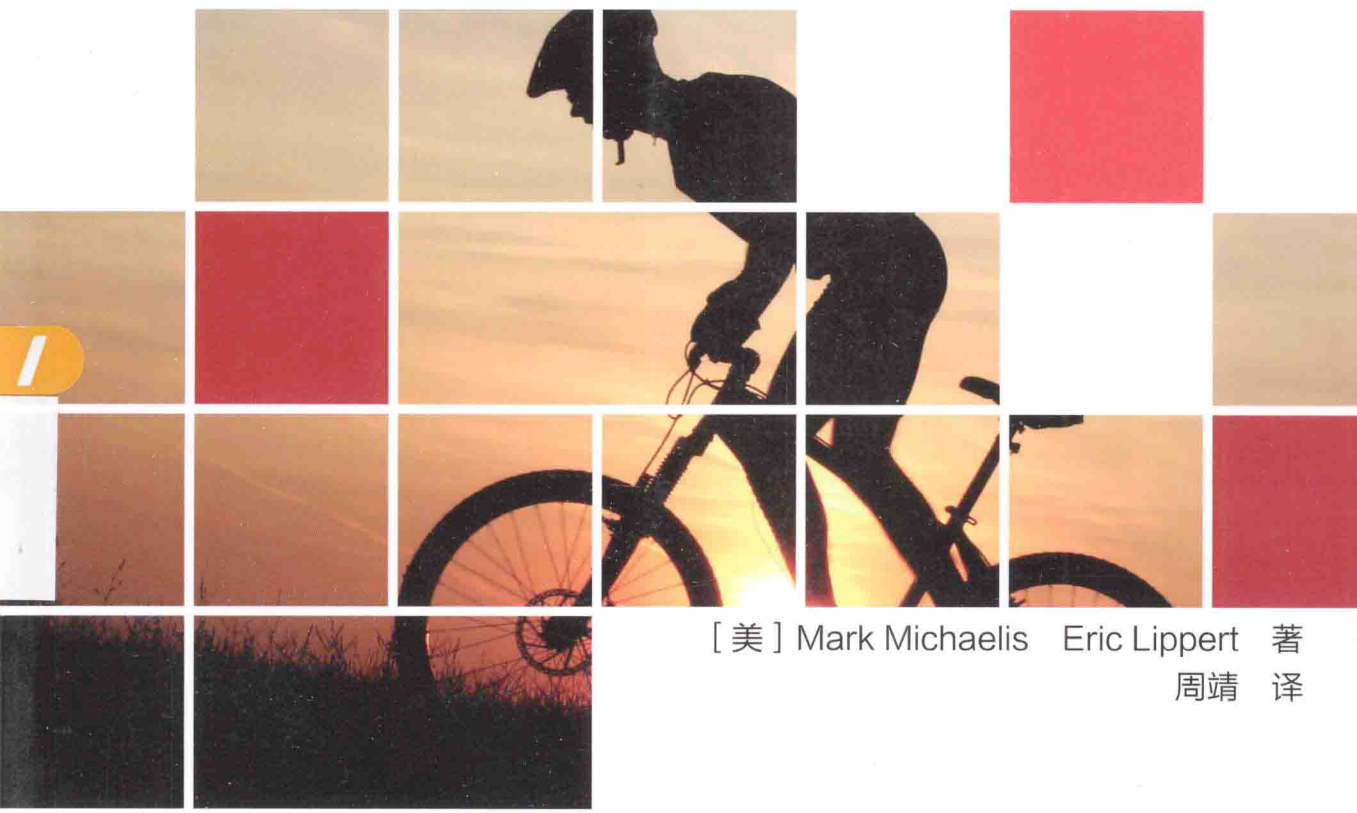
- C# 入门及进阶的首选之作
- 内容全面翔实，讲解精彩权威
- 全新升级版，涵盖 C# 5.0

