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 изменить соответствующее свойство объекта `CurrentCulture` основного потока приложения `CurrentThread` (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. В некоторых случаях решение можно сделать более наглядным, если использовать вспомогательные функции (см. приведенный ниже пример).

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:

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>`

Количество учащихся не превосходит 100, полных однофамильцев (с совпадающей фамилией и инициалами) среди учащихся нет. Класс описывается целым числом в диапазоне от 7 до 11, оценка — целым числом в диапазоне от 2 до 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

В программе используются следующие пространства имен (в дополнение к `System` и `PT4`):

```
System.Threading, System.Globalization, System.IO,
System.Collections.Generic, System.Text, System.Linq.

static string GetVal(Dictionary<string, double> d,
  string k)
{
  double r;
  d.TryGetValue(k, out r);
  return r.ToString("f2");
}

public static void Solve()
{
  Task("LinqObj61");
// Настройка нового десятичного разделителя
  CultureInfo ci = (CultureInfo)
    Thread.CurrentThread.CurrentCulture.Clone();
  ci.NumberFormat.NumberDecimalSeparator = ".";
  Thread.CurrentThread.CurrentCulture = ci;
// Ввод
  var a = File.ReadAllLines(GetString(),
    Encoding.Default);
// Обработка
  var res = a.Select(s =>
    {
      string[] ss = s.Split();
      return new
      {
        Fio = ss[0] + " " + ss[1],
        Cls = int.Parse(ss[2]),
        Subj = ss[3],
        Mark = int.Parse(ss[4])
      };
    })
    .GroupBy(e => e.Fio, (k, ee) => new
      {
        Fio = k,
        AvrMarks =
        ee.GroupBy(e => e.Subj, e => e.Mark)
          .ToDictionary(e => e.Key, e => e.Average())
      })
    .OrderBy(e => e.Fio)
    .Select(e => e.Fio + " " +
       GetVal(e.AvrMarks, "Алгебра") + " " +
       GetVal(e.AvrMarks, "Геометрия") + " " +
       GetVal(e.AvrMarks, "Информатика"));
// Вывод
  File.WriteAllLines(GetString(), res.ToArray(),
    Encoding.Default);
}
```

Знакомство с технологией LINQ: группа LinqBegin

При задании входных и выходных последовательностей вначале указывается их размер (целое число), а затем их элементы. Все входные последовательности являются непустыми. Выходные последовательности могут быть пустыми; в этом случае требуется вывести единственное число 0 — размер данной последовательности.

Для ввода исходных последовательностей с элементами целого и строкового типа можно использовать вспомогательные методы `GetEnumerableInt` и `GetEnumerableString` соответственно (эти методы обеспечивают ввод как количества элементов, так и самих элементов и возвращают введенную последовательность). Для обработки входной последовательности в большинстве заданий достаточно использовать *единственный* оператор, содержащий вызовы нужных методов LINQ to Objects и другие необходимые конструкции, в частности, операцию `??` языка C# (в VB.NET, начиная с версии 9.0, аналогом этой операции является бинарная операция `If`). Для вывода полученной последовательности вместе с ее размером можно использовать перегруженный вариант метода `Put` с параметром-последовательностью.

В качестве примера ниже приводится фрагмент программы, извлекающей из исходной последовательности четные отрицательные числа и заменяющей их порядок следования на обратный:

[C#]
```
// Ввод исходных данных
var a = GetEnumerableInt();
// Обработка
var res = a
  .Where(e => e % 2 == 0 && e < 0).Reverse();
// Вывод результатов
Put(res);
```
[VB.NET]
```
' Ввод исходных данных
Dim a = GetEnumerableInt()
' Обработка
Dim res = a _
  .Where(Function(e) e Mod 2 = 0 AndAlso e < 0) _
  .Reverse
' Вывод результатов
Put(res)
```

В приведенном примере, как и во многих заданиях данной группы, все этапы решения можно объединить в одном операторе:

[C#]
```
Put(GetEnumerableInt()
  .Where(e => e % 2 == 0 && e < 0).Reverse());
```
[VB.NET]
```
Put(GetEnumerableInt _
  .Where(Function(e) e Mod 2 = 0 AndAlso e < 0) _
  .Reverse)
```

Если в задании идет речь о *порядковых номерах* элементов последовательности, то предполагается, что нумерация ведется от 1.

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 целочислен-

ного 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, начиная с последнего элемента, оканчивающегося цифрой (не включая этот элемент). 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 на одну и ту же цифру, и на основе этой группировки получить последовательность строк вида «D:S», где D — *ключ группировки* (т. е. некоторая цифра, на которую 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 на одну и ту же цифру, 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 на одну и ту же букву, 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: группа LinqObj

В каждом задании даются имена одного или нескольких файлов, содержащих исходные последовательности, а также имя файла, в который требуется записать результаты обработки исходных последовательностей (имя результирующего файла указывается последним). Каждая исходная последовательность содержится в отдельном файле; в начале файла с исходной последовательностью могут содержаться дополнительные данные, необходимые для выполнения задания.

Каждый элемент последовательности размещается в отдельной строке файла, поля элемента разделяются ровно одним пробелом, в начале и конце строки пробелы отсутствуют. Длина каждого строкового поля не превосходит 20 символов, инициалы не содержат пробелов и всегда состоят из 4 символов — заглавных букв имени и отчества, за которыми указываются точки (например, «А.Б.»). Все исходные числовые данные яв-

ляются положительными. В качестве десятичного разделителя используется *точка*.

Обработка отдельных последовательностей

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

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

Все данные целочисленные. Значение года лежит в диапазоне от 2000 до 2010, код клиента — в диапазоне 10–99, продолжительность занятий — в диапазоне 1–30. Найти элемент последовательности с минимальной продолжительностью занятий. Вывести эту продолжительность, а также соответствующие ей год и номер месяца (в указанном порядке). 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 поля:

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

Все данные целочисленные. Значение года лежит в диапазоне от 2000 до 2010, код клиента — в диапазоне 10–99, продолжительность занятий — в диапазоне 1–30. Найти элемент последовательности с максимальной продолжительностью занятий. Вывести эту продолжительность, а также соответствующие ей год и номер месяца (в указанном порядке). 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 поля:

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

Все данные целочисленные. Значение года лежит в диапазоне от 2000 до 2010, код клиента — в диапазоне 10–99, продолжительность занятий — в диапазоне 1–30. 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 поля:

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

Все данные целочисленные. Значение года лежит в диапазоне от 2000 до 2010, код клиента — в диапазоне 10–99, продолжительность занятий — в диапазоне 1–30. 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 поля:

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

Все данные целочисленные. Значение года лежит в диапазоне от 2000 до 2010, код клиента — в диапазоне 10–99, продолжительность занятий — в диапазоне 1–30. 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 поля:

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

Все данные целочисленные. Значение года лежит в диапазоне от 2000 до 2010, код клиента — в диапазоне 10–99, продолжительность занятий — в диапазоне 1–30. 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. Исходная последовательность содержит сведения о клиентах фитнес-центра. Вначале указывается код K одного из клиентов, а затем — элементы последовательности, каждый из которых включает следующие поля:

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

Все данные целочисленные. Значение года лежит в диапазоне от 2000 до 2010, код клиента — в диапазоне 10–99, продолжительность занятий — в диапазоне 1–30. 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. Исходная последовательность содержит сведения о клиентах фитнес-центра. Вначале указывается код K одного из клиентов, а затем — элементы последовательности, каждый из которых включает следующие поля:

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

Все данные целочисленные. Значение года лежит в диапазоне от 2000 до 2010, код клиента — в диапазоне 10–99, продолжительность занятий — в диапазоне 1–30. 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 месяц с наибольшим номером; 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. Исходная последовательность содержит сведения о клиентах фитнес-центра. Вначале указывается код K одного из клиентов, а затем — элементы последовательности, каждый из которых включает следующие поля:

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

Все данные целочисленные. Значение года лежит в диапазоне от 2000 до 2010, код клиента — в диапазоне 10–99, продолжительность занятий — в диапазоне 1–30. 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 поля:

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

Все данные целочисленные. Значение года лежит в диапазоне от 2000 до 2010, код клиента — в диапазоне 10–99, продолжительность занятий — в диапазоне 1–30. 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 поля:

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

Все данные целочисленные. Значение года лежит в диапазоне от 2000 до 2010, код клиента — в диапазоне 10–99, продолжительность занятий — в диапазоне 1–30. 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. Исходная последовательность содержит сведения о клиентах фитнес-центра. Каждый элемент последовательности включает следующие поля:

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

Все данные целочисленные. Значение года лежит в диапазоне от 2000 до 2010, код клиента — в диапазоне 10–99, продолжительность занятий — в диапазоне 1–30. Для каждого года, присутствующего в исходных данных, определить количество месяцев, в которых суммарная длительность занятий всех клиентов составляла более 15 % 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>

Номер школы содержит не более двух цифр, годы лежат в диапазоне от 1990 до 2010. Для каждого года, присутствующего в исходных данных, вывести общее число абитуриентов, поступивших в этом году (вначале выводить год, затем число абитуриентов). 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>

Номер школы содержит не более двух цифр, годы лежат в диапазоне от 1990 до 2010. 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>

Номер школы содержит не более двух цифр, годы лежат в диапазоне от 1990 до 2010. 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>

Номер школы содержит не более двух цифр, годы лежат в диапазоне от 1990 до 2010. 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>

Номер школы содержит не более двух цифр, годы лежат в диапазоне от 1990 до 2010. Найти годы, для которых число абитуриентов было не меньше среднего значения по всем годам (вначале указывать год, затем число абитуриентов для этого года). При вычислении среднего значения учитывать только годы, присутствующие в исходных данных; среднее значение может иметь ненулевую дробную часть. Сведения о каждом годе выводить на новой строке и упорядочивать по возрастанию номера года.

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

<Год поступления> <Номер школы> <Фамилия>

Номер школы содержит не более двух цифр, годы лежат в диапазоне от 1990 до 2010. Найти годы, для которых число абитуриентов было не меньше среднего значение по всем годам (вначале указывать число абитуриентов для данного года, затем год). При вычислении среднего значения учитывать только годы, присутствующие в исходных данных; среднее значение может иметь ненулевую дробную часть. 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>

Номер школы содержит не более двух цифр, годы лежат в диапазоне от 1990 до 2010. Для каждого номера школы, присутствующего в исходных данных, вывести общее число абитуриентов за все годы (вначале указывать номер школы, затем число абитуриентов). 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>

Номер школы содержит не более двух цифр, годы лежат в диапазоне от 1990 до 2010. Определить, для каких школ общее число абитуриентов за все годы было наибольшим, и вывести это число, а также количество таких школ. Каждое число выводить на новой строке.

