开源.NET 生态软件开发

C#入门经典

(第 9 版)

[德] 本杰明 • 帕金斯(Benjamin Perkins) 著
乔恩 • D. 里德(Jon D. Reid)
齐立博 译

清华大学出版社
北 京

北京市版权局著作权合同登记号　图字：01-2021-1029

图书在版编目(CIP)数据

C#入门经典：第9版 / (德) 本杰明 • 帕金斯，(德)乔恩 • D. 里德著；齐立博译. —北京：清华大学出版社，2022.3
(开源.NET生态软件开发)
书名原文：Beginning C# and .NET 2021 Edition
ISBN 978-7-302-60303-0

Ⅰ. ①C…　Ⅱ. ①本… ②乔… ③齐…　Ⅲ. ①C 语言—程序设计　Ⅳ. ①TP312.8

中国版本图书馆 CIP 数据核字(2022)第 039218 号

责任编辑：王　军　韩宏志
装帧设计：孔祥峰
责任校对：成凤进
责任印制：宋　林

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　邮　　编：100084
社 总 机：010-83470000　　邮　　购：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 装 者：大厂回族自治县彩虹印刷有限公司
经　　销：全国新华书店
开　　本：170mm×240mm　　**印　　张：**39.5　　**字　　数：**1185 千字
版　　次：2022 年 4 月第 1 版　　**印　　次：**2022 年 4 月第 1 次印刷
定　　价：118.00 元

产品编号：095343-1

译 者 序

.NET 是 Windows 系统下的环境模型工具。.NET 经过几代的发展，现已将功能性与技术性完美结合起来。.NET 可用于创建任意基于 Windows 系统的应用程序，支持各种业务流程，是程序开发必不可少的工具。

.NET 的功能包括：

- 面向对象的编程环境。
- 软件部署和版本控制冲突最小化。
- 可提高代码(包括由未知的或不完全受信任的第三方创建的代码)的执行安全性。
- 可消除脚本环境或解释环境的性能问题。
- 按照行业标准生成所有通信。

C#旨在设计成为一种简单、现代、通用以及面向对象的程序设计语言；此种语言的实现，应提供对于以下软件工程要素的支持：强类型检查、数组维度检查、未初始化的变量引用检测、自动垃圾收集(Garbage Collection，指一种自动内存释放技术)。软件必须做到强大、持久，并具有较高的编程生产效率。C#语言为在分布式环境中的开发提供适用的组件。

本书从初学者角度出发，围绕 C#语言的基础知识和新功能，详细介绍使用 C#进行应用程序开发应该掌握的各方面技术，语言通俗易懂、实例丰富多彩。所有知识都结合具体实例进行介绍，涉及的程序代码给出了详细注释，可使读者轻松领会 C#应用程序开发的精髓，快速提高开发技能。

全书分为 3 个部分，共 21 章。无论是刚开始接触面向对象编程的新手，还是打算迁移到 C#的 C、C++或 Java 程序员，都可以从本书汲取到新的知识。迅速掌握 C#编程技术。

对于这本经典之作，译者本着“诚惶诚恐”的态度，在翻译过程中力求“信、达、雅”，但鉴于译者水平有限，失误在所难免，如有任何意见和建议，请不吝指正。感激不尽！最后，希望读者通过阅读本书能早日步入 C#语言编程的殿堂，领略 C#语言之美！

作 者 简 介

Benjamin Perkins(MBA、MCSD、ITIL)目前在微软(德国慕尼黑)工作，是一位资深的高级工程师，在 IT 行业工作了二十多年。他 11 岁时就开始在 Atari 1200XL 台式机上用 QBasic 编写计算机程序。他喜爱诊断和排除技术问题，品味写出好程序的乐趣。高中毕业后，他加入美国军队。在成功服完兵役后，他进入得克萨斯州的得克萨斯 A&M 大学，在那里他获得管理信息系统的工商管理学士学位。

他在 IT 行业的足迹遍及整个行业，包括程序员、系统架构师、技术支持工程师、团队领导。在受雇于惠普时，他获得了诸多奖项、学位和证书。他对技术和客户服务富有激情，期待排除故障，编写出更多世界级技术解决方案。

“我的方法是烂熟于心之后才编写代码，完整、正确地编写一次，这样就不需要再次考虑它，除非要改进它。”

Benjamin 写过许多杂志文章和培训课程，还是一个活跃的博主。他撰写的书涉及 C#编程、IIS、NHibernate、开源和微软 Azure。

- 在 LinkedIn 上与 Benjamin 联系：www.linkedin.com/in/csharpguitar
- 在 Twitter 上关注 Benjamin：@csharpguitar: twitter.com/csharpguitar
- 阅读 Benjamin 的博客：www.thebestcsharpprogrammerintheworld.com
- 在 GitHub 上访问 Benjamin：github.com/benperk

Benjamin 与妻子 Andrea 以及两个可爱的孩子 Lea 和 Noa 一起快乐地生活。

Jon D. Reid 担任 IFS Field Service Management(www.IFSWORLD.com)的 IFS AB (www.ifs.com)研发项目经理，专注于现场服务管理。他已与他人合著了多本关于微软技术的图书，包括 *Beginning C# 7 Programming with Visual Studio 2017*、*Fast Track C#*和 *Pro Visual Studio .NET* 等。

