# Введение

Язык C# и связанную с ним среду .NET Framework можно без преувеличения назвать самой значительной из предлагаемых в настоящее время технологий для разработки. Платформа .NET спроектирована как среда, позволяющая разрабатывать практически любое приложение для запуска в Windows, а C# является языком программирования, который был специально создан для работы в .NET Framework. С применением C# и .NET Framework можно создавать, к примеру, динамические веб-страницы, приложения Windows Presentation Foundation, веб-службы XML, компоненты для распределенных приложений, компоненты для доступа к базам данных, классические настольные приложения Windows и даже новые приложения интеллектуальных клиентов, которые обладают онлайновыми и автономными возможностями. В этой книге рассматривается версия .NET Framework 4.5. Если вы пишете код с использованием какой-то из предшествующих версий .NET, то некоторые разделы книги вам не подойдут. В книге всегда приводятся уведомления об элементах, которые являются новыми и присутствуют только в .NET Framework 4.5.

Не стоит поддаваться заблуждению, что из-за наличия в названии платформы конструкции ".NET" она ориентирована исключительно на Интернет. Часть ".NET" призвана подчеркнуть, что по убеждению Microsoft *распределенные приложения*, в которых обработка распределяется между клиентом и сервером, являются движением вперед. Важно также понимать, что C# — это не просто язык для написания приложений, способных работать в Интернете и в сети. Он предоставляет средства для кодирования практически любого типа программного обеспечения или компонента для платформы Windows. Язык C# и платформа .NET привели к революционным изменениям в способе написания разработчиками программ и сделали программирование для Windows гораздо более простым, чем когда-либо.

Так что же такого важного в .NET и C#?

## Важность .NET и C#

Для понимания важности .NET должна быть учтена природа многих технологий Windows, которые появлялись на протяжении последних 18 лет. Хотя на первый взгляд они выглядят разными, все операционные системы Windows, начиная с Windows NT 3.1 (выпущенной в 1993 г.) и заканчивая Window 8 и Windows Server 2012, в своей основе имеют тот же самый хорошо знакомый интерфейс Windows API для настольных и серверных Windows-приложений. По мере появления новых версий Windows в этот API-интерфейс добавлялось большое число новых функций, но это был процесс совершенствования и расширения API-интерфейса, а не его замены.

То же самое можно сказать и о многих других технологий и инфраструктурах, используемых при разработке программного обеспечения для Windows. Например, технология *COM* (Component Object Model — объектная модель компонентов) произошла от технологии *OLE* (Object Linking and Embedding — связывание и внедрение объектов). Изначально она по большей части была средством для связывания различных типов документов Office, позволяя, к примеру, размещать таблицу Excel внутри документа Word. После этого она эволюционировала в технологию COM, затем в *DCOM* (Distributed COM — распределенная объектная модель компонентов) и, наконец, в *COM+* — развитую технологию, которая стала основой для обеспечения взаимодействия между всеми компонентами, а также для реализации транзакций, служб обмена сообщениями и организации пула объектов.

Подобный подход с совершенствованием программного обеспечения был избран Microsoft по вполне очевидной причине — для обеспечения обратной совместимости. За многие годы независимыми разработчиками была написана масса программ для Windows, и Windows не была бы такой успешной операционной системой, если бы каждое добавление новой технологии приводило бы к нарушению работы существующей кодовой базы.

Хотя наличие обратной совместимости является важной характеристикой технологий Windows и одной из сильных сторон платформы Windows, с ним также связан серьезный недостаток. При каждом совершенствовании какой-то технологии и добавлении в нее новых возможностей она становилась немного сложнее, чем была раньше.

