ПРОГРАММИСТУ

Дж. Бишоп, Н. Хорспул

# C#

## в кратком изложении

БИНОМ

# Оглавление

# К читателям

Многие книги, посвященные компьютерам, имеют такой объем, что, поднимая их, вы рискуете заработать грыжу, однако материала в них не больше, чем в бульварной газете. Академические учебники зачастую так скучны, что у вас плавятся мозги, хотя они не учат вас ничему, что могло бы пригодиться в реальной жизни.

Но только не эта книга! Этот краткий курс аккумулирует в себе многолетний опыт и знания авторов. В каждую главу включены многочисленные примеры и упражнения, но что мне лично нравится больше всего — так это волшебная россыпь компактных справочных «форм», щедро разбросанная по всей книге.

Широкое использование не зависящего от платформы пространства имен Views, специально разработанного для этой книги и позволяющего создавать элементы графического интерфейса пользователя, облегчает изучение материала и способствует приобретению практического опыта. Еще одним достоинством книги является включение специальной главы, посвященной отладке. Даже в невероятном предположении, что вы при создании программы никогда не вносите в нее ошибки, отладка представляет собой идеальное средство для понимания динамики работы программы; ну а для нас, простых людей, это дополнительное мощное средство локализации наших ошибок.

Имея на своем столе эту книгу, вы превратите каждый рабочий день в праздничный.

*Эрик Мейер* руководитель группы
технической поддержки
Microsoft WebData Team

31 июля 2003 г.

# Предисловие

Уилльяму и Майклу, как и всегда, а также Джанет,
без любви и поддержки которых эта книга не была бы написана
*Дж. Бишоп*

Микаэле, которая в течение подготовки этой книги щедро помогала и
ободряла нас; мы и представить себе не могли,
что эта работа займет столько времени
*Н. Хорспул*

Написание этой книги заняло у нас целый год. Нам хотелось бы объяснить вам, будь вы преподаватель, студент или просто любознательный читатель, зачем мы взяли на себя этот труд и что вы почерпнете из этой книги.

## C# — откуда и куда

C# — это новый язык, разработанный Эндерсом Хейлсбергом (Anders Hejlsberg), Скоттом Уилтамутом (Scott Wiltamuth) и Питером Гоулдом (Peter Golde) в корпорации Microsoft в качестве основной среды разработки для .Net Framework и всех будущих продуктов Microsoft. C# берет свое начало в других языках, в основном, в C++, Java, Delphi, Modula-2 и Smalltalk. Про Хейлсберга следует сказать, что он был главным архитектором Turbo Pascal и Borland Delphi, и его огромный опыт способствовал весьма тщательной проработке нового языка. С одной стороны, для C# в еще большей степени, чем для упомянутых выше языков, характерна внутренняя объектная ориентация; с другой стороны, в нем реализована новая концепция упрощения объектов, что существенно облегчает освоение мира объектно-ориентированного программирования.

В настоящее время C# утверждается среди языков, используемых как для разработки нового программного обеспечения, так и для обучения. Поскольку эта книга является прежде всего учебником, мы хотели бы подчеркнуть, что, по нашему мнению, C# является идеальным языком для начального обучения программированию. Некоторым препятствием здесь является консервативность: в большинстве организаций твердые позиции занял язык Java и такое его положение поддерживается новыми разработками и всей инфрастуктурой.

C# также удобен для более специализированных курсов. Ввиду того что этот язык поддерживает такие современные концепции, как делегирование и перегрузку операторов, и даже делаются предложения о реализации родовых классов, C# представляется превосходной средой для таких, в частности, курсов, как «Современные методы программирования», «Структуры данных», «Сетецентрические модели вычислений» и «Распределенные системы». Использование C# может также оказаться полезным для курсов повышения квалификации с производственным уклоном,

к каковым относятся многие программы обучения Microsoft, поскольку обучающиеся сразу обеспечиваются реальной средой программирования.

В отличие от Java, C# можно использовать только при определенных обстоятельствах. Если академическая организация принадлежит объединению Microsoft Academic Alliance (вступить куда можно за минимальную плату), то программное обеспечение будет доступно как сотрудникам, так и студентам. Язык C# (но не все его библиотеки) утвержден в качестве стандарта Европейской ассоциацией ECMA (в декабре 2001 г.) в результате чего появились и другие (также бесплатные) компиляторы для Windows и других платформ. Сама корпорация Microsoft выпустила три таких продукта под кодовым именем Rotor для платформ Windows, FreeBSD и Macintosh OSX.

