TURING 图灵程序设计丛书 微软技术系列

Addison Wesley

Essential C# 4.0

C#本质论

第3版

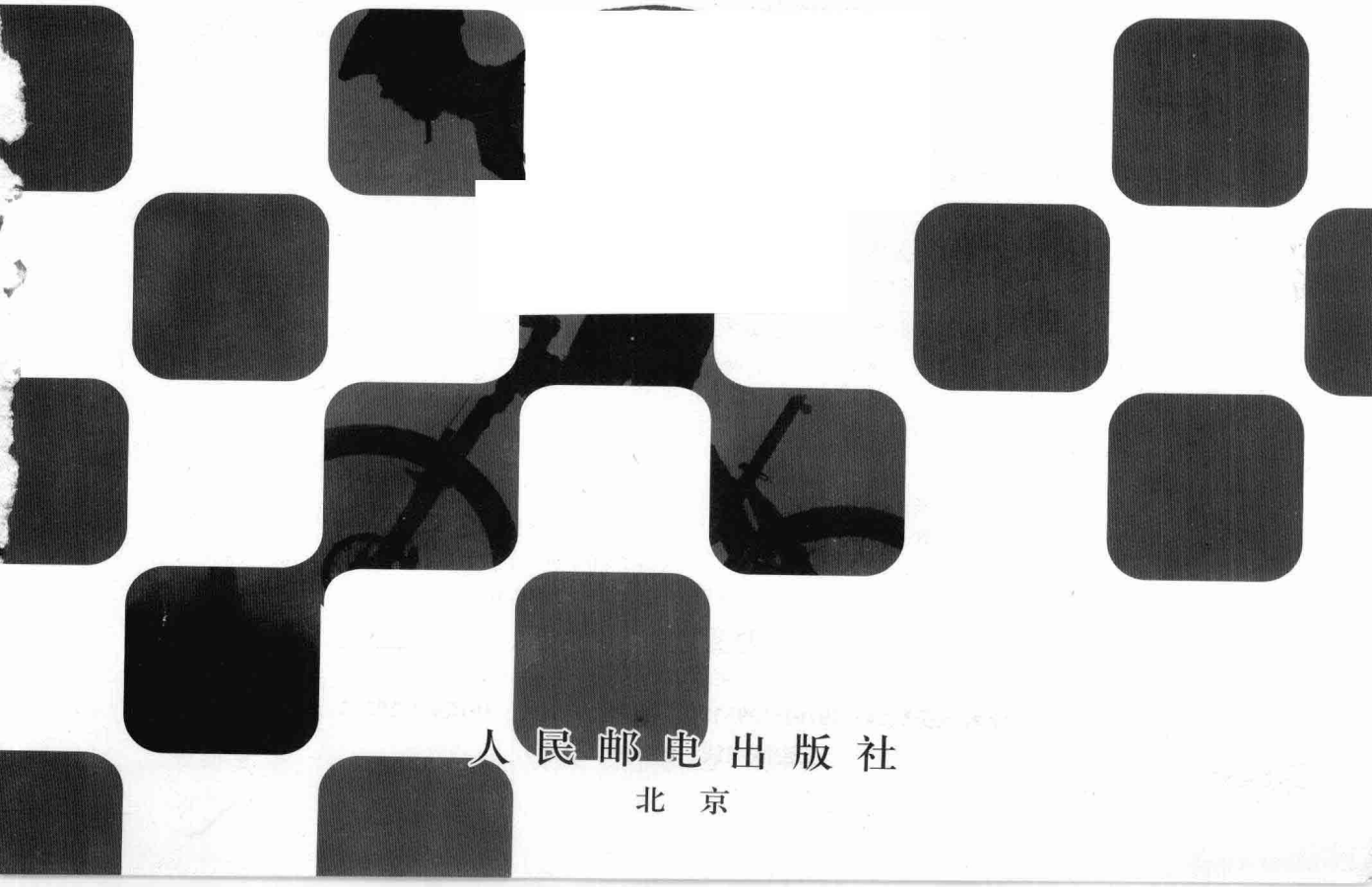
[美] Mark Michaelis 著
周靖 译

人民邮电出版社
POSTS & TELECOM PRESS

TURING 图灵程序设计丛书 微软技术系列

Essential C# 4.0
C#本质论
第3版

[美] Mark Michaelis 著
周靖 译



人民邮电出版社
北京

图书在版编目（CIP）数据

C#本质论 : 第3版 / （美）米凯利斯(Michaelis,M.）著 ; 周靖译. -- 北京 : 人民邮电出版社, 2010.9
（图灵程序设计丛书）
书名原文: Essential C# 4.0
ISBN 978-7-115-23383-7

