

软件开发

# 掌皮卷

DUANPIJUAN KAIZHAI  
YANG SHIQUAN

程序员是怎样炼成的？在持续激励中磨砺，在反复练习中成长，C#疯狂特训内容，尽在本书中……

# 学通 C# 的24堂课

王小科 赵会东 ◎等编著

## 110集大型多媒体教学视频

420个中小实例训练

600余段源码分析

190个应用模块精解

6大项目案例展示



DVD大型多媒体光盘

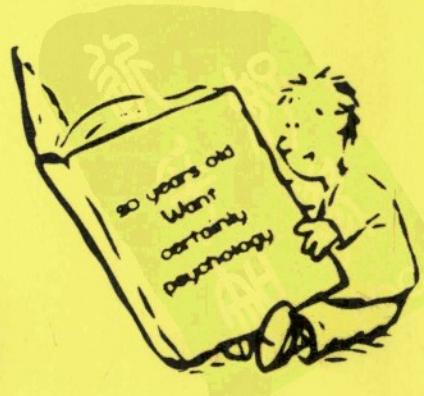
110集教学视频：110集（282段）多媒体教学视频，听程序员现场讲解。

420个中小实例：夯实必备知识，强化基本功训练

600余源码分析：寻找编程感觉，培养编程思想

190个应用模块：激发学习兴趣，突出开发实战

6大项目案例：体验项目开发过程，积累项目开发经验



清华大学出版社

# 《软件开发羊皮卷》丛书



学通Java Web的24堂课

定价：79.80元

ISBN 978-7-302-25539-0



9 787302 255390 >

学通ASP.NET的24堂课

定价：79.80元

ISBN 978-7-302-25540-6



9 787302 255406 >

学通Java的24堂课

定价：79.80元

ISBN 978-7-302-25541-3



9 787302 255413 >

学通PHP的24堂课

定价：79.80元

ISBN 978-7-302-25542-0



9 787302 255420 >

学通C#的24堂课

定价：79.80元

ISBN 978-7-302-25543-7



9 787302 255437 >

学通Visual C++的24堂课

定价：79.80元

ISBN 978-7-302-25652-6



9 787302 256526 >

学通Visual Basic的24堂课

定价：79.80元

ISBN 978-7-302-25639-7



9 787302 256397 >

学通C语言的24堂课

定价：79.80元

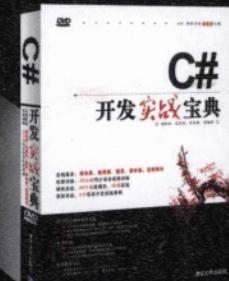
ISBN 978-7-302-25728-8



9 787302 257288 >



## 推荐阅读：实战类图书



《C#开发实战宝典》是一本围绕“实战”展开的、内容丰富的编程参考书，它与本书有内容衔接的地方。主要特点有：

- 模块实战：39个模块实战，68项实验
- 项目实战：8个项目开发案例
- 众多实例：412个小型实例
- 视频讲解：23小时同步语音视频讲解
- 在线服务：提供模块库、案例库、题库、素材库、答疑服务等

ISBN 978-7-302-25543-7



9 787302 255437 >

定价：79.80元（附视频DVD1张）

TP312C  
W383-11



郑州大学 \*04010712110H\*

软件开发羊皮卷

12

# 学通 C# 的 24 堂课

## ( 110 集大型多媒体教学视频 )

王小科 赵会东 等编著



清华大学出版社

北京

TP312C  
W383-11

## 内 容 简 介

本书以 24 堂课的形式, 从初中级用户的角度进行科学合理的设计, 全面讲述了使用 C# 进行程序开发所必备的知识和技能, 突出学、练、用结合。主要内容包括初探 C# 及其开发环境、C# 程序设计基础、程序流程控制、字符及字符串的使用、数组与集合、程序设计中的算法、面向对象程序设计、Windows 窗体设计、Windows 应用程序常用控件、Windows 应用程序高级控件、ADO.NET 数据访问技术、DataGridView 数据控件、面向对象编程高级技术、LINQ 技术的使用、文件及 IO、GDI+ 绘图技术、水晶报表与打印、网络编程、线程的使用、异常处理与程序调试、Windows 应用程序打包部署、企业人事管理系统、房屋中介管理系统、进销存管理系统。

本书适合有志于从事软件开发的初学者、高校计算机相关专业的学生和毕业生, 也适合作为软件开发人员的参考手册或高校的教学参考书。

本书通过教学视频、实例训练、综合应用、项目实践、自我测试、行动指南逐步深入和强化训练等方式, 并辅之以心理励志, 来持续激发读者主动学习、自发学习。

本书给出了 420 个小型实例、190 个综合应用, 6 个项目案例 (部分在光盘中), 各类技巧、试验 200 余个, 测试题目 210 个, 以方便读者训练、测试和快速提升。

本书 DVD 光盘给出了 110 集 (282 段) 多媒体教学视频讲解, 每个实例都给出了相应的源程序, 可直接复制源码学习或应用。

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

学通 C# 的 24 堂课 / 王小科, 赵会东等编著. —北京: 清华大学出版社, 2011.6

(软件开发羊皮卷)

ISBN 978-7-302-25543-7

I. ①学… II. ①王… ②赵… III. ①C 语言 - 程序设计 IV. ①TP312

中国版本图书馆 CIP 数据核字 (2011) 第 091858 号

责任编辑: 赵洛育

版式设计: 文森时代

责任校对: 王国星

责任印制: 杨艳

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: 北京密云胶印厂

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: 203×260 印 张: 46.25 字 数: 1492 千字

(附 DVD 光盘 1 张)

版 次: 2011 年 6 月第 1 版 印 次: 2011 年 6 月第 1 次印刷

印 数: 1~5000

定 价: 79.80 元

---

产品编号: 042544-01

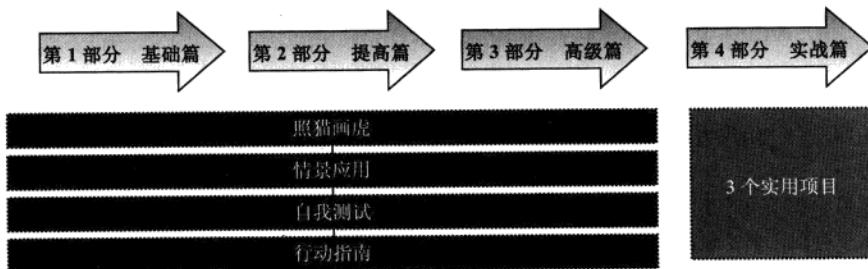
# 前 言

Preface

随着微软.NET 平台的逐步推广，C#作为一种基于.NET 平台的面向对象编程语言，正在被越来越多的程序开发人员所使用。C#自从面世以来一直以易学易用、功能强大的特点得到了广泛的应用，而 Visual Studio 开发平台则凭借其强大的可视化用户界面设计，使程序员从复杂的界面设计中解脱出来，让编程成为一种享受。C#不但可以开发数据库管理系统，而且还可以开发集声音、动画、视频为一体的多媒体应用程序和网络应用程序。

## 本书内容

学、练、用到精通只需 24 堂课。本书从初中级用户的角度进行科学合理的设计，通过 24 堂课全面讲述了使用 C#进行程序开发所必备的知识和技能，如下图所示。



**第 1 部分 基础篇（第 1~7 堂课）：**讲述了初探 C#及其开发环境、C#程序设计基础、程序流程控制、字符及字符串的使用、数组与集合、程序设计中的算法、面向对象程序设计等内容。本篇内容是程序语言的基础，若学通了，再学其他篇相信一定会很简单。

**第 2 部分 提高篇（第 8~12 堂课）：**讲述了使用 C#进行应用程序开发的各种常用技术，包括 Windows 窗体设计、Windows 应用程序常用控件、Windows 应用程序高级控件、ADO.NET 数据访问技术、DataGridView 数据控件等内容。通过本篇内容的学习，读者能够开发小型应用程序以及常用的数据库应用程序。

**第 3 部分 高级篇（第 13~21 堂课）：**讲述了面向对象编程高级技术、LINQ 技术的使用、文件及 IO、GDI+绘图技术、水晶报表与打印、网络编程、线程的使用、异常处理与程序调试、Windows 应用程序打包部署等内容。通过本篇内容的学习，读者可以掌握 C#高级应用程序的开发，如图形图像、多媒体及网络等。

**第 4 部分 实战篇（第 22~24 堂课）：**讲述了企业人事管理系统、房屋中介管理系统和进销存管理系统 3 个完整的项目实例设计全过程。通过本篇内容的学习，读者可以积累项目开发经验。

## 本书特点

- 配备 110 集（282 段）多媒体教学视频讲解。

本书 DVD 光盘提供了覆盖全书的语音视频讲解，读者可以通过视频快速、直观、轻松地学习。

每一堂课都结合“照猫画虎”、“情景应用”。

为了增强读者的动手能力，并激发学习兴趣，本书提供了“照猫画虎”和“情景应用”栏目，根据实例模仿着去做是学习编程的最快方式。

每一堂课都结合励志故事和“行动指南”，时刻进行激励和鼓舞。

我们认为学习中尤其是最初的一段时间非常有必要不断地对学习者以激励和鼓舞，让他们坚持下来是至关重要的，因此书中不间断地用一些励志故事和“行动指南”去鼓舞其信心。

光盘提供了本书的所有代码，即使只有一行。

本书光盘不仅提供了所有实例的源程序，还提供了书中所有示例的源代码，哪怕只有一行。读者可以直接复制，以提高学习效率。

本书所有习题和实战都给出了答案，部分习题还有解析，读者可以对照查阅。

## 读者对象

有志于软件开发的初学者

高等院校计算机相关专业的老师和学生

准备从事软件开发的求职者

参与毕业设计的学生

初中级程序开发人员

程序测试及维护人员

## 本书作者

本书由明日科技组织编写，参加编写的程序员有王小科、赵会东、王国辉、赛奎春、潘凯华、刘欣、李慧、陈丹丹、高春艳、李伟、孙秀梅、杨丽、刘玲玲、朱晓、刘燕、陈英、李鑫、李贺、肖鑫、张丽娜、沈博、刘冠男、曹飞飞、李丽、聂喜婷、王明昭、张英豪、白伟明等。

由于水平有限，书中疏漏和不足之处在所难免，恳请广大读者朋友批评指正。

## 技术支持与服务

秉着“十年服务，始终如一”理念，我们承诺如果您在学习或使用本书的过程中遇到问题或疑惑，可以通过如下方式与我们联系。

登录技术服务网站：[www.mingribook.com](http://www.mingribook.com)，查阅相关问题或者留言。

通过企业服务邮箱：[tmoonbook@sina.com](mailto:tmoonbook@sina.com) 或 [th\\_press@263.net](mailto:th_press@263.net)。

申请加入服务 QQ：100310286。

我们承诺将在 5 个工作日内给您提供解答。

最后，感谢您选择本书，希望本书能成为您编程路上的领航者。

祝读书快乐！

### 特别提醒：

亲爱的读者朋友，由于近期纸张价格和印制成本大幅上涨，为不增加读者朋友的负担，又不减少书的内容，本书的最后几章内容不得不放在配书光盘中（参见目录），由此给您带来了不便，在此深表歉意。

编 者

# 目 录

Contents

## 第1部分 基础篇

<b>第1堂课 初探C#及其开发环境 .....</b>	3
<b>    <b>视频讲解：96分钟</b></b>	
1.1 了解.NET .....	4
1.1.1 .NET概述 .....	4
1.1.2 .NET程序编译原理 .....	4
1.1.3 .NET项目成功案例 .....	5
1.2 C#语言及特点 .....	6
1.2.1 C#与.NET的关系 .....	6
1.2.2 C#语言特点 .....	6
1.2.3 C#语言发展趋势 .....	7
1.3 安装与卸载Visual Studio 2008 .....	7
1.3.1 安装Visual Studio 2008系统必备 .....	7
1.3.2 安装Visual Studio 2008 .....	8
1.3.3 卸载Visual Studio 2008 .....	9
1.4 熟悉Visual Studio 2008开发环境 .....	10
1.4.1 创建控制台应用程序 .....	10
1.4.2 创建Windows应用程序 .....	11
1.4.3 菜单栏介绍 .....	12
1.4.4 工具栏介绍 .....	13
1.4.5 “工具箱”面板介绍 .....	14
1.4.6 “属性”面板介绍 .....	14
1.4.7 解决方案资源管理器介绍 .....	15
1.5 C#编程常用帮助 .....	15
1.5.1 安装MSDN .....	15
1.5.2 使用MSDN .....	17
1.6 照猫画虎——基本功训练 .....	18
1.6.1 基本功训练1——如何开始运行程序 .....	18
1.6.2 基本功训练2——如何中断当前程序的运行 .....	19
1.6.3 基本功训练3——设置程序代码行号 .....	20
1.6.4 基本功训练4——统一窗体中控件的字体设置 .....	20
1.6.5 基本功训练5——通过“格式”菜单布局窗体 .....	21
1.7 情景应用——拓展与实践 .....	22
1.7.1 情景应用1——设置Windows应用程序启动窗体 .....	22
1.7.2 情景应用2——为程序设置版本和帮助信息 .....	23
1.7.3 情景应用3——为项目添加已有窗体 .....	24
1.7.4 情景应用4——动起来的Label控件 .....	24
1.7.5 情景应用5——加法计算器 .....	25
1.8 自我测试 .....	26
1.9 行动指南 .....	27
1.10 成功可以复制——C#语言之父安德斯·海尔斯伯格 .....	28
<b>第2堂课 C#程序设计基础 .....</b>	31
<b>    <b>视频讲解：168分钟</b></b>	
2.1 编写第一个C#程序 .....	32
2.2 分析C#程序结构 .....	33
2.2.1 命名空间介绍 .....	33
2.2.2 类的介绍 .....	34
2.2.3 Main方法的使用 .....	34
2.2.4 认识标识符 .....	35
2.2.5 认识关键字 .....	35
2.2.6 编写C#语句 .....	35
2.2.7 代码注释 .....	36

2.3 数据类型 .....	37	2.9.5 情景应用 5——使用条件运算符判断 指定年份是不是闰年.....	66
2.3.1 值类型的使用.....	37	2.10 自我测试 .....	67
2.3.2 引用类型的使用.....	39	2.11 行动指南 .....	68
2.4 声明并使用变量 .....	40	2.12 成功可以复制——中国第一程序员 求伯君 .....	69
2.4.1 变量的声明及初始化.....	40		
2.4.2 变量的作用域.....	41		
2.5 声明并使用常量 .....	42		
2.6 数据类型转换 .....	43		
2.6.1 隐式类型转换.....	43		
2.6.2 显式类型转换.....	43		
2.6.3 装箱和拆箱.....	45		
2.7 运算符的使用 .....	46		
2.7.1 算术运算符.....	46		
2.7.2 赋值运算符.....	48		
2.7.3 关系运算符.....	49		
2.7.4 逻辑运算符.....	51		
2.7.5 移位运算符.....	54		
2.7.6 其他特殊运算符.....	54		
2.7.7 运算符的优先级.....	56		
<b>2.8 照猫画虎——基本功训练 .....</b>	<b>57</b>		
2.8.1 基本功训练 1——使用 “ <code>///</code> ” 标记给 代码段添加说明 .....	57		
2.8.2 基本功训练 2——使用引号运算符 进行赋值.....	58		
2.8.3 基本功训练 3——使用 <code>checked</code> 关键字处理“溢出”错误.....	58		
2.8.4 基本功训练 4——使用 <code>typeof</code> 关键字 获取类的内部结构.....	59		
2.8.5 基本功训练 5——使用 <code>using</code> 关键字 有效回收资源.....	60		
<b>2.9 情景应用——拓展与实践 .....</b>	<b>61</b>		
2.9.1 情景应用 1——检查对象是否与给定 类型兼容.....	61	3.7.1 基本功训练 1——循环向控制台中输出 内容 .....	95
2.9.2 情景应用 2——使用算术运算符开发 简单计算器.....	62	3.7.2 基本功训练 2——使用 <code>switch</code> 语句 实现数字转换大写 .....	96
2.9.3 情景应用 3——使用 “ <code>^</code> ” 运算符对 数字进行加密 .....	64	3.7.3 基本功训练 3——鸡尾酒排序算法的实现....	97
2.9.4 情景应用 4——巧用移位运算符获取 汉字编码值.....	65	3.7.4 基本功训练 4——判断用户登录身份 .....	99
		3.7.5 基本功训练 5——小明去学校和医院 分别要走哪条路.....	100
		<b>3.8 情景应用——拓展与实践 .....</b>	<b>101</b>

3.8.1 情景应用 1——递归算法的经典面试题 ...	101	字符进行颠倒输出.....	126
3.8.2 情景应用 2——使用流程控制语句 报销业务花销.....	102	4.5.3 基本功训练 3——去掉字符串中的 所有空格.....	126
3.8.3 情景应用 3——使用 switch 语句更改 窗体颜色.....	102	4.5.4 基本功训练 4——获取字符串中汉字的 个数.....	127
3.8.4 情景应用 4——使用 goto 语句在数 组中搜索指定图书.....	103	4.5.5 基本功训练 5——从字符串中分离 文件路径、文件名及扩展名.....	127
3.8.5 情景应用 5——制作一个数字猜猜看 小游戏.....	104	<b>4.6 情景应用——拓展与实践</b> .....	128
3.9 自我测试 .....	105	4.6.1 情景应用 1——字母与 ASCII 码的转换....	128
3.10 行动指南 .....	107	4.6.2 情景应用 2——将汉字转换为拼音 .....	129
3.11 成功可以复制——软件业的华人 教父王嘉廉 .....	108	4.6.3 情景应用 3——批量替换某一类字符串 ...	130
<b>第 4 堂课 字符及字符串的使用</b> .....	111	4.6.4 情景应用 4——对字符串进行加密与 解密 .....	131
 <b>视频讲解：106 分钟</b>		4.6.5 情景应用 5——开发一个进制转换器 ....	133
4.1 字符操作 .....	112	4.7 自我测试 .....	136
4.1.1 Char 类概述.....	112	4.8 行动指南 .....	137
4.1.2 使用 Char 类中的方法对字符进行操作....	112	4.9 成功可以复制——“杀毒王”王江民 不可思议的传奇人生 .....	138
4.1.3 认识并使用转义字符.....	114	<b>第 5 堂课 数组与集合</b> .....	141
4.2 字符串的声明及初始化 .....	114	 <b>视频讲解：139 分钟</b>	
4.3 字符串操作 .....	115	5.1 数组概述 .....	142
4.3.1 比较字符串.....	115	5.2 一维数组的声明和使用 .....	142
4.3.2 格式化字符串.....	116	5.2.1 一维数组的声明.....	142
4.3.3 截取字符串.....	117	5.2.2 一维数组的使用.....	143
4.3.4 分割字符串.....	118	5.3 二维数组的声明和使用 .....	144
4.3.5 插入和填充字符串.....	118	5.3.1 二维数组的声明.....	144
4.3.6 删除字符串.....	120	5.3.2 二维数组的使用.....	144
4.3.7 复制字符串.....	120	5.3.3 动态数组的声明及使用.....	145
4.3.8 替换字符串.....	122	5.4 数组的基本操作 .....	146
4.4 可变字符串类 StringBuilder 的使用....	122	5.4.1 遍历数组中的元素.....	146
4.4.1 StringBuilder 类概述.....	122	5.4.2 添加和删除数组元素.....	147
4.4.2 创建 StringBuilder 对象.....	123	5.4.3 数组的合并与拆分.....	147
4.4.3 StringBuilder 类的使用.....	123	5.5 ArrayList 集合的使用 .....	149
4.4.4 StringBuilder 类与 string 类的区别.....	124	5.5.1 ArrayList 集合概述 .....	149
<b>4.5 照猫画虎——基本功训练</b> .....	125	5.5.2 添加 ArrayList 集合元素 .....	150
4.5.1 基本功训练 1——判断用户名输入的 用户名是否正确.....	125	5.5.3 删 除 ArrayList 集合元素 .....	152
4.5.2 基本功训练 2——将字符串的每个		5.5.4 遍历 ArrayList 集合 .....	154

<b>5.6 照猫画虎——基本功训练 .....</b>	<b>155</b>		
5.6.1 基本功训练 1——获取多维数组的行数与列数.....	155	6.3.1 基本功训练 1——计算 $1+2^2+3^3+\dots+n^n$ 的值.....	183
5.6.2 基本功训练 2——按指定条件在数组中检索元素.....	156	6.3.2 基本功训练 2——计算 $10!$ 的值 .....	183
5.6.3 基本功训练 3——在数组中添加一个元素.....	157	6.3.3 基本功训练 3——求最大公约数 .....	184
5.6.4 基本功训练 4——不改变长度删除数组中的元素.....	159	6.3.4 基本功训练 4——将 B 转换成 GB、MB 和 KB.....	185
5.6.5 基本功训练 5——删除数组元素后改变其长度.....	160	6.3.5 基本功训练 5——0~N 位数的任意组合.....	186
<b>5.7 情景应用——拓展与实践 .....</b>	<b>162</b>	<b>6.4 情景应用——拓展与实践 .....</b>	<b>187</b>
5.7.1 情景应用 1——操作便捷的简单电话簿 ...	162	6.4.1 情景应用 1——身份证号从 15 位升到 18 位算法.....	187
5.7.2 情景应用 2——使用数组解决约瑟夫环问题.....	163	6.4.2 情景应用 2——韩信点兵的算法实现 .....	188
5.7.3 情景应用 3——向班级集合中添加学生信息.....	164	6.4.3 情景应用 3——求水仙花数的算法实现 ...	189
5.7.4 情景应用 4——使用哈希表对 XML 文件进行查询.....	165	6.4.4 情景应用 4——制作一个迷你星座查询器.....	190
5.7.5 情景应用 5——设计一个简单客车售票记录程序.....	166	6.4.5 情景应用 5——设计双色球彩票选号器 ...	194
5.8 自我测试 .....	168	6.5 自我测试 .....	196
5.9 行动指南 .....	169	6.6 行动指南 .....	197
5.10 成功可以复制——善于抓住时机的人徐少春 .....	170	6.7 成功可以复制——缔造华人的硅谷传奇杨致远 .....	197
<b>第 6 堂课 程序设计中的算法 .....</b>	<b>173</b>	<b>第 7 堂课 面向对象程序设计 .....</b>	<b>199</b>
<b>    视频讲解: 57 分钟</b>		<b>    视频讲解: 160 分钟</b>	
6.1 算法基础 .....	174	7.1 面向对象编程概述 .....	200
6.1.1 初识算法.....	174	7.2 属性的定义及使用 .....	200
6.1.2 描述算法的两种常用流程图.....	174	7.2.1 属性概述.....	201
6.2 常用的算法 .....	175	7.2.2 属性的定义.....	201
6.2.1 查找最大、最小值算法的实现.....	176	7.2.3 属性的使用.....	202
6.2.2 杨辉三角算法的实现.....	177	7.3 方法的声明及使用 .....	203
6.2.3 冒泡排序法.....	178	7.3.1 方法概述.....	203
6.2.4 插入排序法.....	179	7.3.2 方法修饰符.....	203
6.2.5 选择排序法.....	180	7.3.3 方法的声明.....	204
6.2.6 希尔排序法.....	181	7.3.4 方法的分类.....	205
6.3 照猫画虎——基本功训练 .....	183	7.3.5 重载方法的实现.....	206

7.5.2 类型参数 T .....	209	编号和姓名.....	224
7.5.3 泛型接口的声明及使用.....	209	7.11.2 基本功训练 2——通过定义方法求一个数的平方 .....	225
7.6 结构的定义及使用 .....	210	7.11.3 基本功训练 3——使用重载方法实现不同类型数据的计算.....	226
7.6.1 结构概述.....	211	7.11.4 基本功训练 4——通过结构计算矩形的面积.....	227
7.6.2 结构的定义.....	211	7.11.5 基本功训练 5——通过类继承计算梯形面积.....	227
7.6.3 结构的使用.....	211		
7.7 类与对象详解 .....	212	<b>7.12 情景应用——拓展与实践.....</b>	<b>229</b>
7.7.1 类的概念.....	212	7.12.1 情景应用 1——通过类的多态性确定人类的说话行为 .....	229
7.7.2 类的声明.....	213	7.12.2 情景应用 2——封装类实现一个简单的计算器.....	230
7.7.3 构造函数和析构函数.....	213	7.12.3 情景应用 3——使用分部类记录学生信息.....	231
7.7.4 对象的声明和创建.....	214	7.12.4 情景应用 4——使用泛型存储不同类型的数据列表.....	232
7.8 面向对象特性之封装 .....	216	7.12.5 情景应用 5——使用泛型去掉数组中的重复数字.....	233
7.8.1 封装概述.....	216	7.13 自我测试 .....	234
7.8.2 封装的实现.....	217	7.14 行动指南 .....	235
7.9 面向对象特性之继承 .....	218	7.15 成功可以复制——百度 CEO 李彦宏...	236
7.9.1 继承概述.....	218		
7.9.2 单继承的使用.....	219		
7.9.3 多重继承的使用.....	219		
7.10 面向对象特性之多态 .....	221		
7.10.1 多态概述.....	221		
7.10.2 多态的实现.....	222		
<b>7.11 照猫画虎——基本功训练 .....</b>	<b>224</b>		
7.11.1 基本功训练 1——使用属性存储用户			

## 第 2 部分

## 提高篇

<b>第 8 堂课 Windows 窗体设计 .....</b>	<b>241</b>
<span style="color: #800000;">■</span> 视频讲解: 139 分钟	
8.1 Form 窗体基础.....	242
8.1.1 Form 窗体概述.....	242
8.1.2 添加和删除 Form 窗体.....	242
8.1.3 添加多窗体.....	243
8.1.4 设置窗体的属性.....	244
8.1.5 窗体的显示与隐藏.....	246
8.1.6 触发窗体事件.....	247
8.2 MDI 窗体设计 .....	248
8.2.1 MDI 窗体概述.....	248
8.2.2 设置 MDI 窗体.....	249

8.2.3 排列 MDI 子窗体.....	249
8.3 继承窗体设计 .....	251
8.3.1 继承窗体概述.....	251
8.3.2 创建继承窗体.....	251
8.3.3 在继承窗体中修改继承的控件属性.....	253
8.4 照猫画虎——基本功训练 .....	254
8.4.1 基本功训练 1——控制窗体加载时的位置 .....	254
8.4.2 基本功训练 2——设置窗体在屏幕中的位置 .....	254
8.4.3 基本功训练 3——使窗体始终在桌面最顶层显示.....	255

8.4.4 基本功训练 4——根据桌面大小调整窗体大小.....	256	9.5.2 分组框控件.....	294
8.4.5 基本功训练 5——使背景图片自动适应窗体的大小.....	256	9.5.3 选项卡控件.....	295
<b>8.5 情景应用——拓展与实践 .....</b>	<b>257</b>	<b>9.6 对话框控件 .....</b>	<b>297</b>
8.5.1 情景应用 1——从上次关闭位置启动窗体.....	257	9.6.1 对话框概述.....	297
8.5.2 情景应用 2——自定义最大化、最小化和关闭按钮.....	258	9.6.2 打开对话框.....	297
8.5.3 情景应用 3——磁性窗体的设计 .....	261	9.6.3 另存为对话框.....	298
8.5.4 情景应用 4——制作鼠标穿透窗体 .....	270	9.6.4 浏览文件夹对话框.....	299
8.5.5 情景应用 5——窗体换肤程序 .....	270	<b>9.7 菜单、工具栏和状态栏控件 .....</b>	<b>300</b>
8.6 自我测试 .....	273	9.7.1 菜单控件.....	300
8.7 行动指南 .....	274	9.7.2 工具栏控件.....	301
8.8 成功可以复制——迅雷创始人邹胜龙... .....	274	9.7.3 状态栏控件.....	302
<b>第 9 堂课 Windows 应用程序常用控件 .....</b>	<b>277</b>	<b>9.8 照猫画虎——基本功训练 .....</b>	<b>303</b>
<b>视频讲解：198 分钟</b>		9.8.1 基本功训练 1——在 ComboBox 下拉列表中显示图片.....	303
9.1 控件概述 .....	278	9.8.2 基本功训练 2——实现带查询功能的 ComboBox 控件.....	304
9.1.1 浏览常用控件.....	278	9.8.3 基本功训练 3——在 RichTextBox 控件中实现关键字描红.....	305
9.1.2 控件的分类及作用.....	278	9.8.4 基本功训练 4——对 ListBox 控件中的数据进行排序.....	306
9.2 控件的相关操作 .....	279	9.8.5 基本功训练 5——具有提示功能的工具栏.....	306
9.2.1 添加控件.....	279	<b>9.9 情景应用——拓展与实践 .....</b>	<b>307</b>
9.2.2 对齐控件.....	280	9.9.1 情景应用 1——只允许输入数字的 TextBox 控件 .....	307
9.2.3 锁定控件.....	280	9.9.2 情景应用 2——判断注册用户操作权限 ...	308
9.2.4 删除控件.....	280	9.9.3 情景应用 3——实现类似 Word 的项目编号功能.....	309
9.3 文本类控件 .....	280	9.9.4 情景应用 4——制作带历史信息的菜单 ...	310
9.3.1 标签控件.....	280	9.9.5 情景应用 5——制作仿 XP 系统的任务栏菜单.....	311
9.3.2 按钮控件.....	281	9.10 自我测试 .....	312
9.3.3 文本框控件.....	282	9.11 行动指南 .....	313
9.3.4 有格式文本控件.....	284	9.12 成功可以复制——前微软 CEO 比尔·盖茨 .....	314
9.4 选择类控件 .....	286	<b>第 10 堂课 Windows 应用程序高级控件 .....</b>	<b>317</b>
9.4.1 下拉组合框控件.....	286	<b>视频讲解：170 分钟</b>	
9.4.2 复选框控件.....	287	10.1 存储图像控件 .....	318
9.4.3 单选按钮控件.....	289		
9.4.4 数值选择控件.....	290		
9.4.5 列表控件.....	291		
9.5 分组控件 .....	293		
9.5.1 容器控件.....	293		