技术编辑简介

Rod Stephens 是一位长期的开发人员和作者，他写了 250 多篇杂志文章和 35 本书，这些文章被翻译成不同语言文字在世界各地转载。在他的职业生涯中，Rod 曾在电话交换、账单、维修调度、税务处理、废水处理、音乐会门票销售、制图和职业足球队训练等领域开发多种应用。

Rod 的 C#助手网站(www.csharphelper.com)每年都有数百万的点击率，其中包含 C#程序员的技巧和示例程序。他的 VB 助手网站(www.vb-helper.com)也为 Visual Basic 程序员提供了类似的资料。

可通过 RodStephens@csharphelper.com 或 RodStephens@vb-helper.com 联系 Rod。

致　　谢

为使本书内容以清晰美观的形式呈现给学生和专业人士，使他们从中获益，需要做大量工作。作者的确有卓越的技术知识和经验与大家分享，但如果没有技术作家、技术评审人员、开发人员、编辑、出版人员、平面设计师等提供有价值的帮助，就不可能编写出高质量的书籍。编程技术日新月异，在有效技术过时之前，个人无法凭一己之力完成所有这些任务。正因为如此，作者只有与伟大的团队合作，才能很快把本书的所有组件组合在一起，才能确保把最新信息传达给读者，帮助读者了解最新功能。感谢 Rod Stephens 在整个过程中做技术审查和提供建议。最后，感谢在幕后帮助本书出版的所有人员。

前　言

C#是 Microsoft 于 2002 年推出.NET Framework 的第 1 版时提供的一种全新语言。C#从那时起迅速流行开来，成为使用.NET Framework 的桌面、Web、云和跨平台开发人员无可争议的选择。开发人员喜欢 C#的一个原因是其继承自 C/C++的简洁明了的语法，这种语法简化了以前给程序员带来困扰的一些问题。尽管做了这些简化，但 C#仍保持了 C++原有的功能，所以现在没理由不从 C++转向 C#。C#语言并不难，也非常适合开发人员学习基本编程技术。易于学习，再加上.NET Framework 的功能，使 C#成为开始你编程生涯的绝佳方式。

C#的最新版本 C# 9 是.NET 5.0 和.NET Framework 4.8 的一部分，它建立在已有的成功基础之上，还添加了一些更吸引人的功能。Visual Studio 的最新版本 Visual Studio 和开发工具的 Visual Studio Code 系列也有许多变化和改进，这大大简化了编程工作，显著提高了效率。

本书将全面介绍 C#编程的所有知识，从该语言本身一直到桌面编程、云编程和跨平台编程，再到数据源的使用，最后是一些新的高级技术。我们还将学习 Visual Studio 的功能和利用它开发应用程序的各种方式。

本书文笔优美流畅，阐述清晰，每一章都以前面章节的内容为基础，便于读者掌握高级技术。每个概念都会根据需要介绍和讨论，而不会突然冒出某个技术术语妨碍读者的阅读和理解。本书尽量减少使用的技术术语数量，但如有必要，将根据上下文进行正确的定义和布置。

本书作者都是各自领域的专家，都是 C#语言和.NET Framework 的爱好者，没人比他们更有资格讲授 C#了，他们将在你掌握从基本原理到高级技术的过程中为你保驾护航。除基础知识外，本书还有许多有益的提示、练习、完全成熟的示例代码(可扫描封底二维码下载)，在你的职业生涯中一定会反复用到它们。

本书将毫无保留地传授这些知识，希望读者能通过阅读本书成为最优秀的程序员。

0.1　本书读者对象

本书面向想学习如何使用.NET 编写 C#程序的所有人。本书针对的是想要通过学习一种干净、现代、优雅的编程语言来掌握程序设计的完完全全的初学者。但是，对于熟悉其他编程语言、想要探索.NET 平台的读者，以及想要了解旗舰语言.NET 的开发人员，本书同样很有价值。

0.2　本书内容

本书前面的章节介绍 C#语言本身，读者不需要具备任何编程经验。以前对其他语言有一定了解的开发人员，会觉得这些章节的内容非常熟悉。C#语法的许多方面都与其他语言相同，许多结构对所有的编程语言来说都是相通的(例如，循环和分支结构)。但是，即使是有经验的程序员也可以通过这些章节理解此类技术应用于 C#的特征，从而从中获益。

如果读者是编程新手，就应从头开始学习，了解基本的编程概念，并熟悉 C#和支持 C#的.NET 平台。如果读者对.NET 比较陌生，但知道如何编程，就应阅读第 1 章，然后快速跳读后面几章，这

样就能掌握 C#语言的应用方式了。如果读者知道如何编程，但以前从未接触过面向对象的编程语言，就应从第 8 章开始阅读。

如果读者对 C#语言比较了解，就可以集中精力学习那些详细论述最新.NET 和 C#语言开发的章节，尤其是集合、泛型和 C#语言新增内容的相关章节(第 11 章和第 12 章)。

