

高等学校通识教育系列教材

GENERAL
EDUCATION

C#程序设计 经典教程（第三版）

罗福强 杨剑 张敏辉 编著



清华大学出版社

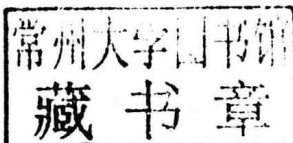


高等学校通识教育系列教材

GENERAL
EDUCATION

C#程序设计 经典教程（第三版）

罗福强 杨剑 张敏辉 编著



清华大学出版社
北京

内 容 简 介

C# 经过近 20 年的不断发展和完善,已经成为一种跨平台的开发语言,如今它不仅能用于开发 Windows 系统中的应用程序,还可以用来开发 Android、iOS、Windows Phone 和 Mac App 应用程序,甚至还能开发物联网嵌入式系统。

全书共 14 章,在介绍 C# 语法的基础之上,深入阐述面向对象的程序设计方法、基于事件驱动的程序设计方法以及面向服务的程序设计方法。不仅如此,本书还全面揭示 C# 的各种应用技术,包括 Windows 程序设计技术、数据库编程技术、文件操作与编程技术、XML 与 LINQ 高级数据访问技术、面向服务编程技术和多媒体处理编程技术等。本书内容丰富,可操作性强,叙述简洁流畅,语言通俗易懂,所有实例都经过精心设计,能够使学生轻松地掌握 C# 的基本语法、编程方法和应用技巧。

本书可作为高等院校计算机相关专业的教材,也可作为初、中级读者和培训班学员的参考用书。

本书封面贴有清华大学出版社防伪标签,无标签者不得销售。

版权所有,侵权必究。侵权举报电话:010-62782989 13701121933

图书在版编目(CIP)数据

C# 程序设计经典教程/罗福强等编著. —3 版. —北京:清华大学出版社,2018

(高等学校通识教育系列教材)

ISBN 978-7-302-49807-0

I. ①C… II. ①罗… III. ①C 语言—程序设计—高等学校—教材 IV. ①TP312.8

中国版本图书馆 CIP 数据核字(2018)第 037258 号

责任编辑:刘向威

封面设计:文 静

责任校对:李建庄

责任印制:丛怀宇

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:010-62770175

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

课件下载: <http://www.tup.com.cn>, 010-62795954

印 装 者:三河市君旺印务有限公司

经 销:全国新华书店

开 本:185mm×260mm

印 张:25.5

字 数:621 千字

版 次:2012 年 2 月第 1 版 2018 年 7 月第 3 版

印 次:2018 年 7 月第 1 次印刷

印 数:1~1500

定 价:65.00 元

产品编号:077519-01

前言

C#是由微软公司推出的完全面向对象的计算机高级语言。经过近20年的发展,如今它不仅能用于开发传统 Windows 环境中的应用程序,还可以用来开发原生的 Android、iOS、Windows Phone 和 Mac App 应用程序,甚至还能整合 Azure 或 Hadoop 技术开发云计算和大数据应用系统。相对于 C++ 来说,C# 更容易被理解和接受;相对于 Java 来说,C# 更好用,开发软件的效率更高。

本书自 2012 年 2 月出版第 1 版以来,受到广大师生的欢迎。2014 年我们组织修订,推出第 2 版。如今,3 年过去了,微软公司已经推出多个 C# 新版本,使 C# 具有大量新特性。为此,我们再次组织编写本教材第 3 版,针对第 2 版主要进行以下修订。

(1) 在第 1 章中增加 .NET 技术体系结构的介绍,使读者对 .NET 技术有更全面的了解。为了便于读者更早和更快地理解 C# 程序,把 C# 程序的特点独立编成 1 节。

(2) 如今海量的文本日志成为构建大数据技术的主要研究内容,特征提取与转换、数据分析与挖掘成为程序设计的重点,为此,第 2 章加强了字符串的内容,包括文本格式化处理的内容等。

(3) 自 C# 3.0 开始,C# 添加很多新特性,例如,引入表达式主体(expression-bodied)来简化对象属性和索引器的定义,引入 Lambda 表达式简化匿名函数的定义,不仅降低了 C# 程序的复杂度,还使 C# 源代码更加优雅。