10.1.1 在 ImageList 控件中添加图像 .....	318	遍历磁盘目录 .....	341
10.1.2 在 ImageList 控件中移除图像 .....	319	10.8.2 情景应用 2——用树型列表动态显示菜单 .....	343
<b>10.2 列表视图控件 .....</b>	<b>320</b>	10.8.3 情景应用 3——设计一个电子万年历 .....	344
10.2.1 在 ListView 控件中添加移除项 .....	320	10.8.4 情景应用 4——制作一个闹钟计时器 .....	345
10.2.2 选择 ListView 控件中的项 .....	322	10.8.5 情景应用 5——弹出模式窗口显示进度条 .....	346
10.2.3 为 ListView 控件中的项添加图标 .....	323		
10.2.4 在 ListView 控件中启用平铺视图 .....	324		
<b>10.3 树控件 .....</b>	<b>324</b>	10.9 自我测试 .....	348
10.3.1 添加和删除树节点 .....	325	10.10 行动指南 .....	349
10.3.2 获取树控件中选中的节点 .....	326	10.11 成功可以复制——图文世界的缔造者约翰·沃洛克 .....	349
10.3.3 为树控件中的节点设置图标 .....	327		
<b>10.4 日期控件 .....</b>	<b>328</b>		
10.4.1 使用 DateTimePicker 控件显示时间 .....	328		
10.4.2 使用 DateTimePicker 控件以自定义格式显示日期 .....	329		
10.4.3 返回 DateTimePicker 控件中选择的日期 .....	330		
<b>10.5 月历控件 .....</b>	<b>331</b>		
10.5.1 在 MonthCalendar 控件中以粗体显示特定日期 .....	331		
10.5.2 在 MonthCalendar 控件中选择日期范围 .....	332		
<b>10.6 其他高级控件 .....</b>	<b>332</b>		
10.6.1 使用 ErrorProvider 控件验证文本框输入 .....	333	11.1 ADO.NET 概述 .....	352
10.6.2 使用 HelpProvider 控件调用帮助文件 .....	334	11.2 使用 Connection 对象连接数据库 .....	352
10.6.3 使用 Timer 控件设置时间间隔 .....	335	11.2.1 Connection 对象概述 .....	352
10.6.4 使用 ProgressBar 控件显示程序运行进度条 .....	336	11.2.2 连接数据库 .....	352
<b>10.7 照猫画虎——基本功训练 .....</b>	<b>337</b>	11.2.3 关闭连接 .....	353
10.7.1 基本功训练 1——在列表视图中拖动视图项 .....	337	11.3 使用 Command 对象执行 SQL 语句 .....	355
10.7.2 基本功训练 2——制作带复选框的 ListView 控件 .....	338	11.3.1 Command 对象概述 .....	355
10.7.3 基本功训练 3——使用 MaskedTextBox 控件实现输入验证 .....	339	11.3.2 设置数据源类型 .....	356
10.7.4 基本功训练 4——使用 Timer 组件实现人物动画效果 .....	340	11.3.3 执行 SQL 语句 .....	357
10.7.5 基本功训练 5——使用 ErrorProvider 组件验证文本框输入 .....	340	11.4 使用 DataReader 对象读取数据 .....	359
<b>10.8 情景应用——拓展与实践 .....</b>	<b>341</b>	11.4.1 DataReader 对象概述 .....	359
10.8.1 情景应用 1——使用 TreeView 控件 .....	341	11.4.2 判断查询结果中是否有值 .....	359
		11.4.3 读取数据 .....	360
		11.5 数据适配器：DataAdapter 对象 .....	361
		11.5.1 DataAdapter 对象概述 .....	361
		11.5.2 填充 DataSet 数据集 .....	361
		11.5.3 更新数据源 .....	362
		11.6 数据集：DataSet 对象 .....	363
		11.6.1 DataSet 对象概述 .....	363
		11.6.2 合并 DataSet 内容 .....	364
		11.6.3 复制 DataSet 内容 .....	365
		11.7 照猫画虎——基本功训练 .....	366
		11.7.1 基本功训练 1——连接加密的 Access 数据库 .....	366
		11.7.2 基本功训练 2——连接文本文件并	

显示其内容.....	367	删除行 .....	389
11.7.3 基本功训练 3——读取 SQL Server 数据库结构.....	368	12.7 禁用 DataGridView 控件的自动排序 功能 .....	389
11.7.4 基本功训练 4——备份指定的 SQL Server 数据库 .....	369	12.8 合并 DataGridView 控件中的单元格 ...	390
11.7.5 基本功训练 5——判断计算机中是否 安装了 SQL 软件 .....	370	12.9 照猫画虎——基本功训练 .....	392
<b>11.8 情景应用——拓展与实践 .....</b>	<b>371</b>	12.9.1 基本功训练 1——设置 DataGridView 控件中网格线的样式.....	392
11.8.1 情景应用 1——向 SQL Server 数据库中 批量写入海量数据.....	371	12.9.2 基本功训练 2——在 DataGridView 控件 中设置数据显示格式.....	392
11.8.2 情景应用 2——使用断开式连接批量 更新数据库中数据.....	373	12.9.3 基本功训练 3——设置 DataGridView 控件单元格的文本对齐方式.....	393
11.8.3 情景应用 3——综合查询职工详细信息...	374	12.9.4 基本功训练 4——在 DataGridView 控件 中实现下拉列表.....	394
11.8.4 情景应用 4——使用二进制存取用户 头像.....	376	12.9.5 基本功训练 5——为 DataGridView 控件 实现复选功能.....	395
11.8.5 情景应用 5——使用存储过程实现员工 自动编号.....	378	<b>12.10 情景应用——拓展与实践 .....</b>	<b>396</b>
11.9 自我测试 .....	380	12.10.1 情景应用 1——在 DataGridView 控件 中验证数据输入.....	396
11.10 行动指南 .....	380	12.10.2 情景应用 2——在 DataGridView 控件 中添加“合计”和“平均值” .....	396
11.11 成功可以复制——微型博客 Twitter 创始人埃文·威廉姆斯 .....	381	12.10.3 情景应用 3——使用交叉表实现商品 销售统计.....	398
<b>第 12 堂课 DataGridView 数据控件 .....</b>	<b>383</b>	12.10.4 情景应用 4——将 DataGridView 中 数据导出到 Word.....	399
<b>视频讲解：103 分钟</b>		12.10.5 情景应用 5——通过 DataGridView 分页查看用户信息.....	401
12.1 DataGridView 控件概述 .....	384	12.11 自我测试 .....	403
12.2 在 DataGridView 控件中显示数据 .....	384	12.12 行动指南 .....	404
12.3 获取 DataGridView 控件中的当前 单元格 .....	385	12.13 成功可以复制——因特网的点火人 马克·安德森 .....	405
12.4 在 DataGridView 控件中修改数据 .....	386		
12.5 选中 DataGridView 控件中的行时 显示不同颜色 .....	387		
12.6 禁止在 DataGridView 控件中添加和			

## 第 3 部分

<b>第 13 堂课 面向对象编程高级技术 .....</b>	<b>409</b>
<b>视频讲解：100 分钟</b>	
13.1 接口的声明及实现 .....	410
13.1.1 接口概述.....	410

## 高级篇

13.1.2 接口的声明.....	410
13.1.3 接口的实现与继承.....	411
13.1.4 显式接口成员实现.....	412
13.2 抽象类的声明及使用 .....	413

13.2.1 抽象类概述.....	413	14.1.2 使用 var 创建隐型局部变量.....	436
13.2.2 抽象类的声明.....	414	14.1.3 Lambda 表达式的使用.....	437
13.2.3 抽象方法的声明.....	414	14.1.4 LINQ 查询表达式.....	438
13.2.4 抽象类的使用.....	414	14.2 LINQ 操作 SQL Server 数据库.....	439
13.2.5 抽象类与接口.....	416	14.2.1 使用 LINQ 查询 SQL Server 数据库.....	439
13.3 密封类的声明及使用 .....	416	14.2.2 使用 LINQ 管理 SQL Server 数据库.....	443
13.3.1 密封类概述.....	416	14.3 LINQ 操作其他数据.....	449
13.3.2 密封类的声明.....	416	14.3.1 使用 LINQ 操作数组和集合.....	449
13.3.3 密封方法的声明.....	417	14.3.2 使用 LINQ 操作 DataSet 数据集.....	450
13.3.4 密封类的使用.....	418	14.3.3 使用 LINQ 操作 XML.....	451
<b>13.4 照猫画虎——基本功训练 .....</b>	<b>419</b>	<b>14.4 照猫画虎——基本功训练 .....</b>	<b>454</b>
13.4.1 基本功训练 1——自定义抽象类计算 圆形的面积.....	419	14.4.1 基本功训练 1——检查序列中是否包含 指定元素.....	454
13.4.2 基本功训练 2——利用接口实现选择 不同的语言.....	420	14.4.2 基本功训练 2——使用 LINQ 生成随机 序列.....	455
13.4.3 基本功训练 3——使用接口作为方法 参数进行编程.....	421	14.4.3 基本功训练 3——统计每种商品的销售 次数.....	456
13.4.4 基本功训练 4——通过重写虚方法实现 加法运算.....	422	14.4.4 基本功训练 4——统计每种商品的销售 均价.....	457
13.4.5 基本功训练 5——使用多重继承实现 教师和学生信息的输出.....	422	14.4.5 基本功训练 5——获取有过返货记录的 商品列表.....	457
<b>13.5 情景应用——拓展与实践 .....</b>	<b>424</b>	<b>14.5 情景应用——拓展与实践 .....</b>	<b>458</b>
13.5.1 情景应用 1——使用迭代器显示公 交车站点.....	424	14.5.1 情景应用 1——使用存储过程查询单表 数据.....	458
13.5.2 情景应用 2——通过迭代器实现文字 的动态效果.....	425	14.5.2 情景应用 2——使用 LINQ 技术防止 SQL 注入式攻击.....	459
13.5.3 情景应用 3——使用分部类实现多种 计算方法.....	427	14.5.3 情景应用 3——使用 LINQ 技术实现 数据分页.....	460
13.5.4 情景应用 4——通过继承泛型类实现 输出学生信息.....	428	14.5.4 情景应用 4——从头开始提取满足指定 条件的记录.....	462
13.5.5 情景应用 5——使用密封类密封用户 信息.....	429	14.5.5 情景应用 5——读取 XML 文件并更新 到数据库.....	463
13.6 自我测试 .....	430	14.6 自我测试 .....	464
13.7 行动指南 .....	431	14.7 行动指南 .....	465
13.8 成功可以复制——征途巨人史玉柱 ..	432	14.8 成功可以复制——中国通信设备 行业的领跑者任正非 .....	466
<b>第 14 堂课 LINQ 技术的使用 .....</b>	<b>435</b>	<b>第 15 堂课 文件及 IO .....</b>	<b>469</b>
<u>  视频讲解：124 分钟</u>		<u>  视频讲解：171 分钟</u>	
14.1 LINQ 基础.....	436	15.1 文件操作基础 .....	470
14.1.1 LINQ 概述 .....	436		

15.1.1 File 类和 FileInfo 类介绍.....	470	第 16 堂课 GDI+绘图技术 .....	501
15.1.2 Directory 类和 DirectoryInfo 类介绍 .....	472	 视频讲解：145 分钟	
15.2 文件基本操作 .....	475	16.1 GDI+绘图基础 .....	502
15.2.1 判断文件是否存在.....	475	16.1.1 GDI+概述 .....	502
15.2.2 创建文件.....	475	16.1.2 创建 Graphics 对象 .....	502
15.2.3 复制文件.....	476	16.1.3 创建 Pen 对象 .....	503
15.2.4 移动文件.....	477	16.1.4 创建 Brush 对象 .....	503
15.2.5 删除文件.....	477	16.2 基本图形绘制 .....	504
15.3 文件夹基本操作 .....	478	16.2.1 绘制直线和矩形.....	505
15.3.1 判断文件夹是否存在.....	478	16.2.2 绘制椭圆、弧和扇形.....	506
15.3.2 创建文件夹.....	478	16.2.3 绘制多边形 .....	508
15.3.3 移动文件夹.....	479	16.2.4 绘制文本 .....	509
15.3.4 删除文件夹.....	479	16.2.5 绘制图形 .....	510
15.4 I/O 输入输出 .....	480	16.3 猫画虎——基本功训练 .....	510
15.4.1 流概述.....	480	16.3.1 基本功训练 1——绘制公章 .....	510
15.4.2 文件 I/O 流介绍 .....	481	16.3.2 基本功训练 2——波形图的绘制 .....	512
15.4.3 使用 I/O 流操作文本文件 .....	482	16.3.3 基本功训练 3——生成图片缩略图 .....	513
15.4.4 使用 I/O 流操作二进制文件 .....	484	16.3.4 基本功训练 4——以任意角度旋转图像 .....	514
15.5 猫画虎——基本功训练 .....	486	16.3.5 基本功训练 5——浮雕效果显示图像 .....	515
15.5.1 基本功训练 1——获取文件基本信息 .....	486	16.4 情景应用——拓展与实践 .....	516
15.5.2 基本功训练 2——遍历文件夹 .....	487	16.4.1 情景应用 1——绘制中文验证码 .....	516
15.5.3 基本功训练 3——使用 C#操作 INI 文件 .....	488	16.4.2 情景应用 2——批量图像格式转换 .....	517
15.5.4 基本功训练 4——按行读取文本文件中 数据 .....	489	16.4.3 情景应用 3——抓取网站整页面 .....	520
15.5.5 基本功训练 5——获取指定文件夹的 上级目录 .....	490	16.4.4 情景应用 4——批量添加图片水印 .....	523
15.6 情景应用——拓展与实践 .....	490	16.4.5 情景应用 5——打造自己的开心农场 .....	527
15.6.1 情景应用 1——根据日期动态建立文件 .....	490	16.5 自我测试 .....	530
15.6.2 情景应用 2——文件批量更名 .....	491	16.6 行动指南 .....	531
15.6.3 情景应用 3——复制文件时显示复制 进度 .....	492	16.7 成功可以复制——“盖茨第二” 马克·扎克伯格 .....	531
15.6.4 情景应用 4——伪装文件夹 .....	494		
15.6.5 情景应用 5——对指定文件夹中的文件 进行分类存储 .....	496		
15.7 自我测试 .....	497	第 17 堂课 水晶报表与打印 .....	533
15.8 行动指南 .....	498	 视频讲解：137 分钟	
15.9 成功可以复制——中国网络游戏 产业的领军人陈天桥 .....	499		

17.3 水晶报表基本操作 .....	536	18.1.1 局域网与因特网介绍.....	572
17.3.1 创建水晶报表并连接数据源.....	536	18.1.2 网络协议介绍.....	572
17.3.2 水晶报表中数据的分组与排序.....	538	18.1.3 端口及套接字介绍.....	574
17.3.3 水晶报表中数据的筛选.....	540	18.2 网络编程基础 .....	575
17.3.4 在水晶报表中使用图表.....	542	18.2.1 System.NET 命名空间及相关类的使用 ...	575
17.3.5 在水晶报表中创建子报表.....	543	18.2.2 System.NET.Sockets 命名空间及相关 类的使用 .....	580
17.4 Windows 打印组件的使用 .....	545	18.2.3 System.NET.Mail 命名空间及相关类的 使用 .....	585
17.4.1 使用 PageSetupDialog 组件设置打印 文档信息.....	545	18.3 照猫画虎——基本功训练 .....	587
17.4.2 使用 PrintDialog 组件显示打印对话框 ...	546	18.3.1 基本功训练 1——通过 IP 地址获取 主机名称.....	587
17.4.3 使用 PrintDocument 组件设置打印文档... 17.4.4 使用 PrintPreviewControl 组件设置 打印预览文档.....	547	18.3.2 基本功训练 2——得到本机 MAC 地址 ...	588
17.4.5 使用 PrintPreviewDialog 组件显示打印 预览 .....	549	18.3.3 基本功训练 3——获取网络中所有工作 组名称.....	588
<b>17.5 照猫画虎——基本功训练 .....</b>	<b>549</b>	18.3.4 基本功训练 4——获取网络中某台计 算机的磁盘信息.....	589
17.5.1 基本功训练 1——自定义横向打印 .....	549	18.3.5 基本功训练 5——编程实现 Ping 操作... 18.4 情景应用——拓展与实践 .....	590
17.5.2 基本功训练 2——自定义打印页码范围 .	551	18.4.1 情景应用 1——获取网络信息及流量 ....	591
17.5.3 基本功训练 3——打印商品入库单据 ....	556	18.4.2 情景应用 2——远程关闭与重启计算机... 18.4.3 情景应用 3——创建 Web 页面浏览器 ...	593
17.5.4 基本功训练 4——使图片成为整个报表 的背景.....	557	18.4.4 情景应用 4——设计点对点聊天程序 ....	594
17.5.5 基本功训练 5——设置水晶报表的打印 日期与时间.....	558	18.4.5 情景应用 5——电子邮件的发送与接收... 18.5 自我测试 .....	596
<b>17.6 情景应用——拓展与实践 .....</b>	<b>560</b>	18.5 自我测试 .....	598
17.6.1 情景应用 1——打印学生个人简历 .....	560	18.6 行动指南 .....	598
17.6.2 情景应用 2——批量打印学生证书 .....	561	18.7 成功可以复制——80 后新贵、 泡泡网 CEO 李想 .....	599
17.6.3 情景应用 3——订货总金额超过 10 万元 显示“恭喜获奖”文字 .....	564		
17.6.4 情景应用 4——部门销售量占公司总 销售量的业绩百分比.....	566		
17.6.5 情景应用 5——按类别分组统计图 书库存 .....	567		
17.7 自我测试 .....	568		
17.8 行动指南 .....	569		
17.9 成功可以复制——3D 王国的开创者 约翰·沃克 .....	569		
<b>第 18 堂课 网络编程 .....</b>	<b>571</b>		
<b>视频讲解：152 分钟</b>			
18.1 计算机网络基础 .....	572		
18.1.1 局域网与因特网介绍.....	572		
18.1.2 网络协议介绍.....	572		
18.1.3 端口及套接字介绍.....	574		
18.2 网络编程基础 .....	575		
18.2.1 System.NET 命名空间及相关类的使用 ...	575		
18.2.2 System.NET.Sockets 命名空间及相关 类的使用 .....	580		
18.2.3 System.NET.Mail 命名空间及相关类的 使用 .....	585		
18.3 照猫画虎——基本功训练 .....	587		
18.3.1 基本功训练 1——通过 IP 地址获取 主机名称.....	587		
18.3.2 基本功训练 2——得到本机 MAC 地址 ...	588		
18.3.3 基本功训练 3——获取网络中所有工作 组名称.....	588		
18.3.4 基本功训练 4——获取网络中某台计 算机的磁盘信息.....	589		
18.3.5 基本功训练 5——编程实现 Ping 操作... 18.4 情景应用——拓展与实践 .....	590		
18.4.1 情景应用 1——获取网络信息及流量 ....	591		
18.4.2 情景应用 2——远程关闭与重启计算机... 18.4.3 情景应用 3——创建 Web 页面浏览器 ...	593		
18.4.4 情景应用 4——设计点对点聊天程序 ....	594		
18.4.5 情景应用 5——电子邮件的发送与接收... 18.5 自我测试 .....	596		
18.5 自我测试 .....	598		
18.6 行动指南 .....	598		
18.7 成功可以复制——80 后新贵、 泡泡网 CEO 李想 .....	599		
<b>第 19 堂课 线程的使用 .....</b>	<b>601</b>		
<b>视频讲解：142 分钟</b>			
19.1 线程概述 .....	602		
19.1.1 线程的定义与分类.....	602		
19.1.2 多线程的使用.....	603		
19.1.3 线程的生命周期.....	603		
19.2 C#中的线程类 Thread .....	604		
19.3 线程调度 .....	606		
19.3.1 创建线程.....	606		
19.3.2 线程的挂起与恢复.....	607		

19.3.3 线程休眠.....	608	19.9 成功可以复制——IT “大王” 王志东 .....	627
19.3.4 终止线程.....	608		
19.3.5 线程的优先级.....	609		
<b>19.4 线程同步 .....</b>	<b>611</b>		
19.4.1 线程同步机制.....	611		
19.4.2 使用 lock 关键字实现线程同步 .....	611		
19.4.3 使用 Monitor 驱动对象实现线程同步 .....	612		
19.4.4 使用 Mutex 类实现线程同步 .....	613		
<b>19.5 照猫画虎——基本功训练 .....</b>	<b>615</b>		
19.5.1 基本功训练 1——判断线程的运行 状态.....	615		
19.5.2 基本功训练 2——使用线程遍历 文件夹.....	615		
19.5.3 基本功训练 3——使用线程休眠控制 图片以百叶窗效果显示.....	616		
19.5.4 基本功训练 4——使用线程读取数据库 中的数据.....	618		
19.5.5 基本功训练 5——使用线程实现大容量 数据的计算.....	618		
<b>19.6 情景应用——拓展与实践 .....</b>	<b>619</b>		
19.6.1 情景应用 1——使用线程扫描局域网 IP 地址.....	619		
19.6.2 情景应用 2——使用线程制作小游戏 .....	621		
19.6.3 情景应用 3——有进度条的文件异步 复制功能.....	622		
19.6.4 情景应用 4——使用线程控制向窗体中 拖放图片并显示.....	624		
19.6.5 情景应用 5——使用多线程制作端口 扫描工具.....	625		
19.7 自我测试 .....	626		
19.8 行动指南 .....	627		
<b>第 20 堂课 异常处理与程序调试 .....</b>	<b>629</b>		
		<b>■ 视频讲解：30 分钟</b>	
20.1 异常处理与程序调试概述 .....	630		
20.2 异常处理语句的使用 .....	630		
20.2.1 使用 throw 语句抛出异常 .....	630		
20.2.2 使用 try...catch 语句捕捉异常 .....	632		
20.2.3 使用 try...catch...finally 语句捕捉 异常 .....	633		
20.3 常用的程序调试操作 .....	634		
20.3.1 断点操作 .....	634		
20.3.2 开始、中断和停止程序的执行 .....	635		
20.3.3 单步执行 .....	636		
20.3.4 运行到指定位置 .....	637		
20.4 成功可以复制——IT 风云人物 鲍岳桥 .....	637		
<b>第 21 堂课 Windows 应用程序打包部署 .....</b>	<b>639</b>		
		<b>■ 视频讲解：14 分钟</b>	
21.1 Windows Installer 介绍 .....	640		
21.2 创建 Windows 安装项目 .....	640		
21.3 制作 Windows 安装程序 .....	641		
21.3.1 添加项目输出 .....	641		
21.3.2 添加内容文件 .....	642		
21.3.3 创建桌面快捷方式 .....	643		
21.3.4 添加注册表项 .....	644		
21.3.5 生成 Windows 安装程序 .....	646		
21.4 部署 Windows 应用程序 .....	646		
21.5 成功可以复制——暴雪公司的 领航者迈克·莫汉 .....	647		

## 第 4 部分 实战篇

<b>第 22 堂课 企业人事管理系统 .....</b>	<b>651</b>		
		<b>■ 视频讲解：150 分钟</b>	
22.1 系统分析 .....	652		
22.1.1 需求分析 .....	652		
22.2 系统设计 .....	654		
22.2.1 系统目标 .....	654		
		22.1.2 可行性分析 .....	652
		22.1.3 编写项目计划书 .....	653

22.2.2 系统功能结构.....	655	22.12.1 设计用户设置窗体.....	705
22.2.3 系统业务流程图.....	655	22.12.2 添加/修改用户信息.....	705
22.2.4 系统编码规范.....	656	22.12.3 删除用户基本信息.....	707
22.3 系统运行环境 .....	656	22.12.4 设置用户操作权限.....	707
22.4 数据库与数据表设计 .....	656	22.13 数据库维护模块设计 .....	708
22.4.1 数据库分析.....	657	22.13.1 设计数据库维护窗体.....	708
22.4.2 创建数据库.....	657	22.13.2 备份数据库.....	709
22.4.3 创建数据表.....	659	22.13.3 还原数据库.....	709
22.4.4 数据表逻辑关系.....	663	22.14 运行项目 .....	711
22.5 创建项目 .....	664	22.15 系统打包部署 .....	712
22.6 公共类设计 .....	665	22.16 开发常见问题与解决 .....	715
22.6.1 MyMeans 公共类 .....	665	22.16.1 程序为什么会无法运行.....	715
22.6.2 MyModule 公共类.....	667	22.16.2 为什么无法添加职工基本信息.....	716
22.7 登录模块设计 .....	679	22.16.3 选择职工头像时出现异常怎么办.....	717
22.7.1 设计登录窗体.....	680	22.16.4 数据库还原不成功应该如何解决.....	717
22.7.2 按回车键时移动鼠标焦点.....	680	22.16.5 出现 Word 引用问题怎么办.....	717
22.7.3 登录功能的实现.....	680	22.16.6 COM 选项卡中没有 Word 9.0 引用 怎么办.....	718
22.8 系统主窗体设计 .....	681	22.16.7 为什么使用全名声明 Word 对象后 还出现错误.....	718
22.8.1 设计菜单栏.....	682	22.17 小结 .....	722
22.8.2 设计工具栏.....	683		
22.8.3 设计导航菜单.....	684		
22.8.4 设计状态栏.....	684		
22.9 人事档案管理模块设计 .....	685	<b>特别提醒：</b>	
22.9.1 设计人事档案管理窗体.....	686	亲爱的读者朋友，由于近期纸张价格和印制成本大幅 上涨，为不增加读者朋友的负担，又不减少书的内容，本 书以下章节内容不得不放在配书光盘中，由此给您带来了 不便，在此深表歉意。	
22.9.2 添加/修改人事档案信息.....	688		
22.9.3 删除人事档案信息.....	690		
22.9.4 单条件查询人事档案信息.....	690		
22.9.5 逐条查看人事档案信息.....	692		
22.9.6 将人事档案信息导出为 Word 文档.....	694		
22.10 人事资料查询模块设计 .....	698		
22.10.1 设计人事资料查询窗体.....	699		
22.10.2 多条件查询人事资料.....	700		
22.11 通讯录模块设计 .....	700		
22.11.1 设计通讯录窗体.....	701		
22.11.2 添加/修改通讯录信息.....	701		
22.11.3 删除通讯录信息.....	703		
22.11.4 查询通讯录信息.....	704		
22.12 用户设置模块设计 .....	704		
		<b>第 23 堂课 房屋中介管理系统 .....</b>	<b>723</b>
		<b>■ 视频讲解：11 分钟</b>	
		<b>(本章内容在配书光盘中)</b>	
23.1 系统分析 .....	724		
23.2 系统设计 .....	724		
23.2.1 系统目标.....	724		
23.2.2 系统功能结构图.....	724		
23.2.3 系统业务流程图.....	725		
23.3 系统运行环境 .....	725		
23.4 数据库与数据表设计 .....	725		
23.4.1 数据库分析.....	725		
23.4.2 数据表设计.....	726		

23.4.3 视图设计 .....	728	24.10 进货管理模块设计 .....	745
23.4.4 存储过程设计 .....	729	24.11 商品销售排行模块设计 .....	746
23.4.5 触发器设计 .....	731	24.12 库存状况管理模块设计 .....	746
23.5 创建项目 .....	732	24.13 库存盘点模块设计 .....	747
23.6 公共类设计 .....	733	24.14 小结 .....	747
23.7 系统主窗体设计 .....	733		
23.8 用户信息管理模块设计 .....	733		
23.9 房源信息设置模块设计 .....	734		
23.10 房源信息查询模块设计 .....	734		
23.11 房源状态查询模块设计 .....	735		
23.12 员工信息设置模块设计 .....	735		
23.13 小结 .....	736		
<b>第 24 堂课 进销存管理系统 .....</b>	<b>737</b>		
<b>■ 视频讲解：13 分钟</b>			
(本章内容在配书光盘中)			
24.1 系统分析 .....	738	A.1 命名规则 .....	749
24.2 系统设计 .....	738	A.1.1 数据类型简写规则 .....	749
24.2.1 系统目标 .....	738	A.1.2 变量及对象名称命名规则 .....	749
24.2.2 系统功能结构图 .....	738	A.1.3 类命名规则 .....	750
24.2.3 系统业务流程图 .....	739	A.1.4 属性命名规则 .....	750
24.3 系统运行环境 .....	739	A.1.5 方法命名规则 .....	750
24.4 数据库与数据表设计 .....	739	A.1.6 接口命名规则 .....	750
24.4.1 数据库分析 .....	740	A.1.7 控件命名规则 .....	751
24.4.2 数据表设计 .....	740	A.2 代码版式 .....	751
24.4.3 数据表逻辑关系 .....	742	A.2.1 统一代码缩进 .....	751
24.5 创建项目 .....	743	A.2.2 合理使用空行 .....	751
24.6 公共类设计 .....	744	A.2.3 代码换行 .....	752
24.7 登录模块设计 .....	744	A.2.4 空格的使用 .....	752
24.8 系统主窗体设计 .....	744	A.3 代码注释 .....	752
24.9 基础数据管理模块设计 .....	745	A.3.1 注释的目的 .....	752
		A.3.2 代码注释规范 .....	752
		A.4 数据库编程命名规范 .....	753
		A.4.1 数据库命名规范 .....	753
		A.4.2 数据表命名规范 .....	753
		A.4.3 字段命名规范 .....	753
		A.4.4 视图命名规范 .....	753
		A.4.5 存储过程命名规范 .....	754
		A.4.6 触发器命名规范 .....	754

# 第1部分

## 基础篇

- ▶ 第1堂课 初探C#及其开发环境
- ▶ 第2堂课 C#程序设计基础
- ▶ 第3堂课 程序流程控制
- ▶ 第4堂课 字符及字符串的使用
- ▶ 第5堂课 数组与集合
- ▶ 第6堂课 程序设计中的算法
- ▶ 第7堂课 面向对象程序设计



# 第 1 堂课

## 初探 C#及其开发环境

(  视频讲解：96分钟 )

C#是微软公司推出的一种语法简洁、类型安全的面向对象的编程语言，开发人员可以通过它编写在.NET Framework 上运行的各种安全可靠的应用程序。本书中涉及的程序都是通过 Visual Studio 2008 开发环境编译的，Visual Studio 2008 开发环境是目前开发 C#应用程序最好的工具。本堂课将详细介绍 C#语言的相关内容，并且通过图文并茂的形式介绍安装与卸载 Visual Studio 2008 开发环境及其 MSDN 帮助的全过程。

学习摘要：

- 了解.NET 及其编译原理
- 了解 C#语言的特点及其发展趋势
- 掌握如何安装与卸载 Visual Studio 2008 开发环境
- 熟悉 Visual Studio 2008 开发环境
- 掌握如何创建控制台应用程序和 Windows 应用程序
- 掌握如何安装 Visual Studio 2008 帮助系统
- 掌握 MSDN 的使用

