

21世纪高等教育计算机规划教材



C# 程序设计

C# Programming

■ 孙志辉 主编

■ 姚琳 万亚东 副主编

— 实例丰富，面向应用

— 循序渐进，深入浅出

— 体系完整，注重细节



人民邮电出版社
POSTS & TELECOM PRESS

21世纪高等教育计算机规划教材

COMPUTER

C# 程序设计

C# Programming

■ 孙志辉 主编

■ 姚琳 万亚东 副主编



人民邮电出版社

北京

C#程序设计 / 孙志辉主编. — 北京 : 人民邮电出版社, 2015.1
21世纪高等教育计算机规划教材
ISBN 978-7-115-37699-2

I. ①C… II. ①孙… 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(2014)第281955号

内 容 提 要

本书是学习 C#语言的基础教程, 采用语法与实例相结合的方式, 介绍利用 C#语言进行 Windows 程序设计的基本方法。全书共分 7 章, 主要内容包括 C#语言概述、C#语言基本语法、C#语言面向对象编程、Windows 程序设计、文件操作、图形操作和 ADO.NET 与数据库操作。本书在介绍 C#语法和各种对象的基础上, 设计了简单屏幕保护程序、自动翻页记事本程序、电子闹钟程序和学生信息管理程序。

本书是一本语法与实例相结合的教程, 可以作为各高校计算机语言课程的教材以及 C#语言爱好者或培训机构的教材。

-
- ◆ 主 编 孙志辉
 - 副 主 编 姚 琳 万亚东
 - 责任编辑 武恩玉
 - 责任印制 沈 蓉 彭志环
 - ◆ 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>
 - 北京鑫正大印刷有限公司印刷
 - ◆ 开本: 787×1092 1/16
 - 印张: 20.5 2015 年 1 月第 1 版
 - 字数: 539 千字 2015 年 1 月北京第 1 次印刷
-

定价: 45.00 元

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

前 言

C#语言作为微软.NET 平台的重要组成部分,以其简单、快捷的编程方式,成为 Windows 窗口程序和 Web 应用程序的主流开发工具,也逐渐成为各高校计算机语言课程的讲授内容。

本书采用语法与实例相结合的方法,既介绍了 C#基本语言以及各种编程对象,又将这些语言和对象融合到几个具体实例中。通过学习这些实例,学生不但可以学会 C#语言的基本语法和主要对象,还能培养出编程的思想,以及解决实际编程问题的能力,真正做到学以致用。

本书由两大部分组成:第一部分是 C#语言的基本语法介绍,由第 1 章到第 3 章组成;第二部分则是介绍进行各种 Windows 程序设计的 C#语言的基本对象和方法,由第 4 章到第 7 章组成。

第 1 章简单介绍 C#语言发展、特点以及 Visual C#.NET 集成开发环境,并以一个控制台程序介绍 C#程序设计的基本方法。

第 2 章介绍 C#语言的基本语法,包括变量定义、运算符、程序流程控制语句以及数组。

第 3 章介绍 C#面向对象程序设计语法,主要包括类及其各成员声明、类的继承、类的多态、接口,并简单介绍了泛型、委托与事件等概念。

第 4 章介绍利用 C#进行 Windows 程序设计的方法,主要包括 Form 类以及标签、文本框、按钮等常用控件和菜单以及工具栏等的设计方法。并利用这些对象设计了简单的屏幕保护程序。

第 5 章介绍利用 C#进行文件操作的方法,主要包括 System.IO 模型的各个对象、文本文件和二进制文件的读写方法。在此基础上,设计了具有自动翻页功能的简单记事本程序。

第 6 章介绍利用 C#进行图形操作的基本方法,主要包括 GDI+各种图形操作对象、Graphics 类。并利用这些对象和类,设计了简单的电子闹钟程序。

第 7 章介绍利用 C#进行数据库操作的基本方法,主要包括 ADO.NET 数据库访问体系及其主要对象、数据的显示对象、SQL 结构化查询语句。并利用这些对象设计了学生信息管理程序。

本书由北京科技大学孙志辉任主编,姚琳、万亚东任副主编。

本教材的编写,得到了“十二五”期间北京科技大学教材建设经费资助。

由于编者水平有限,书中难免会存在一些错误,殷切希望广大读者提出批评指正意见。

编 者

2014 年 12 月

目 录

第 1 章 C#概述 1

- 1.1 .NET Framework 开发平台 1
- 1.2 C#及其与 C、C++的区别 3
- 1.3 C#版本的发展 4
- 1.4 Visual Studio 2013 集成开发环境 5
- 1.5 第一个简单的 C#应用程序 9
 - 1.5.1 新建控制台程序 9
 - 1.5.2 控制台程序解释与完善 12
 - 1.5.3 Console 类 14
 - 1.5.4 与项目有关的菜单与视图 16
 - 1.5.5 C#控制台项目的文件组成 20
- 1.6 习题 20

第 2 章 C#基本语法 22

- 2.1 C#语言基础 22
- 2.2 C#数据类型与变量 22
 - 2.2.1 值类型 23
 - 2.2.2 引用类型与 string 类 26
 - 2.2.3 值类型与引用类型的转换（“装箱”与“拆箱”） 28
 - 2.2.4 基本类型与 string 类型的相互转换 29
 - 2.2.5 C#常量 29
 - 2.2.6 C#变量 32
 - 2.2.7 可空类型 33
- 2.3 C#运算符与表达式 34
 - 2.3.1 算术运算符 34
 - 2.3.2 关系运算符 36
 - 2.3.3 逻辑运算符 36
 - 2.3.4 位运算符 37
 - 2.3.5 赋值运算符 40
 - 2.3.6 条件运算符 40

