Об этой книге

Эта книга посвящена языку C#, начиная с версии 2 и далее — вот так все просто. Я очень мало раскрываю версию C# 1, а библиотеки .NET Framework и общеязыковую исполняющую среду (Common Language Runtime — CLR) описываю, только когда они касаются языка. Это обдуманное решение, в результате которого получилась книга, несколько отличающаяся от большинства виденных мною книг по C# и .NET.

За счет предположения, что читатель располагает разумным объемом знаний C# 1, я избегаю траты сотен страниц на представление материала, который, как я считаю, большинство читателей уже понимает. Это обеспечивает мне пространство для раскрытия деталей более поздних версий C#, из-за которых, как я надеюсь, вы читаете данную книгу. Когда я писал первое издание этой книги, даже версия C# 2 была относительно незнакома некоторым читателям. К настоящему времени почти все разработчики на C# имеют определенный опыт использования средств, введенных в C# 2, но я все равно сохранил этот материал в третьем издании, поскольку он является чрезвычайно фундаментальным для того, что появилось в дальнейшем.

Кто должен читать эту книгу?

Эта книга ориентирована на разработчиков, в какой-то степени уже знающих язык C#. Абсолютно ценно, если вы будете хорошо знать C# 1, но очень немногое относительно последующих версий. Существует не так много читателей, которые находятся в лучшем положении, но я уверен, что все еще есть немало разработчиков, которые могут выиграть от более глубокого исследования версий C# 2 и C# 3, даже если они уже применяют их какое-то время, а многие разработчики даже не пользовались версией C# 4 или C# 5.

Если же вы вообще не знаете C#, то, вероятно, эта книга не для вас. Вы могли бы стараться изо всех сил, пытаясь понять аспекты, которые вам не знакомы, но это не может считаться эффективным способом изучения. Гораздо лучше начать с другой книги, а затем постепенно добавить в свою коллекцию и данную книгу. Доступно широкое многообразие книг, раскрывающих язык C# с нуля и написанных в самых разных стилях. Рекомендуется обратиться к одной из следующих книг (или даже ко всем им): *C# 5.0. Справочник. Полное описание языка* (ИД “Вильямс”, 2013 г.), *Язык программирования C# 5.0 и платформа .NET 4.5* (ИД “Вильямс”, 2013 г.) и *C# 5.0 и платформа .NET 4.5 для профессионалов* (“Диалектика”, 2013 г.).

Я не собираюсь заявлять, что чтение этой книги сделает из вас классного кодировщика. Инженерия программного обеспечения предполагает намного большее, чем одно лишь знание синтаксиса языка, который приходится применять. Хотя я даю некоторые руководящие рекомендации, но, в конечном счете, интуиция и инстинкты при разработке присутствуют в намного большей степени, чем многие бы из нас хотели. Однако я могу утверждать, что если вы прочитаете и поймете эту книгу, то должны чувствовать себя комфортно с C# и свободно следовать своим инстинктам без особого опасения. Речь идет не о том, что можно написать код, который никто другой не поймет, поскольку в нем используются неизвестные закоулки языка, а о наличии уверенности в том, что вы знаете доступные варианты и то, к какому пути следования вас подталкивают идиомы C#.

Дорожная карта

Структура книги проста. Есть пять частей и три приложения. Первая часть служит введением, включая повторение тем, связанных с C# 1, которые важны для понимания более поздних версий языка и часто вызывают путаницу. Во второй части раскрываются новые средства версии C# 2, в третьей рассматривается версия C# 3 и т.д.

Бывают случаи, когда организация материала подобным образом означает, что мы будем возвращаться к какой-то теме пару раз — в частности, делегаты были усовершенствованы в C# 2 и затем еще раз в C# 3, — но в моем безрассудстве смысл все же присутствует. Я предвижу, что некоторые читатели будут применять в разных проектах разные версии языка; например, на работе вы можете использовать C# 4, а дома экспериментировать с C# 5. Это значит, что удобно пояснять, что к какой версии относится. Кроме того, такая организация способствует ощущению контекста и эволюции — она отражает то, каким образом язык развивался с течением времени.