## 1.1 了解.NET

.NET 是在互联网环境中连接各种信息、人员、系统与设备的软件架构，它的最终目标就是形成互联网形式的操作系统，本节将对.NET 的发展历史及其编译原理进行介绍。

### 1.1.1 .NET 概述

.NET 是一个全新的跨语言开发平台，它改进了 Windows 中应用程序的开发与部署。.NET 平台的实现目标如下。

- (1) 提供一个一致的、面向对象的编程环境，无论代码是在本地执行还是分布在 Internet 上均无影响。
- (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) 提供一个能够提高代码执行安全性的代码执行环境。
- (4) 使开发人员的经验在面对类型不同的应用程序（如基于 Windows 的应用程序和基于 Web 的应用程序）时保持一致。
- (5) 按照工业标准生成所有通信，以确保基于.NET Framework 的代码可与任何其他代码集成。

.NET Framework 3.5 是在.NET Framework 1.0、1.1 和 2.0 成功的基础上构建的，用于为 Web 和 Microsoft Windows 客户端应用程序提供最佳运行效果的运行库环境。对于.NET Framework 2.0 应用程序，微软的兼容性目标是这些应用程序能够在.NET Framework 3.5 上顺利运行。.NET Framework 3.5 包括了用户运行使用.NET Framework 开发的应用程序时所需的所有内容。

.NET 版本发展历史如图 1.1 所示。

**说明：**微软公司曾在 2006 年 11 月 6 日发布过.NET 3.0 版本，但由于该版本是基于.NET 2.0 运行的，并且没有相应的开发环境，所以图 1.1 中没有体现。

### 1.1.2 .NET 程序编译原理

.NET 平台中代码的物理单元是可移植可执行程序（Portable Executable，PE）格式，编译程序和库时，和平常一样生成 EXE 与 DLL 文件，但在.NET 框架下，任何可执行程序项目都链接公共语言运行库，并由它代理编译和执行。

.NET 中程序编译的最主要部分是汇编（assembly），其包括一个 manifest，它是一组元数据，标识汇编向其他应用程序提供的文件和类型，另外，manifest 还可以包含强名称（strong name）、组合汇编名、版本信息和可选文化信息。在使用强名称的汇编中，元数据包含一个公用密钥签名，公共语言运行库用其验证汇编自编译之后是否发生改变。

汇编还可以包括数字签名，如用验证码（Authenticode certificate）验证代码源，以解决在公共语言运行库保证汇编签名的真实性。

汇编分为专用汇编和共享汇编两种。专用汇编只在安装这个汇编的应用程序中使用，而共享汇编则安

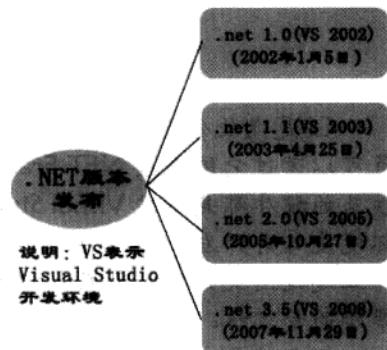


图 1.1 .NET 版本发展历史

装在全局汇编缓冲区（Global Assembly Cache，GAC）中，全局汇编缓冲区用汇编的引用计数和版本信息管理库，包括框架的基类库（Base Class Library，BCL），并通过公共语言运行库避免因库的版本冲突而形成 DLL Hell 的情形。

选择专用汇编和共享汇编是一个重要决策。将汇编安装到全局缓冲区后，就可以在多个应用程序中使用汇编的一个备份，但是还需要用 Windows Installer 之类的安装程序将应用程序安装到计算机上。如果使用专用汇编，则得到 Microsoft 所谓的“XCOPY 部署”，这样用户可以将应用程序复制到目标计算机上，方便地进行安装。

.NET 程序编译过程如图 1.2 所示。

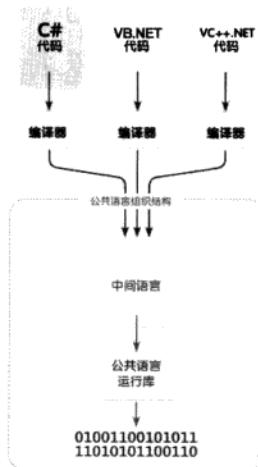


图 1.2 .NET 程序编译过程

### 1.1.3 .NET 项目成功案例

.NET 作为微软全力推出的一个崭新的平台，经过最近几年的发展，在实际生活中已经有了很多成功的项目案例，如世界饮食行业的龙头之一 KFC、中国最成功的游戏之一问道、中华人民共和国人事部以及中国最著名的汽车生产商之一东风汽车公司等，它们的官方网站都使用了.NET 平台，并使用 C# 语言来实现，它们的官方网站首页分别如图 1.3~图 1.6 所示。



图 1.3 KFC 官方网站



图 1.4 《问道》游戏官方网站



图 1.5 中华人民共和国人事部官方网站



图 1.6 东风汽车公司官方网站

## 1.2 C#语言及特点

C#是一种面向对象的编程语言，主要用于开发可以运行在.NET 平台上的应用程序。C#的语言体系都构建在.NET 框架上，近几年 C#呈现逐年上升趋势，这也说明了 C#语言的简单、现代、面向对象和类型安全等特点正在被更多人所认同。

### 1.2.1 C#与.NET 的关系

.NET 框架是微软公司推出的一个全新的编程平台，目前的版本是 3.5。C#是专门为与微软公司的.NET Framework 一起使用而设计的（.NET Framework 是一个功能非常丰富的平台，可开发、部署和执行分布式应用程序），就其本身而言其只是一种语言，尽管它是用于生成面向.NET 环境的代码，但它本身并不是.NET 的一部分。.NET 支持的一些特性，C#并不支持，而 C#语言支持的另一些特性，.NET 也不支持（如运算符重载）。在安装 Visual Studio 2008 的同时，.NET Framework 3.5 也被安装到本地计算机中。

C#与.NET 的关系如图 1.7 所示。

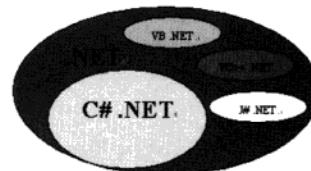


图 1.7 C#与.NET 的关系图

### 1.2.2 C#语言特点

C#是一种面向对象的编程语言，主要用于开发可以在.NET 平台上运行的应用程序，其语言体系都构建在.NET 框架上。C#是从 C 和 C++派生来的一种简单、现代、面向对象和类型安全的编程语言，并且能够与.NET 框架完美结合，其具有以下突出特点。

- (1) 语法简洁，不允许直接操作内存，去掉了指针操作。
- (2) 彻底地面向对象设计，具有面向对象语言所应有的一切特性，如封装、继承和多态。
- (3) 与 Web 紧密结合，支持绝大多数的 Web 标准，如 HTML、XML、SOAP 等。
- (4) 强大的安全性机制，可以消除软件开发中的常见错误（如语法错误），.NET 提供的垃圾回收器能够帮助开发者有效地管理内存资源。
- (5) 兼容性，C#遵循.NET 的公共语言规范（CLS），能够保证与其他语言开发的组件兼容。
- (6) 灵活的版本处理技术，C#语言本身内置了版本控制功能，可以使开发人员更加容易地进行开发和

维护。

(7) 完善的错误、异常处理机制, C#提供了完善的错误和异常处理机制, 使程序在交付应用时能够更加健壮。

### 1.2.3 C#语言发展趋势

根据2009年3月Tiobe编程语言排行榜可以看出, C#语言继续其上升趋势, 同比上升一位, 排在了所有语言的第7位, 而且发展势头良好。如图1.8所示为2009年3月Tiobe编程语言排行榜。

C#语言自从2002年就随着Visual Studio一起被推出, 其发展趋势如图1.9所示。

Position Mar 2009	Position Mar 2008	Delta in Position	Programming Language	Status
1	1	==	Java	A
2	2	==	C	A
3	5	↑↑	C++	A
4	4	==	PHP	A
5	3	↓↓	(Visual) Basic	A
6	7	↑↑	Python	A
7	8	↑↑	C#	A
8	10	↑↑	JavaScript	A
9	6	↓↓	Perl	A
10	9	↓	Delphi	A

图1.8 2009年3月Tiobe编程语言排行榜

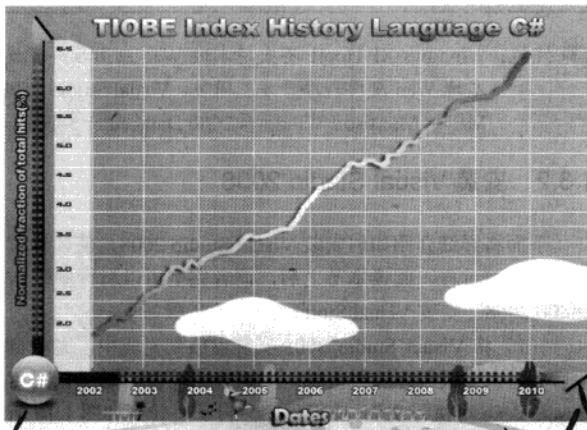


图1.9 C#语言发展趋势

**说明:** 从图1.9中可以看出, C#自从2002年被正式发布以来, 一直呈现稳定的上升趋势, 而且作为微软全力推广的一种语言, 它的发展前景也非常好。

## 1.3 安装与卸载Visual Studio 2008

Visual Studio 2008是微软为了配合.NET战略推出的IDE开发环境, 同时, 它也是目前开发C#应用程序最好的工具, 本节将对Visual Studio 2008的安装与卸载进行详细讲解。

### 1.3.1 安装Visual Studio 2008系统必备

安装Visual Studio 2008之前, 首先要了解安装Visual Studio 2008所需的必备条件, 检查计算机的软硬件配置是否满足Visual Studio 2008开发环境的安装要求, 具体要求如表1.1所示。

表1.1 安装Visual Studio 2008所需的必备条件

软硬件	描述
处理器	600MHz处理器, 建议使用1GHz处理器
RAM	512MB内存, 建议使用1024MB内存

续表

软 硬 件	描 述
可用硬盘空间	如果不安装 MSDN，则系统驱动器上需要 1GB 的可用空间，安装驱动器上需要 2GB 的可用空间；如果安装 MSDN，则系统驱动器上需要 1GB 的可用空间，完整安装 MSDN 的安装驱动器上需要 3.8GB 的可用空间，默认安装 MSDN 的安装驱动器上需要 2.8GB 的可用空间
CD-ROM 或 DVD-ROM 驱动器	必须使用
显示器	分辨率：800×600, 256 色，建议使用 1024×768, 增强色 16 位
操作系统及所需补丁	Windows® 2000 Service Pack 4、Windows XP Service Pack 2、Windows Server 2003 Service Pack 1 或更高版本

◆ 注意：Windows XP Home 不支持本地 Web 应用程序开发，只有在 Windows 专业版和服务器版中才支持本地 Web 应用程序开发。同时，Visual Studio 2008 还不支持 Windows 95、Windows 98、Windows ME 和 Microsoft Windows 2000 Datacenter Server 等平台。

### 1.3.2 安装 Visual Studio 2008

下面将详细介绍如何安装 Visual Studio 2008，使读者掌握每一步的安装过程，阅读本节之后，读者完全可以自行安装 Visual Studio 2008。安装 Visual Studio 2008 的步骤如下。

(1) 将 Visual Studio 2008 安装盘放到光驱中，光盘自动运行后会进入安装程序文件界面，如果光盘不能自动运行，可以双击 setup.exe 可执行文件，应用程序会自动跳转到如图 1.10 所示的“Visual Studio 2008 安装程序”界面。该界面上有 3 个安装选项，即安装 Visual Studio 2008、安装产品文档和检查 Service Release，一般情况下需安装前两项。

(2) 单击“安装 Visual Studio 2008”选项，弹出如图 1.11 所示的“Visual Studio 2008 安装向导”界面。

(3) 单击“下一步”按钮，弹出如图 1.12 所示的“Visual Studio 2008 安装程序-起始页”界面，该界面左边显示关于 Visual Studio 2008 安装程序所需的组件信息，右边显示用户许可协议。



图 1.10 Visual Studio 2008 安装界面



图 1.11 Visual Studio 2008 安装向导

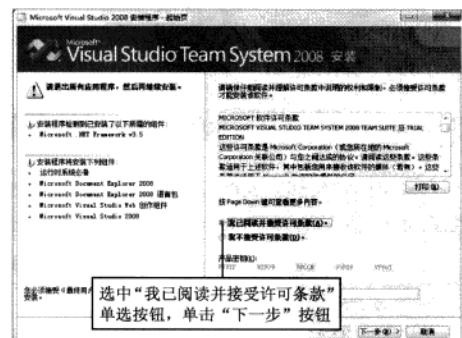


图 1.12 Visual Studio 2008 安装程序-起始页

(4) 选中“我已阅读并接受许可条款”单选按钮，单击“下一步”按钮，弹出如图1.13所示的“Visual Studio 2008安装程序-选项页”界面，用户可以选择要安装的功能和产品安装路径，一般使用默认设置即可，产品默认路径为“C:\Program Files\Microsoft Visual Studio 9.0”。

**说明：**在选择安装选项页中，用户可以选择“默认值”、“完全”和“自定义”3种方式之一。如果选择“默认值”，安装程序会安装系统必备的功能；如果选择“完全”，安装程序会安装系统的所有功能；如果选择“自定义”，用户可以选择希望安装的项目，增加了安装程序的灵活性。通常，用户直接选择“默认值”即可。

(5) 选择好产品安装路径之后单击“安装”按钮，进入如图1.14所示的“Visual Studio 2008安装程序-安装页”界面，显示正在安装组件。

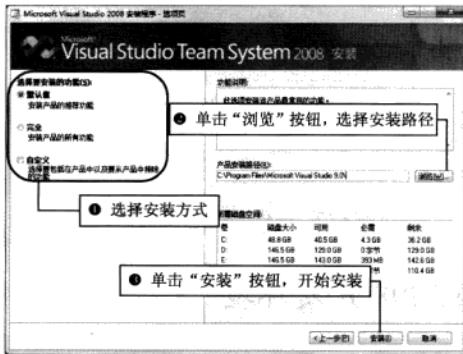


图1.13 Visual Studio 2008 安装程序-选项页



图1.14 Visual Studio 2008 安装程序-安装页

(6) 安装完毕后单击“下一步”按钮，弹出如图1.15所示的“Visual Studio 2008安装程序-完成页”界面，单击“完成”按钮，至此，Visual Studio 2008程序开发环境安装完成。

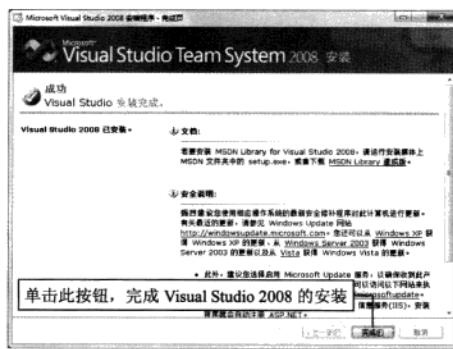


图1.15 Visual Studio 2008 安装程序-完成页

### 1.3.3 卸载Visual Studio 2008

如果想卸载Visual Studio 2008，则可以按以下步骤进行。

(1) 在 Windows 7 操作系统中, 打开“控制面板” / “程序” / “程序和功能”, 在打开的窗口中选中“Microsoft Visual Studio Team System 2008 Team Suite-简体中文”选项, 如图 1.16 所示。

(2) 选中“Microsoft Visual Studio Team System 2008 Team Suite-简体中文”选项后, 单击“卸载/更改”按钮, 弹出“Microsoft Visual Studio 2008 安装程序-维护页”界面, 如图 1.17 所示。单击“卸载 Microsoft Visual Studio 2008”链接进行卸载。



图 1.16 添加或删除程序



图 1.17 卸载 Visual Studio 2008

## 1.4 熟悉 Visual Studio 2008 开发环境

Visual Studio 2008 是一套完整的开发工具集, 用于生成 Windows 桌面应用程序、ASP.NET Web 应用程序、XML Web Services 和移动应用程序, 它提供了在设计、开发、调试和部署 Windows 应用程序、Web 应用程序、XML Web Services 和传统的客户端应用程序时所需的工具。本节将对 Visual Studio 2008 开发环境进行详细介绍。

### 1.4.1 创建控制台应用程序

Visual Studio 2008 中包含的项目主要分为控制台应用程序和 Windows 应用程序, 控制台应用程序是 Windows 系统组件的一部分, 而 Windows 应用程序是指可以在 Windows 平台上运行的所有程序。下面分别介绍控制台应用程序和 Windows 应用程序的创建过程。

创建控制台应用程序的步骤如下。

(1) 选择“开始” / “所有程序” / Microsoft Visual Studio 2008/Microsoft Visual Studio 2008 命令, 如果用户是第一次使用 Visual Studio 2008 开发环境, 则会弹出如图 1.18 所示的“选择默认环境设置”对话框。

(2) 在图 1.18 所示的对话框中选择“Visual C# 开发设置”选项, 单击“启动 Visual Studio”按钮即可进入 Visual Studio 2008 开发环境起始页, 如图 1.19 所示。

(3) 启动 Visual Studio 2008 开发环境之后, 可以通过两种方法创建项目: 一种是在菜单栏中选择“文件” / “新建” / “项目”命令, 另一种是通过选择“起始页” / “最近的项目” / “创建” / “项目”命令, 如图 1.20 所示。



图 1.18 选择默认环境设置



图 1.19 Visual Studio 2008 开发环境起始页



图 1.20 选择新建项目

选择其中一种方法创建项目，弹出如图 1.21 所示的“新建项目”对话框。

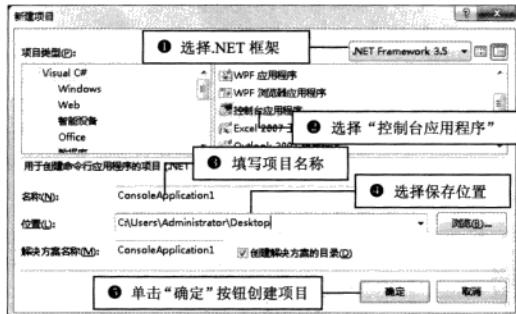


图 1.21 新建项目

(4) 选择要使用的.NET 框架和“控制台应用程序”后，用户可对所要创建的控制台应用程序进行命名、选择存放位置、是否创建解决方案目录的设定，在命名时可以使用用户自定义的名称，也可使用默认名 ConsoleApplication1，用户可以单击“浏览”按钮设置项目存放的位置（需要注意的是，解决方案名称与项目名称必须要统一），然后单击“确定”按钮，完成控制台应用程序的创建。

#### 1.4.2 创建 Windows 应用程序

创建 Windows 应用程序的步骤如下。

(1) 选择“开始” / “所有程序” / Microsoft Visual Studio 2008 / Microsoft Visual Studio 2008 命令，进入 Visual Studio 2008 开发环境，如图 1.19 所示。

(2) 在菜单栏中选择“文件” / “新建” / “项目”命令，弹出如图 1.22 所示的“新建项目”对话框。

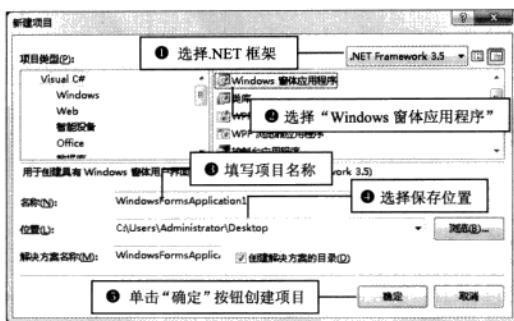


图 1.22 新建项目

(3) 选择要使用的.NET 框架和“Windows 窗体应用程序”后，用户可对所要创建的 Windows 窗体应用程序进行命名、选择存放位置、是否创建解决方案目录的设定，在命名时可以使用用户自定义的名称，也可使用默认名 WindowsFormsApplication1，用户可以单击“浏览”按钮设置项目存放的位置（需要注意的是，解决方案名称与项目名称必须要统一），然后单击“确定”按钮，完成 Windows 窗体应用程序的创建。

### 1.4.3 菜单栏介绍

菜单栏显示了所有可用的命令，通过鼠标单击可以执行菜单命令，也可以通过按 Alt 键加上菜单项上的字母执行菜单命令。常用的菜单命令及其作用如表 1.2 所示。

表 1.2 常用菜单命令及作用

菜单项	菜单命令	功能
文件	新建	建立一个新的项目、网站、文件等
	打开	打开一个已经存在的项目、文件等
	添加	添加一个项目到当前所编辑的项目中
	关闭	关闭当前页面
	关闭解决方案	关闭当前解决方案
	保存 Form1	保存项目中的当前窗体
	Form1 另存为	将项目中当前窗体换名或改变路径保存
	全部保存	将项目中所有文件保存
	导出模板	将当前项目作为模板保存起来，生成.zip 文件
	页面设置	设置打印机及打印属性
	打印	打印选择的指定内容
	最近的文件	打开最近操作的文件（如类文件）
	最近的文档	打开最近操作的文件（如解决方案）
	退出	退出集成开发环境

续表

菜单项	菜单命令	功能
编辑	撤销	撤销上一步操作
	重复	重做上一步所作的修改
	撤销上次全局操作	撤销上一步全局操作
	重复上次全局操作	重做上一步所作的全局修改
	剪切	将选定内容放入剪贴板，同时删除文档中所选的内容
	复制	将选定内容放入剪贴板，但不删除文档中所选的内容
	粘贴	将剪贴板中的内容粘贴到当前光标处
	删除	删除所选内容
	从数据库删除表	将表从数据库中删除
	全选	选择当前文档中全部内容
	查找和替换	在当前窗口文件中查找指定内容，可将查找到的内容替换为指定信息
	转到	选择定位到“结果”窗格的哪一行
	书签	显示书签功能菜单
	代码	显示代码编辑窗口
	设计器	打开设计器窗口
视图	服务器资源管理器	显示服务器资源管理器窗口
	解决方案资源管理器	显示解决方案资源管理器窗口
	类视图	显示类视图窗口
	代码定义窗口	显示代码定义窗口
	对象浏览器	显示对象浏览器窗口
	错误列表	显示错误列表窗口
	输出	显示输出窗口
	属性窗口	显示属性窗口
	任务列表	显示任务列表窗口
	工具箱	显示工具箱窗口
	查找结果	显示查找结果
	其他窗口	显示其他窗口（如命令窗口、起始页等）
	工具栏	打开工具栏菜单（如标准工具栏、调试工具栏）
	显示窗格	用于“查询”和“视图设计器”中的显示窗格
	工具箱	显示工具箱
	全屏显示	将当前窗体全屏显示
	向后定位	将控制权移交给下一任务
	向前定位	将控制权移交给上一任务
	属性页	为用户控件显示属性页

#### 1.4.4 工具栏介绍

为了操作更方便、快捷，菜单项中常用的命令按功能分组分别放在了相应的工具栏中，通过工具栏可以迅速地访问常用的菜单命令。常用的工具栏有Visual Studio 2008标准工具栏和调试工具栏，下面分别介绍。

(1) Visual Studio 2008标准工具栏包括大多数常用的命令按钮，如“新建项目”、“添加新项”、“打

“开文件”、“保存”和“全部保存”等，如图 1.23 所示。

(2) Visual Studio 2008 调试工具栏包括了对应用程序进行调试的快捷按钮，如图 1.24 所示。

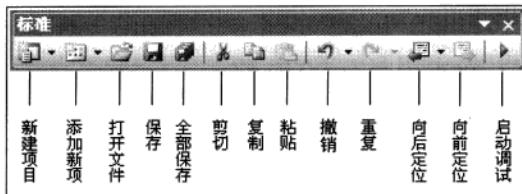


图 1.23 Visual Studio 2008 标准工具栏

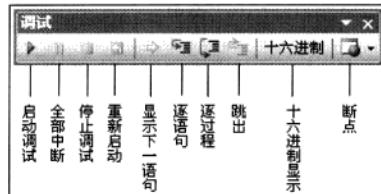


图 1.24 Visual Studio 2008 调试工具栏

技巧：开发人员可以直接按 F5 快捷键调试程序。

#### 1.4.5 “工具箱”面板介绍

工具箱是 Visual Studio 2008 的重要工具，每一个开发人员都必须对这个工具非常熟悉。工具箱中提供了进行 Windows 窗体应用程序开发所必需的控件，通过工具箱，开发人员可以方便地进行可视化的窗体设计，简化程序设计的工作量，从而提高工作效率。根据控件功能的不同，Visual Studio 2008 将工具箱划分为 11 个栏目，如图 1.25 所示。

单击某个栏目后将会显示栏目下的所有控件，如图 1.26 所示。当需要某个控件时，可以通过双击所需要的控件直接将其添加到窗体上，也可以先单击选择需要的控件，再将其拖曳到设计窗体上。工具箱面板中的控件可以通过工具箱右键菜单（如图 1.27 所示）来控制，如实现控件的排序、删除、显示方式等。



图 1.25 “工具箱”面板



图 1.26 展开后的工具箱窗口

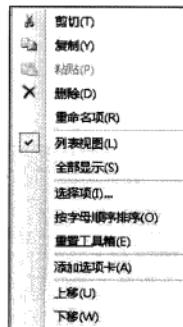


图 1.27 工具箱右键菜单

#### 1.4.6 “属性”面板介绍

“属性”面板是 Visual Studio 2008 中的一个重要窗口，该窗口为 Windows 窗体应用程序的开发提供了全面的属性修改方式。窗体应用程序开发中的各个控件属性都可以由“属性”面板来设置完成。“属性”面板不仅提供了属性的设置及修改功能，还提供了事件的管理功能。另外，“属性”面板可以管理控件的

事件，方便编程时对事件的处理。

“属性”面板同时采用了两种方式管理属性，分别为按分类方式和按字母顺序方式。面板的下方还有简单的帮助说明，方便开发人员对控件的属性进行操作和修改，“属性”面板的左侧是属性名称，相对应的右侧是属性值。“属性”面板如图1.28所示。

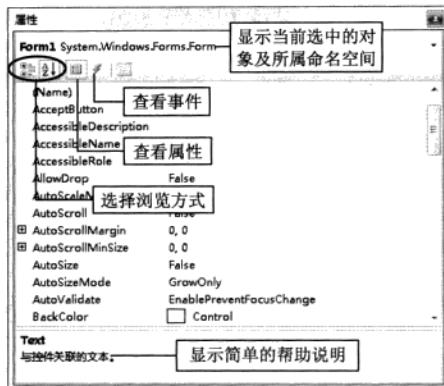


图1.28 “属性”面板

#### 1.4.7 解决方案资源管理器介绍

解决方案资源管理器提供项目及文件的视图，并且提供对项目和文件相关命令的便捷访问。与此窗口关联的工具栏提供适用于列表中突出显示项的常用命令。若要访问解决方案资源管理器，可在“视图”菜单上选择“解决方案资源管理器”选项。解决方案资源管理器如图1.29所示。

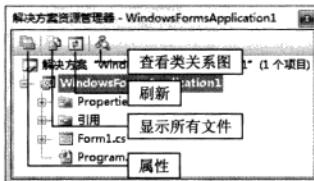


图1.29 解决方案资源管理器

### 1.5 C#编程常用帮助

开发程序时，使用现有的帮助对程序的开发效率是非常有好处的，本节将对C#的一些常用帮助进行详细讲解。

#### 1.5.1 安装MSDN

Visual Studio 2008中提供了一个强大的帮助工具，简称MSDN，它是开发人员在开发项目过程中最好的帮手，其包含了对C#语言各方面知识的讲解，并附有示例代码。本节将对MSDN的安装及使用进行详细讲解。

安装 MSDN 的步骤如下。

(1) 把 Visual Studio 2008 MSDN 安装盘放入光驱中, 光盘自动运行后会进入安装程序文件界面, 如果光盘不能自动运行, 可双击 setup.exe 可执行文件, 应用程序将自动跳转到如图 1.30 所示的 Visual Studio 2008 MSDN 安装向导界面。

(2) 单击“下一步”按钮, 进入到如图 1.31 所示的 Visual Studio 2008 MSDN 安装起始页界面。

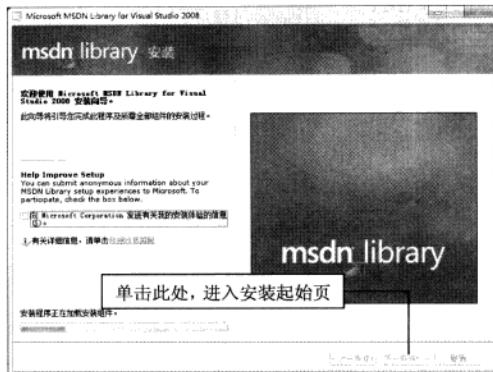


图 1.30 Visual Studio 2008 MSDN 安装向导

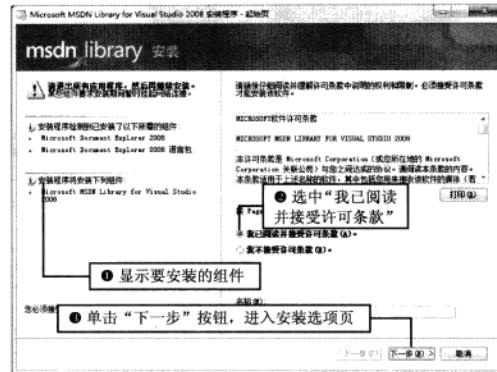


图 1.31 Visual Studio 2008 MSDN 安装起始页

(3) 选中“我已阅读并接受许可条款”单选按钮, 单击“下一步”按钮, 进入到如图 1.32 所示的 Visual Studio 2008 MSDN 安装选项页界面, 这里可以根据需要选择 Visual Studio 2008 MSDN 的 3 种安装类型, 即完全、最小和自定义, 默认为完全安装; 单击“浏览”按钮选择 Visual Studio 2008 MSDN 的安装路径, 默认安装路径为“C:\Program Files\MSDN\MSDN9.0\”。

说明: MSDN 默认为完全安装, 用户可以根据需要进行自定义安装, 如只安装 C#帮助等, 但这里推荐完全安装, 因为在实际开发程序时有可能会用到系统的一些 API 函数或其他语言的类库, 这时使用 MSDN 也可以查找到其用法。