- 2.3.7 is 与 as 运算符 40
- 2.3.8 其他运算符 41
- 2.3.9 运算符的优先级 41
- 2.3.10 表达式中的类型转换 42
- 2.4 C#的条件语句 42
 - 2.4.1 if 语句 43
 - 2.4.2 switch 分支语句 44
- 2.5 C#的循环语句 45
 - 2.5.1 for 循环语句 46
 - 2.5.2 do-while 与 while 循环语句 48
 - 2.5.3 foreach 循环语句 49
- 2.6 C#的跳转语句 49
 - 2.6.1 break 语句 50
 - 2.6.2 continue 语句 50
 - 2.6.3 goto 语句 50
- 2.7 C#的异常处理结构 51
- 2.8 C#的数组 53
 - 2.8.1 一维数组 53
 - 2.8.2 多维数组 55
 - 2.8.3 不规则数组 56
 - 2.8.4 Array 数组类 57
 - 2.8.5 数组元素的插入与删除 58
 - 2.8.6 ArrayList 类 59
- 2.9 C#常用类与结构 60
 - 2.9.1 Math 类 60
 - 2.9.2 Random 类 61
 - 2.9.3 DateTime 结构 62
 - 2.9.4 Convert 类 64
 - 2.9.5 BitConvert 类 64
 - 2.9.6 Encoding 类 65
 - 2.9.7 StringBuilder 类 66
- 2.10 预处理指令 67
- 2.11 习题 68

第3章 C#面向对象编程 72

3.1 面向对象的概念	72
3.1.1 对象与类	72
3.1.2 面向对象程序设计的特征	72
3.2 C#类	73
3.2.1 类的声明	73
3.2.2 创建类对象及其实例化	75
3.2.3 类的成员	75
3.2.4 类的构造函数与析构函数	77
3.2.5 this 关键词	80
3.3 类的方法成员	80
3.3.1 return 语句	81
3.3.2 方法成员声明	81
3.3.3 方法的参数	82
3.3.4 静态方法	86
3.3.5 外部静态方法	87
3.3.6 方法重载	88
3.3.7 泛型方法	89
3.3.8 方法的递归调用	90
3.4 类的运算符重载成员	90
3.5 类的属性成员	92
3.5.1 属性成员声明与应用	92
3.5.2 自动实现属性	94
3.6 类的索引器成员	96
3.7 委托与类的事件成员	97
3.7.1 委托的声明与使用	98
3.7.2 多重委托	99
3.7.3 匿名委托与 λ 表达式	100
3.7.4 类的事件成员	101
3.8 类的继承	103
3.8.1 类的派生	104
3.8.2 派生类的构造函数与析构函数	106
3.9 类的多态	108
3.9.1 利用 new 修饰符隐藏基类方法	108
3.9.2 利用 virtual 和 override 修饰符实现虚方法重写	109
3.9.3 关于 base 关键词	114
3.9.4 抽象类与抽象方法	114
3.9.5 密封类与密封方法	116

3.9.6 基类引用与派生类对象	118
3.10 泛型类	118
3.10.1 泛型类的声明与应用	118
3.10.2 List<T> 泛型类	120
3.11 接口	121
3.11.1 接口的声明与继承	121
3.11.2 接口的实现	122
3.11.3 接口引用	126
3.11.4 IComparable 接口和 IComparer 接口	126
3.11.5 集合与 IEnumerable 接口	129
3.12 结构	130
3.13 泛型委托与泛型约束	131
3.14 Visual C# 代码编辑器中有关类的辅助功能	132
3.15 习题	133

第4章 Windows 程序设计 137

4.1 Windows 程序设计概述	137
4.1.1 Windows 程序设计特点	137
4.1.2 Windows 窗口及其主要概念	139
4.1.3 主要窗口类型	141
4.1.4 Windows 消息	141
4.2 一个简单的 Windows 窗口程序	143
4.2.1 创建 Windows 窗体应用程序	143
4.2.2 分析自动创建的 Windows 窗体应用程序代码	146
4.2.3 给自动创建的 Form1 增加几个控件	148
4.2.4 给新控件增加事件处理方法	149
4.2.5 调整控件布局	151
4.3 Form 类及控件类的公共成员	152
4.3.1 Form 类及控件类的公共属性	152
4.3.2 Form 类及控件类的公共方法	156
4.3.3 Form 类及控件类的公共事件	157
4.4 Form 类	160
4.4.1 Form 类主要属性	161
4.4.2 Form 类主要方法	164
4.4.3 Form 类主要事件	165
4.4.4 MessageBox 类	165
4.4.5 Form 类应用示例	166
4.5 常用 Windows 控件	168