В **главе 1** устанавливается сцена, для чего берется простой фрагмент кода C# 1 и затем развивается, чтобы продемонстрировать, каким образом последующие версии позволяют исходному коду становиться более читабельным и мощным. Мы взглянем на исторический контекст, в котором расширялся язык C#, и технический контекст, в котором он действует в качестве завершенной платформы; C# как язык построен на библиотеках платформы и мощной исполняющей среде для превращения абстракции в реальность.

В **главе 2** мы снова будем иметь дело с C# 1, рассматривая три специфичных аспекта: делегаты, характеристики системы типов и разницу между типами значений и ссылочными типами. Эти темы часто понимаются разработчиками на C# 1 в стиле “лишь относительно хорошо”, но поскольку язык C# развивался и значительно усовершенствовал их, для освоения большинства новых средств требуется глубокое понимание основ.

В **главе 3** обсуждается крупнейшее средство C# 2, потенциально самое трудное для освоения: обобщения. Методы и типы могут быть написаны обобщенным образом, с параметрами типов, указанными вместо реальных типов, которые задаются в вызывающем коде. Поначалу обобщения будут казаться не менее запутанными, чем это описание, но после того, как вы их поймете, вы непременно удивитесь, как в принципе могли обходиться без них ранее. Если вам когда-либо хотелось представлять целочисленное значение, равное `null`, то **глава 4** как раз для вас. В ней рассматриваются типы, допускающие `null`: средство, построенное на основе обобщений, которое использует в своих интересах поддержку со стороны языка, исполняющей среды и инфраструктуры.

В **главе 5** описаны усовершенствования делегатов в C# 2. До сих пор делегаты можно было применять только для обработки событий, таких как щелчки на кнопках. В C# 2 упростилось создание делегатов, а библиотечная поддержка сделала их более удобными в ситуациях, отличных от обработки событий.

В **главе 6** будут исследоваться итераторы и легкий способ их реализации в C# 2. Итераторные блоки используют лишь немногие разработчики, но поскольку технология LINQ to Objects построена на основе итераторов, они будут становиться все более и более важными. Ленивая природа их выполнения также является ключевой частью LINQ.

В **главе 7** представлено несколько мелких средств, введенных в C# 2, каждое из которых делает жизнь чуть более приятной. Проектировщики языка сгладили несколько шероховатостей в C# 1, сделав возможным более гибкое взаимодействие с генераторами кода, улучшив поддержку вспомогательных классов, обеспечив более гибкий доступ к свойствам и т.д.

В **главе 8** снова раскрывается несколько относительно простых средств, но на этот раз из версии C# 3. Почти весь новый синтаксис приспособлен к общей цели техноло-

гии LINQ, но строительные блоки также полезны и сами по себе. Благодаря анонимным типам, автоматически реализуемым свойствам, неявно типизированным локальным переменным и существенно улучшенной поддержке инициализации, C# 3 становится намного более мощным языком для выражения необходимого поведения.

В **главе 9** рассматривается первая крупная тема, связанная с C# 3 — лямбда-выражения. Не довольствуясь и без того достаточно лаконичным синтаксисом, который обсуждался в главе 5, проектировщики языка еще больше упростили создание делегатов по сравнению с C# 2. Лямбда-выражения способны на большее — они могут быть преобразованы в деревья выражений, которые являются мощным способом представления кода в виде данных.

В **главе 10** будут исследоваться расширяющие методы, которые позволяют заставить компилятор считать, что методы, объявленные в одном типе, на самом деле принадлежат другому типу. На первый взгляд они выглядят кошмаром в плане читабельности, но если хорошо подумать, то это исключительно мощное средство, к тому же являющееся жизненно важным для LINQ.

Глава 11 объединяет три предыдущих главы в форме выражений запросов — лаконичным, но мощным способом запрашивания данных. Первоначально мы сосредоточим внимание на LINQ to Objects, но вы увидите, что шаблон выражений запросов применяется так, что позволяет легко подключать других поставщиков данных.

В **главе 12** предлагается краткий обзор разнообразных сценариев использования LINQ. Сначала мы взглянем на преимущества выражений запросов, скомбинированных с деревьями выражений, демонстрируя способность LINQ to SQL преобразовывать то, что выглядит как нормальный код C#, в операторы SQL. Затем мы перейдем к рассмотрению способов проектирования библиотек для хорошего сочетания с LINQ, взяв в качестве примера LINQ to XML. Двумя альтернативными подходами к внутрипроцессным запросам являются Parallel LINQ и Reactive Extensions, а в завершение главы приводятся рассуждения о том, как можно расширить LINQ to Objects собственными операциями LINQ.

