

C# 4.0

НА ПРИМЕРАХ

НОВЫЕ ВОЗМОЖНОСТИ C# 4.0
И .NET 4.0.

ОТ ОСНОВ ЯЗЫКА ДО РЕШЕНИЯ
ТИПИЧНЫХ ЗАДАЧ

БОЛЕЕ 200 ПРОВЕРЕННЫХ
РЕШЕНИЙ, ОПТИМАЛЬНЫХ
ТЕХНИЧЕСКИХ ПРИЕМОВ
И ПРИМЕРОВ ПРОГРАММНОГО
КОДА

BEN WATSON

C# 4.0

HOW-TO

SAMS

800 East 96th Street, Indianapolis, Indiana 46240 USA

Бен Ватсон

C# 4.0

на примерах

Санкт-Петербург

«БХВ-Петербург»

2011

УДК 681.3.068+800.92С#
ББК 32.973.26-018.1
В21

Ватсон Б.

В21 С# 4.0 на примерах. — СПб.: БХВ-Петербург, 2011. — 608 с.: ил.
ISBN 978-5-9775-0608-3

На практических примерах рассмотрено программирование на языке Microsoft C# 4.0, начиная с основ языка и заканчивая решением типичных задач с помощью .NET Framework. Показано создание эффективных классов, интерфейсов и типов, а также программного кода, допускающего многократное использование. Описаны приемы обработки данных, основанные на применении коллекций, сериализации, баз данных и XML. Рассмотрена реализация пользовательского интерфейса с применением технологий WinForms и WPF, а также создание веб-приложений на основе технологий ASP.NET и Silverlight. Показано применение на практике новых возможностей языка C# 4.0. Уделено внимание взаимодействию с ОС Windows и системным ПО, использовано шаблонов для разработки сложных программ и др. Приведено более 200 готовых решений, оптимальных технических приемов и примеров проверенного кода.

Для программистов

УДК 681.3.068+800.92С#
ББК 32.973.26-018.1

Группа подготовки издания:

Главный редактор	<i>Екатерина Кондукова</i>
Зам. главного редактора	<i>Игорь Шишигин</i>
Зав. редакцией	<i>Григорий Добин</i>
Перевод с английского	<i>Сергея Иноземцева</i>
Редактор	<i>Ирина Иноземцева</i>
Компьютерная верстка	<i>Натальи Караваевой</i>
Корректор	<i>Виктория Пиотровская</i>
Дизайн серии	<i>Игоря Цырульникова</i>
Оформление обложки	<i>Елены Беляевой</i>
Зав. производством	<i>Николай Тверских</i>

Authorized translation from the English language edition, entitled C# 4.0 How-To, ISBN 978-0-672-33063-6, by Ben Watson, published by Pearson Education, Inc., Copyright © 2010 by Pearson Education. All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without the prior permission from the publisher. RUSSIAN language edition published by BHV St. Petersburg, Copyright © 2010.

Авторизованный перевод английской редакции C# 4.0 How-To, изданной Pearson Education, Inc., Copyright © 2010 by Pearson Education. Все права защищены. Никакая часть настоящей книги не может быть воспроизведена или передана в какой бы то ни было форме и какими бы то ни было средствами, будь то электронные или механические, включая фотокопирование, запись на магнитный носитель, размещена в любых базах данных и информационно-поисковых системах без предварительного письменного разрешения издателя. Перевод на русский язык "БХВ-Петербург" © 2010.

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 30.09.10.

Формат 70×100¹/₁₆. Печать офсетная. Усл. печ. л. 49,02.

Тираж 1700 экз. Заказ №

"БХВ-Петербург", 190005, Санкт-Петербург, Измайловский пр., 29.

Санитарно-эпидемиологическое заключение на продукцию
№ 77.99.60.953.Д.005770.05.09 от 26.05.2009 г. выдано Федеральной службой
по надзору в сфере защиты прав потребителей и благополучия человека.

Отпечатано с готовых диапозитивов
в ГУП "Типография "Наука"
199034, Санкт-Петербург, 9 линия, 12.

ISBN ISBN 978-0-672-33063-6 (англ.)
ISBN 978-5-9775-0608-3 (рус.)

© 2010 by Pearson Education, Inc.
© Оформление, издательство "БХВ-Петербург", 2010

Оглавление

Об авторе	2
Благодарности	3
Введение.....	5
Краткий обзор книги.....	5
Как извлечь максимум пользы из этой книги	5
Как углубить и расширить свои знания.....	7
ЧАСТЬ I. ОСНОВЫ ПРОГРАММИРОВАНИЯ НА ЯЗЫКЕ C#	9
Глава 1. Основы работы с типами	11
Создание класса	11
Определение полей, свойств и методов.....	12
Определение статических членов.....	14
Написание конструктора	14
Инициализация свойств при конструировании.....	15
Применение модификаторов <i>const</i> и <i>readonly</i>	16
Повторное использование кода в нескольких конструкторах.....	16
Создание производного класса.....	17
Вызов конструктора базового класса.....	18
Переопределение метода или свойства базового класса	18
Создание интерфейса.....	21
Реализация интерфейсов	22
Создание структуры.....	23
Создание анонимного типа	24
Предотвращение создания экземпляра с помощью абстрактного класса	25
Интерфейс или абстрактный базовый класс?.....	25
Глава 2. Создание типов с разносторонней функциональностью.....	27
Форматирование типа методом <i>ToString()</i>	27
Создание типов, допускающих выяснение равенства.....	31

