高等院校课程设计案例精编

赠送光盘中附有完整的案例源代码

C# 课程设计案例精编

段德亮　余　健　张仁才　编著

- 俄罗斯方块游戏
- 贪吃蛇游戏
- 员工管理信息系统
- 房屋出租管理系统
- 仓库管理信息系统
- 研究生管理信息系统
- 图书馆管理信息系统
- 宿舍管理信息系统
- 理财管理信息系统
- IT设备资产管理系统

赠送光盘

清华大学出版社

高等院校课程设计案例精编

C# 课程设计案例精编

段德亮　余　健　张仁才　编著

清华大学出版社
北　京

内 容 简 介

C# 是一种先进的、面向对象的语言，使用 C# 语言可以让开发人员快速地建立大范围的基于 MS 网络平台的应用，并且提供大量的开发工具和服务，帮助开发人员开发基于计算和通信的各种应用。

本书精选了 10 个 C# 开发案例，分别是员工管理信息系统、房屋出租管理系统、仓库管理信息系统、研究生管理信息系统、图书馆管理信息系统、宿舍管理信息系统、理财管理信息系统、IT 设备资产管理系统、俄罗斯方块游戏的编制和贪吃蛇游戏的编制。

本书内容详实、语言简练、思路清晰、图文并茂、理论与实际设计相结合，适合作为高等院校计算机、自动化、机械、电子等相关专业学生课程设计的指导书，也适合作为开发人员的参考用书。

图书在版编目(CIP)数据

C#课程设计案例精编/段德亮，余健，张仁才编著. —北京：清华大学出版社，2008.6
(高等院校课程设计案例精编)
ISBN 978-7-302-17692-3

Ⅰ.C…　Ⅱ.①段…　②余…　③张…　Ⅲ.C 语言—程序设计—高等学校—教学参考资料　Ⅳ.TP312

中国版本图书馆 CIP 数据核字(2008)第 074142 号

责任编辑：李春明　闫光龙
封面设计：山鹰工作室
版式设计：杨玉兰
责任校对：李凤茹
责任印制：王秀菊
出版发行：清华大学出版社　　**地　　址**：北京清华大学学研大厦 A 座
http://www.tup.com.cn　　**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
印 刷 者：北京市清华园胶印厂
装 订 者：北京市密云县京文制本装订厂
经　　销：全国新华书店
开　　本：185×260　**印　张**：23.25　**字　数**：555 千字
附光盘 1 张
版　　次：2008 年 6 月第 1 版　　**印　　次**：2008 年 6 月第1 次印刷
印　　数：1～4000
定　　价：40.00 元

前　言

C# 是一种先进的、面向对象的语言，使用 C# 语言可以让开发人员快速的建立大范围的基于 MS 网络平台的应用，并且提供大量的开发工具和服务，帮助开发人员开发基于计算和通信的各种应用。由于 C# 是一种面向对象的开发语言，所以 C#可以大范围地适用于高层商业应用和底层系统的开发。即使是通过简单的 C# 构造，也可以让各种组件方便地转变为基于 Web 的应用，并且能够通过 Internet 被各种系统或是其他开发语言所开发的应用调用。

本书精选了 8 个信息系统案例和两个游戏案例，按照开发信息系统和游戏的步骤详细阐述了系统的开发过程。这 10 个案例分别是员工管理信息系统、房屋出租管理系统、仓库管理信息系统、研究生管理信息系统、图书馆管理信息系统、宿舍管理信息系统、理财管理信息系统、IT 设备资产管理系统、俄罗斯方块游戏的编制和贪吃蛇游戏的编制。其中，房屋出租管理系统后台数据库采用 Microsoft SQL Server，其他系统后台数据库采用 Microsoft Access。Access 是 Office 系列软件中用来专门管理数据库的应用软件，是一种功能强大并且使用方便的关系型数据库管理系统，一般也称为关系型数据库管理软件。它可运行于各种 Microsoft Windows 系统环境中，由于它继承了 Windows 的特性，不仅易于使用，而且界面友好，如今在世界各地广泛流行。它不需要数据库管理者具有专业的程序设计水平，任何非专业的用户都可以用它来创建功能强大的数据库管理系统。

本书适合作为高等院校计算机、自动化、机械、电子等专业学生课程设计的指导书，也适合作为开发人员的参考书。

本书由段德亮、余健、张仁才等编著。参与编写的作者还包括：张伟、陈嗥、蔚辉、张坤、陈运来、田野、仇亚飞、刘广兴、王翠翠、代小华、王莹莹、韩忠明、张辰威。由于编者水平有限，加上时间仓促，书中难免有一些不足之处，欢迎同行和读者批评指正。