(4) 单击“安装”按钮, 进入到如图 1.33 所示的 Visual Studio 2008 MSDN 安装页界面。

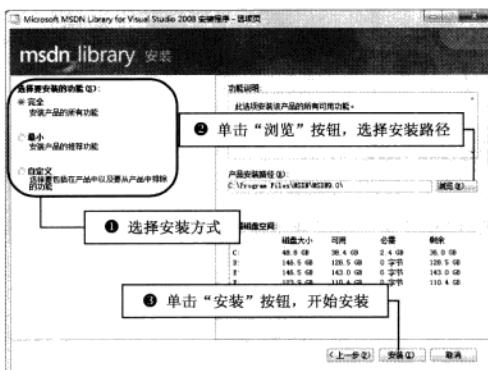


图 1.32 Visual Studio 2008 MSDN 安装选项页

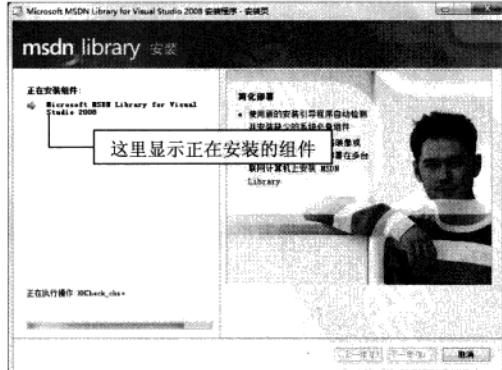


图 1.33 Visual Studio 2008 MSDN 安装页

(5) 程序安装完成之后单击“完成”按钮, Visual Studio 2008 MSDN 安装完成, 如图 1.34 所示。

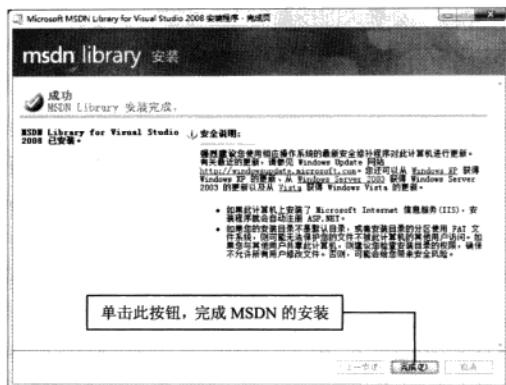


图 1.34 Visual Studio 2008 MSDN 安装完成页

## 1.5.2 使用 MSDN

MSDN 是微软的帮助文档库, 它提供了大量的技术文档, 是开发人员的“左膀右臂”, 下面介绍如何使用 MSDN 帮助, 具体使用步骤如下。

(1) 选择“开始”/“程序”/Microsoft Developer Network/“MSDN Library for Visual Studio 2008 简体中文”命令, 即可进入 MSDN 主界面, 如图 1.35 所示。

(2) 单击 MSDN 主界面工具栏中的“目录”按钮, 可以在主界面的左侧显示“目录”面板, 如图 1.36 所示。在“目录”面板中, 可以让使用者快速地对 MSDN 的结构有一个大致的了解, 并起到了导航的作用。对于 MSDN 文档库较熟悉的读者可以从目录入手, 查找自己感兴趣的内容进行阅读。

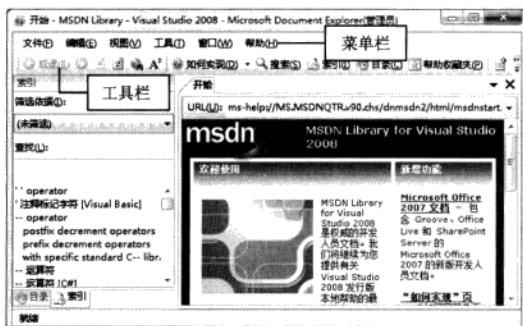


图 1.35 Microsoft Document Explorer (MSDN 主界面)



图 1.36 “目录”面板

(3) MSDN 为不熟悉文档库的读者提供了其他工具, 以方便使用。单击工具栏中的“索引”按钮, 在主界面左侧显示“索引”面板, 该面板为开发人员提供了知识快速检索功能, 如图 1.37 所示。在“查找”

文本框中输入需要查询的内容后，按 Enter 键，MSDN 将自动转入最匹配的技术文档提供给开发人员参考。

(4) MSDN 还为使用者提供了一种强大的搜索功能，可以提供对本地帮助、MSDN Online、Codezone 社区等许多文档库的详细搜索。单击工具栏中的“搜索”按钮，并在文本框中输入搜索的内容摘要，按 Enter 键后，搜索的结果以概要的方式呈现在主界面中，开发人员可以根据自己的需要选择不同的文档进行阅读，其使用示意图如图 1.38 所示。

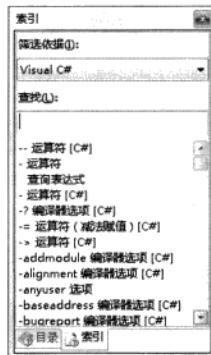


图 1.37 “索引”面板



图 1.38 MSDN 的搜索功能

## 1.6 照猫画虎——基本功训练

### 1.6.1 基本功训练 1——如何开始运行程序

视频讲解：光盘\mr\01\lx\如何开始运行程序.exe

实例位置：光盘\mr\01\zmhh\01

创建完一个应用程序之后，若要查看这个程序的运行结果，就需要运行这个程序，那么该如何运行程序呢？下面通过运行一个控制台应用程序来演示，实现过程如下。

(1) 参照 1.4.1 小节创建一个控制台应用程序，项目名称为 RunProgram，并且让解决方案名称与项目名称相同。

(2) 如图 1.39 所示，在解决方案资源管理器中双击鼠标打开 RunProgram 项目下的 Program.cs 文件。

(3) 打开 Program.cs 文件后，在 Main 方法中输入如下代码。

```
static void Main(string[] args) //应用程序的入口方法
{
    Console.WriteLine("真聪明，你已经学会运行程序了！<@ @>"); //向控制台输出指定文字
    Console.ReadLine(); //等待读取键盘的输入，这里防止程序立刻退出
}
```



图 1.39 打开 Program.cs 文件

**说明:** Program.cs文件是C#程序的类文件,它是在创建控制台应用程序或Windows应用程序时自动生成的。在C#语言中,所有的类文件都以cs为扩展名。另外,Main方法是所有C#应用程序的入口方法,即所有的C#程序都从Main方法开始运行。在Main方法中,可以编写程序启动时需要用到的C#代码,通常该方法位于Program.cs文件中。

(4) 开发人员可以通过单击工具栏中的“启动调试”图标开始运行程序,如图1.40所示。另外,也可以选择菜单栏中的“调试”/“启动调试”或“开始执行(不调试)”命令来运行程序,如图1.41所示。如果选择“启动调试”命令,则在运行程序过程中会自动判断程序中是否有断点或其他标记,以便进行调试;如果选择“开始执行(不调试)”命令,则在运行程序的过程中将完全忽略断点或其他标记。

(5) 程序运行结果如图1.42所示。

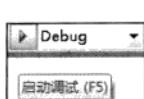


图1.40 “启动调试”图标

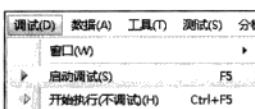


图1.41 选择“启动调试”命令

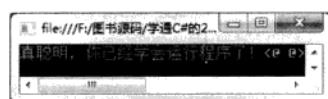


图1.42 运行结果

**技巧:** 从图1.41中可以看到,“启动调试”命令和“开始执行(不调试)”命令后面都有快捷键,所以开发人员也可以通过按F5键或Ctrl+F5键来运行程序。

**熊猫画虎:** 创建一个Windows应用程序,然后运行该应用程序。**提示:** 运行方法与控制台应用程序相同。(20分)(实例位置:光盘\mr\01\zmhh\01\_zmhh)

## 1.6.2 基本功训练2——如何中断当前程序的运行

**视频讲解:** 光盘\mr\01\lx\如何中断当前程序的运行.exe

**实例位置:** 光盘\mr\01\zmhh\02

如果一个应用程序正在运行,那么该如何中断这个程序呢?下面通过运行一个控制台应用程序来演示,实现过程如下。

(1) 参照1.4.1小节创建一个控制台应用程序,项目名称为StopProgram,并且让解决方案名称与项目名称相同。

(2) 在解决方案资源管理器中双击鼠标打开StopProgram项目下的Program.cs文件,并在Main方法中输入如下代码。

```
Console.WriteLine("演示中断当前程序的运行!"); //向控制台输出指定文字
Console.ReadLine(); //等待读取键盘的输入,这里防止程序立刻退出
```

(3) 单击工具栏中的“启动调试”图标运行该程序,运行结果如图1.43所示。

(4) 在程序运行之后可以通过单击工具栏中的“停止调试”按钮中断正在运行的程序,如图1.44所示。

**说明:** 若要停止调试,也可以选择菜单栏中的“调试”/“停止调试”命令来中断程序,如图1.45所示。



图1.43 运行结果

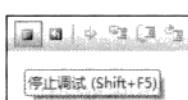


图1.44 “停止调试”图标

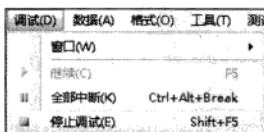


图1.45 选择“停止调试”命令

**熊猫画虎：**创建一个 Windows 应用程序，然后通过单击工具栏中的“停止调试”按钮中断正在运行的程序。(20 分)(实例位置：光盘\mr\01\zmhh\02\_zmhh)

### 1.6.3 基本功训练 3——设置程序代码行号

**视频讲解：**光盘\mr\01\lx\设置程序代码行号.exe

在 Visual Studio 2008 开发环境中可以设置程序代码的行号显示功能，设置完行号之后的程序代码如图 1.46 所示。

通过设置程序代码行号，可以清晰地看到代码置于后台编辑器中的位置以及在程序发生错误时对错误代码的查找，实现过程如下。

(1) 打开 Visual Studio 2008 开发环境，在菜单栏中选择“工具”/“选项”命令，弹出如图 1.47 所示的“选项”对话框。

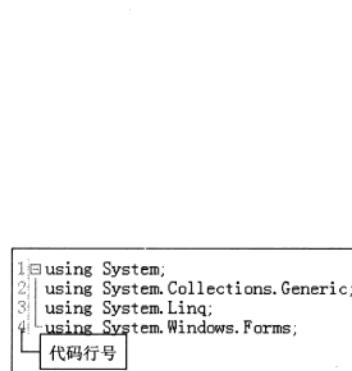


图 1.46 显示代码行号

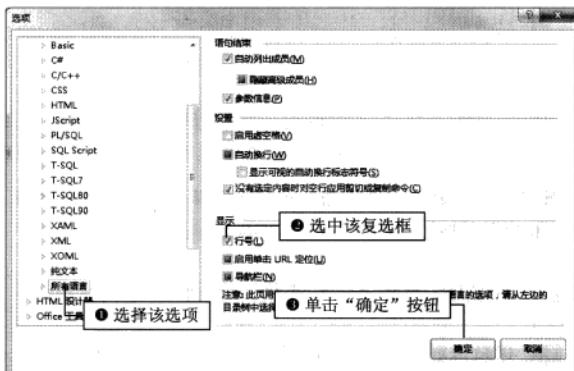


图 1.47 “选项”对话框

(2) 在该对话框中选择“文本编辑器”节点下的“所有语言”选项，在右侧选中“行号”复选框，然后单击“确定”按钮即可。

**技巧：**在 Visual Studio 2008 开发环境中设置完显示代码行号后，如果在调试程序过程中出现错误，可以通过代码行号快速地找到程序出错的位置。

**熊猫画虎：**首先设置显示程序代码行号，然后再取消程序代码行号。(20 分)

### 1.6.4 基本功训练 4——统一窗体中控件的字体设置

**视频讲解：**光盘\mr\01\lx\统一窗体中控件的字体设置.exe

**实例位置：**光盘\mr\01\zmhh\04

当窗体上有很多控件时，如果逐个设置字体属性，会非常繁琐，这时，可以将字体属性设置一致的控件选中进行统一设置，这样可以大大节省开发程序的时间。本实例运行效果如图 1.48 所示。

实现过程如下。

- (1) 打开 Visual Studio 2008 开发环境，新建一个 Windows 窗体应用程序，并将其命名为 SetFont。
- (2) 在默认窗体 Form1 中添加 3 个 Button 控件。

(3) 在 Form1 窗体中选中添加的 3 个 Button 控件（首先单击任意一个 Button 控件，然后按下 Ctrl 键，并用鼠标左键依次选择另外两个控件），然后在任意 Button 控件上单击鼠标右键，在弹出的快捷菜单中选择“属性”命令，在弹出的“属性”对话框中对其字体进行统一设置，这里将字体设置为“宋体”、字体大小设置为 10、字体颜色设置为 Red，如图 1.49 所示。



图 1.48 统一窗体中控件的字体设置



图 1.49 设置统一体字

**熊猫画虎：**创建一个 Windows 应用程序，在默认窗体 Form1 上添加 3 个 Label 控件，然后将这 3 个控件的字体统一设置为“宋体”、字体大小设置为 12、字体颜色设置为 Green。（20 分）（实例位置：光盘\mr\01\zmhh\04\_zmhh）

### 1.6.5 基本功训练 5——通过“格式”菜单布局窗体

视频讲解：光盘\mr\01\lx\通过“格式”菜单布局窗体.exe

实例位置：光盘\mr\01\zmhh\05

在开发程序时，为了使窗体美观大方，需要设置窗体上控件的显示布局。例如，可以设置 3 个 Button 控件之间的水平间距相同，这可以通过“格式”菜单进行设置，本实例运行效果如图 1.50 所示。



图 1.50 通过“格式”菜单布局窗体

实现过程如下。

- (1) 打开 Visual Studio 2008 开发环境，新建一个 Windows 窗体应用程序，并将其命名为 LayoutForm。
- (2) 在默认窗体 Form1 中添加 3 个 Button 控件。
- (3) 在 Form1 窗体中选中添加的 3 个 Button 控件，在菜单栏中依次选择“格式” / “水平间距” / “相同间隔”命令，如图 1.51 所示，使 3 个 Button 控件之间的水平间距相同。

**说明：**开发人员还可以通过“格式”菜单设置窗体中控件的对齐方式、大小及垂直间距等，其对应的子菜单分别如图 1.52、图 1.53 和图 1.54 所示。

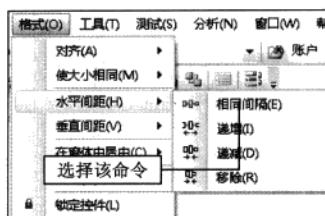


图 1.51 选择“格式”/“水平间距”/“相同间隔”命令

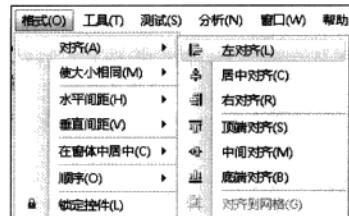


图 1.52 设置对齐方式

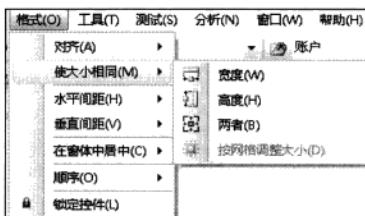


图 1.53 设置控件大小

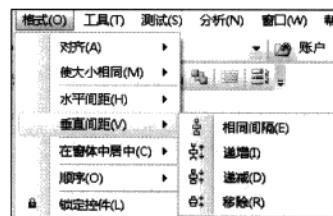


图 1.54 设置垂直间距

**照猫画虎：**创建一个 Windows 应用程序，在默认窗体 Form1 上添加 3 个 TextBox 控件，然后设置这 3 个控件之间的垂直间距相同，并设置左对齐。(20 分)(实例位置：光盘\mr\01\zmhh\05\_zmhh)

照猫画虎栏目分数统计：

照猫画虎题目	1	2	3	4	5	总分数	
分数							

## 1.7 情景应用——拓展与实践

### 1.7.1 情景应用 1——设置 Windows 应用程序启动窗体

视频讲解：光盘\mr\01\lx\设置 Windows 应用程序启动窗体.exe

实例位置：光盘\mr\01\qjyy\01

一个完整的 Windows 应用程序中一般都有多个窗体，如果要调试程序，必须设置首先运行的窗体，这时就需要设置项目的启动窗体。默认情况下，Windows 应用程序的启动窗体是 Form1，本实例实现设置 Form2 窗体为启动窗体，实现过程如下。

- (1) 打开 Visual Studio 2008 开发环境，新建一个 Windows 窗体应用程序，并将其命名为 SetStartForm。
- (2) 在该项目中添加一个 Form2 窗体，以便将其设置为启动窗体。
- (3) 在“解决方案资源管理器”中找到 Program.cs 文件，双击打开该文件，这时即可通过修改 Application.Run 方法中的参数来随意设置启动窗体，如本实例设置 Windows 应用程序的启动窗体为 Form2，具体实现代码如下。

```
Application.Run(new Form2()); //设置 Form2 窗体为启动窗体
```

说明：Program.cs 文件是创建 Windows 应用程序时自动生成的一个类文件，它是程序的主程序文件。

**DIY：**同时启动两个窗体。提示：以上面的例子为基础，使用鼠标右键单击Form2窗体，在弹出的快捷菜单中选择“查看代码”命令，打开该窗体的后台代码，在窗体Form2方法中添加如下代码。（20分）  
**(实例位置：光盘\mr\01\qjyy\01\_diy)**

```
public Form2()
{
    InitializeComponent();
    new Form1().Show();
}
```

## 1.7.2 情景应用2——为程序设置版本和帮助信息

**视频讲解：**光盘\mr\01\lx\为程序设置版本和帮助信息.exe

**实例位置：**光盘\mr\01\qjyy\02

在计算机中使用软件时，经常会在软件中看到其版本、所属公司等信息。例如，C#程序开发人员经常使用的Visual Studio 2008开发环境，当用户将鼠标指针移动到Visual Studio 2008开发环境的setup.exe安装文件上时，会弹出一个信息提示框，该提示框中可以看到软件的说明、公司、版本、创建日期以及大小等信息，如图1.55所示。

本实例将详细讲解如何为程序设置版本及其帮助等信息，实现过程如下。

- (1) 打开Visual Studio 2008开发环境，新建一个Windows窗体应用程序，设置项目名称为SetSoftInfo。
- (2) 在该项目的Properties文件夹下找到AssemblyInfo.cs文件，如图1.56所示。



图1.55 在信息提示框中显示软件版本等信息

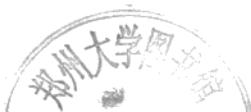


图1.56 查找AssemblyInfo.cs文件

(3) 双击打开AssemblyInfo.cs文件，这时即可为程序设置版本和帮助等信息，具体实现代码如下。

```
//有关程序集的常规信息通过下列属性集控制，更改这些属性值可修改与程序集关联的信息
[assembly: AssemblyTitle("为程序设置版本和帮助信息")]
[assembly: AssemblyDescription("")]
[assembly: AssemblyConfiguration("")]
[assembly: AssemblyCompany("明日科技")]
[assembly: AssemblyProduct("为程序设置版本和帮助信息")]
[assembly: AssemblyCopyright("版权所有 (C) 2010")]
[assembly: AssemblyTrademark("")]
[assembly: AssemblyCulture("")]

//程序集的版本信息由下面4个值组成，即主版本、次版本、内部版本号和修订号
[assembly: AssemblyVersion("1.0.0.0")]
[assembly: AssemblyFileVersion("1.0.0.0")]
```



**技巧：**在 AssemblyInfo 文件中有一个 AssemblyCompany 属性，通过该属性可以设置开发该软件的公司（或作者）名称，以便更好地维护软件著作权。

**DIY：**设置程序的版本为 9.9.9.9，所属公司名称为 Hello。提示：设置 AssemblyFileVersion 属性和 AssemblyCompany 属性。（20 分）（实例位置：光盘\mr\01\qjyy\02\_diy）

### 1.7.3 情景应用 3——为项目添加已有窗体

**视频讲解：**光盘\mr\01\lx\为项目添加已有窗体.exe

**实例位置：**光盘\mr\01\qjyy\03

已有窗体就是用户已经编写好的窗体或别人编写好的窗体，通过添加已有窗体可以节省编写相同功能的窗体所需的时间，提高程序的开发效率。本实例在创建完的项目中添加一个进销存管理系统的登录窗体，添加的进销存管理系统登录窗体效果如图 1.57 所示。

实现过程如下。

- (1) 打开 Visual Studio 2008 开发环境，新建一个 Windows 窗体应用程序，并将其命名为 AddForm。
- (2) 在“解决方案资源管理器”中选中项目名称，单击鼠标右键，在弹出的快捷菜单中选择“添加”/“现有项”命令，弹出“添加现有项”对话框，如图 1.58 所示。



图 1.57 添加的进销存管理系统登录窗体



图 1.58 “添加现有项”对话框

- (3) 在图 1.58 所示的对话框中找到并选中要添加的已有窗体文件（包括窗体的代码文件和设计文件），这里找到并选中进销存管理系统的登录窗体文件，单击“添加”按钮，即可将已有窗体添加到当前项目中。

**DIY：**创建两个 Windows 应用程序，并把默认窗体 Form1 分别重命名为 FormA 和 FormB，然后实现把 FormB 窗体添加到 FormA 所在的项目中。提示：使用 Visual Studio 2008 的添加现有项功能。（20 分）（实例位置：光盘\mr\01\qjyy\03\_diy）

### 1.7.4 情景应用 4——动起来的 Label 控件

**视频讲解：**光盘\mr\01\lx\动起来的 Label 控件.exe

**实例位置：**光盘\mr\01\qjyy\04

普通窗体中的文字位置都是固定的，但在一些窗体中需要让文字动起来，如在一些广告性较强的界面

中需要做一些滚动的字幕。本实例实现了一个具有滚动字幕效果的窗体，如图 1.59 所示。运行本实例，单击“演示”按钮，可以看到窗口中的文字开始滚动；单击“暂停”按钮，可以使字幕停止滚动。

实现过程如下。

(1) 打开 Visual Studio 2008 开发环境，新建一个 Windows 窗体应用程序，并将其命名为 MoveFontInForm。

(2) 更改默认窗体 Form1 的 Name 属性为 Frm\_Main，在该窗体中添加一个 Label 控件，用来显示要滚动的文字信息；添加 3 个 Button 控件，分别用来执行开始滚动、停止滚动和关闭窗体操作；添加一个 Timer 组件，用来控制字幕的滚动。

(3) 程序主要代码如下。

```
private void timer1_Tick(object sender, EventArgs e)      //用 Timer 来控制字幕的滚动
{
    label1.Left -= 2;                                //设置 label1 左边缘与其容器的工作区左边缘之间的距离
    if (label1.Right < 0)                            //当 label1 右边缘与其容器的工作区左边缘之间的距离小于 0 时
    {
        label1.Left = this.Width;                    //设置 label1 左边缘与其容器的工作区左边缘之间的距离为该窗体的宽度
    }
}
private void button1_Click(object sender, EventArgs e)   //单击“演示”按钮
{
    timer1.Enabled = true;                          //开始滚动
}
private void button2_Click(object sender, EventArgs e)   //单击“暂停”按钮
{
    timer1.Enabled = false;                         //停止滚动
}
```

**提示：**本实例用到了 Timer 控件，在 Timer 控件的属性窗口中最好将其 Enabled 属性初始设置为 false，否则程序运行后 TextBox 控件将会自动发生左右滚动。

**DIY：**动起来的 TextBox 控件。**提示：**可参照上面的实例，把 Label 控件换成 TextBox 控件。(20 分)  
(实例位置：光盘\mr\01\qjyy\04\_diy)

## 1.7.5 情景应用 5——加法计算器

**视频讲解：**光盘\mr\01\lx\加法计算器.exe

**实例位置：**光盘\mr\01\qjyy\05

计算器在日常生活和工作中经常被用到，本实例开发一个简单的计算器，该计算器只能进行加法运算，如图 1.60 所示。运行本实例，在窗体中输入加数和被加数，然后单击“计算”按钮，结果会出现在最下面的 TextBox 控件内。

实现过程如下。

(1) 打开 Visual Studio 2008 开发环境，新建一个 Windows 窗体应用程序，并将其命名为 AddCalculator。

(2) 在默认窗体 Form1 上添加 3 个 TextBox 控件，分别用来输入加数、被加数和显示计算结果，并将表示运算结果的 TextBox 控件的 ReadOnly 属性（称为只读属性）设置为 true，以防止手动修改其值；另外再添加两个 Button 控件，用来实现加法计算操作和清空 TextBox 控件的操作。

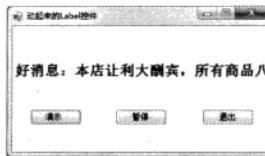


图 1.59 窗体中的滚动字幕



图 1.60 加法计算器

(3) 程序主要代码如下。

```

private void button1_Click(object sender, EventArgs e)      //单击“计算”按钮
{
    int intNum1 = int.Parse(textBox1.Text.Trim());
    int intNum2 = int.Parse(textBox2.Text.Trim());
    textBox3.Text = Convert.ToString(intNum1 + intNum2);  //把表示数字的字符串转换为整数
                                                        //把表示数字的字符串转换为整数
                                                        //显示两个数相加后的结果
}
private void button2_Click(object sender, EventArgs e)      //单击“清空”按钮
{
    textBox1.Text = "";                                //清空 textBox1
    textBox2.Text = "";                                //清空 textBox2
    textBox3.Text = "";                                //清空 textBox3
}

```

**DIY：制作减法计算器。提示：把代码中的加法运算符（+）更改为减法运算符（-）。(20 分)(实例位置：光盘\mr\01\qjyy\05\_diy)**

情景应用 DIY 栏目分数统计：

DIY 题目	1	2	3	4	5	总分数	
分数							

## 1.8 自我测试

### 一、选择题（每题 10 分，5 道题）

- 关于 C#与.NET 框架的关系，下面阐述正确的是（ ）。
 

A. C#是.NET 框架的一部分    B. C#支持的一些特性，.NET 都支持    C. C#不是.NET 的一部分
- 若要运行应用程序，则需要单击工具栏上的哪个按钮（ ）。
 

A.    B.    C.    D.
- 如何查看窗体或控件的事件（ ）。
 

A. 在工具箱中查看    B. 在属性窗口中查看    C. 在解决方案资源管理器中查看
- 输入文字使用哪个控件（ ）。
 

A.    B.    C.
- 在窗体上添加 3 个 TextBox 控件和一个 Button 控件，然后在代码窗口中编写如下事件过程：

```
private void button1_Click(object sender, EventArgs e)      //单击 Button 控件
```

```
{
    textBox1.Text = "我爱 C#";
    textBox2.Text = textBox1.Text;
    textBox3.Text = "我爱 VB";
}
```

程序运行后，单击命令按钮，textBox2 文本框中显示的内容为（ ）。

- A. 我爱 VB    B. 我爱 C#    C. 我爱 VB 我爱 C#

### 二、填空题（每题 10 分，5 道题）

- 要在控制台输出 king's weather 字样，需将下面代码填写完整。

```
static void Main(string[] args)
{
    (
        )
}
```

2. 清空 TextBox 控件中的内容应使用（ ）属性。
3. 运行 C# 程序的快捷键包括（ ）和（ ）。
4. 若要把 TextBox 控件设置为只读状态，应把它的 ReadOnly 属性设置为（ ）。
5. 程序运行后，单击 Button 控件，textBox2 控件中显示的内容为（ ）。

```
private void button1_Click(object sender, EventArgs e) //单击 Button 控件
{
    textBox1.Text = "我爱 C#";
    textBox2.Text = textBox1.Text;
    textBox1.Text = "我爱 VB";
}
```

测试分数统计：

类别	第1题	第2题	第3题	第4题	第5题
选择题分数					
填空题分数					
总分数					

## 1.9 行动指南

开始日期： 年 月 日

结束日期： 年 月 日

序号	内 容	行 动 指 南	
1	照猫画虎栏目 分数（ ）	分数>75 分	优秀，基本功掌握得不错，加油！
		75 分>分数>50 分	及格，知识掌握得不牢，重新做一遍照猫画虎。
		分数<50 分	请使用光盘中提供的“实战能力测试系统”进行提高训练。
	情景应用栏目 分数（ ）	分数>75 分	优秀，综合应用能力很强。
		75 分>分数>50 分	及格，综合应用能力需提高，再练一遍情景应用。
	自我测试栏目 分数（ ）	分数<50 分	请使用光盘中提供的“实战能力测试系统”进行提高训练。
		分数>75 分	优秀，有成为编程高手的潜质。
2	综合评价	请使用光盘中提供的“实战能力测试系统”进行提高训练。 反复训练后，以上各项分数都在 75 分以上，方可进入下一堂课学习。	
	编程能力培养：本栏目提供了一些实践题目，对于培养编程能力很有效，要做好。 、	1. 制作一个最简单的计算器，可以计算加、减、乘、除。	
		2. 制作一个简单的窗体换肤程序（单击窗体，窗体背景变颜色）。提示：设置窗体的 BackColor 属性，可利用 MSDN 或编程词典查一下该属性的用法。	
		3. 制作一个简单的文字放大器。单击一次窗体，窗体中的文字被放大一次，提示：使用 Label 控件显示文字，改变文字大小可通过设置 Label 控件的 Font 属性来实现。	
		4. 制作一个简单的窗体名称变换器。要求程序运行时，在 TextBox 控件中输入文字，单击窗体，窗体的标题名称变为 TextBox 控件中输入的文字。提示：单击窗体会触发窗体的 Click 事件，窗体的标题属性为 Text。	

续表

序号	内 容	行 动 指 南
3	编程习惯培养：本堂课培养读者在学习开发中记笔记的习惯，把开发中遇到的问题、总结的经验记录下来。	
4	创新能力培养：看看身边有没有可以用编程解决的问题，记录到右边的表格中。根据学习进度，尝试编写解决这些问题的小程序。	

## 1.10 成功可以复制——C#语言之父安德斯·海尔斯伯格

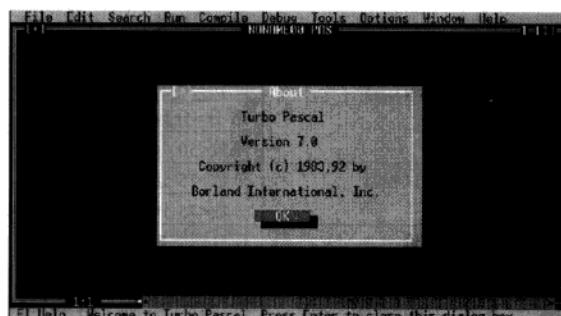
安德斯·海尔斯伯格（Anders Hejlsberg）曾在丹麦技术大学学习工程，但没有毕业。他自学计算机编程，并在丹麦创建了 Poly Data 公司，凭借他的聪明才智和富有挑战的性格，独立开发了 Pascal 语言的编译器核心。

