

# Содержание

<b>Предисловие</b>	19
<b>Благодарности</b>	20
<b>Об этой книге</b>	21
Кто должен читать эту книгу?	21
Дорожная карта	22
Терминология, оформление и загружаемый код	24
От издательства	24
<b>Об авторе</b>	25
<b>Часть I. Подготовка к путешествию</b>	27
<b>Глава 1. Изменение стиля разработки в C#</b>	28
1.1. Простой тип данных	29
1.1.1. Тип Product в C#	29
1.1.2. Строго типизированные коллекции в C#	31
1.1.3. Автоматически реализуемые свойства в C#	32
1.1.4. Именованные аргументы в C#	32
1.2. Сортировка и фильтрация	34
1.2.1. Сортировка товаров по названию	34
1.2.2. Запрашивание коллекций	37
1.3. Обработка отсутствия данных	39
1.3.1. Представление неизвестной цены	40
1.3.2. Необязательные параметры и стандартные значения	41
1.4. Введение в LINQ	42
1.4.1. Выражения запросов и внутренние запросы	42
1.4.2. Запрашивание файла XML	43
1.4.3. LINQ to SQL	44
1.5. COM и динамическая типизация	45
1.5.1. Упрощение взаимодействия с COM	45
1.5.2. Взаимодействие с динамическим языком	46
1.6. Более простое написание асинхронного кода	47
1.7. Разделение платформы .NET	49
1.7.1. Язык C#	49
1.7.2. Исполняющая среда	50
1.7.3. Библиотеки инфраструктуры	50
1.8. Как сделать код фантастическим	51
1.8.1. Представление полных программ в виде набора фрагментов	51
1.8.2. Учебный код не является производственным	52
1.8.3. Спецификация языка как лучший друг	53
1.9. Резюме	53
<b>Глава 2. Язык C# как основа всех основ</b>	54
2.1. Делегаты	55
2.1.1. Рецепт для простых делегатов	55
2.1.2. Объединение и удаление делегатов	60
2.1.3. Краткое введение в события	61
2.1.4. Резюме по делегатам	62

2.2. Характеристики системы типов	63
2.2.1. Место C# в мире систем типов	63
2.2.2. Когда возможности системы типов C# 1 оказываются недостаточными?	67
2.2.3. Резюме по характеристикам системы типов	70
2.3. Типы значений и ссылочные типы	70
2.3.1. Значения и ссылки в реальном мире	70
2.3.2. Основные положения типов значений и ссылочных типов	71
2.3.3. Развенчание мифов	73
2.3.4. Упаковка и распаковка	75
2.3.5. Резюме по типам значений и ссылочным типам	76
2.4. За рамками C# 1: новые возможности на прочной основе	76
2.4.1. Средства, связанные с делегатами	77
2.4.2. Средства, связанные с системой типов	79
2.4.3. Средства, связанные с типами значений	81
2.5. Резюме	82
<b>Часть II. C# 2: решение проблем, присущих C# 1</b>	<b>83</b>
<b>Глава 3. Параметризованная типизация с использованием обобщений</b>	<b>85</b>
3.1. Необходимость в обобщениях	86
3.2. Простые обобщения для повседневного использования	88
3.2.1. Обучение на примерах: обобщенный словарь	88
3.2.2. Обобщенные типы и параметры типов	90
3.2.3. Обобщенные методы и чтение обобщенных объявлений	93
3.3. Дополнительные сведения	97
3.3.1. Ограничения типов	97
3.3.2. Выведение типов для аргументов типов в обобщенных методах	103
3.3.3. Реализация обобщений	104
3.4. Дополнительные темы, связанные с обобщениями	110
3.4.1. Статические поля и статические конструкторы	111
3.4.2. Обработка обобщений JIT-компилятором	112
3.4.3. Обобщенная итерация	114
3.4.4. Рефлексия и обобщения	117
3.5. Недостатки обобщений в C# и сравнение с другими языками	121
3.5.1. Отсутствие обобщенной вариантности	122
3.5.2. Отсутствие ограничений операций или "числового" ограничения	127
3.5.3. Отсутствие обобщенных свойств, индексаторов и других членов типа	128
3.5.4. Сравнение с шаблонами C++	129
3.5.5. Сравнение с обобщениями Java	130
3.6. Резюме	132
<b>Глава 4. Типы, допускающие значения null</b>	<b>133</b>
4.1. Что делать, когда значение просто отсутствует?	134
4.1.1. Почему переменные типов значений не могут быть установлены в null	134
4.1.2. Шаблоны для представления значений null в C# 1	135
4.2. Типы System.Nullable<T> и System.Nullable	137
4.2.1. Введение в Nullable<T>	137
4.2.2. Упаковка и распаковка типа Nullable<T>	140
4.2.3. Равенство экземпляров типа Nullable<T>	142
4.2.4. Поддержка необобщенного класса Nullable	142

