (выводится с двумя дробными знаками), фамилия, номер квартиры. Жильцы, не имеющие долга, при вычислении средней задолженности не учитываются. Сведения о каждом задолжнике выводить на отдельной строке и упорядочивать по возрастанию номеров этажей, а для одинаковых этажей — по возрастанию размера задолженности. Считать, что в наборе исходных данных все задолженности имеют различные значения.

LinqObj37. Исходная последовательность содержит сведения об автозаправочных станциях (АЗС). Каждый элемент последовательности включает следующие поля:

```
<Компания> <Марка бензина> <Цена 1 литра (в копейках)> <Улица>
```

Названия компаний и улиц не содержат пробелов. В качестве марки бензина указываются числа 92, 95 или 98. Каждая компания имеет не более одной АЗС на каждой улице; цены на разных АЗС одной и той же компании могут различаться. Для каждой марки бензина, присутствующей в исходных данных, определить минимальную и максимальную цену литра бензина этой марки (вначале выводить марку, затем цены в указанном порядке). Сведения о каждой марке выводить на новой строке и упорядочивать по убыванию значения марки.

LinqObj38. Исходная последовательность содержит сведения об автозаправочных станциях (АЗС). Каждый элемент последовательности включает следующие поля:

```
<Цена 1 литра (в копейках)> <Марка бензина> <Компания> <Улица>
```

Названия компаний и улиц не содержат пробелов. В качестве марки бензина указываются числа 92, 95 или 98. Каждая компания имеет не более одной АЗС на каждой улице; цены

на разных АЗС одной и той же компании могут различаться. Для каждой марки бензина, присутствующей в исходных данных, определить количество станций, предлагавших эту марку (вначале выводить количество станций, затем номер марки). Сведения о каждой марке выводить на новой строке и упорядочивать по возрастанию количества станций, а для одинакового количества — по возрастанию значения марки.

LinqObj39. Дано целое число *M* — значение одной из марок бензина. Исходная последовательность содержит сведения об автозаправочных станциях (АЗС). Каждый элемент последовательности включает следующие поля:

```
<Улица> <Компания> <Марка бензина> <Цена 1 литра (в копейках)>
```

Названия компаний и улиц не содержат пробелов. В качестве марки бензина указываются числа 92, 95 или 98. Каждая компания имеет не более одной АЗС на каждой улице; цены на разных АЗС одной и той же компании могут различаться. Для каждой улицы определить количество АЗС, предлагавших марку бензина *M* (вначале выводить количество АЗС на данной улице, затем название улицы; количество АЗС может быть равно 0). Сведения о каждой улице выводить на новой строке и упорядочивать по возрастанию количества АЗС, а для одинакового количества — по названиям улиц в алфавитном порядке.

LinqObj40. Исходная последовательность содержит сведения об автозаправочных станциях (АЗС). Каждый элемент последовательности включает следующие поля:

```
<Компания> <Улица> <Марка бензина> <Цена 1 литра (в копейках)>
```

Названия компаний и улиц не содержат пробелов. В качестве марки бензина указываются числа 92, 95 или 98. Каждая компания имеет не более одной АЗС на каждой улице; цены на разных АЗС одной и той же компании могут различаться. Для каждой улицы определить количество АЗС, предлагавших определенную марку бензина (вначале выводить название улицы, затем три числа — количество АЗС для бензина марки 92, 95 и 98; некоторые из этих чисел могут быть равны 0). Сведения о каждой улице выводить на новой строке и упорядочивать по названиям улиц в алфавитном порядке.

LinqObj41. Дано целое число *M* — значение одной из марок бензина. Исходная последовательность содержит сведения об автозаправочных станциях (АЗС). Каждый элемент последовательности включает следующие поля:

```
<Марка бензина> <Улица> <Компания> <Цена 1 литра (в копейках)>
```

Названия компаний и улиц не содержат пробелов. В качестве марки бензина указываются числа 92, 95 или 98. Каждая компания имеет не более одной АЗС на каждой улице; цены на разных АЗС одной и той же компании могут различаться. Для каждой улицы, на которой имеются АЗС с бензином марки *M*, определить максимальную цену бензина этой марки (вначале выводить максимальную цену, затем название улицы). Сведения о каждой улице выводить на новой строке и упорядочивать по возрастанию максимальной цены, а для одинаковой цены — по названиям улиц в алфавитном порядке. Если ни одной АЗС с бензином марки *M* не найдено, то записать в результирующий файл строку «Нет».

LinqObj42. Исходная последовательность содержит сведения об автозаправочных станциях (АЗС). Каждый элемент последовательности включает следующие поля:

```
<Марка бензина> <Компания> <Улица> <Цена 1 литра (в копейках)>
```

Названия компаний и улиц не содержат пробелов. В качестве марки бензина указываются числа 92, 95 или 98. Каждая компания имеет не более одной АЗС на каждой улице; цены на разных АЗС одной и той же компании могут различаться. Для каждой улицы определить минимальную цену бензина каждой марки (вначале выводить название улицы, затем три числа — минимальную цену для бензина марки 92, 95 и 98). При отсутствии бензина нужной марки выводить число 0. Сведения о каждой улице выводить на новой строке и упорядочивать по названиям улиц в алфавитном порядке.

LinqObj43. Дано целое число *M* — значение одной из марок бензина. Исходная последовательность содержит сведения об автозаправочных станциях (АЗС). Каждый элемент последовательности включает следующие поля:

```
<Цена 1 литра (в копейках)> <Марка бензина> <Улица> <Компания>
```

Названия компаний и улиц не содержат пробелов. В качестве марки бензина указываются числа 92, 95 или 98. Каждая компания имеет не более одной АЗС на каждой улице; цены на разных АЗС одной и той же компании могут различаться. Для каждой компании определить разброс цен на бензин указанной марки *M* (вначале выводить разность максимальной и минимальной цены бензина марки *M* для АЗС данной компании, затем — название компании). Если бензин марки *M* не предлагался данной компанией, то разброс положить равным −1. Сведения о каждой компании выводить на новой строке, данные упорядочивать по убыванию значений разброса, а для равных значений разброса — по названиям компаний в алфавитном порядке.

LinqObj44. Исходная последовательность содержит сведения об автозаправочных станциях (АЗС). Каждый элемент последовательности включает следующие поля:

```
<Марка бензина> <Цена 1 литра (в копейках)> <Компания> <Улица>
```

Названия компаний и улиц не содержат пробелов. В качестве марки бензина указываются числа 92, 95 или 98. Каждая компания имеет не более одной АЗС на каждой улице; цены на разных АЗС одной и той же компании могут различаться. Для каждой компании определить разброс цен для всех марок бензина (вначале выводить название компании, затем три числа — разброс цен для бензина марки 92, 95 и 98). При отсутствии бензина нужной марки выводить число −1. Сведения о каждой компании выводить на новой строке и упорядочивать по названиям компаний в алфавитном порядке.