(4) 云计算和大数据技术的基础是面向服务的程序设计思想。要想快速适应云计算和大数据时代的新要求,必须更早地了解或熟悉这种新思想。为此本书第 13 章剔除原来的一部分内容,增加面向服务的编程技术。

本书第 3 版以 Visual Studio .NET 2017 和 C# 7.0 为蓝本,深入介绍 C# 语言及其应用。全书共分 14 章,基本上覆盖了 C# 的主要领域,在讲解 C# 语法的基础上,不仅阐述面向对象、基于事件驱动和面向服务的 3 种不同的程序设计思想,还全面展现 C# 的具体应用技术,包括 Windows 程序设计、数据库应用编程、文件操作与编程、XML 与 LINQ 高级数据访问、面向服务编程和多媒体处理编程技术等。

本书继续保持以下优点:第一,面向应用型本科院校学生,立足于把 C# 的语法讲透彻、讲清楚,文字叙述尽量简练;第二,重点围绕面向对象程序设计思想和可视化的 Windows 程序设计方法展开教学内容;第三,书中所有案例均精心设计,不仅代码完整,还贴近学生实际生活;第四,坚持零起点原则,学生可以在没有 C/C++ 基础的情况下使用本书;第五,坚持应用为纲,全面展示 C# 在各应用领域的编程技巧。

本书可作为高等院校 Visual C# .NET 课程的教材或参考资料,也可供软件开发人员参考使用。

本书由四川大学锦城学院的罗福强老师主持修订。参与本书编写的还有杨剑、张敏辉、熊永福、陈虹君、李瑶、赵力衡等老师。本书长期以来获得清华大学出版社的各级领导的重视和支持,也获得了作者所在单位领导的大力支持。在此,我们对支持本书编写出版并提供过大量帮助的所有人员表示诚挚的感谢!

由于时间仓促,书中难免有不妥之处,我们殷切地期望读者提出宝贵的意见。

编 者

2018 年 4 月

目 录

第 1 章 C# 概述	1
1.1 .NET 与 C# 概述	1
1.1.1 .NET 概述	1
1.1.2 C# 语言的发展	3
1.1.3 C# 语言的特点	4
1.2 我的第一个 C# 程序	5
1.2.1 我的第一个控制台应用程序	5
1.2.2 我的第一个 Windows 应用程序	9
1.2.3 一个具有输入功能的 Win32 应用程序	11
1.2.4 我的第一个 Web 应用程序	14
1.3 C# 项目结构与程序特点	16
1.3.1 C# 项目结构	16
1.3.2 C# 程序的特点	17
习题	18
上机实验 1	19
第 2 章 C# 程序设计基础	20
2.1 常量与变量	20
2.1.1 常量	21
2.1.2 变量	22
2.2 C# 的数据类型	24
2.2.1 简单类型	24
2.2.2 枚举型	25
2.2.3 结构型	27
2.2.4 数据类型转换	28
2.3 运算符与表达式	30
2.3.1 算术运算符与表达式	30
2.3.2 赋值运算符与表达式	32
2.3.3 关系运算符与表达式	33
2.3.4 逻辑运算符与表达式	33

2.4	数组和字符串	35
2.4.1	一维数组	35
2.4.2	多维数组	38
2.4.3	数组型的数组	39
2.4.4	字符串	41
	习题	44
	上机实验 2	45
第 3 章	C# 程序的流程控制	47
3.1	C# 程序的分支语句	47
3.1.1	if 语句	48
3.1.2	多分支 if...else if 语句	49
3.1.3	switch 语句	51
3.1.4	分支语句的嵌套	54
3.2	C# 程序的循环语句	56
3.2.1	while 语句	56
3.2.2	do...while 语句	58
3.2.3	for 语句	60
3.2.4	foreach 语句	61
3.2.5	循环语句的嵌套	63
3.3	跳转语句	65
3.3.1	break 语句	65
3.3.2	continue 语句	66
	习题	68
	上机实验 3	69
第 4 章	面向对象程序设计入门	71
4.1	面向对象的基本概念	71
4.1.1	对象	72
4.1.2	事件与方法	72
4.1.3	类与对象	73
4.1.4	抽象、封装、继承与多态	73
4.2	类的定义与使用	75
4.2.1	类的声明和实例化	75
4.2.2	类的可访问性	78
4.2.3	值类型与引用类型	79
4.3	类的成员及其定义	81
4.3.1	常量与字段	81
4.3.2	属性	82