4.5.1 Label 控件与 LinkLabel 控件.....	168	5.3 文件读写操作	211
4.5.2 Button 控件	169	5.3.1 采用 FileStream 类读写文件	211
4.5.3 TextBox 控件、MaskedTextBox 控件 与 RichTextBox 控件.....	170	5.3.2 读写文本文件	212
4.5.4 CheckBox 控件	173	5.3.3 读写二进制文件.....	215
4.5.5 RadioButton 控件.....	174	5.4 通用对话框	218
4.5.6 ListBox 控件与 CheckedListBox 控件	176	5.4.1 文件选择对话框.....	219
4.5.7 ComboBox 控件.....	178	5.4.2 字体选择对话框.....	221
4.5.8 GroupBox 控件、Panel 控件与 TabControl 控件.....	179	5.4.3 颜色选择对话框.....	222
4.5.9 PictureBox 控件	181	5.4.4 打印对话框与打印.....	223
4.5.10 ImageList 控件	182	5.5 自动翻页记事本程序设计.....	227
4.5.11 Timer 控件	184	5.5.1 基本界面设计	227
4.5.12 DateTimePicker 控件与 MonthCalendar 控件	184	5.5.2 文件基本操作功能.....	228
4.5.13 NumericUpDown 控件.....	186	5.5.3 文本行列信息显示	230
4.5.14 ProgressBar 控件.....	186	5.5.4 文本编辑功能	230
4.5.15 ToolTip 控件.....	187	5.5.5 文本修饰功能	231
4.6 Windows 高级界面设计.....	187	5.5.6 参数设置功能	231
4.6.1 菜单设计	187	5.5.7 自动翻页功能	234
4.6.2 ToolStripMenuItem 类	189	5.5.8 打印及打印预览功能.....	236
4.6.3 ToolStrip 控件与工具栏设计	191	5.5.9 关闭程序提示保存功能.....	237
4.6.4 StatusStrip 控件与状态栏设计	193	5.5.10 改造成 MDI 界面.....	238
4.6.5 对话框设计	194	5.6 习题	240
4.7 简单屏幕保护程序设计	196	第 6 章 图形操作	241
4.7.1 文字动态显示效果实现	196	6.1 图形操作概述	241
4.7.2 由按键或鼠标移动结束程序	197	6.1.1 GDI+	241
4.7.3 增加密码判断功能	198	6.1.2 图形操作基本知识.....	242
4.7.4 隐藏屏幕保护程序的光标	200	6.2 GDI+图形操作类和结构	242
4.8 习题	201	6.2.1 Point、Size 和 Rectangle 结构.....	242
第 5 章 文件操作	204	6.2.2 Pen 类	244
5.1 文件操作概述	204	6.2.3 Brush 类及其派生类	246
5.1.1 文件概念与文件类型	204	6.2.4 Image、Bitmap 类.....	248
5.1.2 System.IO 模型	205	6.2.5 GraphicsPath 类.....	251
5.1.3 Stream 类.....	206	6.2.6 Region 类.....	252
5.2 文件与目录操作	208	6.3 Graphics 类与图形绘制	253
5.2.1 目录操作	208	6.3.1 创建 Graphics 对象	253
5.2.2 文件常规操作	210	6.3.2 Matrix 类与坐标变换.....	253
		6.3.3 Graphics 类的基本属性和方法	255
		6.3.4 绘制基本图形	256
		6.3.5 填充基本图形	260
		6.3.6 绘制文字	261

6.3.7 绘制图像.....	262	7.4.3 读取对象 OleDbDataReader.....	290
6.3.8 裁剪区域.....	264	7.4.4 数据适配器对象 OleDbDataAdapter.....	291
6.3.9 图形绘制的其他说明.....	265	7.4.5 参数对象 OleDbParameter.....	293
6.4 电子闹钟程序设计.....	266	7.5 DataSet 对象及其相关对象.....	294
6.4.1 电子闹钟绘制与时间显示.....	266	7.5.1 DataSet 对象.....	294
6.4.2 完善电子闹钟程序.....	269	7.5.2 DataTable 对象.....	295
6.4.3 以文字形式显示的电子闹钟程序.....	271	7.5.3 DataColumn 对象.....	296
6.5 习题.....	272	7.5.4 DataRow 对象.....	297
第 7 章 ADO.NET 与数据库操作... 274		7.5.5 DataView 对象.....	298
7.1 数据库基础知识.....	274	7.5.6 Constraint 类.....	299
7.1.1 数据库基本概念.....	274	7.5.7 DataRelation 类.....	300
7.1.2 ODBC 数据源.....	276	7.6 数据显示.....	301
7.1.3 Access 数据库.....	277	7.6.1 数据绑定.....	301
7.2 结构化查询语言 SQL.....	279	7.6.2 DataGridView 控件.....	305
7.2.1 SQL 概述.....	279	7.7 学生信息管理程序设计.....	308
7.2.2 SQL 主要语句.....	279	7.7.1 数据表定义.....	308
7.3 ADO.NET 概述.....	282	7.7.2 学生信息维护功能.....	309
7.3.1 ADO.NET 简介.....	282	7.7.3 数据查询与统计功能.....	311
7.3.2 ADO.NET 数据库访问体系结构.....	283	7.8 习题.....	313
7.3.3 ADO.NET 数据库访问流程.....	284	附录 上机练习.....	316
7.4 ADO.NET 数据库访问对象.....	285	参考文献.....	320
7.4.1 连接对象 OleDbConnection.....	285		
7.4.2 命令对象 OleDbCommand.....	288		

第 1 章

C#概述

本章主要介绍 C#语言的发展过程以及 Visual C#编程环境。通过用 C#编写一个简单的控制台程序，读者可以了解 C#程序的基本结构和编程方法，为后续章节的学习以及编程练习奠定基础。

1.1 .NET Framework 开发平台

2000 年 6 月，微软公司推出了新一代的程序开发平台——Microsoft.NET（以下简称.NET）。借助.NET 平台，程序员可以创建 Windows 应用程序、Web 站点以及 Web 服务。.NET 是一种面向网络的开发环境，特别是对网络程序的开发与发布提供了强大的支持；实现了普通 Windows 应用程序与 Web 程序统一的开发平台。

为了实现微软的战略目标，.NET 的编程模式将开发语言与运行平台分离，以实现独立于语言的组件技术。通过不同的运行平台，.NET 语言可以被扩展到个人计算机、PDA、手机和嵌入式设备上。

.NET Framework（.NET 框架）是微软.NET 战略的核心，这个框架执行 Windows 应用程序和 Web 程序，包括.NET 框架类库、公共语言运行时（Common Language Runtime，CLR）。

.NET 框架是生成、部署和运行 Web 服务及应用程序的平台。它提供了一个生产率高且基于标准的多语言环境，用于将现有投资与下一代应用程序和服务集成，同时提供了解决 Internet 应用程序的部署和操作的灵活性。

.NET 框架的主要目标如下。

- （1）提供一个一致的面向对象的编程环境。编写的代码既可以在本地存储和执行，也可以在远程执行，还可以发布在网络上。
- （2）提供一个将软件部署和版本控制冲突最小化的代码执行环境。
- （3）提供一个可提高安全性的代码执行环境。
- （4）使得开发人员在面向 Windows 应用程序和 Web 应用程序时保持一致。
- （5）可以保证基于.NET 框架的代码与任何其他代码集成。