LinqObj45. Исходная последовательность содержит сведения об автозаправочных станциях (АЗС). Каждый элемент последовательности включает следующие поля:

```
<Компания> <Цена 1 литра (в копейках)> <Марка бензина> <Улица>
```

Названия компаний и улиц не содержат пробелов. В качестве марки бензина указываются числа 92, 95 или 98. Каждая компания имеет не более одной АЗС на каждой улице; цены на разных АЗС одной и той же компании могут различаться. Для каждой улицы определить количество АЗС (вначале выводить название улицы, затем количество АЗС). Сведения о каждой улице выводить на новой строке и упорядочивать по названиям улиц в алфавитном порядке.

LinqObj46. Исходная последовательность содержит сведения об автозаправочных станциях (АЗС). Каждый элемент последовательности включает следующие поля:

```
<Улица> <Марка бензина> <Цена 1 литра (в копейках)> <Компания>
```

Названия компаний и улиц не содержат пробелов. В качестве марки бензина указываются числа 92, 95 или 98. Каждая компания имеет не более одной АЗС на каждой улице; цены на разных АЗС одной и той же компании могут различаться. Для каждой компании определить количество АЗС, предлагавших все три марки бензина (вначале выводить количество

АЗС, затем — название компании; количество может быть равно 0). Сведения о каждой компании выводить на новой строке и упорядочивать по убыванию количества АЗС, а при равных количествах — по названиям компаний в алфавитном порядке.

LinqObj47. Исходная последовательность содержит сведения об автозаправочных станциях (АЗС). Каждый элемент последовательности включает следующие поля:

```
<Цена 1 литра (в копейках)> <Компания>
<Улица> <Марка бензина>
```

Названия компаний и улиц не содержат пробелов. В качестве марки бензина указываются числа 92, 95 или 98. Каждая компания имеет не более одной АЗС на каждой улице; цены на разных АЗС одной и той же компании могут различаться. Вывести данные обо всех АЗС, предлагавших не менее двух марок бензина (вначале выводится название компании, затем название улицы, затем количество предлагавшихся марок бензина). Сведения о каждой АЗС выводить на новой строке и упорядочивать по названиям компаний в алфавитном порядке, а для одинаковых компаний — по названиям улиц (также в алфавитном порядке). Если ни одной требуемой АЗС не найдено, то записать в результирующий файл строку «Нет».

LinqObj48. Исходная последовательность содержит сведения об автозаправочных станциях (АЗС). Каждый элемент последовательности включает следующие поля:

```
<Цена 1 литра (в копейках)> <Улица> <Марка
бензина> <Компания>
```

Названия компаний и улиц не содержат пробелов. В качестве марки бензина указываются числа 92, 95 или 98. Каждая компания имеет не более одной АЗС на каждой улице; цены на разных АЗС одной и той же компании могут различаться. Перебрать все возможные комбинации улиц и компаний, содержащихся в исходном наборе данных, и для каждой пары «улица–компания» вывести название улицы, название компании и количество марок бензина, которое предлагает АЗС данной компании, расположенная на данной улице (если АЗС отсутствует, то количество полагается равным 0). Сведения о каждой паре выводить на новой строке и упорядочивать по названиям улиц в алфавитном порядке, а для одинаковых названий улиц — по названиям компаний (также в алфавитном порядке).

LinqObj49. Исходная последовательность содержит сведения о результатах сдачи учащимися ЕГЭ по математике, русскому языку и информатике (в указанном порядке). Каждый элемент последовательности включает следующие поля:

```
<Фамилия> <Инициалы> <Номер школы> <Баллы
ЕГЭ>
```

Баллы ЕГЭ представляют собой три целых числа в диапазоне от 0 до 100, которые отделяются друг от друга одним пробелом. Определить наименьший суммарный балл и вывести его. Вывести также сведения обо всех учащихся, получивших наименьший суммарный балл (для каждого учащегося указывать фамилию, инициалы и номер школы). Сведения о каждом учащемся выводить на отдельной строке и располагать в порядке их следования в исходном наборе.

LinqObj50. Исходная последовательность содержит сведения о результатах сдачи учащимися ЕГЭ по математике, русскому языку и информатике (в указанном порядке). Каждый элемент последовательности включает следующие поля:

```
<Фамилия> <Инициалы> <Баллы ЕГЭ> <Номер
школы>
```

Баллы ЕГЭ представляют собой три целых числа в диапазоне от 0 до 100, которые отделяются друг от друга одним пробелом. Определить два наибольших суммарных балла и вывести эти баллы на одной строке в порядке убывания (считать, что в исходных данных всегда присутствуют учащиеся с различными суммарными баллами). Также вывести сведения обо всех учащихся, получивших один из двух наибольших суммарных баллов (для каждого учащегося указывать фамилию, инициалы и суммарный балл). Сведения о каждом учащемся выводить на отдельной строке и располагать в порядке их следования в исходном наборе.

LinqObj51. Исходная последовательность содержит сведения о результатах сдачи учащимися ЕГЭ по математике, русскому языку и информатике (в указанном порядке). Каждый элемент последовательности включает следующие поля:

```
<Баллы ЕГЭ> <Фамилия> <Инициалы> <Номер
школы>
```

Баллы ЕГЭ представляют собой три целых числа в диапазоне от 0 до 100, которые отделяются друг от друга одним пробелом. Для каждой школы вывести сведения об учащемся, набравшем наибольший балл ЕГЭ по информатике среди учащихся этой школы. Если таких учащихся несколько, то вывести сведения о первом учащемся в порядке их следования в исходном наборе. Сведения о каждом учащемся выводить на отдельной строке, указывая номер школы, фамилию учащегося, его инициалы и балл ЕГЭ по информатике. Данные упорядочивать по возрастанию номера школы.

LinqObj52. Исходная последовательность содержит сведения о результатах сдачи учащимися ЕГЭ по математике, русскому языку и информатике (в указанном порядке). Каждый элемент последовательности включает следующие поля:

```
<Номер школы> <Фамилия> <Инициалы> <Баллы
ЕГЭ>
```

Баллы ЕГЭ представляют собой три целых числа в диапазоне от 0 до 100, которые отделяются друг от друга одним пробелом. Для каждой школы вывести сведения об учащемся, набравшем наименьший суммарный балл ЕГЭ среди учащихся этой школы. Если таких учащихся несколько, то вывести сведения о первом учащемся в алфавитном порядке их фамилий и инициалов. Сведения о каждом учащемся выводить на отдельной строке, указывая номер школы, суммарный балл ЕГЭ, фамилию учащегося и его инициалы. Данные упорядочивать по убыванию номера школы.

LinqObj53. Исходная последовательность содержит сведения о результатах сдачи учащимися ЕГЭ по математике, русскому языку и информатике (в указанном порядке). Каждый элемент последовательности включает следующие поля:

```
<Фамилия> <Инициалы> <Номер школы> <Баллы
ЕГЭ>
```

Баллы ЕГЭ представляют собой три целых числа в диапазоне от 0 до 100, которые отделяются друг от друга одним пробелом. Для каждой школы определить количество учащихся, суммарный балл которых превышает 150 баллов (вначале выводится количество учащихся, набравших в сумме более 150 баллов, затем номер школы; количество учащихся может быть равно 0). Сведения о каждой школе выводить на новой строке и упорядочивать по убыванию количества учащихся, а для одинакового количества — по возрастанию номера школы.