4.3.3	方法	85
4.3.4	构造函数	87
4.4	方法的参数传递	90
4.4.1	按值传参	90
4.4.2	按引用传参	92
4.4.3	输出参数	93
4.4.4	引用类型的参数	94
4.4.5	数组型参数	96
4.5	方法的重载	97
4.5.1	方法的重载	97
4.5.2	构造函数的重载	99
4.6	对象的生命周期	102
4.6.1	对象的生命周期	102
4.6.2	终结器	102
	习题	103
	上机实验 4	104
第 5 章	面向对象的高级程序设计	107
5.1	静态成员与静态类	107
5.1.1	类的静态成员	107
5.1.2	静态构造函数	110
5.1.3	静态类	110
5.2	类的继承性	112
5.2.1	派生类的声明	112
5.2.2	构造函数	113
5.2.3	密封类	117
5.3	类的多态性	117
5.3.1	使用 new 重新定义类的成员	117
5.3.2	用 virtual 和 override 定义类的成员	118
5.3.3	访问基类的成员	120
5.4	抽象类	123
5.4.1	抽象类及其抽象成员	123
5.4.2	重载抽象方法	125
5.5	接口	127
5.5.1	接口的声明	127
5.5.2	接口的实现	128
5.5.3	接口的继承性	128
5.5.4	多重接口实现	129
5.5.5	访问接口的成员	130

5.5.6 抽象类与接口的比较	135
5.6 嵌套类、分部类与命名空间	135
5.6.1 嵌套类	135
5.6.2 分部类	137
5.6.3 命名空间	138
习题	140
上机实验 5	141
第 6 章 集合、索引器与泛型	146
6.1 集合	146
6.1.1 集合概述	146
6.1.2 ArrayList	147
6.1.3 哈希表 Hashtable	152
6.1.4 栈和队列	154
6.2 索引器	155
6.2.1 索引器的定义	155
6.2.2 索引器的使用	157
6.2.3 索引器的重载	157
6.2.4 接口中的索引器	159
6.2.5 索引器与属性的比较	160
6.3 泛型	160
6.3.1 泛型概述	160
6.3.2 泛型集合	161
6.3.3 自定义泛型	163
6.3.4 泛型的高级应用	167
习题	170
上机实验 6	171
第 7 章 程序调试与异常处理	172
7.1 程序错误	172
7.1.1 程序错误分类	172
7.1.2 调试程序错误	174
7.2 程序的异常处理	177
7.2.1 异常的概念	177
7.2.2 异常处理	178
7.2.3 try...catch 语句	180
7.2.4 finally 语句	181
7.2.5 throw 语句与抛出异常	182
习题	183

上机实验 7	184
第 8 章 基于事件驱动的程序设计技术	187
8.1 基于事件的编程思想	187
8.2 委托	189
8.2.1 委托的概述	189
8.2.2 委托的声明、实例化与使用	189
8.2.3 委托与匿名函数	191
8.2.4 多路广播与委托的组合	192
8.3 事件	194
8.3.1 事件的声明	194
8.3.2 订阅事件	196
8.3.3 触发事件	196
8.4 基于事件的 Windows 编程	200
8.4.1 Windows 应用程序概述	200
8.4.2 Windows 窗体与事件驱动编程	202
习题	207
上机实验 8	207
第 9 章 Windows 程序的界面设计	213
9.1 窗体与控件概述	214
9.1.1 Windows 窗体	214
9.1.2 窗体的控件	215
9.2 按钮与文本显示、编辑控件	217
9.2.1 按钮控件	217
9.2.2 文本显示控件	218
9.2.3 文本编辑控件	218
9.2.4 应用实例——用户登录	220
9.3 列表与选择控件	222
9.3.1 RadioButton 控件	222
9.3.2 CheckBox 控件	223
9.3.3 ListBox 控件	223
9.3.4 ComboBox 控件	224
9.3.5 其他常用控件	225
9.3.6 应用实例——添加个人收支明细	226
9.4 图形显示控件	229
9.4.1 PictureBox 控件	229
9.4.2 ImageList 控件	229
9.4.3 应用实例——关于我们	230