本书章节的编排方式可以达到两个目的：可以按顺序阅读这些章节，将其视为 C#语言的一个完整教程；还可以按照需要深入学习这些章节，将其作为一本参考资料。

除核心内容外，从第 3 章开始，大多数章节的末尾还包含一组习题，完成这些习题有助于读者理解所学的内容。习题包括简单的选择题、判断题以及需要修改或创建应用程序的较难问题。附录中给出了全部习题的答案。这些习题也可以通过本书的配套网站 www.wrox.com 下载，它们是 wrox.com 代码下载的一部分。

随着 C#和.NET 新版本的发布，对每一章都进行了彻底的检查，删掉了不太相关的内容，增加了新内容。所有代码都在最新版本的开发工具上进行了测试，所有屏幕截图都在 Windows 操作系统上重新截取，以提供最新的窗口和对话框。

本书的亮点包括：

- 增加并改进了代码示例。
- 增加了编写跨平台运行的 ASP.NET Core 应用程序的示例。
- 增加了编写云应用程序的示例，并使用 Azure SDK 创建和访问云资源。

0.3 本书结构

本书分为 3 大部分。

- **C#语言**：介绍 C#语言的所有内容，从基础知识到面向对象的技术，一应俱全。
- **数据访问**：介绍如何在应用程序中使用数据，包括存储在硬盘文件中的数据、以XML格式存储的数据和数据库中的数据。
- **云和跨平台编程**：讲述使用C#和.NET的一些额外方式，包括云和跨平台开发、ASP .NET Web API、Windows Presentation Foundation (WPF)、Windows Communication Foundation (WCF)和Universal Windows Applications。

下面介绍本书 3 个重要部分中的章节。

0.3.1 C#语言(第 1～13 章)

第 1 章介绍 C#及其与.NET 的关系，了解在这个环境下编程的基础知识，以及 Visual Studio 与它的关系。

第 2 章开始介绍如何编写 C#应用程序，学习 C#的语法，并将 C#和示例命令行、Windows 应用程序结合起来使用。这些示例将说明如何快速轻松地启动和运行 C#，并附带介绍 Visual Studio 开发环境以及本书将要使用的基本窗口和工具。

接着将学习 C#语言的基础知识。第 3 章介绍变量的含义以及如何操纵它们。第 4 章将用流程控制(循环和分支)改进应用程序的结构，第 5 章介绍一些更高级的变量类型，如数组。第 6 章开始以函数形式封装代码，这样就更易于执行重复操作，使代码更容易让人理解。

从第 7 章将运用 C#语言的基础知识，调试应用程序。这包括在运行应用程序时输出跟踪信息，使用 Visual Studio 查找错误，在强大的调试环境中找出解决问题的办法。

第 8 章将学习面向对象编程(Object-Oriented Programming，OOP)。首先了解这个术语的含义，回答“什么是对象？”OOP 初看起来是较难的问题。我们将用一整章的篇幅来介绍它，解释对象的强大之处。直到该章的最后才会真正使用 C#代码。

第 9 章将理论知识应用于实践，当开始在 C#应用程序中使用 OOP 时，这才体现出 C#的真正威力。在第 9 章介绍如何定义类和接口之后，第 10 章将探讨类成员(包括字段、属性和方法)，在该章的最后将开始创建一个扑克牌游戏，这个游戏将在后续章节中逐步开发完成，它非常有助于理解 OOP。

学习了 OOP 在 C#中的工作原理后，第 11 章将介绍几种常见的 OOP 场景，包括处理对象集合、比较和转换对象。第 12 章讨论.NET 2.0 中引入的一个非常有用的 C#特性——泛型，利用它可以创建非常灵活的类。第 13 章通过其他一些技术(主要是事件，它在 Windows 编程中非常重要)继续讨论 C#语言和 OOP。最后介绍 C#最新版本中引入的新特性。

0.3.2　数据访问(第 14～17 章)

第 14 章介绍应用程序如何将数据保存到磁盘以及如何检索磁盘上的数据(作为简单的文本文件或者更复杂的数据表示方式)。该章还将讨论如何压缩数据，以及如何监视和处理文件系统的变化。

第 15 章介绍数据交换的事实标准 XML，简要论述 JSON 格式。该章将讨论 XML 的基本规则，论述 XML 的所有功能。

该部分的其余章节介绍 LINQ(这是内置于.NET 中的查询语言)。第 16 要介绍 LINQ。第 17 论如何使用 LINQ 访问数据库和其他数据。

0.3.3　其他技术(第 18～21 章)

第 18 章介绍.NET Standard 和.NET Core，它们是面向任何应用程序类型(如 WPF、Windows 和 ASP.NET)的工具，新兴的应用程序可以在 Linux 或 macOS 等平台上运行。该章讨论.NET 标准库的安装、创建和实现指令，还描述了 ASP.NET 和它的许多不同类型(例如，ASP.NET Webforms、ASP.NET MVC 和 ASP .NET Core)。

第 19 章首先描述什么是云编程，并讨论了云优化的堆栈。云环境与传统的程序编码方式不同，因此讨论了一些云编程模式。要完成这一章，需要一个 Azure trail 账户，它是免费创建的，并附带一些积分，这样就可以创建和测试一个 App Service Web 应用程序。然后使用 Azure SDK 和 C#，创建并访问 ASP .NET Web 应用程序中的存储账户。