Создание типов, хешируемых методом <i>GetHashCode()</i>	32
Создание сортируемых типов.....	33
Создание индекса у типов.....	34
Уведомление клиентов об изменении состояния объекта.....	36
Перегрузка операций.....	37
Преобразование одного типа в другой.....	38
Предотвращение наследования.....	40
Разрешение типу значения принимать значение <i>null</i>	40
Глава 3. Общие принципы кодирования.....	42
Объявление переменных.....	42
Откладывание проверки типов до этапа выполнения (динамические типы).....	43
Применение динамической типизации для упрощения взаимодействия с COM.....	45
Объявление массивов.....	46
Создание многомерных массивов.....	46
Создание псевдонима для пространства имен.....	47
Применение условной операции (?:).....	48
Применение операции проверки на <i>null</i> (??).....	48
Добавление методов в существующие типы с использованием методов расширения.....	49
Вызов методов с параметрами по умолчанию.....	51
Вызов методов с именованными параметрами.....	51
Откладывание вычисления значения до момента обращения к нему.....	52
Создание контрактов кода.....	53
Глава 4. Исключения.....	57
Возбуждение исключения.....	57
Обработка исключения.....	57
Обработка нескольких исключений.....	58
Повторное возбуждение исключения.....	59
Практически гарантированное выполнение кода с использованием блока <i>finally</i>	60
Получение информации от исключения.....	61
Создание собственного исключения.....	63
Перехват необработанных исключений.....	65
Советы по работе с исключениями.....	69
Глава 5. Числа.....	70
Выбор между типами <i>float</i> , <i>double</i> и <i>decimal</i>	70
Работа с очень большими числами (тип <i>BigInteger</i>).....	71
Работа с комплексными числами.....	72
Форматирование числа в строку.....	74
Преобразование строки в число.....	78
Преобразование числа из одной системы счисления в другую.....	79

Преобразование числа в байты (и обратно)	81
Выяснение четности числа.....	82
Выяснение, является ли число степенью двойки.....	83
Выяснение, является ли число простым	83
Подсчет количества установленных битов.....	84
Пересчет градусов в радианы	84
Округление	85
Генерирование "хороших" случайных чисел	87
Генерирование уникальных идентификационных номеров (GUID).....	88
Глава 6. Перечисления	90
Объявление перечисления.....	90
Объявление флагов в виде перечисления.....	91
Выяснение, установлен ли флаг	92
Преобразование перечисления в целое (и обратно)	92
Проверка допустимости значений перечисления	93
Получение списка значений перечисления	93
Преобразование строки в перечисление	93
Добавление метаданных к перечислению с помощью методов расширения	94
Советы по использованию перечислений.....	96
Глава 7. Строки.....	97
Преобразование строки в байтовое представление (и обратно).....	97
Создание собственной схемы кодирования	98
Корректное сравнение строк.....	102
Корректная смена регистра.....	103
Распознавание пустых строк.....	103
Конкатенация строк: обязательно ли использовать класс <i>StringBuilder</i> ?	104
Конкатенация элементов коллекции в одну строку	106
Добавление символа новой строчки	107
Разбивка строки.....	107
Преобразование двоичных данных в строку (кодировка base-64).....	109
Изменение порядка слов на обратный	110
Естественная сортировка строк	111
Глава 8. Регулярные выражения.....	116
Поиск в тексте	116
Извлечение фрагментов текста.....	117
Замена фрагмента текста.....	117
Проверка допустимости	118
Повышение производительности регулярных выражений	120

Глава 9. Универсальные типы.....	121
Создание универсального списка.....	121
Создание универсального метода.....	122
Создание универсального интерфейса.....	123
Создание универсального класса.....	124
Создание универсального делегата.....	125
Работа с несколькими универсальными типами.....	126
Накладывание ограничений на универсальный тип.....	127
Преобразование <i>IEnumerable<string></i> в <i>IEnumerable<object></i> (ковариантность).....	129
Преобразование <i>IComparer<Child></i> в <i>IComparer<Parent></i> (контравариантность).....	130
Создание кортежей (пар, троек и т. д.).....	131
ЧАСТЬ II. ОБРАБОТКА ДАННЫХ.....	133
Глава 10. Коллекции.....	135
Выбор подходящего класса-коллекции.....	135
Инициализация коллекции.....	137
Перебор элементов коллекции без привязки к ее реализации.....	137
Создание собственной коллекции.....	138
Создание собственных итераторов для коллекции.....	142
Изменение порядка элементов массива на обратный.....	145
Изменение порядка элементов связанного списка на обратный.....	146
Извлечение уникальных элементов из коллекции.....	147
Подсчет количества вхождений элемента.....	147
Реализация очереди с приоритетами.....	148
Создание префиксного дерева.....	152
Глава 11. Файлы и сериализация.....	156
Создание, чтение и запись файла.....	156
Удаление файла.....	159
Комбинирование потоков данных (сжатие файла).....	159
Выяснение размера файла.....	161
Получение информации, связанной с безопасностью.....	161
Проверка существования файла и каталога.....	162
Список дисков.....	163
Список каталогов и файлов.....	164
Обзор каталогов.....	165
Поиск файла или каталога.....	165
Манипуляции с путями к файлам.....	167
Создание уникальных или временных имен для файлов.....	169
Отслеживание изменений в файловой системе.....	169