PEARSON

Essential C# 5.0

C#本质论（第4版）

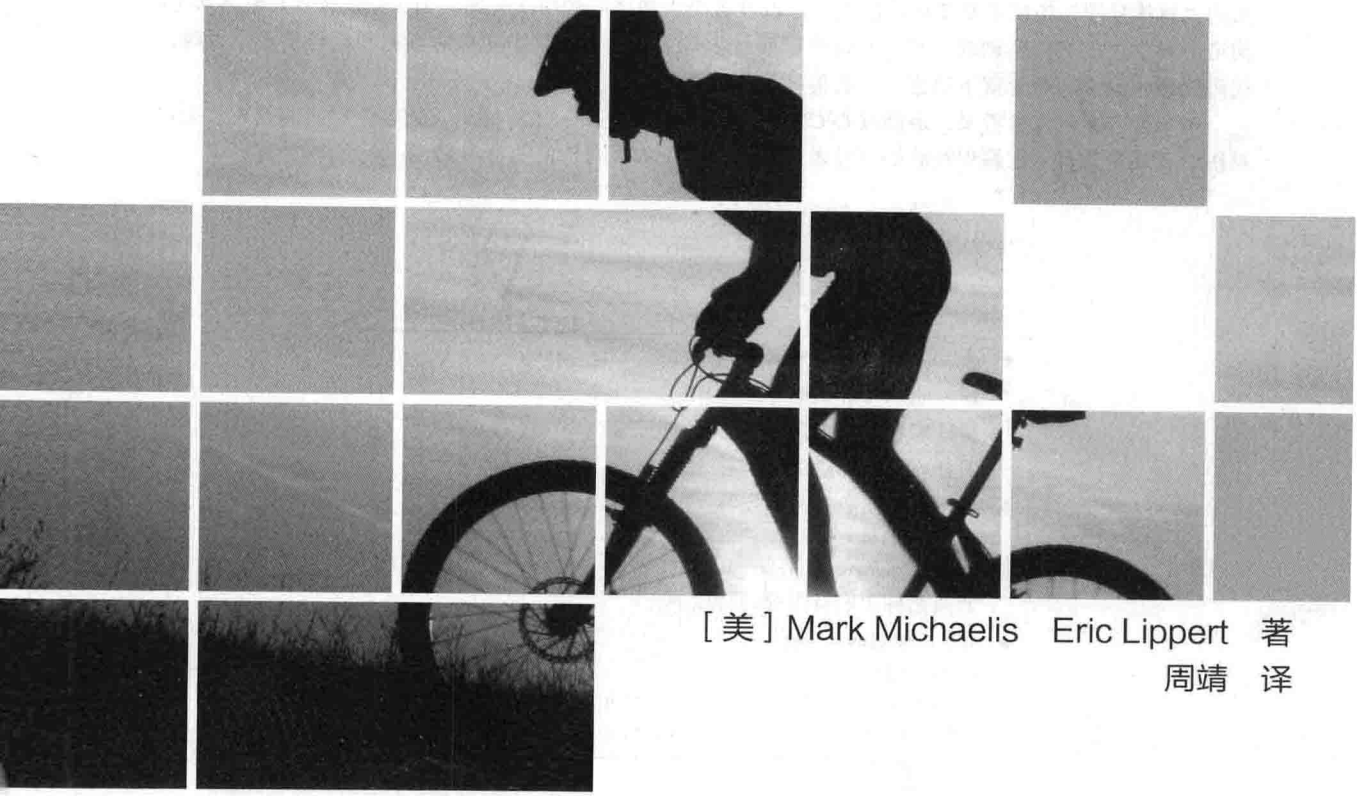


[美] Mark Michaelis　Eric Lippert　著
周靖　译

人民邮电出版社
POSTS & TELECOM PRESS

Essential C# 5.0

C#本质论（第4版）



[美] Mark Michaelis　Eric Lippert　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4版 / (美) 米凯利斯(Michaelis,M.), (美) 利珀特 (Lippert,E.) 著 ; 周靖译. -- 北京 : 人民邮电出版社, 2014.3
书名原文: Essential C# 5.0
ISBN 978-7-115-33675-0

Ⅰ. ①C… Ⅱ. ①米… ②利… ③周… Ⅲ. ①C语言－程序设计 Ⅳ. ①TP312

中国版本图书馆CIP数据核字(2013)第273860号

内 容 提 要

这是 C# 领域中一部广受好评的名作，作者用一种易于理解的方式详细介绍了 C# 语言的各个方面。全书共有 21 章和 4 个附录，介绍了 C# 语言的数据类型、操作符、方法、类、接口、异常处理等基本概念，深入讨论了泛型、迭代器、反射、线程和互操作性等高级主题，还介绍了 LINQ 技术，以及与其相关的扩展方法、分部方法、Lambda 表达式、标准查询操作符和查询表达式等内容。每章开头的“思维导图”指明本章要讨论的主题，以及各个主题之间的层次关系。为了帮助读者理解各种 C# 构造，书中用丰富的示例演示每一种特性，而且为每个概念都提供了相应的规范和最佳实践，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。