Продукты Microsoft обычно отличаются интенсивным потреблением ресурсов и требуют таких компьютеров, которые находятся за пределами возможностей многих студенческих лабораторий. В силу этого бесплатные компиляторы могут оказаться предпочтительнее. Однако в стандарте ECMA имеется серьезное упущение — отсутствует прикладной интерфейс Windows.Form, используемый для программирования графических интерфейсов пользователя. Мы работаем с корпорацией Microsoft над восполнением этого пробела с апреля 2002 г., и результатом этой работы явился продукт Views, небольшая экономная система графического интерфейса пользователя (GUI), основанная на языке XML. Разработаны варианты Views как для стандартных Windows-систем, так и для других платформ, воспринимающих систему Rotor. Совокупность системы Views и бесплатных компиляторов C# дает вполне жизнеспособную и технологически современную альтернативу комбинации лицензионных C# и Visual Studio корпорации Microsoft, обеспечивая при этом еще и независимость от платформы.

C# несомненно имеет большое будущее как в обучении и исследованиях, так и в системных разработках. Целью этой книги является приближение этого будущего путем максимально широкого распространения языка C#.

## Обзор книги

Для современных учебников по программированию стала общим местом декларация использования подхода «объекты в первую очередь». Внимательное изучение учебников и докладов на образовательных конференциях по компьютерным направлениям показывает, что смысл выражения «объекты в первую очередь» довольно туманен. Более того, в этих работах редко указывается, что же понимается под второй или третьей очередью.

Несомненно, настоящая книга использует подход «объекты в первую очередь», при этом, используя новаторскую методику изложения, нам удалось, надеемся, вложить в это выражение более четкий смысл. В гл. 2 мы вводим объекты действительно в первую очередь путем *использования* их, а не определения для них типов. Мы основываем изложение на двух

встроенных структурных типах, DateTime (структура) и Random (класс), и развиваем концепции реализации, переменных, методов и присваивания значительно более естественным путем, чем было бы возможно в случае требования первоначального определения типов.

Далее в гл. 3 мы рассматриваем вопрос определения типа, возвращаясь назад и обобщая уже знакомые читателю концепции, а также развивая формализм таких понятий, как параметры и свойства. Мы приняли твердое решение оставаться на этой стадии в мире значений переменных, используя для создания объектов структуры и упоминая ссылки лишь как средство передачи параметров. Мы уверены, что лучшим педагогическим принципом является строго последовательное рассмотрение концепций, и, откладывая до гл. 7 детальное рассмотрение аппарата ссылок, мы тем самым устраняем часто возникающие недоумения по поводу использования значений переменных и ссылок на них. Ввиду такого решения обсуждение классов начинается лишь в гл. 8. C# в большей степени, чем другие современные объектно-ориентированные языки, опирается на понятие типа, причем это понятие интерпретируется одинаково, независимо от того, идет ли речь о скалярной переменной типа int, структуре или классе. Мы придерживаемся того же принципа, и откладывание обсуждения классов на более поздний этап не приводит к каким-либо неестественным программным конструкциям.

На вопрос «если объекты в первую очередь, то что во вторую?» мы отвечаем в гл. 4: управляющие структуры, строки и массивы. Все они представляют собой базовые строительные блоки, используемые при разработке программ. Без этих блоков нельзя полноценно использовать более мощные объектно-ориентированные средства — коллекции и полиморфизм.

Перед тем, однако, как перейти к этим полезным предметам, мы потратим две главы на то, чтобы заложить основы целенаправленного программирования. Глава 5 описывает систему Views, которую мы разработали в качестве дополнения, облегчающего обслуживание графических пользовательских интерфейсов в любой C#-программе. Имея такое инструментальное средство, вы получаете возможность быстрее разрабатывать программы, взаимодействующие с пользователем, причем независимо от платформы.

В гл. 7 мы завершаем исследование ввода и вывода, рассмотрев хранение данных в файлах, а также представление файлов в C# в виде потоков данных. Гл. 6, посвященная ошибкам и процедурам отладки, почти уникальна для учебников по программированию. В этой главе описываются как стандартные, так и новые методики отладки, от выводимых на экран сообщений до контрольных вызовов. Большая часть этих методик не требует никакого дополнительного программного обеспечения кроме компилятора C#. В отдельном разделе описывается использование отладчика, а в Приложении Ж приводятся дополнительные сведения об отладочных возможностях Microsoft Visual Studio.