Получение пути к каталогам My Documents, My Pictures и т. д.	171
Сериализация объектов	171
Сериализация в буфер в памяти	174
Хранение данных приложения, имеющего ограниченные права	175
Глава 12. Работа в сетях и во Всемирной паутине	177
Определение IP-адреса по имени хоста	177
Выяснение имени хоста и IP-адреса у данного компьютера	178
"Пингование" компьютера	178
Выяснение информации о сетевой карте	179
Создание сервера и клиента на базе TCP/IP	179
Отправка электронного письма по протоколу SMTP	183
Загрузка содержимого веб-страницы по протоколу HTTP	184
Выгрузка файла по протоколу FTP	188
Удаление тегов из HTML-кода	188
Встраивание веб-браузера в приложение	189
Прием RSS-ленты новостей	191
Динамическое генерирование RSS-ленты новостей в IIS	194
Взаимодействие между процессами на одном компьютере (WCF)	196
Взаимодействие между двумя компьютерами в одной сети (WCF)	203
Взаимодействие через Интернет (WCF)	204
Обнаружение служб на этапе выполнения (WCF)	206
Глава 13. Базы данных	210
Создание базы данных в Visual Studio	210
Соединение с базой и чтение данных	212
Добавление данных в таблицу	218
Удаление данных из таблицы	219
Выполнение хранимой процедуры	219
Транзакции	220
Связывание данных с элементом управления при помощи класса <i>DataSet</i>	222
Выяснение доступности соединения с базой данных	230
Автоматическое отображение данных на объекты с помощью платформы Entity Framework	231
Глава 14. Язык XML	234
Сериализация объекта в XML и десериализация его	234
Создание XML-документа "с нуля"	238
Чтение XML-файла	240
Проверка корректности XML-документа	242
Выдача запроса к XML-документу с помощью XPath	243
Преобразование информации из базы данных в XML-документ	245
Преобразование XML-документа в HTML-документ	246

ЧАСТЬ III. Взаимодействие с пользователем.....	249
Глава 15. Делегаты, события и анонимные методы.....	251
Динамический вызов метода	251
Подписка на событие.....	253
Публикация события.....	254
Гарантия обновления пользовательского интерфейса в потоке пользовательского интерфейса	256
Присваивание анонимного метода делегату	258
Использование анонимных методов в качестве простых обработчиков событий	259
Использование преимуществ контравариантности	261
Глава 16. Технология Windows Forms.....	264
Создание модальных и немодальных форм	264
Добавление строки меню	265
Динамический перевод пунктов меню в неактивное состояние	268
Добавление строки состояния	268
Добавление панели инструментов.....	269
Создание интерфейса, включающего в себя подокно	270
Наследование формы	271
Создание собственного элемента управления.....	275
Применение таймера.....	280
Использование общих и пользовательских настроек конфигурации	281
Эффективное использование элемента <i>List View</i> в виртуальном режиме	284
Наклон колесика мыши для горизонтальной прокрутки	286
Реализация <i>Cut</i> и <i>Paste</i>	290
Автоматический сброс индикатора ожидания	295
Глава 17. Графика с применением Windows Forms и GDI+.....	296
Определение цвета	296
Использование системного элемента управления для выбора цвета.....	297
Преобразование цветов между системами RGB и HSV	297
Рисование фигур	301
Создание перьев	304
Создание кистей с произвольными характеристиками	306
Преобразования.....	308
Рисование текста	310
Расположение текста по диагонали	310
Вывод изображений	310
Вывод прозрачных изображений.....	311
Рисование в буфере.....	312

Прямое обращение к пикселям для повышения эффективности	312
Рисование со сглаживанием.....	314
Перерисовка без мерцания.....	315
Изменение размеров изображения.....	316
Создание миниатюры	316
Захват многоэкранного изображения.....	318
Вычисление расстояния от указателя мыши до заданной точки	320
Выяснение местоположения точки относительно прямоугольника	320
Выяснение местоположения точки относительно круга.....	321
Выяснение местоположения точки относительно эллипса	321
Выяснение факта пересечения двух прямоугольников.....	322
Печать и предварительный просмотр	322

Глава 18. WPF..... 328

Открытие окна.....	328
Выбор компоновки интерфейса.....	329
Добавление строки меню	330
Добавление строки состояния	331
Добавление панели инструментов.....	332
Использование стандартных команд	333
Использование нестандартных команд.....	333
Перевод команд из неактивного состояния в активное и обратно.....	336
Сворачивание и разворачивание группы элементов управления.....	337
Реагирование на события	338
Отделение внешнего вида от функциональности.....	339
Применение триггеров для смены стилей на этапе выполнения	340
Связывание свойств элемента управления с другим объектом.....	341
Форматирование значений при связывании данных	346
Преобразование типов значений при связывании данных	346
Связывание с коллекцией.....	348
Контроль за представлением связанных данных.....	348
Определение внешнего вида элементов управления с помощью шаблонов	349
Анимирование свойств элементов	351
Отображение трехмерной графики	352
Размещение видео на поверхности трехмерной фигуры	355
Размещение интерактивных элементов управления на поверхности трехмерной фигуры	358
Применение WPF в приложении WinForms.....	362
Применение WinForms в WPF-приложении.....	363

