“十二五”普通高等教育本科国家级规划教材
普通高等学校计算机教育“十二五”规划教材

C# 程序设计教程

（第3版）

C# PROGRAMMING
(3rd edition)

马骏 ◆ 主编

人民邮电出版社
POSTS & TELECOM PRESS

“十二五”普通高等教育本科国家级规划教材
普通高等学校计算机教育“十二五”规划教材

C# 程序设计教程

（第3版）

C# PROGRAMMING
(3rd edition)

马骏 ◆ 主编

人 民 邮 电 出 版 社

图书在版编目（CIP）数据

C#程序设计教程 / 马骏主编. -- 3版. -- 北京 : 人民邮电出版社, 2014.1
普通高等学校计算机教育“十二五”规划教材
ISBN 978-7-115-33100-7

Ⅰ. ①C… Ⅱ. ①马… Ⅲ. ①C语言－程序设计－高等学校－教材 Ⅳ. ①TP312

中国版本图书馆CIP数据核字(2013)第240540号

内 容 提 要

本书主要介绍C#语言、WinForm和WPF应用程序开发的基础知识。全书共14章，前6章介绍C#语言和WinForm开发的基础知识，包括开发环境、基本数据类型、流程控制语句、类和结构、接口委托与事件、泛型与LINQ、目录与文件操作等；后8章介绍如何开发WPF应用程序，包括WPF控件、资源与样式控制、动画与多媒体、数据绑定与数据验证、数据库与实体数据模型、二维图形图像处理、三维图形和三维呈现。同时在附录中给出了本书的上机练习和综合实验。

本书提供配套的PPT课件以及在VS2012下调试通过的所有参考源程序和全部习题参考解答。

本书可作为高等院校计算机及相关专业的教材，也可作为初、中级程序员的参考用书。

◆ 主　　编　马　骏
责任编辑　邹文波
责任印制　彭志环　焦志炜

◆ 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
北京昌平百善印刷厂印刷

◆ 开本：787×1092　1/16
印张：21.75　　2014年1月第3版
字数：571千字　　2014年1月北京第1次印刷

定价：42.00元

读者服务热线：(010)81055256　印装质量热线：(010)81055316
反盗版热线：(010)81055315

第 3 版前言

C#语言是一种完全面向对象的基于.NET 的编程语言，先后被欧洲计算机制造商协会和国际标准化组织批准为高级语言开发标准（ECMA-334、ISO/IEC 23270）。随着.NET 技术的普及，C#语言已经成为开发基于.NET 的企业级应用程序的首选语言。

本书第 2 版以高度的实用性和通俗易懂的讲解，受到读者的普遍欢迎。第 3 版在继承第 2 版特色的基础上，结合作者多年的教学经验和项目研发经验，并根据近几年教学改革的实践以及对人才培养的高标准要求，对内容做了进一步的精选、优化和完善。特别是针对初学者比较容易糊涂的地方做了更为精准的阐述。

本书共分 14 章，各章的主要内容如下。

第 1 章介绍 C#代码的编写基础，包括 C#的开发环境、项目组织结构等，这些知识是学习 C#应用程序项目开发的重要基础。

第 2 章介绍 C#基本类型和流程控制语句，这是理解和学习后续章节内容的基础，要求读者必须掌握。

第 3 章和第 4 章主要介绍 C#面向对象编程的基本知识。其中第 3 章介绍类和结构的定义等面向对象编程基础知识，第 4 章主要介绍接口、委托、事件等高级面向对象编程的知识。

第 5 章和第 6 章分别介绍泛型类的用法、LINQ 查询和目录与文件管理和读写操作的相关用法。其中第 5 章介绍的 LINQ 查询是为了对后续章节学习 ADO.Net Entity Framework 打下基础。