Ⅰ. ①C… Ⅱ. ①米… ②周… Ⅲ. ①C语言－程序设计 Ⅳ. ①TP312

中国版本图书馆CIP数据核字(2010)第143086号

内 容 提 要

本书是一部好评如潮的语言参考书，作者用一种非常合理的方式来组织本书的内容，由浅入深地介绍了C# 语言的各个方面。全书共包括 21 章及 6 个附录，每章开头的“思维导图”指明了本章要讨论的主题，以及各个主题之间的层次关系。书中所包含的丰富的示例代码和精要的语言比较，都有助于读者理解 C# 语言。本书首先介绍了 C# 语言的基础知识，随后深入讲解了泛型、迭代器、反射、线程、互操作性和语言集成查询（LINQ）等高级主题，还涉及了动态编程、使用 TPL 进行多线程编程以及用 PLINQ 进行并行查询处理等 C# 4.0 新增内容，此外还介绍了与这些内容相关的隐式类型变量、扩展方法、分部方法、Lambda 语句和表达式、标准查询操作符和查询表达式以及并发集合等内容。

本书适合对 C# 感兴趣的各层次读者，无论对初学者还是 C# 专家，本书都是一本很有价值的参考书。

图灵程序设计丛书

C#本质论（第3版）

◆ 著 [美] Mark Michaelis
译 周 靖
责任编辑 傅志红
执行编辑 李 静

◆ 人民邮电出版社出版发行 北京市崇文区夕照寺街14号
邮编 100061 电子函件 315@ptpress.com.cn
网址 http://www.ptpress.com.cn
北京艺辉印刷有限公司印刷

◆ 开本：800×1000 1/16
印张：45.25
字数：1125千字 2010年9月第1版
印数：1－3 500册 2010年9月北京第1次印刷

著作权合同登记号 图字：01-2010-4021号

ISBN 978-7-115-23383-7

定价：99.00元

读者服务热线：(010)51095186 印装质量热线：(010)67129223
反盗版热线：(010)67171154

版 权 声 明

献给我的家人：Elisabeth、Benjamin、Hanna 和 Abigail。

感谢你们容忍我花费漫长的时间来写作，

而且经常是在你们最需要我的时候。谢谢你们！

序

Mark Michaelis 的这本 C#语言著作已成为开发人员的一本标准参考书。针对世界上最流行的计算机语言之一，本书第 3 版提供了一份包罗万象的、精彩纷呈的指南。在本书以前版本建立的良好基础上，Mark 添加了新的章节来讲解 C#和.NET Framework 的最新功能。

本书最重要的两项增补内容是用于并行编程的最新工具，以及 C# 4.0 新增的动态功能。利用新增的动态功能，C#语言允许开发者访问 Python 和 Ruby 这样的后期绑定语言。为 COM Interop 提供的增强支持允许开发人员以一种直观且易于使用的语法访问 Microsoft Office，该语法使得这种工具也易于使用。Mark 对这些重要主题的精彩讲述，并连同他对并行开发最新进展的权威解释，极大地增加了本书的“含金量”。如果 C#开发人员想要磨砺他们的技能，并掌握 C#语言最好和最关键的部分，本书就是必读的。

作为 C#团队的社区项目经理（Community Program Manager），我的职责就是随时跟进社区的需求。我经常都能听到这样的抱怨：“微软的新技术太多了，简直让人应接不暇。我需要阅读详细解释这种技术的文档，而且我希望这些文档很容易看懂。”对于每一名希望了解微软最新技术的 C#开发者，Mark Michaelis 都是他们最为可靠的“单人解决方案”。

第一次与 Mark 见面是在华盛顿州雷德蒙德市，那是 2006 年夏天的一个清新、阳光明媚的早上。我喜欢晚睡，但我已被提前告知，Mark 是一位非常积极的社区成员。所以，那天我不得不起得很早去见他。幸好我早起去见他了！时过多年，那天早上他给我留下的深刻印象至今依然记忆犹新。

Mark 是一个高个子的运动员，来自南非，说一口清楚、流利的英语，有一点口音，不过相信大多数美国人都分辨不出其中的区别。他是铁人三项的运动员，有着从事那种运动的人特有的精干、活跃的外表。虽然表面上乐呵呵的，但他做起事情来却显得非常老练。给人的感觉就是他有一种特殊的本事，能在有限的时间里安排尽可能多的活动。