编　者

目　录

第 1 章　C# 基础知识

通过本章学习应该重点掌握 Microsoft .NET 的基本概念和 C#的基本语法。

1.1　Visual Studio.NET

1.1.1　什么是.NET

2006 年 6 月 22 日，微软公司正式推出了其下一代计算计划——Microsoft.NET(以下简称.NET)。根据微软的定义：.NET is a “revolutionary new platform，built on open Internet protocols and standards，with tools and services that meld computing and communications in new ways”。即.NET=新平台+标准协议+统一开发工具。

. NET 首先是一个开发平台，它定义了一个公用语言子集(Comment Language Subset，CLS)，这是一种为符合其规范的语言与类库之间提供无缝集成的混合语。.NET 统一的编程类库，提供了对下一代网络通信标准可扩展标记语言(Extensible Markup Language，XML)的完全支持，使应用程序的开发变得更容易，更简单。

1.1.2　.NET 结构

.NET 结构是通用语言基础结构(Comment Language Infrastructure，CLI)的微软实现加上几个支持用户界面、数据和 XML、Web 服务和基类库的程序包。.NET 结构分为三个主要的子集：通用语言运行时(Comment Language Runtime，CLR)、库和语言。图 1-1 就是.NET 结构。

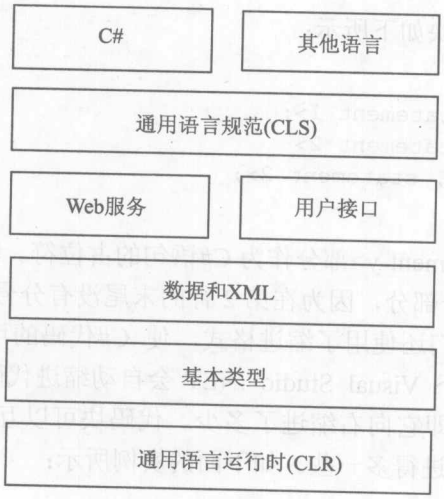

图 1-1 .NET 结构

1.2 基本 C#

1.2.1 什么是 C#

C# 语言是一种简单、现代、优雅、面向对象、类型安全、平台独立的新型组件编程语言，是微软公司为了能够完全利用.NET 平台优势而开发的一种新型编程语言。其语法风格源自 C/C++ 家族，融合了 Visual Basic 的高效和 C/C++的强大，是微软为奠定其下一互联网霸主地位而打造的 Microsoft.Net 平台的主流语言。其一经推出便因其强大的操作能力，优雅的语法风格，创新的语言特性，第一等的面向组件编程的支持而深受世界各地程序员的好评和喜爱。尽管它借鉴了 C 和 C++的许多特性，但是在一些诸如名字空间、类、方法和异常处理等特定领域，它们之间还存在着巨大的差异。

1.2.2 C# 代码结构

C# 代码的外观和操作方式与 C++ 和 Java 非常类似，在 C# 编程中，使用的样式是比较清晰的，不用花太多的力气就可以编写出可读性很强的代码。

与其他语言的编译器不同，无论代码中是否有空格、回车符或 Tab 字符(这些字符统称为空白字符)，C# 编译器都不考虑这些字符。这样格式化代码时就有很大的自由度，但遵循某些规则将有助于使代码易于阅读。

C# 代码由一系列语句组成，每个语句都用一个分号来结束。因为空格被忽略，所以一行可以有多个语句，但从可读性的角度来看，通常在分号的后面加上回车符，这样就不能在一行上放置多个语句了。但一句代码放在多个行上是可以的(也比较常见)。

C# 是一个块结构的语言，所有的语句都是代码块的一部分。这些块用花括号来界定({ 和 })，代码块可以包含任意行语句，或者根本不包含语句。注意花括号字符不需要附带分号。

所以，简单的 C#代码块如下所示：

```
    {
[1]     <code line 1, statement 1>;
[2]     <code line 2, statement 2>
[3]         <code line 3, statement 2>;
    }
```

其中<code line x, statement y>部分作为 C#语句的占位符。注意在这段代码中，第 2、3 行代码是同一个语句的一部分，因为在第 2 行的末尾没有分号。