第 7 章和第 8 章介绍 WPF 应用程序入门知识和 WPF 控件的基本方法。

第 9 章到第 11 章介绍 WPF 中资源与样式控制的方法、使用 WPF 实现动画的基本方法和 WPF 中的数据验证方法。

第 12 章主要介绍数据库以及 ADO.Net Entity Framework 的相关知识，通过本章的学习，读者应学会如何对数据库进行操作。

第 13 章和第 14 章介绍二维图形图像处理和三维图形处理的相关知识。

各高校在教学过程中，可以根据专业课程体系和学期总学时数，选取本书的全部或部分内容讲解，建议各章学时分配如下。

54 学时				72 学时			
第 1 章	4 学时	第 8 章	6 学时	第 1 章	4 学时	第 8 章	9 学时
第 2 章	4 学时	第 9 章	4 学时	第 2 章	4 学时	第 9 章	6 学时
第 3 章	3 学时	第 10 章	4 学时	第 3 章	4 学时	第 10 章	6 学时
第 4 章	2 学时	第 11 章	2 学时	第 4 章	3 学时	第 11 章	3 学时
第 5 章	4 学时	第 12 章	4 学时	第 5 章	5 学时	第 12 章	6 学时
第 6 章	5 学时	第 13 章	3 学时	第 6 章	6 学时	第 13 章	4 学时
第 7 章	6 学时	第 14 章	3 学时	第 7 章	8 学时	第 14 章	4 学时

本书由马骏担任主编，侯彦娥、谢毅、周兵、杨阳、靳冰担任副主编，马骏对全书进行了规划、组稿、统稿、修改和定稿。马骏、侯彦娥、谢毅、周兵、杨阳、李爱萍、相洁、韩道军、王洪海、靳冰、马巧梅、谭秋林、黄亚博、刘扬、左宪禹、李铁柱、王强分别承担了本书各章的具体编写工作。

为了配合教学需要，本书还提供与教材配套的PPT教学课件、书中所有源程序代码，以及全部习题参考解答。读者可到人民邮电出版社教学服务与资源网（http://www.ptpedu.com.cn）上下载。

由于编者水平有限，书中难免存在错误之处，敬请读者批评指正。

编　者

2013年10月

目 录

第 1 篇 C#程序设计基础

第 1 章 C#代码编写基础 ……2

1.1 C#语言和 VS2012 开发环境……2
1.1.1 C#语言和.NET 框架 ……2
1.1.2 VS2012 开发环境……3
1.2 C#项目的组织 ……4
1.2.1 命名空间……4
1.2.2 using 关键字 ……4
1.2.3 Main 方法 ……5
1.2.4 代码注释……5
1.2.5 通过断点调试 C#程序 ……6
1.3 控制台应用程序 ……7
1.3.1 控制台应用程序的输入与输出 ……7
1.3.2 在控制台应用程序中输出格式化数据……8
1.4 Windows 窗体应用程序 ……12
1.4.1 Windows 窗体应用程序的特点……12
1.4.2 Windows 窗体应用程序的启动和退出 ……13
1.4.3 窗体的创建、显示、隐藏和关闭……13
1.4.4 消息框（MessageBox） ……16
1.4.5 利用 WinForm 控件实现输入和输出……17
1.4.6 错误提示（ErrorProvider） ……21
1.5 WPF 和 Silverlight 应用程序 ……23
1.5.1 WPF 应用程序……23
1.5.2 Silverlight 应用程序 ……23
1.6 其他应用程序模板 ……24
习题……24

第 2 章 基本数据类型和流程控制语句 ……25

2.1 数据类型和运算符 ……25
2.1.1 C#的类型系统 ……25
2.1.2 常量与变量……26
2.1.3 运算符与表达式……27
2.2 简单类型 ……28
2.2.1 整型……29
2.2.2 浮点型……29
2.2.3 布尔型（bool） ……30
2.2.4 字符（char） ……30
2.2.5 枚举（enum） ……31
2.3 字符串 ……33
2.3.1 字符串的创建与表示形式 ……33
2.3.2 字符串的常用操作方法 ……34
2.3.3 String 与 StringBuilder……37
2.4 数组 ……37
2.4.1 一维数组……38
2.4.2 多维数组……38
2.4.3 交错数组……39
2.4.4 数组的常用操作方法 ……41
2.5 数据类型之间的转换 ……43
2.5.1 值类型之间的数据转换 ……43
2.5.2 值类型和引用类型之间的转换 ……44
2.6 流程控制语句 ……45
2.6.1 分支语句……45
2.6.2 循环语句……51
2.6.3 跳转语句……54
2.6.4 异常处理语句……56
习题……57

第3章 类和结构 ……59

3.1 自定义类（class）和结构（struct）……59
3.1.1 类的定义和成员组织……59
3.1.2 访问修饰符……60
3.1.3 静态成员和实例成员……62
3.1.4 构造函数……62
3.1.5 字段和局部变量……64
3.1.6 结构的定义和成员组织……65
3.2 属性和方法……67
3.2.1 属性（Property）……67
3.2.2 方法……68
3.3 类的继承与多态性……72
3.3.1 封装……72
3.3.2 继承……73
3.3.3 多态（new、virtual、override）……77
3.4 常用结构和类的用法……80
3.4.1 Math类……80
3.4.2 DateTime结构和TimeSpan结构……81
3.4.3 秒表、计时和随机数（Stopwatch、Timer、Random）……83
习题……85