## 8 Содержание

4.3. Синтаксический сахар C# 2 для работы с типами, допускающими null	143
4.3.1. Модификатор ?	144
4.3.2. Присваивание и сравнение с null	145
4.3.3. Преобразования и операции над типами, допускающими null	147
4.3.4. Булевская логика, допускающая значение null	150
4.3.5. Использование операции as с типами, допускающими null	152
4.3.6. Операция объединения с null	153
4.4. Новаторское использование типов, допускающих null	156
4.4.1. Проба выполнения операции без использования выходных параметров	156
4.4.2. Безболезненные сравнения с использованием операции объединения с null	158
4.5. Резюме	161
<b>Глава 5. Оперативно о делегатах</b>	<b>162</b>
5.1. Прощание с неуклюжим синтаксисом для делегатов	163
5.2. Преобразования групп методов	165
5.3. Ковариантность и контравариантность	166
5.3.1. Контравариантность для параметров делегата	167
5.3.2. Ковариантность возвращаемых типов делегатов	168
5.3.3. Небольшой риск несовместимости	169
5.4. Встраивание действий делегатов с помощью анонимных методов	170
5.4.1. Начинаем с простого: действие над одним параметром	171
5.4.2. Возвращение значений из анонимных методов	173
5.4.3. Игнорирование параметров делегата	175
5.5. Захватывание переменных в анонимных методах	177
5.5.1. Определение замыканий и различных типов переменных	177
5.5.2. Исследование поведения захваченных переменных	178
5.5.3. Смысл захваченных переменных	180
5.5.4. Продленное время жизни захваченных переменных	180
5.5.5. Создание экземпляров локальных переменных	182
5.5.6. Смесь разделяемых и отдельных переменных	184
5.5.1. Руководящие принципы и резюме по захваченным переменным	186
5.6. Резюме	187
<b>Глава 6. Простой способ реализации итераторов</b>	<b>188</b>
6.1. C# 1: сложность написанных вручную итераторов	189
6.2. C# 2: простые итераторы с операторами yield	192
6.2.1. Появление итераторных блоков и оператора yield return	192
6.2.2. Визуализация рабочего потока итератора	194
6.2.3. Расширенный поток выполнения итератора	196
6.2.4. Индивидуальные особенности реализации	200
6.3. Реальные примеры использования итераторов	201
6.3.1. Итерация по датам в расписании	201
6.3.2. Итерация по строкам в файле	203
6.3.3. Ленивая фильтрация элементов с использованием итераторного блока и предиката	206
6.4. Написание псевдосинхронного кода с помощью библиотеки Concurrency and Coordination Runtime	208
6.5. Резюме	210