9.5	容器控件	230
9.5.1	GroupBox 控件	231
9.5.2	Panel 控件	231
9.5.3	TabControl 控件	231
9.5.4	应用实例——添加收支项目	232
9.6	对话框	235
9.6.1	对话框概述	235
9.6.2	消息框	237
9.6.3	通用对话框	239
9.6.4	应用实例——简单的文本编辑器	241
9.7	菜单、工具栏和状态栏	242
9.7.1	菜单	242
9.7.2	工具栏	244
9.7.3	状态栏	245
9.7.4	应用实例——个人理财系统的主窗口设计	246
9.8	SDI 和 MDI 应用程序	248
9.8.1	创建 SDI 应用程序	248
9.8.2	创建 MDI 应用程序	248
9.8.3	应用实例——个人理财的 MDI 设计	248
	习题	249
	上机实验 9	250
第 10 章	C# 数据库编程技术	255
10.1	数据库与 ADO.NET 概述	255
10.1.1	数据库概述	255
10.1.2	SQL 概述	258
10.1.3	ADO.NET 概述	259
10.1.4	ADO.NET 访问数据库的一般步骤	260
10.2	Connection 与 Command 对象的使用	261
10.2.1	Connection 对象	261
10.2.2	Command 对象	262
10.2.3	应用实例——实现用户登录	263
10.2.4	应用实例——实现收支类别的添加	264
10.3	DataReader 对象的使用	266
10.3.1	DataReader 对象	266
10.3.2	应用实例——实现收支项目的添加	267
10.3.3	应用实例——实现收支明细的添加	270
10.4	DataSet 与 DataAdapter 对象的使用	273
10.4.1	DataSet 与 DataAdapter 对象	273

10.4.2	DataGridView 控件	275
10.4.3	应用实例——实现收支明细的查询	276
习题		278
上机实验 10		279
第 11 章	文件操作与编程技术	281
11.1	文件的输入/输出	281
11.1.1	文件 I/O 与流	281
11.1.2	读写文本文件	282
11.1.3	读写二进制文件	284
11.1.4	对象的序列化	286
11.2	文件操作控件	290
11.2.1	SaveFileDialog 控件	290
11.2.2	OpenFileDialog 控件	291
11.2.3	FolderBrowserDialog 控件	293
11.2.4	应用实例——简易的写字板程序	294
习题		297
上机实验 11		297
第 12 章	高级数据访问与处理技术	302
12.1	XML 编程	302
12.1.1	XML 概述	302
12.1.2	XML 文档的创建	304
12.1.3	XML 文档的查询	310
12.1.4	XML 文档的编辑	313
12.2	LINQ 编程	316
12.2.1	LINQ 概述	316
12.2.2	LINQ 的查询子句	319
12.2.3	LINQ to XML 的应用	322
12.2.4	LINQ to SQL 的应用	325
习题		331
上机实验 12		332
第 13 章	面向服务编程技术	334
13.1	面向服务编程基础	334
13.1.1	计算机网络的概述	334
13.1.2	计算机网络的通信协议	335
13.1.3	面向服务编程概述	336
13.2	.NET 网络编程基础	338

13.2.1	System.Net 概述	338
13.2.2	Socket 编程概述	340
13.2.3	TCP 应用编程	343
13.2.4	UDP 应用编程	348
13.3	基于 Web API 的面向服务编程	351
13.3.1	ASP.NET Web API 概述	351
13.3.2	Web API 服务器端编程	352
13.3.3	HttpClient 客户端编程	359
习题		363
上机实验 13		364
第 14 章	多媒体编程技术	366
14.1	GDI+ 绘图	366
14.1.1	GDI+ 概述	366
14.1.2	System.Drawing 命名空间	367
14.1.3	创建 Graphics 对象	368
14.1.4	颜料、钢笔和画笔	369
14.1.5	点、线和图形	371
14.1.6	图像和文本	377
14.1.7	坐标系统及变换	380
14.2	Windows Media Player 组件的使用	386
14.2.1	Windows Media Player 组件的介绍	386
14.2.2	Windows Media Player 组件的使用	388
习题		391
上机实验 14		392
参考文献		393