另外在代码块中，我们还使用了缩进格式，使 C#代码的可读性更高，这是一个标准规则，实际上在默认情况下 Visual Studio .NET 会自动缩进代码。一般情况下，每个代码块都有自己的缩进级别，即它向右缩进了多少。代码块可以互相嵌套(即块中可以包含其他块)，而被嵌套的块要缩进得多一些。如下面的实例所示：

```
    {
[1]     <code line 1>;
        {
```

```
[2]        <code line 2>;
[3]        <code line 3>;
       }
[4]    <code line 4>;
   }
```

其中，第 2、3 行代码就是被嵌套的块。另外，前面代码的续行通常也要缩进得多一些，如上面第一个示例中的第 3 行代码。

1.2.3　C# 注释

在 C# 代码中，另一个常见的语句是注释。注释并不是严格意义上的 C# 代码，但代码最好有注释。注释就是解释，即给代码添加描述性文本，编译器会忽略这些内容。在开始处理比较长的代码段时，注释可用于给正在进行的工作添加提示，例如“这行代码要求用户输入一个数字”，或“这段代码由 Bob 编写”。另外，注释增加了代码的可读性。

C# 添加注释的方式有两种——单行和多行。

1. 单行注释

单行注释是一次只在一行注释。它们用双正斜线(//)标记开始。单行注释可以从给定行的任何位置开始，并通过回车符结束。下面实例中的第 1、4 和 7 行所示就是单行注释：

```
[1] //主窗体被关闭的时候,断开与数据库的连接
[2] private void MainForm_Closed(object sender, System.EventArgs e)
[3] {
[4]     //判断与数据库的连接是否断开
[5]     if(oleConnection1!=null)
[6]     {
[7]         //断开数据库的连接
[8]         oleConnection1.Close();
[9]     }
[10] }
```

在 C# 中，还有第三类注释，严格地说，这是“//”语法的扩展。它们都是单行注释，用三个“/”符号来开头，而不是两个。下面实例就是第三类注释。

```
/// A special comment
```

在正常情况下，编译器会忽略它们，就像其他注释一样，但可以配置 Visual Studio .NET，在编译项目时，提取这些注释后面的文本，创建一个特殊格式的文本文件，该文件可用于创建文档说明书。

2. 多行注释

多行注释在一组注释是定界符内有一行或多行叙述。注释定界符有开始注释标记(/*)和结束注释标记(*/)组成。在这两个标记符中的所有内容都被看做是注释，在编译器读取源代码时被忽略掉。下面的实例就是一个多行注释。

```
[1]     /*
[2]      * 作者：段德亮
[3]      * 开始时间：6 月 2 号
[4]      * 功能：建立数据库连接
[5]      * */
```

但是，需要注意的一点是，在 C# 中不允许出现多行注释的嵌套。例如下面的注释嵌套。

```
[1]     /*
[2]      * 作者：段德亮
[3]      * 开始时间：6 月 2 号
[4]      * 功能：建立数据库连接
[5]      *       /*第一次修改时间：6 月 3 号
[6]      *          第二次修改时间：6 月 6 号
[7]      *          第二次修改人：张涛
[8]      *       */
[9]      * */
```

在第 1 行的开始注释标记了多行注释，第 5 行的第 2 个开始注释标记作为注释中的一个字符而被忽略。在第 8 行的结束注释标记与第 1 行的开始注释标记相匹配，最后第 9 行的结束注释标记则会被编译器报告为语法错误，因为没有与它相匹配的开始注释标记。

1.2.4 标识符与关键字

标识符是唯一的标识代码中的各种程序元素的名称，例如变量或字段等元素。它们由程序员指定，并且应该具有标识它们的目的的名称。

关键字是 C# 语言中的保留字，由于它们是保留的，所以不能把它们用作标识符。

1. 标识符

标识符是用于识别代码元素的名称，可以在程序中用作变量、对象、类、结构、枚举、类型、函数等的名字。

C# 中标识符的定义需要遵守以下几条规则：

(1) 只能由大写字母、小写字母、下划线、数字(0～9)组成。

(2) 必须以大写字母、小写字母或下划线开头。

(3) 标识符大小写敏感，比如变量名 fl 和 F1 代表不同的变量。尽管如此，我们仍不建议仅利用大小写不同来代表不同的标识符，在大多数情况下，标识符应该是望名知义。