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

<Фамилия> <Год поступления> <Номер школы>

Номер школы содержит не более двух цифр, годы лежат в диапазоне от 1990 до 2010. Определить, для каких школ общее количество абитуриентов за все годы было наибольшим, и вывести это количество, а также номера школ, для которых оно было достигнуто (номера школ упорядочивать по убыванию, каждое число выводить на новой строке).

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

<Фамилия> <Номер школы> <Год поступления>

Номер школы содержит не более двух цифр, годы лежат в диапазоне от 1990 до 2010. Для каждого номера школы, присутствующего в исходных данных, вывести общее число абитуриентов за все годы (вначале указывать число абитуриентов, затем номер школы). Сведения о каждой школе выводить на новой строке и упорядочивать по возрастанию числа абитуриентов, а для одинаковых чисел — по возрастанию номеров школ.

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

<Фамилия> <Год поступления> <Номер школы>

Номер школы содержит не более двух цифр, годы лежат в диапазоне от 1990 до 2010. Найти школы, для которых общее число абитуриентов за все годы было не меньше среднего значения по всем школам (вначале указывать номер школы, затем число абитуриентов для этой школы). При вычислении среднего значения учитывать только школы, присутствующие в исходных данных; среднее значение может иметь ненулевую дробную часть. Сведения о каждой школе выводить на новой строке и упорядочивать по убыванию номера школы.

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

<Фамилия> <Номер школы> <Год поступления>

Номер школы содержит не более двух цифр, годы лежат в диапазоне от 1990 до 2010. Найти школы, для которых общее число абитуриентов за все годы было не меньше среднего значения по всем годам (вначале указывать число абитуриентов для данной школы, затем номер школы). При вычислении среднего значения учитывать только школы, присутствующие в исходных данных; среднее значение может иметь ненулевую дробную часть. Сведения о каждой школе выводить на новой строке и упорядочивать по возрастанию числа абитуриентов, а для одинаковых чисел — по убыванию номера школы.

LinqObj25. 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.

LinqObj26. 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 4 подъездов дома вывести сведения о задолжниках, живущих в этом подъезде: число задолжников, номер подъезда, 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 убыванию числа задолжников, а для совпадающих чисел — 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:

<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 дробны-

ми 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 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:

<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.

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:

<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 жильца с наибольшей задолженностью и вывести сведения о нем: 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. Если в каком-либо подъезде задолжники отсутствуют, то данные об этом подъезде 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:

<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.

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: задолженность (выводится с двумя дробными знаками), номер этажа, 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. Если на каком-либо этаже задолжники отсутствуют, то данные об этом этаже не выводить.

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>

Имеется не более 20 различных компаний и не более 30 различных улиц; названия компаний и улиц не содержат пробелов. В качестве марки бензина указываются числа 92, 95 или 98. Цена задается целым числом в диапазоне от 2000 до 3000. 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>

Имеется не более 20 различных компаний и не более 30 различных улиц; названия компаний и улиц не содержат пробелов. В качестве марки бензина указываются числа 92, 95 или 98. Цена задается целым числом в диапазоне от 2000 до 3000. 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. Исходная последовательность содержит сведения об автозаправочных станциях (АЗС). Вначале указывается значение М одной из марок бензина, а затем — элементы последовательности, каждый из которых включает следующие поля:

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

Имеется не более 20 различных компаний и не более 30 различных улиц; названия компаний и улиц не содержат пробелов. В качестве марки бензина указываются числа 92, 95 или 98. Цена задается целым числом в диапазоне от 2000 до 3000. 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 М (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)>

Имеется не более 20 различных компаний и не более 30 различных улиц; названия компаний и улиц не содержат пробелов. В качестве марки бензина указываются числа 92, 95 или 98. Цена задается целым числом в диапазоне от 2000 до 3000. 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. Исходная последовательность содержит сведения об автозаправочных станциях (АЗС). Вначале указывается значение М одной из марок бензина, а затем — элементы последовательности, каждый из которых включает следующие поля:

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

Имеется не более 20 различных компаний и не более 30 различных улиц; названия компаний и улиц не содержат пробелов. В качестве марки бензина указываются числа 92, 95 или 98. Цена задается целым числом в диапазоне от 2000 до 3000. 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 М, 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 М 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)>

Имеется не более 20 различных компаний и не более 30 различных улиц; названия компаний и улиц не содержат пробелов. В качестве марки бензина указываются числа 92, 95 или 98. Цена задается целым числом в диапазоне от 2000 до 3000. 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. Исходная последовательность содержит сведения об автозаправочных станциях (АЗС). Вначале указывается значение M одной из марок бензина, а затем — элементы последовательности, каждый из которых включает следующие поля:

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

Имеется не более 20 различных компаний и не более 30 различных улиц; названия компаний и улиц не содержат пробелов. В качестве марки бензина указываются числа 92, 95 или 98. Цена задается целым числом в диапазоне от 2000 до 3000. 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>

Имеется не более 20 различных компаний и не более 30 различных улиц; названия компаний и улиц не содержат пробелов. В качестве марки бензина указываются числа 92, 95 или 98. Цена задается целым числом в диапазоне от 2000 до 3000. 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>

Имеется не более 20 различных компаний и не более 30 различных улиц; названия компаний и улиц не содержат пробелов. В качестве марки бензина указываются числа 92, 95 или 98. Цена задается целым числом в диапазоне от 2000 до 3000. 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>

Имеется не более 20 различных компаний и не более 30 различных улиц; названия компаний и улиц не содержат пробелов. В качестве марки бензина указываются числа 92, 95 или 98. Цена задается целым числом в диапазоне от 2000 до 3000. 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 (вначале выводить количество АЗС, затем — название компании). 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)> <Улица> <Марка бензина> <Компания>

Имеется не более 20 различных компаний и не более 30 различных улиц; названия компаний и улиц не содержат пробелов. В качестве марки бензина указываются числа 92, 95 или 98. Цена задается целым числом в диапазоне от 2000 до 3000. 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 все три марки бензина (вначале выводится название улицы, затем название компании). Сведения о каждой АЗС выводить на новой строке и упорядочивать по названиям улиц в алфавитном порядке, а для одинаковых названий улиц — по названиям компаний (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)> <Компания> <Улица> <Марка бензина>

Имеется не более 20 различных компаний и не более 30 различных улиц; названия компаний и улиц не содержат пробелов. В качестве марки бензина указываются числа 92, 95 или 98. Цена задается целым числом в диапазоне от 2000 до 3000. 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 АЗС). 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».

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>

Количество учащихся не превосходит 100. Номер школы содержит не более двух цифр, 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>

Количество учащихся не превосходит 100. Номер школы содержит не более двух цифр, 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 в 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>

Количество учащихся не превосходит 100. Номер школы содержит не более двух цифр, баллы ЕГЭ представляют собой три целых числа в диапазоне от 0 до 100, которые отделяются друг от друга одним пробелом. Для каждой школы, присутствующей в наборе исходных данных, 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>

Количество учащихся не превосходит 100. Номер школы содержит не более двух цифр, баллы ЕГЭ представляют собой три целых числа в диапазоне от 0 до 100, которые отделяются друг от друга одним пробелом. Для каждой школы, присутствующей в наборе исходных данных, вывести сведения об учащемся, набравшем наименьший суммарный балл ЕГЭ среди учащихся этой школы. Если таких учащихся несколько, то вывести сведения о последнем учащемся (в порядке их следования в исходном наборе). 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:

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

Количество учащихся не превосходит 100. Номер школы содержит не более двух цифр, баллы ЕГЭ представляют собой три целых числа в диапазоне от 0 до 100, которые отделяются друг от друга одним пробелом. Для каждой школы, присутствующей в наборе исходных данных, 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.

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:

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

Количество учащихся не превосходит 100. Номер школы содержит не более двух цифр, баллы ЕГЭ представляют собой три целых числа в диапазоне от 0 до 100, которые отделяются друг от друга одним пробелом. Для каждой школы, присутствующей в наборе исходных данных, 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.

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:

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

Количество учащихся не превосходит 100. Номер школы содержит не более двух цифр, баллы ЕГЭ представляют собой три целых числа в диапазоне от 0 до 100, которые отделяются друг от друга одним пробелом. Для каждой школы, присутствующей в наборе исходных данных, 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 (в указанном порядке). Данные упорядочивать по убыванию номера школы.

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:

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

Количество учащихся не превосходит 100. Номер школы содержит не более двух цифр, баллы ЕГЭ представляют собой три целых числа в диапазоне от 0 до 100, которые отделяются друг от друга одним пробелом. Для каждой школы, присутствующей в наборе исходных данных, 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.

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>

Количество учащихся не превосходит 100. Номер школы содержит не более двух цифр, 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 менее 50 баллов по каждому предмету (вначале выводится номер школы, затем фамилия, инициалы и суммарный балл ЕГЭ по всем предметам). Сведения о каждом учащемся выводить на отдельной строке и упорядочивать по возрастанию номера школы, а для совпадающих номеров — 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».

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:

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

Количество учащихся не превосходит 100. Номер школы содержит не более двух цифр, 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».

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:

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

Количество учащихся не превосходит 100. Номер школы содержит не более двух цифр, 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 менее 10 баллов хотя бы по одному из предметов (вначале выводится номер школы, затем фамилия и инициалы). Сведения о каждом учащемся выводить на отдельной строке и располагать по убыванию номера школы, а для совпадающих номеров — 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».

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>

Количество учащихся не превосходит 100. Номер школы содержит не более двух цифр, 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».