<b>Глава 7. Заключительные штрихи C# 2: финальные возможности</b>	212
7.1. Частичные типы	213
7.1.1. Создание типа с помощью нескольких файлов	214
7.1.2. Использование частичных типов	216
7.1.3. Частичные методы (только C# 3)	218
7.2. Статические классы	220
7.3. Отдельные модификаторы доступа для средств получения/установки свойств	222
7.4. Псевдонимы пространств имен	223
7.4.1. Уточнение псевдонимов пространств имен	224
7.4.2. Псевдоним глобального пространства имен	225
7.4.3. Внешние псевдонимы	226
7.5. Директивы <code>pragma</code>	228
7.5.1. Директивы <code>#pragma warning</code>	228
7.5.2. Директивы <code>#pragma checksum</code>	229
7.6. Буферы фиксированного размера в небезопасном коде	230
7.7. Открытие внутренних членов для избранных сборок	232
7.7.1. Дружественные сборки в простом случае	232
7.7.2. Причины использования атрибута <code>InternalsVisibleTo</code>	233
7.7.3. Атрибут <code>InternalsVisibleTo</code> и подписанные сборки	234
7.8. Резюме	235
<b>Часть III. C# 3: революционные изменения в доступе к данным</b>	237
<b>Глава 8. Отбрасывание мелочей с помощью интеллектуального компилятора</b>	239
8.1. Автоматически реализуемые свойства	240
8.2. Неявная типизация локальных переменных	243
8.2.1. Использование ключевого слова <code>var</code> для объявления локальной переменной	243
8.2.2. Ограничения неявной типизации	245
8.2.3. Доводы за и против неявной типизации	246
8.2.4. Рекомендации	248
8.3. Упрощенная инициализация	249
8.3.1. Определение нескольких демонстрационных типов	249
8.3.2. Установка простых свойств	250
8.3.3. Установка свойств встроенных объектов	251
8.3.4. Инициализаторы коллекций	252
8.3.5. Использование средств инициализации	255
8.4. Неявно типизированные массивы	256
8.5. Анонимные типы	258
8.5.1. Знакомство с анонимными типами	258
8.5.2. Члены анонимного типа	260
8.5.3. Инициализаторы проекций	261
8.5.4. В чем смысл существования анонимных типов?	262
8.6. Резюме	264
<b>Глава 9. Лямбда-выражения и деревья выражений</b>	265
9.1. Лямбда-выражения как делегаты	267
9.1.1. Подготовительные работы: знакомство с типами делегатов <code>Func&lt;...&gt;</code>	267
9.1.2. Первая трансформация в лямбда-выражение	268
9.1.3. Использование одиночного выражения в качестве тела	269
9.1.4. Списки неявно типизированных параметров	269
9.1.5. Сокращение для единственного параметра	270

## 10 Содержание

9.2. Простые примеры использования типа List<T> и событий	272
9.2.1. Фильтрация, сортировка и действия на списках	272
9.2.2. Регистрация внутри обработчика событий	273
9.3. Деревья выражений	275
9.3.1. Построение деревьев выражений программным образом	275
9.3.2. Компиляция деревьев выражений в делегаты	276
9.3.3. Преобразование лямбда-выражений C# в деревья выражений	278
9.3.4. Деревья выражений являются основой LINQ	281
9.3.5. Использование деревьев выражений за рамками LINQ	283
9.4. Изменения в выведении типов и распознавании перегруженных версий	285
9.4.1. Причины внесения изменений: упрощение вызова обобщенных методов	285
9.4.2. Выведение возвращаемых типов анонимных функций	286
9.4.3. Двухэтапное выведение типов	288
9.4.4. Выбор правильного перегруженного метода	292
9.4.5. Итоги по выведению типов и распознаванию перегруженных версий	294
9.5. Резюме	294
<b>Глава 10. Расширяющие методы</b>	<b>295</b>
10.1. Ситуация до появления вспомогательных методов	296
10.2. Синтаксис расширяющих методов	298
10.2.1. Объявление расширяющих методов	299
10.2.2. Вызов расширяющих методов	300
10.2.3. Обнаружение расширяющих методов	301
10.2.4. Вызов метода на ссылке null	303
10.3. Расширяющие методы в .NET 3.5	305
10.3.1. Первые шаги в работе с классом Enumerable	305
10.3.2. Фильтрация с помощью метода Where ( ) и соединение обращений к методам в цепочку	307
10.3.3. Антракт: разве мы не видели метод Where ( ) раньше?	308
10.3.4. Проецирование с использованием метода Select ( ) и анонимных типов	309
10.3.5. Сортировка с использованием метода OrderBy ( )	310
10.3.6. Бизнес-примеры, предусматривающие соединение вызовов в цепочки	312
10.4. Идеи и руководство по использованию	313
10.4.1. “Расширение мира” и совершенствование интерфейсов	313
10.4.2. Текущие интерфейсы	314
10.4.3. Разумное использование расширяющих методов	316
10.5. Резюме	317
<b>Глава 11. Выражения запросов и LINQ to Objects</b>	<b>319</b>
11.1. Введение в LINQ	320
11.1.1. Фундаментальные концепции LINQ	320
11.1.2. Определение эталонной модели данных	325
11.2. Простое начало: выборка элементов	326
11.2.1. Превращение начального источника в выборку	327
11.2.2. Трансляция компилятором как основа выражений запросов	327
11.2.3. Переменные диапазонов и нетривиальные проекции	330
11.2.4. Cast ( ), OfType ( ) и явно типизированные переменные диапазонов	332
11.3. Фильтрация и упорядочение последовательности	335
11.3.1. Фильтрация с использованием конструкции where	335
11.3.2. Вырожденные выражения запросов	336
11.3.3. Упорядочение с использованием конструкции orderby	337