总体要求

- 了解 C# 语言的特点及其发展。
- 了解 C# 应用程序的结构及其特点。

相关知识点

- 了解计算机软件、计算机语言及分类的知识。
- 熟悉 Windows 系统基础知识及操作。

学习重点

- C# 程序的结构与特点。

学习难点

- 控制台应用程序与 Win32 应用程序的区别。

1.1 .NET 与 C# 概述

1.1.1 .NET 概述

1. .NET 技术体系结构

.NET 平台是微软公司在 20 世纪末为了迎接互联网的挑战而推出的 Windows 应用程序运行平台。经过近 20 年的发展,它如今已经成为一个可以跨越任何硬件系统的开发平台,在这个平台上可以构建和运行 Windows 应用程序、Web 应用程序、Azure 云应用程序、移动 App 应用程序、Unity 游戏等。.NET 建立在开放体系结构基础之上,集 Microsoft 在软件领域的主要技术成就于一身,如图 1-1 所示。

.NET 技术的核心是 .NET Framework,它为 .NET 平台下应用程序的运行提供基本框架,如果把 Windows 操作系统比作一幢摩天大楼的地基,那么 .NET Framework 就是摩天大楼中由钢筋和混凝土搭成的框架。为了实现跨平台运行的目标,微软公司新推出了 .NET Core,其核心 .NET Core Framework 是参考 .NET Framework 重新开发的,它支持 Windows,Mac OS,Linux 等操作系统,可以用于嵌入式或物联网解决方案之中。为了使 .NET 应用程序能在智能终端设备之上运行,微软启动了 Mono 项目,该项目可以看作是 .NET Framework 的开源实现。

.NET Framework 以微软的 Windows 操作系统为基础,由不同的组件组成(如图 1-1 所示),能够与 Windows 的各种应用程序服务组件(如消息队列服务、COM+ 组件服务、Internet 信息服务(IIS)、Windows 管理工具等)整合,来开发各种应用程序。

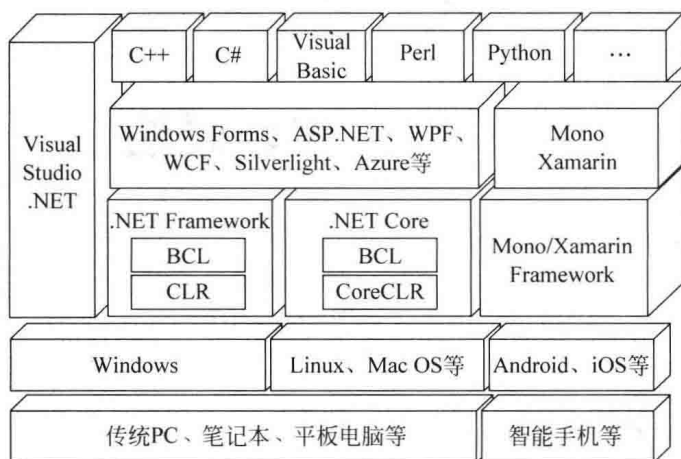


图 1-1 .NET 平台的体系结构

在 .NET Framework 的最顶层是程序设计语言，.NET Framework 支持诸如 VB、C#、C++、F#、Perl、Python 等几十种高级程序设计语言。在 Visual Studio .NET 开发环境中，可直接使用 VB、C#、C++、F#、TypeScript、Python 等多种语言开发应用程序；利用新推出的移动应用跨平台开发插件 Xamarin^①，用户还可以直接开发 iOS、Android、Windows Phone 和 Mac App 等应用，而不需要转移到 Eclipse 或者额外购买 Mac 和使用 Xcode。