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>

Количество учащихся не превосходит 100, полных однофамильцев (с совпадающей фамилией и инициалами) среди учащихся нет. Класс описывается целым числом в диапазоне от 7 до 11, оценка — целым числом в диапазоне от 2 до 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>

Количество учащихся не превосходит 100, полных однофамильцев (с совпадающей фамилией и инициалами) среди учащихся нет. Класс описывается целым числом в диапазоне от 7 до 11, оценка — целым числом в диапазоне от 2 до 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>

Количество учащихся не превосходит 100, полных однофамильцев (с совпадающей фамилией и инициалами) среди учащихся нет. Класс описывается целым числом в диапазоне от 7 до 11, оценка — целым числом в диапазоне от 2 до 5. Название предмета указывается с заглавной буквы. Вывести сведения об учащихся, имеющих по алгебре среднюю оценку не более 4.0: фамилию, инициалы, номер класса и среднюю оценку по алгебре (выводится с двумя дробными знаками). Учащихся, не имеющих ни одной оценки по алгебре, не учитывать. Сведения о каждом учащемся выводить на отдельной строке и располагать в алфавитном порядке их фамилий и инициалов. Если ни один из учащихся не удовлетворяет ука-

занным 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>

Количество учащихся не превосходит 100, полных однофамильцев (с совпадающей фамилией и инициалами) среди учащихся нет. Класс описывается целым числом в диапазоне от 7 до 11, оценка — целым числом в диапазоне от 2 до 5. Название предмета указывается с заглавной буквы. Вывести сведения об учащихся, имеющих по информатике среднюю оценку не менее 4.0: 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. Исходная последовательность содержит сведения об оценках учащихся по трем предметам: алгебре, геометрии и информатике. Вначале указывается название одного из предметов S, а затем — элементы последовательности, каждый из которых содержит данные об одной оценке и включает следующие поля:

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

Количество учащихся не превосходит 100, полных однофамильцев (с совпадающей фамилией и инициалами) среди учащихся нет. Класс описывается целым числом в диапазоне от 7 до 11, оценка — целым числом в диапазоне от 2 до 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. Учащихся, не имеющих ни одной оценки по данному предмету, не учитывать. 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. Исходная последовательность содержит сведения об оценках учащихся по трем предметам: алгебре, геометрии и информатике. Вначале указывается название одного из предметов S, а затем — элементы последовательности, каждый из которых содержит данные об одной оценке и включает следующие поля:

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

Количество учащихся не превосходит 100, полных однофамильцев (с совпадающей фамилией и инициалами) среди учащихся нет. Класс описывается целым числом в диапазоне от 7 до 11, оценка — целым числом в диапазоне от 2 до 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>

Количество учащихся не превосходит 100, полных однофамильцев (с совпадающей фамилией и инициалами) среди учащихся нет. Класс описывается целым числом в диапазоне от 7 до 11, оценка — целым числом в диапазоне от 2 до 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>

Количество учащихся не превосходит 100, полных однофамильцев (с совпадающей фамилией и инициалами) среди учащихся нет. Класс описывается целым числом в диапазоне от 7 до 11, оценка — целым числом в диапазоне от 2 до 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>

Количество учащихся не превосходит 100, полных однофамильцев (с совпадающей фамилией и инициалами) среди учащихся нет. Класс описывается целым числом в диапазоне от 7 до 11, оценка — целым числом в диапазоне от 2 до 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>

Количество учащихся не превосходит 100, полных однофамильцев (с совпадающей фамилией и инициалами) среди учащихся нет. Класс описывается целым числом в диапазоне от 7 до 11, оценка — целым числом в диапазоне от 2 до 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 магазинов, 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 одина-

ковых 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 выводится 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 упорядочивать 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» вы-

водить 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 ситуа-

ция, 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. Сведения 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.

Технология LINQ to XML. Учебные задания

Узлом (node) XML-документа называется любой его компонент, в частности, комментарий (comment), инструкция обработки (processing instruction), обычный текст. *Элементом* (element) XML-документа называется именованный компонент, который может содержать другие компоненты, а также иметь *атрибуты* (attributes). В объектной модели X-DOM, входящей в состав интерфейса LINQ to XML, с каждым видом компонентов XML-документа связан соответствующий класс: XNode — общий предок всех компонентов, XText — текстовый узел, т. е. узел, представляющий собой обычный текст, XComment — комментарий, XProcessingInstruction — инструкция обработки, XElement — элемент, XAttribute — атрибут, XDocument — XML-документ в целом.

Если некоторый узел B XML-документа содержится внутри некоторого элемента A, то элемент A называется *предком* (ancestor) узла B, а узел B — *потомком* (descendant) элемента A. Если элемент A является ближайшим предком узла B, то B называется *дочерним узлом* (child node) элемента A, а элемент A — *родительским элементом* (parent) узла B; если при этом узел B является элементом, то он называется *дочерним элементом* (child element) элемента A. Первый элемент XML-документа называется *корневым элементом* (root); корневой элемент является предком для всех других элементов XML-документа, сам корневой элемент предков не имеет.

Корневой элемент считается элементом *нулевого уровня*, его дочерние узлы/элементы — узлами/элементами *первого уровня*, их дочерние узлы/элементы — узлами/элементами *второго уровня*, и т. д. В XML-документе имеется единственный элемент нулевого уровня (корневой элемент), однако могут присутствовать несколько узлов нулевого уровня (комментариев или инструкций обработки).

Если в задании говорится, что элемент содержит текстовую строку (или число), то это означает, что соответствующая строка (или строковое представление числа) является дочерним текстовым узлом данного элемента.

Во всех заданиях предполагается, что элемент имеет не больше одного дочернего текстового узла, а текстовый узел содержит хотя бы один значащий символ (т. е. символ, отличный от пробела и управляющих символов). При сохранении XML-документа следует использовать метод Save класса XDocument с единственным параметром — именем файла; это обеспечит автоматическое форматирование сохраняемого документа и удаление всех текстовых узлов, не содержащих значащих символов.

Элементы, не содержащие дочерних узлов, могут представляться в двух вариантах: в виде *парных тегов*, между которыми отсутствует текст (<a>), и в виде одного *комбинированного тега* (<a />). Элементы, не содержащие узлов, могут иметь атрибуты.

Если в условии сказано, что дан XML-документ, то это означает, что дано имя файла, содержащего этот документ. Преобразование XML-документа всегда должно завершаться сохранением преобразованного документа в том же файле, из которого был считан исходный вариант этого документа.

Если в условии упоминаются порядковые номера элементов некоторой последовательности, то предполагается, что нумерация начинается от 1.

В заданиях подгрупп, предшествующих подгруппе «Работа с пространствами имен XML-документа», предполагается, что имена всех элементов и атрибутов XML-документа имеют пустое пространство имен.

Указания, приведенные к некоторым заданиям, следует учитывать и при выполнении последующих заданий текущей подгруппы.

Создание XML-документа

Во всех заданиях данной подгруппы предполагается, что исходные текстовые файлы содержат текст в кодировке «windows-1251», а все файловые строки являются непустыми. Создаваемые XML-документы также должны иметь кодировку «windows-1251».

LinqXml1. Даны имена существующего текстового файла и создаваемого XML-документа. Создать XML-документ с корневым элементом `root` и элементами первого уровня `line`, каждый из которых содержит одну строку из исходного файла.

Указание. В конструкторе корневого элемента использовать последовательность объектов XElement, полученную методом Select из исходного набора строк.

LinqXml2. Даны имена существующего текстового файла и создаваемого XML-документа. Создать XML-документ с корневым элементом `root` и элементами первого уровня, каждый из которых содержит одну строку из исходного файла и имеет имя `line` с приписанным к нему порядковым номером строки (`line1`, `line2` и т. д.).

LinqXml3. Даны имена существующего текстового файла и создаваемого XML-документа. Создать XML-документ с корневым элементом `root` и элементами первого уровня `line`, каждый из которых содержит одну строку из исходного файла. Элемент, содержащий строку с порядковым номером N (1, 2, …), должен иметь атрибут `num` со значением, равным N.

LinqXml4. Даны имена существующего текстового файла и создаваемого XML-документа. Каждая строка текстового файла содержит несколько (одно или более) слов, разделенных ровно одним пробелом. Создать XML-документ с корневым элементом `root`, элементами первого уровня `line` и элементами второго уровня `word`. Элементы `line` соответствуют строкам исходного файла и не содержат дочерних текстовых узлов, элементы `word` каждого элемента `line` содержат по одному слову из соответствующей строки (слова располагаются в алфавитном порядке).

LinqXml5. Даны имена существующего текстового файла и создаваемого XML-документа. Каждая строка текстового файла содержит несколько (одно или более) слов, разделенных ровно одним пробелом. Создать XML-документ с корневым элементом `root`, элементами первого уровня `line` и элементами второго уровня `word`. Элементы `line` соответствуют строкам исходного файла и не содержат дочерних текстовых узлов, элементы `word` каждого элемента `line` содержат по одному слову из соответствующей строки (слова располагаются в порядке их следования в исходной строке). Элемент `line` должен содержать атрибут `num`, равный порядковому номеру строки в исходном файле, элемент `word` должен содержать атрибут `num`, равный порядковому номеру слова в строке.

LinqXml6. Даны имена существующего текстового файла и создаваемого XML-документа. Каждая строка текстового файла содержит несколько (одно или более) целых чисел, разделенных ровно одним пробелом. Создать XML-документ с корневым элементом `root`, элементами первого уровня `line` и элементами второго уровня `number`. Элементы `line` соответствуют строкам исходного файла и не содержат дочерних текстовых узлов, элементы `number` каждого элемента `line` содержат по одному числу из соответствующей строки (числа располагаются в порядке убывания). Элемент `line` должен содержать атрибут `sum`, равный сумме всех чисел из соответствующей строки.

LinqXml7. Даны имена существующего текстового файла и создаваемого XML-документа. Каждая строка текстового файла содержит несколько (одно или более) целых чисел, разделенных ровно одним пробелом. Создать XML-документ с корневым элементом `root`, элементами первого уровня `line` и элементами второго уровня `sum-positive` и `number-negative`. Элементы `line` соответствуют строкам исходного файла и не содержат дочерних текстовых узлов, элемент `sum-positive` является первым дочерним элементом каждого элемента `line` и содержит сумму всех положительных чисел из соответствующей строки, элементы `number-negative` содержат по одному отрицательному числу из соответствующей строки (числа располагаются в порядке, обратном порядку их следования в исходной строке).

LinqXml8. Даны имена существующего текстового файла и создаваемого XML-документа. Каждая строка текстового файла содержит несколько (одно или более) слов, разделенных ровно одним пробелом. Создать XML-документ с корневым элементом `root`, элементами первого уровня `line`, элементами второго уровня `word` и элементами третьего уровня `char`. Элементы `line` и `word` не содержат дочерних текстовых узлов. Элементы `line` соответствуют строкам исходного файла, элементы `word` каждого элемента `line` соответствуют словам из этой строки (слова располагаются в алфавитном порядке), элементы `char` каждого элемента `word` содержат по одному символу из соответствующего слова (символы располагаются в порядке их следования в слове).

LinqXml9. Даны имена существующего текстового файла и создаваемого XML-документа. Создать XML-документ с корневым элементом `root`, элементами первого уровня `line` и комментариями (комментарии являются дочерними узлами корневого элемента). Если строка текстового файла начинается с текста «comment:», то она (без текста «comment:») добавляется в XML-документ в качестве очередного комментария, в противном случае строка добавляется в качестве дочернего текстового узла в очередной элемент `line`.