标识符规则完全符合 Unicode 标准推荐的规则，但以下情况除外：

(1) 允许将下划线用作初始字符。

(2) 允许在标识符中使用 Unicode 转义序列。

(3) 允许“@”字符作为前缀以使关键字能够用作标识符。

下面是一些合法的 C# 标识符的例子：

```
oleConnection1
_command
```

```
@string
```

下面是一些无效的标识符的例子：

```
1tree
namespace
```

其中，1tree 无效是因为第 1 个字符是数字，标识符不允许以数字开始。标识符的第 1 个字符必须是字母或下划线。namespace 无效是因为它是保留字，不能用来作标识符。

2. 关键字

关键字是对编译器具有特殊意义的预定义保留标识符，不能在程序中用作标识符，除非有一个@前缀。例如，@if 是一个合法的标识符，而 if 不是合法的标识符，因为它是关键字。C#的关键字清单如表 1-1 所示。

表 1-1　C#关键字

abstract	event	new	struct	case	float	params	typeof	continue
in	return	using	as	explicit	null	switch	catch	for
private	uint	decimal	int	sbyte	virtual	base	extern	object
this	char	foreach	protected	ulong	default	interface	sealed	volatile
bool	false	operator	throw	checked	goto	public	unchecked	delegate
internal	short	void	break	finally	out	true	class	if
readonly	unsafe	do	is	sizeof	while	byte	fixed	override
try	const	implicit	ref	ushort	double	lock	stackalloc	string
else	long	static	enum	namespace				

1.3　C# 基本类型

应用程序总是要处理数据，而现实世界中的数据类型多种多样，因此必须要让计算机了解需要处理什么样的数据，以及采用哪种方式处理，按什么样的格式保存数据等。

任何一个完整的应用程序都可以看做是数据和作用于这些数据上的操作的说明。每一种高级语言都为开发人员提供了一组数据类型，不同的语言提供的数据类型不尽相同。

变量关系到数据的存储。实际上，可以把计算机内存中的变量看作架子上的盒子。在这些盒子中，可以放入一些东西，再把它们取出来，或者只是看看盒子里是否有东西。变量也是这样，数据可以放在变量中，也可以从变量中取出数据或查看数据。

尽管计算机中的所有数据都是相同的东西(一组 0 和 1)，但变量有不同的内涵，称为类型。下面再使用盒子来类比，盒子有不同的形状和尺寸，某些东西只能放在特定的盒子中。建立这个类型系统的原因是，不同类型的数据需要用不同的方法来处理。变量限定为不同的类型，可以避免混淆它们。例如，组成数字图片的 0 和 1 序列与组成声音文件的 0 和 1 序列，其处理方式是不同的。

实际上，可以使用的变量类型是无限多的。其原因是可以自己定义类型，存储各种复杂的数据。尽管如此，C# 中有两种基本数据类型：值类型和引用类型。值类型是直接存储它的数据内容，而引用类型存储的是对象的引用，这两种类型对变量的赋值有着不同的含义。

1.3.1 值类型

基于值类型的变量直接包含值，将一个值类型变量赋给另一个值类型变量时，将复制包含的值。这与引用类型变量的赋值不同，引用类型变量的赋值只复制对对象的引用，而不复制对象本身。

C# 的值类型可以分为以下几种：

- 简单类型(Simple Types)；
- 结构类型(Struct Types)；
- 枚举类型(Enumeration Types)。

简单类型就是组成应用程序中基本组成部件的类型，例如数值类型和布尔类型。数值类型又进一步分为整数类型、实数类型和字符类型。

1. 布尔类型

布尔类型是用来表示“真”和“假”这两个概念的。布尔类型表示的逻辑变量只有两种取值：“真”和“假”。在 C# 中，分别采用 true 和 false 这两个值来表示。在 C 和 C++中，用 0 来表示“假”，其他任何非 0 的值表示“真”，这种表示方法在 C#中是错误的。在 C#中，true 值不能被任何非 0 的值代替，在整数类型和布尔类型之间不存在任何转换。下面是一些例子：

```
Bool x = true;  //正确
Bool y = 1;     //错误
```

2. 整数类型

整数类型变量的值为整数。数学上的整数可以是从负无穷大到正无穷大，但由于计算机的存储单元是有限的，所以计算机语言提供的整数类型有一个取值范围。根据该类型的变量在内存中占的位数，可以将 C# 中的整数类型分为 8 种：短字节型(sbyte)、字节型(byte)、短整型(short)、无符号短整型(ushort)、整型(int)、无符号整型(uint)、长整型(long)、无符合长整型(ulong)。一些变量名称前面的“u”是 unsigned 的缩写，表示不能在这些类型的变量中存储负数。