由此，可以看出，.NET 框架不仅仅是个程序开发平台，还是程序的执行平台。也就是说，利用.NET 框架开发的任何应用程序都必须在.NET 框架下被执行。

.NET 框架最基础的是公共语言运行时 CLR，可以将 CLR 看作一个在执行代码时的代理。它的作用是负责执行程序，提供内存管理、线程管理、安全管理、异常管理、生命周期监控等核心服务。其主要功能包括如下几点。

- 代码管理（加载和执行代码）。

- 应用程序内存隔离。
- 类型安全验证。
- 微软中间语言 MSIL 到本机代码的转换。
- 元数据（增强的类型信息）访问。
- 为托管对象管理内存。
- 强制代码访问安全。
- 异常处理，包括跨语言异常。
- 托管代码、COM 对象和现有 DLL（非托管代码和数据）之间的互操作。
- 自动进行对象布局。
- 对开发人员服务（配置、调试等）的支持。

CLR 所执行的程序代码称为托管代码，托管代码产生的步骤如下。

（1）用符合 .NET 规范的程序设计语言（如 C#.NET、VB.NET）编写托管代码的源程序。

（2）将托管代码源程序编译成微软中间语言 MSIL。这些中间语言的文件扩展名也是 .exe，但是与采用传统的 VB 或 VC 所编写的可执行文件不同，它们并非机器语言。

（3）执行程序时，由 .NET 框架的 JIT（Just In Time）编译器将中间语言编译成真正的机器码。

自动内存管理是 CLR 在托管执行过程中提供的服务之一。CLR 的垃圾回收器为应用程序管理内存的分配与释放。对于开发人员而言，在开发托管程序时不必编写执行内存管理任务的代码。自动内存管理可以解决程序中常见的内存问题，如忘记释放对象导致的内存泄露，或者尝试访问已经释放对象的内存等。

.NET 框架的另外一个主要组件是类库。它是一个综合性的面向对象的可重用类型集合，可以使用它开发从传统的命令行（控制台程序）或图形用户界面应用程序（Windows 窗口程序）到包括基于 ASP.NET 的新型应用程序（Web 窗体和 XML Web 服务）在内的各种应用程序。利用 .NET 框架可以开发下列各种应用程序。

- 控制台应用程序。
- 脚本应用程序。
- Windows 窗体程序。
- ASP.NET 应用程序（Web 窗体）。
- XML Web Services。
- Windows 服务。

.NET 框架支持多种编程语言，如 C#、Visual Basic、VC++ 等。虽然这些语言的语法不同，但编译成的中间语言是一致的。这些语言的数据类型、垃圾处理、异常处理都是在 CLR 层面实现的，与具体语言无关。

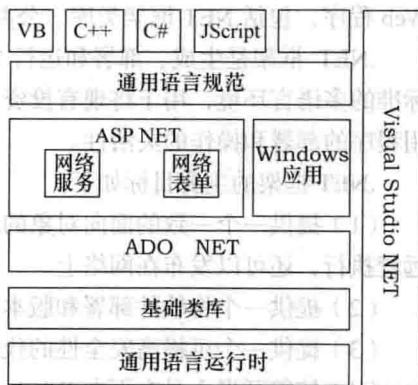


图 1-1 .NET 框架体系结构

.NET 框架的体系结构如图 1-1 所示。

随着 .NET 技术的不断发展，.NET 框架也从 .NET Framework 1.0 发展到最新的 .NET Framework 4.5。 .NET 框架的程序开发工具是 Visual Studio.NET，其最新版本是 2012 年推出的 Visual Studio 2012，它的 .NET 框架版本为 4.5（需要 Windows 7 及以上版本操作系统）。目前，广泛使用的 Visual Studio 版本还有 Visual Studio 2003（.NET Framework 1.1）、Visual Studio 2005（.NET Framework 2.0）、Visual Studio 2008（.NET Framework 3.5）和 Visual Studio 2010（.NET Framework 4.0）。本教材采用 Visual Studio 2013 作为开发运行环境。

1.2 C#及其与 C、C++的区别

C#（读作 C Sharp）是一种新的编程语言，属于 Microsoft Visual Studio.NET 的一部分，是编写下一代窗口应用程序和 Web 应用程序的主要语言。C#是.NET 的核心开发语言，是专门为.NET 开发而量身定制的语言，因而是.NET 开发的最佳语言。C#是一种简单、类型安全、面向对象的编程语言，其语言风格源于 C/C++语言，语法与思维方式与 Java 语言相似，兼容了 Visual Basic 语言的易用性和 C/C++语言的高执行效率。

C#继承了 C++语言的大多数语法，去掉了 C++中比较难以理解和容易出错的部分，并借鉴了 Java 语言的许多特性，使其变得易于使用，不易出错，是升华了的 C/C++。它几乎综合了目前所有高级语言的优点。

C#语言的主要特点如下。

（1）简单性。

C#去掉了 C/C++的指针、类的多重继承、宏、模板等容易出错的语法，规范了数据类型标准。

指针是 C/C++语言的重要特征，提供了直接操作内存的机制，如果使用不当，很容易出错。特别是动态申请的内存，如果不释放，就会造成严重的内存泄露。C#使用引用代替指针，可以避免在访问不允许访问的内存和指针时产生误操作，内存的释放由.NET 的垃圾回收器处理，程序员不必担心内存泄露。对于类的实例化，采用引用方式，其成员的访问采用“.”操作符，避免了 C/C++中“.”和“->”的混淆。即使是 C/C++的函数指针，虽然 C#不再支持，但是可以通过托管实现，其安全性比 C/C++高。