11.4. Конструкции <code>let</code> и прозрачные идентификаторы	339
11.4.1. Добавление промежуточных вычислений с помощью конструкции <code>let</code>	339
11.4.2. Прозрачные идентификаторы	340
11.5. Соединения	342
11.5.1. Внутренние соединения с использованием конструкций <code>join</code>	342
11.5.2. Групповые соединения с использованием конструкций <code>join...into</code>	346
11.5.3. Перекрестные соединения и выравнивание последовательностей с использованием нескольких конструкций <code>from</code>	349
11.6. Группирование и продолжение	353
11.6.1. Группирование с помощью конструкции <code>group...by</code>	353
11.6.2. Продолжение запроса	356
11.7. Выбор между выражениями запросов и точечной нотацией	359
11.7.1. Операции, которые требуют точечной нотации	359
11.7.2. Использование выражений запросов в ситуациях, когда точечная нотация может быть проще	360
11.7.3. Ситуации, когда выражения запросов блестящи	361
11.8. Резюме	362
<b>Глава 12. LINQ за рамками коллекций</b>	<b>363</b>
12.1. Запрашивание базы данных с помощью LINQ to SQL	364
12.1.1. Начало работы: база данных и модель	365
12.1.2. Начальные запросы	367
12.1.3. Запросы, в которых задействованы соединения	370
12.2. Трансляция с использованием <code>IQueryable</code> и <code>IQueryProvider</code>	372
12.2.1. Введение в <code>IQueryable&lt;T&gt;</code> и связанные интерфейсы	372
12.2.2. Имитация: реализация интерфейсов для регистрации вызовов	374
12.2.3. Интеграция выражений: расширяющие методы из класса <code>Queryable</code>	376
12.2.4. Имитированный поставщик запросов в действии	378
12.2.5. Итоги по интерфейсу <code>IQueryable</code>	379
12.3. API-интерфейсы, дружественные к LINQ, и LINQ to XML	380
12.3.1. Основные типы в LINQ to XML	380
12.3.2. Декларативное конструирование	382
12.3.3. Запросы для одиночных узлов	385
12.3.4. Выравнивающие операции запросов	386
12.3.5. Работа в гармонии с LINQ	388
12.4. Замена LINQ to Objects технологией Parallel LINQ	388
12.4.1. Отображение множества Мандельброта с помощью одного потока	389
12.4.2. Введение в <code>ParallelEnumerable</code> , <code>ParallelQuery</code> и <code>AsParallel()</code>	390
12.4.3. Подстройка параллельных запросов	392
12.5. Инвертирование модели запросов с помощью LINQ to Rx	394
12.5.1. <code>IObservable&lt;T&gt;</code> и <code>IObserver&lt;T&gt;</code>	394
12.5.2. Простое начало (снова)	396
12.5.3. Запрашивание наблюдаемых объектов	397
12.5.4. Какой в этом смысл?	400
12.6. Расширение LINQ to Objects	400
12.6.1. Руководство по проектированию и реализации	401
12.6.2. Пример расширения: выборка случайного элемента	402
12.7. Резюме	404

