高等学校计算机教材

C#

实用教程

郑阿奇 主编 梁敬东 钱晓军 朱毅华 时跃华 赵青松 编著

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http: //www.phei.com.cn

高等学校计算机教材

C#实用教程

郑阿奇　主编

梁敬东　钱晓军　朱毅华　时跃华　赵青松　编著

電子工業出版社

Publishing House of Electronics Industry

北京・BEIJING

内 容 简 介

C#是目前主流的程序设计语言之一，本书以 Microsoft Visual Studio 2005 作为平台，系统地介绍 C#编程基础、面向对象编程、Windows 应用程序、GDI+编程、文件操作、数据库应用、多线程技术和Web应用程序。本书包含实用教程、习题、实验和综合应用实习四部分。习题主要突出基本编程和基本概念；实验主要锻炼学生编程和应用的能力，读者先跟着做，然后自己练习。综合应用实习突出 C#的主要应用，实习 1 为 Windows 应用程序开发，实习 2 为 ASP.NET 应用程序开发。本书配有教学课件和应用实例源文件。

本书可作为大学本科、高职高专有关专业 C#课程教材，也可供广大 C#开发用户参考。

图书在版编目（CIP）数据

C#实用教程/郑阿奇主编.—北京：电子工业出版社，2008.1
高等学校计算机教材
ISBN 978-7-121-05478-5

Ⅰ.C… Ⅱ.郑… Ⅲ.C 语言—程序设计—高等学校—教材 Ⅳ.TP312

中国版本图书馆 CIP 数据核字（2007）第 184042 号

策划编辑：童占梅
责任编辑：童占梅　秦淑灵
印　　刷：北京市顺义兴华印刷厂
装　　订：三河市双峰印刷装订有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编　100036
开　　本：787×1092　1/16　印张：24.25　字数：600 千字
印　　次：2008 年 1 月第 1 次印刷
印　　数：3 000 册　定价：34.80 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系。联系及邮购电话：（010）88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：（010）88258888。

前　言

C#是.NET 平台为应用开发而全新设计的一种现代编程语言，除了 Windows 基本功能外，它在用户交互界面、Web 应用、数据库应用等方面功能更强，深受大家欢迎。

本书以 Microsoft Visual Studio 2005 作为操作平台，包含**实用教程、习题、实验和综合应用实习**四部分，可以**满足 C#课程的所有教学环节**。

实用教程首先通过一个简单实例介绍操作过程，然后比较系统地介绍 C#的编程基础、面向对象编程，系统介绍 Windows 应用程序、GDI+编程、文件操作、数据库应用、多线程技术和Web 应用程序。

习题主要突出基本编程和基本概念。

实验主要锻炼学生的编程和应用能力，读者先跟着做，然后自己练习。

综合应用实习突出 C#的主要应用，实习 1 为 Windows 应用程序开发，实习 2 为 ASP.NET 应用程序开发。

一般来说，在较短的时间内，通过实用教程学习、习题练习，特别是认真的上机和综合应用实习后，学生基本能够应用 C#解决一些小的应用问题。

本书**配有教学课件**和**应用实例的源文件**，教师可用于辅助教学，学生可用于模仿和修改。需要者可免费在华信教育资源网 http://www.hxedu.com.cn 或 http://www.huaxin.edu.cn 注册下载。

本书由南京农业大学梁敬东、钱晓军、朱毅华、时跃华和赵青松编写，由南京师范大学郑阿奇统编和定稿，马亮做了大量的工作。

参加本套丛书编写的有郑阿奇、梁敬东、顾韵华、王洪元、杨长春、姜乃松、刘启芬、殷红先、彭作民、丁有和、徐文胜、曹弋、张为民、郑进、王一莉、刘毅、周怡君等。

由于作者水平有限，不当之处在所难免，恳请读者批评指正。

编　者

目　录

第 1 部分　实用教程

第 2 部分 习题

第 3 部分　实验

第 4 部分　综合应用实习

第1部分 实用教程

第1章 C#快速入门

C#是.NET 平台为应用开发而全新设计的一种现代编程语言，随着微软.NET 战略进入开发人员的视野，C#很快成为 Windows 应用开发语言中的宠儿。本章提供给读者 C#快速入门的指导，帮助读者对 C#有一个准确的定位，并能对自己的学习做好安排。