第 20 章将学习如何创建 ASP.NET Web API，并通过 Blazor WebAssembly App 使用它。然后，该章介绍了 Windows Communication Foundation (WCF)，它为在企业级以编程方式跨本地网络和 Internet 访问信息和功能提供了许多工具。该章将介绍如何以平台无关的方式使用 WCF，向 Web 应用程序和桌面应用程序公开复杂的数据和功能。

第 21 章首先介绍什么是 Windows 编程，并看看如何在 Visual Studio 中实现。将 WPF (Windows Presentation Foundation)作为一种工具，以图形化方式构建桌面应用程序，并以最少的努力和时间组装高级应用程序。你将从 WPF 编程的基础知识开始，逐步积累到更高级的概念。

0.4　使用本书的要求

本书中 C#和.NET Framework 的代码和描述都适用于 C# 9 和.NET Framework 4.8。除了.NET 之外，不需要其他组件就可以理解本书这方面的内容，但书中许多示例都需要使用开发工具。本书将 Visual

Studio Community 2019 作为主要开发工具。使用 Visual Studio Community 2019 来创建 Windows 应用程序、云应用程序、跨平台的应用程序，以及访问数据库的 SQL Server Express 应用程序。

可扫描封底二维码下载全书代码。

0.5 本书约定

为了帮助读者在阅读本书的过程中获取最多信息，并随时了解当前处理的事项，本书使用了许多约定。

试一试

“试一试”是一个应该跟随书中的文本完成的练习。

1. 这些练习通常包括一组步骤。
2. 每一步都有一个数字。
3. 按照这些步骤走到底。

示例说明

在每个“试一试”之后，会详细解释输入的代码。

警告：

包含重要且应该记住的信息，这些信息与周围的文字直接关联。

注意：

表示注释、提示、暗示、技巧或对当前讨论的弦外之音。

本书通过两种方式来显示代码：

- 对于大多数代码示例，使用没有突出显示的等宽字体来表示。
- 对在当前上下文中特别重要的代码，用粗体字强调显示。

0.6 源代码

在读者学习本书中的示例时，可以手工输入所有的代码，也可以使用本书附带的源代码文件。本书使用的所有源代码都可通过扫描封底二维码下载。

大部分代码都以.ZIP、.RAR 或者适合平台的类似归档格式进行了压缩。下载代码后，只需要用合适的解压缩工具对它进行解压缩即可。

目　录

第 I 部分
C# 语 言

第1章 C# 简 介

本章内容：

- .NET 的含义
- C#的含义
- Visual Studio

本书第 I 部分将介绍使用最新版本的 C# 语言所需的基础知识。第 1 章将概述.NET 和 C#，包括这两项技术的含义、作用及相互关系。

首先讨论.NET。这种技术包含的许多概念初看起来都不是很容易掌握。也就是说，我们必须在很短的篇幅介绍许多新概念，但快速浏览这些基础知识对于理解如何利用 C#进行编程是非常重要的，本书后面将详细论述这里提到的许多话题。

在对.NET 进行了全面介绍后，本章提供了 C#本身的基本描述，包括它的起源以及与 C++的相似之处。最后，你将看到本书中使用的主要工具：Visual Studio(VS)。Visual Studio 是一个集成开发环境(IDE)，微软从 20 世纪 90 年代末开始开发它，并定期更新它的特性。Visual Studio 包括各种各样的功能，包括对桌面、云、Web、移动、数据库、机器学习、人工智能和跨平台编程的完整支持。

1.1 .NET 的含义

.NET 是 Microsoft 为开发应用程序创建的一个具有革命意义的平台。首先，请注意.NET 提供的不仅是创建针对 Windows 操作系统的程序的方法。.NET 是完全开源的，完全支持跨平台运行。跨平台意味着用.NET 编写的代码也可以在 Linux 和 macOS 操作系统上运行。.NET 的源代码是开源的，可在 github.com/dotnet/core 上找到。

.NET 软件框架由预先编写的计算机代码组成，提供了对基本计算资源(如硬盘驱动器和计算机内存)的简单访问。这个框架的一个方面称为基类库(BCL)，包含 System 类。随着阅读的深入，你会对它非常熟悉。更深入地研究 System 类内部的源代码，会发现它包括数据类型的定义，如字符串、整数、布尔值和字符。如果程序需要这些数据类型中的一种来存储信息，就可以使用已经编写好的.NET 代码来实现。如果这样的代码还不存在，就需要使用汇编或机器码等低级编程语言来自己分配和管理所需的内存。System 类中的基本类型还促进了.NET 编程语言之间的互操作性，这一概念被称为公共

类型系统(Common Type System，CTS)。互操作性意味着 C#中的字符串与 Visual Basic 或 F#中的字符串具有相同的属性和行为。除了提供这个源代码库，.NET 还包括公共语言运行库(CLR)，CLR 负责执行使用.NET 库开发的所有应用程序；稍后再详细介绍。