第4章 接口、委托与事件……86

4.1 接口……86
4.1.1 接口的声明和实现……86
4.1.2 显式方式实现接口……88
4.1.3 利用接口实现多继承……89
4.2 委托……90
4.2.1 定义委托类型……91
4.2.2 通过委托调用方法……91
4.3 事件……93
4.3.1 事件的声明和引发……93
4.3.2 具有标准签名的事件……94
习题……95

第5章 泛型与LINQ……96

5.1 C#的类型扩展……96
5.1.1 匿名类型和隐式类型的局部变量……96
5.1.2 对象初始化和集合初始化……96
5.2 泛型和泛型集合……100
5.2.1 列表和排序列表……100
5.2.2 字典和排序字典……102
5.3 LINQ查询表达式……104
5.3.1 延迟执行和立即执行……104
5.3.2 from子句……105
5.3.3 where子句……106
5.3.4 orderby子句……107
5.3.5 group子句……108
5.3.6 select子句……109
5.3.7 查询多个对象……109
习题……111

第6章 目录与文件操作……112

6.1 目录和文件管理……112
6.1.1 Environment类和DriveInfo类……112
6.1.2 Path类……114
6.1.3 目录管理……114
6.1.4 文件管理……116
6.2 文件的读写……117
6.2.1 文件编码……117
6.2.2 文本文件的读写……118
6.2.3 StreamReader类和StreamWriter类……119
习题……120

第2篇 WPF应用程序

第7章 WPF应用程序入门……122

7.1 WPF应用程序和XAML标记……122
7.1.1 WPF应用程序的关闭模式及Shutdown方法……122
7.1.2 XAML命名空间和x:前缀编程构造……125

7.1.3 XAML 基本语法 ····· 126
7.2 窗口和对话框 ····· 129
7.2.1 WPF 窗口 ····· 129
7.2.2 在主窗口显示前先显示登录窗口或者欢迎窗口 ····· 130
7.2.3 对话框 ····· 133
7.2.4 WPF 页和页面导航 ····· 134
7.3 颜色和形状 ····· 134
7.3.1 Brushe 类和 Colors 类 ····· 134
7.3.2 Color 结构 ····· 135
7.3.3 形状 ····· 135
7.4 画笔（Brush） ····· 138
7.4.1 画笔分类 ····· 138
7.4.2 利用 WPF 设计器实现画笔变换 ····· 140
7.5 属性和事件 ····· 141
7.5.1 依赖项属性和附加属性 ····· 141
7.5.2 事件 ····· 142
习题 ····· 148

第 8 章 WPF 控件 ····· 149

8.1 控件模型和内容模型 ····· 149
8.1.1 WPF 控件模型 ····· 149
8.1.2 WPF 内容模型 ····· 155
8.2 常用布局控件 ····· 157
8.2.1 WPF 的布局分类 ····· 157
8.2.2 网格（Grid） ····· 157
8.2.3 堆叠面板（StackPanel） ····· 158
8.2.4 画布（Canvas） ····· 159
8.2.5 边框（Border） ····· 160
8.2.6 停靠面板（DockPanel） ····· 160
8.3 常用基本控件 ····· 161
8.3.1 按钮（Button、RepeatButton） ····· 161
8.3.2 文本块（TextBlock）和标签（Label） ····· 162
8.3.3 文本框（TextBox、PasswordBox、RichTextBox） ····· 163
8.3.4 单选按钮（RadioButton） ····· 165
8.3.5 复选框（CheckBox） ····· 167
8.3.6 列表框（ListBox）和下拉框（ComboBox） ····· 168
8.4 菜单、工具条和状态条 ····· 171
8.4.1 菜单（Menu）和快捷菜单（ContextMenu） ····· 171
8.4.2 工具条（ToolBar、ToolBarTray）和状态条（StatusBar） ····· 173
8.5 图像（Image） ····· 177
习题 ····· 178