针对 C/C++的类继承，C#仅仅支持单一继承，避免了多重继承带来的混乱问题。对于需要多重继承的情况，C#可以通过接口实现。

C/C++的宏虽然增加了代码修改的灵活性，但其定义格式一旦出错，会导致编译预处理时所替换的 C/C++语法错误。

C/C++模板的使用可以实现不同数据类型同一功能的实现统一化，是实现代码重用的一项重要技术。但是模板涉及功能过于复杂，C#通过泛型仅仅实现了 C++模板中最基本的功能，简单实用。

C++语言是从 C 发展而来的，最早应用于 16 位的操作系统 DOS 平台下，其数据类型占用字节数与 C 语言一致。例如，整数类型 int 占用 2 个字节（进入 32 位的 Windows 操作系统后，为了适应 32 位的特点，int 类型变成了占用 4 个字节）。C#作为一个全新的语言，不必兼容以前的 C/C++，其数据类型占用字节数与操作系统和 CPU 无关。

C++中的布尔类型实际是一种特殊的整型，可以在两者之间轻易进行转换，在条件判断中，布尔类型往往直接采用整型代替（非 0 的整数为 true，0 为 false）。C#语言中布尔类型是一个独立的类型，是与整型完全不同的类型，两者之间不能转换，判断语句中必须用布尔类型，不能再用整型，这样就避免了比较与赋值的混淆。

C++中，if(x==1){} 与 if(i=1){} 都是合法的语句，但是作用不同。前者表示比较变量 x 是否等于 1，而后者先将 1 赋值给 i，再判断 i 是否不为 0，这是初学者非常容易弄错的地方。C#中只有第一条语句是合法的。

（2）面向对象。

C++虽然是面向对象编程的计算机语言，但是为了兼容 C 语言，仍然保留了 C 语法中部分结

结构化程序设计的特点,如函数、全局变量等。C#去掉了函数与全局变量,所有的数据及其操作都封装在类中,对数据的操作称为类的方法,数据和方法要么是作为类的实例成员(通过类的实例对象访问),要么是作为类的静态成员(通过类名称访问),C#是完全的面向对象编程语言。即使是 C/C++ 中用于整个程序启动的主函数 `main` 和 `WinMain` 函数也变成了类中的一个静态成员方法 `Main`。这样使得程序的可读性强,减少了潜在的命名冲突。

(3) 类型安全。

C#拥有强大的安全机制和完善的错误异常处理机制。C#实现了最严格的类型安全来保护其自身和垃圾回收器。

C/C++定义变量时一般不给予其赋初值,采用未赋初值的变量进行计算,会得到意想不到的结果。C#中的类成员变量由编译器自动给予其赋初值 0,局部变量虽然不能自动赋初值,但如果使用了未赋初值的变量,编译器会提醒。

C/C++对数组不提供越界检查,例如,定义了一个 10 个元素的数组 `a[0]` 到 `a[9]`,给第 11 个元素赋值 `a[10]`也是允许的。而实际的效果可能是给这个数组之后的变量进行了赋值,导致不正常的结果。C#进行数组边界检查,从而避免了这种现象。

C/C++对变量的溢出不做处理,如 `char` 类型的变量表示数据的范围是 -128 ~ 127,给予其赋值 200 也不会出错。但是,实际赋给这变量的值并非是 200 (已经超过了其表示范围),而是变成了 -56 (是把表示 200 的 8 位二进制数 11001000 理解为一个补码表示的负数)。C#则提供了变量的算术溢出检查,避免了这样的错误赋值。

(4) 兼容性。

C#遵守 .NET 通用语言规范 CLS,可以保证与其他语言开发的组件兼容,还可以提供特定的机制访问 COM 组件(这些组件往往是由不遵守 CLS 的 VC++ 等语言开发的)。

(5) 便捷的窗口设计方法。

Visual C++ 采用 MFC 创建 Windows 窗口应用程序,庞大而复杂的 MFC 类库是学习 Visual C++ 最困难的地方。C#采用与 Visual Basic 类似的窗口程序设计方法,通过继承 `Form` 类就可以创建窗口,并且将 Windows 的消息演变成了事件处理,只要编写相应的事件处理方法即可,不需要再利用 MFC 的消息映射模式。实际上,在 .NET 框架下,C#和 VB.NET 都是遵守通用语言规范 CLCS 的语言,除了语法不同外,两者编写 Windows 应用程序和 Web 应用程序的编程思路和方法几乎完全一样。可以说,C#很像是一个披着 C/C++ 语法“外衣”的 VB。

1.3 C#版本的发展

1998 年 12 月,微软公司启动了一个全新的语言项目——COOL,这是一款专门为 CLR 设计的纯面向对象的语言,也正是 C#的前身。历时半年有余,1999 年 7 月,微软完成了 COOL 语言的一个内部版本。直到 2000 年 2 月,微软才正式将 COOL 语言更名为 C#。又历经了一系列的修改,微软终于在 2000 年 7 月发布了 C#语言的第一个预览版。在此后的一年多时间里,微软一直在修补各个测试版本中的 BUG。直到 2002 年 2 月,微软终于推出了迟迟未上市的 Visual Studio 7.0,并将其定名为“Visual Studio 2002”。随着这套开发环境的出炉,开发者们终于看到了 C#语言的第一个正式版本——C# 1.0。在 2003 年 5 月,微软如期推出了 Visual Studio 2003,同时也发布了 C#的改进版本——C# 1.1。这一时期的 C#(以下称为 C# 1.x)提出了纯粹的面向对象的概念,在

C# 1.x 中,所有面向对象的概念都在语言中得到了非常好的体现。同时,C#还通过类类型、值类型和接口类型的概念形成了统一的类型系统。C#使用了大家所熟知的语法实现了方法,以至于很多人认为 C#和 Java、C 等面向对象语言“非常相像”。此外,C#还通过无参数列表的方法声名语法,结合 get/set 访问器实现了类的属性语法。通过委托,结合关键字 event 和订阅,C#提供了事件概念以及对象的事件处理机制。