除了 System 类，.NET 还包含许多其他类，通常称为模块。有些人会说它是一个庞大的 OOP 代码库，这些代码被分类为不同的模块——根据想要实现的结果，可以使用其中的一部分。例如，System.IO 和 System.Text 是用于读取和写入计算机硬盘驱动器上的文件的类。程序员可以简单地使用 System.IO 类中已经存在的代码来操作文件的内容，而不需要管理句柄或从硬盘驱动器将文件加载到内存。.NET 中存在许多帮助程序员快速编写程序的类，因为完成任务需要的所有底层代码都已经编写好了，编程人员只需要知道他们需要哪些类来实现编程目标。

.NET 不仅加速了应用程序的开发，还可以被包括 C#在内的其他许多编程语言所利用。用 C++、F#、Visual Basic 甚至 COBOL 等较老语言编写的程序都可以使用.NET 中存在的类。这些语言可以访问.NET 库中的代码，但是用一种编程语言编写的代码可以与来自另一种编程语言的代码进行通信。例如，用 C#编写的程序可以使用 Visual Basic 或 F#编写的代码，反之亦然。所有这些例子都使.NET 成为构建定制软件的正确选择。

1.1.1 .NET Framework、.NET Standard 和.NET Core

当.NET 框架最初创建时，它面向 Windows 操作系统平台。多年来，.NET Framework 代码被分叉，以支持许多其他平台，如物联网设备、台式机、移动设备和其他操作系统。你可能认识一些以.NET Compact Framework、.NET Portable 或.NET Micro Framework 的名称命名的分支。每个分叉都包含自己的稍微修改过的 BCL。请注意，BCL 不仅仅是字符串、布尔值和整数，还包括文件访问、字符串操作、管理流、在集合中存储数据、安全属性等功能。

即使有一个稍微不同的 BCL，程序员也需要学习、开发和管理每个.NET 分支的 BCL 之间的细微差别。尽管每个程序都使用.NET，但针对桌面、互联网或移动平台的.NET Framework 的每个分支都可能有显著的实现差异。对于公司来说，在不同的平台上运行相同程序逻辑的桌面、网站和电话应用程序是十分常见的(现在仍然如此)。这种情况下，使用.NET 需要为每个平台提供公司应用程序的版本；这是没有效率的。这就是.NET Standard 解决的问题。.NET Standard 为程序员提供了一个位置来创建可以跨.NET Framework 的任何分支使用的应用程序逻辑。.NET Standard 通过将公司的程序逻辑与特定于平台的依赖解耦，使得不同平台(如桌面、移动和 Web)与 BCL 无关。

.NET Core 是.NET 库的开源、跨平台版本。这个代码分支可用来创建针对众多不同平台和操作系统的程序，如 Linux、macOS 和 Windows。它也最终成为.NET 源代码库唯一维护的分支。到 2020 年，对.NET Framework、.NET Standard 和.NET Core 的了解已经不再像以前那样重要了。这里必须提到.NET 的这三个分支，因为在未来几年里，仍然可能会看到它们、读到它们并面对它们。重要的是，要知道它们是什么以及它们的目的，以便可以在项目中实现它们。到 2020 年，.NET 有了一个新版本，直接简称为“.NET”。.NET 是完全开源的、完全跨平台的，可以在许多平台上使用，而不必支持程序的多个版本和分支。

1.1.2 使用.NET 编写程序

用.NET 创建计算机程序意味着使用.NET 库中的现有代码来编程。本书使用 Visual Studio 来开发程序。Visual Studio 是一个强大的集成开发环境，支持 C#(以及 C++、Visual Basic 和 F#等)。这种环境的优势在于，可轻松地将.NET 特性集成到代码中。所创建的代码将完全是 C#的；但要全面使用.NET，必要时还可在 Visual Studio 中使用一些额外的工具。要执行 C#代码，必须将其转换为目标

操作系统能够理解的语言，即本地代码。这种转换称为编译，是由编译器执行的动作，分为两个阶段。

1. CIL 和 JIT

在编译使用.NET 库的代码时，不是立即创建专用于操作系统的本机代码，而是把代码编译为通用中间语言(Common Intermediate Language，CIL)代码，这些代码并非专门用于任何一种操作系统，也非专门用于 C#。其他.NET 语言(如 Visual Basic .NET 或 F#)也会在第一阶段编译为这种语言。开发 C#应用程序时，这个编译步骤由 Visual Studio 完成。

显然，要执行应用程序，必须完成更多工作，这是 Just-In-Time(JIT)编译器的任务，它把 CIL 编译为专用于 OS 和目标机器架构的本机代码。这样 OS 才能执行应用程序。这里编译器的名称 Just-In-Time 反映了 CIL 代码仅在需要时才编译的事实。这种编译可以在应用程序的运行过程中动态发生，不过开发人员一般不需要关心这个过程。除非要编写性能十分关键的高级代码，否则知道这个编译过程会在后台自动进行，并不需要人工干预就可以了。