Mark 经常要到微软总部来参加有关未来技术的一些研讨会，或者就团队的一些未来计划提供咨询。从华盛顿州斯波堪市飞到这里，Mark 有一个清楚的日程表。他知道自己在 Microsoft 总部的职责，他在那里会将全部身心都放到工作上，而在工作结束后又会风尘仆仆地赶回斯波堪与家人团聚。有时，他能抽出一些时间来和我匆匆见上一面，而每次会面都让我感到高兴。他是一个乐观、精力充沛的家伙，对微软开发的新技术或程序总能提出一些令人振奋的见解。

通过以上对 Mark 的简单描述，相信你对本书的内容已经心中有数了。这是一本主题鲜明的

书，结构提纲非常合理，而且通篇文字都简洁流畅，没有任何废话。Mark 着意挑选了需要详尽解释的核心语言部分，并用他平时说话的方式讲出来——表达清楚、有血有肉，而且丝毫没有居高临下的口吻。Mark 知道读者想要听什么，而他也做到了知无不言、言无不尽。

Mark 不光懂 C#，还懂英语。他知道如何正确地造句，如何把他的思想灌注到看似普通的段落和小节中，以及如何引入和总结主题。他善于用清楚且易于理解的方式来解释一个复杂的主题。Mark 这本书的第一版我是逐页看过的，当时全神贯注地阅读，只花了我几个晚上的时间。和最新版本一样，整个阅读体验是令人愉快和满足的。Mark 精心选择了要讲述的主题，并用尽可能简单的话来解释它们。他知道哪些应该包括进来，哪些可以暂时放在一边。如果他想探讨一个高级主题，就会非常贴心地把它同书的其余部分隔开。他从不炫耀，一直都以读者的感受为重。

在本书的这一版中，一个相当重要的主题就是 LINQ。对于许多开发者来说，LINQ 所采用的说明性编程风格属于一种全新的技术，需要为之发展出一套新的编程习惯和新的思维方式。

C# 3.0 引入了几项支持 LINQ 的新特性。本书这一版的主要目标之一就是详细讨论这些特性。对 LINQ 及其关联技术进行解释绝不是一项轻松的任务，Mark 发挥了其作家兼教师职业的特长，使这些主题尽可能地清楚和容易理解。

为了理解 LINQ 而需掌握的所有核心技术都在本书得到了精心的解释，其中包括：

- 分部方法
- 自动属性
- 对象初始化器
- 集合初始化器
- 匿名类型
- 隐式局部变量（var）
- Lambda
- 扩展方法
- 表达式树
- IEnumerable<T>和 IQueryable<T>
- LINQ 查询运算符
- 查询表达式

在本书中，LINQ 之旅的起点是 Mark 对一些重要的 C# 2.0 技术的讲解，比如泛型和委托。然后，他会手把手地教你如何从委托迁移到 Lambda。他解释了为什么 Lambda 会成为 C# 3.0 的一部分，及其在 LINQ 中扮演的重要角色。他还讲解了扩展方法，以及在实现 LINQ 查询运算符时，这些扩展方法所扮演的角色。

在详细解释查询表达式时，他对 C# 3.0 新特性的关注达到了顶峰。在此期间，他讲述了查询表达式的关键特性，比如投射、筛选、排序、分组和为了理解 LINQ 而需要掌握的重要概念。然后，他解释了如何将查询表达式转换成实际由编译器来执行的 LINQ 查询方法语法。等你阅读完有关查询表达式的内容之后，应该就已经掌握了理解 LINQ 需要的全部知识，并可以在自己的程序中使用这一重要技术了。

如果你想成为一名C#开发者，或者想提高现有的C#编程技能，一本切中主题、精心组织的书是最为有用的。你现在就拿着这样的一本书。像这样的一本书，首先会让你掌握语言是如何工作的，然后在你需要快速找到答案的时候，它又可以作为一本参考书供你使用。对于那些想要了解微软最新技术的开发者，这本书也会成为你的良师益友，帮助你理清不断变化的技术趋势。究竟哪种语言正在快速演变成当今最高级、最重要的编程语言？本书为你展示了最准确、最新颖的观点。

Charlie Calvert

微软 Visual C#社区项目经理

2010年1月

前　言

在软件工程的发展历史中，用于编写计算机程序的方法经历了几次思维模式的重大转变。每一种思维模式都是以前一种为基础的，其宗旨都是增强代码的组织，并降低复杂性。本书将带领你体验相同的思维模式转变过程。