第 9 章 资源与样式控制 ····· 179

9.1 XAML 资源和样式控制 ····· 179
9.1.1 XAML 资源 ····· 179
9.1.2 Style 元素 ····· 181
9.1.3 在 Style 元素中设置属性和事件 ····· 182
9.1.4 样式的级联控制 ····· 183
9.1.5 使用 C#代码定义和引用样式 ····· 188
9.2 在 Style 元素中使用模板和触发器 ····· 190
9.2.1 模板 ····· 190
9.2.2 触发器 ····· 192
习题 ····· 194

第 10 章 动画与多媒体 ····· 195

10.1 WPF 动画基础 ····· 195
10.1.1 WPF 动画的分类 ····· 195
10.1.2 Storyboard 和 Timeline ····· 196
10.2 基本动画（From/To/By） ····· 203
10.2.1 基本动画类型 ····· 203
10.2.2 用 Storyboard 实现基本动画 ····· 204
10.3 关键帧动画 ····· 206
10.3.1 关键帧动画类型 ····· 206
10.3.2 利用 Blend for VS2012 制作关键帧动画 ····· 207
10.4 路径动画 ····· 209
10.4.1 使用 PathGeometry 绘制路径 ····· 209
10.4.2 路径动画类型 ····· 211
10.4.3 利用 Blend for VS2012 制作路径动画 ····· 213
10.5 语音、音频和视频 ····· 216
10.5.1 语音 ····· 216
10.5.2 音频和视频（MediaElement） ····· 218

习题……221

第11章　数据绑定与数据验证……222

11.1　数据绑定……222
11.1.1　数据绑定基本概念……222
11.1.2　简单数据绑定……226
11.1.3　数据模板化……235
11.1.4　通过数据模板和视图绑定到集合……239
11.2　数据验证……241
11.2.1　数据验证的基本概念……241
11.2.2　利用验证规则和绑定模型实现验证……243
习题……250

第12章　数据库与实体数据模型……251

12.1　创建数据库和表……251
12.1.1　ADO.NET 数据访问技术……251
12.1.2　SQL Server 2012 简介……252
12.1.3　创建 LocalDB 数据库……253
12.2　利用实体框架创建实体数据模型……256
12.2.1　实体框架基本概念……256
12.2.2　实体框架开发模式……256
12.2.3　从数据库创建实体数据模型……257
12.3　使用 LINQ to Entities 访问实体对象……258
12.3.1　创建实体框架上下文（DbContext）实例……258
12.3.2　加载相关对象……260
12.3.3　查询数据……261
12.3.4　修改数据……263
12.3.5　添加或删除数据……265
12.4　DataGrid 控件……267
12.4.1　绑定各种类型的数据……267
12.4.2　标题和行列控制……272
习题……275

第13章　二维图形图像处理……276

13.1　图形图像处理基础……276
13.1.1　与二维图形图像处理相关的类……276
13.1.2　创建本章例子的主程序……277
13.2　二维图形处理……278
13.2.1　二维几何图形和路径标记语法……278
13.2.2　绘制基本图形……281
13.2.3　将格式化文本转换为图形……288
13.3　图像处理……290
13.3.1　图像处理常用类……290
13.3.2　图像的编码和解码……291
13.4　利用画笔绘制图形图像……295
13.4.1　TileBrush 类……296
13.4.2　图像画笔（ImageBrush）……299
习题……300

第14章　三维图形和三维呈现……301

14.1　WPF 三维设计基本知识……301
14.1.1　Viewport3D 控件……301
14.1.2　照相机（Camera）……304
14.1.3　三维几何模型（GeometryModel3D）……306
14.1.4　光照类型……306
14.1.5　材料（Meterial）……307
14.2　在窗口或页面中呈现三维场景……310
14.2.1　利用相机变换制作 3D 场景观察器……310
14.2.2　动态显示相机的属性……310
14.2.3　三维网格几何（MeshGeometry3D）……312
14.3　三维建模和自定义三维模型类……316
14.3.1　利用模型编辑器创建和编辑三维模型……316
14.3.2　创建自定义三维模型类……319
14.3.3　利用三维模型库简化场景构建……321
14.4　对模型进行变换处理……324
14.4.1　三维变换处理基础……324
14.4.2　将三维变换封装到模型库中……326
习题……328