本书是一本语言参考书，遵循核心 C# 5.0 语言规范，适合对 C# 感兴趣的各个层次的读者。无论对初学者还是具有一定编程经验的开发者，本书都是一本很有价值的参考书。

◆ 著　[美] Mark Michaelis　Eric Lippert
译　周　靖
责任编辑　杨海玲
责任印制　程彦红　杨林杰

◆ 人民邮电出版社出版发行　北京市丰台区成寿寺路 11 号
邮编　100164　电子邮件　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：787×1092　1/16
印张：42.25
字数：901 千字　2014 年 3 月第 4 版
印数：13 501 – 17 000 册　2014 年 3 月北京第 1 次印刷

著作权合同登记号　图字：01-2012-9275 号

定价：108.00 元

读者服务热线：(010)81055410　印装质量热线：(010)81055316
反盗版热线：(010)81055315
广告经营许可证：京崇工商广字第 0021 号

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

强强联手，造就了这本伟大的C#书籍！Mark Michaelis的《C#本质论》系列早已成为经典，如今更与著名的C#博主Eric Lippert携手，使之升华为无与伦比的大师级著作。

一般人的印象是Eric写博客，而Mark写书，但我刚认识他们的时候可不是这样的。

2005年LINQ（语言集成查询）公布时，我才刚加入Microsoft，正好见证了PDC会议上的令人激动的公开发布。虽然对技术本身几乎没有什么贡献，但它的宣传造势我可是全程参加了。那个时候人人都在谈论它，宣传小册子满天飞。那是C#和.NET的大日子，至今仍然令人难忘。

但会场的实验室区域却相当安静，在那儿，人们可以自行按照精心编写的实验指南进行技术预览。我就是在那儿遇见Mark的。不用说，他一点儿没有按部就班的意思。他在做自己的实验，梳理文档，和别人沟通，忙着收集自己的照片。

作为C#社区的新人，我在那次会议上见到了许多名人。但老实说，后来都记不大清楚了。唯一记得住的就是Mark。因为当我问他是否喜欢这个新技术时，他不像别人那样马上开始滔滔不绝，而是非常冷静地说："还不确定，我还没有完全了解它。"他希望完整地理解并消化这一技术，而在这之前，他不希望别人的先入之见影响他的思考。

所以并没有像我预想的那样发生一次快餐式的对话。相反，我们的对话相当坦诚而且颇有裨益。像这样的交流好多年都没有发生了。新技术的细节、造成的后果和存在的问题都涉及了。对我们这些语言设计者来说，Mark是最有价值的社区成员，因为他非常聪明，会打破砂锅问到底，能深刻理解一种技术对于真正的开发人员的影响。但是，最根本的原因可能还是他的坦诚，他从不惧怕说出自己的想法。如果我们开发的某种技术能够通过Mark的测试，就没有什么好担心的了！

这些特质也使Mark成为一名出色的作家。他的文字直指技术的本质，向读者提供最完整的信息，没有废话，能敏锐地看出技术的真正价值和问题。

Eric是C#团队中和我共事7年的同事。他在这里的资格比我老得多。回想第一次见到他时，他正在向团队解释如何理清乱局。确切地说，当时C#编译器代码库正需要在架构上进行一些重大调整，而新功能却很难加入进来——我们迫切需要LINQ来解决这些问题。Eric已经调查好了我们需要有什么样的架构（阶段啊阶段！当时甚至根本就没有这个概

念！）。更重要的是，如何逐步跟进。令人惊讶的是，虽然这个主题是如此复杂，作为团队和代码库新人的我，居然立即就明白了他所讲的东西！

从博客就能看出，他讲东西超清楚，而且结构合理，读者能快速地把一个问题搞明白，除了能收获解决方案之外，阅读过程同样令人愉悦。还不止如此！每次Eric钻研一个复杂的问题并与团队分享见解时，他的电子邮件都是那么一丝不苟又充满乐趣。基本上不可能忽视Eric提出的任何问题，因为你会迫不及待地想要看完他的文章。除此之外，他的文字还非常大气！所以，我超喜欢看他的东西，包括经常更新的博客文章。作为C#编译器团队和语言设计团队的一员，他为我们带来的愉悦和帮助真是太多了。