这些整型变量的大小和取值范围如表 1-2 所示。

表 1-2 整型变量

数据类型	大小(以位为单位)	取值范围
sbyte	8	-128～127
byte	8	0～255
short	16	-32768～32767

续表

数据类型	大小(以位为单位)	取值范围
ushort	16	0～65535
int	32	-2147483648～2147483647
uint	32	0～4294967295
long	64	-9223372036854775808～9223372036854775807
ulong	64	0～18446744073709551615

3．实数类型

C# 中实数类型又分为了浮点类型和十进制类型。

C# 中表示浮点数变量类型有两种：单精度(float)和双精度(double)。它们的差别在于取值范围和精度的不同。对精度要求不高的浮点数计算中，可以采用 float 类型，而采用 double 类型的计算结果更加准确，但是将会占用更多的内存单元，加重计算机的负担。

十进制类型(decimal)是 C# 中专门定义的一种数据类型，它主要是为了方便财务方面的计算。十进制类型的取值范围比双精度类型小的多，但它的精度更高。

当定义一个十进制变量并赋值时，可以用后缀 M 或 m 来表明它是十进制类型，可以用后缀 F 或 f 来表明它是单精度浮点型，用后缀 D 或 d 表明它是双精度浮点型。用后缀来表示该变量所需要的类型能够确保表达式进行正确的计算。在计算表达式时，编译器会把没有后缀的 float 和 decimal 都作为 double 类型处理。如下面例子所示：

```
float f_value = 3.679f;
double d_value = 893.8D;
decimal num = 43222222.99M;
```

实数类型的特征如表 1-3 所示。

表 1-3　实数类型

数据类型	大小(以位为单位)	精　度	取值范围
float	32	7 位数字	1.5×10^{-45}～3.4×10^{38}
double	64	15～16 位数字	5.0×10^{-324}～1.7×10^{308}
decimal	128	28～29 位数字	1.0×10^{-28}～7.9×10^{28}

4．字符类型

除了数字，计算机所有处理的信息主要就是字符。C#提供的字符类型按照国际上公认的标准，采用 Unicode 字符集，一个 Unicode 的标准字符长度为 16 位。

我们可以按以下方法给字符变量赋值：

```
char c = 'A';
```

也可以通过十六进制转义符(前缀\x)或 Unicode 表示法(前缀\u)给字符变量赋值。如下所示：

```
char c = '\x0041';
char c = '\x005A';
```

5. 结构类型

在实际生活中，我们经常把一组相关的信息放在一起。把一系列相关的变量组织成为一个单一实体的过程，就是结构的生成过程。这个单一实体的类型就是结构类型，它里面的每一个变量都被称为结构的成员。结构类型的变量采用 struct 进行声明。例如下面地址信息结构的定义：

```
struct position{
    public string city;
    public string street;
    public uint no;
}
position p1;
```

city、street 和 no 都是 position 结构类型的成员，p1 是它的变量名，public 表示对结构成员的访问权限。对结构成员的访问是通过结构变量名加上访问符“.”号，再跟上成员的名称实现：

```
p1.city = "北京";
p1.no = 356;
```

6. 枚举类型

enum 关键字用于声明枚举，即一种由一组称为枚举数列表的命名常数组成的独特类型。每种枚举类型都有基础类型，该基础类型可以是除 char 以外的任何整型。枚举元素的默认基础类型为 int。默认情况下，第一个枚举数的值为 0，后面每个枚举数的值依次递增 1。例如：

```
enum Days {Sat, Sun, Mon, Tue, Wed, Thu, Fri};
```

在此枚举中，Sat 为 0，Sun 为 1，Mon 为 2，依此类推。枚举数可以重写默认值的初始值设定项。例如：

```
enum Days {Sat=1, Sun, Mon, Tue, Wed, Thu, Fri};
```

在此枚举中，强制元素序列从 1 而不是从 0 开始。

枚举类型的变量值只能是其中的一个变量值。例如：

```
enum Days {Sat, Sun, Mon, Tue, Wed, Thu, Fri};
Days day;
day = Mon;
```

1.3.2 引用类型

引用类型是 C#中另一大数据类型，该类型的变量不直接存储所包含的值，而是指向所要存储的值，即引用类型存储的是实际数据的引用值的地址。