Глава 19. ASP.NET..... 364

Просмотр отладочной и трассировочной информации.....	364
Выяснение возможностей браузера	366

Перенаправление на другую страницу	367
Аутентификация с помощью формы для входа пользователя в систему	368
Использование главных страниц для достижения единства оформления	372
Добавление меню	373
Связывание данных с элементом управления <i>GridView</i>	374
Создание пользовательского элемента управления	376
Создание гибкого пользовательского интерфейса с элементами Web Parts	380
Создание простой страницы с использованием технологии AJAX	385
Проверка допустимости данных	387
Поддержание состояния приложения	392
Поддержание состояния пользовательского интерфейса	393
Поддержание пользовательских данных на протяжении сеанса	393
Сохранение состояния сеанса	395
Восстановление состояния сеанса с помощью cookie	396
Использование надстройки MVC над ASP.NET	398

Глава 20. Silverlight..... 404

Создание проекта Silverlight	404
Воспроизведение видео	405
Создание индикатора загрузки и воспроизведения	409
Реакция пользовательского интерфейса на события таймера	411
Создание трехмерной перспективы для содержимого страницы	413
Выполнение приложения за пределами браузера	414
Захват изображения с веб-камеры	415
Распечатка документа	417

ЧАСТЬ IV. Более сложные элементы языка C#..... 419

Глава 21. LINQ 421

Запрос к коллекции объектов	421
Упорядочивание результатов	423
Фильтрация коллекции	423
Получение коллекции на основе отдельных полей объектов (проекция)	424
Выполнение объединения	424
Запрос к XML-документу	425
Создание XML-документа	426
Запрос к Entity Framework	426
Запрос к веб-службе (LINQ to Bing)	428
Ускорение запросов с помощью PLINQ (Parallel LINQ)	430

Глава 22. Управление памятью 432

Измерение объема памяти, нужного приложению	432
Освобождение неуправляемых ресурсов с помощью финализации	433

Освобождение управляемых ресурсов с помощью шаблона Dispose.....	435
Принудительная сборка мусора.....	440
Создание кэша, позволяющего выполнять сборку мусора	440
Работа с указателями	443
Ускорение доступа к массивам.....	444
Предотвращение перемещения объектов в памяти	445
Выделение неуправляемой памяти.....	446

Глава 23. Потоки выполнения. Асинхронное и параллельное программирование..... 448

Распределение работы между несколькими процессорами.....	448
Использование структур данных в разных потоках выполнения	452
Асинхронный вызов метода.....	452
Работа с пулом потоков выполнения	454
Создание потока выполнения	454
Обмен данными с потоком выполнения	455
Защита данных, используемых в нескольких потоках выполнения	456
Применение методов класса <i>Interlocked</i> вместо блокировок	459
Защита данных в нескольких процессах	460
Ограничение количества экземпляров приложения до одного	461
Ограничение количества потоков выполнения, обращающихся к ресурсу	462
Отправка сигналов потокам выполнения с помощью механизма событий	464
Использование многопоточкового таймера	468
Использование блокировки чтения/записи	468
Применение асинхронной модели программирования	470

Глава 24. Применение отражения и создание добавляемых модулей..... 474

Перечисление типов в сборке	474
Добавление нестандартного атрибута	475
Динамическое создание экземпляра класса	477
Вызов метода динамически созданного экземпляра класса	477
Реализация архитектуры с добавляемыми модулями	478

Глава 25. Шаблоны приложений и полезные советы по проектированию 483

Применение секундомера для профилирования кода	483
Пометка устаревшего кода.....	484
Объединение нескольких событий в одно.....	485
Реализация шаблона с наблюдателем (подписчиком)	489
Использование брокера событий.....	492
Запоминание местоположения на экране	495
Реализация отмены с помощью командных объектов	497
Применение модели Model-View-ViewModel в технологии WPF	504
Локализация.....	514

Локализация приложения Windows Forms	515
Локализация приложения ASP.NET	516
Локализация WPF-приложения	518
Локализация Silverlight-приложения	522
Развертывание приложений с использованием ClickOnce	524
Глава 26. Взаимодействие с операционной системой и аппаратной частью	527
Получение информации об операционной системе, пакете обновления и версии CLR	527
Получение информации о ЦПУ и других аппаратных средствах	528
Вызов UAC для запрашивания прав администратора	530
Запись в журнал событий	533
Обращение к Реестру	534
Управление службами Windows	536
Создание службы Windows	536
Вызов низкоуровневых функций Windows с помощью P/Invoke	539
Вызов библиотечных функций, написанных на языке C, из кода на языке C#	540
Работа с файлами, отображенными в память	541
Обеспечение работы приложения в 32-битовом и 64-битовом окружении	542
Реагирование на изменение системной конфигурации	544
Использование некоторых возможностей Windows 7	544
Получение информации о режиме питания	546
Глава 27. Полезные мелочи	547
Создание прямоугольного окна	547
Создание информационной пиктограммы	551
Создание хранителя экрана в WPF	555
Вывод заставки	563
Воспроизведение звукового файла	568
Перетасовка карт	569
Приложение. Необходимые инструменты	571
Утилита Reflector	571
Среда NUnit	572
Программа NDepend	575
Программа FxCop	576
Виртуальная машина	577
Утилиты Process Explorer и Process Monitor	578
Программа RegexBuddy	579
Приложение LINQPad	580
Поиск инструментальных средств	581
Предметный указатель	583