LinqXml10. Даны имена существующего текстового файла и создаваемого XML-документа. Создать XML-документ с корневым элементом `root`, элементами первого уровня `line` и инструкциями обработки (инструкции обработки являются дочерними узлами корневого элемента). Если строка текстового файла начинается с текста «data:», то она (без текста «data:») добавляется в XML-документ в качестве данных к очередной инструкции обработки с именем `instr`, в противном случае строка добавляется в качестве дочернего текстового узла в очередной элемент `line`.

Анализ содержимого XML-документа

LinqXml11. Дан XML-документ. Найти все различные имена его элементов и вывести каждое найденное имя вместе с числом его вхождений в документ. Имена элементов выводить в порядке их первого вхождения.
Указание. Использовать метод GroupBy.

LinqXml12. Дан XML-документ, содержащий хотя бы один элемент первого уровня. Найти все различные имена элементов первого уровня и вывести каждое найденное имя вместе с числом его вхождений в документ в качестве имени элемента первого уровня. Имена элементов выводить в алфавитном порядке.

LinqXml13. Дан XML-документ, содержащий хотя бы один атрибут. Вывести все различные имена атрибутов, входящих в документ. Порядок имен атрибутов должен соответствовать порядку их первого вхождения в документ.
Указание. Использовать методы SelectMany и Distinct.

LinqXml14. Дан XML-документ. Найти элементы второго уровня, имеющие дочерний текстовый узел, и вывести количество найденных элементов, а также имя каждого найденного элемента и значение его дочернего текстового узла. Порядок вывода элементов должен соответствовать порядку их следования в документе.

LinqXml15. Дан XML-документ, содержащий хотя бы один элемент первого уровня. Для каждого элемента первого уровня найти количество его потомков, имеющих не менее двух атрибутов, и вывести имя элемента первого уровня и найденное количество его потомков. Элементы выводить в алфавитном порядке их имен, а при совпадении имен — в порядке возрастания найденного количества потомков.

LinqXml16. Дан XML-документ, содержащий хотя бы один элемент первого уровня. Для каждого элемента первого уровня найти суммарное количество атрибутов у его элементов-потомков второго уровня (т. е. элементов, являющихся дочерними элементами его дочерних элементов) и вывести найденное количество атрибутов и имя элемента. Элементы выводить в порядке убывания найденного количества атрибутов, а при совпадении количества атрибутов — в алфавитном порядке имен.

LinqXml17. Дан XML-документ, содержащий хотя бы один текстовый узел. Найти все различные имена элементов, имеющих дочерний текстовый узел, и вывести эти имена, а также значения всех связанных с ними дочерних текстовых узлов. Порядок имен должен соответствовать порядку их первого вхождения в документ; текстовые значения, связанные с каждым именем, выводить в алфавитном порядке.

LinqXml18. Дан XML-документ, содержащий хотя бы один атрибут. Найти все различные имена атрибутов и вывести эти имена, а также все связанные с ними значения (все значения считаются текстовыми). Имена выводить в алфавитном порядке; значения, связанные с каждым именем, выводить в порядке их появления в документе.

LinqXml19. Дан XML-документ, содержащий хотя бы один элемент первого уровня. Для каждого элемента первого уровня найти его дочерние элементы, имеющие максимальное количество потомков (при подсчете числа потомков учитывать также потомки-узлы, не являющиеся элементами). Перебирая элементы первого уровня в порядке их появления в XML-документе, вывести имя элемента, число N — максимальное количество потомков, имеющихся у его дочерних элементов (значение N может быть равно 0), — и имена всех дочерних элементов, имеющих N потомков (имена дочерних элементов выводить в алфавитном порядке; среди этих имен могут быть совпадающие). Если элемент первого уровня не содержит дочерних элементов, то в качестве значения N выводить −1, а в качестве имени дочернего элемента — текст «no child».

LinqXml20. Дан XML-документ, содержащий хотя бы один элемент первого уровня. Для каждого элемента первого уровня найти его элементы-потомки, имеющие максимальное количество атрибутов. Перебирая элементы первого уровня в порядке их появления в XML-документе, вывести имя элемента, число N — максимальное количество атрибутов у его потомков (значение N может быть равно 0) — и имена потомков, имеющих N атрибутов (имена потомков выводить в алфавитном порядке; среди этих имен могут быть совпадающие). Если элемент первого уровня не содержит элементов-потомков, то в качестве значения N выводить −1, а в качестве имени потомка — текст «no child».

Преобразование XML-документа

LinqXml21. Дан XML-документ и строка S. В строке записано имя одного из некорневых элементов исходного документа. Удалить из документа все элементы первого уровня с именем S.

Указание. Применить метод Remove к последовательности Elements(S) корневого элемента.

LinqXml22. Дан XML-документ и строка S. В строке записано имя одного из некорневых элементов исходного документа. Удалить из документа все элементы с именем S.

LinqXml23. Дан XML-документ. Удалить из документа все инструкции обработки.

Указание. Для получения последовательности всех инструкций обработки воспользоваться методом OfType<XProcessingInstruction>.

LinqXml24. Дан XML-документ. Удалить из документа все комментарии, являющиеся узлами первого или второго уровня (т. е. имеющие своим родительским элементом корневой элемент или элемент первого уровня).

LinqXml25. Дан XML-документ. Для всех элементов первого и второго уровня, имеющих более одного атрибута, удалить все их атрибуты.

LinqXml26. Дан XML-документ. Для всех элементов документа удалить все их атрибуты, кроме первого.

Указание. В предикате метода Where, отбирающем все атрибуты элемента, кроме первого, использовать метод PreviousAttribute класса XAttribute.

LinqXml27. Дан XML-документ. Для всех элементов второго уровня удалить все их дочерние узлы, кроме последнего.

LinqXml28. Дан XML-документ. Удалить дочерние текстовые узлы для всех элементов третьего уровня. Если текстовый узел является единственным дочерним узлом элемента, то после его удаления элемент должен быть представлен в виде комбинированного тега.

Указание. Использовать метод OfType<XText>.

LinqXml29. Дан XML-документ. Удалить из документа все элементы первого и второго уровня, не содержащие дочерних узлов.

LinqXml30. Дан XML-документ. Удалить из документа все элементы третьего уровня, представленные комбинированным тегом.

Указание. Использовать свойство IsEmpty класса XElement.

LinqXml31. Дан XML-документ и строки S_1 и S_2. В строке S_1 записано имя одного из элементов исходного документа, строка S_2 содержит допустимое имя элемента XML. После каждого элемента первого уровня с именем S_1 добавить элемент с именем S_2. Атрибуты и потомки добавленного элемента должны совпадать с атрибутами и потомками предшествующего элемента.

Указание. Для каждого элемента S_1 вызвать метод AddAfterSelf с тремя параметрами: строкой S_2 и последовательностями Attributes и Nodes элемента S_1.

LinqXml32. Дан XML-документ и строки S_1 и S_2. В строке S_1 записано имя одного из элементов исходного документа, строка S_2 содержит допустимое имя элемента XML. Перед каждым элементом второго уровня с именем S_1 добавить элемент с именем S_2. Добавленный элемент должен содержать последний атрибут и первый дочерний элемент последующего элемента (если они есть). Если элемент S_1 не имеет дочерних элементов, то добавленный перед ним элемент S_2 должен быть представлен в виде комбинированного тега.

Указание. Использовать методы FirstOrDefault и LastOrDefault.

LinqXml33. Дан XML-документ. Для каждого элемента первого уровня, имеющего атрибуты, добавить в конец его дочерних узлов элемент с именем `attr` и атрибутами, совпадающими с атрибутами обрабатываемого элемента первого уровня, после чего удалить все атрибуты у обрабатываемого элемента. Добавленный элемент `attr` должен быть представлен в виде комбинированного тега.

Указание. Использовать метод ReplaceAttributes, указав в качестве параметра новый дочерний элемент.

LinqXml34. Дан XML-документ. Для каждого элемента первого уровня, имеющего атрибуты, добавить в конец его дочерних узлов элементы с именами, совпадающими с именами его атрибутов, и текстовыми значениями, совпадающими со значениями соответствующих атрибутов, после чего удалить все атрибуты обрабатываемого элемента первого уровня.

Указание. Использовать метод ReplaceAttributes, указав в качестве параметра последовательность элементов, полученную методом Select из последовательности атрибутов.

LinqXml35. Дан XML-документ. Для каждого элемента второго уровня добавить в конец списка его атрибутов атрибут `child-count` со значением, равным количеству всех дочерних узлов этого элемента. Если элемент не имеет дочерних узлов, то атрибут `child-count` должен иметь значение 0.

LinqXml36. Дан XML-документ. Для каждого элемента второго уровня, имеющего потомков, добавить в конец списка его атрибутов атрибут `node-count` со значением, равным количеству узлов-потомков этого элемента (всех уровней).

LinqXml37. Дан XML-документ. Для каждого элемента второго уровня, имеющего потомков, добавить к его текстовому содержимому текстовое содержимое всех элементов-потомков, после чего удалить все его узлы-потомки, кроме дочернего текстового узла.

Указание. Использовать метод SetValue и свойство Value класса XElement.

LinqXml38. Дан XML-документ. Для каждого элемента, кроме корня, изменить его имя, добавив к нему слева исходные имена всех его предков, разделенные символом «-» (дефис). Например, если корневой элемент имеет имя `root`, то элемент `bb` второго уровня, родительский элемент которого имеет имя `aa`, должен получить имя `root-aa-bb`.

Указание. Перебирая все элементы последовательности Descendants корневого элемента, использовать их свойства Parent и Name.

LinqXml39. Дан XML-документ. Для каждого элемента, кроме корня, изменить его имя, добавив к нему слева исходное имя его родительского элемента, дополненное символом «-» (дефис). Например, элемент `cc` третьего уровня, родительский элемент которого имеет имя `bb`, должен получить имя `bb-cc`.

Указание. Организовать перебор последовательности Descendants корневого элемента в обратном порядке (используя метод Reverse).

LinqXml40. Дан XML-документ. Изменить имена атрибутов всех элементов, добавив слева к исходному имени атрибута имя содержащего его элемента, дополненное символом «-» (дефис).

Указание. Так как свойство Name класса XAttribute доступно только для чтения, следует сформировать новую последовательность атрибутов с требуемыми именами и значениями (применяя метод Select к последовательности Attributes), после чего указать ее в качестве параметра метода ReplaceAttributes.

Преобразование типов при обработке XML-документа

LinqXml41. Дан XML-документ. Любой его элемент содержит либо набор дочерних элементов, либо текстовое представление вещественного числа. Добавить к каждому элементу, содержащему дочерние элементы, атрибут `sum`, равный сумме чисел, указанных в дочерних элементах. Сумма округляется до двух дробных знаков, незначащие нули не отображаются. Если ни один из дочерних элементов не содержит текстовое представление числа, то атрибут `sum` должен иметь значение 0.