1.1 C#的优势

作为编程语言，C#是现代的、简单的、完全面向对象的，而且是类型安全的。重要的是，C#是一种现代编程语言。在类、名字空间、方法重载和异常处理等方面，C#去掉了 C++中的许多复杂性，借鉴和修改了 Java 的许多特性，使其更加易于使用，不易出错。

下面列举了一些 C#在设计上的优点：

（1）简单性

C#在设计上的首要目标就是简单性。

没有指针是 C#的一个显著特性。在默认情况下，用户使用一种可操控的（Managed）代码进行工作时，一些不安全的操作，如直接的内存存取，将是不允许的。

在 C#中不再需要记住那些源于不同处理器结构的数据类型，如可变长的整数类型，C#在 CLR 层面统一了数据类型，使得.NET 上的不同语言具有相同的类型系统。可以将每种类型看作一个对象，不管它是初始数据类型还是完全的类。

整型和布尔数据类型是完全不同的类型。这意味着 if 判别式的结果只能是布尔数据类型，如果是别的类型则编译器会报错。那种搞混了比较和赋值运算的错误不会再发生。

（2）现代性

许多在传统语言中必须由用户自己来实现的或者干脆没有的特征，都成为基础 C#实现的一个部分。金融类型对于企业级编程语言来说是很受欢迎的一个附加类型。用户可以使用一个新的 decimal 数据类型进行货币计算。

安全性是现代应用的头等要求，C#通过代码访问安全机制来保证安全性，根据代码的身份来源，可以分为不同的安全级别，不同级别的代码在被调用时会受到不同的限制。

（3）面向对象

C#支持面向对象的所有关键概念：封装、继承和多态性。整个 C#的类模型是建立在.NET 虚拟对象系统（VOS，Virtual Object System）之上的，这个对象模型是基础架构的一部分，而不再是编程语言的一部分——它们是跨语言的。

C#中没有全局函数、变量或常数。每样东西必须封装在一个类中，或者作为一个实例成

员（通过类的一个实例对象来访问），或者作为一个静态成员（通过类型来访问），这会使用户的C#代码具有更好的可读性，并且减少了发生命名冲突的可能性。

多重继承的优劣一直是面向对象领域争论的话题之一，然而在实际的开发中很少用到，在多数情况下，从多个基类派生所带来的问题比这种做法所能解决的问题要更多，因此C#的继承机制只允许一个基类。如果需要多重继承，用户可以使用接口。

（4）类型安全性

当用户在C/C++中定义了一个指针后，就可以自由地把它指向任意一个类型，包括做一些相当危险的事，如将一个整型指针指向双精度型数据。只要内存支持这一操作，它就会凑合着工作，这当然不是用户所设想的企业级编程语言类型的安全性。与此相反，C#实施了最严格的类型安全机制来保护它自身及其垃圾收集器。因此，程序员必须遵守关于变量的一些规定，如不能使用未初始化的变量。对于对象的成员变量，编译器负责将它们置零。局部变量用户应自己负责。如果使用了未经初始化的变量，编译器会提醒用户。这样做的好处是：用户可以摆脱因使用未初始化变量得到一个可笑结果的错误。