本书开始的几章指导你学习**顺序编程结构**（sequential programming structure）。在这种编程结构中，语句是按照执行顺序来写的。这种结构的问题在于，随着需求的增加，复杂性也将呈指数级增加。为了降低复杂性，将代码块转变成方法，产生了**结构化编程模型**（structured programming model）。在这种模型中，可以从一个程序中的多个位置调用同一个代码块，而不必在程序中重复这些代码。然而，即使有这种结构，程序还是会很快变得臃肿不堪，需要进行进一步抽象。所以，在此基础上，人们又提出了面向对象编程的概念，这将在第 5 章进行讨论。在此之后，你将继续学习其他编程方法，比如基于接口的编程和 LINQ（以及它促使集合 API 发生的改变），并最终学习通过特性进行初级的声明性编程[①]（第 17 章）。

本书有以下 3 个主要职能。

- 全面讲述 C#语言，其内容已经远远超过了一本简单的教程，为你进行高效率软件开发打下坚实的基础。
- 对于已经熟悉了 C#的读者，本书探讨了一些较为复杂的编程思想，并深入讨论了在这种语言的最新版本（C# 4.0 和.NET 4）中引入的特性。
- 它是你永远的案头参考——即便在你精通了这种语言之后。

成功学习 C#的关键在于，要尽可能快地开始编程。不要等自己成为一名理论方面的“专家”之后，才开始写代码。所以，不要犹豫，马上开始写程序吧。作为迭代开发[②]思想的追随者，我希望即使是一名刚开始学习编程的新手，在学到本书第 2 章末尾的时候，也能动手开始写基本的 C#代码。

有许多主题都没有在本书中进行讨论。你在本书找不到 ASP.NET、ADO.NET、智能客户端开发以及分布式编程等主题。虽然这些主题与.NET Framework 有关，但它们都值得用专门的书分专题进行讲述。幸运的是，已经有丰富的图书供读者选择。本书的重点在于 C#以及

① 与声明性编程（declarative programming）对应的是命令式编程；前者用于表述问题，后者用于实际解决问题。——译者注

② 简单地说，迭代开发是指分周期、分阶段进行一个项目，以增量方式逐渐对其进行改进的过程。——译者注

基类库中的类型。在读完本书之后，你在上述任何领域继续深入学习都会有游刃有余的感觉。

本书读者对象

写作本书时，我面临的一个挑战是如何在持续吸引高级开发人员眼球的同时，不因使用类似assembly、link、chain、thread和fusion的字眼而打击初学者的信心，否则许多人会认为这是一本冶金方面的书，而不是讲程序设计的[①]。本书的主要读者是已经有一定编程经验，并想多学一种语言来“傍身”的开发者。但我还是小心地编排了本书的内容，使其对于各种层次的开发者来说，都有足够大的价值。

- *初学者*：假如你是一名编程新手，本书将帮助你从入门级的程序员过渡成为一名C#开发者，消除对以后摆在你面前的任何C#编程任务的害怕心理。本书不仅要教会你语法，还要教你养成良好的编程习惯，为你将来的编程生涯打下良好的基础。
- *熟悉结构化编程的程序员*：学习外语最好的方法就是“沉浸法”[②]。类似地，学习一门计算机语言最好的方法就是在动手中学习，而不是等熟知了它的所有“理论”之后再动手。基于这个前提，本书最开始的内容是那些熟悉结构化编程的开发者很容易上手的。到第4章结束时，这些开发者应该可以开始写基本的控制流程序。然而，要成为一名真正的C#开发者，记住语法只是第一步。为了从简单程序过渡到企业级开发，C#开发者必须熟练地从对象及其关系的角度来思考问题。为此，第5章的“初学者主题”开始介绍类和面向对象开发。对于C、COBOL和FORTRAN等结构化编程语言来说，虽然它们仍在发挥作用，但作用会越来越小。所以，软件工程师们应该逐渐开始了解面向对象开发。C#是进行这一思维模式转变的理想语言，因为它本来就是基于“面向对象开发”这一中心思想来设计的。
- *熟悉“基于对象”和“面向对象”理念的开发者*：C++和Java程序员以及许多有经验的Visual Basic程序员都可归于此类。对于分号和大括号，他们可是一点儿都不陌生！简单浏览一下第1章的代码，你会发现，从核心上讲，C#类似于你熟知的C和C++风格的语言。
- C#*专家*：对于已经精通C#的人，本书可供你参考不太常见的语法。此外，对于在其他地方强调较少的一些语言细节以及微妙之处，我提出了自己的见解。最重要的是，本书提供了编写可靠和易维护代码的指导原则及模式。你教别人学C#时，本书也颇有助益。C# 3.0和C# 4.0的一些最重要的增强包括：
  - 隐式类型的变量（参见第2章）；
  - 扩展方法（参见第5章）；
  - 分部方法（参见第5章）；
  - 匿名类型（参见第11章）；