附录A　上机练习……329

A.1　上机练习要求……329
A.2　第1章和第2章上机练习……330

A.2.1　密码输入和显示练习（WinForm）……330
A.2.2　简单计算器设计练习（WinForm）……330
A.2.3　字符提取和整数整除练习（Console）……331
A.2.4　数组排序和计算练习（Console）……331
A.3　第 3 章和第 4 章上机练习……331
A.3.1　类及其属性和方法的实现练习（WinForm）……331
A.3.2　定时器和随机数练习（WinForm）……332
A.4　第 5 章和第 6 章上机练习……332
A.4.1　泛型和 LINQ 练习（WinForm）……332
A.4.2　文本文件读写练习（WinForm）……333
A.5　第 7 章和第 8 章上机练习……333
A.5.1　用户登录练习（WPF）……333
A.5.2　控件基本功能练习（WPF）……334
A.5.3　数学测验过关小游戏（WPF）……334
A.6　第 9 章和第 10 章上机练习……335
A.6.1　样式定义和应用练习……335
A.6.2　垂直柱状图动画练习（WPF）……335
A.7　第 11 章和第 12 章上机练习……335
A.7.1　数据验证练习（WPF）……335
A.7.2　数据库设计练习（WPF）……336
附录 B　综合实验……337
B.1　系统功能要求……337
B.2　成果提交……338

第 1 篇
C#程序设计基础

C#是一种完全面向对象的高级程序设计语言，同时也是一种面向组件的编程语言。用 C#开发应用程序项目时，必须首先掌握一些基本知识，包括：数据类型、语句、类和对象、结构、属性、方法、继承、接口、委托、事件以及 LINQ 等。

控制台是一个操作系统级别的命令行窗口，学习 C#基本概念及其用法时，一般用控制台应用程序来实现，这样可以避免其他代码的干扰，但这些基本知识同样适用于 Windows 窗体应用程序、WPF 应用程序、Silverlight 应用程序，以及其他类型的应用程序。

Windows 窗体应用程序是微软公司推出 Windows XP 操作系统时重点推出的基于.NET 和 GDI+的应用程序编程模型，开发在 Windows XP 操作系统上运行的应用程序项目时，一般用这种编程模型来实现。

WPF 应用程序和 Silverlight 应用程序是微软公司推出的基于.NET 和 DirectX 的应用程序编程模型。开发在 Windows 7 操作系统上运行的客户端应用程序时，建议用 WPF 应用程序来实现，这样可以充分发挥 GPU 硬件加速的性能优势。

但是，由于 WPF 应用程序的很多概念是建立在开发人员已经对 Windows 窗体应用程序有所了解的前提下来进一步介绍的，因此在学习 WPF 应用程序之前，我们还需要学习 Windows 窗体应用程序的一些基本概念和用法。

本书第 1 篇（第 1 章～第 6 章）分别用控制台应用程序和 Windows 窗体应用程序来讲解 C#编程的基本知识。

第 1 章 C#代码编写基础

作为本书的入门知识，这一章我们先简单了解 C#语言、VS2012 开发环境以及常用应用程序的分类，并通过控制台应用程序和 WinForm 应用程序学习 C#的基本编程方法，主要目标是理解如何用它设计最基本的界面，以及实现数据的输入和输出，体会不同应用程序的优缺点和适用环境，为后续章节的学习做好准备。

1.1 C#语言和 VS2012 开发环境

C#（读作“C sharp”）是一种完全面向对象的基于 Microsoft.NET 框架（简称.NET）的高级程序设计语言，Visual Studio 2012（简称 VS2012）是微软研制的、基于.NET 平台的软件开发工具，该开发工具除了支持 C#以外，还支持 C++等其他开发语言。

1.1.1 C#语言和.NET 框架

C#是专门为快速编写在.NET 框架上运行的各种应用程序而设计的。该语言在保持 C 和 C++语言风格的表现力和雅致特征的同时，简化了 C++的复杂性（例如没有宏以及不允许多重继承等），同时综合了 VB 简单的可视化操作和 C++的高运行效率，以其强大的操作能力、优雅的语法风格、创新的语言特性和便捷的面向组件编程的支持，而备受开发人员青睐。C#凭借许多方面的创新，实现了应用程序的快速开发，成为.NET 平台的首选编程语言。

1. C#语言的特点

C#语言主要有以下特点。

（1）语法简洁。C#用最简单、最常见的形式进行类型描述，语法简洁、优雅。

（2）精心的面向对象设计。C#语言一开始就是完全按照面向对象的思想来设计的，因此，它具有面向对象所应有的所有特性。除此之外，C#还为面向组件的开发提供了方便的实现技术。