Из-за этого со временем стало ясно, что нужно что-то менять. В Microsoft не могли до бесконечности расширять одни и те же языки и средства разработки, постоянно делая их все более и более сложными из-за удовлетворения конфликтующих между собой потребностей в поддержке новейшего оборудования и обеспечении обратной совместимости с тем, что было в ходу, когда Windows впервые обрела популярность в начале 90-х годов прошлого века. Настал момент начать с чистого листа и создать простой, но при этом совершенный набор языков, сред и средств разработки, который бы позволил разработчиками легко писать современное программное обеспечение.

Для этого изначально и были предназначены язык C# и среда .NET. Грубо говоря, .NET представляет собой инфраструктуру — или API-интерфейс — для программирования на платформе Windows. Вместе с .NET Framework язык C# был разработан с нуля специально для работы в .NET, а также для объединения всех достижений, которые были сделаны в средах разработки, и всех принципов объектно-ориентированного программирования, которые были выведены за последние 25 лет.

Прежде чем продолжить, следует отметить, что обратная совместимость при этом не была утрачена. Существующие программы продолжают успешно функционировать, потому что в .NET предусмотрена возможность для работы с существующим программным обеспечением. В настоящее время коммуникации между программными компонентами в Windows происходят почти полностью с использованием COM. С учетом этого в .NET Framework предлагается возможность создания оболочек вокруг существующих компонентов COM, что позволяет компонентам .NET взаимодействовать с ними.

Изучать язык C# для того, чтобы писать код .NET, вовсе не обязательно. В Microsoft расширили язык C++, а также внесли значительные изменения в Visual Basic, сделав его более мощным языком. Это позволяет писать код, ориентированный на .NET, на любом из этих языков. Тем не менее, языки C++ и Visual Basic являются результатом их совершенствования на протяжении многих лет, а не созданными заново с учетом современных технологий.

В настоящей книге показано, как программировать на C#, а также описана работа самой архитектуры .NET. Здесь рассматриваются не только фундаментальные аспекты языка C#, но и приводятся примеры приложений, в которых применяются разнообразные связанные технологии, включая доступ к базам данных, динамические веб-страницы, усовершенствованную графику и доступ в каталоги.

В то время как интерфейс Windows API лишь развивался и расширялся с первых дней после появления Windows NT в 1993 г., в 2002 г. платформа .NET Framework претерпела крупное изменение, связанное со способом написания программ, и теперь, в 2012 г., снова подвергается очередному большому изменению. Происходят ли такие изменения каждые 10 лет? Операционная система Windows 8 предлагает новый API-интерфейс — Windows Runtime (WinRT) для приложений Windows Store. Эта исполняющая среда является машинным API-интерфейсом (подобным Windows API), который не построен на основе исполняющей среды .NET, но предоставляет великолепные новые средства, базирующиеся на идеях .NET. В Windows 8 включен первый выпуск этого API-интерфейса для приложений современного стиля. Хотя это не основано на .NET, в приложениях Windows Store все же можно использовать подмножество .NET и писать эти приложения на языке C#. В ближайшие годы эта новая исполняющая среда будет развиваться с появлением новых выпусков Windows. В настоящей книге вы получите начальные сведения по разработке приложений Windows Store с помощью C# и WinRT.

## Преимущества .NET

До сих пор приводились только общие утверждения о том, насколько замечательной является платформа .NET, но не была дана конкретная информация о способах облегче-

ния ею процесса разработки. В этом разделе кратко описаны некоторые функциональные возможности .NET.