过去，经常需要把代码编译为几个应用程序，每个应用程序都用于特定的操作系统和 CPU 架构。这通常是一种优化形式(例如，为了让代码在 AMD 芯片组上运行得更快)，但有时则是非常重要的(例如，使应用程序可以同时工作在 Win9*x* 和 WinNT/2000 环境下)。现在就没必要了，因为 JIT 编译器使用 CIL 代码，而 CIL 代码是独立于计算机、操作系统和 CPU 的。目前有几种 JIT 编译器，每种编译器都用于不同的架构，CLR 会使用合适的编译器创建所需的本机代码。

这样，开发人员需要做的工作就比较少了。实际上，可以忽略与系统相关的细节，将注意力集中在代码的功能上就够了。

注意：

读者可能遇到过 Microsoft Intermediate Language(MSIL)这一术语，它是 CIL 原来的名称，许多开发人员仍沿用这个术语。可以访问 https://en.wikipedia.org/wiki/Common_Intermediate_Language 获取 CIL 的更多信息。

2. 程序集

编译应用程序时，所创建的 CIL 代码存储在一个程序集(assembly)中。程序集包括可执行的应用程序文件(这些文件可以直接在 Windows 上运行，不需要其他程序，其扩展名是.exe)和其他应用程序使用的库(其扩展名是.dll)。

除包含 CIL 外，程序集还包含元信息(即程序集中包含的数据的信息，也称为元数据)和一些可选的资源(CIL 使用的其他数据，例如，声音文件和图片)。元信息允许程序集是完全自描述的。不需要其他信息就可以使用程序集，也就是说，我们不会遇到没有把需要的数据添加到系统注册表中这样的问题，而在使用其他平台进行开发时这个问题常常出现。

因此，部署应用程序就非常简单了，只需要把文件复制到远程计算机上的目录下即可。因为不需要目标系统上的其他信息，所以对于针对.NET 的应用程序，只需要从该目录中运行可执行文件即可(假定安装了.NET CLR)。根据部署场景，运行该程序需要的所有模块都包含在部署包中，不需要进行其他配置。

在.NET 中，不必将运行应用程序需要的所有信息都安装到一个地方。可以编写一些代码来执行多个应用程序所要求的任务。此时，通常把这些可重用的代码放在所有应用程序都可以访问的地方。在.NET 中，这个地方是全局程序集缓存(Global Assembly Cache，GAC)，把代码放在这个缓存中十分简单，只需要把包含代码的程序集放在包含该缓存的目录中即可。

3. 托管代码

在将代码编译为 CIL，再用 JIT 编译器将它编译为本机代码后，CLR 的任务尚未全部完成，还需要管理正在执行的用.NET 编写的代码(这个执行代码的阶段通常称为运行时或运行库(runtime))。即 CLR 管理着应用程序，其方式是管理内存、处理安全性以及允许进行跨语言调试等。相反，不受 CLR 控制运行的应用程序属于非托管类型，某些语言(如 C++)可以用于编写此类应用程序，例如，访问操作系统的底层功能的应用程序。但是在 C#中，只能编写在托管环境下运行的代码。我们将使用 CLR 的托管功能，让.NET 处理与操作系统的任何交互。

4. 垃圾回收

托管代码最重要的一个功能是垃圾回收(garbage collection)。这种.NET 方法可确保应用程序不再使用某些内存时，就会完全释放这些内存。在.NET 推出以前，这项工作主要由程序员负责，代码中的几个简单错误会把大块内存分配到错误的地方，使这些内存神秘失踪。这通常意味着计算机的速度逐渐减慢，最终导致系统崩溃。

.NET 垃圾回收会定期检查计算机的内存，从中删除不再需要的内容。执行垃圾回收的时间并不固定，可能一秒钟内会进行数千次的检查，也可能每几秒钟才检查一次，不过一定会进行检查。

这里要给程序员一些提示。因为是在不可预知的时间执行这项工作，所以在设计应用程序时，必须留意这一点。需要许多内存才能运行的代码应自行完成清理工作，而不是坐等垃圾回收，但这不像听起来那样难。