<b>Часть IV. C# 4: изящная игра с другими</b>	407
<b>Глава 13. Небольшие изменения, направленные на упрощение кода</b>	408
13.1. Необязательные параметры и именованные аргументы	409
13.1.1. Необязательные параметры	409
13.1.2. Именованные аргументы	416
13.1.3. Объединение двух средств	420
13.2. Модернизация взаимодействия с COM	424
13.2.1. Ужасы автоматизации Word до выхода C# 4	425
13.2.2. Реванш необязательных параметров и именованных аргументов	426
13.2.3. Ситуации, когда параметр <code>ref</code> в действительности таковым не является	427
13.2.4. Вызов именованных индексаторов	428
13.2.5. Связывание основных сборок взаимодействия	429
13.3. Обобщенная вариантность для интерфейсов и делегатов	432
13.3.1. Типы вариантности: ковариантность и контравариантность	432
13.3.2. Использование вариантности в интерфейсах	434
13.3.3. Использование вариантности в делегатах	437
13.3.4. Сложные ситуации	438
13.3.5. Ограничения и замечания	440
13.4. Мелкие изменения в блокировке и событиях, подобных полям	443
13.4.1. Надежная блокировка	443
13.4.2. Изменения в событиях, подобных полям	445
13.5. Резюме	446
<b>Глава 14. Динамическое связывание в статическом языке</b>	447
14.1. Что? Когда? Почему? Как?	449
14.1.1. Что такое динамическая типизация?	449
14.1.2. Когда динамическая типизация удобна и почему?	450
14.1.3. Как в C# 4 поддерживается динамическая типизация?	451
14.2. Пятиминутное руководство по <code>dynamic</code>	452
14.3. Примеры применения динамической типизации	455
14.3.1. COM в общем и Microsoft Office в частности	455
14.3.2. Динамические языки, подобные IronPython	457
14.3.3. Динамическая типизация в полностью управляемом коде	462
14.4. Заглядывая за кулисы	468
14.4.1. Введение в DLR	468
14.4.2. Основные концепции DLR	470
14.4.3. Как компилятор C# обрабатывает динамическое поведение	474
14.4.4. Компилятор C# становится еще интеллектуальнее	478
14.4.5. Ограничения, накладываемые на динамический код	481
14.5. Реализация динамического поведения	484
14.5.1. Использование <code>ExpandableObject</code>	484
14.5.2. Использование <code>DynamicObject</code>	488
14.5.3. Реализация интерфейса <code>IDynamicMetaObjectProvider</code>	495
14.6. Резюме	499
<b>Часть V. C# 5: упрощение асинхронности</b>	501
<b>Глава 15. Асинхронность с помощью <code>async/await</code></b>	502
15.1. Введение в асинхронные функции	504
15.1.1. Первые встречи с асинхронностью	504
15.1.2. Разбор первого примера	506