.NET Framework 具有两个主要组件：公共语言运行时(Common Language Runtime, CLR)和基础类库(Base Class Lib, BCL)，除此之外还包括 ADO.NET、ASP.NET、WCF、Azure、Workflow 框架等。

CLR 是 .NET Framework 的基础，是应用程序与操作系统之间的“中间人”，它为应用程序提供内存管理、线程管理和远程处理等核心服务。在 .NET 平台上，应用程序无论使用何种语言编写，在编译时都会被语言编译器编译成 MSIL(微软中间语言代码)，在运行应用程序时 CLR 自动启用 JIT(Just in Time)编译器将 MSIL 再次编译成操作系统能够识别的本地机器语言代码(简称本地代码)，然后运行并返回运行结果。因此，CLR 是所有 .NET 应用程序的托管环境。这种运行在 .NET 之上的应用程序被称为托管应用程序，而传统的直接在操作系统基础之上运行的应用程序则被称为非托管应用程序。

BCL 类库是一个综合性的面向对象的可重用类型集合，包括集合类、文件系统处理类、XML 处理类、网络通信接口类、异步 Task 类等，利用它可以开发多种应用程序，包括传统的命令行、图形用户界面(GUI)应用程序、Web 应用程序等。

ADO.NET 是 .NET Framework 提供的微软新一代的面向对象数据处理技术，利用它可以简便、快捷地开发数据库应用程序。

ASP.NET 是 .NET Framework 提供的全新的 Web 应用程序开发技术，利用它开发 Web 应用程序如同开发 Windows 应用程序一样简单。

WCF(Windows Communication Foundation)、WPF(Windows Presentation Foundation)以及 Silverlight 等技术是微软推出的全新 .NET 技术。WCF 可以理解 Windows 通信接口，它整

^① Xamarin 始创于 2011 年，2016 年 2 月被微软公司收购。如今，Xamarin 已经被微软内置到 Visual Studio .NET 2017 之中。此外，微软还开源了 Xamarin SDK，免费供用户使用。

合了 TCP/IP、XML、SOAP、JSON 等技术,因此简化了 XML Web 服务的设计与实现。WPF 为用户界面、2D/3D 图形、文档和媒体提供了统一的描述和操作方法。Silverlight 为开发具有专业图形、音频和视频处理的 Web 应用程序提供了全新的解决方案。

2. .NET Framework 的优点

.NET Framework 的目标是为应用程序开发人员提供了一个与平台无关的开发环境,具有以下优点。

(1) 基于 Web 的标准

.NET Framework 完全支持现有的 Internet 技术,包括 HTML(超文本标记语言)、HTTP(超文本传输协议)、XML(可扩展标记语言)、SOAP(简单对象访问协议)、XSLT(可扩展样式表语言转换)、XPath(XML 路径语言)、JSON(JavaScript 对象表示方法)和其他 Web 标准。

(2) 使用统一的应用程序模型

任何与 .NET 兼容的语言都可以使用 .NET Framework 类库。.NET Framework 为 Windows 应用程序、Web 应用程序、云计算服务、跨平台的智能手机应用提供了统一的应用程序模型,因此同一段代码可被这些应用程序无障碍地使用。

(3) 便于开发人员使用

在 .NET Framework 中,代码被组织在不同的命名空间和类中,而命名空间采用树形结构,以便开发人员引用。当开发人员调用 .NET Framework 类库的类时,只需将该类属性命名空间添加到引用解决方案中即可。

(4) 可扩展类

.NET Framework 提供了通用类型系统,它根据面向对象的思想把一个命名空间或类中代码的实现细节隐藏,开发人员可以通过继承来访问类库中的类,也可以扩展类库中的类,甚至构建自己的类库。

1.1.2 C# 语言的发展

在过去的 30 年里,C 和 C++ 已经成为在商业软件的开发领域中使用最广泛的语言。它们为程序员提供了十分灵活的操作,不过同时也牺牲了一定的效率。与 Visual Basic 等语言相比,同等级别的 C/C++ 应用程序往往需要更长时间来开发。由于 C/C++ 语言的复杂性,许多程序员都试图寻找一种新的语言,希望能在功能与效率之间找到一个更为理想的权衡点。