К этому моменту программисты смогут справиться с любой задачей, доступной решению на более старых языках, но в объектно-ориентированном стиле. Главы 8 и 9 завершают тему объектов рассмотрением коллек-

ций (с которыми особенно удобно работать на C#), а также расширяемости и полиморфизма. В гл. 9 мы проводим обсуждение принципов, а затем показываем, как они могут быть реализованы с помощью любой из трех имеющихся методик — наследования, интерфейсов и абстрактных классов.

В девяти первых главах книги мы рассмотрели большую часть языка C#, хотя, разумеется, остался еще целый пласт функций прикладного интерфейса пользователя (API). Какие из них особенно важны? Какие следует тщательно изучить? Книга не может считаться завершенной, если в ней на рассказывается, как выполнять рисование различных изображений. Поэтому мы посвятили гл. 10 рассмотрению API пространства имен System.Drawing, показав использование этих возможностей как независимо, так и интегрированными в систему View. Далее, неотъемлемой частью современной жизни является распределенная обработка данных, и мы посвятили несколько страниц способам, с помощью которых компьютеры могут взаимодействовать в C#. Мы показываем, как эти способы позволяют с легкостью реализовывать даже весьма сложные программы.

## Наш подход

Настоящая книга выполняет свои обучающие функции с помощью смеси равных долей формализма и примеров. Каждый новый структурный компонент или встроенный тип описывается сначала в виде «формы» — таблицы, в которой даются синтаксис и семантика объекта, затем приводятся простые примеры и, наконец, рассматривается полный текст работоспособной программы. Таких программ насчитывается в книге более 50; некоторые из них развиваются по мере перехода от главы к главе, демонстрируя, как новые средства позволяют получить более полное решение исходной задачи.

Важными аспектами программирования являются графический интерфейс пользователя (GUI) и тестирование программ. Везде, где можно, для получения тестовых данных используется генератор случайных чисел, что позволяет должным образом испытать программы и получить разумный объем выходных данных.

Никакой язык нельзя представить в виде набора изолированных друг от друга средств, и C# не является исключением. При рассмотрении некоторых средств, таких как вывод, строки, циклы и массивы, мы использовали «спиральный» подход, сначала вводя новые понятия в элементарной форме, и лишь в дальнейшем завершая их полное описание.

Хороший учебник должен быть максимально интерактивным, и мы предусмотрели ряд дополнительных средств обучения как в самой книге, так и на соответствующем Web-сайте. Эти средства включают в себя:

- ряд приложений с перечнем ключевых прикладных интерфейсов и средств языка, объединенных в группы для облегчения поиска;

- контрольные вопросники с альтернативными ответами в конце каждой главы[2];
- разнообразные упражнения разного уровня сложности в конце большинства глав;
- все примеры, доступные в сети по адресам http://www.cs.up.ac.za/csharp или http://www.booksites.net/bishop;
- полная система Views, со справочником и примерами;
- дискуссионный форум для вопросов, касающихся C#, книги и Views;
- набор слайдов и ответов для преподавателя.

## Система Views

Система Views представляет собой специальный класс для создания прикладных GUI независящим от поставщика способом с использованием спецификации XML. Система была разработана авторами в качестве проекта в рамках инициативы Rotor, организованной корпорацией Microsoft в апреле 2002 г. Система тестировалась студентами первого и второго года обучения, которые с энтузиазмом использовали ее в качестве альтернативы обычному программированию посредством API Windows.Form или Visual Studio. Последнюю версию Views можно переписать по сети с Web-сайта этой книги по адресу http://www.cs.up.ac.za/csharp или http:// www.booksites.net/bishop.

## Благодарности

Мы хотели бы поблагодарить за значительный практический вклад в эту книгу и в систему Views студентов Polelo Group в Университете Претории — Бэзила Уоррала (Basil Worrall), Катрину Берг (Kathrin Berg), Джони Ло (Johny Lo), Дейвида-Джона Миллера (David-John Miller), Мэнди ван Шалквик (Mandy van Schalkwyk) и Сфисо Щабалала (Sfiso Tshabalala). Глубокая благодарность Эдвину Пиру (Edwin Peer), который разработал и обслуживал Web-репозиторий, столь хорошо поддерживающий эту книгу. В Университете Виктории Джонатан Мейсон (Jonathan Mason) помог найти и устранить ошибки в программах View, Раджвиндер Пейнсар-Уэйлаведж (Rajwinder Panesar-Walawedge) отвечал за реализацию варианта View для системы Rotor, а Бернард Шольц (Bernhard Scholz) (в творческом отпуске, предоставленном Венским Техническим Университетом) работал над важными программами клиентской части Views, которые будут выпущены позже.