- **Объектно-ориентированное программирование**. Платформа .NET Framework и язык C# с самого начала основаны на принципах объектно-ориентированного программирования.
- **Качественное проектное решение**. Поставляемая библиотека базовых классов построена полностью с нуля и является интуитивно понятной.
- **Независимость от языка**. Благодаря .NET код, написанный на любом языке, т.е. Visual Basic, C# или управляемом C++, компилируется в код на *промежуточном языке* (Intermediate Language — IL). Это обеспечивает высочайший уровень взаимодействия между языками.
- **Улучшенная поддержка для динамических веб-страниц**. Хотя в классической технологии ASP предлагалась довольно высокая гибкость, она была недостаточной из-за использования ею интерпретируемых сценарных языков, а отсутствие объектно-ориентированного подхода часто приводило к получению запутанного кода ASP. В .NET имеется интегрированная поддержка для создания веб-страниц с помощью ASP.NET. В ASP.NET код страниц компилируется и может быть написан на любом высокоуровневом языке .NET, таком как C# или Visual Basic 2012. В новой версии .NET появилась также поддержка последних технологий, подобных Ajax и jQuery.
- **Эффективный доступ к данным**. Набор компонентов .NET, известный под названием ADO.NET, позволяет получать эффективный доступ к реляционным базам данных и многим другим источникам данных. Также предлагаются компоненты для доступа к файловой системе и каталогам. В частности, в .NET встроена поддержка XML, позволяющая манипулировать данными, которые могут импортироваться и экспортироваться на платформы, отличные от Windows.
- **Разделение кода**. В .NET полностью переделан способ разделения кода между приложениями за счет введения понятия *сборки*, заменившей собой традиционную библиотеку DLL. Сборки обладают формальными средствами для управления версиями и сборки разных версий могут существовать бок о бок.
- **Усовершенствованная безопасность**. Каждая сборка может также содержать встроенную информацию, связанную с безопасностью, которая позволяет указывать, какому пользователю или категории пользователей либо процессов разрешено вызывать методы тех или иных классов. Это предоставляет высокий уровень контроля над использованием развертываемых сборок.
- **Установка с нулевым воздействием**. Сборки бывают двух типов: разделяемые и закрытые. Разделяемые сборки — это общие библиотеки, доступные всему программному обеспечению, а закрытые сборки предназначены для использования только определенными приложениями. Закрытые сборки являются полностью самодостаточными, поэтому процесс их установки прост. В системный реестр ничего не заносится; нужные файлы просто помещаются в соответствующую папку файловой системы.
- **Поддержка для веб-служб**. В .NET имеется полностью интегрированная поддержка для разработки веб-служб так же просто, как приложений любых других типов.
- **Visual Studio 2012**. Вместе с .NET поставляется среда разработки Visual Studio 2012, которая способна одинаково хорошо справляться как с кодом на языке C++, C# и Visual Basic 2012, так и с кодом ASP.NET или XML. В Visual Studio 2012 интегрированы все наилучшие возможности сред, специфичных для языков.
- **C#**. Язык C# представляет собой мощный и популярный объектно-ориентированный язык, предназначенный для использования с .NET.

Преимущества архитектуры .NET более подробно обсуждаются в главе 1.

# Что нового в .NET Framework 4.5

Первая версия .NET Framework (1.0) была выпущена в 2002 г. и встречена с большим энтузиазмом. Версия .NET Framework 2.0 вышла в 2005 г. и считается важным выпуском платформы. Основными новыми возможностями версии .NET 2.0 была поддержка обобщений как в C#, так и в исполняющей среде (код IL был изменен для обобщений), а также новые классы и интерфейсы. Версия .NET 3.0 была основана на исполняющей среде 2.0 и предоставила новый способ создания пользовательских интерфейсов (WPF с XAML и векторной графикой вместо растровой), а также новую технологию для коммуникаций (WCF). В версии .NET 3.5 вместе с C# 3 появился язык LINQ, позволяющий применять один синтаксис запросов для всех источников данных. Версия .NET 4.0 стала еще одним важным выпуском платформы, в котором была предложена новая версия исполняющей среды (4.0) и новая версия языка C# (4.0), обеспечивающая интеграцию с динамическими языками и поддерживающая крупную новую библиотеку для параллельного программирования. Платформа .NET Framework 4.5 основана на обновленной версии исполняющей среды 4.0 со многими ожидаемыми новыми возможностями.