目前有些语言,以牺牲灵活性的代价来提高效率。可是这些灵活性正是 C/C++ 程序员所需要的。这些解决方案对编程人员的限制过多(如屏蔽一些底层代码控制的机制),其所提供的功能难以令人满意。这些语言无法方便地同原来的系统交互,也无法与当前的网络编程很好地结合。

对于 C/C++ 用户来说,最理想的解决方案无疑是在快速开发的同时又可以调用底层平台的所有功能。他们想要一种和最新的网络标准保持同步并且能和已有的应用程序良好整合的环境。另外,一些 C/C++ 开发人员还需要在必要的时候进行一些底层的编程。

C#(读作 C Sharp)是微软对这一问题的解决方案。C# 是一种最新的、面向对象的编程语言。它是一种简单但功能强大的编程语言,使程序员可以快速地编写各种基于

Microsoft .NET 平台的应用程序。

它从 C 和 C++ 语言演化而来。它在语句、表达式和运算符方面使用了许多 C++ 功能。它在类型安全性、版本转换、事件和垃圾回收等方面进行了相当大的改进和创新。它提供对常用 API(例如, .NET Framework、COM+ 等)的访问。

C# 自推出以来,已得到不断的改进和优化,通常同 .NET Framework 一起,随新版的 Visual Studio .NET 的发布而更新。目前,C# 最新的版本是 C# 7.0,该版本是 2017 年 3 月 8 日微软公司正式发布 Visual Studio .NET 2017 时发布的。

本书以 .NET Framework 4.6.2 和 Visual Studio .NET 2017 为范本,所有案例均在 Visual Studio .NET 2017 中经过调试运行无误。

1.1.3 C# 语言的特点

C# 是一种简洁、类型安全的面向对象的语言,开发人员可以用它来构建运行在 .NET Framework 上的各种安全、可靠的应用程序,包括控制台应用程序、Windows 窗体应用程序、Web 应用程序等。借助 Xamarin 插件,C# 还可以用于开发 iOS、Android、Windows Phone 和 Mac App 等应用等。

作为一种面向对象的语言,C# 支持封装、继承和多态性的概念。所有的变量和方法,包括 Main 方法(应用程序的入口点),都封装在类定义中。C# 程序的生成过程比 C 和 C++ 简单,比 Java 更为灵活,没有单独的头文件,也不要求按照特定顺序声明方法和类型。C# 源文件可以定义任意数量的类、结构、接口和事件。

相对其他计算机程序设计语言来说,C# 具有如下优点。

(1) C# 是一种精确、简单、类型安全、面向对象的语言。正是由于 C# 面向对象的卓越设计,使它成为构建各种应用程序组件的理想之选——无论是高级的商业对象还是系统级的应用程序。

(2) C# 具有生成持久系统级组件的能力,提供 COM+ 或其他技术平台支持以集成现有代码,提供垃圾回收和类型安全以实现应用程序的可靠性,提供内部代码信任机制以保证应用程序的安全性。

(3) C# 利用 .NET Framework 的通用类型系统能够与其他程序设计语言交互操作。C# 应用程序能跨语言、跨平台互相调用。使用 C# 语言可实现具有不同专业技术背景的人员协同工作,完成软件系统的设计和开发。

(4) C# 支持 MSMQ(微软消息队列服务)、COM+ 组件服务、WCF 服务和 .NET Framework。使用 C# 语言,一方面实现组件之间的相互调用,也就实现了使用不同软件技术开发组件之间的集成应用。另一方面能够把传统的组件转化为 XML Web 服务,实现了组件之间的跨互联网调用。

(5) C# 语言允许自定义数据类型,用来扩展元数据。这些元数据可以应用于任何对象。项目构建者可以定义领域特有的属性并把他们应用于任何语言元素——类、接口等。然后,开发人员可以编程检查每个元素的属性。这样,很多工作都变得方便多了,比如编写一个小工具,用来自动检查每个类或接口是否被正确定义为某个抽象商业对象的一部分,或者只是创建一份基于对象领域特有属性的报表。定制的元数据和程序代码之间的紧密对应有助于加强程序的预期行为和真正实现之间的对应关系。