（3）与 Web 的紧密结合。在 C#语言中，复杂的 Web 编程看起来更像是对本地对象进行操作，从而简化了大规模、深层次的分布式开发。用 C#语言构建的 Web 组件能够方便地作为 Web 服务（Web Service），并可以通过 Internet 被各种编程语言所调用。

（4）可靠的安全性与强大的错误处理能力。语言的安全性与错误处理能力是衡量一种语言是否优秀的重要依据。C#语言可以消除许多软件开发中的常见错误，并提供了完整的安全开发性能。例如，提供了完善的边界与溢出检查，不允许使用未初始化的局部变量等。另外，自动垃圾回收机制也极大地减轻了开发人员对内存管理的负担。

（5）可靠的版本控制技术。C#语言内置了版本控制功能，不会出现发布或安装应用程序时与其他软件冲突等情况。同时，智能客户端技术使客户端软件的下载、升级变得非常简单，开发人员只需要关注软件开发，而软件部署以及升级后的更新和维护则由系统自动去实现。

（6）灵活性和兼容性。灵活性是指用 C#语言编写的组件可以与其他语言编写的组件进行交互，兼容性是指 C#语言也可以与 COM 以及操作系统底层的 API 进行交互。

2. Microsoft.NET 框架

Microsoft.NET 框架是生成、运行.NET 应用程序和 Web Service 的组件库。基于.NET 框架开发的应用程序，不论使用的是哪种高级语言，均必须在安装了.NET 框架的计算机上才能运行。这种架构与 Java 应用程序必须由 Java 虚拟机支持相似。

.NET 框架包括两个主要组件，一个是公共语言运行库（Common Language Runtime，CLR），另一个是类库。

公共语言运行库提供.NET 应用程序所需要的核心服务；类库是与公共语言运行库紧密集成的可重用的类的集合，旨在为开发和运行.NET 应用程序提供各种支持。

.NET 框架 4.5 版的类库由近 5 000 个类组成，这些类提供了 Internet 和企业级开发所需要的各种功能，为开发各种.NET 应用程序提供了很大的方便。

类库中的每个类均按照功能划分到不同的命名空间下。

1.1.2　VS2012 开发环境

微软公司推出的 C#语言开发环境是 Visual Studio 系列开发工具，目前的最新版本是 VS2012。在学习本书之前，需要安装和配置开发环境。

1. 安装 VS2012

VS2012 分为速成版（Express Edition，免费，但功能有限制）、专业版（Professional Edition，收费，团队开发，有些功能不提供）和旗舰版（Ultimate Edition，功能最全的版本）。

本书所有的例子全部都在 VS2012 简体中文旗舰版开发环境下调试通过。

调试本书例子的具体安装要求如下。

（1）操作系统：Windows 7（32 位或者 64 位），建议使用 64 位 Windows 7。

（2）内存：至少 2GB。建议 4GB 或者更高。

由于 VS2012 的安装过程比较简单，所以这里不再介绍具体安装步骤。

2. 安装 VS2012 SP2

Blend for Visual Studio 2012 是微软公司研制的、针对 VS2012 的界面和原型设计工具。在设计 XAML 动画和生成 XAML 格式的三维模型时，我们需要使用这个工具。

在 Windows 8 下安装 VS2012 后，它会自动安装 Blend for VS2012，但该版本在没有安装 VS2012 SP2 的情况下无法在 Windows 7 操作系统下运行。由于本书使用的操作系统是 Windows 7，因此安装 VS2012 后，还需要安装 VS2012 SP2 才能使用 Blend for VS2012。

从微软公司的网站上下载 VS2012 SP2 后直接安装即可。

1.2 C#项目的组织

在VS2012中，C#程序是通过解决方案中的项目来组织的。默认情况下，一个解决方案中只包含一个项目，解决方案名和项目名默认相同。但是，在实际的应用开发中，一个解决方案往往会包含多个项目。

C#源文件的扩展名为.cs，如Welcome.cs。一个C#源文件中一般只包含一个类，但也可以包含多个类，文件名和类名可以相同，也可以不同。

在VS2012调试环境下，项目编译后生成的文件默认保存在项目的bin\debug文件夹下。

1.2.1 命名空间

VS2012开发环境为开发人员提供了非常多的类，利用这些类可快速完成各种各样复杂的功能。根据功能不同，这些类分别划分到不同的命名空间中。

命名空间的划分方法有点类似于子目录和文件划分的方式，不同命名空间下的类名可以相同，也可以不同。调用某命名空间下某个类提供的方法时，命名空间、类名之间都用点（.）分隔，一般语法如下：

