Содержание

Предисловие

Книга, предлагаемая вашему вниманию, представляет собой практическое введение в технологию LINQ платформы .NET. Эта технология входит в состав платформы .NET Framework, начиная с версии 3.5; она предоставляет программисту, использующему языки C# или Visual Basic .NET, средства высокого уровня для обработки различных наборов данных. С каждой категорией данных связываются особые интерфейсы (LINQ API), в частности интерфейс LINQ to Objects предназначен для работы с локальными коллекциями, LINQ to XML – для обработки данных в формате XML, интерфейсы LINQ to SQL и LINQ to Entities (Entity Framework) – для работы с удаленными хранилищами данных. При этом базовый набор средств обработки (*запросов LINQ*) остается неизменным для любого интерфейса, обеспечивая универсальный характер технологии LINQ.

К преимуществам технологии LINQ, помимо ее универсального характера, можно отнести краткость и наглядность программ, полученных с ее применением, а также простоту реализации достаточно сложных алгоритмов обработки данных. В то же время, для того чтобы получить эффективные реализации алгоритмов, основанных на применении технологии LINQ, необходимо владеть всем арсеналом доступных запросов и других компонентов LINQ API, выбирая и комбинируя их способом, наиболее подходящим для каждого конкретного случая.

Один из подходов к практическому освоению технологии LINQ (как и любой технологии программирования) состоит в решении набора учебных задач, связанных с различными возможностями этой технологии. Между тем среди книг, посвященных технологии LINQ и включающих фундаментальные руководства [4–5] и справочники [3] (следует также отметить подробные разделы о технологии LINQ в книгах [1–2]), подобные задачники отсутствуют. Предлагаемая книга призвана восполнить этот пробел.

В книге содержатся 250 учебных задач, посвященных интерфейсам LINQ to Object и LINQ to XML. Выбор указанных интерфейсов обосновывается в гл. 1 и обусловлен тем, что первый из них позволяет изучить базовый набор запросов LINQ, а второй дает возможность закрепить навыки в использовании технологии LINQ, применяя ее к задачам обработки данных в формате XML – одном из наиболее распространенных в настоящее время универсальных форматов хранения информации.

Задачи разбиты на три группы:

- LinqBegin (60 задач, посвященных конкретным запросам LINQ, – см. гл. 2);
- LinqObj (100 задач на обработку локальных коллекций – см. гл. 3);
- LinqXml (90 задач, связанных с обработкой документов XML, – см. гл. 4).

Поскольку при выполнении заданий важно ориентироваться на эффективные приемы применения технологии LINQ, в книге приводится большое число примеров решения типовых задач из указанных групп (см. гл. 5–7). Формулировки решенных задач, приведенные в гл. 2–4, снабжаются примечанием, в котором указывается номер пункта с решением данной задачи.

Книга также включает описание новых возможностей языка C# версии 3.0, непосредственно связанных с технологией LINQ (гл. 8), и краткий обзор особенностей интерфейсов LINQ to SQL и LINQ to Entities (гл. 9).

Все задачи, приведенные в книге, входят в состав *электронного задачника Programming Taskbook for LINQ,* являющегося одним из дополнений универсального задачника по программированию Programming Taskbook. Электронный задачник предоставляет программам наборы исходных данных, проверяет правильность полученных результатов, диагностирует различные виды ошибок в программах и отображает на экране все данные, связанные с заданием.

Автоматическая генерация вариантов исходных данных, предоставляемых задачником программе, обеспечивает ее надежное тестирование. Эта возможность оказывается особенно полезной в ситуациях, когда исходные данные являются достаточно сложными и действия по их подготовке и вводу в программу требуют больших усилий. В качестве примеров можно привести задания на обработку массивов, файлов, линейных динамических структур и деревьев. К заданиям со сложными исходными данными можно отнести и за-

дания, связанные с применением технологии LINQ, поскольку все они посвящены обработке последовательностей (в заданиях группы LinqBegin это числовые или строковые последовательности, в заданиях LinqObj это последовательности записей, хранящиеся в текстовых файлах, в заданиях LinqXml – XML-документы).

Задачник включает средства ввода-вывода, позволяющие легко организовать чтение исходных данных и запись результатов. Кроме того, в нем предусмотрены средства отладки, упрощающие поиск и исправление ошибок, и средства визуализации всех данных, связанных с выполняемым заданием. Все эти средства подробно описываются в гл. 5–7, содержащих описания решений типовых задач.