Моим родителям, Майку и Диане Ватсон, которые с пониманием отнеслись к моей ребяческой просьбе купить компилятор C++ и всемерно поощряли мои интересы и развивали мои способности. Я ничего бы не добился без их личного примера, любви и советов.

Моей замечательной жене Летиции, проявившей громадное терпение и большую любовь, когда я взялся за этот новый и пугающий проект на фоне перемен, произошедших в нашей жизни.

Об авторе

Бен Ватсон (Ben Watson) программирует на платформе .NET Framework с первых дней ее существования. До перехода в корпорацию Microsoft он работал ведущим программистом в GeoEye, компании по обработке спутниковых изображений, где создавал системы морской связи на платформе .NET. Последнее время он трудился в команде разработчиков Query Pipeline для Bing, участвуя в реализации масштабируемых распределенных систем и других компонентов этой поисковой службы.

Благодарности

Я очень благодарен тем, кто приложил все свои усилия, помогая мне работать над этой книгой.

Бруку Фарлингу (Brook Farling), редактору, который нашел меня и предоставил мне возможность написать книгу, я благодарен за то, что он провел большую координаторскую работу, проявил огромное терпение и вложил много энергии в доведение дела до конца.

Марку Стромайеру (Mark Strawmyer), техническому редактору, я говорю спасибо за то, что он прочитал внушительный объем текста и кода с целью обеспечения высокого качества материала.

Благодарю всю команду издательства Sams, участвовавшую в подготовке книги к публикации. Это Марк Ренфроу (Mark Renfrow), Лори Лайонз (Lori Lyons), Барт Рид (Bart Reed), Нони Рэтклиф (Nonie Ratcliff) и Шери Кейн (Sheri Cain).

Введение

Краткий обзор книги

Подход, предпринятый в этой книге, весьма далек от "библейского", охватывающего максимум материала по теме. Книга имеет структуру "сборника рецептов", то есть состоит из типичных задач и легко воспроизводимых способов их решения. Как автор, я стремился свести к минимуму теоретические пояснения и сосредоточиться преимущественно на самом коде. Многие неочевидные моменты разъясняются прямо в комментариях к коду.

Строго говоря, книга не является описанием языка. Наряду с обсуждением функциональных возможностей языка материал включает в себя конкретные примеры приложений, полезные алгоритмы и советы, применимые во многих ситуациях.

Программисты, как начинающие, так и опытные, найдут в книге сотни интересных тем, таких как малоизвестные операторы C#, сортировка строк, содержащих числа, или реализация функции "Отмена" в пользовательском интерфейсе. И каждая тема сопровождается программными решениями, которые пригодятся в самых разных ситуациях, независимо от профессионального уровня разработчика.

Короче говоря, я написал книгу, которую сам хотел бы иметь под рукой в то время, когда еще изучал основы программирования на языке C#, и сейчас, когда мне требуется получить справку или освежить в памяти тот или иной технический прием.

Как извлечь максимум пользы из этой книги

Мы создали книгу, которую легко читать "от корки до корки". Наша цель — дать читателю полное представление о языке C# 4.0. Материал книги поделен на четыре части, содержащие главы, которые легко читать и среди которых легко найти нужную.

Часть I, "Основы программирования на языке C#", посвящена наиболее общим функциональным возможностям языка C#, которые важны при решении любых программистских задач. Хотя приведенные сведения могут показаться элементарными, в главах этой части содержится множество полезных советов, помогающих досконально овладеть темой.

□ Глава 1. "Основы работы с типами"

□ Глава 2. "Создание типов с разносторонней функциональностью"

- Глава 3. "Общие принципы кодирования"
- Глава 4. "Исключения"
- Глава 5. "Числа"
- Глава 6. "Перечисления"
- Глава 7. "Строки"
- Глава 8. "Регулярные выражения"
- Глава 9. "Универсальные типы"

Часть II, "Обработка данных", содержит сведения о том, как хранить и обрабатывать данные, в том числе и полученные из Интернета.

- Глава 10. "Коллекции"
- Глава 11. "Файлы и сериализация"
- Глава 12. "Работа в сетях и во Всемирной паутине"
- Глава 13. "Базы данных"
- Глава 14. "Язык XML"

Часть III, "Взаимодействие с пользователем", затрагивает наиболее популярные парадигмы пользовательского интерфейса на платформе .NET, причем речь идет как о локально работающих программах, так и о веб-приложениях.

- Глава 15. "Делегаты, события и анонимные методы"
- Глава 16. "Технология Windows Forms"
- Глава 17. "Графика с применением Windows Forms и GDI+"
- Глава 18. "WPF"
- Глава 19. "ASP.NET"
- Глава 20. "Silverlight"

Часть IV, "Более сложные элементы языка C#", содержит материал о нетривиальных возможностях языка C#, позволяющих перевести ваши приложения на более высокий уровень в том, что касается их производительности, шаблонов проектирования, полезных алгоритмов и т. д.