总之，能和这两位出色的伙伴共事，我深感荣幸。Eric帮我理清思路，Mark帮我说出真心话。他们帮助读者认清楚问题并掌握解决方案。通过分别主攻C#的“内”和“外”，本书的这个版本在完整性方面达到了前所未有的高度。没有人能像这两位大师一样帮助你正确地理解C# 5.0。

请好好享用本书！

——Mads Torgersen
微软公司C#项目经理

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范型的重大转变。每一种范型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。本书将带领你体验这样的范型转变过程。

本书开始的几章指导你学习**顺序编程结构**。在这种编程结构中，语句是按照编写的顺序来执行的。这种结构的问题在于，随着需求的增加，复杂性会按指数级增长。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**。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种主要职能。

- 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范型，并深入讨论了语言最新版本（C# 5.0和.NET 4.5）的新功能。
- 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]思想的笃信者，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。幸运的是对于这些主题，都已经有丰富的图书供读者选择了。

① 与声明性编程或者宣告式编程（declarative programming）对应的是命令式编程；前者表述问题，后者实际解决问题。——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

本书的重点在于C#以及基类库中的类型。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。

本书面向的读者

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assembly、link、chain、thread和fusion等字眼而打击初学者的信心，否则许多人会误以为这是一本讲冶金而不是程序设计的书[①]。本书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。但我还是认真编排了本书的内容，以便使各种层次的开发者都能够从中获益。

- *初学者*：假如你是编程新手，本书将帮助你从入门级程序员过渡成为C#开发者，消除以后在面临任何C#编程任务时的害怕心理。本书不仅要教会你语法，还要教你养成良好的编程习惯，为将来的编程生涯打下良好基础。
- *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，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规范及模式。你教别人学C#时，本书也颇有助益。随着C# 3.0、C# 4.0和C# 5.0的出现，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）；
  - 泛型（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

- 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）；
- 用任务编程库和async进行多线程编程（参见第18章）；
- 用PLINQ进行并行查询处理（参见第18章）；
- 并发集合（第19章）。

考虑到许多人还不熟悉这些主题，因此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#5.0 语言规范。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规范和最佳实践，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。

C#编码规范

本书新版本最重大的改进之一就是增加了大量编码规范，例如第16章中的一个规范如下所示：

> **规范**
>
> 要确保相等的对象有相同的散列码。
>
> 要确保对象的散列码在一个散列表中永不变化。
>
> 要确保散列算法快速产生良好分布的散列码。
>
> 要确保散列算法在任何可能的对象状态中的健壮性。

一名知道语法的程序员和一名能因时宜地写出最高效代码的专家的区别，关键就是这些编码规范。专家不仅让代码通过编译，还遵循最佳实践，降低产生bug的概率，并使代码的维护变得更容易。编码规范强调了一些关键原则，开发时务必注意。

代码示例

本书大多数代码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Microsoft .NET平台。除了在解释只和某些平台有关的重要概念的地方（例如，解释如何正确处理Windows单线程用户界面），我很少使用平台或厂商特有的库。

下面是一个示例代码清单。

代码清单1-9　变量的声明和赋值

```
class MiracleMax
{
  static void Main()
  {
        数据类型
      string max;
             变量
      max = "Have fun storming the castle!";

      System.Console.WriteLine(max);
  }
}
```