Указание. Для преобразования текстового представления вещественного числа в само число достаточно выполнить приведение к типу double элемента XML, содержащего это текстовое представление. Для указания числового значения атрибута `sum` достаточно передать в качестве второго параметра конструктора XAttribute вещественное число, округленное требуемым образом (с помощью функции Math.Round с двумя параметрами).

LinqXml42. Дан XML-документ, в котором значения всех атрибутов являются текстовыми представлениями вещественных чисел. Добавить к каждому элементу первого уровня, содержащему дочерние элементы, дочерний элемент `sum`, содержащий текстовое представление суммы атрибутов всех дочерних элементов данного элемента. Сумма округляется до двух дробных знаков, незначащие нули не отображаются. Если ни один из дочерних элементов не содержит атрибутов, то элемент `sum` должен иметь значение 0.

LinqXml43. Дан XML-документ, в котором значения всех атрибутов являются текстовыми представлениями вещественных чисел. Добавить к каждому элементу первого уровня, содержащему дочерние элементы, дочерний элемент `max`, содержащий текстовое представление максимального из значений атрибутов всех элементов-потомков данного элемента. Если ни один из элементов-потомков не содержит атрибутов, то элемент `max` не добавлять.

Указание. Для единообразной обработки двух ситуаций (наличие или отсутствие атрибутов у потомков) можно построить по последовательности атрибутов последовательность числовых значений Nullable-типа double?, применить к ней метод Max и добавить новый элемент `max` с помощью метода SetElementValue, указав в качестве второго параметра результат, возвращенный методом Max. При отсутствии атрибутов у потомков метод Max вернет значение null; в этом случае метод SetElementValue не будет создавать новый элемент.

LinqXml44. Дан XML-документ и строка S. В строке записано имя одного из атрибутов исходного документа; известно, что все атрибуты с указанным именем содержат текстовое представление вещественного числа. Для каждого элемента выполнить следующие действия: перебирая всех его потомков, содержащих атрибут S, найти минимальное значение данного атрибута и записать это значение в новый атрибут `min` обрабатываемого элемента. Если ни один из потомков элемента не содержит атрибут S, то атрибут `min` к этому элементу не добавлять.

Указание. Использовать приведение атрибута Attribute(S) к Nullable-типу double?; если атрибут с указанным именем отсутствует, то будет возвращено значение null. Для создания нового атрибута с найденным минимальным значением использовать метод SetAttributeValue; в случае значения null атрибут создан не будет.

LinqXml45. Дан XML-документ. Для каждого элемента, имеющего атрибуты, добавить в начало его набора дочерних узлов элемент с именем `odd-attr-count` и логическим значением, равным true, если суммарное количество атрибутов данного элемента и всех его элементов-потомков является нечетным, и false в противном случае.

Указание. В качестве параметра конструктора XElement, определяющего значение элемента, следует использовать логическое выражение; это позволит отобразить значение логической константы в соответствии со стандартом XML.

LinqXml46. Дан XML-документ. Для каждого элемента, имеющего дочерние элементы, добавить в конец его набора атрибутов атрибут с именем `odd-node-count` и логическим значением, равным true, если суммарное количество дочерних узлов у всех его дочерних элементов является нечетным, и false в противном случае.

LinqXml47. Дан XML-документ. Для каждого элемента, имеющего хотя бы один дочерний элемент, добавить дочерний элемент с именем `has-comments` и логическим значением, равным true, если данный элемент содержит в числе своих узлов-потомков один или более комментариев, и false в противном случае. Новый элемент добавить после первого имеющегося дочернего элемента.

LinqXml48. Дан XML-документ. Для каждого элемента, имеющего не менее двух дочерних узлов, добавить дочерний элемент с именем `has-instructions` и логическим значением, равным true, если данный элемент содержит в числе своих дочерних узлов одну или более инструкций обработки, и false в противном случае. Новый элемент добавить перед последним имеющимся дочерним узлом.

LinqXml49. Дан XML-документ и строка S. В строке записано имя одного из атрибутов исходного документа; известно, что все атрибуты с указанным именем содержат представление некоторого промежутка времени (в днях, часах, минутах и секундах) в формате, принятом в стандарте XML. Добавить в корневой элемент атрибут `total-time`, равный суммарному значению промежутков времени, указанных во всех атрибутах S исходного документа.

Указание. Использовать приведение объекта Attribute(S) к типу TimeSpan или TimeSpan? (в зависимости от выбранного способа построения последовательности требуемых атрибутов). Для суммирования полученных отрезков времени использовать метод Aggregate и операцию «+» для типа TimeSpan.

LinqXml50. Дан XML-документ. С каждым элементом документа связывается некоторый промежуток времени (в днях, часах, минутах и секундах). Этот промежуток либо явно указывается в атрибуте `time` данного элемента (в формате, принятом в стандарте XML), либо, если данный атрибут отсутствует, считается равным одному дню. Добавить в начало набора дочерних узлов корневого элемента элемент `total-time` со значением, равным суммарному значению промежутков времени, связанных со всеми элементами первого уровня.

Указание. Использовать приведение объекта Attribute к Nullable-типу TimeSpan? и операцию ?? языка C# (бинарную операцию If языка VB.NET).

LinqXml51. Дан XML-документ. Любой его элемент содержит либо набор дочерних элементов, либо текстовое представление некоторой даты, соответствующее стандарту XML. Изменить все элементы, содержащие дату, следующим образом: добавить атрибут `year`, содержащий значение года из исходной даты, и дочерний элемент `day` с текстовым значением, равным значению дня из исходной даты, после чего удалить из обрабатываемого элемента исходную дату.

Указание. Использовать приведение элемента к типу DateTime.

LinqXml52. Дан XML-документ. С каждым элементом документа связывается некоторая дата, определяемая номерами года, месяца и дня. Компоненты этой даты указываются в атрибутах year, month, day. Если некоторые из этих атрибутов отсутствуют, то соответствующие компоненты определяются по умолчанию: 2000 для года, 1 для месяца и 10 для дня. Для каждого элемента добавить в начало его набора дочерних узлов элемент date с текстовым представлением даты, связанной с обрабатываемым элементом (дата записывается в формате, принятом в стандарте XML), и удалить имеющиеся атрибуты, связанные с компонентами даты.

Работа с пространствами имен XML-документа

LinqXml53. Дан XML-документ. В каждом элементе первого уровня определено пространство имен, распространяющееся на все его элементы-потомки. Для каждого элемента первого уровня добавить в конец его набора дочерних узлов элемент с именем namespace и значением, равным пространству имен обрабатываемого элемента первого уровня (пространство имен добавленного элемента должно совпадать с пространством имен его родительского элемента).

LinqXml54. Дан XML-документ, у которого корневой элемент и, возможно, какие-либо другие элементы имеют непустое пространство имен. Связать с пространством имен корневого элемента все элементы первого и второго уровня; для элементов более высоких уровней оставить их прежние пространства имен.

LinqXml55. Дан XML-документ. Преобразовать имена всех элементов второго уровня, удалив из них пространства имен (для элементов других уровней пространства имен не изменять).

LinqXml56. Дан XML-документ. В корневом элементе документа определен единственный префикс пространства имен. Снабдить этим префиксом имена всех элементов первого уровня.

LinqXml57. Дан XML-документ и строки S_1 и S_2, содержащие различные пространства имен. Определить в корневом элементе два префикса пространств имен: префикс x, связанный с S_1, и префикс y, связанный с S_2. Снабдить префиксом x все элементы первого уровня, а префиксом y — все элементы второго и последующих уровней.

LinqXml58. Дан XML-документ и строка S, содержащая некоторое пространство имен. Определить в корневом элементе префикс node, связанный с пространством имен, заданным в строке S, и добавить в каждый элемент первого уровня два атрибута: атрибут node:count со значением, равным количеству потомков-узлов для данного элемента, и атрибут xml:count со значением, равным количеству потомков-элементов для данного элемента (xml — префикс пространства имен XML).

Указание. Использовать свойство Xml класса XNamespace.

LinqXml59. Дан XML-документ. В каждом из элементов первого уровня определен единственный префикс пространства имен, причем известно, что все атрибуты с этим префиксом имеют целочисленные значения. Для каждого элемента первого уровня и его элементов-потомков удвоить значения всех атрибутов с префиксом пространства имен, определенным в этом элементе.

LinqXml60. Дан XML-документ, корневой элемент которого содержит определения двух префиксов пространств имен с именами x и y. Эти префиксы используются далее в именах некоторых элементов (у атрибутов префиксы отсутствуют). Удалить определение префикса y и для всех элементов, снабженных этим префиксом, заменить его на префикс x, а для всех элементов, снабженных префиксом x, заменить его на префикс xml пространства имен XML.

Дополнительные задания на обработку XML-документа

Во всех заданиях данной подгруппы предполагается, что в корневом элементе исходного XML-документа определено некоторое пространство имен, распространяющееся на все его элементы-потомки. Это же пространство имен необходимо использовать для имен всех элементов преобразованного документа. Префиксы пространств имен в заданиях данной подгруппы не используются.

LinqXml61. Дан XML-документ с информацией о клиентах фитнес-центра. Образец элемента первого уровня:

```
<record>
  <id>10</id>
  <date>2000-05-01T00:00:00</date>
  <time>PT5H13M</time>
</record>
```

Здесь id — код клиента (целое число), date — дата с информацией о годе и месяце, time — продолжительность занятий (в часах и минутах) данного клиента в течение указанного месяца. Преобразовать документ, изменив элементы первого уровня следующим образом:

```
<time id="10" year="2000" month="5">PT5H13M</time>
```

Порядок следования элементов первого уровня не изменять.

LinqXml62. Дан XML-документ с информацией о клиентах фитнес-центра. Образец элемента первого уровня (смысл данных тот же, что и в LinqXml61):

```
<time year="2000" month="5" id="10">PT5H13M</time>
```

Преобразовать документ, изменив элементы первого уровня следующим образом:

```
<id10 date="2000-05-01T00:00:00">313</id10>
```

Имя элемента должно иметь префикс id, после которого указывается код клиента; в значении атрибута date день должен быть равен 1, а время должно быть нулевым. Значение элемента равно продолжительности занятий клиента в данном месяце, переведенной в минуты. Элементы должны быть отсортированы по возрастанию кода клиента, а для одинаковых значений кода — по возрастанию даты.