---

[2] Издательство сочло целесообразным дополнить русское издание книги списком правильных ответов. — *Прим. ред.*

Нам хотелось бы поблагодарить корпорацию Microsoft за их щедрый дар в поддержку этой книги и системы Views, а Дж. Бишоп особо отмечает инициативу и поддержку со стороны Лианн Скотт-Уилльямс (Leanne Scott-Williams) из корпорации Microsoft (Южная Африка). Эта работа также поддерживалась грантом THRIP из южноафриканского фонда National Research Foundation. Н. Хорспул также благодарит Памелу Лоз (Pamela Lauz) из Microsoft Canada за поддержку.

Выполняя совместную работу при наличии 10-часовой разницы во времени, мы не могли бы выжить без превосходной интернет-поддержки, предоставленной нашими кафедрами в Университетах Претории и Виктории, а также Microsoft Research Laboratory в Кембридже и Венском Техническом Университете, где эта книга была завершена. Дж. Бишоп особо благодарит д-ра Пола Мира (Paul Meyer) и штат Адденбрукского госпиталя в Кембридже за то, что они поставили ее на ноги в тот момент, когда работа над книгой дошла до завершающих стадий.

Поскольку значительная часть работы над книгой выполнялась в необычное время суток, Н. Хорспул хотел бы поблагодарить Torrefazione Italia за весьма необходимый и прекрасно приготовленный кофе, а KBSG Seattle за предоставление удачной фоновой музыки. Дж. Бишоп также ценит вклад винной индустрии Кейптауна и Late Late Show от Classic FM.

Наконец, наша общая признательность нашим семьям за их терпение в течение многих часов, проведенных нами дома в работе над этим проектом, а также за стимулирующие обсуждения, в которых они, такие же компьютерно-ориентированные, как и мы сами, могли принимать участие. Мы надеемся, что наши семьи будут так же довольны получившимся результатом, как и мы сами.

*Джудит Бишоп*
Претория, Южная Африка и Кембридж, Англия

*Найджел Хорспул*
Виктория, Канада и Вена, Австрия

Август 2003 г.

# Введение 1

Изучение техники программирования — увлекательнейшее занятие. Столько надо понять перед тем, как удастся написать хотя бы простейшую программу! В первой главе мы постараемся дать именно эти начальные знания. Мы обсудим роль языков программирования и их развитие во времени. Общее направление их совершенствования — позволить программисту меньше думать о деталях и сосредоточиться на решении поставленной перед ним задачи. Для ввода программы в компьютер и ее запуска должны быть предусмотрены какие-то средства, и мы рассмотрим две имеющиеся возможности. Некоторое внимание будет также уделено разработке программного обеспечения.

## 1.1 Преамбула

> Любая достаточно развитая технология неотличима от магии.
>
> *Артур К. Кларк*

Любой пользователь компьютера, наверное, полностью согласится с этой цитатой знаменитого писателя-фантаста. (Приведенное высказывание известно также под именем Третьего Закона Кларка.) С момента появления электронных компьютеров в начале 40-х гг. управляющее их работой программное обеспечение становилось все более сложным и изощренным. На сегодня оно достигло такой степени сложности, что его функционирование представляется недалеким от магии. Программное обеспечение компьютера приобрело такой объем, что никакой человек не может охватить его целиком. Не следует даже пытаться разобраться во всех деталях его работы.

Рассмотрим, однако, некоторую аналогию. Допустим, что мы изучаем юриспруденцию. Количество законов, принятых в стране, и сложность этих законов также превышают возможности их понимания конкретным человеком. Студент-юрист и не пытается запомнить все законы. Вместо этого он постарается изучить основные принципы права, построит некоторую систему знаний, касающихся юриспруденции в целом, освоит специфический язык, используемый в текстах законов, научится рассуждать и делать заключения на основе принципов права и, возможно, постарается

приобрести глубокие знания законодательства в одной конкретной области юриспруденции (уголовное право, право собственности, налоговое право и т. д.).

Изучая программное обеспечение компьютера, мы следуем тому же подходу. Мы рассматриваем базовые принципы организации программ и работы компьютера. Мы знакомимся с языком, на котором составляют программы, учимся применять принципы на практике и, возможно, становимся специалистами в одной узкой области использования программного обеспечения.

В этой книге мы концентрируем наше внимание на единственном языке программирования, называемом C# (произносится «си шарп»). Мы научимся использовать язык C# для разработки простых прикладных программ. По ходу дела мы также познакомимся с некоторыми сторонами функционирования компьютеров.