Задания могут выполняться как на языке C#, так и на Visual Basic .NET в любой из версий среды программирования Microsoft Visual Studio, поддерживающих технологию LINQ (версии 2008, 2010, 2012).

Электронный задачник Programming Taskbook и его дополнение Programming Taskbook for LINQ доступны на сайте http://ptaskbook.com/.

Приведенные к книге примеры кода доступны на сайте издательства «ДМК Пресс» www.дмк.рф.

Глава 1. Технология LINQ и ее изучение с применением задачника Programming Taskbook for LINQ

1.1. Технология LINQ и связанные с ней программные интерфейсы

Технология LINQ (**L**anguage **In**tegrated **Q**uery – «запрос, интегрированный в язык»; произносится «линк») расширяет возможности языка программирования путем включения в него дополнительных средств высокого уровня (*запросов LINQ*), предназначенных для преобразования различных наборов данных. «LINQ – это технология Microsoft, предназначенная для обеспечения механизма поддержки уровня языка для опроса данных всех типов» ([4], с. 21). «LINQ – это набор функциональных возможностей языка C# 3.0 и платформы .NET Framework 3.5, обеспечивающих написание безопасных в смысле типизации структурированных запросов к локальным коллекциям объектов и удаленным источникам данных» ([1], с. 315). В каждом из приведенных определений подчеркивается *универсальность* технологии LINQ. Входящие в нее средства можно применять для обработки наборов данных самых разных видов: массивов и других *локальных коллекций* данных, удаленных баз данных, XML-документов, а также любых других источников данных, для которых реализован программный интерфейс LINQ (LINQ API).

В версии .NET Framework 3.5 – первой из версий .NET, поддерживающих технологию LINQ, – были представлены следующие компоненты LINQ API:

- *LINQ to Objects* – базовый интерфейс для стандартных запросов к локальным коллекциям; данный интерфейс основан на

методах класса System.Linq.Enumerable (включает около 40 видов запросов) и позволяет обрабатывать любые объекты, реализующие интерфейс IEnumerable<T>;

- *LINQ to XML* – интерфейс, предназначенный для обработки XML-документов; он включает не только набор запросов, дополняющих стандартные запросы LINQ to Objects и реализованных в методах класса System.Xml.Linq.Extensions (около 10 видов запросов), но и новую *объектную модель* XML-документов (XML DOM), реализованную в виде иерархии классов из пространства имен System.Xml.Linq;
- *LINQ to SQL* и *LINQ to Entities* (*Entity Framework*) – интерфейсы, предназначенные для взаимодействия с удаленными базами данных в качестве источников наборов данных; эти интерфейсы основаны на методах класса System.Linq.Queryable и предназначены для обработки объектов, реализующих интерфейс IQueryable<T>.

Одновременно с включением в .NET интерфейсов LINQ API в языки C# и Visual Basic .NET были добавлены новые возможности, позволяющие максимально упростить программный код, связанный с запросами LINQ. Основными из этих возможностей являются *лямбда-выражения*, *методы расширения* и *анонимные типы* (а также *деревья выражений*, используемые в интерфейсах LINQ to SQL и LINQ to Entities). Кроме того, в языки были включены новые конструкции (так называемые *выражения запросов*), позволяющие представить наиболее сложные запросы LINQ не только в виде цепочек методов расширения, содержащих лямбда-выражения, но и в более наглядном виде, подобном выражениям языка структурированных запросов SQL (такой вариант синтаксиса запросов LINQ получил в русскоязычной литературе название *синтаксиса, облегчающего восприятие*).

Технология LINQ обладает расширяемой архитектурой, которая позволяет разрабатывать дополнительные интерфейсы LINQ API, обеспечивающие обработку специализированных наборов данных. Подобные интерфейсы могут быть реализованы сторонними разработчиками для обеспечения доступа к их хранилищам данных, представленным в специальном формате. В Интернете можно найти ряд дополнительных интерфейсов LINQ, реализованных с той или иной степенью полноты (в качестве примера можно указать *LINQ to Google* и *LINQ to Wiki*), хотя в настоящее время широкое

распространение получили лишь перечисленные выше интерфейсы, реализованные самой компанией Microsoft.

В версии 4.0 .NET Framework набор интерфейсов LINQ был дополнен новым интерфейсом *PLINQ* (*Parallel LINQ*). Данный интерфейс содержит дополнительный набор запросов, связанных с «распараллеливанием» обработки локальных коллекций, и новые реализации всех стандартных запросов LINQ, предусматривающие их выполнение с использованием нескольких потоков (threads). Интерфейс PLINQ основан на методах класса System.Linq.ParallelEnumerable (около 10 методов) и обеспечивает обработку локальных коллекций типа ParallelQuery<T>.