5. 把它们组合在一起

在继续学习之前，先总结一下上述所讨论的创建.NET 应用程序所需的步骤：

(1) 使用某种.NET 兼容语言(如 C#)编写应用程序代码，如图 1-1 所示。

(2) 把代码编译为 CIL，存储在程序集中，如图 1-2 所示。

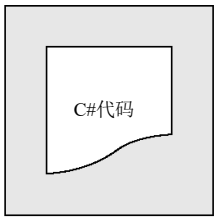

图 1-1

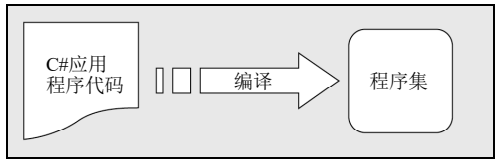

图 1-2

(3) 在执行代码时(如果这是一个可执行文件，就自动运行，或者在其他代码使用它时运行)，首先必须使用 JIT 编译器将代码编译为本机代码，如图 1-3 所示。

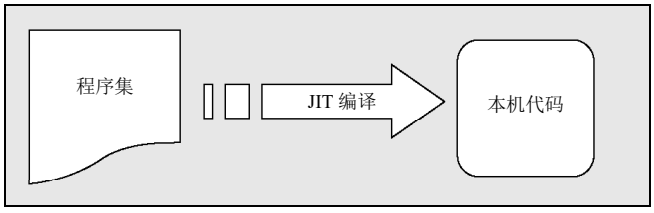

图 1-3

(4) 在托管的 CLR 环境下运行本机代码，以及其他应用程序或进程，如图 1-4 所示。

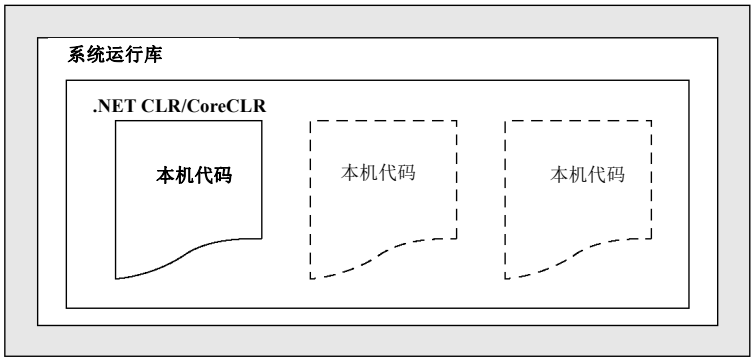

图　1-4

6. 链接

在上述过程中还有一点要注意。在第(2)步中编译为 CIL 的 C#代码未必包含在一个单独文件中，可以把应用程序代码放在多个源代码文件中，再把它们编译到一个单独的程序集中。这个过程称为链接(linking)，是非常有用的。原因是处理几个较小的文件比处理一个大文件要简单得多。可以把逻辑上相关的代码分解到一个文件中，以便单独进行处理，这也更便于在需要时找到特定的代码块，让开发小组把编程工作分解为一些可管理的块，让每个人编写一小块代码，而不会破坏已编写好的代码部分或其他人正在处理的部分。

1.2　C#的含义

如上所述，C#是可用于创建要运行在.NET CLR 上的应用程序的语言之一。它从 C 和 C++语言演化而来，是 Microsoft 专门为使用.NET 平台而创建的。C#吸取了以往语言失败的教训，融合了其他语言的许多优点，并解决了它们存在的问题。

使用 C#开发应用程序比使用 C++简单，因为其语法更简单。但 C#是一种强大的语言，在 C++中能完成的任务几乎都能利用 C#完成。虽然如此，C#中与 C++高级功能等价的功能(例如直接访问和处理系统内存)，只能在标记为 unsafe 的代码中使用。顾名思义，这种高级编程技术存在潜在威胁，因为它可能覆盖系统中重要的内存块，导致严重后果。因此，本书不讨论这个问题。

C#代码通常比 C++代码略长一些。这是因为 C#是一种类型安全的语言(与 C++不同)。在外行人看来，这表示一旦为某个数据指定了类型，就不能转换为另一种不相关的类型。所以，在类型之间转换时，必须遵守严格的规则。执行相同的任务时，用 C#编写的代码通常比用 C++编写的代码长。但 C#代码更健壮，调试起来也比较简单，.NET 始终可以随时跟踪数据的类型。在 C#中，不能完成诸如“把 4 字节的内存分配给这个数据后，我们使其有 10 字节长，并把它解释为 X”等任务，但这并不是一件坏事。

C#只是用于.NET 开发的一种语言，但它是最好的一种语言。C#的优点是，它是唯一彻头彻尾为.NET 设计的语言，是在移植到其他操作系统上的.NET 版本中使用的主要语言。要使诸如 Visual Basic .NET 的语言尽可能类似于其以前的语言，且仍遵循 CLR，就不能完全支持.NET 代码库的某些功能，至少需要不常见的语法。

C#能使用.NET 代码库提供的每种功能，但并非所有的功能都已移植到.NET 版本中。而且，.NET 的每个新版本都在 C#语言中添加了新功能，满足了开发人员的要求，使之更强大。

1.2.1 用 C#能编写什么样的应用程序

如前所述，.NET 没有限制应用程序的类型。C#使用的是.NET Framework，所以也没有限制应用程序的类型。这里仅讨论几种常见的应用程序类型。

- **桌面应用程序** 这些应用程序(如 Microsoft Office)具有我们很熟悉的 Windows 外观和操作方式，使用.NET 的 Windows Presentation Foundation(WPF)模块就可以简便地生成这种应用程序。WPF 模块是一个控件库，其中的控件(例如按钮、工具栏和菜单等)可用于建立 Windows 用户界面(UI)。
- **云/Web 应用程序** .NET 包括一个动态生成 Web 内容的强大系统——ASP.NET Core，允许进行个性化和实现安全性等。另外，这些应用程序可以在云中驻留和访问，例如 Microsoft Azure 平台。
- **移动应用程序** 使用 C#和 Xamarin 移动 UI 框架，可以创建面向 Android 操作系统的移动应用程序。
- **Web API** 这是建立 REST 风格的 HTTP 服务的理想框架，支持许多客户端，包括移动设备和浏览器。它们也称为 REST API。
- **WCF 服务** 这是一种灵活创建各种分布式应用程序的方式。使用 WCF 服务可以通过局域网或 Internet 交换几乎各种数据。无论使用什么语言创建 WCF 服务，也无论 WCF 服务驻留在什么系统上，都使用一样简单的语法。这是一项较老的技术，需要使用较老版本的.NET Framework 来创建。