边界检查。当数组实际上只有 n–1 个元素时，不可能访问到它“额外”的数组元素 n，这使重写未经分配的内存成为不可能。

算术运算溢出检查。C#允许在应用级或语句级检查这类操作中的溢出，当溢出发生时会出现一个异常。

C#中传递的引用参数是类型安全的。

（5）版本处理技术

在过去的几年中，几乎所有的程序员都和所谓的“DLL 地狱”打过交道，产生这个问题是因为许多计算机上安装了同一 DLL 的不同版本。DLL 是 Dynamic Link Library 的缩写，是一种编译为二进制机器代码的函数库。DLL 在调用程序运行时才被调入内存执行，而不是在编译时链接到可执行程序内部，这样可以使程序代码在二进制级别实现共享，而不必在每个应用程序中编译一个副本，如果 DLL 中的代码更新了，只需要替换 DLL 文件即可更新所有使用该 DLL 的程序。然而这同时带来了 DLL 文件版本的问题，不同版本的 DLL 可能与不同调用程序不兼容，同一版本 DLL 也不能同时为不同的调用程序服务，结果造成应用程序出现无法预料的错误，或者在用户计算机中不得不更换文件名来保存同一 DLL 的多个版本。这种混乱的状态被称为“DLL 地狱”。C#则尽其所能支持这种版本处理功能，虽然 C#自己并不能保证提供正确的版本处理结果，但它为程序员提供了这种版本处理的可能性。有了这个适当的支持，开发者可以确保当他开发的类库升级时，会与已有的客户应用保持二进制级别上的兼容性。

1.2 第一个 C#程序

让我们从经典的“Hello World!”程序开始 C#之旅。这里将给出两个版本的“Hello World!”代码，一个使用标准的控制台输出信息，另一个使用 Windows 消息对话框输出信息，对应的模板是“控制台应用程序”和“Windows 应用程序”。读者可以发现，基于.NET 框架的强大的类库在进行应用开发时是多么简单快捷。

首先看控制台应用程序的版本。

【例 1.1】 在控制台窗口中输出“Hello World!”字样。

在 Visual C#.NET 开发环境中新建一个控制台应用程序项目，并在源代码文件中输入如下语句：

```
using System;
class HelloWorld
{
    public static void Main()
    {
        Console.WriteLine("Hello World!");
    }
}
```

别忘了将此内容保存到文件 EX1_1.cs 中，然后选择菜单“调试”→“启动”或直接按 F5 键运行此程序。可以看到运行结果出现在控制台窗口，并且在窗口中显示出“Hello World!”字样。如果读者的计算机速度较快，这个窗口会一闪而过，因为程序运行后就退出了。

读者也可以不使用 Visual C#.NET 环境，直接用文本编辑工具输入以上代码，并且保存到 Helloworld.cs 文件中，再通过开始菜单的“程序”→“Microsoft Visual Studio 2005”→“Visual Studio Tools”→“Visual Studio 2005 命令提示”打开命令行窗口，在命令窗口中输入：

```
csc /target:exe EX1_1.cs
```

以上假设读者已经切换到 EX1_1.cs 源文件所在的目录中。编译器编译该程序后，就可以输入“EX1_1”来运行该程序了。程序运行结果如图 1.1 所示。

```
Hello World!
```

图 1.1 控制台版本程序运行结果

下面再看 Windows 应用程序的版本。

【例 1.2】 弹出一个消息对话框显示“Hello World!”字样。

用“Windows 应用程序”模板建立项目，或者使用文本编辑工具输入源代码如下：

```
using System;
using System.Windows.Forms;
class HelloWorld
{
    public static void Main()
    {
        MessageBox.Show("Hello World","Message from C#");
    }
}
```

这次需要增加对 System.Windows.Forms 命名空间的引用，选择菜单“项目”→“添加

引用...”打开“添加引用”对话框，在列表中找到并选中“System.Windows.Forms.dll”，然后单击“选择”按钮，最后单击“确定”按钮完成添加。将文件另存为 EX1_2.cs，在开发环境中按 F5 键编译运行程序，如果用命令行方式编译请参照控制台版，编译命令如下：

```
csc /target:winexe EX1_2.cs
```