- Глава 21. "LINQ"
- Глава 22. "Управление памятью"
- Глава 23. "Потоки выполнения. Асинхронное и параллельное программирование"
- Глава 24. "Отражение и создание добавляемых модулей"
- Глава 25. "Шаблоны приложений и полезные советы по проектированию"
- Глава 26. "Взаимодействие с операционной системой и аппаратной частью"
- Глава 27. "Полезные мелочи"
- Приложение. "Необходимые инструменты"

Весь приведенный код был создан в предварительных выпусках Visual Studio 2010, но в большинстве случаев вы можете использовать более ранние версии, особенно для написания кода, не связанного с .NET 4. Если у вас нет Visual Studio, вы можете загрузить издание Express Edition, доступное по адресу www.microsoft.com/express/default.aspx. Эта версия позволит построить почти любой пример, код которого приведен в книге.

Вы можете зарегистрироваться на сайте книги. Зайдите на страницу informit.com/register, выполните процедуру бесплатной регистрации и укажите ISBN-номер книги¹. Затем загляните на страницу **Account** (Учетная запись) и в разделе **Registered Products** (Зарегистрированные товары) найдите ссылку **Access Bonus Content** (Получить доступ к бонусному содержанию). Вы получите доступ к обновлениям и загрузочным файлам, если таковые имеются для этой книги, а также к списку замеченных ошибок.

Как углубить и расширить свои знания

Невозможно написать книгу, охватывающую все, что связано с С# и .NET Framework. Пожалуй, даже нельзя надеяться на полное раскрытие хотя бы одной темы из этой области. Если такая книга все-таки была бы написана, то никто не смог бы поднять ее и, тем более, прочитать за разумное время.

Когда вы овладеете основами, в вашем распоряжении окажется множество ресурсов, где вы найдете ответы на возникающие вопросы и получите более глубокие знания о платформе .NET.

К счастью, MSDN-документация по платформе .NET написана очень хорошо (она доступна по адресу <http://msdn.microsoft.com/en-us/library/aa139615.aspx>). Большинство ее разделов включает в себя код и разъяснения по его использованию. В качестве дополнительного бонуса в конце многих разделов можно найти полезные советы, оставленные другими разработчиками.

Форумы разработчиков .NET (<http://social.msdn.microsoft.com/Forums/en-US/category/netdevelopment>³) — это отличное место для того, чтобы задать вопрос. Вам ответят эксперты, многие из которых на практике занимались разработкой и тестированием .NET.

Кроме того, я нашел в Интернете отличное место, где можно получить ответы на интересующие вас вопросы: **StackOverflow.com**.

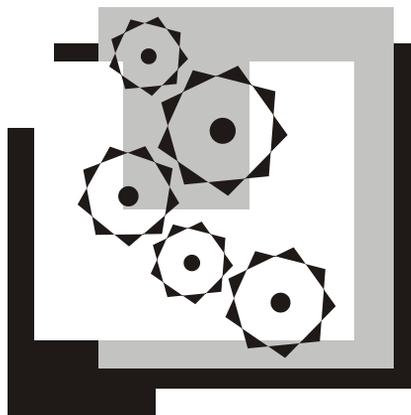
На мой взгляд, самым лучшим советом по поводу расширения и углубления знаний будет такой: пишите программы. Занимайтесь этим постоянно, обдумывайте новые проекты, применяйте новые технологии, стремитесь выйти за пределы своих возможностей. Вообще говоря, книги полезны лишь до определенного момента, после чего вы должны будете приступить к написанию кода. И тогда эта книга станет надежным справочным руководством, незаменимым в тех случаях, когда вы не знаете, как подступиться к решению задачи.

Желаю вам успехов в программировании.

¹ Сайт informit.com англоязычный. Следует указывать ISBN-номер оригинального издания: 978-0-672-33063-6 — *прим. ред.*

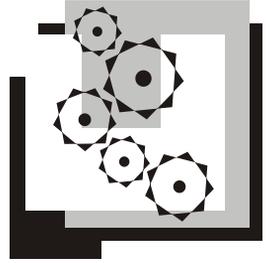
² Документация на русском языке находится по адресу <http://msdn.microsoft.com/ru-ru/library/aa139615.aspx> — *прим. перев.*

³ Русскоязычный форум: <http://social.msdn.microsoft.com/Forums/ru-RU/category/netdevelopment> — *прим. перев.*



ЧАСТЬ I

**Основы программирования
на языке C#**



Глава 1

Основы работы с типами

В этой главе разъясняются основы создания типов в языке C#. Если вы уже знакомы с ним, большая часть этой главы не содержит ничего нового для вас.

После обсуждения членов класса, таких как поля, свойства и методы, вы прочитаете о конструкторах, о том, как создавать и реализовывать интерфейсы, и о том, когда следует использовать структуры.

Информация в этой главе элементарна, но очень важна. Как и во многих ситуациях, если вы не овладеете основами, у вас ничего не получится.

Создание класса

Задача. Вам необходимо написать объявление класса.