LinqXml63. Дан XML-документ с информацией о клиентах фитнес-центра. Образец элемента первого уровня (смысл данных тот же, что и в LinqXml61):

```
<record date="2000-05-01T00:00:00" id="10" time="PT5H13M" />
```

Преобразовать документ, выполнив группировку данных по кодам клиентов и изменив элементы первого уровня следующим образом:

```
<client id="10">
  <time year="2000" month="5">PT5H13M</time>
  ...
</client>
```

Элементы первого уровня должны быть отсортированы по возрастанию кода клиента, их дочерние элементы — по возрастанию номера года, а для одинаковых значений года — по возрастанию номера месяца.

LinqXml64. Дан XML-документ с информацией о клиентах фитнес-центра. Образец элемента первого уровня (смысл данных тот же, что и в LinqXml61):

```
<client id="10">
  <date>2000-05-01T00:00:00</date>
  <time>PT5H13M</time>
</client>
```

Преобразовать документ, сгруппировав данные по годам, а в пределах каждого года — по месяцам. Изменить элементы первого уровня следующим образом:

```
<y2000>
  <m5>
    <client id="10" time="313" />
    ...
  </m5>
  ...
</y2000>
```

Имя элемента первого уровня должно иметь префикс y, после которого указывается номер года; имя элемента второго уровня должно иметь префикс m, после которого указывается номер месяца. Атрибут time должен содержать продолжительность занятий в минутах. Элементы первого уровня должны быть отсортированы по убыванию номера года, их дочерние элементы — по возрастанию номера месяца. Элементы третьего уровня, имеющие общего родителя, должны быть отсортированы по возрастанию кодов клиентов.

LinqXml65. Дан XML-документ с информацией о клиентах фитнес-центра. Образец элемента первого уровня (смысл данных тот же, что и в LinqXml61, данные сгруппированы по кодам клиентов; коды клиентов, снабженные префиксом id, указываются в качестве имен элементов первого уровня):

```
<id10>
  <info>
    <date>2000-05-01T00:00:00</date>
    <time>PT5H13M</time>
  </info>
  ...
</id10>
```

Преобразовать документ, сгруппировав данные по годам и изменив элементы первого уровня следующим образом:

```
<year value="2000">
  <total-time id="10">860</total-time>
  ...
</year>
```

Значение элемента второго уровня должно быть равно общей продолжительности занятий (в минутах) клиента с указанным кодом в течение указанного года. Элементы первого уровня должны быть отсортированы по возрастанию номера года, их дочерние элементы — по возрастанию кодов клиентов.

LinqXml66. Дан XML-документ с информацией о клиентах фитнес-центра. Образец элемента первого уровня (смысл данных тот же, что и в LinqXml61, данные сгруппированы по кодам клиентов):

```
<client id="10">
  <info date="2000-05-01T00:00:00" time="PT5H13M" />
  ...
</client>
```

Преобразовать документ, сгруппировав данные по годам и месяцам и оставив сведения только о тех месяцах, в которых посещали занятия не менее трех клиентов. Изменить элементы первого уровня следующим образом:

```
<d2000-5 total-time="956" client-count="3">
  <id10 time="313" />
  ...
</d2000-5>
```

Имя элемента первого уровня должно иметь префикс d, после которого указывается номер года и, через дефис, номер месяца (незначащие нули не отображаются). Имя элемента второго уровня должно иметь префикс id, после которого указывается код клиента. Атрибут total-time должен содержать суммарную продолжительность занятий (в минутах) всех клиентов в данном месяце, атрибут client-count — количество клиентов, занимавшихся в этом месяце. Атрибут time для элементов второго уровня должен содержать продолжительность занятий (в минутах) клиента с указанным кодом в данном месяце. Элементы первого уровня должны быть отсортированы по возрастанию номера года, а для одинаковых номеров года — по возрастанию номера месяца; их дочерние элементы должны быть отсортированы по возрастанию кодов клиентов.

LinqXml67. Дан XML-документ с информацией о клиентах фитнес-центра. Образец элемента первого уровня (смысл данных тот же, что и в LinqXml61):

```
<client id="10" time="PT5H13M">
  <year>2000</year>
  <month>5</month>
</client>
```

Преобразовать документ, сгруппировав данные по годам и месяцам и изменив элементы первого уровня следующим образом:

```
<y2000>
  <m1 total-time="0" client-count="0" />
  ...
  <m5 total-time="956" client-count="3" />
  ...
</y2000>
```

Имя элемента первого уровня должно иметь префикс y, после которого указывается номер года; имя элемента второго уровня должно иметь префикс m, после которого указывается номер месяца. Атрибут total-time должен содержать суммарную продолжительность занятий (в минутах) всех клиентов в данном месяце, атрибут client-count — количество клиентов, занимавшихся в этом месяце. Каждый элемент первого уровня должен содержать элементы второго уровня, соответствующие всем месяцам года; если в каком-либо месяце занятия не проводились, то атрибуты для этого месяца должны иметь нулевые значения. Элементы первого уровня должны быть отсортированы по возрастанию номера года, их дочерние элементы — по возрастанию номера месяца.

Указание. Для эффективного формирования последовательностей, связанных со всеми месяцами, использовать вспомогательную последовательность Enumerable.Range(1, 12) и метод GroupJoin.

LinqXml68. Дан XML-документ с информацией о ценах автозаправочных станций на бензин. Образец элемента первого уровня:

```
<record>
  <company>Лидер</company>
  <street>ул.Чехова</street>
  <brand>92</brand>
  <price>2200</price>
</record>
```

Здесь street — название улицы, company — название компании (названия улиц и компаний не содержат пробелов и являются допустимыми именами XML), brand — марка бензина (числа 92, 95 или 98), price — цена 1 литра бензина в копейках (целое число). Каждая компания имеет не более одной АЗС на каждой улице, цены на разных АЗС одной и той же компании могут различаться. Преобразовать документ, изменив элементы первого уровня следующим образом:

```
<station company="Лидер" street="ул.Чехова">
  <info>
    <brand>92</brand>
    <price>2200</price>
  </info>
</station>
```

Порядок следования элементов первого уровня не изменять.

LinqXml69. Дан XML-документ с информацией о ценах автозаправочных станций на бензин. Образец элемента первого уровня (смысл данных тот же, что и в LinqXml68):

```
<station company="Лидер">
  <info street="ул.Чехова">
    <brand>92</brand>
    <price>2200</price>
  </info>
</station>
```

Преобразовать документ, изменив элементы первого уровня следующим образом:

```
<b92 company="Лидер" street="ул.Чехова" price="2200" />
```

Имя элемента должно иметь префикс b, после которого указывается марка бензина. Элементы представляются комбинированными тегами и должны быть отсортированы по возрастанию марок бензина, для одинаковых марок — в алфавитном порядке названий компании, а для одинаковых компаний — в алфавитном порядке названий улиц.

LinqXml70. Дан XML-документ с информацией о ценах автозаправочных станций на бензин. Образец элемента первого уровня (смысл данных тот же, что и в LinqXml68):

```
<station brand="98" price="2850">
  <company>Лидер</company>
  <street>ул.Авиаторов</street>
</station>
```

Преобразовать документ, выполнив группировку данных по названиям компаний, а в пределах каждой компании — по маркам бензина. Изменить элементы первого уровня следующим образом:

```
<company name="Лидер">
  <brand value="98">
    <price street="ул.Авиаторов">2850</price>
    ...
  </brand>
  ...
</company>
```

Элементы первого уровня должны быть отсортированы в алфавитном порядке названий компаний, а их дочерние элементы — по убыванию марок бензина. Элементы третьего уровня, имеющие общего родителя, должны быть отсортированы в алфавитном порядке названий улиц.

LinqXml71. Дан XML-документ с информацией о ценах автозаправочных станций на бензин. Образец элемента первого уровня (смысл данных тот же, что и в LinqXml68):

```
<station street="ул.Авиаторов" company="Лидер">
  <info brand="98" price="2850" />
</station>
```

Преобразовать документ, выполнив группировку данных по маркам бензина, а в пределах каждой марки — по ценам 1 литра бензина. Изменить элементы первого уровня следующим образом:

```
<b98>
  <p2850>
    <info street="ул.Авиаторов" company="Лидер" />
    ...
  </p2850>
  ...
</b98>
```

Имя элемента первого уровня должно иметь префикс b, после которого указывается марка бензина; имя элемента второго уровня должно иметь префикс p, после которого указывается цена 1 литра бензина. Элементы первого уровня должны быть отсортированы по убыванию марок бензина, а их дочерние элементы — по убыванию цен. Элементы третьего уровня, имеющие общего родителя, должны быть отсортированы в алфавитном порядке названий улиц, а для одинаковых улиц — в алфавитном порядке названий компаний.

LinqXml72. Дан XML-документ с информацией о ценах автозаправочных станций на бензин. Образец элемента первого уровня (смысл данных тот же, что и в LinqXml68, данные сгруппированы по названиям компаний; названия компаний указываются в качестве имен элементов первого уровня):

```
<Лидер>
  <price street="ул.Чехова" brand="92">2200</price>
  ...
</Лидер>
```

Преобразовать документ, выполнив группировку данных по названиям улиц, а в пределах каждой улицы — по маркам бензина. Изменить элементы первого уровня следующим образом:

```
<ул.Чехова>
  <b92>
    <min-price company="Премьер-нефть">2050</min-price>
    ...
  </b92>
  ...
</ул.Чехова>
```

Имя элемента первого уровня совпадает с названием улицы, имя элемента второго уровня должно иметь префикс b, после которого указывается марка бензина. Значение элемента третьего уровня равно минимальной цене бензина данной марки на данной улице, его атрибут company содержит название компании, на АЗС которой предлагается минимальная цена. Элементы первого уровня должны быть отсортированы в алфавитном порядке названий улиц, а их дочерние элементы — по возрастанию марок бензина. Если имеется несколько элементов третьего уровня, имеющих общего родителя (это означает, что на одной улице имеется несколько АЗС, на которых бензин данной марки имеет минимальную цену), то эти элементы должны быть отсортированы в алфавитном порядке названий компаний.

LinqXml73. Дан XML-документ с информацией о ценах автозаправочных станций на бензин. Образец элемента первого уровня (смысл данных тот же, что и в LinqXml68, данные сгруппированы по названиям улиц; названия улиц указываются в качестве имен элементов первого уровня):

```
<ул.Чехова>
  <company name="Лидер">
    <brand>92</brand>
    <price>2200</price>
  </company>
  ...
</ул.Чехова>
```

Преобразовать документ, сгруппировав данные по названиям компаний и названиям улиц и оставив сведения только о тех АЗС, в которых предлагаются не менее двух марок бензина.