При использовании любых интерфейсов LINQ необходимо прежде всего владеть базовым набором методов LINQ, чтобы наиболее эффективным образом формировать из них последовательность запросов, которая приводит к требуемому преобразованию исходного набора данных. Для изучения методов LINQ проще всего обратиться к интерфейсу LINQ to Objects, поскольку он ориентирован на наиболее простой вид последовательностей – локальные коллекции объектов (например, массивы или объекты типа List<T>).

Среди трех специализированных интерфейсов (LINQ to XML, LINQ to SQL, LINQ to Entities) особый интерес представляет LINQ to XML. Это связано с тем обстоятельством, что с помощью формата XML можно обеспечить представление любых структур данных, причем полученное представление будет платформенно-, аппаратно- и программно-независимым. Кроме того, во многих предметных областях уже имеются XML-спецификации для представления данных. Таким образом, при разработке современных программ достаточно часто будет требоваться включение в них средств, связанных с обработкой данных в формате XML. Интерфейс LINQ to XML предоставляет все необходимые средства обработки XML-данных, причем по удобству использования и наглядности получаемого кода они превосходят средства стандартной модели W3C DOM, предложенной консорциумом W3C – официальным разработчиком стандарта XML. Поэтому в предлагаемой книге наряду с интерфейсом LINQ to Objects подробно рассматривается и интерфейс LINQ to XML.

При использовании интерфейсов LINQ to SQL и LINQ to Entities необходимо предварительно установить связь с удаленными наборами данных, однако после установки такой связи программный код для обработки удаленных наборов методами LINQ не будет отличаться (или будет иметь незначительные отличия) от кода,

обеспечивающего аналогичную обработку локальных коллекций. Поэтому при условии предварительного изучения LINQ to Objects применение технологии LINQ для обработки удаленных наборов данных не должно вызывать особых трудностей; необходимо лишь дополнительно ознакомиться с действиями, требуемыми для установления связи с удаленной базой данных и получения от нее наборов данных для обработки. Однако указанные вопросы относятся, скорее, к технологиям работы с базами данных, поэтому в книге они обсуждаются очень кратко (см. гл. 9). Дополнительные сведения можно почерпнуть из более подробных руководств по использованию технологии LINQ, например [4–5].

Особенности применения интерфейса PLINQ в книге не рассматриваются, поскольку они относятся скорее к особенностям параллельного многопоточного программирования, чем к собственно технологии LINQ. Не случайно в фундаментальном руководстве [2] интерфейс PLINQ описывается в главе 23 «Параллельное программирование», а не в главах 8–10, специально посвященных технологии LINQ и ее интерфейсам.

1.2. Общее описание групп заданий LinqBegin, LinqObj, LinqXml

Как было отмечено выше, книга посвящена базовым средствам технологии LINQ, реализованным в интерфейсе LINQ to Objects, а также дополнительным возможностям, входящим в интерфейс LINQ to XML и связанным с применением технологии LINQ для обработки данных в формате XML. Для изучения указанных средств применяется «практический» подход, основанный на решении большого числа учебных задач – как относящихся к отдельным категориям запросов LINQ, так и требующих применения всей совокупности средств, входящих в тот или иной интерфейс. Изучив описания решений типовых задач, приведенных в гл. 5–7, читатель должен попытаться применить полученные знания для самостоятельного решения хотя бы части задач, формулировки которых содержатся в гл. 2–4.

Книга содержит 250 задач, разбитых на три группы: LinqBegin, LinqObj и LinqXml.

Группа LinqBegin (60 заданий) предназначена для ознакомления с базовыми запросами LINQ. Каждой категории запросов в ней посвящена отдельная подгруппа; описание подгрупп и связанных с ними заданий приводится в табл. 1.1.