程序的运行结果如图 1.2 所示。

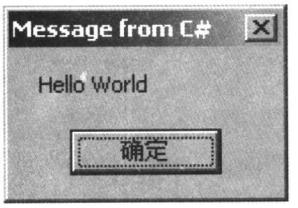

图 1.2　消息对话框版本程序运行结果

下面通过上述两段代码来认识 C#。

代码最前面是以 using 关键字开始的命名空间导入语句，然后是使用 class 关键字对类 HelloWorld 的定义。

命名空间是为了防止相同名字的不同标识符发生冲突而设计的隔离机制。比如，读者开发了一个二维的图形组件，将类命名为 Point，而他的同事开发的一个三维图形组件恰好也命名为 Point，用户需要在应用程序中使用所开发的组件，在编译时编译器将无法判断引入哪一个版本的组件，通过将类的命名放在不同的命名空间中就可以加以区别，要使用哪一个类，通过 using 关键字打开其所在的命名空间即可。在 C#中（确切地说，是在.NET 框架类库中）使用了一种树状的类似于“中国→江苏→南京”这样的地址编码方式来对命名空间进行管理，通过引入命名空间 MyClass 和 YourClass 就可以用 MyClass.Point 和 YourClass.Point 这样的方式对相同名称的标识进行识别了，即使是同时使用这两个组件，编译器也不会迷惑。

在.NET 框架类库中提供的不同组件都被包含在一定的命名空间中，所以要使用这些组件也必须通过 using 关键字开放相应的命名空间，使得相应的标识符对编译器可见，如果没有使用 using 关键字，那么相应的标识符就应包含完整的空间路径。

在 C#程序中，源代码块包含在花括号{ }中，因此，即使读者以前没有使用 C++或 Java 编程的经验，也可以判断出类的内容包括在最外面的花括号中，其中 Main 函数是类的方法，方法的代码又包括在里面的花括号中。

由于 C#是一种完全的面向对象的语言，所以不会有独立于类的代码出现，应用程序的入口也必须是类的方法，C#规定命名为 Main 的方法作为程序的入口（注意，与 C/C++语言不同，Main 的第一个字母是大写的，而且 **C#是一种大小写敏感的语言**，所以不要写错了）。public、static 关键字是对方法的修饰，其含义在后续章节中将详细介绍，我们要知道的是，public 使得这个方法能被外部访问，（不然程序如何被操作系统执行？）static 使得这个方法在类的实例被建立之前就可以调用（在程序入口的时候自然还不会有任何类的实例生成）。Main 前面的关键字 void 代表该方法没有返回值，这与 C/C++和 Java 是一样的。

方法中的代码“Console.WriteLine("Hello World!");”和“MessageBox.Show("Hello World", "Message from C#");”是调用.NET 框架类库中对象的方法来向控制台或消息对话框输出信息。可以看出，本程序的**核心代码所实现的功能全部来自.NET 框架类库，而 C#只是提供了一个语法框架**，所以这里要再次强调学习.NET 类库的重要性，C#开发实际上是用 C#语言将.NET 框架类库中的组件加以组织，实现应用程序的业务逻辑。

到这里读者已经跟随我们进入了 C#编程的大门，并且亲自实现了一个程序，通过本章的学习，读者应该掌握.NET 的基本架构及 C#与.NET 之间的关系，并且对 C#程序的结构有一个大概的认识。正确理解.NET 与 C#的关系对以后的学习非常重要，因为 C#开发的很多方面与.NET 分不开。

第 2 章　C#编程基础

C#的基本数据类型、变量、常量、表达式、程序流程控制语句及数组等概念是 C#程序设计的基础，掌握这些基本知识是编写正确程序的前提。

2.1　基本类型

C#语言是一种强类型语言，在程序中用到的变量、表达式和数值等都必须有类型，编译器检查所有数据类型操作的合法性，非法数据类型操作编译时不能通过。这种特点保证了变量中存储的数据的安全性。在 C#中，数据类型分成两大类：一类是值类型（Value Types），一类是引用类型（Reference Types）。

2.2.1　值类型

所谓值类型就是一个包含实际数据的量。当定义一个值类型的变量时，C#会根据它所声明的类型，以堆栈方式分配一块大小相适应的存储区域给这个变量，随后对这个变量的读/写操作就直接在这块内存区域进行。