В каждом выпуске .NET Framework в Microsoft всегда старались минимизировать число критических изменений, нарушающих работоспособность ранее написанного кода. До сих пор им удавалось довольно успешно справляться с этой задачей. В последующих разделах описаны изменения, появившиеся в C# 5.0 и .NET Framework 4.5.

## Асинхронное программирование

Блокирование пользовательского интерфейса — недружественное к пользователю действие; если интерфейс не реагирует, пользователь проявляет нетерпение. Возможно, вы сталкивались с таким поведением даже в Visual Studio. Есть и хорошая новость: в большинстве сценариев среда Visual Studio стала реагировать намного быстрее.

В .NET Framework всегда существовала возможность вызывать методы асинхронным образом. Однако использовать синхронные методы было намного проще, чем обращаться к их асинхронным вариантам. В версии C# 5 это изменилось. Асинхронное программирование стало таким же простым, как синхронное. Новые ключевые слова C# основаны на применении библиотеки Task Parallel Library, доступной, начиная с .NET 4. Теперь и язык предлагает ее эффективные возможности.

## Приложения Windows Store и среда Windows Runtime

Приложения Windows Store можно программировать на C# с применением среды Windows Runtime и подмножества .NET Framework. Среда Windows Runtime — это новый машинный API-интерфейс, который предоставляет классы, методы, свойства и события подобно .NET, но в то же время является машинным. Платформа .NET была расширена для поддержки средств языковой проекции. В версии .NET 4.5 исполняющая среда .NET 4.0 была обновлена.

## Усовершенствования доступа к данным

Инфраструктура ADO.NET Entity Framework предлагает важные новые возможности. Ее версия была изменена с 4.0 в .NET 4.0 на 5.0 в .NET 4.5. После выхода .NET 4.0 инфраструктура Entity Framework получала обновления в своих версиях 4.1, 4.2 и 4.3. Теперь доступны такие новые средства, как Code First, пространственные типы, перечисления и функции для работы с табличными значениями.

## Усовершенствования WPF

Инфраструктура WPF, используемая при программировании настольных Windows-приложений, также была расширена. Теперь можно заполнять коллекции в потоке, отлич-

ном от потока пользовательского интерфейса; ленточный элемент управления стал частью инфраструктуры; работа со слабыми ссылками на события упростилась; проверка достоверности может выполняться асинхронно с помощью интерфейса `INotifyDataErrorInfo`; динамическое формирование позволяет легко динамически сортировать и группировать данные, которые изменились.

### Инфраструктура ASP.NET MVC

В состав версии Visual Studio 2010 входила инфраструктура ASP.NET MVC 2.0. С выходом Visual Studio 2012 стала доступной версия ASP.NET MVC 4.0. В ASP.NET MVC предлагаются средства для создания приложений ASP.NET с применением архитектурного шаблона "модель-представление-контроллер", который хорошо известен многим разработчикам. Инфраструктура ASP.NET MVC обеспечивает для разработчиков приложений возможность тестирования, высокую гибкость и эффективное сопровождение. ASP.NET MVC не предназначена служить заменой ASP.NET Web Forms, а является просто другим способом конструирования приложений.

## Для чего подходит C#

В определенном смысле C# по отношению к языкам программирования представляет собой то же самое, что платформа .NET по отношению к среде Windows. Точно так же, как Microsoft добавляла все больше и больше средств к Windows и Windows API в течение последних 15 лет, так и Visual Basic 2012 и C++ подвергались постоянному расширению. Хотя Visual Basic и C++ в результате этого стали довольно мощными языками, оба они страдают от проблем, связанных с "тяжелым наследием" их эволюции.

В Visual Basic 6 и более ранних версиях сильная сторона языка определялась простотой его понимания и легкостью решения многих задач программирования с сокрытием от разработчиков приложений деталей Windows API и инфраструктуры компонентов COM. Отрицательная сторона этого была связана с тем, что Visual Basic никогда не был по-настоящему объектно-ориентированным, так что крупные приложения быстро теряли внутреннюю организацию и становились трудными в сопровождении. К тому же, поскольку синтаксис Visual Basic был унаследован от ранних версий языка BASIC (который, в свою очередь, был предназначен для обучения начинающих программистов, а не для написания крупных коммерческих приложений), он не был приспособлен для построения хорошо структурированных объектно-ориентированных программ.