① 上述每个单词在计算机和冶金领域都有专门的含义，所以作者用它们开了一个玩笑。例如，assembly既是“程序集”，也是“装配件”；thread既是“线程”，也是“螺纹”。——译者注

② 沉浸法，即immersion approach，是指想办法让学习者泡到一个全外语的环境中，比如孤身一人在国外生活或学习。——译者注

- 泛型（参见第11章）；
- Lambda语句和表达式（参见第12章）；
- 表达式树（参见第12章）；
- 标准查询操作符（参见第14章）；
- 查询表达式（参见第15章）；
- 动态编程（参见第17章）；
- 用任务编程库进行多线程编程（参见第18章）；
- 用PLINQ进行并行查询处理；
- 并发集合（第19章）。

考虑到许多人还不熟悉这些主题，本书围绕它们展开了详细的讨论。涉及高级 C#开发的还有“指针”这一主题，该主题将在第 21 章讨论。就算是有经验的 C#开发者，也未必能很透彻地理解这一主题。

本书特色

本书是一本语言参考书，它遵循核心 C# Language 4.0 Specification。为了帮助读者理解各种C#构造，书中用大量例子演示了每一种特性，而且为每个概念都提供了相应的指导原则和最佳实践，以确保代码能顺利编译、避免留下隐患，并获得最佳的可维护性。

为了增强可读性，所有代码均进行了特殊格式处理，而且每一章的内容都使用思维导图来概括。

代码示例

本书大多数代码片段都能在公共语言基础结构（Common Language Infrastructure，CLI）的任何实现上运行，其中包括 Mono、Rotor 和微软.NET 平台。我很少使用平台或厂商特有的库，除非需要解释只和那些平台有关的某些重要概念（例如，解释如何正确处理 Windows 单线程用户界面）。任何需要 C# 3.0 或 C# 4.0 相容性的代码都在本书末尾的 C# 3.0 和 C# 4.0 索引中进行了标注。

下面是一个示例代码清单。

代码清单 1-17　在代码中添加注释

```
class CommentSamples
{
  static void Main()
  {
                                        单行注释
      string firstName; // Variable for storing the first name
      string lastName;  // Variable for storing the last name

      System.Console.WriteLine("Hey you!");
```

```
                                    语句内部带分隔符的注释
        System.Console.Write /* No new line */ (
            "Enter your first name: ");
        firstName = System.Console.ReadLine();

        System.Console.Write /* No new line */ (
            "Enter your last name: ");
        lastName = System.Console.ReadLine();

        /* Display a greeting to the console
           using composite formatting. */          } 带分隔符的注释
        System.Console.WriteLine("Your full name is {0} {1}.",
            firstName, lastName);
        // This is the end
        // of the program listing
    }
}
```

下面简单介绍一下代码格式。

- 注释以斜体表示。

```
/* Display a greeting to the console
   using composite formatting. */
```

- 关键字加粗。

```
static void Main()
```

- 有的代码被突出显示，是为了指明这些代码与之前列出的有所区别，或是为了演示正文中介绍的概念。

```
System.Console.Write /* No new line */ (
```

突出显示的内容可能是一整行，也可能仅仅是一行中的几个字符。

```
System.Console.WriteLine(
    "Your full name is {0} {1}.",
```

- 不完整的程序清单包含一个省略号，指出无关的代码已省略。

```
// ...
```

- 在代码清单之后，列出了对应的控制台输出，如下例所示：

输出 1-4

```
>HeyYou.exe
Hey you!
Enter your first name: Inigo
Enter your last name: Montoya
```