微软在 2004 年 6 月发布了 Visual Studio 2005 的第一个 Beta 版,同时向开发者展示了 C#语言的 2.0 版本。C# 2.0 为开发者带来的最主要的特性就是泛型编程能力。C# 2.0 的另一个突出的特性就是匿名方法,用来取代一些短小的并且仅出现一次的委托,使得语言结构更加紧凑。在 C# 2.0 中,属性语法中的 get 和 set 访问器可以拥有不同的权限,这就使得定义一个在类内部可读写而在类外部只读的属性成为可能。同时,C# 2.0 还提供了迭代器的概念,这使得一个类无须实现 IEnumerator 和 IEnumerable 接口即可实现一个可以进行遍历的类型,并且无须在类型中维护迭代状态。

2007 年 8 月 20 日,微软继发布了 .NET Framework 3.0 后,推出了 C# 3.0 语言规范。C# 3.0 增加了 Lambda 表达式和 LINQ 语言集成查询、隐式类型本地变量、对象和集合初始化器、匿名类型、扩展方法、宽松委托、自动实现属性、分部方法等新的特性。2007 年 11 月 19 日,微软发布了 Visual Studio 2008 和 .NET Framework 3.5,其中的 C#采用 3.0 规范。

2010 年 4 月 12 日,微软发布了 Visual Studio 2010 和 .NET Framework 4.0,其中的 C#采用 4.0 规范。C# 4.0 新增加了动态编程、命名参数和可选参数、特定于 COM 的互操作特性等新特性。

2012 年 9 月 12 日,微软发布了 Visual Studio 2012 和 .NET Framework 4.5,可以用于开发 Windows 8 应用程序。2013 年 11 月 13 日,微软发布了最新的 Visual Studio 2013 和 .NET Framework 4.5.1 预览版。

1.4 Visual Studio 2013 集成开发环境

C#语言是 Visual Studio 中的一种语言,其开发环境与其他 .NET 框架语言一起集成在 Visual Studio .NET 开发环境中,在其中称为 Visual C#。

启动 Microsoft Visual Studio 2013 后,出现如图 1-2 所示的主界面(起始页)。

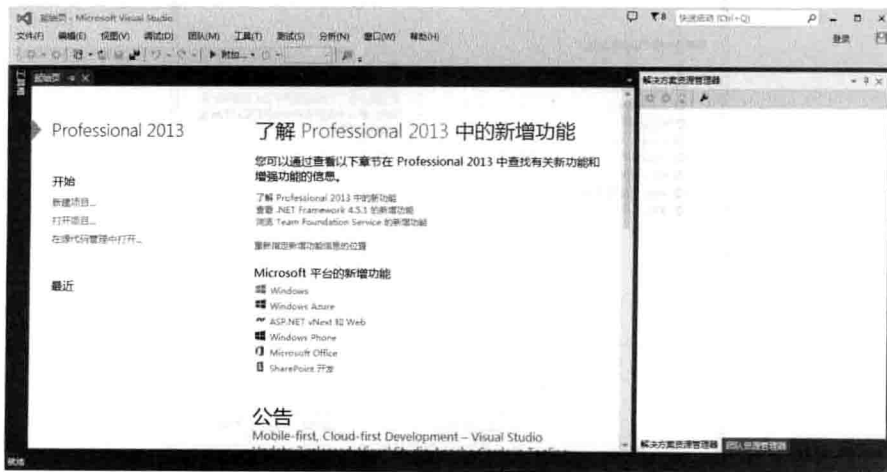


图 1-2 Microsoft Visual Studio 2013 起始页

起始页左侧显示最近打开的项目，单击这些项目就可以打开；也可以单击“打开”或“新建”后面的“项目”选择其他项目打开或新建项目。左侧下部列出了与当前默认语言有关介绍（图 1-2 中已经选择 Visual C# 作为默认语言）。起始页中部显示 Visual Studio 2013 新增功能内容。右侧为“解决方案资源管理器”。

安装 Microsoft Visual Studio 2013 后第一次运行时，会出现“选择默认环境设置”对话框以选择默认语言。如果希望改变 Microsoft Visual Studio 2013 的默认语言，可以选择“工具”菜单下的“导入和导出设置”，弹出如图 1-3 所示的向导窗口。

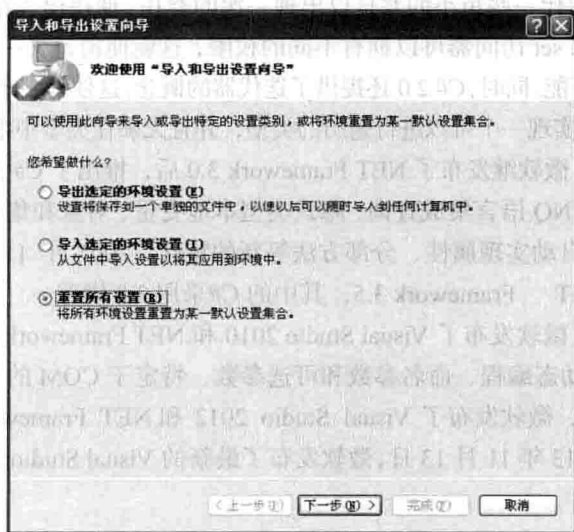


图 1-3 “导入和导出设置向导”窗口

选择图 1-3 中的“重置所有设置”，再单击“下一步”按钮，在弹出的窗口中选择是否保存当前设置，再单击“下一步”按钮，弹出如图 1-4 所示的窗口，选择其中的“Visual C#”，单击“完成”按钮，即可将 Visual C# 设置为 Microsoft Visual Studio 2013 的默认语言。