15.2. Обдумывание асинхронности	507
15.2.1. Фундаментальные основы асинхронного выполнения	508
15.2.2. Моделирование асинхронных методов	510
15.3. Синтаксис и семантика	511
15.3.1. Объявление асинхронного метода	511
15.3.2. Возвращаемые типы асинхронных методов	512
15.3.3. Шаблон ожидания	513
15.3.4. Поток выражений <code>await</code>	516
15.3.5. Возвращение значений из асинхронных методов	521
15.3.6. Исключения	521
15.4. Асинхронные анонимные функции	530
15.5. Детали реализации: трансформация компилятора	532
15.5.1. Обзор сгенерированного кода	533
15.5.2. Структура каркасного метода	536
15.5.3. Структура конечного автомата	537
15.5.4. Одна точка входа для управления всем	539
15.5.5. Поток управления для выражений <code>await</code>	540
15.5.6. Отслеживание стека	542
15.5.7. Дополнительные сведения	543
15.6. Практическое использование <code>async/await</code>	544
15.6.1. Асинхронный шаблон, основанный на задачах	544
15.6.2. Объединение асинхронных операций	548
15.6.3. Модульное тестирование асинхронного кода	552
15.6.4. Возвращение к шаблону ожидания	555
15.6.5. Асинхронные операции в WinRT	556
15.7. Резюме	557
<b>Глава 16. Дополнительные средства C# 5 и заключительные размышления</b>	559
16.1. Изменения в захваченных переменных внутри циклов <code>foreach</code>	560
16.2. Атрибуты информации о вызывающем компоненте	560
16.2.1. Базовое поведение	561
16.2.2. Регистрация в журнале	563
16.2.3. Реализация интерфейса <code>INotifyPropertyChanged</code>	563
16.2.4. Использование атрибутов информации о вызывающем компоненте без .NET 4.5	565
16.3. Заключительные размышления	565
<b>Приложение А. Стандартные операции запросов LINQ</b>	567
А.1. Агрегирование	567
А.2. Конкатенация	568
А.3. Преобразование	569
А.4. Операции элементов	571
А.5. Эквивалентность	572
А.6. Генерация	573
А.7. Группирование	573
А.8. Соединения	574
А.9. Разделение	575
А.10. Проецирование	576
А.11. Квантификаторы	577
А.12. Фильтрация	578
А.13. Операции, основанные на множествах	578
А.14. Сортировка	579



<b>Приложение Б. Обобщенные коллекции в .NET</b>	581
Б.1. Интерфейсы	581
Б.2. Списки	583
Б.2.1. List<T>	583
Б.2.2. Массивы	584
Б.2.3. LinkedList<T>	585
Б.2.4. Collection<T>, BindingList<T>, ObservableCollection<T> и KeyedCollection<TKey, TItem>	586
Б.2.5. ReadOnlyCollection<T> и ReadOnlyObservableCollection<T>	587
Б.3. Словари	587
Б.3.1. Dictionary<TKey, TValue>	587
Б.3.2. SortedList<TKey, TValue> и SortedDictionary<TKey, TValue>	588
Б.3.3. ReadOnlyDictionary<TKey, TValue>	589
Б.4. Множества	589
Б.4.1. HashSet<T>	590
Б.4.2. SortedSet<T> (.NET 4)	590
Б.5. Queue<T> и Stack<T>	591
Б.5.1. Queue<T>	591
Б.5.2. Stack<T>	591
Б.6. Параллельные коллекции (.NET 4)	592
Б.6.1. IProducerConsumerCollection<T> и BlockingCollection<T>	592
Б.6.2. ConcurrentBag<T>, ConcurrentQueue<T> и ConcurrentStack<T>	593
Б.6.3. ConcurrentDictionary<TKey, TValue>	593
Б.7. Интерфейсы, допускающие только чтение (.NET 4.5)	593
Б.8. Резюме	595
<b>Приложение В. Итоговые сведения по версиям</b>	596
В.1. Главные выпуски инфраструктуры для настольных приложений	596
В.2. Средства языка C#	597
В.2.1. C# 2.0	597
В.2.2. C# 3.0	598
В.2.3. C# 4.0	598
В.2.4. C# 5.0	598
В.3. Средства библиотек инфраструктуры	598
В.3.1. .NET 2.0	598
В.3.2. .NET 3.0	599
В.3.3. .NET 3.5	599
В.3.4. .NET 4	600
В.3.5. .NET 4.5	601
В.4. Средства исполняющей среды (CLR)	601
В.4.1. CLR 2.0	601
В.4.2. CLR 4.0	601
В.5. Связанные инфраструктуры	602
В.5.1. Compact Framework	602
В.5.2. Silverlight	603
В.5.3. Micro Framework	603
В.5.4. Windows Runtime (WinRT)	604
В.6. Резюме	604
<b>Предметный указатель</b>	605