Решение. Начнем с объявления простого класса, содержащего координаты точки в трехмерном пространстве:

```
// Необходимо импортировать пространства имен по умолчанию,  
// которые Visual Studio включит в каждый файл  
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
  
namespace ClassSample  
{  
    // Использован модификатор public,  
    // чтобы класс был виден за пределами сборки  
    public class Vertex3d  
    {  
    }  
}
```

Класс, который мы объявили, пока пуст, но мы заполним его по ходу этой главы.

Класс определен как открытый, то есть он виден каждому типу, ссылающемуся на сборку. В языке C# существует несколько модификаторов доступа, и они собраны в табл. 1.1.

Таблица 1.1. Модификаторы доступа

Модификатор	Применяется к	Описание
Public	типам, членам	доступен любому коду, даже за пределами сборки
Private	типам, членам	доступен коду в том же типе
Internal	типам	доступен коду в той же сборке
Protected	членам	доступен коду в том же или в производном типе
Protected internal	членам	доступен коду в той же сборке или в производном классе в другой сборке

Если видимость класса не указана, по умолчанию предполагается модификатор `internal`.

Определение полей, свойств и методов

Задача. Вам нужно добавить поля, свойства и методы в определение класса.

Решение. Придадим некоторую осмысленность классу `Vertex3d`:

```
public class Vertex3d
{
    // Поля
    private double _x;
    private double _y;
    private double _z;
    // Свойства
    public double X
    {
        get { return _x; }
        set { _x = value; }
    }
    public double Y
    {
        get { return _y; }
        set { _y = value; }
    }
    public double Z
    {
        get { return _z; }
        set { _z = value; }
    }
    // Метод
```

```
public void SetToOrigin()
{
    X = Y = Z = 0.0;
}
}
```

Сделаем несколько замечаний относительно этого кода.

- ❑ Поля объявлены с модификатором `private`, что, вообще говоря, является хорошим стилем.
- ❑ Свойства объявлены как `public`, но могли быть `private`, `protected` или `protected internal`, по желанию программиста.
- ❑ Свойства могут включать в себя метод `get` или `set` (или оба).
- ❑ В свойствах (то есть в коде) подразумевается наличие аргумента `value`.

В следующем примере значение `13.0` будет передано свойству `x` через аргумент `value`:

```
Vertex3d v = new Vertex3d();
v.X = 13.0;
```

Применение автоматически реализуемых свойств

Вы часто будете встречать код такого вида:

```
class MyClass
{
    private int _field = 0;
    public int Field { get { return _field; } set { _field = value; } }
}
```

В языке `C#` для этой конструкции имеется сокращенный вариант:

```
class MyClass
{
    public int Field {get; set;}
    // Теперь необходимо инициализировать value в конструкторе
    public MyClass()
    {
        this.Field = 0;
    }
}
```

ПРИМЕЧАНИЕ

Нельзя иметь автоматически реализуемое свойство при наличии только метода `get` (в самом деле, если нет поля с резервным значением, как вы установите свойство?). Однако можно иметь закрытый (`private`) метод `set`:

```
public int Field { get ; private set; }
```

Определение статических членов

Задача. Вам нужно определить данные или методы, применимые к типу, а не отдельным экземплярам. Они часто используются в методах, действующих на несколько экземпляров типа.

Решение. Воспользуйтесь модификатором `static`, как сделано в следующем методе для создания двух объектов `Vertex3d`:

```
public class Vertex3d
{
    ...
    public static Vertex3d Add(Vertex3d a, Vertex3d b)
    {
        Vertex3d result = new Vertex3d();
        result.X = a.X + b.X;
        result.Y = a.Y + b.Y;
        result.Z = a.Z + b.Z;
        return result;
    }
}
```

Этот статический метод вызывается так:

```
Vertex3d a = new Vertex3d(0,0,1);
Vertex3d b = new Vertex3d(1,0,1);
Vertex3d sum = Vertex3d.Add(a, b);
```

Написание конструктора

Задача. Вам необходима автоматическая инициализация создаваемых объектов класса.

Решение. Определите специальный метод, называемый *конструктором*. Он должен носить то же имя, что и класс, но не возвращать никакого значения. Конструктор работает во время создания типа, — он никогда не вызывается напрямую.

Рассмотрим два конструктора для класса `Vertex3d`: один имеет аргументы, а другой выполняет инициализацию значением по умолчанию.

```
class Vertex3d
{
    public Vertex3d()
    {
        _x = _y = _z = 0.0;
    }
    public Vertex3d(double x, double y, double z)
    {
        this._x =x;
```

```
    this._y = y;
    this._z = z;
}
}
```

ПРИМЕЧАНИЕ

Конструкторам необязательно быть открытыми. Например, можно иметь защищенный (`protected`) конструктор, доступный только из производных классов. Более того, вы можете создать закрытый конструктор, предотвращающий создание экземпляров (для утилитных классов), или доступный только из других методов того же класса (возможно, статических генерирующих методов).

Создание статического конструктора и выполнение инициализации

Задача. У класса есть статические данные, подлежащие инициализации.