LinqObj54. Исходная последовательность содержит сведения о результатах сдачи учащимися ЕГЭ по математике, русскому языку и информатике (в указанном порядке). Каждый элемент последовательности включает следующие поля:

```
<Баллы ЕГЭ> <Номер школы> <Фамилия>
<Инициалы>
```

Баллы ЕГЭ представляют собой три целых числа в диапазоне от 0 до 100, которые отделяются друг от друга одним пробелом. Для каждой школы найти среднее значение суммарного

балла ЕГЭ, набранного учащимися этой школы (среднее значение является целым числом — результатом *деления нацело* суммы баллов всех учащихся на количество учащихся). Сведения о каждой школе выводить на отдельной строке, указывая средний суммарный балл ЕГЭ и номер школы. Данные упорядочивать по убыванию среднего балла, а при равных значениях среднего балла — по возрастанию номера школы.

LinqObj55. Исходная последовательность содержит сведения о результатах сдачи учащимися ЕГЭ по математике, русскому языку и информатике (в указанном порядке). Каждый элемент последовательности включает следующие поля:

```
<Баллы ЕГЭ> <Фамилия> <Инициалы> <Номер школы>
```

Баллы ЕГЭ представляют собой три целых числа в диапазоне от 0 до 100, которые отделяются друг от друга одним пробелом. Вывести сведения об учащихся, набравших не менее 50 баллов по каждому предмету (вначале выводится фамилия и инициалы, затем номер школы и суммарный балл ЕГЭ по всем предметам). Сведения о каждом учащемся выводить на отдельной строке в алфавитном порядке фамилий и инициалов, а при их совпадении — в порядке следования учащихся в наборе исходных данных. Если ни один из учащихся не удовлетворяет указанным условиям, то записать в результирующий файл текст «Требуемые учащиеся не найдены».

LinqObj56. Исходная последовательность содержит сведения о результатах сдачи учащимися ЕГЭ по математике, русскому языку и информатике (в указанном порядке). Каждый элемент последовательности включает следующие поля:

```
<Фамилия> <Инициалы> <Баллы ЕГЭ> <Номер школы>
```

Баллы ЕГЭ представляют собой три целых числа в диапазоне от 0 до 100, которые отделяются друг от друга одним пробелом. Вывести сведения об учащихся, набравших более 90 баллов хотя бы по одному из предметов (вначале выводится фамилия и инициалы, затем номер школы). Сведения о каждом учащемся выводить на отдельной строке и располагать в алфавитном порядке фамилий и инициалов, а при их совпадении — по возрастанию номера школы. Если ни один из учащихся не удовлетворяет указанным условиям, то записать в результирующий файл текст «Требуемые учащиеся не найдены».

LinqObj57. Исходная последовательность содержит сведения о результатах сдачи учащимися ЕГЭ по математике, русскому языку и информатике (в указанном порядке). Каждый элемент последовательности включает следующие поля:

```
<Номер школы> <Фамилия> <Инициалы> <Баллы ЕГЭ>
```

Баллы ЕГЭ представляют собой три целых числа в диапазоне от 0 до 100, которые отделяются друг от друга одним пробелом. Для каждой школы вывести фамилии и инициалы трех первых учащихся (в алфавитном порядке), набравших менее 50 баллов по каждому предмету (вначале выводится номер школы, затем фамилия и инициалы). Сведения о каждом учащемся выводить на отдельной строке и упорядочивать по возрастанию номера школы, а для совпадающих номеров — в алфавитном порядке фамилий и инициалов. Если для некоторой школы имеется менее трех учащихся, удовлетворяющих указанным условиям, то вывести сведения обо всех таких учащихся. Если в исходном наборе нет ни одного учащегося, удовлетворяющего указанным условиям, то записать в результирующий файл текст «Требуемые учащиеся не найдены».

LinqObj58. Исходная последовательность содержит сведения о результатах сдачи учащимися ЕГЭ по математике, русскому языку и информатике (в указанном порядке). Каждый элемент последовательности включает следующие поля:

```
<Фамилия> <Инициалы> <Номер школы> <Баллы ЕГЭ>
```

Баллы ЕГЭ представляют собой три целых числа в диапазоне от 0 до 100, которые отделяются друг от друга одним пробелом. Для каждой школы найти трех первых учащихся (в алфавитном порядке), набравших менее 50 баллов хотя бы по одному из предметов, и вывести их фамилию, инициалы и номер школы. Сведения о каждом учащемся выводить на отдельной строке и упорядочивать в алфавитном порядке фамилий и инициалов, а при их совпадении — по возрастанию номера школы. Если для некоторой школы имеется менее трех учащихся, удовлетворяющих указанным условиям, то вывести сведения обо всех таких учащихся. Если в исходном наборе нет ни одного учащегося, удовлетворяющего указанным условиям, то записать в результирующий файл текст «Требуемые учащиеся не найдены».

LinqObj59. Исходная последовательность содержит сведения о результатах сдачи учащимися ЕГЭ по математике, русскому языку и информатике (в указанном порядке). Каждый элемент последовательности включает следующие поля:

```
<Номер школы> <Баллы ЕГЭ> <Фамилия> <Инициалы>
```

Баллы ЕГЭ представляют собой три целых числа в диапазоне от 0 до 100, которые отделяются друг от друга одним пробелом. Для каждой школы и каждого предмета определить количество учащихся, набравших не менее 50 баллов по этому предмету (вначале выводится номер школы, затем три числа — количество учащихся этой школы, набравших требуемое число баллов по математике, русскому языку и информатике; некоторые из чисел могут быть равны 0). Сведения о каждой школе выводить на новой строке и упорядочивать по возрастанию номера школы.

LinqObj60. Исходная последовательность содержит сведения о результатах сдачи учащимися ЕГЭ по математике, русскому языку и информатике (в указанном порядке). Каждый элемент последовательности включает следующие поля:

```
<Фамилия> <Инициалы> <Баллы ЕГЭ> <Номер школы>
```

Баллы ЕГЭ представляют собой три целых числа в диапазоне от 0 до 100, которые отделяются друг от друга одним пробелом. Для каждой школы и каждого предмета найти среднее значение балла ЕГЭ, набранного учащимися этой школы (среднее значение является целым числом — результатом *деления нацело* суммы баллов всех учащихся на количество учащихся). Сведения о каждой школе выводить на отдельной строке, указывая номер школы и средние баллы по математике, русскому языку и информатике. Данные упорядочивать по убыванию номера школы.

LinqObj61. 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.

LinqObj62. 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:

```
<Класс> <Фамилия> <Инициалы> <Оценка>
<Название предмета>
```

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 количество оценок по каждому предмету (если по какому-либо предмету учащийся не получил ни одной оценки, то вывести для этого предмета число 0). Сведения о каждом учащемся выводить на отдельной строке, указывая класс, фамилию, инициалы и количество оценок по алгебре, геометрии и информатике. Данные располагать в порядке возрастания номера класса, а для одинаковых классов — в алфавитном порядке фамилий и инициалов.