用户在执行程序时输入的内容显示为斜体。

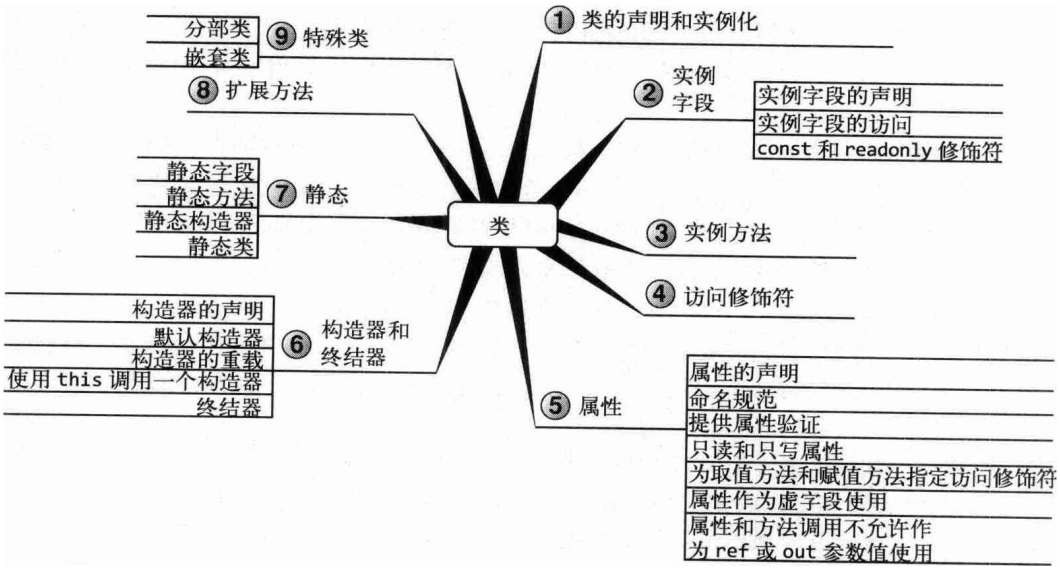
虽然提供完整的代码示例可以方便你在自己的程序中使用它们，但这样做会分散你对特

定主题的注意力。因此，你需要自行修改代码示例，然后才能把它们集成到自己的程序中。此外，书中的代码示例没有显式地包含 `using System` 语句，在所有的例子中，这个语句都是必需的。

可以从 intelliTechture.com/EssentialCSharp 和 informit.com/msdotnetseries 上下载示例代码和素材。

思维导图

每一章开头都包含一幅“**思维导图**[①]”。作为一个提纲，它的作用是为读者提供对每章内容的一个快速参考。下面是一个例子（摘自第 5 章）。



每一章的主题显示在思维导图的中心，高级主题围绕这个中心展开。利用思维导图，读者可以方便地搭建自己的知识体系，可以从一个主题出发，更清楚地理解其周边的各个具体概念，避免中途纠缠于一些不相干的枝节问题。

分类解说

根据自己的编程水平，书中特殊的代码块和页面边缘的灰色竖线条可以帮你轻松地找到适合自己的内容。

- *初学者主题*：特别针对入门级程序员提供的定义或解释。

① 思维导图，即 mind map，又称脑图、心智图，其作用是帮助你学习、组织和存储你想要的所有信息，它以自然的方式对信息进行分类，使你能够立即得到你想要的一切。你可以将其想象成一幅帮助自己记忆和思考的思维路线图。——译者注

- 高级主题：可以让有经验的开发者将注意力放在他们最关心的内容上。
- 标注：用标注框[①]来强调关键原则，使读者对其重要性一目了然。
- 语言对比：分散在正文中的补充内容，描述了C#和其他语言的关键差异，为熟悉其他语言的读者提供指引。

本书内容组织

总的来说，软件工程的宗旨就是对复杂性进行管理。本书正是基于这个宗旨来组织内容的。第 1 章～第 4 章介绍的是结构化编程，学习了这些内容后，你可以立即开始写一些功能简单的代码。第 5 章～第 9 章介绍了 C#的面向对象构造，新手应在完全理解了这几章的内容之后，再开始接触本书其余部分的更高级主题。第 11 章～第 13 章介绍了更多用于降低复杂性的构造，讲解了当今几乎所有程序都要用到的通用设计模式。在理解了它们之后，你可以更加轻松地理解如何通过反射和特性来实现动态编程。在后续的章节中，将广泛运用反射和特性来实现线程处理和互操作性。

本书最后专门用一章（第 21 章）来讲述 CLI。这一章在开发平台的背景下对 C#语言进行了描述。之所以要放到最后，是因为它并不是 C#特有的，而且不会涉及语法和编程风格问题。不过，本章适合在任何时候阅读，或许最恰当的时机是在阅读了本书的第 1 章之后。

下面是每一章的内容提要。（**加黑**的章标题表明那一章含有 C# 3.0 或 C# 4.0 的内容。）

- 第1章——C#概述：这一章在展示了一个用C#编写的HelloWorld程序之后，进一步剖析了C#。这应当能使读者熟悉C#程序的“外观和感觉”。这一章提供了让读者编译和调试自己的程序所需的足够多的细节。此外，这一章还简单描述了执行C#程序的上下文及其中间语言（intermediate language，IL）。
- 第2章——数据类型：任何有用的程序都要处理数据，这一章介绍了C#的两种基本数据类型，即值类型和引用类型，另外还讨论了类型间的转换以及C#对数组的支持。
- 第3章——运算符和控制流：计算机擅长重复性操作，为了利用计算机的这个能力，需要知道如何在自己的程序中添加循环和条件逻辑。这一章还讨论了C#运算符、数据转换以及预处理器指令。
- 第4章——方法和参数：这一章研究了有关方法及其参数的细节，其中包括通过参数来传值、传引用和返回数据。C# 4.0添加了对默认参数的支持，本章解释了如何使用它们。
- 第5章——类：前面已经学过了类的基本构成元素，这一章将这些构造合并到一起，从而获得具有完整功能的类型。类是面向对象技术的核心，它为一“类”对象定义了一个模板。
- 第6章——继承：虽然继承是许多开发者的基本编程手段，但C#提供了一些独一无二的构造，比如new修饰符。这一章讨论了继承语法的细节，其中包括重写（overriding）。
- 第7章——接口：这一章展示了如何利用接口来定义类之间的“可以进行版本控制的交互