Рассмотрение версии C# 4 начинается в **главе 13**, в которой раскрываются именованные аргументы и необязательные параметры, улучшения во взаимодействии с COM и обобщенная вариантность. В некотором смысле все они являются совершенно разными средствами, но именованные аргументы и необязательные параметры способствуют взаимодействию с COM, а также более специфичным возможностям, которые доступны только при работе с объектами COM.

В **главе 14** описана единственное крупное средство C# 4: динамическая типизация. Возможность динамической привязки членов во время выполнения вместо статической на этапе компиляции является серьезным концептуальным отклонением для языка C#, но она применяется выборочно — выполняться динамически будет только код, в котором задействовано динамическое значение.

Глава 15 посвящена асинхронности. В C# 5 содержится только одно главное средство — возможность написания асинхронных функций. Это единственное средство одновременно сложно в понимании и элегантно в использовании. Наконец-то стало возможным написание кода, который не выглядит подобно спагетти.

В **главе 16** рассматриваются оставшиеся средства C# 5 (оба они крошечные) и приводятся некоторые мысли о будущем.

Приложения содержат справочный материал. В **приложении А** описаны стандартные операции запросов LINQ с несколькими примерами. В **приложении Б** рассматриваются основные обобщенные классы и интерфейсы коллекций. В **приложении В** предоставляется краткое описание разных версий .NET, включая разновидности наподобие Compact Framework и Silverlight.

Терминология, оформление и загружаемый код

Большая часть терминологии, применяемой в книге, объясняется по ходу дела, но есть несколько определений, о которых полезно упомянуть здесь. Я использую понятия C# 1, C# 2, C# 3, C# 4 и C# 5 вполне очевидным образом, но в других книгах и на веб-сайтах вы можете встретить варианты C# 1.0, C# 2.0, C# 3.0, C# 4.0 и C# 5.0. Как по мне, указание дополнительной конструкции ".0" избыточно, поэтому я ее опускаю — надеюсь, смысл понятен.

Я позаимствовал пару терминов из книги по C#, написанной Марком Михаэлисом. Во избежание путаницы между *исполняющей средой* (runtime), как в "общеязыковой исполняющей среде", и моментом времени, как в "переопределение происходит во время выполнения", для последней концепции Марк применяет понятие *время выполнения* (execution time), обычно сопоставляя его с *этапом компиляции* (compile time). Мне кажется, что это совершенно здравая идея, которая, как я надеюсь, будет принята широким сообществом. Следуя его примеру, я внес свою лепту, используя данные термины в этой книге.

Я часто ссылаюсь на "спецификацию языка" или просто "спецификацию" — если только не указано иное, это означает спецификацию языка C#. Тем не менее, доступно множество версий этой спецификации, частично из-за наличия разных версий самого языка, а частично из-за процесса стандартизации. Все упоминаемые номера разделов относятся к спецификации по языку C# 5.0, написанной в Microsoft.

В настоящей книге содержатся многочисленные фрагменты кода, которые представлены с помощью моноширинного шрифта; этот же шрифт также применяется в выводе кода из листингов. Некоторые листинги сопровождаются аннотациями, а иногда отдельные части кода выделены полужирным, чтобы обозначить изменение, улучшение или добавление. Почти весь код приведен в форме фрагментов, позволяющих ему оставаться компактным, но по-прежнему выполняемым, правда, в надлежащей среде. Такой средой является Snippy — специальный инструмент, который представлен в разделе 1.8. Инструмент Snippy доступен для загрузки наряду с кодом всех примеров, рассмотренных в книге (в форме фрагментов, в виде полноценных решений Visual Studio или чаще в обоих видах), на веб-сайте книги csharpindepth.com, а также 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@williamspublishing.com
WWW:	http://www.williamspublishing.com

Информация для писем из:

России:	127055, г. Москва, ул. Лесная, д. 43, стр. 1
Украины:	03150, Киев, а/я 152