20世纪80年代，安德斯去了美国，他凭借掌握的 Pascal 语言的编译器核心技术和著名数学家 Philippe Kahn 共同创建了世界著名的第三大软件公司——Borland。

安德斯首先在 Borland 公司主持开发了享誉盛名的 Turbo Pascal 语言，Turbo Pascal 语言的设计成功使得他的名声享誉全球 IT 界。但安德斯并未因此而沾沾自喜、止步不前，而是接着挑战自己目前的成功。接下来，安德斯带领他的研发小组继续升级 Turbo Pascal 语言，最终将 Turbo Pascal 变成一种面向对象的、拥有可视化环境和卓越的数据库访问特性的应用程序开发语言——Delphi，发布后的 Delphi 语言立刻在全球 IT 界获得了开发工具史上奇迹之称的美誉。

20世纪90年代，Java 语言出现了，安德斯在 Borland 一直感觉郁郁不得志，这时慧眼识才的比尔·盖茨三顾茅庐，邀请安德斯加入微软。最开始，微软的重金开价并未打动安德斯，因为凭借他在 Borland 的多年工作，并作为公司的创始人之一，他应该拥有大量的财富，早已是亿万富翁级的身份。实际上，在安德斯的内心深处更需要一个宽松的工作环境和编程领域内更大的挑战，但他的这些想法，当时的 Borland 公司是无法给予他的。当比尔·盖茨知道他的想法后，马上给他一个非常大的挑战，许诺给他一个宽松的工作环境，让其领导 Visual J++ 语言设计小组，这样他欣然加入了。在安德斯的带领下，Visual J++ 语言的开发非常成功，很快 Visual J++ 将 Java 开发人员拉拢到它的周围了。于是在 1999 年安德斯被授予“distinguished engineer”（卓越工程师），在微软仅有 16 人获得这样的荣誉。

后来 SUN 公司以微软破坏了 Java 的跨平台性为由，终止了对微软的 Java 授权，这样促使微软决心自



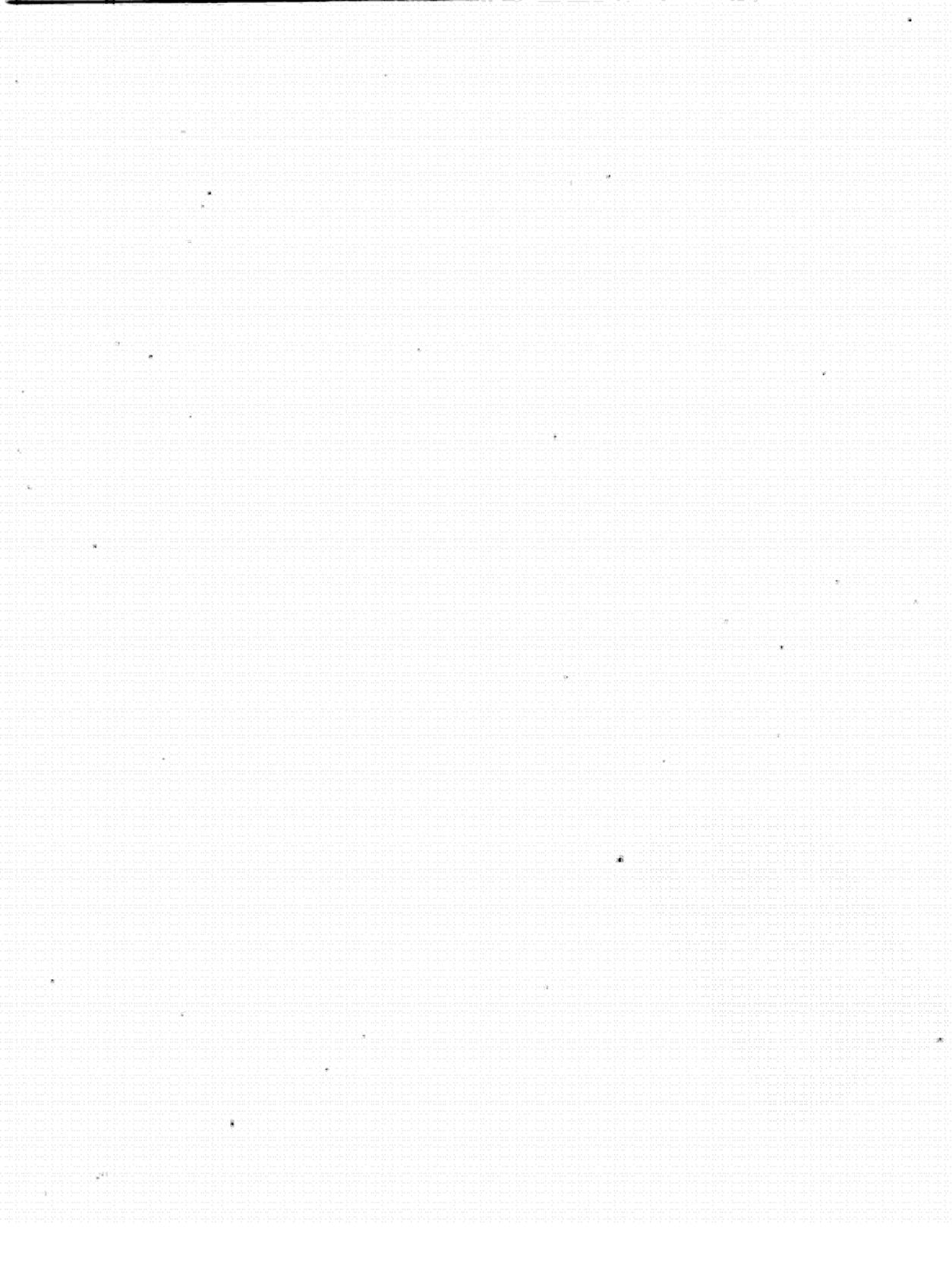
已研发能够面向网络的、完全面向对象的编程语言——C#。微软把这个在编程领域具有革命意义的重任交给了安德斯，安德斯又一次非常高兴地接受了挑战。而后，安德斯带领他的C#研发团队，经过几年时间的埋头苦干，终于使C#成为微软反击Java语言的最有力武器。

经典语录

每两年，CPU的速度要加倍，内存容量是原来的4倍，但是我们的大脑还是原来的大小，很显然，将需要更多的程序员来补充这些。

深度评价

从安德斯编写Pascal语言的编译器核心开始，到他后来设计Turbo Pascal语言、Delphi语言、Visual J++语言、C#语言可以看到，他的每一步都获得了巨大成功，但他从未止步不前，而是把每一次的成功作为迎接更大挑战的基石，安德斯的这种富于挑战、不断超越自我的精神非常值得我们大家学习。学习编程也一样，只有不断地向上攀登，才会有更大的收获和成功。



# 第 2 堂课

## C#程序设计基础

( 视频讲解：168分钟)

学习任何一门语言都不能一蹴而就，必须遵循一个客观的原则，即从基础学起，有了牢固的基础，再进阶学习有一定难度的技术就会变得轻松。本堂课将从初学者的角度考虑，对C#程序设计的一些基础知识，如Main方法的使用、关键字、标识符、代码注释、数据类型、变量、常量、数据类型转换及各种运算符的使用等进行详细讲解。

学习摘要：

- » 掌握命名空间的定义与引用
- » 熟悉标识符的定义
- » 了解关键字的使用
- » 掌握代码注释的几种方式
- » 熟悉C#数据类型
- » 掌握变量的声明及使用
- » 掌握常量的声明及使用
- » 掌握常用的数据类型转换方式
- » 掌握算术运算符、赋值运算符和关系运算符的使用
- » 熟悉逻辑运算符和移位运算符的使用
- » 了解C#中其他特殊运算符（如is、new、typeof等）的使用
- » 了解运算符的优先级顺序

## 2.1 编写第一个 C# 程序

本节使用 Visual Studio 2008 开发环境和 C# 语言编写一个简单的“Hello World!”程序，该程序在控制台上显示字符串“Hello World！”，具体步骤如下。

(1) 选择“开始”/“程序”/Microsoft Visual Studio 2008/Microsoft Visual Studio 2008 命令，打开 Visual Studio 2008 开发环境。

(2) 选择 Visual Studio 2008 开发环境工具栏中的“文件”/“新建”/“项目”命令，打开“新建项目”对话框，如图 2.1 所示。

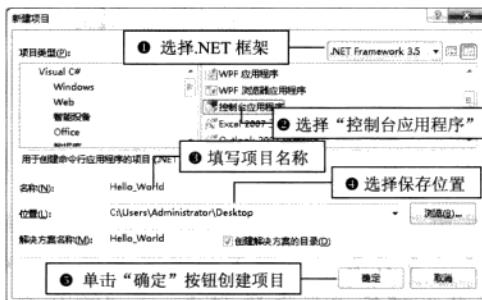


图 2.1 “新建项目”对话框

(3) 选择“控制台应用程序”，将其命名为 Hello\_World，保存在桌面上，然后单击“确定”按钮创建一个控制台应用程序。

(4) 在 Main 方法中输入代码。

**例 2.01** 创建一个控制台应用程序，使用 WriteLine 方法输出“Hello World！”字符串，代码如下。（实例位置：光盘\mr\02\sl\2.01）

```
static void Main(string[] args)           //Main 方法，此方法下编写代码输出数据
{
    Console.WriteLine("Hello World!");      //输出“Hello World！”
    Console.ReadLine();
}
```

程序运行结果如图 2.2 所示。

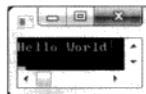


图 2.2 输出字符串“Hello World！”

**技巧：**许多人在开发应用程序时会遇到这样一种情况：程序中需要使用某一个类或某一个方法，但忘记了类或方法的完整名称，只记得前几个字母。这里笔者介绍一种简单的方法来帮助大家迅速地查找到类或方法名。在代码编辑器窗口中输入类或方法的开始字母，Visual Studio 2008 会显示一个“完全补词”窗口，如图 2.3 所示，该窗口将显示以输入字母开头的所有类或方法名，这样极大地方便了用户查找类和方法，提高了编程效率。

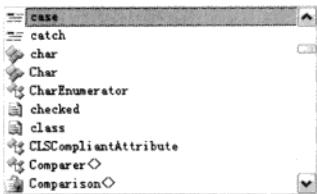


图 2.3 “完全补词”窗口

## 2.2 分析 C#程序结构

一个 C# 程序的结构大体可以分为命名空间、类、Main 方法、标识符、关键字、语句等，图 2.4 描述了一个基本的 C# 程序结构。

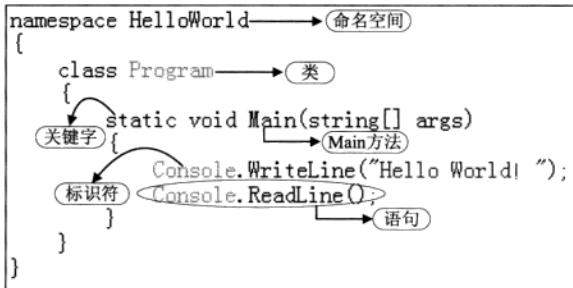


图 2.4 C#程序结构

### 2.2.1 命名空间介绍

C# 程序是利用命名空间组织起来的，命名空间既用作程序的“内部”组织系统，也用作向“外部”公开的组织系统（即一种向其他程序公开自己拥有的程序元素的方法）。如果要调用某个命名空间中的类或方法，首先需要使用 `using` 指令引入命名空间，`using` 指令将命名空间名所标识的命名空间内的类型成员导入当前编译单元中，从而可以直接使用每个被导入的类型的标识符，而不必加上它们的完全限定名。

`using` 指令的基本形式如下。

`using 命名空间名;`

**例 2.02** 创建一个控制台应用程序，在其主程序文件 `Program.cs` 中建立一个命名空间 `N1`，在命名空间 `N1` 中自定义一个类 `A`，在项目中使用 `using` 指令引入命名空间 `N1`，然后在命名空间 `UseNameSpace` 中即可创建命名空间 `N1` 中的类，然后调用此类中的 `MyIs` 方法，代码如下。（实例位置：光盘\mr\02\sl\2.02）

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using N1;
namespace UseNameSpace
{
    //使用 using 指令引入命名空间 N1
}

```

```

class Program
{
    static void Main(string[] args)
    {
        A oa = new A(); //创建 N1 中的类 A
        oa.MyIs(); //调用类 A 中的 MyIs 方法
    }
}
namespace N1
{
    class A //自定义类 A
    {
        public void MyIs()
        {
            Console.WriteLine("学通 C# 的 24 堂课"); //输出字符串
            Console.ReadLine();
        }
    }
}

```

程序的运行结果为“学通 C# 的 24 堂课”。

## 2.2.2 类的介绍

类是一种数据结构，它可以封装数据成员、方法成员和其他的类，同时它也是创建对象的模板，C#中所有的语句都必须包含在类内，因此，类是 C# 语言的核心和基本构成模块。C# 支持自定义类，使用 C# 编程，实质上就是编写自己的类来描述实际需要解决的问题。

使用任何新的类之前都必须声明它，一个类一旦被声明，就可以当作一种新的类型来使用，在 C# 中通过使用 `class` 关键字来声明类，声明形式如下。

[类修饰符] `class` [类名] [基类或接口]  
{  
 [类体]  
}

在 C# 中，类名是一种标识符，必须符合标识符的命名规则。类名要能够体现类的含义和用途，而且一般采用第一个字母大写的名词，另外，也可以采用多个词构成的组合词。

**例 2.03** 声明一个最简单的类，此类没有任何意义，只演示如何声明一个类，代码如下。

```

class MyClass
{
}

```

## 2.2.3 Main 方法的使用

Main 方法是程序的入口点，程序将在此处创建对象和调用其他方法，一个 C# 程序中只能有一个入口点，每新建一个项目，程序都会自动生成一个 Main 方法，默认的 Main 方法代码如下。

```

static void Main(string[] args)
{
}

```

### 说明：Main方法默认访问级别为private。

可以用3个修饰符修饰Main方法，分别是public、static和void。

- public：说明Main方法是公有的，在类的外部也可以调用整个方法。
- static：说明Main方法是一个静态方法，即这个方法属于类的本身而不是这个类的特定对象。调用静态方法不能使用类的实例对象，而必须直接使用类名来调用。
- void：此修饰符说明Main方法无返回值。

Main方法是一个特别重要的方法，使用时需要注意以下几点。

- Main方法是程序的入口点，程序控制在该方法中开始和结束。
- Main方法在类或结构的内部声明。它必须为静态方法，而且不应该为公共方法。
- Main方法可以具有void或int返回类型。
- 在声明Main方法时既可以使用参数，也可以不使用参数。
- 参数可以作为从零开始索引的命令行参数来读取。

### 2.2.4 认识标识符

标识符是指在程序中用来表示事物的单词，其命名有3个基本规则。

- 标识符只能由数字、字母和下划线组成。
- 标识符必须以字母或下划线开头。
- 标识符不能是关键字。

例如，在“Hello World！”程序中的Console就是标识符，如图2.5所示。

```
static void Main(string[] args)
{
    Console.WriteLine("Hello World!");
    Console.ReadLine();
}
```

图2.5 标识符

### 2.2.5 认识关键字

关键字是指在C#语言中具有特殊意义的单词，它们被C#设定为保留字，不能随意使用。例如，在“Hello World！”程序中的static、void和string都是关键字，如图2.6所示。

```
static void Main(string[] args)
{
    Console.WriteLine("Hello World!");
    Console.ReadLine();
}
```

图2.6 关键字

### 2.2.6 编写C#语句

语句是构成所有C#程序的基本单位，在语句中可以声明局部变量或常数、调用方法、创建对象或将值

赋给变量、属性或字段，语句通常以分号终止。

**例 2.04** 下面的代码就是一条语句。

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

上面的代码用来调用 Console 类中的 WriteLine 方法输出“Hello World！”字符串。

## 2.2.7 代码注释

代码注释的主要功能是对某行或某段代码进行说明，方便对代码的理解与维护，程序运行时不执行注释的代码。注释可以分为行注释和块注释两种，行注释均以“//”开头。

**例 2.05** 在“Hello World！”程序中使用行注释，代码如下。

```
static void Main(string[] args) //程序的 Main 方法
{
    Console.WriteLine("Hello World!"); //输出 "Hello World! "
    Console.ReadLine();
}
```

如果注释的行数较少，则一般使用行注释。对于连续多行的大段注释，则使用块注释，块注释通常以“/\*”开始，以“\*/”结束，注释的内容放在它们之间。

**例 2.06** 在“Hello World！”程序中使用块注释，代码如下。

```
class Program
{
    /*程序的 Main 方法中可以输出 "Hello World!" 字符串 //块注释开始
    static void Main(string[] args) //Main 方法
    {
        Console.WriteLine("Hello World!"); //输出 "Hello World! " 字符串
        Console.ReadLine();
    }
}
```

 注意：注释可以出现在代码的任意位置，但是不能分隔关键字和标识符。

 技巧：① 在 Visual Studio 2008 中对代码进行注释时，可以通过单击工具栏中的图标实现，取消注释时，可以通过单击工具栏中的图标实现。

② 在 Visual Studio 2008 开发环境中，如果对一段代码整体进行注释，可以在其上方输入“//”，这时会在相应的位置自动编写如下注释语句，开发人员只需在其中填写注释的文字即可，如下面代码。

```
/// <summary>
///
/// </summary>
/// <param name="args"></param>
```

③ 在 Visual Studio 2008 开发环境中，可以使用#region 和#endregion 关键字指定可展开或折叠的代码块，这样使得代码可以更好 地进行布局。例如，下面代码使用#region 和#endregion 关键字注释 Main 方法。

```
#region
static void Main(string[] args)
```

```
{
}
#endregion
```

## 2.3 数据类型

C#中的数据类型根据其定义可以分为两种：一种是值类型，另一种是引用类型。这两种类型的差异在于数据的存储方式不同，值类型直接存储数据；而引用类型则存储实际数据的引用，程序通过此引用找到真正的数据。本节将对C#中的数据类型进行详细讲解。

### 2.3.1 值类型的使用

值类型直接存储数据值，它主要包含整数类型、浮点类型及布尔类型等。由于值类型在堆栈中进行分配，因此效率很高，使用值类型主要目的是为了提高性能。值类型具有如下特性。

- 值类型变量都存储在堆栈中。
- 访问值类型变量时，一般都是直接访问其实例。
- 每个值类型变量都有自己的数据副本，因此对一个值类型变量的操作不会影响其他变量。
- 复制值类型变量时，复制的是变量的值，而不是变量的地址。
- 值类型变量不能为null，而必须具有一个确定的值。

值类型是从System.ValueType类继承而来的类型，下面详细介绍值类型中包含的几种数据类型。

#### 1. 整数类型

整数类型代表一种没有小数点的整数值，在C#中内置的整数类型如表2.1所示。

表2.1 C#中内置的整数类型

类 型	说 明	范 围
sbyte	8位有符号整数	-128~127
short	16位有符号整数	-32768~32767
int	32位有符号整数	-2147483648~2147483647
long	64位有符号整数	-9223372036854775808~9223372036854775807
byte	8位无符号整数	0~255
ushort	16位无符号整数	0~65535
uint	32位无符号整数	0~4294967295
ulong	64位无符号整数	0~18446744073709551615

**技巧：**byte类型和short类型是范围比较小的整数，如果正整数的范围没有超过65535，则只需声明为ushort类型即可；当然更小的数值直接以byte类型作处理即可，但是使用这种类型时必须注意数值的大小，否则可能会导致运算溢出错误。

**例2.07** 创建一个控制台应用程序，在其中声明一个int类型的变量intOne，并初始化为300；声明一个byte类型的变量btOne，并初始化为220，最后输出，代码如下。（实例位置：光盘\mr02\sl\2.07）

```
using System;
using System.Collections.Generic;
```

```

using System.Linq;
using System.Text;
namespace UseInt
{
    class Program
    {
        static void Main(string[] args)
        {
            int intOne = 300; //声明一个 int 类型的变量 intOne
            byte btOne = 220; //声明一个 byte 类型的变量 btOne
            Console.WriteLine("intOne={0}", intOne); //输出 int 类型变量 intOne
            Console.WriteLine("btOne={0}", btOne); //输出 byte 类型变量 btOne
            Console.ReadLine();
        }
    }
}

```

程序运行结果如图 2.7 所示。

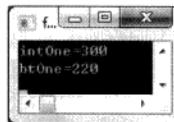


图 2.7 程序运行结果

 **说明：**如果将 byte 类型的变量 btOne 赋值为 266，重新编译程序，就会出现错误提示，主要原因是 byte 类型的变量是 8 位无符号整数，它的范围在 0~255 之间，而 266 已经超出了 byte 类型的范围，所以编译程序会出现错误提示。

## 2. 浮点类型

浮点类型主要用于处理含有小数的数值数据，它主要包含 float 和 double 两种数值类型，表 2.2 列出了这两种数值类型的描述信息。

表 2.2 浮点类型及描述

类 型	说 明	范 围
float	精确到 7 位数	$1.5 \times 10^{-45} \sim 3.4 \times 10^{38}$
double	精确到 15~16 位数	$5.0 \times 10^{-324} \sim 1.7 \times 10^{308}$

如果不做任何设置，则包含小数点的数值都被认为是 double 类型，如 9.27 这个数字，如果没有特别指定，这个数值的类型是 double 类型。如果要将数值以 float 类型来处理，就应通过强制使用 f 或 F 将其指定为 float 类型。

**例 2.08** 下面的代码用来将数值强制指定为 float 类型。

```

float theMySum = 9.27f; //使用 f 强制指定为 float 类型
float theMuSums = 1.12F; //使用 F 强制指定为 float 类型

```

如果要将数值强制指定为 double 类型，则需要使用 d 或 D 进行设置。

**例 2.09** 下面的代码用来将数值强制指定为 double 类型。

```

double myDou = 927d; //使用 d 强制指定为 double 类型
double mudou = 112D; //使用 D 强制指定为 double 类型

```

注意：程序中使用 float 类型时，必须在数值的后面跟随 f 或 F，否则编译器会直接将其作为 double 类型处理。

### 3. 布尔类型

布尔类型主要用来表示 True/False 值，一个布尔类型的变量，其值只能是 True 或 False，不能将其他的值指定给布尔类型变量，布尔类型变量不能与其他类型进行转换。

**例 2.10** 将 927 赋值给布尔类型变量 x，代码如下。

```
bool x = 927;
```

这样赋值显然是错误的，编译器会返回错误提示“常量值 927 无法转换为 bool”。布尔类型大多数被应用到流程控制语句中，如循环语句或 if 语句等。

### 2.3.2 引用类型的使用

引用类型是构建 C# 应用程序的主要对象类型数据，引用类型的变量又称为对象，可存储对实际数据的引用。C# 支持两个预定义的引用类型 object 和 string，其说明如表 2.3 所示。

表 2.3 C#预定义的引用类型及说明

类 型	说 明
object	object 类型在.NET Framework 中是 Object 的别名。在 C# 的统一类型系统中，所有类型（预定义类型、用户定义类型、引用类型和值类型）都是直接或间接从 Object 继承的
string	string 类型表示零或更多 Unicode 字符组成的序列

说明：尽管 string 是引用类型，但如果用到了相等运算符（== 和 !=），则表示比较 string 对象（而不是引用）的值。

在应用程序执行的过程中，引用类型以 new 创建对象实例，并且存储在堆栈中。堆栈是一种由系统弹性配置的内存空间，没有特定大小及存活时间，因此可以被弹性地运用于对象的访问。

引用类型具有如下特征。

- 必须在托管堆中为引用类型变量分配内存。
- 必须使用 new 关键字来创建引用类型变量。
- 在托管堆中分配的每个对象都有与之相关联的附加成员，且这些成员必须被初始化。
- 引用类型变量是由垃圾回收机制来管理的。
- 多个引用类型变量可以引用同一对象，在这种情形下，对一个变量的操作会影响另一个变量所引用的同一对象。
- 引用类型被赋值前的值都是 null。

所有被称为“类”的都是引用类型，主要包括类、接口、数组和委托等。下面通过一个实例来演示如何使用引用类型。

**例 2.11** 创建一个控制台应用程序，在其中创建一个类 C，在此类中建立一个字段 Value，并初始化为 0，然后在程序的其他位置通过 new 创建对此类的引用类型变量，最后输出，代码如下。（实例位置：光盘\mr\02\sl\2.11）

```
using System;
using System.Collections.Generic;
using System.Linq;
```

```

using System.Text;
namespace UseReference
{
    class Program
    {
        class C
        {
            public int Value = 0;
        }
        static void Main(string[] args)
        {
            int v1 = 0;
            int v2 = v1;
            v2 = 927;
            C r1 = new C();
            C r2 = r1;
            r2.Value = 112;
            Console.WriteLine("Values:{0},{1}", v1, v2);
            Console.WriteLine("Refs:{0},{1}", r1.Value, r2.Value); //输出引用类型对象的 Value 值
            Console.ReadLine();
        }
    }
}

```

程序运行结果如图 2.8 所示。



图 2.8 程序运行结果

## 2.4 声明并使用变量

程序开发时常常用到变量，变量被用来存储特定类型的数据，开发人员可以根据需要随时改变变量中所存储的数据值。变量具有名称、类型和值，其中，变量名是变量在程序源代码中的标识；变量类型确定它所代表的内存大小和类型；变量值是指它所代表的内存块中的数据。本节将对变量的声明、初始化及其作用域进行详细讲解。

### 2.4.1 变量的声明及初始化

#### 1. 变量的声明

变量的声明是指指定变量的名称和类型。变量的声明非常重要，未经声明的变量本身并不合法，也无法在程序中使用。在 C# 中，声明一个变量是由一个类型和跟在后面的一个或多个变量名组成的，多个变量之间用逗号分开，声明变量以分号结束。

**例 2.12** 声明一个整型变量 intOne，同时声明 3 个字符型变量 str1、str2 和 str3，代码如下。

```
int intOne; //声明一个整型变量
string str1, str2, str3; //同时声明3个字符串型变量
```

在第一行代码中声明了一个名称为 intOne 的整型变量；在第二行代码中声明了 3 个字符串型的变量，分别为 str1、str2 和 str3。

## 2. 变量的初始化

在声明变量时，还可以初始化变量，即在每个变量名后面加上为变量赋初始值的指令。

**例 2.13** 声明一个整型变量 a，并且赋值为 927，然后，再同时声明 3 个字符串型变量，并初始化，代码如下。

```
int a = 927; //初始化整型变量 a
string x = "明日科技", y = "C#编程词典", z = "C#从基础到项目实战"; //初始化字符串型变量 x、y 和 z
```

在声明变量时，要注意变量名的命名规则。C#的变量名是一种标识符，因此应符合标识符的命名规则。变量名是区分大小写的，下面列出变量名的命名规则。

- 变量名只能由数字、字母和下划线组成。
- 变量名的第一个字符只能是字母和下划线，而不能是数字。
- 不能使用关键字作为变量名。
- 一旦在一个语句块中定义了一个变量名，那么在变量的作用域内不能再定义同名的变量。

例 2.13 中的变量赋值方式其实是一种特殊的赋值方式，它在声明变量的同时给变量赋值，另外，还可以使用赋值运算符“=”（等号）来单独给变量赋值，将等号右边的值赋给左边的变量。

**例 2.14** 声明一个变量，并给变量赋值，代码如下。

```
int sum; //声明一个变量
sum = 2008; //使用赋值运算符“=”给变量赋值
```

在给变量赋值时，等号右边也可以是一个已经被赋值的变量。

**例 2.15** 首先声明两个变量 sum 和 num，然后将变量 sum 赋值为 927，最后将变量 sum 赋值给变量 num，代码如下。

```
int sum,num; //声明两个变量
sum = 927; //给变量 sum 赋值为 927
num = sum; //将变量 sum 赋值给变量 num
```

## 2.4.2 变量的作用域

变量的作用域就是可以访问该变量的代码区域。一般情况下，可以通过以下规则确定变量的作用域。

- 只要字段所属的类在某个作用域内，其字段也在该作用域内。
- 局部变量存在于表示声明该变量的块语句或方法结束的封闭花括号之前的作用域内。
- 在 for、while 或类似语句中声明的局部变量存在于该循环体内。

**例 2.16** 创建一个控制台应用程序，主要用来使用 for 循环将 0~20 的数字显示出来，该程序在实现时所声明的变量 i 就是一个局部变量，其作用域只限于 for 循环体内，代码如下。（实例位置：光盘\mr\02\sl\2.16）

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace VarField
{
    class Program
```

```

    {
        static void Main(string[] args)
        {
            //调用 for 语句循环输出数字
            for (int i = 0; i <= 20; i++)           //for 循环内的局部变量 i
            {
                Console.WriteLine(i.ToString());   //输出 0~20 的数字
            }
            Console.ReadLine();
        }
    }
}

```

程序运行结果为 0~20 的数字。

## 2.5 声明并使用常量

常量就是其值固定不变的数据，常量的值在编译时就已经确定了。常量的类型只能为下列类型之一，即 sbyte、byte、short、ushort、int、uint、long、ulong、char、float、double、decimal、bool 和 string 等。C# 中使用关键字 const 来声明常量，并且在声明常量时必须对其进行初始化。

**例 2.17** 声明一个正确的常量，同时再声明一个错误的常量，以便读者对比参考，代码如下。

```

const double PI = 3.1415926;           //正确的声明方法
const int MyInt;                      //错误：定义常量时没有初始化

```

与变量不同，常量一旦被定义，在常量的作用域内，常量的值就不能改变了。下面通过一个例子演示常量与变量的差异。

**例 2.18** 创建一个控制台应用程序，声明一个变量 myInt 并赋值为 927，然后再声明一个常量 myCInt 并赋值为 112，最后将变量 myInt 赋值为 1039，关键代码如下。（实例位置：光盘\mr\02\sl\2.18）