Решение. Статические поля можно инициализировать двумя способами. Первый заключается в использовании статического конструктора, аналогичного стандартному, но лишенного модификатора доступа и аргументов:

```
public class Vertex3d
{
    private static int _numInstances;
    static Vertex3d()
    {
        _numInstances = 0;
    }
    ...
}
```

Однако для повышения производительности предпочтительнее инициализировать статические поля в коде, когда это возможно:

```
public class Vertex3d
{
    private static int _numInstances = 0;
    ...
}
```

Инициализация свойств при конструировании

Задача. Вы хотите инициализировать свойства класса в момент создания переменной, даже если у конструктора нет для этого аргументов.

Решение. Воспользуйтесь синтаксисом инициализации объекта, как показано в следующем примере:

```
class Person
{
```

```

public int Id { get; set; }
public string Name { get; set; }
public string Address { get; set; }
}
...
Person p = new Person()
    { Id = 1, Name = "Ben", Address = "Redmond, WA" };

```

Применение модификаторов *const* и *readonly*

Задача. Требуется определить поля, значения которых не могут быть изменены на этапе выполнения.

Решение. Для определения неизменяющихся данных служат модификаторы `const` и `readonly`. Между ними есть важное различие: поля с модификатором `const` должны быть определены при объявлении. По этой причине и в силу факта неизменности их значений подразумевается, что они принадлежат типу как статические поля. Что касается полей с модификатором `readonly`, они могут быть заданы при объявлении или в конструкторе, но больше нигде.

```

public class Vertex3d
{
    private const string Name = "Vertex";
    private readonly int ver;
    public Vertex3d()
    {
        ver = Config.MyVersionNumber; // Ok
    }
    public void SomeFunction()
    {
        ver = 13; // Ошибка!
    }
}

```

Повторное использование кода в нескольких конструкторах

Задача. У вас имеется несколько конструкторов, обладающих сходной функциональностью в некоторых частях. Вы хотите избежать дублирования кода. В языке C++ и в некоторых старых языках нередко возникала ситуация, в которой класс с несколькими конструкторами должен был вызывать один и тот же код инициализации. Тогда общий код обычно оформлялся в виде функции, которую вызывал каждый конструктор.

```

// Пример на языке C++
class MyCppClass

```

```
{
public:
    MyCppClass() { Init(); }
    MyCppClass(int arg) { Init(); }
private:
    void Init() { /* Здесь код инициализации*/ };
}
```

Решение. В языке C# можно вызывать другие конструкторы в пределах класса, для чего используется ключевое слово `this`:

```
public class Vertex3d
{
    public Vertex3d(double x, double y, double z)
    {
        this._x = x;
        this._y = y;
        this._z = z;
    }
    public Vertex3d(System.Drawing.Point point)
        :this(point.X, point.Y, 0)
    {
    }
    ...
}
```

Создание производного класса

Задача. Вам нужно специализировать класс, добавив новые или переопределив имеющиеся функциональные возможности.

Решение. Добавьте новую функциональность к коду базового класса с помощью наследования.

```
public class BaseClass
{
    private int _a;
    protected int _b;
    public int _c;
}
public class DerivedClass : BaseClass
{
    public DerivedClass()
    {
        _a = 1; // Так нельзя! Переменная имеет модификатор private в BaseClass
        _b = 2; // ок
    }
}
```

```
    _c = 3; // ok
}
public void DoSomething()
{
    _c = _b = 99;
}
}
```

При создании производного класса сохраняется доступ к открытым и защищенным членам базового класса, но не к его закрытым членам.

Вызов конструктора базового класса

Задача. У вас есть класс-потомок, и вы хотите, чтобы его конструктор вызывал конкретный конструктор базового класса.

Решение. Аналогично тому, как из конструктора класса вызываются другие конструкторы, можно вызывать и конструкторы базового класса. Если конструктор не уточняется, будет вызван конструктор базового класса, установленного по умолчанию. Если он имеет аргументы, их нужно будет указать.

```
public class BaseClass
{
    public BaseClass(int x, int y, int z)
    { ... }
}
public class DerivedClass : BaseClass
{
    public DerivedClass()
    : base(1, 2, 3)
    {
    }
}
```

Переопределение метода или свойства базового класса

Задача. Вам нужно переопределить поведение базового класса в его потомке.

Решение. Нужно объявить метод или свойство базового класса с модификатором `virtual`, и этот член должен быть доступен из производного класса. У производного класса должен быть модификатор `override`.

```
public class Base
{
    Int32 _x;
    public virtual Int32 MyProperty
```

```
{
    get
    {
        return _x;
    }
}
public virtual void DoSomething()
{
    _x = 13;
}
}
public class Derived : Base
{
    public override Int32 MyProperty
    {
        get
        {
            return _x * 2;
        }
    }
    public override void DoSomething()
    {
        _x = 14;
    }
}
```

Из базового класса можно ссылаться на его экземпляры и на экземпляры любого производного класса. Например, следующий код выведет 28, а не 13:

```
Base d = new Derived();
d.DoSomething();
Console.WriteLine(d.MyProperty().ToString());
```

Кроме того, вы можете вызывать функции базового класса из производного класса. Для этого пользуйтесь ключевым словом `base`.

```
public class Base
{
    public virtual void DoSomething()
    {
        Console.WriteLine("Base.DoSomething");
    }
}
public class Derived
{
    public virtual void DoSomething()
```