LinqObj63. 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:

```
<Название предмета> <Фамилия> <Инициалы>
<Класс> <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. Вывести сведения об учащихся, имеющих по алгебре среднюю оценку не более 4: фамилию, инициалы, номер класса и среднюю оценку по алгебре (выводится с двумя дробными знаками). Для учащихся, не имеющих ни одной оценки по алгебре, считать среднюю оценку равной 0.00. Сведения о каждом учащемся выводить на отдельной строке и располагать в алфавитном порядке их фамилий и инициалов. Если ни один из учащихся не удовлетворяет указанным условиям, то записать в результирующий файл текст «Требуемые учащиеся не найдены».

LinqObj64. 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:

```
<Класс> <Фамилия> <Инициалы> <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. Вывести сведения об учащихся, имеющих по информатике среднюю оценку не менее 4: номер класса, фамилию, инициалы и среднюю оценку по информатике (выводится с двумя дробными знаками). Сведения о каждом учащемся выводить на отдельной строке и располагать в порядке возрастания классов, а для одинаковых классов — в алфавитном порядке фамилий и инициалов. Если ни один из учащихся не удовлетворяет указанным условиям, то записать в результирующий файл текст «Требуемые учащиеся не найдены».

LinqObj65. Дана строка *S* — название одного из трех предметов: алгебры, геометрии или информатики. Исходная последовательность содержит сведения об оценках учащихся по этим трем предметам. Каждый элемент последовательности содержит данные об одной оценке и включает следующие поля:

```
<Фамилия> <Инициалы> <Название предмета>
<Оценка> <Класс>
```

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 класса, присутствующего в наборе исходных данных, определить число учащихся, имеющих по предмету *S* среднюю оценку не более 3.5 или не имеющих ни одной оценки по этому предмету. Сведения о каждом классе выводить на отдельной строке, указывая число найденных учащихся (число может быть равно 0) и номер класса. Данные упорядочивать по возрастанию числа учащихся, а для совпадающих чисел — по убыванию номера класса.

LinqObj66. Дана строка *S* — название одного из трех предметов: алгебры, геометрии или информатики. Исходная последовательность содержит сведения об оценках учащихся по этим трем предметам. Каждый элемент последовательности содержит данные об одной оценке и включает следующие поля:

```
<Название предмета> <Фамилия> <Инициалы>
<Оценка> <Класс>
```

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 класса, присутствующего в наборе исходных данных, определить число учащихся, имеющих по предмету *S* среднюю оценку не менее 3.5 и при этом не получивших ни одной двойки по этому предмету. Сведения о каждом классе выводить на отдельной строке, указывая номер класса и число найденных учащихся (число может быть равно 0). Данные упорядочивать по возрастанию номера класса.

LinqObj67. 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:

```
<Класс> <Название предмета> <Фамилия>
<Инициалы> <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. Найти всех *двоечников* — учащихся, получивших хотя бы одну двойку по какому-либо предмету. Вывести сведения о каждом из двоечников: номер класса, фамилию, инициалы и полученное число двоек. Сведения о каждом двоечнике выводить на отдельной строке и располагать по убыванию классов, а для одинаковых классов — в алфавитном порядке фамилий и инициалов. Если в наборе исходных данных нет ни одной двойки, то записать в результирующий файл текст «Требуемые учащиеся не найдены».

LinqObj68. 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:

```
<Класс> <Оценка> <Фамилия> <Инициалы>
<Название предмета>
```

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. Найти всех *хорошистов* — учащихся, не получивших ни одной двойки и тройки, но имеющих хотя бы одну четверку по какому-либо предмету. Вывести сведения о каждом хорошисте: полученное число

четверок, фамилию, инициалы и номер класса. Сведения о каждом учащемся выводить на отдельной строке и располагать по возрастанию количества четверок, а при их равенстве — в алфавитном порядке фамилий и инициалов. Если в наборе исходных данных нет ни одного учащегося, удовлетворяющего указанным условиям, то записать в результирующий файл текст «Требуемые учащиеся не найдены».

LinqObj69. 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:

`<Класс> <Фамилия> <Инициалы> <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 класса найти *злостных двоечников* — учащихся, получивших в данном классе максимальное суммарное число двоек по всем предметам (число не должно быть нулевым). Вывести сведения о каждом из злостных двоечников: фамилию, инициалы, номер класса и полученное число двоек. Сведения о каждом двоечнике выводить на отдельной строке и располагать в алфавитном порядке их фамилий и инициалов (сортировку по классам не проводить). Если в наборе исходных данных нет ни одной двойки, то записать в результирующий файл текст «Требуемые учащиеся не найдены».

LinqObj70. 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:

`<Оценка> <Класс> <Фамилия> <Инициалы> <Название предмета>`

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 класса найти *лучших учеников* — учащихся, получивших в данном классе максимальное суммарное число пятерок по всем предметам (число не должно быть нулевым). При поиске лучших учеников (в частности, при определении максимального суммарного числа пятерок) не следует учитывать учащихся, получивших хотя бы одну двойку или тройку. Вывести сведения о каждом из лучших учеников: номер класса, фамилию, инициалы и полученное число пятерок. Сведения о каждом учащемся выводить на отдельной строке и располагать по возрастанию классов, а для одинаковых классов — в алфавитном порядке фамилий и инициалов. Если в наборе исходных данных нет ни одного учащегося, удовлетворяющего указанным условиям, то записать в результирующий файл текст «Требуемые учащиеся не найдены».

Обработка нескольких взаимосвязанных последовательностей

В каждом задании данной подгруппы требуется обработать несколько (от двух до четырех) последовательностей из следующего набора:

A: сведения о потребителях, содержащие поля «Код потребителя», «Год рождения», «Улица проживания»;

B: сведения о товарах, содержащие поля «Артикул товара», «Категория», «Страна-производитель»;

C: скидки для потребителей в различных магазинах, содержащие поля «Код потребителя», «Название магазина», «Скидка (в процентах)»;

D: цены товаров в различных магазинах, содержащие поля «Артикул товара», «Название магазина», «Цена (в рублях)»;

E: сведения о покупках потребителей в различных магазинах, содержащие поля «Код потребителя», «Артикул товара», «Название магазина».

Порядок следования полей для элементов каждой последовательности определяется в формулировке задания.

В последовательности *A* все элементы имеют различные значения поля «Код потребителя». В последовательности *B* все элементы имеют различные значения поля «Артикул товара». В последовательности *C* все элементы имеют различные комбинации полей «Код потребителя» и «Название магазина». В последовательности *D* все элементы имеют различные комбинации полей «Артикул товара» и «Название магазина». Последовательность *E* может содержать одинаковые элементы (это соответствует ситуации, при которой один и тот же потребитель приобрел в одном и том же магазине несколько одинаковых товаров).

Все значения кодов потребителей и артикулов товаров, присутствующие в последовательностях *C*, *D* и *E*, обязательно содержатся в последовательностях *A* и *B*. Некоторые значения кодов потребителей и артикулов товаров, присутствующие в последовательностях *A* и *B*, могут отсутствовать в остальных последовательностях. Любая комбинация «магазин–товар», присутствующая в последовательности *E*, обязательно присутствует и в последовательности *D*. Комбинация «потребитель–магазин», присутствующая в последовательности *E*, может отсутствовать в последовательности *C*; это означает, что при покупке указанного товара в данном магазине потребитель не имел скидки (т. е. скидка была равна 0).

Коды потребителей, годы рождения, скидки и цены задаются целыми числами; значения скидок лежат в диапазоне от 5 до 50. Прочие поля являются строковыми и не содержат пробелов. Артикулы товаров имеют формат «AAddd-dddd», где на позициях, помеченных символом «A», располагается какая-либо заглавная латинская буква, а на позициях, помеченных символом «d», — какая-либо цифра.

Если потребитель приобрел товар, имеющий цену p, со скидкой d процентов, то размер скидки на данный товар должен вычисляться по формуле $p \cdot d$ / 100, где символ «/» обозначает операцию целочисленного деления (иными словами, при вычислении размера скидки копейки отбрасываются).

LinqObj71. Даны последовательности *A* и *C*, включающие следующие поля:

A: `<Код потребителя> <Год рождения> <Улица проживания>`

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

Свойства последовательностей описаны в преамбуле к данной подгруппе заданий. Для каждого магазина определить потребителей, имеющих максимальную скидку в этом магазине (вначале выводится название магазина, затем код потребителя, его год рождения и значение максимальной скидки). Если для некоторого магазина имеется несколько потребителей с максимальной скидкой, то вывести данные о потребителе с минимальным кодом. Сведения о каждом магазине выводить на новой строке и упорядочивать по названиям магазинов в алфавитном порядке.

LinqObj72. Даны последовательности *A* и *C*, включающие следующие поля:

A: `<Код потребителя> <Улица проживания> <Год рождения>`

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

Свойства последовательностей описаны в преамбуле к данной подгруппе заданий. Для каждого потребителя, указанного в *A*, определить количество магазинов, в которых ему предоставляется скидка (вначале выводится количество магази-

нов, затем код потребителя, затем его улица проживания). Если у некоторого потребителя нет скидки ни в одном магазине, то для него выводится количество магазинов, равное 0. Сведения о каждом потребителе выводить на новой строке и упорядочивать по возрастанию количества магазинов, а при равном количестве — по возрастанию кодов потребителей.

LinqObj73. Даны последовательности *A* и *C*, включающие следующие поля:

A: `<Год рождения> <Код потребителя> <Улица проживания>`
C: `<Код потребителя> <Название магазина> <Скидка (в процентах)>`

Свойства последовательностей описаны в преамбуле к данной подгруппе заданий. Для каждого магазина и каждой улицы определить количество потребителей, живущих на этой улице и имеющих скидку в этом магазине (вначале выводится название магазина, затем название улицы, затем количество потребителей со скидкой). Если для некоторой пары «магазин–улица» потребители со скидкой не найдены, то данные об этой паре не выводятся. Сведения о каждой паре «магазин–улица» выводить на новой строке и упорядочивать по названиям магазинов в алфавитном порядке, а для одинаковых названий магазинов — по названиям улиц (также в алфавитном порядке).

LinqObj74. Даны последовательности *A* и *C*, включающие следующие поля:

A: `<Улица проживания> <Код потребителя> <Год рождения>`
C: `<Название магазина> <Скидка (в процентах)> <Код потребителя>`

Свойства последовательностей описаны в преамбуле к данной подгруппе заданий. Для каждого магазина и каждого года рождения из *A* определить среднюю скидку в данном магазине (в процентах) для потребителей с этим годом рождения (вначале выводится название магазина, затем год рождения, затем значение средней скидки в процентах). Дробная часть найденного среднего значения отбрасывается, скидка выводится как целое число. При вычислении средней скидки учитываются только потребители данного года рождения, имеющие скидку в данном магазине. Если таких потребителей для данного магазина нет, то для этой пары «магазин–год» в качестве средней скидки выводится 0. Сведения о каждой паре «магазин–год» выводить на новой строке и упорядочивать по названиям магазинов в алфавитном порядке, а для одинаковых названий магазинов — по возрастанию номеров года.

LinqObj75. Даны последовательности *B* и *D*, включающие следующие поля:

B: `<Артикул товара> <Категория> <Страна-производитель>`
D: `<Название магазина> <Артикул товара> <Цена (в рублях)>`

Свойства последовательностей описаны в преамбуле к данной подгруппе заданий. Для каждого магазина и каждой категории товаров определить количество различных артикулов товаров данной категории, имеющихся в данном магазине (вначале выводится название магазина, затем категория, затем количество различных артикулов товаров). Если для некоторого магазина товары данной категории отсутствуют, то информация о соответствующей паре «магазин–категория» не выводится. Сведения о каждой паре «магазин–категория» выводить на новой строке и упорядочивать по названиям магазинов в алфавитном порядке, а для одинаковых названий магазинов — по названиям категорий (также в алфавитном порядке).

LinqObj76. Даны последовательности *B* и *D*, включающие следующие поля:

B: `<Страна-производитель> <Категория> <Артикул товара>`
D: `<Артикул товара> <Название магазина> <Цена (в рублях)>`

Свойства последовательностей описаны в преамбуле к данной подгруппе заданий. Для каждой страны-производителя определить количество магазинов, предлагающих товары, произведенные в этой стране, а также минимальную цену для товаров из данной страны по всем магазинам (вначале выводится количество магазинов, затем страна, затем минимальная цена). Если для некоторой страны не найдено ни одного товара, представленного в каком-либо магазине, то количество магазинов и минимальная цена полагаются равными 0. Сведения о каждой стране выводить на новой строке и упорядочивать по возрастанию количества магазинов, а в случае одинакового количества — по названиям стран в алфавитном порядке.

LinqObj77. Даны последовательности *B* и *D*, включающие следующие поля:

B: `<Категория> <Артикул товара> <Страна-производитель>`
D: `<Артикул товара> <Цена (в рублях)> <Название магазина>`

Свойства последовательностей описаны в преамбуле к данной подгруппе заданий. Для каждой категории товаров определить количество магазинов, предлагающих товары данной категории, а также количество стран, в которых произведены товары данной категории, представленные в магазинах (вначале выводится количество магазинов, затем название категории, затем количество стран). Если для некоторой категории не найдено ни одного товара, представленного в каком-либо магазине, то информация о данной категории не выводится. Сведения о каждой категории выводить на новой строке и упорядочивать по убыванию количества магазинов, а в случае одинакового количества — по названиям категорий в алфавитном порядке.

LinqObj78. Даны последовательности *D* и *E*, включающие следующие поля:

D: `<Цена (в рублях)> <Артикул товара> <Название магазина>`
E: `<Код потребителя> <Название магазина> <Артикул товара>`

Свойства последовательностей описаны в преамбуле к данной подгруппе заданий. Для каждого товара определить количество покупок данного товара и максимальную цену покупки (вначале выводится количество покупок, затем артикул товара, затем максимальная цена покупки). Если некоторый товар ни разу не был продан, то информация об этом товаре не выводится. Сведения о каждом товаре выводить на новой строке и упорядочивать по возрастанию количества покупок, а в случае одинакового количества — по артикулам товаров в алфавитном порядке.

LinqObj79. Даны последовательности *D* и *E*, включающие следующие поля:

D: `<Название магазина> <Цена (в рублях)> <Артикул товара>`
E: `<Код потребителя> <Артикул товара> <Название магазина>`

Свойства последовательностей описаны в преамбуле к данной подгруппе заданий. Для каждого магазина и каждого потребителя определить суммарную стоимость покупок, сделанных этим потребителем в данном магазине (вначале вы-

водится название магазина, затем код потребителя, затем суммарная стоимость покупок). Если потребитель не приобрел ни одного товара в некотором магазине, то информация о соответствующей паре «магазин–потребитель» не выводится. Сведения о каждой паре «магазин–потребитель» выводить на новой строке и упорядочивать по названиям магазинов в алфавитном порядке, а в случае одинаковых названий — по возрастанию кодов потребителей.

LinqObj80. Даны последовательности *D* и *E*, включающие следующие поля:

D: `<Цена (в рублях)> <Название магазина> <Артикул товара>`
E: `<Артикул товара> <Название магазина> <Код потребителя>`

Свойства последовательностей описаны в преамбуле к данной подгруппе заданий. Для каждой пары «магазин–товар», указанной в *D*, определить суммарную стоимость продаж этого товара в данном магазине (вначале выводится название магазина, затем артикул товара, затем суммарная стоимость его продаж). Если товар ни разу не был продан в некотором магазине, то для соответствующей пары «магазин–товар» в качестве суммарной стоимости выводится 0. Сведения о каждой паре «магазин–товар» выводить на новой строке и упорядочивать по названиям магазинов в алфавитном порядке, а в случае одинаковых названий — по артикулам товаров (также в алфавитном порядке).

LinqObj81. Даны последовательности *B*, *D* и *E*, включающие следующие поля:

B: `<Артикул товара> <Страна-производитель> <Категория>`
D: `<Название магазина> <Артикул товара> <Цена (в рублях)>`
E: `<Название магазина> <Код потребителя> <Артикул товара>`

Свойства последовательностей описаны в преамбуле к данной подгруппе заданий. Для каждой страны-производителя определить количество приобретенных товаров из данной страны и суммарную стоимость приобретенных товаров (вначале выводится название страны, затем количество товаров, затем суммарная стоимость). Если сведения о проданных товарах для некоторой страны-производителя отсутствуют, то информация об этой стране не выводится. Сведения о каждой стране выводить на новой строке и упорядочивать по названиям стран в алфавитном порядке.

LinqObj82. Даны последовательности *B*, *D* и *E*, включающие следующие поля:

B: `<Категория> <Страна-производитель> <Артикул товара>`
D: `<Цена (в рублях)> <Артикул товара> <Название магазина>`
E: `<Артикул товара> <Код потребителя> <Название магазина>`

Свойства последовательностей описаны в преамбуле к данной подгруппе заданий. Для каждого потребителя определить количество категорий приобретенных им товаров и максимальную цену одной его покупки (вначале выводится количество категорий товаров, затем код потребителя, затем максимальная цена покупки). Сведения о каждом потребителе выводить на новой строке и упорядочивать по убыванию количества категорий, а при совпадении количества категорий — по возрастанию кодов потребителей.

LinqObj83. Даны последовательности *B*, *D* и *E*, включающие следующие поля:

B: `<Страна-производитель> <Артикул товара> <Категория>`
D: `<Цена (в рублях)> <Название магазина> <Артикул товара>`
E: `<Название магазина> <Артикул товара> <Код потребителя>`

Свойства последовательностей описаны в преамбуле к данной подгруппе заданий. Для каждого магазина, указанного в *E*, и каждой страны-производителя определить суммарную стоимость товаров из данной страны, проданных в данном магазине (вначале выводится название магазина, затем название страны, затем суммарная стоимость). Если для некоторой пары «магазин–страна» отсутствует информация о проданных товарах, то в качестве суммарной стоимости выводится 0. Сведения о каждой паре «магазин–страна» выводить на новой строке и упорядочивать по названиям магазинов в алфавитном порядке, а для одинаковых названий магазинов — по названиям стран (также в алфавитном порядке).

LinqObj84. Даны последовательности *C*, *D* и *E*, включающие следующие поля:

C: `<Скидка (в процентах)> <Название магазина> <Код потребителя>`
D: `<Артикул товара> <Название магазина> <Цена (в рублях)>`
E: `<Артикул товара> <Название магазина> <Код потребителя>`

Свойства последовательностей описаны в преамбуле к данной подгруппе заданий. Для каждого магазина и каждого товара определить количество покупок этого товара со скидкой в данном магазине и суммарную стоимость этих покупок с учетом скидки (вначале выводится название магазина, затем артикул товара, затем количество покупок со скидкой и их суммарная стоимость). При вычислении размера скидки на товар копейки отбрасываются. Если для некоторой пары «магазин–товар» не найдено ни одной покупки со скидкой, то информация о данной паре не выводится. Если не найдено ни одной подходящей пары «магазин–товар», то записать в результирующий файл текст «Требуемые данные не найдены». Сведения о каждой паре «магазин–товар» выводить на новой строке и упорядочивать по названиям магазинов в алфавитном порядке, а для одинаковых названий — по артикулам товаров (также в алфавитном порядке).