命名空间.命名空间……*命名空间.类名称.静态方法名*（*参数*，……）；

若方法为实例方法，需要先进行类的实例化，然后再通过实例访问方法，一般语法如下：

实例名称.方法名（*参数*，……）；

语法中的下画线表示其内容需要用实际的代码替换。例如：

```
System.Console.WriteLine("Hello World");
```

这条语句调用System命名空间下Console类的WriteLine方法输出字符串“Hello World”。

1.2.2 using关键字

在C#中，有一个特殊的关键字称为“using”。该关键字有3种用途：

（1）作为引用指令，用于为命名空间导入其他命名空间中定义的类型。

为了快速引用需要的功能，一般在程序的开头引用命名空间来简化代码表示形式。如果在程序的开头加上：

```
using System;
```

则：

```
System.Console.WriteLine("Hello World");
```

就可以直接写为

```
Console.WriteLine("Hello World");
```

（2）作为别名指令，用于简化命名空间的表达形式。

除了使用using关键字指定引用的命名空间外，还可以使用using简化命名空间的层次表达形式，例如：

```
using System.Windows.Forms;
```

可以表示为

```
using WinForm=System.Windows.Form;
```

这样一来，语句

```
System.Windows.Form.MessageBox.Show("hello");
```

就可以简写为

```
WinForm.MessageBox.Show("hello");
```

（3）作为语句，用于定义一个范围。

在 C#语言中，using 关键字还可用来创建 using 语言，该语句的作用是定义一个用大括号包围的范围，程序执行到此范围的末尾，就会立即释放在 using 的小括号内创建的对象。例如：

```
static void Main()
{
    using (TextWriter w = File.CreateText("test.txt"))
    {
        w.WriteLine("Line one");
        w.WriteLine("Line two");
        w.WriteLine("Line three");
    }
}
```

这段代码中的 using 语句表示程序执行到它所包含的语句块的末尾“}”时，会立即释放 TextWriter 对象占用的内存资源。

如果某个范围内有多个需要立即释放的对象，可以用嵌套的 using 语句来实现。

1.2.3　Main 方法

C#的每一个应用程序都有一个入口点，以便让系统知道该程序从哪里开始执行。为了让系统能找到入口点，入口方法名规定为 Main。注意：“Main”的首字母大写，而且 Main 方法后面的小括号不能省略。

Main 方法只能声明为 public static int 或者 public static void，这是 C#程序的规定。另外，每一个方法都要有一个返回值，对于没有返回值的方法，必须声明返回值类型为 void。

Mian 方法的返回值只能有两种类型，一种是 int，另一种是 void。int 类型的返回值用于表示应用程序终止时的状态码，其用途是退出应用程序时返回程序运行的状态（0 表示成功返回，非零值一般表示某个错误编号，错误编号所代表的含义也可以由程序员自己规定）。

当 Main 方法的返回类型为 void 时，返回值始终为零。

Main 方法可以放在任何一个类中，但为了让开发人员容易找到入口点，控制台应用程序和 Windows 窗体应用程序默认将其放在 Program.cs 文件的 Program 类中。

1.2.4　代码注释

在源代码中加上注释是优秀编程人员应该养成的好习惯。C#语言中添加注释的方法主要有以下几种形式。

1. 常规注释方式

单行注释：以“//”符号开始，任何位于“//”符号之后的本行文字都视为注释。

块注释：以“/*”开始，再以“*/”结束。任何介于这对符号之间的文字块都视为注释。

2. XML 注释方式

“///”符号是一种 XML 注释方式，只要在用户自定义的类型（如类、接口、枚举等）或者在其成员上方，或者命名空间的声明上方连续键入 3 个斜杠字符“/”，系统就会自动生成对应的 XML 注释标记。添加 XML 注释的步骤举例如下：

（1）首先定义一个类、方法、属性、字段或者其他类型。例如，在 Studentinfo.cs 中定义一个 PrintInfo 方法。

（2）在类、方法、属性、字段或者其他类型声明的上面键入 3 个斜杠符号（“/”），此时开发环境就会自动添加对应的 XML 注释标记。例如，先编写一个 PrintInfo 方法，然后在该方法的上面键入 3 个斜杠符号后，就会得到下面的 XML 注释代码：

```
/// <summary>
///
/// </summary>
public void PrintInfo ( )
{
    Console.WriteLine("姓名: {0},年龄: {1}", studentName, age);
}
```