① 标注框，即 callout box，是一种需要准确排版的书稿元素。比如：Call out box。——译者注

契约”（versionable interaction contract）。C#同时包含显式和隐式的接口成员实现，可以实现一个额外的封装等级，这是其他大多数语言所不支持的。

- 第8章——值类型：尽管不如引用类型那么流行，但有些情况下仍然有必要定义行为类似于C#内置基本类型的值类型。这一章要介绍如何定义结构（structure），同时揭示它们的特性。
- 第9章——合式类型：这一章讨论了一些更高级的类型定义，解释了如何实现运算符，比如“+”和转型运算符，并描述了如何将多个类封装到一个库中。除此之外，这一章还演示了如何定义命名空间和XML注释，并讨论了如何基于垃圾回收机制来设计令人满意的类。
- 第10章——异常处理：这一章是对第4章引入的异常处理机制的一个延伸讨论，描述了如何利用异常层次结构来创建自定义异常。此外，它还强调了异常处理的一些最佳实践。
- 第11章——泛型：从某种意义上说，泛型或许是C# 1.0缺少的一个最重要的特性。这一章全面讨论了自2.0引入的这个特性。除此之外，C# 4.0增加了对协变和逆变的支持。本章将在泛型的背景中探讨它们。
- 第12章——委托和Lambda表达式：正是因为委托，才使C#与其前身语言（C和C++等）有了显著的不同，它定义了在代码中处理事件的模式。这几乎完全消除了写轮询例程的必要。Lambda表达式是使C# 3.0的LINQ成为可能的关键概念。通过这一章的学习，你将知道Lambda表达式是在委托的基础上构建起来的，它提供了比委托更加优雅和简洁的语法。本章的内容是第14章讨论的新的集合API的基础。
- 第13章——事件：封装起来的委托（称为事件）是公共语言运行时（Common Language Runtime，CLR）的一个核心构造。本章还探讨了匿名方法，它是在C# 2.0中增加的一个特性。
- 第14章——支持标准查询运算符的集合接口：我们通过讨论新的`Enumerable`类的扩展方法，向你介绍C# 3.0引入的一些简单但又非常强大的改变。`Enumerable`类使一个全新的集合API成为可能，这个API称为“标准查询运算符”，本章对它进行了详细讨论。
- 第15章——使用查询表达式的LINQ：如果只使用标准查询操作符，就会形成让人难以辨认的长语句。然而，查询表达式提供了一种类似SQL风格的语法，能够有效地解决这个问题。这一章会详细讨论这种表达式。
- 第16章——构建自定义集合：在构建用于操纵业务对象的自定义API时，经常都需要创建自定义的集合。本章讨论了具体如何做；同时，还介绍了能使自定义集合的构建变得更简单的上下文关键字。
- 第17章——反射、特性和动态编程：20世纪80年代末，程序结构的思维模式发生了根本性的变化，面向对象的编程是这个变化的基础。类似地，特性使说明性编程和嵌入元数据成为可能，因而引入了一种新的思维模式。这一章探讨了特性的方方面面，并讨论了如何通过反射机制来获取它们。这一章还讨论了如何通过基类库（Base Class Library，BCL）中的序列化框架来实现文件的输入和输出。C# 4.0增加了一个新的关键字，即`dynamic`。该关键字将所有类型检查都移至运行时进行，因而极大扩展了C#能做的事情。