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，表示无关的代码已省略。

```
//…
```

- 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
```

- 执行程序时要由用户输入的内容加粗显示。

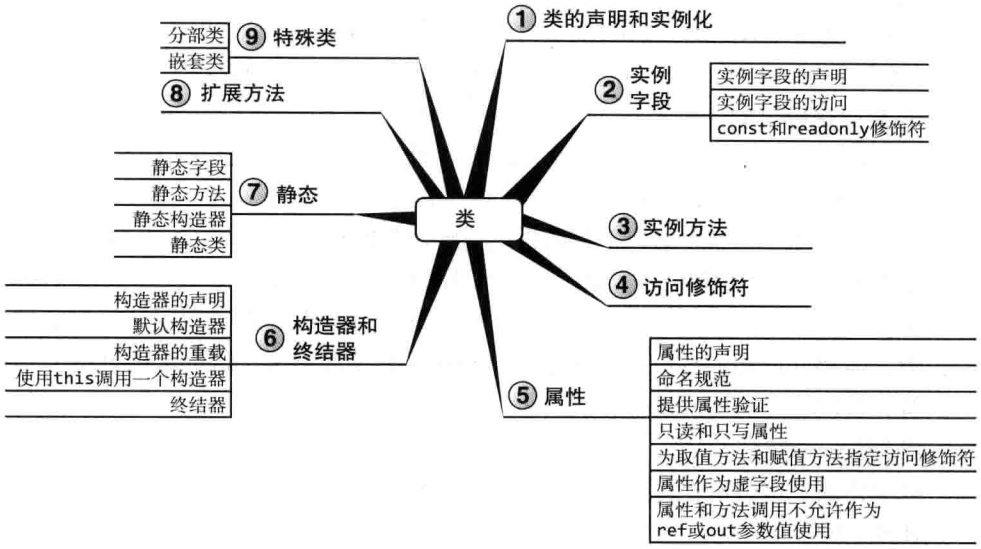
虽然提供完整的代码示例可以方便地复制到自己的程序中，但这样做会降低学习特定主题的意义。因此，需要自行修改代码示例，然后再把它们集成到自己的程序中。代码示例中最大的问题是省略了像异常处理这样的错误检查。此外，书中的代码示例没有显式地包含using System语句；在所有的例子中，这个语句都是必需的。

请访问intellitect.com/essentialcsharp和informit.com/mswinseries下载示例代码。[①]

① 本书中文版的资源下载和勘误也通过译者博客（transbot.blog.163.com）提供。——译者注

思维导图

每章开头都有一幅**思维导图**[①]。作为提纲，它的作用是为读者提供对每章内容的快速参考。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

根据自己的经验水平，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。
- *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[②]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

总地说来，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更高级的主题。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

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

② 标注框，即callout box，是一种需要准确排版的书稿元素。比如： 

。——译者注

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。这一章针对开发平台对C#语言进行描述。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第1章之后。

下面是每一章的内容提要（**使用黑体**的章号表明那一章的内容在C# 3.0~C#5.0版本中都有）。

- 第1章——C#概述：这一章在展示了用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，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——操作符和控制流：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#操作符、数据转换以及预处理指令。
- **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#提供了独特的构造，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，解释了如何实现操作符，比如“+”和转型操作符，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）的核心构造。本章还探讨了匿名方法，这也是C# 2.0新增的。
- **第14章**——支持标准查询操作符的集合接口：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“标准查询操作符”，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**——线程同步：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——Tic-Tac-Toc源代码清单：第3章和第4章用到的程序在这里提供了完整源代码。
- 附录C——使用TPL和C# 5.0之前的多线程处理模式：这个附录详细说明了在使用C# 5.0和/或任务并行库（TPL）之前的版本进行开发时的多线程处理模式。
- 附录D——C# 5.0的Async/Await模式之前的计时器：这个附录描述了在.NET 4.5/C# 5.0之前使用的三种不同类型的计时器。

希望本书成为你学习和掌握C#技能的一个好帮手。另外，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。

——Mark Michaelis

IntelliTect.com/mark

Twitter: @Intellitect, @MarkMichaelis

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。家里的大事小情全靠她一个人，她独自承担家庭的重任。我原本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。还要感谢Mandy Frei辛苦记录大量再版时需要的改动。

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，尤其是他在术语方面表现出的力求完美。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兢兢业业地审阅了这一版的每一章，他审得非常仔细，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做得更出色。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专门审阅了重写的（嗯，这是第三次了）关于多线程的两章，并重点检查了C# 5.0的异步支持。谢谢你，Stephen！

感谢Addison-Wesley的所有员工，感谢他们在与我合作期间表现出来的耐心，容忍我将注意力频频转移到书稿之外的其他事情。感谢Elizabeth Ryan、Audrey Doyle、Vicki Rowland、Curt Johnson和Joan Murray。尤其要感谢Joan，虽然很多次我不仅交稿延误，而且回邮件还特别慢，但她却总是富有耐心。