（3）在 XML 注释标记内添加注释内容。例如，在<summary>和</summary>之间添加方法的功能描述。

以后调用该方法时，就可以在键入方法名和参数的过程中直接看到用 XML 注释的智能提示。

1.2.5 通过断点调试 C#程序

断点是调试程序时使用的一种特殊标识。如果在某条语句前设置断点，则当程序执行到这条语句时会自动中断程序运行，进入调试状态（注意此时还没执行该语句）。断点的设置方法与使用的调试工具有关。

利用断点查找程序运行的逻辑错误，是调试程序常用的手段之一。

1. 设置和取消断点

在 VS2012 中，设置和取消断点的方法有下面几种。

方法 1：用鼠标单击某代码行左边的灰色区域。单击一次设置断点，再次单击取消断点。

方法 2：用鼠标右键单击某代码行，从弹出的快捷菜单中选择【断点】→【插入断点】或者【删除断点】命令。

方法 3：用鼠标单击某代码行，直接按<F9>键设置或取消断点。

断点设置成功后，在对应代码行的左边会显示一个红色的实心圆标志，同时该行代码也突出显示。

断点可以有一个，也可以有多个。

2. 利用断点调试程序

设置断点后，即可运行程序。程序执行到断点所在的行，就会中断运行。需要注意的是，程序中断后，断点所在的行还没有执行。

当程序中断后，如果将鼠标放在变量或实例名的上面，调试器就会自动显示执行到断点时该变量的值或实例信息。

观察以后，可以按<F5>键继续执行到下一个断点。

如果大范围调试仍然未找到错误之处，也可以在调试器执行到断点处停止后，按<F11>键逐语句执行，按一次执行一条语句。

还有一种调试的方法，即按<F10>键“逐过程”执行，它和“逐语句”执行的区别是，系统把一个过程（例如类、方法等）当作一条语句，而不再转入到过程内部。

1.3　控制台应用程序

控制台是一个操作系统级别的命令行窗口。利用它，用户可通过键盘输入文本字符串，并在显示器上将文本显示出来。在 Windows 系列操作系统中，控制台也称为命令提示窗口，可以接受 MS-DOS 命令。

控制台应用程序的优点是占用的内存资源极少，特别适用于对界面要求不高的场合。不论是哪种编程语言，控制台应用程序都是最基本的应用程序。

另外，学习和理解 C#基本概念时，使用控制台应用程序可以让我们重点关注基本知识和程序逻辑，而不被其他代码干扰。C#语言的这些基本知识和概念同样适用于其他各种类型的应用程序。

1.3.1　控制台应用程序的输入与输出

控制台应用程序实际上应该在命令行窗口下运行，即在命令行提示符下，键入可执行文件名（.exe 文件）和参数，然后按回车键运行程序。但是，在 VS2012 开发环境下，为了避免频繁地在开发环境和命令行提示窗口之间切换，也可以直接按<F5>键编译并调试运行控制台应用程序，可在运行时，程序员会发现屏幕一闪就过去了，无法看清楚输出的内容。为了在调试环境下能直接观察输出的结果，一般在 Main 方法结束前加上 Console.ReadKey();语句，意思是读取键盘输入的任意一个字符，按一下键盘上的空格键、回车键或者其他任何一个字符键，就又返回到开发环境下了。

1. 控制台输出

默认情况下，System 命名空间下的 Console 类提供的 Write 方法和 WriteLine 方法自动将各种类型的数据转换为字符串写入标准输出流，即输出到控制台窗口中。

Write 方法与 WriteLine 方法的区别是后者在输出数据后，还自动输出一个回车换行符，即将光标自动转到下一行。

下面是 Write 方法和 WriteLine 方法的一些基本用法示例。

```
int age = 18;
string s = "abc";
Console.Write(age);
Console.Write(s);
Console.WriteLine(age);
Console.WriteLine(s);
```

这段代码的输出结果如下：

```
18abc
18
abc
```

2. 控制台输入

System 命名空间下的 Console 类提供了一个 ReadLine 方法，该方法从标准输入流依次读取从键盘输入的字符，并将被按下的字符立即显示在控制台窗口中，而且在用户按下回车键之前会一直等待输入，直到用户按下回车键为止。

下面的代码演示了 ReadLine 方法的简单用法。

```
string s = Console.ReadLine();
if (s == "abc")
```