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace UseConst
{
    class Program
    {
        static void Main(string[] args)
        {
            int myInt = 927;                   //声明一个整型变量
            const int myCInt = 112;            //声明一个整型常量
            Console.WriteLine("变量 myInt={0}", myInt); //输出变量
            Console.WriteLine("常量 myCInt={0}", myCInt); //输出常量
            myInt = 1039;                    //重新将变量赋值为 1039
            Console.WriteLine("变量 myInt={0}", myInt); //输出变量
            Console.ReadLine();
        }
    }
}

```

程序运行结果如图 2.9 所示。

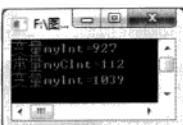


图 2.9 程序运行结果

变量 myInt 的初始化值为 927，而常量 myCInt 的值等于 112，由于变量的值是可以修改的，所以变量 myInt 可以重新被赋值为 1039 后再输出。通过查看输出结果，可以看到变量 myInt 的值可以被修改，但如果尝试修改常量 myCInt 的值，编译器会出现错误信息，阻止进行这样的操作。

## 2.6 数据类型转换

类型转换就是将一种类型转换成另一种类型，转换可以是隐式转换或者显式转换，本节将详细介绍这两种转换方式，并讲解有关装箱和拆箱的内容。

### 2.6.1 隐式类型转换

隐式类型转换就是指不需要声明就能进行的转换。在进行隐式类型转换时，编译器不需要进行检查就能安全地进行转换。表 2.4 列出了可以进行隐式类型转换的数据类型。

表 2.4 隐式类型转换表

源 类 型	目 标 类 型
sbyte	short、int、long、float、double、decimal
byte	short、ushort、int、uint、long、ulong、float、double 或 decimal
short	int、long、float、double 或 decimal
ushort	int、uint、long、ulong、float、double 或 decimal
int	long、float、double 或 decimal
uint	long、ulong、float、double 或 decimal
char	ushort、int、uint、long、ulong、float、double 或 decimal
float	double
ulong	float、double 或 decimal
long	float、double 或 decimal

说明：从 int、uint、long 或 ulong 到 float，以及从 long 或 ulong 到 double 的转换可能导致精度损失，但不会影响它的数量级。其他的隐式转换不会丢失任何信息。

**例 2.19** 将 int 类型的值隐式转换成 long 类型，代码如下。

```
int i = 927; //声明一个整型变量 i 并初始化为 927
long j = i; //隐式转换成 long 类型
```

### 2.6.2 显式类型转换

显式类型转换也可以称为强制类型转换，它需要在代码中明确地声明要转换的类型。如果在不存在隐

式转换的类型之间进行转换，就需要使用显式类型转换。表 2.5 列出了需要进行显式类型转换的数据类型。

表 2.5 显式类型转换表

源 类型	目 标 类型
sbyte	byte、ushort、uint、ulong 或 char
byte	sbyte 和 char
short	sbyte、byte、ushort、uint、ulong 或 char
ushort	sbyte、byte、short 或 char
int	sbyte、byte、short、ushort、uint、ulong 或 char
uint	sbyte、byte、short、ushort、int 或 char
char	sbyte、byte 或 short
float	sbyte、byte、short、ushort、int、uint、long、ulong、char 或 decimal
ulong	sbyte、byte、short、ushort、int、uint、long 或 char
long	sbyte、byte、short、ushort、int、uint、ulong 或 char
double	sbyte、byte、short、ushort、int、uint、ulong、long、char 或 decimal
decimal	sbyte、byte、short、ushort、int、uint、ulong、long、char 或 double

 **说明：**由于显式类型转换包括所有隐式类型转换和显式类型转换，因此总是可以使用强制转换表达式从任何数值类型转换为任何其他的数值类型。

**例 2.20** 创建一个控制台应用程序，将 double 类型的 x 进行显式类型转换，转换为 int 类型，代码如下。  
(实例位置：光盘\mr\02\s\l\2.20)

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace ShowConvert
{
    class Program
    {
        static void Main(string[] args)
        {
            double x = 19810927.0112;           //建立 double 类型变量 x
            int y = (int)x;                   //显式转换成整型变量 y
            Console.WriteLine(y);             //输出整型变量 y
            Console.ReadLine();
        }
    }
}
```

程序运行结果为 19810927。

另外，也可以通过 Convert 关键字进行显式类型转换，上面的例子还可以通过下面代码实现。

**例 2.21** 通过 Convert 关键字实现例 2.20 中的功能，代码如下。

```
double x = 19810927.0112;           //建立 double 类型变量 x
int y = Convert.ToInt32(x);          //通过 Convert 关键字转换
Console.WriteLine(y);               //输出整型变量 y
Console.ReadLine();
```

### 2.6.3 装箱和拆箱

C#语言类型系统中有两个重要的概念，分别是装箱和拆箱，通过装箱和拆箱，任何值类型都可以被当作 object 引用类型来看。本节将对装箱和拆箱进行详细讲解。

#### 1. 装箱

装箱实质上就是将值类型转换为引用类型的过程，如下面代码用来对 int 类型的变量 i 进行装箱操作。

```
int i = 2009; //声明一个值类型变量
object obj = i; //对值类型变量进行装箱操作
```

**例 2.22** 创建一个控制台应用程序，声明一个整型变量 i 并赋值为 2009，然后将其复制到装箱对象 obj 中，最后再改变变量 i 的值，代码如下。（实例位置：光盘\mr\02\s\2.22）

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace UpBox
{
    class Program
    {
        static void Main(string[] args)
        {
            int i = 2009; //声明一个 int 类型变量 i，并初始化为 2009
            object obj = i; //声明一个 object 类型对象 obj，其初始化值为 i
            Console.WriteLine("1、i 的值为{0}，装箱之后的对象为{1}", i, obj);
            i = 927; //重新将 i 赋值为 927
            Console.WriteLine("2、i 的值为{0}，装箱之后的对象为{1}", i, obj);
            Console.ReadLine();
        }
    }
}
```

程序运行结果如图 2.10 所示。

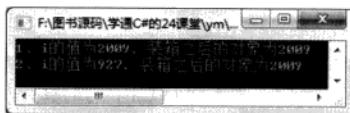


图 2.10 程序运行结果

从程序运行结果可以看出，值类型变量的值复制到装箱得到的对象中，装箱后改变值类型变量的值，并不会影响装箱对象的值。

#### 2. 拆箱

拆箱实质上就是将引用类型转换为值类型的过程，其执行过程大致可以分为以下两个阶段。

- (1) 检查对象的实例，看它是不是值类型的装箱值。
- (2) 把这个实例的值复制给值类型的变量。

**例 2.23** 创建一个控制台应用程序，声明一个整型变量 i 并赋值为 112，然后将其复制到装箱对象 obj

中，最后进行拆箱操作将装箱对象 obj 赋值给整型变量 j，代码如下。（实例位置：光盘\mr\02\s\2.23）

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace DownBox
{
    class Program
    {
        static void Main(string[] args)
        {
            int i = 112; //声明一个 int 类型的变量 i，并初始化为 112
            object obj = i; //执行装箱操作
            Console.WriteLine("装箱操作：值为{0}，装箱之后对象为{1}", i, obj);
            int j = (int)obj; //执行拆箱操作
            Console.WriteLine("拆箱操作：装箱对象为{0}，值为{1}", obj, j);
            Console.ReadLine();
        }
    }
}
```

程序运行结果如图 2.11 所示。

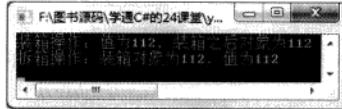


图 2.11 程序运行结果

从程序运行结果可以看出，拆箱后得到的值类型数据的值与装箱对象相等。

**◆ 注意：**在执行拆箱操作时需符合类型一致的原则，否则会出现异常。

## 2.7 运算符的使用

运算符是一种专门用来处理数据运算的特殊符号，其针对操作数进行运算，同时产生运算结果，数据变量结合运算符可形成完整的程序运算语句，本节将对 C# 中的运算符及其使用进行详细讲解。

### 2.7.1 算术运算符

+、-、\*、/ 和 % 运算符都称为算术运算符，它们分别用于进行加、减、乘、除和模（求余数）运算。下面对这几种算术运算符进行详细讲解。

#### 1. 加法运算符

加法运算符（+）通过两个数相加来执行标准的加法运算。

**例 2.24** 创建一个控制台应用程序，声明两个整型变量 M1 和 M2，并将 M1 赋值为 927，然后使 M2 的值为 M1 与 M1 相加之后的值，代码如下。（实例位置：光盘\mr\02\s\2.24）

```

static void Main(string[] args)
{
    int M1 = 927;                                //声明整型变量 M1，并赋值为 927
    int M2;                                       //声明整型变量 M2
    M2 = M1 + M1;                                 //M2 的值为 M1 与 M1 相加之后的值
    Console.WriteLine(M2.ToString());
    Console.Read();
}

```

程序的运行结果为 1854。

## 2. 减法运算符

减法运算符（-）通过从一个表达式中减去另外一个表达式的值来执行标准的减法运算。

**例 2.25** 创建一个控制台应用程序，声明两个 decimal 类型的变量 R1 和 R2，并分别赋值为 1112.82 和 9270.81，然后再声明一个 decimal 类型的变量 R3，使其值为 R2 减去 R1 之后得到的值，代码如下。（实例位置：光盘\mr\02\s\2.25）

```

static void Main(string[] args)
{
    decimal R1 = (decimal)1112.82;                //声明整型变量 R1，并赋值为 1112.82
    decimal R2 = (decimal)9270.81;                  //声明整型变量 R2，并赋值为 9270.81
    decimal R3;                                     //声明整型变量 R3
    R3 = R2 - R1;                                  //R3 的值为 R2 减去 R1 得到的值
    Console.WriteLine(R3.ToString());
    Console.Read();
}

```

程序运行结果如图 2.12 所示。

## 3. 乘法运算符

乘法运算符（\*）将两个表达式进行乘法运算并返回它们的乘积。

**例 2.26** 创建一个控制台应用程序，声明两个整型变量 int1 和 int2，并分别赋值为 10 和 20。再声明一个整型变量 intMul，使其值为 int1 和 int2 的乘积，代码如下。（实例位置：光盘\mr\02\s\2.26）

```

static void Main(string[] args)
{
    int int1 = 10;                                 //声明整型变量 int1，并赋值为 10
    int int2 = 20;                                 //声明整型变量 int2，并赋值为 20
    int intMul;                                    //声明整型变量 intMul
    intMul = int1 * int2;                          //使 intMul 的值为 int1 和 int2 的乘积
    Console.WriteLine(intMul.ToString());
    Console.Read();
}

```

程序的运行结果为 200。

## 4. 除法运算符

除法运算符（/）执行算术除运算，它用被除数表达式除以除数表达式而得到商。

 注意：除数表达式的结果不能为 0，否则将会出现异常。

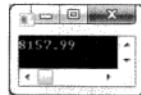


图 2.12 程序运行结果

**例 2.27** 创建一个控制台应用程序，声明两个整型变量 chushu1 和 chushu2，并分别赋值为 45 和 5。再声明一个整型变量 shang，使其值为 chushu1 除以 chushu2 得到的值，代码如下。（实例位置：光盘\mr\02\sl\2.27）

```
static void Main(string[] args)
{
    int chushu1 = 45; //声明整型变量 chushu1，并赋值为 45
    int chushu2 = 5; //声明整型变量 chushu2，并赋值为 5
    int shang; //声明整型变量 shang
    shang = chushu1 / chushu2; //使 shang 的值为 chushu1 除以 chushu2 得到的值
    Console.WriteLine(shang.ToString());
    Console.Read();
}
```

程序的运行结果为 9。

## 5. 求余运算符

求余运算符（%）返回被除数与除数相除之后的余数，通常用这个运算符来创建余数在特定范围内的等式。

**例 2.28** 创建一个控制台应用程序，声明两个整型变量 I1 和 I2，并分别赋值为 55 和 10。再声明一个整型变量 I3，使其值为 I1 与 I2 求余运算之后的值，代码如下。（实例位置：光盘\mr\02\sl\2.28）

```
static void Main(string[] args)
{
    int I1 = 55; //声明整型变量 I1，并赋值为 55
    int I2 = 10; //声明整型变量 I2，并赋值为 10
    int I3; //声明整型变量 I3
    I3 = I1 % I2; //使 I3 等于 I1 与 I2 求余运算之后的值
    Console.WriteLine(I3.ToString());
    Console.Read();
}
```

程序的运行结果为 5。

## 2.7.2 赋值运算符

赋值运算符用来为变量、属性和事件等元素赋值，它主要包括=、+=、-=、\*=、/=、%=、&=、|=、^=、<<=和>>=等运算符。赋值运算符的左操作数必须是变量、属性、索引器或事件类型的表达式，如果赋值运算符两边的操作数的类型不一致，就需要首先进行类型转换，然后再进行赋值。

在使用赋值运算符时，右操作数表达式所属的类型必须可隐式转换为左操作数所属的类型，运算符将右操作数的值赋给左操作数指定的变量、属性或索引器元素。C#中的赋值运算符及其运算规则如表 2.6 所示。

表 2.6 C#中的赋值运算符及其运算规则

名 称	运 算 符	运 算 规 则	意 义
赋值	=	将表达式赋值给变量	将右边的值赋给左边
加赋值	+=	x+=y	x=x+y
减赋值	-=	x-=y	x=x-y
除赋值	/=	x/=y	x=x/y
乘赋值	*=	x*=y	x=x*y
模赋值	%=	x%-=y	x=x%y
位与赋值	&=	x&=y	x=x&y
位或赋值	=	x =y	x=x y

续表

名称	运算符	运算规则	意义
右移赋值	$>>=$	$x >>= y$	$x = x >> y$
左移赋值	$<<=$	$x <<= y$	$x = x << y$
异或赋值	$^=$	$x ^= y$	$x = x ^ y$

下面以加赋值（ $+=$ ）运算符为例，举例说明赋值运算符的用法。

**例 2.29** 创建一个控制台应用程序，声明一个 int 类型的变量 i，并初始化为 927，然后通过加赋值运算符改变 i 的值，使其在原有的基础上增加 112，代码如下。（实例位置：光盘\mr\02\s\2.29）

```
static void Main(string[] args)
{
    int i = 927; //声明一个 int 类型的变量 i 并初始化为 927
    i += 112; //使用加赋值运算符
    Console.WriteLine("最后 i 的值为：{0}", i); //输出最后变量 i 的值
    Console.ReadLine();
}
```

程序运行结果如图 2.13 所示。

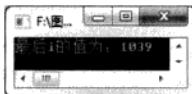


图 2.13 程序运行结果

### 2.7.3 关系运算符

关系运算符可以实现对两个值的比较运算，并且在比较运算之后会返回一个代表运算结果的布尔值。常见的关系运算符及说明如表 2.7 所示。

表 2.7 常见的关系运算符及说明

关系运算符	说 明	关系运算符	说 明
$==$	等于	$!=$	不等于
$>$	大于	$\geq$	大于等于
$<$	小于	$\leq$	小于等于

下面对这几种关系运算符进行详细讲解。

#### 1. 相等运算符

要查看两个表达式是否相等，可以使用相等运算符（ $==$ ）。相等运算符对整型、浮点型和枚举类型数据的操作是相同的，它只是简单地比较两个表达式，并返回一个布尔结果。

**例 2.30** 创建一个控制台应用程序，声明两个 decimal 类型变量 L1 和 L2，并分别赋值为 1981.00m 和 1982.00m，然后再声明一个 bool 类型的变量 result，使其值等于 L1 和 L2 进行相等运算后的返回值，代码如下。（实例位置：光盘\mr\02\s\2.30）

```
decimal L1 = 1981.00m; //声明 decimal 类型变量 L1
decimal L2 = 1982.00m; //声明 decimal 类型变量 L2
bool result; //声明 bool 类型变量 result
```

```

result = L1 == L2;           //使 result 等于 L1 和 L2 进行相等运算后的返回值
Console.WriteLine(result);   //输出运行结果
Console.ReadLine();
程序的运行结果为 False。

```

## 2. 不等运算符

不等运算符 ( $!=$ ) 是与相等运算符相反的运算符, C#中有两种格式的不等运算符可以应用到表达式中, 一种是普通的不等运算符 ( $!=$ ), 另外一种是相等运算符的否定!( $a == b$ )。

**例 2.31** 创建一个控制台应用程序, 声明两个 decimal 类型变量 S1 和 S2, 并分别赋值为 1981.00m 和 1982.00m, 然后再声明两个 bool 类型变量 result 和 result1, 使它们的值分别等于两种不等运算返回的值, 代码如下。 (实例位置: 光盘\mr\02\sl\2.31)

```

decimal S1 = 1981.00m;          //声明 decimal 类型变量 S1
decimal S2 = 1982.00m;          //声明 decimal 类型变量 S2
bool result;                   //声明 bool 类型变量 result
bool result1;                  //声明 bool 类型变量 result1
result = S1 != S2;             //获取不等运算返回值第一种方法
result1 = !(S1 == S2);         //获取不等运算返回值第二种方法
Console.WriteLine(result);
Console.WriteLine(result1);
Console.ReadLine();

```

程序运行结果如图 2.14 所示。

## 3. 小于运算符

如果要比较一个值是否小于另外一个值, 可以使用小于运算符 ( $<$ )。当使用该运算符时, 如果左边表达式的值小于右边表达式的值, 则结果为真; 否则, 结果为假。

**例 2.32** 创建一个控制台应用程序, 声明两个整型变量 U1 和 U2, 并分别赋值为 1112 和 927, 再声明一个 bool 类型变量 result, 使其值等于 U1 和 U2 进行小于运算的返回值, 代码如下。 (实例位置: 光盘\mr\02\sl\2.32)

```

int U1 = 1112;                //声明整型变量 U1
int U2 = 927;                 //声明整型变量 U2
bool result;                  //声明 bool 型变量 result
result = U1 < U2;            //使 result 等于 U1 和 U2 进行小于运算的返回值
Console.WriteLine(result);
Console.ReadLine();

```

程序的运行结果为 False。

## 4. 大于运算符

如果比较一个值是否大于另外一个值, 可以使用大于运算符 ( $>$ )。当使用该运算符时, 如果左边表达式的值大于右边表达式的值, 则结果为真; 否则, 结果为假。

**例 2.33** 创建一个控制台应用程序, 声明两个整型变量 F1 和 F2, 并分别赋值为 18 和 8, 再声明一个 bool 类型变量 result, 使其值等于 F1 和 F2 进行大于运算的返回值, 代码如下。(实例位置: 光盘\mr\02\sl\2.33)

```

int F1 = 18;                  //声明整型变量 F1
int F2 = 8;                   //声明整型变量 F2
bool result;                  //声明 bool 型变量 result
result = F1 > F2;            //使 result 等于 F1 和 F2 进行大于运算的返回值

```



图 2.14 程序运行结果

```
Console.WriteLine(result); //输出结果
Console.ReadLine();
程序的运行结果为 True。
```

### 5. 小于或等于运算符

小于或等于运算符 ( $\leq$ ) 用于查看某个值是否小于或等于另外一个值，当左边表达式的值小于或等于右边表达式的值时，结果为真；否则，结果为假。

**例 2.34** 创建一个控制台应用程序，声明两个整型变量 X1 和 X2，并分别赋值为 12 和 9，然后再声明一个 bool 类型变量 result，使其值等于 X1 和 X2 进行小于或等于运算的返回值，代码如下。（实例位置：光盘\mr\02\sl\2.34）

```
int X1 = 12; //声明整型变量 X1
int X2 = 9; //声明整型变量 X2
bool result; //声明 bool 型变量 result
//使 result 等于 X1 和 X2 进行小于或等于运算的返回值
result = X2 <= X1;
Console.WriteLine(result); //输出结果
Console.ReadLine();
```

程序运行结果如图 2.15 所示。

### 6. 大于或等于运算符

大于或等于运算符 ( $\geq$ ) 用于查看某个值是否大于或等于另外一个值，当运算符左边表达式的值大于或等于右边表达式的值时，结果为真；否则，结果为假。

**例 2.35** 创建一个控制台应用程序，声明两个整型变量 T1 和 T2，并分别赋值为 1112 和 927，再声明一个 bool 类型变量 result，使其值等于 T1 和 T2 进行大于或等于运算的返回值，代码如下。（实例位置：光盘\mr\02\sl\2.35）

```
int T1 = 1112; //声明整型变量 T1
int T2 = 927; //声明整型变量 T2
bool result; //声明 bool 型变量 result
//使 result 等于 T1 和 T2 进行大于或等于运算的返回值
result = T2 >= T1;
Console.WriteLine(result); //输出结果
Console.ReadLine();
```

程序运行结果如图 2.16 所示。



图 2.15 程序运行结果



图 2.16 程序运行结果

## 2.7.4 逻辑运算符

逻辑运算符对两个表达式执行布尔逻辑运算，C#中的逻辑运算符大体可以分为按位逻辑运算符和布尔逻辑运算符。

按位逻辑运算符是对两个整数表达式的相应位执行位逻辑运算，而布尔逻辑运算符是对两个布尔表达式执行布尔逻辑运算。

### 1. 按位“与”运算符

按位“与”运算符 (&) 比较两个整数的相应位，执行位“与”运算时，如果两个数的对应位都是 1，则返回相应的结果位是 1；而如果两个整数的相应位都是 0 或者其中一个位是 0，则返回相应的结果位是 0。

**例 2.36** 创建一个控制台应用程序，对变量 num1 和 num2 进行按位“与”运算，并输出结果，代码如下。（实例位置：光盘\mr\02\sl\2.36）

```
int num1 = 1;                                // 声明一个整型变量 num1
int num2 = 85;                                // 声明一个整型变量 num2
bool iseven;                                 // 声明一个 bool 类型的变量 iseven
iseven = (num1 & num2) == 0;                  // 获取两个变量位“与”运算后的返回值
Console.WriteLine(iseven);                   // 输出结果
Console.ReadLine();
```

为了使读者更好地理解按位“与”运算符的用法，下面对上述代码进行分析。

首先十进制数 1 对应的二进制数为 00000001，十进制数 85 对应的二进制数为 01010101，根据按位“与”运算符的定义可以得出 $(1 \& 85) = 1$ ，而上述代码中使 $(1 \& 85) = 0$ ，所以返回 False。

程序的运行结果为 False。

### 2. 按位“或”运算符

按位“或”运算符 (|) 用于比较两个整数的相应位，执行位“或”运算时，如果两个整数的对应位有一个是 1 或都是 1，则返回相应的结果位是 1；而如果两个整数的相应位都是 0，则返回相应的结果位是 0。

**例 2.37** 创建一个控制台应用程序，对变量 num1 和 num2 进行按位“或”运算，并输出结果，代码如下。（实例位置：光盘\mr\02\sl\2.37）

```
int num1 = 1;                                // 声明一个整型变量 num1
int num2 = 85;                                // 声明一个整型变量 num2
int iseven;                                 // 声明一个整型变量 iseven
iseven = (num1 | num2);                      // 获取两个变量位“或”运算后的返回值
Console.WriteLine(iseven);                   // 输出结果
Console.ReadLine();
```

下面对上述代码进行分析，十进制数 1 对应的二进制数为 00000001，十进制数 85 对应的二进制数是 01010101，根据按位“或”运算符的原理可以得出 $(1 | 85) = 85$ 。

程序运行结果如图 2.17 所示。

### 3. 按位“异或”运算符

按位“异或”运算符 (^) 比较两个整数的相应位，执行位“异或”运算时，如果两个整数的对应位一个是 1，而另外一个是 0，则返回相应的结果位是 1；而如果两个整数的相应位都是 1 或者都是 0，则返回相应的结果位是 0。

**例 2.38** 创建一个控制台应用程序，对变量 num1 和 num2 进行按位“异或”运算，并输出结果，代码如下。（实例位置：光盘\mr\02\sl\2.38）

```
int num1 = 1;                                // 声明一个整型变量 num1
int num2 = 85;                                // 声明一个整型变量 num2
int iseven;                                 // 声明一个整型变量 iseven
iseven = (num1 ^ num2);                      // 获取两个变量位“异或”运算后的返回值
Console.WriteLine(iseven);                   // 输出结果
Console.ReadLine();
```

下面对上述代码进行分析，十进制数 1 对应的二进制数为 00000001，十进制数 85 对应的二进制数是

01010101，根据按位“异或”运算符的原理可以得出 $(1^85)=84$ 。

程序的运行结果为 84。

#### 4. 布尔“与”运算符

布尔逻辑运算符对两个布尔表达式执行布尔逻辑计算，在运算时，先计算左边的表达式，然后计算右边的表达式，最后通过两个表达式之间的布尔逻辑运算符对两个表达式进行计算，并根据使用的运算符的类型返回布尔结果。

布尔“与”运算符（&）用于计算两个布尔表达式，当两个布尔表达式的结果都为真时，返回结果为真；否则，返回结果为假。

**例 2.39** 创建一个控制台应用程序，利用布尔“与”运算符判断运算返回值，代码如下。（实例位置：

光盘\mr\02\sl\2.39）

```
bool Bls = false; //声明一个 bool 类型变量 Bls
int Inum = 20; //声明一个整型变量 Inum
bool result; //声明一个 bool 类型变量 result
result=Bls&(Inum<30); //获取布尔“与”运算后的返回值
Console.WriteLine(result); //输出结果
Console.ReadLine();
```

程序的运行结果为 False。

#### 5. 布尔“或”运算符

布尔“或”运算符（|）用于计算两个布尔表达式的结果。当两个布尔表达式中有一个表达式返回真时，则结果为真；当两个布尔表达式的计算结果都为假时，则结果为假。

**例 2.40** 创建一个控制台应用程序，利用布尔“或”运算符判断运算返回值，代码如下。（实例位置：

光盘\mr\02\sl\2.40）

```
bool Bls = false; //声明一个 bool 类型变量 Bls
int Inum = 20; //声明一个整型变量 Inum
bool result; //声明一个 bool 类型变量 result
result=Bls | (Inum<30); //获取布尔“或”运算后的返回值
Console.WriteLine(result); //输出结果
Console.ReadLine();
```

程序运行结果如图 2.18 所示。



图 2.17 程序运行结果



图 2.18 程序运行结果

#### 6. 布尔“异或”运算符

布尔“异或”运算符（^）用于计算两个布尔表达式的结果，只有当其中一个表达式为真而另外一个表达式为假时，该表达式返回的结果才为真；而当两个表达式的计算结果都为真或都为假时，返回的结果为假。

**例 2.41** 创建一个控制台应用程序，利用布尔“异或”运算符，判断返回值是 True 还是 False，代码如下。（实例位置：光盘\mr\02\sl\2.41）

```
bool Bls = true; //声明一个 bool 类型变量 Bls
int Inum = 20; //声明一个整型变量 Inum
bool result; //声明一个 bool 类型变量 result
```

```

result=Bls ^ (Inum<30);
Console.WriteLine(result);
Console.ReadLine();
程序的运行结果为 False。
    
```

## 2.7.5 移位运算符

“<<”和“>>”运算符用于执行移位运算，分别称为左移位运算符和右移位运算符。对于  $X << N$  或  $X >> N$  形式的运算，含义是将  $X$  向左或向右移动  $N$  位，得到的结果的类型与  $X$  相同，在此处， $X$  的类型只能是 int、uint、long 或 ulong， $N$  的类型只能是 int，或者显示转换为这些类型之一，否则编译程序时会出现错误。

### 1. 左移位运算符

使用左移位运算符（<<）可以将数向左移位，其作用是所有的位都向左移动指定的次数，高位自动丢失，低位以零来填充。

**例 2.42** 创建一个控制台应用程序，使变量 intmax 向左移位 8 次，并输出结果，代码如下。（实例位置：光盘\mr\02\sl\2.42）

```

uint intmax = 4294967295;
uint bytemask;
bytemask = intmax << 8;
Console.WriteLine(bytemask);
Console.ReadLine();
    
```

//声明 uint 类型变量 intmax  
//声明 uint 类型变量 bytemask  
//使 intmax 左移 8 次  
//输出结果

程序运行结果如图 2.19 所示。

### 2. 右移位运算符

右移位运算符（>>）是把数向右移位，其作用是所有的位都向右移动指定的次数，低位自动丢失，高位以零来填充。

**例 2.43** 创建一个控制台应用程序，使变量 intmax 向右移位 16 次，并输出结果，代码如下。（实例位置：光盘\mr\02\sl\2.43）

```

uint intmax = 4294967295;
uint bytemask;
bytemask = intmax >> 16;
Console.WriteLine(bytemask);
Console.ReadLine();
    
```

//声明 uint 类型变量 intmax  
//声明 uint 类型变量 bytemask  
//使 intmax 右移 16 次  
//输出结果

程序的运行结果为 65535。

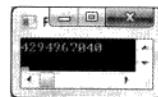


图 2.19 程序运行结果

## 2.7.6 其他特殊运算符

在 C# 中还有一些运算符不能简单地归到某个类型中，下面对这些特殊的运算符进行详细讲解。

### 1. is 运算符

is 运算符用于检查变量是否为指定的类型。如果是，返回真；否则，返回假。

**例 2.44** 创建一个控制台应用程序，使用 is 运算符判断变量 i 是否为整型，代码如下。（实例位置：光盘\mr\02\sl\2.44）

```

int i = 0;
bool result = i is int;
    
```

//声明整型变量 i  
//判断 i 是否为整型

```
Console.WriteLine(result); //输出结果
Console.ReadLine();
```

因为 i 是整型，所以运行程序返回值为 True。

## 2. 条件运算符

条件运算符(?)用来根据布尔型表达式的值返回两个值中的一个，如果条件为 True，则计算第一个表达式并以它的计算结果为准；如果为 False，则计算第二个表达式并以它的计算结果为准。

**例 2.45** 创建一个控制台应用程序，该程序中首先输入一个年份，然后按 Enter 键，使用条件运算符判断输入的年份是否为闰年，代码如下。（实例位置：光盘\mr\02\sl\2.45）

```
static void Main(string[] args)
{
    Console.Write("请输入一个年份："); //屏幕输入提示字符串
    string str = Console.ReadLine(); //获取用户输入的年份
    int year = Int32.Parse(str); //将输入的年份转换成 int 类型
    bool isleapyear=(year%400==0)||((year%4)==0)&&((year%100)!=0)); //计算输入的年份是否为闰年
    string yesno = isleapyear ? "是" : "不是";
    Console.WriteLine("{0}{1}年{2}闰年", year, yesno); //利用条件运算符输入“是”或“不是”
    Console.ReadLine(); //输出结果
}
```