Изменить элементы первого уровня следующим образом:

```
<Лидер_ул.Чехова brand-count="2">
  <b92 price="2200" />
  <b95 price="2450" />
</Лидер_ул.Чехова>
```

Имя элемента первого уровня содержит название компании, после которого следует символ подчеркивания и название улицы; имя элемента второго уровня должно иметь префикс b, после которого указывается марка бензина. Атрибут brand-count должен содержать количество марок бензина, предлагаемых на данной АЗС. Элементы первого уровня должны быть отсортированы в алфавитном порядке названий компаний, а для одинаковых названий компаний — в алфавитном порядке названий улиц; их дочерние элементы должны быть отсортированы по возрастанию марок бензина.

LinqXml74. Дан XML-документ с информацией о ценах автозаправочных станций на бензин. Образец элемента первого уровня (смысл данных тот же, что и в LinqXml68, марки бензина, снабженные префиксом brand, указываются в качестве имен элементов первого уровня; атрибут station содержит названия улицы и компании, разделенные символом подчеркивания):

```
<brand92 station="ул.Чехова_Лидер" price="2200" />
```

Преобразовать документ, сгруппировав данные по названиям компаний и изменив элементы первого уровня следующим образом:

```
<Лидер>
  <ул.Садовая brand92="0" brand95="0" brand98="0" />
  <ул.Чехова brand92="2200" brand95="2450" brand98="0" />
  ...
</Лидер>
```

Имя элемента первого уровня совпадает с названием компании, имя элемента второго уровня совпадает с названием улицы. Атрибуты элементов второго уровня имеют префикс brand, после которого указывается марка бензина; их значением является цена 1 литра бензина указанной марки или число 0, если на данной АЗС бензин указанной марки не предлагается. Для каждой компании должна выводиться информация по каждой улице, имеющейся в исходном документе, даже если на этой улице отсутствует АЗС данной компании (в этом случае значения всех атрибутов brand должны быть равны 0). Элементы первого уровня должны быть отсортированы в алфавитном порядке названий компаний, а их дочерние элементы — в алфавитном порядке названий улиц.

LinqXml75. Дан XML-документ с информацией о ценах автозаправочных станций на бензин. Образец элемента первого уровня (смысл данных тот же, что и в LinqXml68, названия компании и улицы, разделенные символом подчеркивания, указываются в качестве имен элементов первого уровня):

```
<Лидер_ул.Чехова>
  <brand>92</brand>
  <price>2200</price>
</Лидер_ул.Чехова>
```

Преобразовать документ, сгруппировав данные по названиям улиц и изменив элементы первого уровня следующим образом:

```
<ул.Чехова>
  <brand98 station-count="0">0</brand98>
  <brand95 station-count="0">0</brand95>
  <brand92 station-count="3">2255</brand92>
</ул.Чехова>
```

Имя элемента первого уровня совпадает с названием улицы, имя элемента второго уровня имеет префикс brand, после которого указывается марка бензина. Атрибут station-count равен количеству АЗС, расположенных на данной улице и предлагающих бензин данной марки; значением элемента второго уровня является средняя цена 1 литра бензина данной марки по всем АЗС, расположенным на данной улице. Средняя цена находится по следующей формуле: «*суммарная цена по всем станциям*»/«*число станций*», где операция «/» обозначает целочисленное деление. Если на данной улице отсутствуют АЗС, предлагающие бензин данной марки, то значение соответствующего элемента второго уровня и значение его атрибута station-count должны быть равны 0. Элементы первого уровня должны быть отсортированы в алфавитном порядке названий улиц, а их дочерние элементы — по убыванию марок бензина.

LinqXml76. Дан XML-документ с информацией о задолженности по оплате коммунальных услуг. Образец элемента первого уровня:

```
<record>
  <house>12</house>
  <flat>129</flat>
  <name>Сергеев Т.М.</name>
  <debt>1833.32</debt>
</record>
```

Здесь house — номер дома (целое число), flat — номер квартиры (целое число), name — фамилия и инициалы жильца (инициалы не содержат пробелов и отделяются от фамилии одним пробелом), debt — размер задолженности в виде дробного числа: целая часть — рубли, дробная часть — копейки (незначащие нули не указываются). Все дома являются 144-квартирными, имеют 9 этажей и 4 подъезда. Преобразовать документ, изменив элементы первого уровня следующим образом:

```
<debt house="12" flat="129">
  <name>Сергеев Т.М.</name>
  <value>1833.32</value>
</debt>
```

Порядок следования элементов первого уровня не изменять.

LinqXml77. Дан XML-документ с информацией о задолженности по оплате коммунальных услуг. Образец элемента первого уровня (смысл данных тот же, что и в LinqXml76):

```
<debt house="12" flat="129" name="Сергеев Т.М.">1833.32</debt>
```

Преобразовать документ, изменив элементы первого уровня следующим образом:

```
<address12-129 name="Сергеев Т.М." debt="1833.32" />
```

Имя элемента должно иметь префикс address, после которого указывается номер дома и, через дефис, номер квартиры. Элементы представляются комбинированными тегами и должны быть отсортированы по возрастанию номеров домов, а для одинаковых номеров домов — по возрастанию номеров квартир.

LinqXml78. Дан XML-документ с информацией о задолженности по оплате коммунальных услуг. Образец элемента первого уровня (смысл данных тот же, что и в LinqXml76):

```
<debt house="12" flat="23">
  <name>Иванов А.В.</name>
  <value>1245.64</value>
</debt>
```

Преобразовать документ, выполнив группировку данных по номеру дома, а в пределах каждого дома — по номеру подъ-

езда. Изменить элементы первого уровня следующим образом:

```
<house number="12">
  <entrance number="1">
    <debt name="Иванов А.В." flat="23">1245.64</debt>
    ...
  </entrance>
  ...
</house>
```

Элементы первого уровня должны быть отсортированы по возрастанию номеров домов, а их дочерние элементы — по возрастанию номеров подъездов. Элементы третьего уровня, имеющие общего родителя, должны быть отсортированы по убыванию размера задолженности (предполагается, что размеры всех задолженностей являются различными).

LinqXml79. Дан XML-документ с информацией о задолженности по оплате коммунальных услуг. Образец элемента первого уровня (смысл данных тот же, что и в LinqXml76):

```
<house value="12">
  <flat value="129" />
  <name value="Сергеев Т.М." />
  <debt value="1833.32" />
</house>
```

Преобразовать документ, выполнив группировку данных по номеру дома, а в пределах каждого дома — по номеру этажа. Изменить элементы первого уровня следующим образом:

```
<house12>
  <floor6>
    <Сергеев_Т.М. flat="129" debt="1833.32" />
    ...
  </floor6>
  ...
</house12>
```

Имя элемента первого уровня должно иметь префикс `house`, после которого указывается номер дома; имя элемента второго уровня должно иметь префикс `floor`, после которого указывается номер этажа. Имя элемента третьего уровня совпадает с фамилией и инициалами жильца; фамилия отделяется от инициалов символом подчеркивания. Элементы первого уровня должны быть отсортированы по возрастанию номеров домов, а их дочерние элементы — по убыванию номеров этажей. Элементы третьего уровня, имеющие общего родителя, должны быть отсортированы в алфавитном порядке фамилий и инициалов жильцов.

LinqXml80. Дан XML-документ с информацией о задолженности по оплате коммунальных услуг. Образец элемента первого уровня (смысл данных тот же, что и в LinqXml76, данные сгруппированы по номерам домов; в качестве элементов второго уровня указываются фамилии и инициалы жильцов, фамилия отделяется от инициалов символом подчеркивания):

```
<house number="12">
  <Иванов_А.В.>
    <flat value="23" />
    <debt value="1245.64" />
  </Иванов_А.В.>
  ...
</house>
```

Преобразовать документ, сохранив группировку данных по номеру дома, выполнив в пределах каждого дома группировку по номеру подъезда и изменив элементы первого уровня следующим образом:

```
<house12>
  <entrance1 total-debt="2493.38" count="3">
    <flat23 name="Иванов А.В." />
    ...
  </entrance1>
  ...
</house12>
```

Имя элемента первого уровня должно иметь префикс `house`, после которого указывается номер дома, имя элемента второго уровня должно иметь префикс `entrance`, после которого указывается номер подъезда, имя элемента третьего уровня должно иметь префикс `flat`, после которого указывается номер квартиры. Атрибут `total-debt` равен суммарной задолженности жильцов данного подъезда (значение задолженности должно округляться до двух дробных знаков, незначащие нули не отображаются), атрибут `count` равен количеству задолжников в данном подъезде. Элементы первого уровня должны быть отсортированы по возрастанию номеров домов, а их дочерние элементы — по возрастанию номеров подъездов. Элементы третьего уровня, имеющие общего родителя, должны быть отсортированы по возрастанию номеров квартир. Подъезды, в которых отсутствуют задолжники, не отображаются.

LinqXml81. Дан XML-документ с информацией о задолженности по оплате коммунальных услуг. Образец элемента первого уровня (смысл данных тот же, что и в LinqXml76, данные сгруппированы по номерам домов; в качестве имен элементов первого уровня указываются номера домов, снабженные префиксом `house`, а в качестве имен элементов второго уровня — номера квартир, снабженные префиксом `flat`):

```
<house12>
  <flat23 name="Иванов А.В." debt="1245.64" />
  ...
</house12>
```

Преобразовать документ, сохранив группировку данных по номеру дома, выполнив в пределах каждого дома группировку по номеру подъезда и оставив сведения только о тех жильцах, размер задолженности которых не меньше среднего размера задолженности по данному подъезду. Изменить элементы первого уровня следующим образом:

```
<house number="12">
  <entrance number="1" count="4" avr-debt="1136">
    <debt flat="23" name="Иванов А.В.">1245.64</debt>
    <debt flat="28" name="Сидоров П.К.">1383.27</debt>
  </entrance>
  ...
</house>
```

Атрибут `count` равен количеству задолжников в данном подъезде, атрибут `avr-debt` определяет среднюю задолженность по данному подъезду в рублях (целое число), вычисленную по следующей формуле: «*суммарная задолженность в копейках*»/(«*количество задолжников*»*100) (символ «/» обозначает операцию целочисленного деления). Элементы третьего уровня содержат сведения о тех жильцах, размер задолженности которых не меньше величины `avr-debt` для данного подъезда. Элементы первого уровня должны быть отсортированы по возрастанию номеров домов, а их дочерние элементы — по возрастанию номеров подъездов. Элементы третьего уровня, имеющие общего родителя, должны быть отсортированы по возрастанию номеров квартир. Подъезды, в которых отсутствуют задолжники, не отображаются.