С другой стороны, корни C++ относятся к определению языка ANSI C++. Он не полностью совместим со стандартом ANSI (хотя и близок к нему) по той простой причине, что Microsoft впервые разработала свой компилятор C++ до того, как определение ANSI стало официальным. К сожалению, это привело к возникновению двух проблем. Во-первых, корни ANSI C++ лежат в технологиях десятилетней давности, поэтому ему не хватает поддержки современных концепций (таких как строки Unicode и генерация XML-документации), и он содержит некоторые архаичные синтаксические конструкции, предназначенные для компиляторов прошлых лет (такие как отделение объявления функций-членов от их определения). Во-вторых, разработчики Microsoft одновременно пытаются превратить C++ в язык, предназначенный для выполнения высокопроизводительных задач под Windows, поэтому вынуждены были добавлять к нему огромное количество специфичных для Microsoft ключевых слов и разнообразных библиотек. В результате язык C++ для Windows стал чрезвычайно запутанным. Достаточно лишь спросить у разработчиков на C++, сколько определений строковых типов они знают: `char*`, `LPTSTR`, `string`, `CString` (версия MFC), `CString` (версия WTL), `wchar_t*`, `OLECHAR*` и т.д.

Теперь появилась .NET — полностью новая среда, которая повлекла за собой появление новых расширений в обоих языках. Разработчики из Microsoft добавили в C++ еще больше специфичных ключевых слов. Кроме того, они провели полную переделку Visual

Basic, приведя его к текущей версии Visual Basic 2012 — языку, который унаследовал некоторый базовый синтаксис VB, но положенное в его основу проектное решение полностью отличается от того, к чему мы привыкли. С практической точки зрения Visual Basic стал совершенно новым языком.

В упомянутом контексте в Microsoft решили предложить разработчикам альтернативу — язык, ориентированный специально на .NET, к тому же спроектированный с чистого листа. Так и появился язык C#. Официально в Microsoft описывают C# как простой, современный, объектно-ориентированный и безопасный к типам язык программирования, производный от C и C++. Большинство независимых обозревателей, вероятно, изменили бы это на "производный от языков C, C++ и Java". Такое описание является формально точным, но мало что говорит об элегантности языка. Синтаксически C# похож на C++ и Java, в том смысле, что многие ключевые слова являются теми же самыми; кроме того, C# также разделяет с языками C++ и Java ту же блочную структуру с фигурными скобками для организации блоков кода и точками с запятой для завершения операторов. На первый взгляд фрагмент кода C# выглядит почти как код на C++ или Java. Тем не менее, несмотря на внешнее сходство, C# намного проще в изучении, чем C++; по сложности освоения он сравним с Java. Положенное в его основу проектное решение в большей степени соответствует современным инструментам разработки, чем у обоих его предшественников, и он предлагает простоту применения, характерную для Visual Basic, вместе с высокой производительностью и низкоуровневым доступом к памяти, присущими для C++, когда это необходимо. Ниже перечислены важные характеристики C#.

- Полная поддержка классов и объектно-ориентированного программирования, включая наследование интерфейсов и реализации, виртуальные функции и перегрузку операций.
- Согласованный и четко определенный набор базовых типов.
- Встроенная поддержка автоматического построения XML-документации.
- Автоматическая очистка динамически выделяемой памяти.
- Средство маркировки классов и методов с помощью атрибутов, определяемых пользователем. Это полезно для документирования и может оказывать влияние при компиляции (например, помеченные методы могут компилироваться только для отладочных сборок).
- Полный доступ к библиотеке базовых классов .NET и простое обращение к Windows API (если это действительно нужно, что случается нечасто).
- Возможность работы с указателями и прямым доступом в память, когда это необходимо, но язык спроектирован так, что в большинстве случаев без этого можно обойтись.
- Поддержка свойств и событий в стиле Visual Basic.
- Возможность построения либо исполняемой сборки, либо библиотеки компонентов .NET, к которым можно обращаться из другого кода таким же образом, как к элементам управления Active X (компонентам COM), за счет простого изменения опций компилятора.
- Возможность написания динамических веб-страниц ASP.NET и веб-служб XML.