Таблица 1.1. Подгруппы группы LinqBegin

Название подгруппы	Изучаемые запросы LINQ	Число задач	Номера задач	Решенные задачи
Поэлементные операции, агрегирование и генерирование последовательностей	First, FirstOrDefault, Last, LastOrDefault, Single, SingleOrDefault; Count, Sum, Average, Max, Min, Aggregate; Range	15	1–15	4 (п. 5.1), 15 (п. 5.2)
Фильтрация, сортировка, теоретико-множественные операции	Where, TakeWhile, SkipWhile, Take, Skip; OrderBy, OrderByDescending, ThenBy, ThenByDescending; Distinct, Reverse; Union, Intersect, Except	16	16–31	31 (п. 5.3)
Проецирование	Select, SelectMany	12	32–43	43 (п. 5.4)
Объединение и группировка	Concat; Join, GroupJoin; DefaultIfEmpty; GroupBy	17	44–60	52, 54 (п. 5.5), 60 (п. 5.6)

Группа LinqObj предназначена для закрепления навыков применения запросов LINQ и включает 100 заданий на использование интерфейса LINQ to Objects для обработки последовательностей и наборов взаимосвязанных последовательностей. При выполнении заданий группы LinqObj необходимо самостоятельно определить набор методов LINQ, обеспечивающих требуемое преобразование исходных последовательностей.

Задания группы LinqObj разбиты на *серии*, каждая из которых связана с определенной предметной областью и включает задачи различного уровня сложности. Описание подгрупп и серий, входящих в группу LinqObj, приводится в табл. 1.2.

Таблица 1.2. Подгруппы и серии задач группы LinqObj

Название подгруппы	Серии задач	Число задач	Номера задач	Решенные задачи
Обработка отдельных последовательностей	Клиенты фитнес-центра	12	1–12	4 (п. 6.1)
	Абитуриенты	12	13–24	
	Задолжники по коммунальным платежам	12	25–36	
	Автозаправочные станции	12	37–48	41 (п. 6.2)
	Баллы ЕГЭ	12	49–60	
	Оценки по предметам	10	61–70	61 (п. 6.2)
Обработка нескольких взаимосвязанных последовательностей	Продажи товаров	30	71–100	98 (п. 6.3), 71 (п. 9.4)

Формулировки заданий группы LinqObj, относящихся к первым четырем сериям, содержат достаточно подробные указания или ссылки на похожие задания; в двух следующих сериях указаниями

снабжаются лишь отдельные задания, а в последней серии (гл. 3.2) указания отсутствуют.

Группа LinqXml (90 заданий) посвящена интерфейсу LINQ to XML и содержит как подгруппы, посвященные отдельным видам преобразований XML-документа, так и завершающую подгруппу с дополнительными заданиями на обработку XML-документа, в которых требуется совместно использовать различные средства интерфейса LINQ to XML. Описание подгрупп, входящих в группу LinqXml, приводится в табл. 1.3.

Таблица 1.3. Подгруппы группы LinqXml

Название подгруппы	Число задач	Номера задач	Решенные задачи
Создание XML-документа	10	1–10	10 (п. 7.1)
Анализ содержимого XML-документа	10	11–20	20 (п. 7.2)
Преобразование XML-документа	20	21–40	28, 32, 37 (п. 7.3)
Преобразование типов при обработке XML-документа	12	41–52	50 (п. 7.4)
Работа с пространствами имен XML-документа	8	53–60	57 (п. 7.5)
Дополнительные задания на обработку XML-документов	30	61–90	61, 82 (п. 7.6)

Завершающая подгруппа группы LinqXml состоит из *серий* задач, связанных с теми же предметными областями, что и задачи группы LinqObj. Каждая серия содержит задачи трех видов: (1) простые задачи, требующие лишь изменения способа представления данных в XML-документе (без изменения его структуры); (2) задачи средней сложности, требующие выполнения дополнительной группировки исходных данных; (3) сложные задачи, требующие группировки или объединения исходных данных, а также включения в документ новых данных, полученных из исходных путем их некоторого преобразования (в частности, агрегирования). Описание серий задач из завершающей подгруппы приводится в табл. 1.4.

Таблица 1.4. Серии задач завершающей подгруппы группы LinqXml

Серия задач	Задачи на изменение структуры	Задачи на группировку	Задачи на определение новых данных
Клиенты фитнес-центра	61–62	63–64	65–67
Автозаправочные станции	68–69	70–71	72–75
Задолжники по коммунальным платежам	76–77	78–79	80–82
Оценки по предметам	83–84	85–86	87–90

1.3. Особенности выполнения заданий с использованием задачника Programming Taskbook for LINQ

Все описанные группы задач входят в состав электронного задачника *Programming Taskbook for LINQ*, являющегося дополнением к универсальному задачнику по программированию Programming Taskbook. Использование задачника предоставляет учащемуся следующие дополнительные возможности:

- при выполнении заданий программа получает от задачника набор исходных тестовых данных для обработки (данные генерируются с использованием датчика случайных чисел);
- задачник автоматически контролирует правильность действий по вводу-выводу данных, информируя учащегося о характере обнаруженных ошибок;
- результаты, полученные программой, передаются задачнику для проверки (сравнения с образцом правильных данных);
- полученные результаты наряду с исходными данными и примером правильного решения отображаются в наглядном виде в окне задачника;
- задание считается выполненным (и информация об этом заносится в специальный файл результатов), если программа учащегося успешно обработает подряд несколько наборов исходных данных, предложенных задачником.