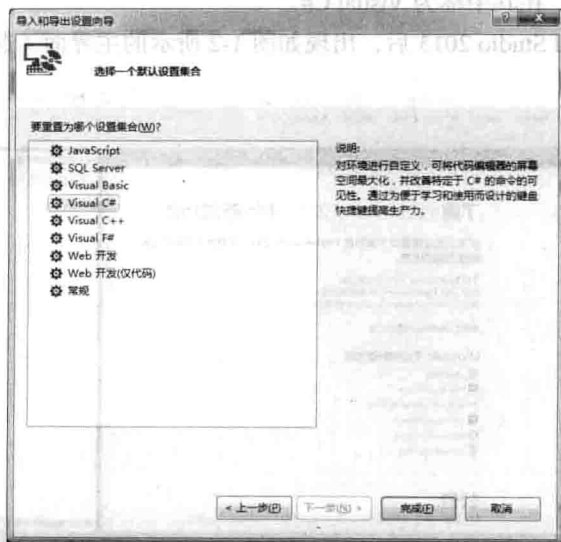


图 1-4 默认语言选择窗口

Microsoft Visual Studio 2013 主界面的许多视图都是可以自动隐藏的。例如，图 1-2 所示最左侧就隐藏了“工具箱”。如果某个视图右上角中间按钮显示是 ，表示正常显示视图，如果是 ，表示自动隐藏，可以通过单击该按钮切换两种显示方式。

用鼠标可以将某个视图拖动到任意位置。如果希望将某个视图重新停靠到某个固定位置，在视图窗口标题栏单击鼠标右键，从弹出窗口中选择“可停靠”，然后拖动窗口，出现如图 1-5 所示的停靠位置箭头，将视图拖动到某个箭头处，就可以实现视图窗口的停靠。

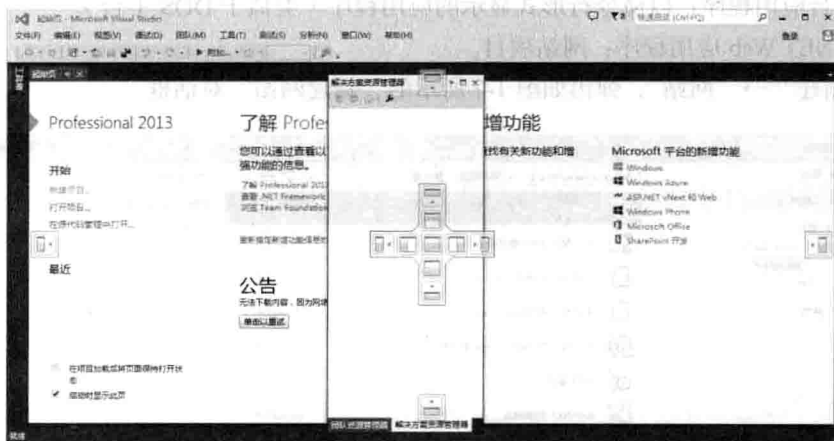


图 1-5 停靠箭头

Microsoft Visual Studio 2013 主界面窗口的菜单随着用户选择的项目类型不同而不同。这里先介绍几个常用的通用菜单项。

(1) “文件”菜单。

包含了若干与文件或项目有关的基本操作。“新建”用于创建新的项目、网站、文件，以及利用已有文件创建项目等。

选择“新建”→“项目”，弹出如图 1-6 所示的“新建项目”对话框。

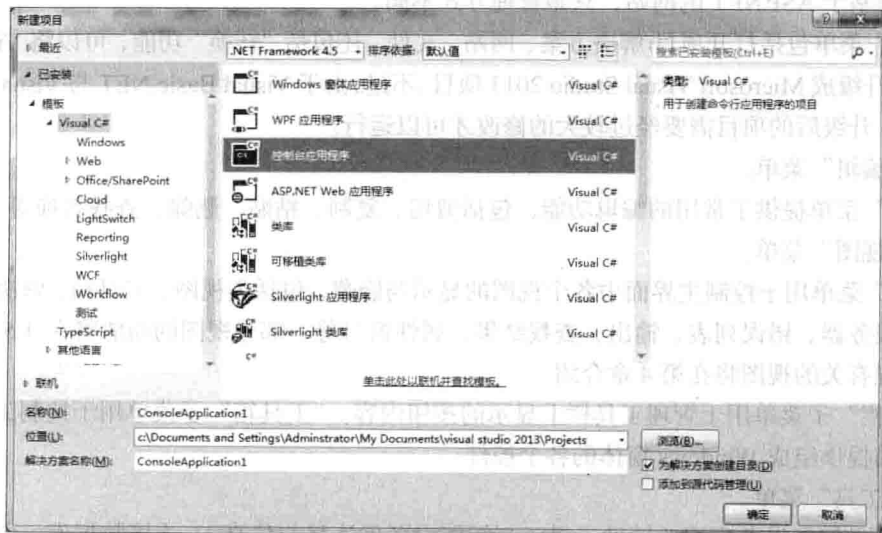


图 1-6 “新建项目”对话框

对话框中列出了已经安装过的不同语言项目模板。需要从这里选择合适的编程语言和应用程序模板,并选择项目存储的文件夹路径和项目文件名称,以及解决方案名称。解决方案是 Microsoft Visual Studio 2013 进行多项目管理的方案。同一个解决方案中可以创建或打开多个项目。在图 1-6 的右上角还可以选择 .NET Framework 的版本。

常用的项目模板如下。

- Windows 窗体应用程序: 创建窗口形式的 Windows 应用程序。
- 控制台应用程序: 以命令行形式显示的应用程序(类似于 DOS 平台)。
- ASP.NET Web 应用程序: 网站项目。