## 1.2 Роль языков программирования

### Компьютер

Компьютер представляет собой электронное устройство, состоящее из нескольких основных узлов. Одним из таких компонентов является память, которая хранит комбинации единиц и нолей. Эти единицы и ноли могут физически реализовываться в виде крошечных электронных устройств (в которых протекание тока в одном направлении означает единицу, а в другом — ноль, или устройство может содержать конденсатор, заряженное состояние которого означает единицу, а разряженное — ноль) или микроскопических магнитных полей (направление магнитного потока по часовой стрелке может означать единицу, а против часовой стрелки — ноль), или отражающей способности миниатюрных углублений в металлическом слое вращающегося компакт-диска, или различных состояний любого физического объекта. Возможности здесь неисчерпаемы.

Компьютер содержит блоки памяти разных видов, работающие по-разному и предназначенные для выполнения различных задач. К счастью, детали функционирования памяти компьютера обычно не важны для программиста. Программа манипулирует с комбинациями единиц и нолей безотносительно к их физической реализации. Программа использует эти комбинации для обозначения чисел в двоичной форме или букв алфавита, или любой другой информации в соответствии с нашей схемой ее представления.

Другим стандартным компонентом компьютера является центральный процессорный узел (central processor unit, CPU) или просто *процессор*. Процессор может выполнять элементарные действия над комбинациями единиц и нолей в памяти компьютера. Например, процессор может заглянуть в две ячейки компьютерной памяти, извлечь из них комбинации, представляющие два числа, сложить эти числа и сохранить результат в виде новой комбинации единиц и нолей в другой ячейке памяти.

Понятия, затронутые в гл.1:

| | |
|---|---|
| программирование | язык ассемблера |
| двоичная система | шестнадцатеричная система |
| кодирование символов | оператор присваивания |
| интерактивная среда разработки | запуск программы |

Рассмотренные операторы:

=

Определения и синтаксические формы:

| | |
|---|---|
| языки программирования | определение компьютера |
| команда csc | команды командной строки (выборка) |
| разработка программ | цикл while |

## Контрольные вопросы

Контрольные вопросы в этой книге снабжены альтернативными ответами. Приводимые ниже пять вопросов позволят вам проверить усвоение вами основных понятий этой главы.

**1.1.** Сколько может быть различных комбинаций битов в одном байте?

(а) 64 (б) 16

(в) 8 (г) 256

**1.2.** Если в регистре eax содержится число 7, то что, по вашему мнению, будет содержаться в этом регистре после выполнения приведенной ниже команды Intel Pentium?

```
add eax,eax
```

(а) невозможно определить (б) 7

(в) 14 (г) 8

**1.3.** Какое десятичное число представляет двоичное число 0011111?

(а) 63 (б) 31

(в) 111111 (г) 6

**1.4.** Если код ASCII буквы «A» равен 65, код «B» — 66 и т. д., какая строка текста представлена приведенной ниже последовательностью чисел:?

```
72  69  76  76  79
```

(а) hello (б) GEKKO

(в) OLLEH (г) HELLO

**1.5.** Если в системе Windows текущим является каталог C:\a\b\c\d, в какой каталог мы попадем после выполнения в командном окне приведенной ниже цепочки команд?

```
cd ..
cd ..
cd c
cd flub
```

(а) `C:\a\b\c\d\flub` (б) `C:\a\b\c\flub`

(в) `C:\a\c\flub` (г) `C:\c\flub`

## Упражнения

Упражнения заключаются в том, что вы должны что-то сделать. В последующих главах упражнения потребуют от вас составления или модификации C#-программ. Поскольку мы пока еще не дошли до практического программирования, предлагаемые ниже упражнения попроще.

**1.1.** Для каких языков программирования есть компиляторы на вашем компьютере? Перечислите все, которые вам удастся найти.

**1.2.** Является ли программа Command Prompt на компьютере с системой Windows компилятором? Обоснуйте ваш ответ.

**1.3.** Каждый символ (буква, цифра, знак препинания и т. д.) требуют в кодировке Unicode двух байтов памяти. Оцените, сколько потребуется байтов для хранения символов, из которых состоит вся эта книга.

**1.4.** Распознает ли ваш текстовый редактор синтаксис языков программирования с окраской различных элементов языка в разные цвета? Можете ли вы установить такой режим для C#?

**1.5.** Определите, сколько памяти имеет ваш компьютер и какой объем памяти занимают ваш компилятор C#, редактор и IDE.