这些类型的应用程序也可能需要某种形式的数据库访问，这可以通过.NET 的 Active Data Objects .NET (ADO.NET)部分、Entity Framework 或 C#的 LINQ(Language Integrated Query)功能来实现。对于需要数据库访问的.NET Core 应用程序，将使用 Entity Framework Core 库。也可以使用许多其他资源，例如，创建联网组件、输出图形、执行复杂数学任务的工具来实现。

1.2.2 本书中的 C#

本书第 I 部分介绍 C#语言的语法和用法，但不过分强调.NET。这是必需的，因为我们不能没有一点儿 C#编程基础就使用.NET。首先介绍一些比较简单的内容，把较复杂的面向对象编程(Object-Oriented Programming，OOP)主题放在基础知识的后面论述。假定读者没有一点儿编程的知识，这些是首要原则。

学习了基础知识后，本书还将介绍如何开发更复杂、更有用的应用程序。本书第 II 部分将介绍数据访问(ORM 数据库概念、文件系统和 XML 数据)和 LINQ，第 III 部分将详细讨论其他技术，例如 REST API、云和 Windows 桌面。

1.3 Visual Studio

本书使用 Visual Studio 开发工具的最近版本进行所有的 C#编程，包括简单的命令行应用程序，乃至较复杂的项目类型。Visual Studio 不是开发 C#应用程序必需的开发工具或集成开发环境(IDE)，但使用它可以使任务更简单一些。如果愿意的话，可在基本的文本编辑器(如常见的记事本应用程序)中处理 C#源代码文件，再使用.NET 中包含的命令行编译器把代码编译到程序集中。但是，为什么不

使用功能完备的 IDE 呢？

1.3.1　Visual Studio 产品

Microsoft 提供了如下几个 Visual Studio 版本：

- Visual Studio Community
- Visual Studio Professional
- Visual Studio Enterprise
- Visual Studio Code
- Visual Studio for Mac

其中，Visual Studio Code、Mac 和 Community 版本可从 visualstudio.microsoft.com/downloads 获得。但 Professional 和 Enterprise 版本提供了一些额外的功能，但需要购买。

各种 Visual Studio 产品可以创建所需的几乎所有 C#应用程序。Visual Studio Code 是一个简单但健壮的代码编辑器，它运行在 Windows、Linux 和 iOS 操作系统上。与 Visual Studio Code 不同，Visual Studio Community 在外观和操作方式上类似于 Visual Studio Professional 和 Enterprise。虽然 Microsoft 在 Visual Studio Community 中提供了许多与 Professional 和 Enterprise 版本相同的功能，但还是缺少一些重要功能，比如深度调试功能和代码优化工具。但是缺少的特性并不影响使用 Community 版本来学习本书的各个章节。本书示例使用的 IDE 版本就是 Visual Studio Community。

1.3.2　解决方案

在使用 Visual Studio 开发应用程序时，可以通过创建解决方案来完成。在 Visual Studio 术语中，解决方案不仅是一个应用程序，它还包含项目，可以是控制台应用程序、WPF 项目、云/Web 应用程序项目和 ASP.NET Core 项目等。但是，解决方案可以包含多个项目，这样，即使相关的代码最终在硬盘上的多个位置被编译为多个程序集，也可以把它们组合到一处。

这是非常有用的，因为它可以处理“共享”代码(这些代码放在 GAC 中)，同时，应用程序也使用这段共享代码。在使用唯一的开发环境时，调试代码是非常容易的，因为可在多个代码模块中单步调试指令。

1.4　本章要点

主题	要点
.NET 基础	.NET Framework 是 Microsoft 的代码开发平台。它包括一个公共类型系统(CTS)和一个公共语言运行库(CLR/CoreCLR)。.NET Framework 应用程序使用面向对象编程(OOP)的方法论编写，通常包含托管代码。托管代码的内存管理由.NET 运行库处理，其中包括垃圾回收
.NET 应用程序	用.NET 编写的应用程序首先编译为 CIL。在执行应用程序时，JIT 把 CIL 编译为本机代码。应用程序编译后，把不同的部分链接到包含 CIL 的程序集中
.NET Core 应用程序	.NET Core 应用程序的工作方式与.NET Framework 应用程序类似，但不使用 CLR，而使用 CoreCLR。.NET Core 是原始.NET Framework 的一个分支，可以跨平台运行
.NET Standard	.NET Standard 提供了一个统一的类库，多个.NET 平台(如.NET Framework、.NET Core 和 Xamarin)都可将它作为目标

(续表)

主题	要点
C#基础	C#是包含在.NET 中的一种语言，它是已有语言(如 C++)的一种演变，可用于编写任意应用程序，包括 Web 应用程序、跨平台应用程序和桌面应用程序
集成开发环境(IDE)	可在 Visual Studio 2017 中用 C#编写任意类型的.NET 应用程序，还可以在免费的但功能稍弱的 Community 产品中用 C#创建.NET 应用程序。IDE 使用解决方案，解决方案可以包含多个项目