LinqObj85. Даны последовательности *C*, *D* и *E*, включающие следующие поля:

C: `<Название магазина> <Код потребителя> <Скидка (в процентах)>`
D: `<Артикул товара> <Цена (в рублях)> <Название магазина>`
E: `<Артикул товара> <Код потребителя> <Название магазина>`

Свойства последовательностей описаны в преамбуле к данной подгруппе заданий. Для каждой пары «потребитель–магазин», указанной в *E*, определить суммарный размер скидок на все товары, приобретенные этим потребителем в данном магазине (вначале выводится код потребителя, затем название магазина, затем суммарный размер скидки). При вычислении размера скидки на каждый приобретенный товар копейки отбрасываются. Если потребитель приобретал товары в некотором магазине без скидки, то информация о соответствующей паре «потребитель–магазин» не выводится. Если не найдено ни одной подходящей пары «потребитель–магазин», то записать в результирующий файл текст «Требуемые данные не найдены». Сведения о каждой паре «потребитель–магазин» выводить на новой строке и упорядочи-

вать по возрастанию кодов потребителей, а для одинаковых кодов — по названиям магазинов в алфавитном порядке.

LinqObj86. Даны последовательности *C*, *D* и *E*, включающие следующие поля:

C: `<Скидка (в процентах)> <Код потребителя> <Название магазина>`
D: `<Название магазина> <Цена (в рублях)> <Артикул товара>`
E: `<Код потребителя> <Название магазина> <Артикул товара>`

Свойства последовательностей описаны в преамбуле к данной подгруппе заданий. Для каждой пары «товар–магазин», указанной в *E*, определить максимальный размер скидки на этот товар при его приобретении в данном магазине (вначале выводится артикул товара, затем название магазина, затем максимальный размер скидки). При вычислении размера скидки на товар копейки отбрасываются. Если все продажи товара в некотором магазине проводились без скидки, то в качестве максимального размера скидки для данной пары выводится 0. Сведения о каждой паре «товар–магазин» выводить на новой строке и упорядочивать по артикулам товаров в алфавитном порядке, а для одинаковых артикулов — по названиям магазинов (также в алфавитном порядке).

LinqObj87. Даны последовательности *A*, *D* и *E*, включающие следующие поля:

A: `<Год рождения> <Улица проживания> <Код потребителя>`
D: `<Артикул товара> <Название магазина> <Цена (в рублях)>`
E: `<Название магазина> <Код потребителя> <Артикул товара>`

Свойства последовательностей описаны в преамбуле к данной подгруппе заданий. Для каждой улицы проживания и каждого магазина, указанного в *E*, определить суммарную стоимость покупок в данном магазине, сделанных всеми потребителями, живущими на данной улице (вначале выводится название улицы, затем название магазина, затем суммарная стоимость покупок). Если для некоторой пары «улица–магазин» отсутствует информация о проданных товарах, то в качестве суммарной стоимости выводится 0. Сведения о каждой паре «улица–магазин» выводить на новой строке и упорядочивать по названиям улиц в алфавитном порядке, а для одинаковых названий улиц — по названиям магазинов (также в алфавитном порядке).

LinqObj88. Даны последовательности *A*, *D* и *E*, включающие следующие поля:

A: `<Улица проживания> <Год рождения> <Код потребителя>`
D: `<Артикул товара> <Цена (в рублях)> <Название магазина>`
E: `<Название магазина> <Артикул товара> <Код потребителя>`

Свойства последовательностей описаны в преамбуле к данной подгруппе заданий. Для каждого года рождения, указанного в *A*, и каждого товара, указанного в *E*, определить суммарную стоимость покупок данного товара, сделанных всеми потребителями с данным годом рождения (вначале выводится год рождения, затем артикул товара, затем суммарная стоимость покупок). Если для некоторой пары «год–товар» отсутствуют сведения о проданных товарах, то информация об этой паре не выводится. Сведения о каждой паре «год–товар» выводить на новой строке и упорядочивать по убыванию номеров года, а для одинаковых номеров — по артикулам товаров в алфавитном порядке.

LinqObj89. Даны последовательности *A*, *D* и *E*, включающие следующие поля:

A: `<Код потребителя> <Год рождения> <Улица проживания>`
D: `<Название магазина> <Цена (в рублях)> <Артикул товара>`
E: `<Код потребителя> <Артикул товара> <Название магазина>`

Свойства последовательностей описаны в преамбуле к данной подгруппе заданий. Для каждого магазина, указанного в *E*, определить потребителя с наименьшим годом рождения, купившего один или более товаров в данном магазине (вначале выводится название магазина, затем код потребителя, год его рождения и суммарная стоимость товаров, приобретенных потребителем в данном магазине). Если для некоторого магазина имеется несколько покупателей с наименьшим годом рождения, то выводятся данные обо всех таких покупателях. Сведения о каждой паре «магазин–потребитель» выводить на новой строке и упорядочивать по названиям магазинов в алфавитном порядке, а для одинаковых названий магазинов — по возрастанию кодов потребителей.

LinqObj90. Даны последовательности *A*, *B* и *E*, включающие следующие поля:

A: `<Улица проживания> <Код потребителя> <Год рождения>`
B: `<Страна-производитель> <Категория> <Артикул товара>`
E: `<Артикул товара> <Код потребителя> <Название магазина>`

Свойства последовательностей описаны в преамбуле к данной подгруппе заданий. Для каждого года рождения из *A* определить страну, в которой было произведено максимальное количество товаров, приобретенных потребителями этого года рождения (вначале выводится год, затем название страны, затем максимальное количество покупок). Если для некоторой пары «год–страна» отсутствует информация о проданных товарах, то эта пара не обрабатывается (в частности, если потребители некоторого года рождения не сделали ни одной покупки, то информация об этом годе не выводится). Если для какого-либо года рождения имеется несколько стран с наибольшим числом приобретенных товаров, то выводятся данные о первой из таких стран (в алфавитном порядке). Сведения о каждом годе выводить на новой строке и упорядочивать по убыванию номера года.

LinqObj91. Даны последовательности *A*, *B* и *E*, включающие следующие поля:

A: `<Улица проживания> <Год рождения> <Код потребителя>`
B: `<Артикул товара> <Страна-производитель> <Категория>`
E: `<Название магазина> <Артикул товара> <Код потребителя>`

Свойства последовательностей описаны в преамбуле к данной подгруппе заданий. Для каждой улицы и каждой категории товаров определить количество стран, в которых были произведены товары данной категории, купленные потребителями, живущими на этой улице (вначале выводится название улицы, затем название категории, затем количество стран). Если для какой-либо категории отсутствует информация о товарах, проданных жителям некоторой улицы, то информация о соответствующей паре «улица–категория» не выводится. Сведения о каждой паре «улица–категория» выводить на новой строке и упорядочивать по названиям улиц в алфавитном порядке, а для одинаковых улиц — по названиям категорий (также в алфавитном порядке).

LinqObj92. Даны последовательности *A*, *B* и *E*, включающие следующие поля:

A: `<Год рождения> <Улица проживания> <Код потребителя>`
B: `<Страна-производитель> <Артикул товара> <Категория>`
E: `<Артикул товара> <Название магазина> <Код потребителя>`

Свойства последовательностей описаны в преамбуле к данной подгруппе заданий. Для каждой категории товаров определить улицу с максимальным суммарным количеством товаров данной категории, приобретенных жителями этой улицы (вначале выводится название категории, затем название улицы, затем максимальное суммарное количество покупок). Если для какой-либо категории отсутствует информация о товарах, проданных жителям некоторой улицы, то суммарное количество покупок для соответствующей пары «категория–улица» считается равным 0 (при этом возможна ситуация, когда наибольшее количество покупок для какой-либо категории равно 0). Если для некоторой категории имеется несколько улиц с наибольшим количеством покупок, то выводятся данные обо всех таких улицах. Сведения о каждой паре «категория–улица» выводить на новой строке и упорядочивать по названиям категорий в алфавитном порядке, а для одинаковых категорий — по названиям улиц (также в алфавитном порядке).

LinqObj93. Даны последовательности *A*, *B*, *C* и *E*, включающие следующие поля:

A: `<Код потребителя> <Улица проживания> <Год рождения>`
B: `<Категория> <Страна-производитель> <Артикул товара>`
C: `<Название магазина> <Код потребителя> <Скидка (в процентах)>`