LinqXml82. Дан XML-документ с информацией о задолженности по оплате коммунальных услуг. Образец элемента перво-

го уровня (смысл данных тот же, что и в LinqXml76, в качестве имени элемента первого уровня указываются номера дома и квартиры, разделенные символом «-» (дефис) и снабженные префиксом addr, а в качестве значения этого элемента указывается размер задолженности для данной квартиры):

```
<addr12-23>1245.64</addr12-23>
```

Преобразовать документ, выполнив группировку данных по номеру дома, а в пределах каждого дома — по номеру этажа. Изменить элементы первого уровня следующим образом:

```
<house12>
  <floor1 count="0" total-debt="0" />
  ...
  <floor6 count="1" total-debt="1245.64" />
  ...
  <floor9 count="3" total-debt="3142.7" />
</house12>
```

Имя элемента первого уровня должно иметь префикс house, после которого указывается номер дома, имя элемента второго уровня должно иметь префикс floor, после которого указывается номер этажа. Атрибут count равен числу задолжников на данном этаже, атрибут total-debt определяет суммарную задолженность по данному этажу, округленную до двух дробных знаков (незначащие нули не отображаются). Если на данном этаже отсутствуют задолжники, то для соответствующего элемента второго уровня значения атрибутов count и total-debt должны быть равны 0. Элементы первого уровня должны быть отсортированы по возрастанию номеров домов, а их дочерние элементы — по возрастанию номеров этажей.

LinqXml83. Дан XML-документ с информацией об оценках учащихся по различным предметам. Образец элемента первого уровня:

```
<record>
  <class>9</class>
  <name>Степанова Д.Б.</name>
  <subject>Физика</subject>
  <mark>4</mark>
</record>
```

Здесь class — номер класса (целое число от 7 до 11), name — фамилия и инициалы учащегося (инициалы не содержат пробелов и отделяются от фамилии одним пробелом), subject — название предмета, не содержащее пробелов, mark — оценка (целое число в диапазоне от 2 до 5). Полных однофамильцев (с совпадающей фамилией и инициалами) среди учащихся нет. Преобразовать документ, изменив элементы первого уровня следующим образом:

```
<mark subject="Физика">
  <name class="9">Степанова Д.Б.</name>
  <value>4</value>
</mark>
```

Порядок следования элементов первого уровня не изменять.

LinqXml84. Дан XML-документ с информацией об оценках учащихся по различным предметам. Образец элемента первого уровня (смысл данных тот же, что и в LinqXml83):

```
<pupil class="9" name="Степанова Д.Б.">
  <subject>Физика</subject>
  <mark>4</mark>
</pupil>
```

Преобразовать документ, изменив элементы первого уровня следующим образом:

```
<class9 name="Степанова Д.Б." subject="Физика">4</class9>
```

Имя элемента должно иметь префикс class, после которого указывается номер класса. Элементы должны быть отсортированы по возрастанию номеров классов, для одинаковых номеров классов — в алфавитном порядке фамилий и инициалов учащихся, для каждого учащегося — в алфавитном порядке названий предметов, а для одинаковых предметов — по возрастанию оценок.

LinqXml85. Дан XML-документ с информацией об оценках учащихся по различным предметам. Образец элемента первого уровня (смысл данных тот же, что и в LinqXml83):

```
<info class="9" name="Степанова Д.Б." subject="Физика" mark="4" />
```

Преобразовать документ, выполнив группировку данных по номеру класса, в пределах каждого класса — по учащимся, а для каждого учащегося — по предметам. Изменить элементы первого уровня следующим образом:

```
<class number="9">
  <pupil name="Степанова Д.Б.">
    <subject name="Физика">
      <mark>4</mark>
      ...
    </subject>
    ...
  </pupil>
  ...
</class>
```

Элементы первого уровня должны быть отсортированы по возрастанию номеров классов, а их дочерние элементы — в алфавитном порядке фамилий и инициалов учащихся. Элементы третьего уровня, имеющие общего родителя, должны быть отсортированы в алфавитном порядке названий предметов, а элементы четвертого уровня, имеющие общего родителя, должны быть отсортированы по убыванию оценок.

LinqXml86. Дан XML-документ с информацией об оценках учащихся по различным предметам. Образец элемента первого уровня (смысл данных тот же, что и в LinqXml83):

```
<pupil name="Степанова Д.Б." class="9">
  <info mark="4" subject="Физика" />
</pupil>
```

Преобразовать документ, выполнив группировку данных по учащимся и изменив элементы первого уровня следующим образом:

```
<Степанова_Д.Б. class="9">
  <mark4 subject="Физика" />
  ...
</Степанова_Д.Б.>
```

Имя элемента первого уровня совпадает с фамилией и инициалами учащегося (пробел между фамилией и инициалами заменяется символом подчеркивания), имя элемента второго уровня должно иметь префикс mark, после которого указывается оценка. Элементы первого уровня должны быть отсортированы в алфавитном порядке фамилий и инициалов учащихся, их дочерние элементы — по убыванию оценок, а для одинаковых оценок — в алфавитном порядке названий предметов.

LinqXml87. Дан XML-документ с информацией об оценках учащихся по различным предметам. Образец элемента первого уровня (смысл данных тот же, что и в LinqXml83, данные сгруппированы по учащимся):

```
<pupil name="Степанова Д.Б." class="9">
  <mark subject="Физика">4</mark>
  ...
</pupil>
```

Преобразовать документ, выполнив группировку данных по названиям предметов и изменив элементы первого уровня следующим образом:

```
<Физика>
  <class9>
    <mark-count>4</mark-count>
    <avr-mark>4.1</avr-mark>
  </class9>
  ...
</Физика>
```

Имя элемента первого уровня совпадает с названием предмета, имя элемента второго уровня должно иметь префикс `class`, после которого указывается номер класса. Значение элемента `mark-count` равно количеству оценок по данному предмету, выставленных в данном классе; значение элемента `avr-mark` равно среднему значению этих оценок, найденному по следующий формуле: 10*«*сумма оценок*»/«*количество оценок*»*0.1 (символ «/» обозначает операцию целочисленного деления, полученное значение должно содержать не более одного дробного знака, незначащие нули не отображаются). Элементы первого уровня должны быть отсортированы в алфавитном порядке названий предметов, а их дочерние элементы — по возрастанию номеров классов.

LinqXml88. Дан XML-документ с информацией об оценках учащихся по различным предметам. Образец элемента первого уровня (смысл данных тот же, что и в LinqXml83, данные сгруппированы по классам):

```
<class number="9">
  <pupil name="Степанова Д.Б." subject="Физика" mark="4" />
  ...
</class>
```

Преобразовать документ, выполнив группировку данных по предметам и оставив сведения только о тех учащихся, которые получили по данному предмету более двух оценок. Изменить элементы первого уровня следующим образом:

```
<subject name="Физика">
  <pupil class="9" name="Степанова Д.Б." m1="4" m2="3" m3="3" />
  ...
</subject>
```

Оценки каждого учащегося по данному предмету указываются в атрибутах, имеющих префикс `m`, после которого следует порядковый номер оценки. Элементы первого уровня должны быть отсортированы в алфавитном порядке названий предметов, их дочерние элементы — по возрастанию номеров классов, а для одинаковых классов — в алфавитном порядке фамилий и инициалов учащихся. Оценки для каждого учащегося должны располагаться в порядке убывания. Если для некоторого предмета не найдены учащиеся, имеющие по нему более двух оценок, то соответствующий элемент первого уровня должен быть представлен комбинированным тегом, например:

```
<subject name="Химия" />
```

LinqXml89. Дан XML-документ с информацией об оценках учащихся по различным предметам. Образец элемента первого уровня (смысл данных тот же, что и в LinqXml83, в качестве имен элементов первого уровня указываются фамилии и инициалы учащихся; при этом пробел между фамилией и инициалами заменяется символом подчеркивания):

```
<Петров_С.Н. class="11" subject="Физика">4</Петров_С.Н.>
```

Преобразовать документ, выполнив группировку данных по предметам, а для каждого предмета — по классам. Изменить элементы первого уровня следующим образом:

```
<Физика>
  <class7 pupil-count="0" mark-count="0" />
  ...
  <class11 pupil-count="3" mark-count="5" />
</Физика>
```

Имя элемента первого уровня совпадает с названием предмета, имя элемента второго уровня должно иметь префикс `class`, после которого указывается номер класса. Значение атрибута `pupil-count` равно количеству учащихся данного класса, имеющих хотя бы одну оценку по данному предмету, значение атрибута `mark-count` равно количеству оценок по данному предмету в данном классе. Для каждого предмета должна быть выведена информация по каждому классу (от 7 до 11); если в некотором классе по данному предмету не было опрошено ни одного учащегося, то атрибуты `pupil-count` и `mark-count` должны быть равны 0. Элементы первого уровня должны быть отсортированы в алфавитном порядке названий предметов, а их дочерние элементы — по возрастанию номеров классов.

LinqXml90. Дан XML-документ с информацией об оценках учащихся по различным предметам. Образец элемента первого уровня (смысл данных тот же, что и в LinqXml84; в качестве имен элементов первого уровня указываются фамилии с инициалами учащихся и номера классов; между фамилией и инициалами указывается символ подчеркивания, а между инициалами и номером класса — дефис):

```
<Степанова_Д.Б.-9 subject="Физика" mark="4" />
```

Преобразовать документ, сгруппировав данные по номерам классов, а для каждого класса — по учащимся. Изменить элементы первого уровня следующим образом:

```
<class9>
  <Степанова_Д.Б.>
    <История count="0">0</История>
    ...
    <Физика count="3">3.3</Физика>
  </Степанова_Д.Б.>
  ...
</class9>
```

Имя элемента первого уровня должно иметь префикс `class`, после которого указывается номер класса, имя элемента второго уровня совпадает с фамилией и инициалами учащегося, между которыми указывается символ подчеркивания. Имя элемента третьего уровня совпадает с названием предмета. Значение атрибута `count` равно количеству оценок по данному предмету, полученных данным учащимся. Значение элемента третьего уровня равно средней оценке по данному предмету для данного учащегося; средняя оценка вычисляется по следующей формуле: 10*«*сумма оценок*»/«*количество оценок*»*0.1 (символ «/» обозначает операцию целочисленного деления, полученное значение должно содержать не более одного дробного знака, незначащие нули не отображаются). Для каждого учащегося должна быть выведена информация по каждому предмету, входящему в исходный XML-документ; если по некоторому предмету учащийся не имеет оценок, то значение соответствующего элемента третьего уровня и значение его атрибута `count` должны быть равны 0. Элементы первого уровня должны быть отсортированы по возрастанию номеров классов, а их дочерние элементы — в алфавитном порядке фамилий учащихся. Элементы третьего уровня, имеющие общего родителя, должны быть отсортированы в алфавитном порядке названий предметов.