选择“新建”→“网站”,弹出如图 1-7 所示的“新建网站”对话框。

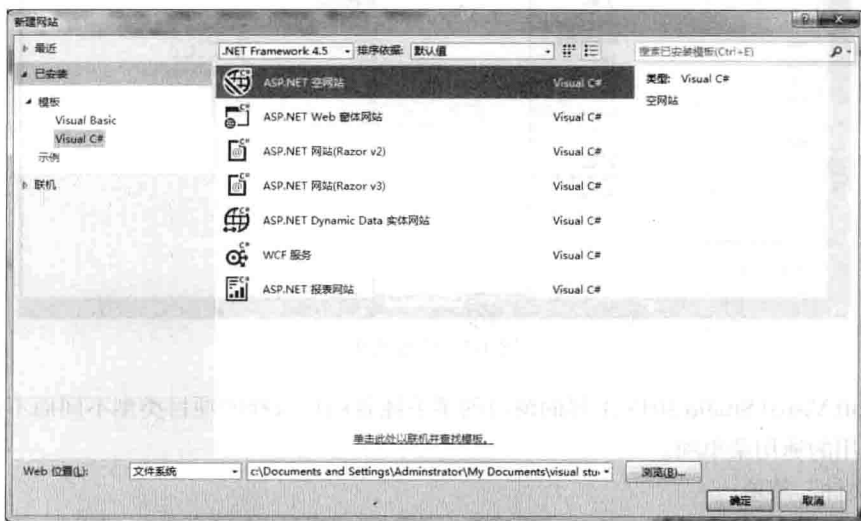


图 1-7 “新建网站”对话框

这里一般选择“ASP.NET 空网站”,该模板与新建项目中的“ASP.NET Web 应用程序”都可以用于创建基于 ASP.NET 的网站,只是管理方式不同。

“打开”菜单包括打开项目/解决方案、网站、文件。还包括“转换”功能,可以将 Visual Basic 6.0 的项目升级成 Microsoft Visual Studio 2013 项目。不过,由于 Visual Basic.NET 与 Visual Basic 6.0 差异很大,升级后的项目需要经过较大的修改才可以运行。

(2) “编辑”菜单。

“编辑”菜单提供了常用的编辑功能,包括剪切、复制、粘贴、删除、查找替换等。

(3) “视图”菜单。

“视图”菜单用于控制主界面中各个视图的显示与隐藏,包括类视图、工具箱、解决方案资源管理器、服务器、错误列表、输出、查找结果、属性窗口等,部分视图的功能将在 1.5 节介绍,与窗体设置有关的视图将在第 4 章介绍。

“工具栏”子菜单用于管理工具栏上显示的按钮内容。“工具箱”子菜单用于控制工具箱的显示,工具箱提供组成 Windows 窗体的各个控件。

(4) “工具”菜单。

提供开发智能设备程序的辅助工具(与智能设备的连接与仿真)、连接数据库、创建组件的 GUID(全局唯一标识符)、集成开发环境的选项设置等。

“选项”子菜单用于设置集成开发环境的外观和行为，包括环境设置、项目与解决方案设置、文本编辑器设置等，设置窗口如图 1-8 所示。

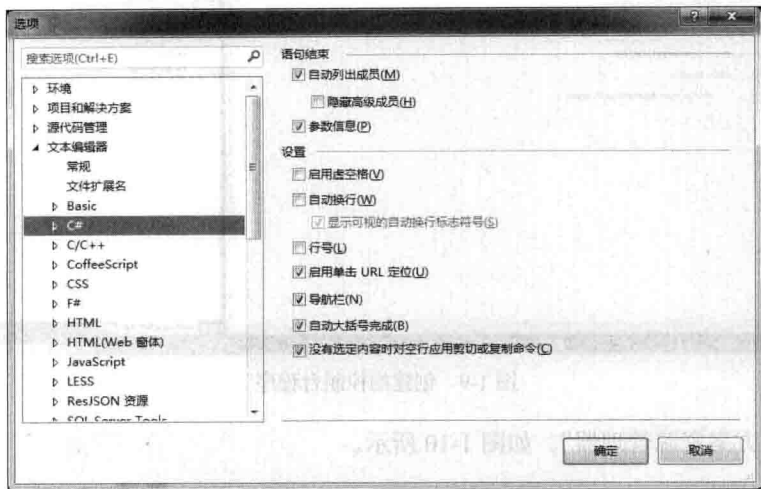


图 1-8 “选项”对话框

图 1-8 中列出了有关 C#代码编辑的选项。

1.5 第一个简单的 C#应用程序

控制台程序是 C#所能够创建项目中最简单的程序，它以命令行窗口形式显示程序运行的结果。前 3 章介绍 C#语法部分的示例程序就以控制台程序为主。

1.5.1 新建控制台程序

选择“文件”菜单的“新建”→“项目”，在弹出的对话框中选择 Visual C++的“控制台应用程序”，默认情况下，新创建的项目存放在“我的文档”下面的子文件夹“Visual Studio 2013\Projects”中，单击“浏览”可以选择其他存放位置。

新建的控制台程序的项目名称和解决方案名称默认为“ConsoleApplication1”，虽然两者名称相同，但是在保存时项目名称与解决方案名称采用不同的扩展名。

选中“创建解决方案的目录”，则在上述文件夹下会创建解决方案名称的子文件夹。保持默认值不变，单击“确定”，则会创建一个控制台程序，如图 1-9 所示。

这时，整个 Visual Studio 2013 的主界面中间部分增加了选项卡，其标题为“Program.cs”，在该选项卡下，显示出新创建的控制台程序的源程序，这是 Visual C#的代码编辑器。

Visual C#代码编辑器中用于编辑 C#代码，与 Word 编辑器类似，在代码编辑器中，可以简单地利用拖放或“Ctrl+拖放”实现选中代码的移动与复制。

集成开发环境的工具栏的“Debug”组合框，表示当前项目的配置。

主界面右侧上方的“解决方案资源管理器”中以树形结构显示出新创建项目的组成部分，右侧下方显示“属性”视图。“属性”视图用于显示当前所选择内容的有关属性。