Бо́льшая часть перечисленного выше также касается Visual Basic 2012 и управляемого C++. Поскольку язык C# с самого начала проектировался для работы с .NET, это означает, что он поддерживает средства .NET в более полной мере и предлагает более подходящий синтаксис, чем другие языки. Хотя язык C# похож на Java, в нем есть ряд существенных преимуществ; в частности, Java не предназначен для работы в среде .NET.

Прежде чем завершить тему, следует отметить пару ограничений языка C#. Одна область, для которой данный язык не предназначен — это критичный ко времени или исключительно высокопроизводительный код, когда приходится принимать во внимание, сколько тактов процессора требуется для выполнения цикла — 1000 или 1050, и необходимо очищать ресурсы в течение считанных миллисекунд после того, как они больше не нужны. Скорее всего, C++ продолжит быть самым непревзойденным из высокоуровневых языков в этой области. Языку C# недостает ряда базовых средств для построения исключительно высокопроизводительных приложений, в том числе возможности указания встраиваемых функций и деструкторов, которые гарантированно запускаются в определенных точках кода. Тем не менее, относительная доля приложений такой категории довольно невелика.

# Что необходимо для написания и выполнения кода C#

Платформа .NET Framework 4.5 может выполняться под управлением клиентских операционных систем Windows Vista, Windows 7 и Windows 8, а также серверных операционных систем Windows Server 2008, Windows Server 2008 R2 и Windows Server 2012. Для написания кода, использующего .NET, необходимо установить комплект .NET 4.5 SDK.

Кроме того, если только не планируется писать код C# с применением какого-то текстового редактора или посторонней среды разработки, имеет смысл установить Visual Studio 2012. Для выполнения управляемого кода полный комплект SDK не нужен, но необходима исполняющая среда .NET. Может также понадобиться распространять эту исполняющую среду вместе с разработанным кодом для тех клиентов, у которых она не установлена.

# Как организована эта книга

Эта книга начинается с обзора общей архитектуры .NET в главе 1, давая представление о том, что требуется для написания управляемого кода. Книга состоит из нескольких частей, в которых описан язык C# и его применение в разнообразных областях.

## Часть I. Язык C#

В этой части предоставляются базовые сведения о языке C#. Здесь не предполагается наличие знаний какого-то другого языка, а только опыта программирования. Сначала рассматривается базовый синтаксис и типы данных C#, после чего исследуются объектно-ориентированные возможности C#, а затем и более сложные темы, связанные с программированием на C#.

## Часть II. Visual Studio

В этой части рассматривается основная IDE-среда, которой пользуются разработчики приложений на C# по всему миру — Visual Studio 2012. В двух главах этой части показано, как лучше всего применять этот инструмент для построения приложений на основе .NET Framework 4.5. Кроме того, описаны способы развертывания проектов.

## Часть III. Основы

В этой части рассматриваются принципы программирования в среде .NET. В частности, вы узнаете о безопасности, многопоточности, локализации, транзакциях, приемах построения служб Windows, способах генерации собственных библиотек в виде сборок, а также многое другое. Здесь объясняется взаимодействие с машинным кодом и сборками с использованием P/Invoke и взаимодействия с COM. Также предоставляются сведения о том, чем отличаются среды Windows Runtime и .NET, и рассматривается написание программ в стиле Windows 8.