Отмеченные возможности позволяют существенно ускорить процесс выполнения задач, упрощают поиск и исправление ошибок и обеспечивают надежную проверку правильности предложенных решений. Заметим, что все эти возможности доступны при выполнении заданий из любых групп, входящих в задачник Programming Taskbook.

Процесс выполнения задания с использованием задачника Programming Taskbook for LINQ, начиная с создания проекта-заготовки и заканчивая просмотром содержимого файла результатов с информацией обо всех тестовых запусках программы, подробно описан в п. 5.1.

В задачнике предусмотрены специальные средства ввода-вывода, позволяющие программе учащегося получить исходные данные, сгенерированные задачником, и передать ему найденные результаты. При выполнении заданий группы LinqBegin можно использовать

функции, вызов которых обеспечивает ввод или вывод не отдельных элементов данных, а последовательности в целом (см. п. 5.1.2 и 5.3).

В заданиях групп LinqObj и LinqXml исходные наборы данных должны быть считаны из текстовых файлов (предварительно сформированных задачником), а результаты – записаны в новый текстовый файл. Действия по чтению данных из текстовых файлов и записи в них результатов легко реализуются с помощью средств стандартной библиотеки платформы .NET (см. п. 6.1 и 7.1). Содержимое всех файлов, связанных с выполняемым заданием, автоматически отображается в окне задачника; это позволяет ознакомиться с вариантами исходных данных и примером правильного решения, просмотреть файл, созданный программой учащегося, а также сравнить его с «эталонным» файлом с правильными результатами. Возможности задачника, связанные с просмотром файловых данных, подробно описываются в п. 6.1.1.

Задачник позволяет выполнять отладочную печать в специальный раздел окна (раздел отладки). В частности, предусмотрен метод расширения Show, осуществляющий отладочную печать всех элементов последовательности; данный метод может включаться непосредственно в цепочку методов LINQ, используемых для решения поставленной задачи. Отладочные средства задачника подробно описываются в п. 5.3.

Глава 2. Знакомство с запросами LINQ: группа LinqBegin

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

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

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

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

- методы GetEnumerableInt и GetEnumerableString обеспечивают ввод исходных последовательностей с элементами целого и строкового типа соответственно (выполняется ввод размера последовательности и всех ее элементов, возвращается введенная последовательность);
- метод Put является методом расширения для последовательности и обеспечивает вывод этой последовательности (выводятся размер последовательности и все ее элементы);
- метод Show также является методом расширения для последовательности; он обеспечивает печать последовательности в разделе отладки окна задачника и возвращает эту же последовательность (отладочная печать может сопровождаться *ком-*

ментарием, который указывается в качестве необязательного строкового параметра метода Show).

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

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

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

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

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

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

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

2.1. Поэлементные операции, агрегирование и генерирование последовательностей

Изучаемые запросы LINQ:

- First, FirstOrDefault, Last, LastOrDefault, Single, SingleOrDefault (поэлементные операции);
- Count, Sum, Average, Max, Min, Aggregate (агрегирование);
- Range (генерирование последовательностей).

Перед выполнением заданий из данного пункта следует ознакомиться с примером решения задачи LinqBegin4, приведенным в п. 5.1.

Описание особенностей использования методов Range и Aggregate приводится в примере решения задачи LinqBegin15 (см. п. 5.2).

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

LinqBegin2. Даны 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. Даны 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. Даны 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.

Примечание. Решение данной задачи приведено в п. 5.1.

LinqBegin5. Даны 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. Даны 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 $1 + (1/2) + ... + (1/N)$ (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: $(A^2 + (A + 1)^2 + ... + B^2)/(B - A + 1)$ (как вещественное число).

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

Примечание. Решение данной задачи приведено в п. 5.2.

2.2. Фильтрация, сортировка, теоретико-множественные операции

Изучаемые запросы 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).

Перед выполнением заданий из данного пункта следует ознакомиться с примером решения задачи LinqBegin31, приведенным в п. 5.3.

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. Даны 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. Даны 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. Даны 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. Даны целое число K (> 0) и строковая последовательность A. Из элементов A, предшествующих элементу с порядко-