- 第18章——多线程处理：大多数现代的程序都要求使用线程来执行长时间运行的任务，还要确保对并发的事件进行快速响应。随着程序变得越来越复杂，必须采取其他措施来保护这些高级环境中的数据。多线程应用程序的编写是一项复杂的任务。这一章讨论了如何操纵线程，并讲述了如何采取一些必要的措施来防止将多线程应用程序弄得一团糟。
- 第19章——同步和其他多线程处理模式：这一章以第18章为基础，演示了如何利用一些内建的线程处理模式来简化对多线程代码的显式控制。
- 第20章——平台互操作性和不安全的代码：必须认识到的是，C#是相对年轻的一种语言，许多现有的代码是用其他语言写成的。为了用好这些现有的代码，C#通过P/Invoke提供了对互操作性（非托管代码的调用）的支持。除此之外，C#允许使用指针，也允许执行直接内存操作。虽然使用了指针的代码要求特殊的权限才能运行，但它具有与C风格的API完全兼容的能力。
- 第21章——CLI：事实上，C#被设计成一种在CLI的顶部工作的最有效的编程语言。这一章讨论了C#程序与底层“运行时”及其规范的关系。
- 附录A——下载和安装C#编译器与CLI平台：这个附录介绍了如何安装微软.NET和Mono，它们是编译和运行C#代码的基础平台。
- 附录B——完整源代码清单：本书许多章都将源代码分散到多个代码清单中。如果代码清单较大，就会使读者难以跟进。这个附录将第3章、第11章、第12章、第14章和第17章的代码清单整合成完整的程序，便于读者从整体上理解各个单独的代码清单。
- 附录C——来自System.Collections.Concurrent的并发类：本附录提供了.NET Framework 4新增的并发集合示意图。
- 附录D～附录F——C# 2.0、C# 3.0、C# 4.0主题：这些附录为C# 2.0、C# 3.0或C# 4.0内容提供了快速参考。它们旨在帮助程序员快速上手并用好C#功能。

希望本书成为你学习和掌握C#技能的一个好帮手。另外，希望以后需要了解C#的一些特殊主题及其内部工作原理的时候，本书也是一本出色的参考书。

致　谢

世界上没有任何一本书是作者单枪匹马就能出版的，在此，我要向此过程中帮助过我的所有人致以衷心的感谢。

表达感激之情的顺序并不重要，我是想到谁就感谢谁。到现在为止，为了让我顺利完成此书，我的家人做出了巨大的牺牲。在 Banjamin、Hanna 和 Abigail 眼中，他们的爸爸经常因为此书而无暇顾及他们，但 Elisabeth 承受的更多。家里的大事小事全靠她一个人，她独自承担家庭的重任。我也挺希望本书每一次出版都变得更容易一些，但遗憾的是，实情并非如此。随着孩子们越来越大，生活越来越紧张和忙碌。没有我，Elisabeth 几乎时时刻刻都在在承受高强度的压力。我感到万分抱歉，谢谢你！

为保证本书技术上的准确性，许多技术编辑对本书中的各章都进行了仔细审阅。我常常惊讶于他们的认真程度，任何不易察觉的小错误都逃不过他们的火眼金睛，他们是 Paul Bramsman、Kody Brown、Ian Davis、Doug Dechow、Gerard Frantz、Thomas Heavey、Anson Horton、Brian Jones、Shane Kercheval、Angelika Langer、Eric Lippert、John Michaelis、Jason Morse、Nicholas Paldino、Jon Skeet、Michael Strokesbary、Robert Stokesbary、John Timney 和 Stephen Toub。

特别要感谢 Michael，在我编写本书期间，他在编辑技术内容和完善本书结构方面，给予我莫大的帮助，更不用说他给予我的珍贵的友情。同时，我还要特别感谢两位 C# MVP（微软最有价值专家）Nicholas 和 John，他们对这种编程语言某些方面的了解仅次于 C#开发团队。

Eric 给了我太多的惊奇。他对 C#术语的掌握程度令人“望而生畏”，我非常欣赏他的修改，尤其是这反映出他在术语方面力求完美。他对 C# 3.0 相关章节做了不小的改进，在本书的第 2 版中，我唯一感到遗憾的就是未能让他审阅所有章节。然而，这个遗憾终于得到了弥补。Eric 兢兢业业地审阅了本书的每一章，他审得非常细，也非常严谨。正是因为他的辛勤付出，才使本书变得比前两版还要好，我对此致以由衷的感谢。谢谢你，Eric！我想象不出还有谁能比你干得更出色。正因为你，本书才真正实现了从“很好”到极好的飞越。

就像 Eric 之于 C#，很少有人像 Stephen Toub 那样对.NET Framework 多线程处理有如此深刻的理解。Stephen 专门帮我审阅了重写的关于多线程的两章，并重点检查了并行编程的主题。由于几次 Beta 版本发生了一些变化，我在根据 Stephen 的第一次审阅进行了更新之后，请求他对修改过的内容进行二次审阅——他欣然接受了。我真的想不出还有比他更好的人选来帮我审查这方面的内容。谢谢你，Stephen！尤其要感谢在我熟悉新的 API 的过程中对我的耐心。