## Часть IV. Данные

В этой части вы узнаете о доступе к данным с использованием ADO.NET и изучите инфраструктуру ADO.NET Entity Framework. Для достижения наилучшей производительности можно применять ядро ADO.NET; инфраструктура ADO.NET Entity Framework обеспечивает простоту использования при отображении объектов в отношениях. Рассматриваются различные доступные модели программирования — Model First, Database First и Code First. В этой части также подробно описана поддержка XML в .NET и применение LINQ для запрашивания источников данных XML.

## Часть V. Презентация

Эта часть начинается с описания способов построения приложений, основанных на инфраструктуре Windows Presentation Foundation (WPF). Подробно рассматриваются не только типы элементов управления, стили, ресурсы и привязка данных, но также создание фиксированных и потоковых документов и вывод на печать. Здесь вы узнаете о создании приложений Windows Store, использовании изображений для улучшения пользовательского интерфейса, применении сеток и работе с контрактами для взаимодействия с другими приложениями. Наконец, в этой части приводятся сведения о многочисленных средствах, предлагаемых ASP.NET, построению веб-сайтов с помощью ASP.NET Web Forms, архитектуре ASP.NET MVC и динамических данных.

## Часть VI. Коммуникации (на веб-сайте)

Эта часть целиком посвящена коммуникациям. Здесь описаны службы для независимых от платформы коммуникаций, реализуемые с использованием Windows Communication Foundation (WCF), и применение WCF для доступа к данным с помощью WCF Data Services. Также рассматривается организация отключенных асинхронных коммуникаций с использованием Message Queuing, работа с Windows Workflow Foundation (WF) и взаимодействие в одноранговых сетях.

# Соглашения

Чтобы помочь читателям извлечь максимальную пользу из текста и дать более четкое представление о том, что происходит, в книге используется несколько соглашений.

**Внимание!** В таких врезках приводится важная для запоминания информация, которая имеет непосредственное отношение к окружающему тексту.

**На заметку!** В таких врезках приводятся замечания, советы, подсказки, полезные приемы и сведения, которые дополняют текущее обсуждение.

В тексте встречаются следующие стили.

- Новые термины и важные слова при первом упоминании выделяются *курсивом*.
- Сочетания клавиш представляются следующим образом: <Ctrl+A>.
- Имена файлов, URL-адреса и код внутри текста выделяются так: `persistence.properties`.
- Примеры кода представлены двумя разными способами:

В большинстве примеров кода используется моноширинный шрифт без выделения. **Полужирным начертанием выделяется код, который особенно важен в текущем контексте, либо изменения в коде по сравнению с предыдущим примером.**

# Исходный код

Работая с примерами, вы можете либо вводить их код вручную, либо использовать файлы исходного кода, доступные для загрузки на веб-сайте издательства.

# От издательства

Вы, читатель этой книги, и есть главный ее критик и комментатор. Мы ценим ваше мнение и хотим знать, что было сделано нами правильно, что можно было сделать лучше и что еще вы хотели бы увидеть изданным нами. Нам интересно услышать и любые другие замечания, которые вам хотелось бы высказать в наш адрес.

Мы ждем ваших комментариев и надеемся на них. Вы можете прислать нам бумажное или электронное письмо, либо просто посетить наш веб-сервер и оставить свои замечания там. Одним словом, любым удобным для вас способом дайте нам знать, нравится или нет вам эта книга, а также выскажите свое мнение о том, как сделать наши книги более интересными для вас.

Посылая письмо или сообщение, не забудьте указать название книги и ее авторов, а также ваш обратный адрес. Мы внимательно ознакомимся с вашим мнением и обязательно учтем его при отборе и подготовке к изданию последующих книг.

Наши координаты:

E-mail: info@dialektika.com

WWW: http://www.dialektika.com

Информация для писем из:

России: 127055, г. Москва, ул. Лесная, д. 43, стр. 1

Украины: 03150, Киев, а/я 152