例如：

```
int  iNum=10;              // 分配一个 32 位内存区域给变量 iNum，并将 10 放入该内存区域
iNum=iNum+10;              // 从变量 iNum 中取出值，加上 10，再将计算结果赋给 iNum
```

图 2.1 给出了这个值类型的操作示意。

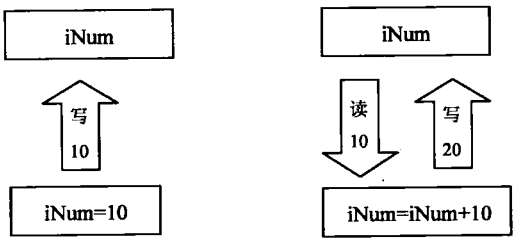

图 2.1　值类型操作示意图

C#中的值类型包括：简单类型、枚举类型和结构类型。

1．简单类型

简单类型是系统预置的，一共有 13 个数值类型，如表 2.1 所示。

表 2.1　C#简单类型

C#关键字	.NET CTS 类型名	说　明	范围和精度
bool	System.Boolean	逻辑值（真或假）	true, false
sbyte	System.SByte	8 位有符号整数类型	−128~127
byte	System.Byte	8 位无符号整数类型	0~255

续表

C#关键字	.NET CTS 类型名	说　明	范围和精度
short	System.Int16	16 位有符号整数类型	–32768~32767
ushort	System.UInt16	16 位无符号整数类型	0~65535
int	System.Int32	32 位有符号整数类型	–2147483648~2147483647
uint	System.Uint32	32 位无符号整数类型	0~4294967295
long	System.Int64	64 位有符号整数类型	–9223372036854775808~9223372036854775807
ulong	System.UInt64	64 位无符号整数类型	0~18446744073709551615
char	System.Char	16 位字符类型	所有的 Unicode 编码字符
float	System.Single	32 位单精度浮点类型	$\pm1.5\times10^{-45}$ ~ $\pm3.4\times10^{38}$ （大约 7 个有效十进制数位）
double	System.Double	64 位双精度浮点类型	$\pm5.0\times10^{-324}$ ~ $\pm3.4\times10^{308}$ （大约 15~16 个有效十进制数位）
decimal	System.Decimal	128 位高精度十进制数类型	$\pm1.0\times10^{-28}$ ~ $\pm7.9\times10^{28}$ （大约 28~29 个有效十进制数位）

表中“C#关键字”是指在 C#中声明变量时可使用的类型说明符。

例如：

```
int myNum    // 声明 myNum 为 32 位的整数类型
```

.NET 的 CTS 包含所有简单类型，它们位于.NET 框架的 System 名字空间。C#的类型关键字就是.NET 中 CTS 所定义类型的别名。从表 2.1 可见，C#的简单数据类型可以分为整数类型（包括字符类型）、实数类型和布尔类型。

整数类型共有 9 种，它们的区别在于所占存储空间的大小，带不带符号位及所能表示的数的范围，这些是程序设计时定义数据类型的重要参数。char 类型归属于整型类别，但它与整型有所不同，不支持从其他类型到 char 类型的隐式转换。即使 sbyte、byte、ushort 这些类型的值在 char 表示的范围之内，也不存在其隐式转换。

实数类型有三种，其中浮点类型 float、double 采用 IEEE754 格式来表示，因此浮点运算一般不产生异常。decimal 类型主要用于财务和货币计算，它可以精确地表示十进制小数数字（如 0.001）。虽然它具有较高的精度，但取值范围较小，因此从浮点类型到 decimal 的转换可能会产生溢出异常；而从 decimal 到浮点类型的转换则可能导致精度的损失，所以浮点类型与 decimal 之间不存在隐式转换。

布尔类型表示布尔逻辑量，它与其他类型之间不存在标准转换，即不能用一个整型数表示 true 或 false，反之亦然，这点与 C/C++不同。

2. 枚举类型

关于枚举类型将在后面章节详细介绍。

3. 结构类型

关于结构类型将在后面章节详细介绍。

2.1.2 引用类型

引用类型包括 class（类）、interface（接口）、数组、delegate（委托）、object 和 string。