程序运行结果如图 2.20 所示。

## 3. new 运算符

new 运算符用于创建一个新的类型实例，它有以下 3 种形式。

- 对象创建表达式，用于创建一个类类型的实例。
- 数组创建表达式，用于创建一个数组类型实例。
- 代表创建表达式，用于创建一个新的代表类型实例。

**例 2.46** 创建一个控制台应用程序，使用 new 运算符创建一个数组，并向数组中添加项目，然后输出数组中的项，代码如下。（实例位置：光盘\mr\02\sl\2.46）

```
static void Main(string[] args)
{
    string[] str = new string[5]; //创建具有 5 个元素的 string 类型数组
    str[0] = "mr1"; //为数组第 1 项赋值
    str[1] = "mr2"; //为数组第 2 项赋值
    str[2] = "mr3"; //为数组第 3 项赋值
    str[3] = "mr4"; //为数组第 4 项赋值
    str[4] = "mr5"; //为数组第 5 项赋值
    Console.WriteLine(str[0]); //输出数组第 1 项
    Console.WriteLine(str[1]); //输出数组第 2 项
    Console.WriteLine(str[2]); //输出数组第 3 项
    Console.WriteLine(str[3]); //输出数组第 4 项
    Console.WriteLine(str[4]); //输出数组第 5 项
    Console.ReadLine();
}
```

程序的运行结果为：

```
mr1
mr2
```



图 2.20 程序运行结果

mr3  
mr4  
mr5

#### 4. typeof 运算符

typeof 运算符用于获得系统原型对象的类型，也就是 Type 对象，其包含关于值类型和引用类型的信息。typeof 运算符可以在 C# 代码中的各个位置使用，以找出关于引用类型和值类型的信息。

**例 2.47** 创建一个控制台应用程序，利用 typeof 运算符获取引用类型的信息，并输出结果，代码如下。

(实例位置：光盘\mr\02\s\2.47)

```
static void Main(string[] args)
{
    Type mytype = typeof(int); //获取引用类型的信息
    Console.WriteLine("类型：{0}", mytype); //输出结果
    Console.ReadLine();
}
```

程序运行结果如图 2.21 所示。



图 2.21 程序运行结果

### 2.7.7 运算符的优先级

当表达式中包含一个以上的运算符时，程序会根据运算符的优先级进行运算，优先级高的运算符会比优先级低的运算符先执行。在表达式中，可以通过括号“()”来调整运算符的运算顺序，开发人员可以将想要优先运算的运算符放置在括号“()”中，当程序开始执行时，括号“()”内的运算符会被优先执行。表 2.8 列出了 C# 中所有运算符从高到低的优先级顺序。

表 2.8 运算符的优先级顺序

分 类	运 算 符	优先级顺序
基本	x.y、f(x)、a[x]、x++、x--、new、typeof、checked、unchecked	
一元	+、-、!、~、++、--、(T)x、	高
乘除	*、/、%	
加减	+	
移位	<<、>>	
比较	<、>、<=、>=、is、as	
相等	==、!=	
位与	&	
位异或	^	
位或		
逻辑与	&&	
逻辑或		
条件	?:	
赋值	=、+=、-=、*=、/=、%=<=、&=<=、^=<=、<<=<=、>>=<=	低

**技巧：**很多开发人员在程序中使用多种运算符时，都喜欢按照运算符的优先级进行使用，但是这样编写的代码复杂难懂，不利于以后的维护和他人阅读，建议读者使用“（”和“）”符号来区分运算符的运算顺序，这样代码会显得简单易懂。

## 2.8 照猫画虎——基本功训练

### 2.8.1 基本功训练1——使用“///”标记给代码段添加说明

**视频讲解：**光盘\mr\02\lx\使用“///”标记给代码段添加说明.exe

**实例位置：**光盘\mr\02\zmhh\01

使用“//”标记可以注释单行代码，使用“///”标记也可以注释单行代码，而且“///”标记单行注释的使用方法几乎与“//”标记相同。如果在某一行中出现“///”标记，那么“///”标记所在行后面的内容均为注释内容。但是“///”标记很少用于单行代码的注释，其一般用于为代码段添加说明，如图2.22所示。

```
/// <summary>
/// 程序入口方法
/// </summary>
/// <param name="args">用于向方法传递多个参数</param> -> 方法参数说明
/// <returns>方法返回整型数值</returns> -> 方法返回值说明
static int Main(string[] args)
{
    return 0;
}
```

图2.22 使用“///”标记为代码段添加说明

实现过程如下。

(1) 打开Visual Studio 2008开发环境，新建一个控制台应用程序，并将其命名为CodeCommentate。

(2) 程序主要代码如下。

```
///<summary>
///定义 program 类
///</summary>
class program
{
    ///<summary>
    ///程序入口方法
    ///</summary>
    ///<param name="args">用于向方法传递多个参数</param>
    ///<returns>方法返回整型数值</returns>
    static int Main(string[] args)
    {
        return 0;
    }
}
```

**技巧：**在代码的编写过程中应养成良好的编码习惯，适当地在代码中加入注释，使用“///”标记为代码段添加说明。

**照猫画虎：**创建一个 Windows 应用程序，尝试使用 “`///`” 标记给默认窗体 Form1 的后台代码添加说明。  
**(20 分)** (实例位置：光盘\mr\02\zmhh\01\_zmhh)

## 2.8.2 基本功训练 2——使用引号运算符进行赋值

**视频讲解：**光盘\mr\02\lx\使用引号运算符进行赋值.exe

**实例位置：**光盘\mr\02\zmhh\02

“字符串”是如此的常用，以至于开发人员几乎每天都要用到它，但是在使用字符串的过程中也许会发现，一些字符如双引号、反斜杠等，它们使用正常方法是无法显示出来的，那么，遇到这种情况就需要使用一些特殊的方法来正常显示字符串，下面的实例将实现输出双引号，实例运行效果如图 2.23 所示。



图 2.23 使用引号运算符进行赋值

实现过程如下。

- (1) 打开 Visual Studio 2008 开发环境，新建一个 Windows 窗体应用程序，并将其命名为 GetString。
- (2) 更改默认窗体 Form1 的 Name 属性为 Frm\_Main，向窗体中添加一个 Button 按钮，更改 Button 按钮的 Name 属性为 Btn\_Get，更改 Text 属性为“输出字符串”，此按钮控件用来显示字符串信息。
- (3) 程序主要代码如下。

```
private void btn_Get_Click(object sender, EventArgs e)
{
    string str = "=\"";
    string P_strOne, P_strTwo, P_strThree;
    P_strOne = "Hello," + "C#!";
    P_strTwo = "Hello," + '\u0022' + "C#" + '\u0022' + ";";
    P_strThree = "Hello," + str + "C#" + str + ";";
    MessageBox.Show(
        P_strOne + "      " + P_strTwo + "      " + P_strThree);
}
```

**说明：**从上面的代码中可以看到 “`\u0022`” 表示一个双引号字符。

**照猫画虎：**创建一个控制台应用程序，然后在控制台中输出带有单引号的字符串——‘我喜欢 C#’。  
**提示：**若要输出单引号，可以使用 “`\u0027`” 或 “`\'`” 这两个转义序列。  
**(20 分)** (实例位置：光盘\mr\02\zmhh\02\_zmhh)

## 2.8.3 基本功训练 3——使用 checked 关键字处理“溢出”错误

**视频讲解：**光盘\mr\02\lx\使用 checked 关键字处理“溢出”错误.exe

**实例位置：**光盘\mr\02\zmhh\03

在进行数学运算时，由于变量类型不同，数值的值域也有所不同。如果变量中的数值超出了变量的值域，则会出现“溢出”情况，出现该情况时变量中的数值将不准确。如何有效地防止溢出呢？本实例演示了使用 checked 关键字检查算术运算溢出的情况，实例运行效果如图 2.24 所示。

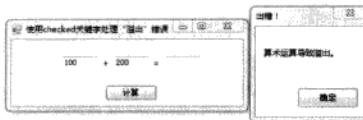


图 2.24 使用 checked 关键字处理“溢出”错误

实现过程如下。

(1) 打开 Visual Studio 2008 开发环境，新建一个 Windows 窗体应用程序，并将其命名为 CalcRAreaByAbstractClass。

(2) 更改默认窗体 Form1 的 Name 属性为 Checked，向窗体中添加 3 个 TextBox 控件，分别用于输入计算溢出的数值；添加一个 Button 按钮，用于数值计算。

(3) 程序主要代码如下。

```
private void btn_Get_Click(object sender, EventArgs e)
{
    byte bt_One, bt_Two;
    if (byte.TryParse(
        txt_Add_One.Text, out bt_One)
        && byte.TryParse(txt_Add_Two.Text, out bt_Two))
    {
        try
        {
            checked { bt_One += bt_Two; }
            txt_Result.Text = bt_One.ToString();
        }
        catch (OverflowException ex)
        {
            MessageBox.Show(ex.Message, "出错！");
        }
    }
    else
    {
        MessageBox.Show("请输入 255 以内的数字!");
    }
}
```

//定义两个 byte 类型变量  
//对两个 byte 类型变量赋值

//使用 checked 关键字判断是否有溢出  
//输出相加后的结果

//输出异常信息

//输出错误信息

**说明：**本实例在演示溢出时所使用的数值类型为 byte，byte 类型的值域为 0~255，所以如果 byte 类型变量的值高于 255 或小于 0 都会出现溢出。

**熊猫画虎：**创建一个控制台应用程序，声明 3 个 sbyte 类型的变量 sbNum1、sbNum2 和 sbResult，并将变量 sbNum1 和 sbNum2 分别初始化为 100。然后使 sbResult 等于 sbNum1 和 sbNum2 的乘积，并使用 checked 关键字检查算术运算溢出的情况。提示：sbyte 类型的取值范围是 127~128。（20 分）（实例位置：光盘\mr\02\zmhh\03\_zmhh）

## 2.8.4 基本功训练 4——使用 typeof 关键字获取类的内部结构

**视频讲解：**光盘\mr\02\lx\使用 typeof 关键字获取类的内部结构.exe

**实例位置：**光盘\mr\02\zmhh\04

typeof 关键字用于获取类的 System.Type 对象，调用 Type 对象的 GetMethods 方法，可以得到类中定义的方法。

法对象的集合；调用方法对象集合中每一个方法对象的 GetParameters 方法，会得到每一个方法的参数集合。本实例实现获取 System.Int32 类的内部结构，并通过 richTextBox 控件显示该类中定义的方法及方法参数的信息，实例运行效果如图 2.25 所示。

实现过程如下。

(1) 打开 Visual Studio 2008 开发环境，新建一个 Windows 窗体应用程序，并将其命名为 TypeOf。

(2) 更改默认窗体 Form1 的 Name 属性为 Frm\_Main，向窗体中添加一个 richTextBox 控件，用于显示类的内部结构；添加一个 Button 按钮，用于执行显示操作。

(3) 程序主要代码如下。

```
private void btn_Get_Click(object sender, EventArgs e)
{
    Type type = typeof(System.Int32);
    foreach (MethodInfo method in type.GetMethods())
    {
        rtbox_text.AppendText(
            "方法名称：" + method.Name + Environment.NewLine); //输出方法名称
        foreach (ParameterInfo parameter in method.GetParameters()) //遍历公共方法中的所有参数
        {
            rtbox_text.AppendText(
                "    参数：" + parameter.Name + Environment.NewLine); //输出参数名称
        }
    }
}
```

//获得 Int32 类的 Type 对象  
//遍历 Int32 类中定义的所有公共方法

**技巧：**除了上述用法外，还可以使用 `typeof` 关键字判断窗体控件类型，并清空窗体中所有 TextBox 控件的文本信息。

**熊猫虎：**使用 `typeof` 关键字获取 Object 类的内部结构，并输出到控制台。(20 分)(实例位置：光盘\mr\02\zmhh\04\_zmhh)

## 2.8.5 基本功训练 5——使用 `using` 关键字有效回收资源

视频讲解：光盘\mr\02\lx\使用 using 关键字有效回收资源.exe

实例位置：光盘\mr\02\zmhh\05

在进行文件操作后要显式的调用文件流的 `Close` 方法释放文件资源，在使用数据库连接时也要调用连接对象的 `Close` 方法释放数据库资源。如果忘记调用 `Close` 方法，有可能会导致程序发生异常，而且还会导致垃圾收集器多次回收对象，造成内存的浪费。本实例演示使用 `using` 关键字有效回收资源，实例运行效果如图 2.26 所示。

实现过程如下。

(1) 打开 Visual Studio 2008 开发环境，新建一个 Windows 窗体应用程序，并将其命名为 Using。



图 2.25 使用 `typeof` 关键字获取类的内部结构

```
//获得 Int32 类的 Type 对象
//遍历 Int32 类中定义的所有公共方法
```

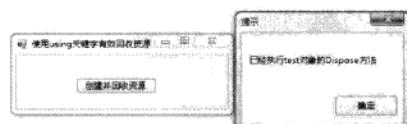


图 2.26 使用 `using` 关键字有效回收资源

(2) 更改默认窗体 Form1 的 Name 属性为 Frm\_Main，向窗体中添加一个 Button 按钮，用于演示使用 using 关键字回收资源。

(3) 程序主要代码如下。

```
private void btn_true_Click(object sender, EventArgs e)
{
    using (new test())
    {
    }
}
class test : IDisposable
{
    public void Dispose()
    {
        MessageBox.Show(
            "已经执行 test 对象的 Dispose 方法", "提示");
    }
}
```

注意：使用 using 关键字时，在其后面小括号内创建的对象必须要实现 IDisposable 接口，或者此对象的基类已经实现了 IDisposable 接口。

照猫画虎：创建一个控制台应用程序，在 Program.cs 文件中定义一个类 MyUsing，然后让该类实现 IDisposable 接口，并且在 Dispose 方法中输出“Dispose 被调用”字样，最后在 Main 方法中使用 using 关键字创建 MyUsing 类的对象。(20 分)(实例位置：光盘\mr\02\zmhh\05\_zmhh)

照猫画虎栏目分数统计：

照猫画虎题目	1	2	3	4	5	总分数	
分数							

## 2.9 情景应用——拓展与实践

### 2.9.1 情景应用 1——检查对象是否与给定类型兼容

视频讲解：光盘\mr\02\lx\检查对象是否与给定类型兼容.exe

实例位置：光盘\mr\02\qjyy\01

程序开发过程中经常会使用类型转换，如果类型转换不成功则会出现异常，从抛出异常到捕获并处理异常，无形中增加了系统的开销，而且太过频繁的处理异常还会严重影响系统的稳定性。is 关键字可以有效地解决上面出现的问题，其用于检查对象是否与给定类型兼容，如果兼容将返回 true，如果不兼容则返回 false。在进行类型转换前，可以先使用 is 关键字判断对象是否与指定类型兼容，如果兼容才进行转换，这样的转换是安全的，本实例运行效果如图 2.27 所示。



图 2.27 检查对象是否与给定类型兼容

实现过程如下。

- (1) 打开 Visual Studio 2008 开发环境，新建一个 Windows 窗体应用程序，并将其命名为 Equal。
- (2) 更改默认窗体 Form1 的 Name 属性为 Frm\_Main，向窗体中添加一个 Button 按钮，用于判断选定对象与类型是否兼容；添加两个 GroupBox 控件，分别向两个 GroupBox 控件中添加两个 RadioButton 单选按钮，用于定义选择的对象与类型。

(3) 程序主要代码如下。

```
private void btn_Get_Click(object sender, EventArgs e)
{
    object P_obj = rbtn_target1.Checked ? //正确地为变量添加引用
        (object)"C# 编程词典" : new System.IO.FileInfo(@"d:\");
    if (rbtn_class1.Checked) //判断选择了哪一个类型
    {
        if (P_obj is System.String) //判断对象是否为字符串类型
            MessageBox.Show("对象与指定类型兼容", "提示！");
        else //提示兼容信息
            MessageBox.Show("对象与指定类型不兼容", "提示！");
    }
    else //提示不兼容信息
    {
        if (P_obj is System.IO.FileInfo) //判断对象是否为文件类型
            MessageBox.Show("对象与指定类型兼容", "提示！");
        else //提示兼容信息
            MessageBox.Show("对象与指定类型不兼容", "提示！");
    }
}
```

**DIY：** 创建一个控制台应用程序，定义一个整型变量，然后使用 is 关键字判断该变量是否与 Object 类型兼容，并将判断结果输出到控制台中。(20 分)(实例位置：光盘\mr\02\qjyy\01\_diy)

## 2.9.2 情景应用 2——使用算术运算符开发简单计算器

■ 视频讲解：光盘\mr\02\lx\使用算术运算符开发简单计算器.exe

■ 实例位置：光盘\mr\02\qjyy\02

算术运算符主要用于算术运算，本实例使用算术运算符开发一个简单计算器，该计算器只是进行累加运算。首先使用泛型集合记录每一个将要进行累加计算的数值，然后使用 foreach 遍历集合累加所有数值，同时显示计算表达式，最后通过算术运算符累加得到计算结果，实例运行效果如图 2.28 所示。

实现过程如下。

- (1) 打开 Visual Studio 2008 开发环境，新建一个 Windows 窗体应用程序，并将其命名为 Calculator。

- (2) 更改默认窗体 Form1 的 Name 属性为 Frm\_Main，向窗体中添加一个 TextBox 控件，用于显示计算结果，向窗体中添加两个容器控件 GroupBox。在第一个 GroupBox 中放入多个 Button 按钮，用于输入数值；在第二个 GroupBox 中放 3 个 Button 按钮，分别用来实现加法运算、显示结果和清空信息 3 种操作。



图 2.28 使用算术运算符  
开发简单计算器

(3) 程序主要代码如下。

```

private List<double> G_list_value = new List<double>(); //记录累加数值
private bool G_bt_add = false; //判断是否刚刚按下+号
private bool G_bt_value = false; //判断是否刚刚按下=号
private bool G_bt_key = false; //防止连续按+号
void GetValue() //该方法用于计算累加数值并输出
{
    double P_dbl_temp = 0; //定义局部变量
    foreach (double d in G_list_value) //遍历集合
    {
        P_dbl_temp += d; //计算累加结果
    }
    txt_value.Text = P_dbl_temp.ToString(); //显示累加结果
}
string GetString() //该方法用于得到数值的字符串表示
{
    string P_str_temp = string.Empty; //定义局部变量
    for (int i = 0; i < G_list_value.Count; i++) //遍历集合
    {
        if (i != 0) //判断是否是第一个数值
        {
            P_str_temp += //产生字符串
                "+" + G_list_value[i].ToString();
        }
        else
        {
            P_str_temp = //产生字符串
                G_list_value[i].ToString();
        }
    }
    return P_str_temp; //返回字符串
}
private void btn_add_Click(object sender, EventArgs e)
{
    if (G_bt_value) //判断是否刚刚按下=号
    {
        G_bt_value = false; //设置刚刚按下的不是=号
        G_bt_key = true; //设置刚刚按下的的是+号
    }
    else
    {
        if (!G_bt_key) //判断是否连续按+号
        {
            G_list_value.Add(double.Parse(txt_value.Text)); //向集合中添加累加的数值
            GetValue(); //计算累加数值并输出
            lb_express.Text = GetString(); //得到数值的字符串表示
            G_bt_add = true; //设置已经按下+号
            G_bt_key = true; //防止多次按下+号
        }
    }
}
}

```

注意：在使用加法运算符时要注意，一些数值，如两个 byte 数值相加后会将数值的类型提升为 int 类型。

DIY：制作一个具有累乘功能的简单计算器。提示：只需修改上面实例中 GetValue 方法的相关代码即可。（20 分）（实例位置：光盘\mr\02\qjyy\02\_diy）

### 2.9.3 情景应用 3——使用“^”运算符对数字进行加密

视频讲解：光盘\mr\02\lx\使用“^”运算符对数字进行加密.exe

实例位置：光盘\mr\02\qjyy\03

在介绍实例之前，首先来了解一下加密的概念，加密是指通过某种特殊的方法，更改已有信息的内容，这使得未授权的用户即使得到了加密后的信息，如果没有正确的解密方法，那么也无法得到该信息的真实内容。在本实例中，通过使用异或运算符“^”简单地实现了对数字加密的功能，实例运行效果如图 2.29 所示。

实现过程如下。

(1) 打开 Visual Studio 2008 开发环境，新建一个 Windows 窗体应用程序，并将其命名为 Encrypt。

(2) 更改默认窗体 Form1 的 Name 属性为 Frm\_Main，向窗体中添加两个容器控件 GroupBox。在第一个GroupBox 中放入 3 个 TextBox 控件和一个 Button 按钮，分别用于输入数值、输入加密数值、显示加密后的数值和计算加密信息；在第二个GroupBox 中放入一个 TextBox 控件和一个 Button 按钮，分别用于显示解密后的信息和计算解密信息。

(3) 程序主要代码如下。

```
private void btn_Encrypt_Click(object sender, EventArgs e)
{
    int P_int_Num, P_int_Key;
    if (int.TryParse(txt_Num.Text, out P_int_Num) && int.TryParse(txt_Key.Text, out P_int_Key))
        //判断输入是否是数值
    {
        txt_Encrypt.Text = (P_int_Num ^ P_int_Key).ToString();
        //加密数值
    }
    else
    {
        MessageBox.Show("请输入数值", "出现错误！");
    }
}
private void btn_Revert_Click(object sender, EventArgs e)
{
    int P_int_Key, P_int_Encrypt;
    if (int.TryParse(txt_Encrypt.Text, out P_int_Encrypt)
        && int.TryParse(txt_Key.Text, out P_int_Key))
    {
        txt_Revert.Text = (P_int_Encrypt ^ P_int_Key).ToString();
        //解密数值
    }
}
```

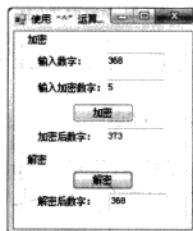


图 2.29 使用“^”运算符对数字进行加密

```
        MessageBox.Show("请输入数值", "出现错误!"); //提示输入信息不正确  
    }  
}
```

**注意：**由于在本实例中只是简单地使用了异或运算符计算两个整型数值以达到加密目的，所以本实例只可以对整型数值进行加密运算，并不适合其他数据的加密。

DIY：使用“^”运算符对字母进行加密。提示：首先将字母转换为其对应的 ASCII 码，然后再使用“^”运算符加密，解密时需要将得到的 ASCII 值转换为对应的字母。(20 分)(实例位置：光盘\mr\02\qjyy\03\div)

#### 2.9.4 情景应用 4——巧用移位运算符获取汉字编码值

视频讲解：光盘\mr\02\lx\巧用移位运算符获取汉字编码值.exe

 实例位置：光盘\mr\02\qjvv\04

我们伟大的祖国拥有着五千年的悠久历史和灿烂的文化，而汉字是几千年来文化的一种沉积。汉字在计算机中以字符的形式出现，并且每个汉字在计算机中都对应着一个数字形式的编码，本实例通过使用移位运算符获取汉字编码值，实例运行效果如图 2.30 所示。

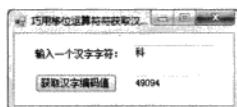


图 2.30 巧用移位运算符获取汉字编码值

实现过程如下。

- (1) 打开 Visual Studio 2008 开发环境，新建一个 Windows 窗体应用程序，并将其命名为 GetCode。
  - (2) 更改默认窗体 Form1 中的 Name 属性为 Frm\_Main，向窗体中添加两个 TextBox 控件和一个 Button 按钮。两个 TextBox 控件分别用于添加汉字和显示汉字编码值，Button 按钮用于计算汉字编码值。
  - (3) 程序主要代码如下。

```
private void btn_Get_Click(c
```

```
try
{
    char chr = txt_chr.Text[0]; //获得一个汉字字符
    byte[] gb2312_bt = //使用gb2312编码方式获得字节序列
        Encoding.GetEncoding("gb2312").GetBytes(new Char[] { chr });
    int n = (int)gb2312_bt[0] << 8; //将字节序列的第一个字节向左移8位
    n += (int)gb2312_bt[1]; //第一个字节移8位后与第二个字节相加得到汉字编码
    txt_Num.Text = n.ToString(); //显示汉字编码
}
catch (Exception)
{
    MessageBox.Show("请输入汉字字符！", "出现错误！"); //异常提示信息
}
```

**技巧:** 在进行移位运算时, 当数值的二进制数每次向左移 1 位就相当于乘以 2, 当数值每次向右移 1 位就相当于除以 2。

DIY：上面的实例只能获取一个汉字的编码，这里尝试使用移位运算符同时获取多个汉字的编码。提示：可以使用 for 语句循环输出多个汉字对应的编码，关于 for 语句的使用可以查阅 CSDN 的相关文档。(20)

分) (实例位置: 光盘\mr\02\qjyy\04\_diy)

## 2.9.5 情景应用 5——使用条件运算符判断指定年份是不是闰年

视频讲解: 光盘\mr\02\lx\使用条件运算符判断指定年份是不是闰年.exe

实例位置: 光盘\mr\02\qjyy\05

闰年是为了弥补因为历法规规定造成的每一年的天数与地球实际公转周期的时间差而设定的, 而补上时间差的年份被称为闰年, 闰年共有 366 天。那么如何计算闰年呢? 计算闰年的方法很简单, 指定年份如果能被 400 整除就为闰年, 或者指定年份可以整除 4 但不能整除 100 也为闰年。本实例使用条件运算符判断用户输入的年份是否为闰年, 实例运行效果如图 2.31 所示。



图 2.31 使用条件运算符判断指定年份是不是闰年

实现过程如下。

- (1) 打开 Visual Studio 2008 开发环境, 新建一个 Windows 窗体应用程序, 并将其命名为 GetYear。
- (2) 更改默认窗体 Form1 中的 Name 属性为 Frm\_Main, 向窗体中添加一个 TextBox 控件和一个 Button 按钮。TextBox 控件用于用户输入年份, Button 按钮用于计算输入年份是否为闰年。

(3) 程序主要代码如下。

```
private void btn_GetMessage_Click(object sender, EventArgs e)
{
    ushort P_usint_temp;
    if (ushort.TryParse(txt_year.Text, out P_usint_temp))
    {
        MessageBox.Show(
            (P_usint_temp % 4 == 0 && P_usint_temp % 100 != 0) //输出计算结果
            || P_usint_temp % 400 == 0 ? "输入的是闰年!" : "输入的不是闰年!",
            "提示!");
    }
    else
    {
        MessageBox.Show("请输入正确数值!", "提示!"); //提示输入数值不正确
    }
}
```

说明: 条件运算符(?)又称为三元运算符, 它会根据布尔类型值或布尔类型表达式返回两个值中的一个。

DIY: 使用条件运算符判断奇偶数。提示: 判断一个整数是奇数或偶数, 可以让该整数除以 2, 若余数等于 0, 则为偶数, 否则为奇数。(20 分)(实例位置: 光盘\mr\02\qjyy\05\_diy)

情景应用 DIY 栏目分数统计:

DIY 题目	1	2	3	4	5	总分数	
分数							

## 2.10 自我测试

### 一、选择题（每题 10 分，5 道题）

1. 下面哪些类型不是值类型（ ）。  
A. 布尔类型      B. 结构类型      C. 枚举类型      D. String 类型
2. 关于 long 类型描述正确的是（ ）。  
A. 有符号 64 位整数      B. 有符号 32 位整数      C. 无符号 16 位整数      D. 无符号 32 位整数
3. 下面哪些概念不可以定义为常量（ ）。  
A. 光速      B. 圆周率      C. 每年的月份数      D. 一年内总秒数
4. 关于装箱和拆箱，论述不正确的是（ ）。  
A. 装箱实质上就是将值类型转换为引用类型      B. 拆箱实质上就是将引用类型转换为值类型  
C. 在执行拆箱操作时，要符合类型一致的原则      D. 装箱和拆箱都不需要强制类型转换
5. 下面是一个关于转义字符使用的控制台应用程序：

```
static void Main(string[] args)
{
    String str = "大家" + '\u0022' + "好" + '\n';
    Console.WriteLine(str);
    Console.ReadLine();
}
```

程序运行后，其输出结果应为（ ）。

- A. 大家好      B. “大家好”      C. 大家“好”      D. ‘大家好’

### 二、填空题（每题 10 分，5 道题）

1. 在编写代码中，通常使用（ ）关键字来定义命名空间，使用（ ）关键字来引入命名空间。

2. 基本数据类型包括（ ）和（ ）两大类。
3. C#有两个预定义类型，它们分别是（ ）类型和（ ）类型。

4. 关于以下的 C#程序代码：

```
public class Person
{
    String Name;
    int Age;
}
public static void Main()
{
    Person per = new Person();
    int PerCount = 100;
}
```

在上面的代码中定义了 4 个变量，其中（ ）和（ ）是值类型变量；而（ ）和（ ）是引用类型变量。

5. 在 C#中，下列代码运行后，变量 intNum 的值是（ ）。

```
int a = 4, b = 9, c = 12, intNum = 0;
intNum = a < b ? a : b;
intNum = c > intNum ? c : intNum;
```

## 测试分数统计：

类别	第 1 题	第 2 题	第 3 题	第 4 题	第 5 题
选择题分数					
填空题分数					
总分数					

## 2.11 行动指南

开始日期：\_\_\_\_年\_\_\_\_月\_\_\_\_日

结束日期：\_\_\_\_年\_\_\_\_月\_\_\_\_日

序号	内 容	行动指南
1	照猫画虎栏目 分数( )	分数>75 分 优秀，基本功掌握得不错，加油！
		75 分>分数>50 分 及格，知识掌握得不牢，重新做一遍照猫画虎。
		分数<50 分 请使用光盘中提供的“实战能力测试系统”进行提高训练。
	情景应用栏目 分数( )	分数>75 分 优秀，综合应用能力很强。
		75 分>分数>50 分 及格，综合应用能力需提高，再练一遍情景应用。
		分数<50 分 请使用光盘中提供的“实战能力测试系统”进行提高训练。
	自我测试栏目 分数( )	分数>75 分 优秀，有成为编程高手的潜质。
		分数<75 分 请使用光盘中提供的“实战能力测试系统”进行提高训练。
	综合评价	反复训练后，以上各项分数都在 75 分以上，方可进入下一堂课学习。
2	编程能力培养：本栏目提供了一些实践题目，对于培养编程能力很有效，要做好。	1. 设计用户登录程序。提示：在控制台中输入用户名和密码，然后使用关系运算符“==”和逻辑运算符“&&”判断输入的用户名和密码是否与指定的用户名和密码相匹配，最后使用条件运算符“?:”判断用户是否登录成功。
		2. 编写判断某个数是否为素数的程序。提示：判断素数的方法主要是对给出的数值进行开方，并利用这个值除以从一到开方或仅小于开方后的最大整数，如果不能被整除，则是素数，否则不是素数。
		3. 设计一个能够自动清除窗体中所有 TextBox 控件的方法。提示：可使用 typeof 关键字判断窗体上的控件类型，若判断该控件的类型为 TextBox，则设置其 Text 属性值为空字符串。
3	编程习惯培养：本堂课培养读者在学习开发中记笔记的习惯，把开发中遇到的问题、总结的经验记录下来。	
4	创新能力培养：看看身边有没有可以用编程解决的问题，记录到右边的表格中。根据学习进度，尝试编写解决这些问题的小程序。	