E: `<Код потребителя> <Артикул товара> <Название магазина>`

Свойства последовательностей описаны в преамбуле к данной подгруппе заданий. Для каждой страны-производителя и улицы проживания определить максимальный размер скидки (в процентах) для изделий, произведенных в данной стране и приобретенных потребителями, живущими на данной улице (вначале выводится название страны, затем название улицы, затем максимальный размер скидки). Если для некоторой пары «страна–улица» все товары были приобретены без скидки, то в качестве максимального размера скидки выводится 0. Если для некоторой пары «страна–улица» отсутствует информация о приобретенных товарах, то информация о данной паре не выводится. Сведения о каждой паре «страна–улица» выводить на новой строке и упорядочивать по названиям стран в алфавитном порядке, а для одинаковых названий — по названиям улиц (также в алфавитном порядке).

LinqObj94. Даны последовательности *A*, *B*, *C* и *E*, включающие следующие поля:

A: `<Год рождения> <Код потребителя> <Улица проживания>`
B: `<Артикул товара> <Категория> <Страна-производитель>`
C: `<Код потребителя> <Название магазина> <Скидка (в процентах)>`
E: `<Название магазина> <Код потребителя> <Артикул товара>`

Свойства последовательностей описаны в преамбуле к данной подгруппе заданий. Для каждого магазина, указанного в *E*, и каждой категории товаров определить минимальное значение года рождения среди тех потребителей, которые приобрели в данном магазине один или более товаров данной категории, и количество товаров данной категории, приобретенных со скидкой в данном магазине потребителями этого года рождения (вначале выводится название магазина, затем название категории, затем номер минимального года рождения и количество товаров, приобретенных со скидкой). Если для некоторой пары «магазин–категория» информация о проданных товарах отсутствует, то данные об этой паре не выводятся. Если для некоторой пары «магазин–категория» покупатели с минимальным годом рождения приобрели все товары без скидки, то в качестве значения количества товаров, приобретенных со скидкой, выводится 0. Сведения о каждой паре «магазин–категория» выводить на новой строке и упорядочивать по названиям магазинов в алфавитном порядке, а для одинаковых названий магазинов — по названиям категорий (также в алфавитном порядке).

LinqObj95. Даны последовательности *A*, *C*, *D* и *E*, включающие следующие поля:

A: `<Код потребителя> <Улица проживания> <Год рождения>`
C: `<Название магазина> <Скидка (в процентах)> <Код потребителя>`
D: `<Название магазина> <Артикул товара> <Цена (в рублях)>`
E: `<Код потребителя> <Название магазина> <Артикул товара>`

Свойства последовательностей описаны в преамбуле к данной подгруппе заданий. Для каждого артикула товара, указанного в *E*, и каждой улицы проживания определить суммарный размер скидки на изделия данного артикула, приобретенные потребителями, живущими на данной улице (вначале выводится артикул товара, затем название улицы, затем суммарный размер скидки). При вычислении размера скидки на товар копейки отбрасываются. Если на проданный товар скидка отсутствует, то ее размер полагается равным 0. Если для некоторой пары «товар–улица» отсутствует информация о приобретенных товарах, то данные об этой паре не выводятся. Если для некоторой пары «товар–улица» все изделия были приобретены без скидок, то в качестве суммарной скидки для этой пары выводится 0. Сведения о каждой паре «товар–улица» выводить на новой строке и упорядочивать по артикулам товаров в алфавитном порядке, а для одинаковых артикулов — по названиям улиц (также в алфавитном порядке).

LinqObj96. Даны последовательности *A*, *C*, *D* и *E*, включающие следующие поля:

A: `<Улица проживания> <Год рождения> <Код потребителя>`
C: `<Код потребителя> <Скидка (в процентах)> <Название магазина>`
D: `<Цена (в рублях)> <Артикул товара> <Название магазина>`
E: `<Название магазина> <Артикул товара> <Код потребителя>`

Свойства последовательностей описаны в преамбуле к данной подгруппе заданий. Для каждого года рождения, указанного в *A*, и каждого магазина, указанного в *E*, определить суммарную стоимость всех товаров (с учетом скидки), проданных в данном магазине потребителям данного года рождения (вначале выводится номер года, затем название магазина, затем суммарная стоимость проданных товаров с учетом скидки). При вычислении размера скидки на товар копейки отбрасываются. Если на проданный товар скидка отсутствует, то ее размер полагается равным 0. Если для некоторой пары «год–магазин» отсутствует информация о проданных товарах, то данные об этой паре не выводятся. Све-

дения о каждой паре «год–магазин» выводить на новой строке и упорядочивать по возрастанию номеров года, а для одинаковых номеров — по названиям магазинов в алфавитном порядке.

LinqObj97. Даны последовательности *B*, *C*, *D* и *E*, включающие следующие поля:

B: `<Категория> <Артикул товара> <Страна-производитель>`
C: `<Скидка (в процентах)> <Название магазина> <Код потребителя>`
D: `<Цена (в рублях)> <Название магазина> <Артикул товара>`
E: `<Название магазина> <Код потребителя> <Артикул товара>`

Свойства последовательностей описаны в преамбуле к данной подгруппе заданий. Для каждой страны-производителя и каждого потребителя определить суммарную стоимость (с учетом скидки) всех товаров, произведенных в данной стране и приобретенных этим потребителем (вначале выводится название страны, затем код потребителя, затем суммарная стоимость с учетом скидки). При вычислении размера скидки на товар копейки отбрасываются. Если на проданный товар скидка отсутствует, то ее размер полагается равным 0. Если для некоторой пары «страна–потребитель» данные о покупках отсутствуют, то информация об этой паре не выводится. Сведения о каждой паре «страна–потребитель» выводить на новой строке и упорядочивать по названиям стран в алфавитном порядке, а для одинаковых названий стран — по возрастанию кодов потребителей.

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).

LinqObj99. Даны последовательности *A*, *B*, *D* и *E*, включающие следующие поля:

A: `<Код потребителя> <Улица проживания> <Год рождения>`
B: `<Категория> <Страна-производитель> <Артикул товара>`
D: `<Артикул товара> <Название магазина> <Цена (в рублях)>`
E: `<Артикул товара> <Название магазина> <Код потребителя>`

Свойства последовательностей описаны в преамбуле к данной подгруппе заданий. Для каждой категории товаров и каждой улицы проживания определить магазин, продавший товар данной категории по минимальной цене потребителю, живущему на данной улице (вначале выводится название категории, затем название улицы, затем название магазина и минимальная цена товара). Если для некоторой пары «категория–улица» отсутствует информация о проданных товарах, то данные об этой паре не выводятся. Если для некоторой пары «категория–улица» имеется несколько магазинов, продавших товар по минимальной цене, то выводятся данные обо всех таких магазинах. Сведения о каждой тройке «категория–улица–магазин» выводить на новой строке и упорядочивать по названиям категорий в алфавитном порядке, для одинаковых названий категорий — по названиям улиц, а для одинаковых названий улиц — по названиям магазинов (также в алфавитном порядке).

LinqObj100. Даны последовательности *A*, *B*, *D* и *E*, включающие следующие поля:

A: `<Улица проживания> <Код потребителя> <Год рождения>`
B: `<Артикул товара> <Страна-производитель> <Категория>`
D: `<Название магазина> <Цена (в рублях)> <Артикул товара>`
E: `<Артикул товара> <Код потребителя> <Название магазина>`

Свойства последовательностей описаны в преамбуле к данной подгруппе заданий. Для каждой страны-производителя и каждого магазина определить потребителя с наибольшим годом рождения, купившего в данном магазине один или более товаров, произведенных в данной стране (вначале выводится название страны, затем название магазина, затем год рождения потребителя, его код, а также суммарная стоимость товаров из данной страны, купленных в данном магазине). Если для некоторой пары «страна–магазин» отсутствует информация о проданных товарах, то данные об этой паре не выводятся. Если для некоторой пары «страна–магазин» имеется несколько потребителей с наибольшим годом рождения, то выводятся данные обо всех таких потребителях. Сведения о каждой тройке «страна–магазин–потребитель» выводить на новой строке и упорядочивать по названиям стран в алфавитном порядке, для одинаковых названий стран — по названиям магазинов (также в алфавитном порядке), а для одинаковых магазинов — по возрастанию кодов потребителей.