## 2.12 成功可以复制——中国第一程序员求伯君

求伯君，1964年11月26日出生于浙江新昌县。1980年，求伯君考入国防科技大学系统工程与数学系学习，因系统工程需要大量使用计算机，求伯君也比同龄人有了更早接触计算机的机会，这也成为求伯君的第一次人生机遇。1983年，求伯君利用业余时间为国防科技大学图书馆开发了图书馆借还书管理系统，这是求伯君编写的第一个能用的程序，为此求伯君获得了40多元钱的奖励，从此，求伯君与程序结下了不解之缘。

1986年，求伯君改进编写了一个支持多种打印机的西山文字打印驱动程序，以2000元的价格卖给了四通公司，从此求伯君也成为了四通公司的一员。四通公司将该程序定价为500元一套，卖了好几百套。

1988年，求伯君离开四通公司，加入香港金山公司，负责开发用于DOS系统的中文文字处理软件——WPS。为了加快开发速度，求伯君把自己关在深圳的一个房间里，只要是醒着，就不停地写。什么时候困了，就睡一会儿，饿了就吃方便面。在这样的一年零4个月中，求伯君生了3次病，第一次肝炎，第二次肝炎复发，第三次又复发，每次住院一个月到两个月。第二次肝炎复发正是软件开发最紧要的关头，求伯君把计算机搬到病房里继续工作。求伯君在这种孤独中写下了有十几万行代码的WPS软件。WPS一进入市场就凭着良好的口碑迅速占领市场，当年就卖出了3万多套，销售额达6000多万元。

1994年，求伯君在珠海成立珠海金山电脑公司，继续从事办公软件开发。开发的第一个软件产品——盘古组件（里面有WPS、电子表和字典），遇到了Word的挑战，最终以失败告终。1996年金山陷入苦难时期，运营资金紧张、员工信心不足。

在危难时刻，求伯君毅然卖掉了别墅，凭着对WPS的深厚感情克服了重重困难与障碍，1997年推出WPS 97。WPS 97推出仅两个多月，就卖出了13000套，金山也借此迈过最艰难的时刻。

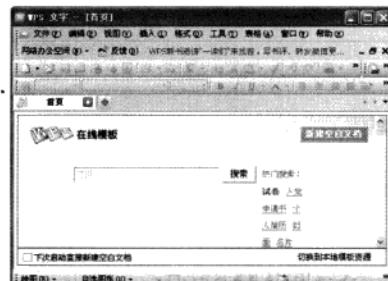
如今的金山软件已占据通用软件全国第一，2009年总营业收入达10.22亿元。

### 经典语录

如果从开始就想着怎样赚钱，我也不会有今天。事业和金钱无关。当你全身心投入开发的时候，不给你钱你也要干。开发时，根本没有心思考虑报酬。只有先成就了业，才有资格谈报酬。

### 深度评价

求伯君的成功告诉我们，热爱编程的程序员们要耐得住寂寞，坚定信念，用自己的热情全身心地投入才能创造成功的事业。



WPS 软件界面



# 第 3 堂课

## 程序流程控制

( 视频讲解：116分钟)

程序设计中最主要的就是算法，而算法的基础是语句，语句是程序完成一次完整操作的基本单位，默认情况下，程序的语句是顺序执行的。但是，如果一个程序只能顺序执行各语句，那么程序功能会受到很大的限制。C#中有很多流程控制语句，通过这些语句可以控制程序代码的执行顺序，提高程序的灵活性，从而实现比较复杂的程序。本堂课将对流程控制语句进行详细讲解。

学习摘要：

- ▶ 掌握 C# 中常用的程序设计算法
- ▶ 掌握 if 语句的使用
- ▶ 掌握嵌套 if 语句的使用
- ▶ 掌握 switch 语句的使用
- ▶ 熟悉 if…else 语句与 switch 语句的区别
- ▶ 掌握 while 语句的使用方法
- ▶ 掌握 do…while 语句的使用方法
- ▶ 熟悉 while 语句与 do…while 语句的区别
- ▶ 掌握 for 语句的使用方法
- ▶ 掌握 foreach 语句的使用方法
- ▶ 掌握常用的几种跳转语句的使用方法

## 3.1 C# 程序设计算法

一般来说，每一个程序都包括两方面内容，一是对数据的描述，即数据结构，二是对操作的描述，即算法。算法是程序设计的灵魂，对程序开发人员来说，算法是至关重要的，不了解算法，就谈不上程序设计。

算法是一系列解决问题的清晰指令，不同的算法可能用不同的时间、空间或效率来完成同样的任务，一个算法的设计是否合理可以用空间复杂度与时间复杂度来衡量。其中，空间复杂度是指算法需要消耗的空间资源，而时间复杂度是指算法需要消耗的时间资源。

常用的算法主要有回溯法、递归法、递推法、迭代法及穷举搜索法等，下面通过著名的八皇后问题说明程序中如何使用算法。

八皇后问题是一个古老而著名的问题，它实质上就是使棋盘上的 8 个皇后不能在同一行、同一列或同一条斜线上，共有 92 种方法。实现八皇后问题主要用到了回溯算法，其具体实现流程如下。

(1) 从第一列开始，为皇后找到安全位置，然后跳转到下一列。

(2) 如果在第 n 列出现死胡同，判断该列是第几列，如果为第一列，则棋局失败，否则返回到上一列，再进行回溯。

(3) 如果在第 8 列上找到了安全位置，则棋局成功。

**例 3.01** 创建一个控制台应用程序，并将其命名为 EightQueen。在该程序中首先自定义一个无返回值类型的方法 QueenArithmetic，用来实现八皇后问题；该方法中有一个 int 类型的参数，用来表示皇后的数量。QueenArithmetic 方法实现代码如下。（实例位置：光盘\mr\03\sl\3.01）

```
#region 解决八皇后问题
///<summary>
///解决八皇后问题
///</summary>
///<param name="size">皇后数量</param>
static void QueenArithmetic(int size)
{
    int[] Queen = new int[size];
    int y, x, i, j, d, t = 0;
    y = 0;
    Queen[0] = -1;
    while (true)
    {
        for (x = Queen[y] + 1; x < size; x++)
        {
            for (i = 0; i < y; i++)
            {
                j = Queen[i];
                d = y - i;
                if ((j == x) || (j == x - d) || (j == x + d))
                    break;
            }
            if (i >= y)
                break;
        }
        if (x == size)
            //没有合适的位置
    }
}
//每行皇后的位置
//定义 6 个 int 类型变量
//y 用来标记结束点
//设置初始检索点
//使用 while 语句循环检索皇后位置
//调整皇后位置
//检查新皇后是否能与以前的皇后相互攻击
//不攻击
```

```

if (0 == y)
{
    Console.WriteLine("Over"); //回溯到第一行
    break; //结束
}
Queen[y] = -1; //回溯
y--;
}
else
{
    Queen[y] = x; //确定皇后的位置
    y++;
    if (y < size)
        Queen[y] = -1;
    else
    {
        Console.WriteLine("\n" + ++t + '!');
        //所有的皇后都排完之后输出
        for (i = 0; i < size; i++)
        {
            for (j = 0; j < size; j++)
                Console.Write(Queen[i] == j ? 'Q' : '*'); //输出皇后的位置
            Console.WriteLine();
        }
        y = size - 1; //回溯
    }
}
}
Console.ReadLine();
}
#endregion

```

在 Main 方法中，首先定义一个 int 类型的变量，用来记录皇后的数量，之后调用自定义方法 QueenArithmetic 将八皇后的不同排列组合显示在控制台中。Main 方法实现代码如下。

```

static void Main(string[] args)
{
    int size = 8; //皇后数
    QueenArithmetic(size);
}

```

程序运行结果如图 3.1 所示。



图 3.1 程序运行结果

## 3.2 if 语句的使用

if 语句是 C# 语言中实现选择结构最常用的方式，当 if 语句与 else 语句组合时，可以实现更灵活、复杂的选择结构。本节将对 if 语句的使用进行详细讲解。

### 3.2.1 使用 if 和 if…else 语句实现条件选择

if 语句用于根据一个布尔表达式的值选择一条语句来执行，其执行流程如图 3.2 所示。

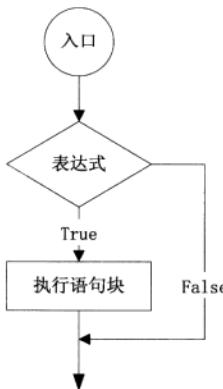


图 3.2 if 语句执行流程

if 语句的基本格式如下。

```
if(布尔表达式)
{
    【语句块】
}
```

如果使用上述格式，则只有当布尔表达式的值是 True 时，才执行语句块，否则跳过 if 语句，执行其他程序代码。

**例 3.02** 使用 if 语句判断变量 i 是否大于 927，如果返回值为 True，则输出字符串，代码如下。

```
int i = 928;                                // 声明一个 int 类型变量 i
if (i > 927)                                 // 调用 if 语句判断 i 是否大于 927
{
    Console.WriteLine("i 大于 927");
}
```

除了上述的基本格式外，if 语句还可以与 else 语句组合使用，其形式如下。

```
if(布尔表达式)
{
    【语句块】
}
else
{
    【语句块】
}
```

## 【语句块】

}

在上述格式中，【语句块】可以只有一条语句或为空语句，如果有多条语句，则可以将这些语句放在大括号（{}）中。

**例 3.03** 创建一个控制台应用程序，声明一个 int 类型的变量 i，将其初始化为 927；然后通过 if…else 语句判断变量 i 的值是否大于 927，如果大于则输出“i 大于 927”，否则执行 else 子句，输出“i 不大于 927”，代码如下。（实例位置：光盘\mr\03\sl\3.03）

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace UseIF
{
    class Program
    {
        static void Main(string[] args)
        {
            int i = 927;                                //声明一个 int 类型变量 i
            if (i > 927)                               //调用 if 语句判断 i 是否大于 927
            {
                Console.WriteLine("i 大于 927");          //如果大于 927，则输出 "i 大于 927"
            }
            else
            {
                Console.WriteLine("i 不大于 927");        //否则输出 "i 不大于 927"
            }
            Console.ReadLine();
        }
    }
}
```

程序的运行结果为“i 不大于 927”。

**技巧：**在编写程序时，读者要养成良好的编码习惯。在使用 if 语句时，通常是在 if 语句和 else 语句后使用大括号，甚至在只有一条语句时也使用大括号，并且对大括号内的语句使用缩进，这样在以后添加其他语句时会变得很容易，同时也增加了代码的可读性，有助于避免出现错误。

另外，如果有多个选择条件，则开发人员可以使用 if 语句与 else if 及 else 语句组合使用，其形式如下。

```
if(布尔表达式)
{
    【语句块】
}
else if(布尔表达式)
{
    【语句块】
}
else if(布尔表达式)
{
    【语句块】
}
```

```

...
else
{
    【语句块】
}

```

在上述格式中，【语句块】可以只有一条语句或为空语句，如果有多条语句，则可以将这些语句放在大括号（{}）中。另外，else if 语句可以根据选择条件的多少增加任意多个。

**例 3.04** 创建一个控制台应用程序，声明一个 int 类型的变量 i，用来记录输入的分数，然后使用 if…else if…else 语句判断输入的分数所处的等级，并输出提示信息，代码如下。（实例位置：光盘\mr\03\sl\3.04）

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace UseElseif
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("请输入分数：");
            int i = Convert.ToInt32(Console.ReadLine()); //记录输入的分数
            if (i < 60) //调用 if 语句判断 i 是否小于 60
            {
                Console.WriteLine("不及格，好好努力");
            }
            else if (i >= 60 && i < 70) //调用 else if 语句判断 i 是否大于 60 小于 70
            {
                Console.WriteLine("刚及格，仍需努力");
            }
            else if (i >= 70 && i < 80) //调用 else if 语句判断 i 是否大于 70 小于 80
            {
                Console.WriteLine("中，仍需努力");
            }
            else if (i >= 80 && i < 90) //调用 else if 语句判断 i 是否大于 80 小于 90
            {
                Console.WriteLine("良，继续努力");
            }
            else if (i >= 90 && i < 100) //调用 else if 语句判断 i 是否大于 90 小于 100
            {
                Console.WriteLine("非常优秀");
            }
            else //如果以上条件都不满足，则说明考了满分
            {
                Console.WriteLine("您考了满分");
            }
            Console.ReadLine();
        }
    }
}

```

程序运行结果如图 3.3 所示。

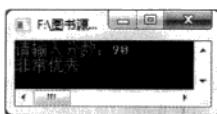


图 3.3 if…else if…else 语句的使用

### 3.2.2 if 语句的嵌套使用

当程序的条件表达式不止一个时，开发人员可以使用嵌套的 if 语句，即在 if 或 else 语句的程序块中加入另一段 if 语句或 if…else 语句，其基本格式如下。

```
if(布尔表达式)
{
    if(布尔表达式)
    {
        【语句块 1】
    }
    else
    {
        【语句块 2】
    }
}
else
{
    if(布尔表达式)
    {
        【语句块 3】
    }
    else
    {
        【语句块 4】
    }
}
```

从上面的格式中可以看出，在 if…else 语句中加入其他的 if 或 if…else 语句，实现了 if 语句的嵌套使用。下面通过一个实例演示如何实现 if 语句的嵌套。

**例 3.05** 创建一个控制台应用程序，使用嵌套的 if 语句判断用户输入的年龄，并根据年龄输出相应的字符串，代码如下。（实例位置：光盘\mr\03\s\3.05）

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace QTIF
{
    class Program
    {
        static void Main(string[] args)
        {
```

```

const int i = 18; //声明一个 int 类型的常量 i, 值为 18
const int j = 30; //声明一个 int 类型的常量 j, 值为 30
const int k = 50; //声明一个 int 类型的常量 k, 值为 50
int YouAge = 0; //声明一个 int 类型的变量 YouAge, 值为 0
Console.WriteLine("请输入您的年龄：");
YouAge = int.Parse(Console.ReadLine()); //获取用户输入的数据
if (YouAge <= i) //调用 if 语句判断输入的数据是否小于等于 18
{
    //如果输入的年龄小于等于 18 岁则输出提示信息
    Console.WriteLine("您的年龄还小，要努力奋斗哦！");
}
else
{
    if (i < YouAge && YouAge <= j) //判断是否大于 18 岁小于 30 岁
    {
        //如果输入的年龄大于 18 岁并且小于 30 岁则输出提示信息
        Console.WriteLine("您现在的阶段正是努力奋斗的黄金阶段！");
    }
    else
    {
        if (j < YouAge && YouAge <= k) //判断输入的年龄是否大于 30 岁小于等于 50 岁
        {
            //如果输入的年龄大于 30 岁而小于等于 50 岁则输出提示信息
            Console.WriteLine("您现在的阶段正是人生的黄金阶段！");
        }
        else
        {
            //输出提示信息
            Console.WriteLine("最美不过夕阳红！");
        }
    }
}
Console.ReadLine();
}

```

程序运行结果如图 3.4 所示。

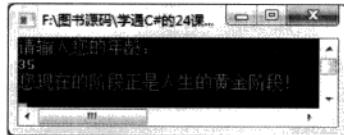


图 3.4 if 语句的嵌套使用

### 3.3 switch 语句的使用

`switch` 语句是 C# 语言中实现选择结构的另外一种方式，它非常适用于多路选择的实现。本节将对 `switch`

语句的使用进行详细讲解。

### 3.3.1 使用 switch 语句实现多分支选择

switch 语句是多分支选择语句，它根据表达式的值来使程序从多个分支中选择一个用于执行的分支，其基本格式如下。

```
switch(【表达式】)
{
    case 【常量表达式】:【语句块】
        break;
    case 【常量表达式】:【语句块】
        break;
    ...
    default:【语句块】
        break;
}
```

switch 关键字后面的括号()中是条件表达式，大括号{}中的程序代码是由数个 case 子句组成的语句块，这些语句块都是 switch 语句可能执行的语句块。如果条件表达式的值符合 case 子句指定的值，则其下的语句块就会被执行，当语句块执行完毕后，紧接着会执行 break 语句，使程序跳出 switch 语句。在 switch 语句中，【表达式】的类型必须是 sbyte、byte、short、ushort、int、uint、long、ulong、char、string 和枚举类型中的一种。【常量表达式】的值必须是与【表达式】的类型兼容的常量，并且在一个 switch 语句中，不同 case 关键字后面的【常量表达式】必须不同，如果指定了相同的【常量表达式】，则会导致编译时出错。另外，一个 switch 语句中只能有一个 default 标签。

switch 语句的执行流程如图 3.5 所示。

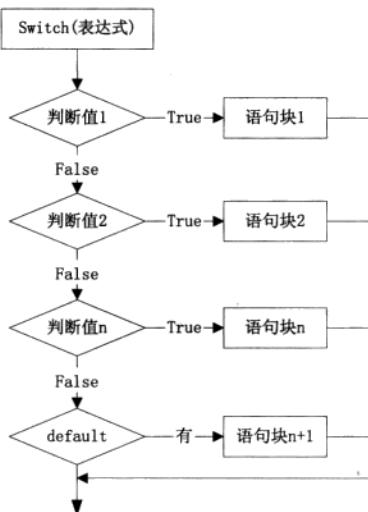


图 3.5 switch 语句执行流程

下面通过实例演示如何使用 switch 语句实现多分支选择。

**例 3.06** 创建一个控制台应用程序，声明一个 string 类型变量 myStr，并初始化为用户输入的内容，然后使用 switch 语句根据变量 myStr 选择要执行的语句，最后使用 break 语句跳出 switch 语句，代码如下。（实例位置：光盘\mr\03\s\3.06）

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace UseSwitch
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.Write("请输入内容：");
            string myStr = Console.ReadLine(); //声明一个字符串变量 myStr 并初始化
            switch (myStr) //调用 switch 语句
            {
                case "C#从基础到项目实战": //如果 myStr 的值是“C#从基础到项目实战”，执行分支 1
                    Console.WriteLine("精品图书");
                    break;
                case "C#编程词典": //如果 myStr 的值是“C#编程词典”，执行分支 2
                    Console.WriteLine("经典软件");
                    break;
                case "明日科技": //如果 myStr 的值是“明日科技”，执行分支 3
                    Console.WriteLine("公司");
                    break;
                default: //如果 myStr 的值不符合以上分支的内容，则执行 default 语句
                    Console.WriteLine(myStr);
                    break;
            }
            Console.ReadLine();
        }
    }
}

```

程序运行结果如图 3.6 所示。

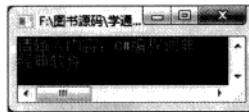


图 3.6 switch 语句的使用

### 3.3.2 if…else 语句与 switch 语句

在许多情况下，switch 语句可以简化 if…else 语句，而且执行效率更高，但是由于 switch 语句的选择表达式的值必须是整数，所以如果判断表达式为浮点数或布尔类型，就无法使用 switch 语句了。

**例 3.07** 创建一个控制台应用程序，在该程序中要求首先输入一个月份，然后根据该月份判断其所在的

季节，如果使用 if…else 语句实现以上功能，需要嵌套很多 if 语句，这样显得非常麻烦，而如果使用 switch 语句，则只需要通过 case 子句指定季节，然后判断即可。本实例使用 switch 语句来实现判断某月属于某个季节的功能，代码如下。（实例位置：光盘\mr03\sl\3.07）

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace IfToSwitch
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("请您输入一个月份："); //输出提示信息
            int myMonth = int.Parse(Console.ReadLine()); //声明一个 int 类型变量用于获取用户输入的数据
            string mySeason; //声明一个字符串变量
            switch (myMonth) //使用 switch 语句
            {
                case 12:
                case 1:
                case 2:
                    mySeason = "您输入的月份属于冬季！"; //如果输入的数据是 1、2 或 12 则执行此分支
                    break; //跳出 switch 语句
                case 3:
                case 4:
                case 5:
                    mySeason = "您输入的月份属于春季！"; //如果输入的数据是 3、4 或 5 则执行此分支
                    break; //跳出 switch 语句
                case 6:
                case 7:
                case 8:
                    mySeason = "您输入的月份属于夏季！"; //如果输入的数据是 6、7 或 8 则执行此分支
                    break; //跳出 switch 语句
                case 9:
                case 10:
                case 11:
                    mySeason = "您输入的月份属于秋季！"; //如果输入的数据是 9、10 或 11 则执行此分支
                    break; //跳出 switch 语句
                default: //如果输入的数据不满足以上分支的内容则执行 default 语句
                    mySeason = "月份输入错误！";
                    break; //跳出 switch 语句
            }
            Console.WriteLine(mySeason); //输出字符串 mySeason
            Console.ReadLine();
        }
    }
}

```

程序运行结果如图 3.7 所示。

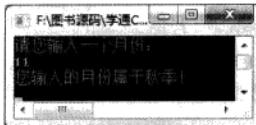


图 3.7 根据输入的月份输出相应的季节

**说明:** 上面实例如果使用 if…else 语句实现，则具体代码如下。

```
Console.WriteLine("请您输入一个月份: "); //输出提示信息
int myMonth = int.Parse(Console.ReadLine()); //声明一个 int 类型变量用于获取用户输入的数据
string mySeason; //声明一个字符串变量
if (myMonth == 1 || myMonth == 2 || myMonth == 12)
{
    mySeason = "您输入的月份属于冬季！"; //如果输入的数据是 1、2 或 12 则执行此分支
}
else if (myMonth == 3 || myMonth == 4 || myMonth == 5)
{
    mySeason = "您输入的月份属于春季！"; //如果输入的数据是 3、4 或 5 则执行此分支
}
else if (myMonth == 6 || myMonth == 7 || myMonth == 8)
{
    mySeason = "您输入的月份属于夏季！"; //如果输入的数据是 6、7 或 8 则执行此分支
}
else if (myMonth == 9 || myMonth == 10 || myMonth == 11)
{
    mySeason = "您输入的月份属于秋季！"; //如果输入的数据是 9、10 或 11 则执行此分支
}
else //如果输入的数据不满足以上分支的内容则执行 else 语句
{
    mySeason = "月份输入错误！";
}
Console.WriteLine(mySeason); //输出字符串 mySeason
Console.ReadLine();
```

## 3.4 while 和 do…while 语句的使用

当程序要反复执行某一操作时，就必须使用循环结构，如遍历二叉树、输出数组元素等。C#中的循环语句主要包括 while、do…while、for 和 foreach 语句，本节主要介绍 while 和 do…while 语句。

### 3.4.1 使用 while 语句实现代码循环

while 语句用于根据条件值执行一条语句零次或多次，当每次 while 语句中的代码执行完毕时，将重新查看是否符合条件值，若符合则再次执行相同的程序代码，否则跳出 while 语句，执行其他程序代码。while 语句的基本格式如下。

```
while(【布尔表达式】)
{
    【语句块】
}
```

while语句的执行顺序如下。

- (1) 计算【布尔表达式】的值。
- (2) 如果【布尔表达式】的值为 True，则执行【语句块】；执行完毕后重新计算【布尔表达式】的值是否为 True。如果【布尔表达式】的值为 False，则将控制转移到 while语句的结尾。

while语句的执行流程如图 3.8 所示。

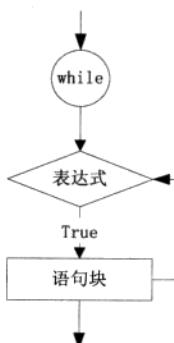


图 3.8 while语句执行流程

下面通过实例演示如何使用 while语句实现代码的循环执行。

**例 3.08** 创建一个控制台应用程序，声明一个 int类型的数组，并初始化数组，然后通过 while语句循环输出数组中的所有成员，代码如下。（实例位置：光盘\mr\03\s\3.08）

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace UseWhile
{
    class Program
    {
        static void Main(string[] args)
        {
            int[] myNum = new int[6] { 927, 112, 111, 524, 521, 2009 }; // 声明一个 int类型的数组并初始化
            int s = 0; // 声明一个 int类型的变量 s 并初始化为 0
            while (s < 6) // 调用 while语句当 s 小于 6时执行
            {
                Console.WriteLine("myNum[{0}] 的值为{1}", s, myNum[s]); // 输出数组中的值
                s++; // s 自增 1
            }
            Console.ReadLine();
        }
    }
}
  
```

程序的运行结果为：

myNum[0]的值为 927

myNum[1]的值为 112

```
myNum[2]的值为 111
myNum[3]的值为 524
myNum[4]的值为 521
myNum[5]的值为 2009
```

在 while 语句的嵌入语句块中可以使用 break 语句将控制转到 while 语句的结束点，而 continue 语句则可用于将控制直接转到下一次循环。下面通过一个实例讲解如何在 while 语句中使用 break 和 continue 语句。

**例 3.09** 创建一个控制台应用程序，声明两个 int 类型的变量 s 和 num，并分别初始化为 0 和 80。然后通过 while 语句循环输出，当 s 大于 40 时，使用 break 语句终止循环；当 s 是偶数时，使用 continue 语句将程序转到下一次循环，从而实现输出 40 以内的所有奇数，代码如下。（实例位置：光盘\mr\03\s\3.09）

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace BCWhile
{
    class Program
    {
        static void Main(string[] args)
        {
            int s = 0, num = 80; //声明两个 int 类型的变量并初始化
            while (s < num) //调用 while 语句，当 s 小于 num 时执行
            {
                s++; //s 自增 1
                if (s > 40) //使用 if 语句判断 s 是否大于 40
                {
                    break; //使用 break 语句终止循环
                }
                if ((s % 2) == 0) //调用 if 语句判断 s 是否为偶数
                {
                    continue; //使用 continue 语句将程序转到下一次循环
                }
                Console.WriteLine(s); //输出 s
            }
            Console.ReadLine();
        }
    }
}
```

程序的运行结果为 1~39 的所有奇数。

### 3.4.2 使用 do…while 语句实现至少执行一次循环

do…while 语句与 while 语句相似，但它的判断条件在循环后，这样使得程序中至少能执行一次代码块，其基本形式如下。

```
do
{
    【语句块】
}while(【布尔表达式】);
```

注意: while(【布尔表达式】)之后必须加分号(;)。

do...while语句的执行顺序如下。

(1) 程序首先执行【语句块】。

(2) 当程序到达【语句块】的结束点时, 计算【布尔表达式】的值, 如果【布尔表达式】的值是True, 则程序转到 do...while语句的开头, 重新执行【语句块】; 否则, 结束循环。

do...while语句的执行流程如图3.9所示。

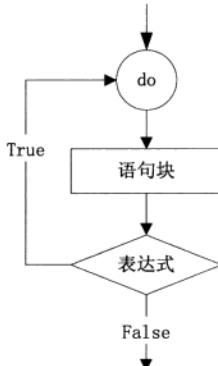


图3.9 do...while语句执行流程

**例3.10** 创建一个控制台应用程序, 声明一个bool类型的变量term, 并初始化为false; 再声明一个int类型的数组, 并初始化数组; 然后调用do...while语句, 通过for语句循环输出数组中的值, for语句将在3.5节中讲解, 代码如下。(实例位置: 光盘\mr\03\s\3.10)

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace UseDoWhile
{
    class Program
    {
        static void Main(string[] args)
        {
            bool term = false; //声明一个bool型变量term并初始化为false
            int[] myArray = new int[5] { 0, 1, 2, 3, 4 }; //声明一个int类型数组并初始化
            do //调用do...while语句
            {
                for (int i = 0; i < myArray.Length; i++) //调用for语句输出数组中的所有数据
                {
                    Console.WriteLine(myArray[i]); //输出数组中数据
                }
            } while (term); //设置do...while语句的条件
            Console.ReadLine();
        }
    }
}
  
```

程序的运行结果为：

```
0
1
2
3
4
```

从上述代码中不难看出，bool 类型变量 term 被初始化为 false，但是 do…while 语句依然执行了一次 for 循环，将数组中的值输出。由此可以说明，do…while 语句至少要执行代码一次，无论最后的条件是 true 还是 false。

### 3.4.3 while 和 do…while 语句的区别

while 语句和 do…while 语句都用来控制代码的循环，但 while 语句适合于先条件判断，再执行循环结构的场合；而 do…while 语句则适合于先执行循环结构，再进行条件判断的场合。具体来说，使用 while 语句时，如果条件不成立，则循环结构一次都不会执行，而如果使用 do…while 语句时，即使条件不成立，程序也至少会执行一次循环结构。

## 3.5 for 和 foreach 语句的使用

在 3.4 节中介绍了 while 和 do…while 语句，本节主要介绍 for 和 foreach 语句的使用。for 语句循环执行一个语句或语句块；foreach 语句为数组或对象集合中的每个元素重复执行一个嵌入语句组，下面对这两种语句进行详细介绍。

### 3.5.1 使用 for 语句实现代码循环

for 语句用于计算一个初始化序列，当某个条件为真时，重复执行嵌套语句并计算一个迭代表达式序列，如果为假，则退出 for 循环。for 语句的基本格式如下。

```
for(【初始化表达式】;【条件表达式】;【迭代表达式】)
{
    【语句块】
}
```

【初始化表达式】由一个局部变量声明或一个逗号分隔的表达式列表组成，用【初始化表达式】声明的局部变量的作用域从变量的声明开始，一直到嵌入语句的结尾；【条件表达式】必须是一个布尔表达式；【迭代表达式】必须包含一个用逗号分隔的表达式列表。

for 语句的执行顺序如下。

(1) 如果有【初始化表达式】，则按变量初始值设定项或语句表达式的书写顺序指定它们，此步骤只执行一次。

(2) 如果存在【条件表达式】，则计算它。

(3) 如果不存在【条件表达式】，则程序将转移到嵌入语句，如果程序到达了嵌入语句的结束点，则按顺序计算迭代表达式，然后从上一步中 for 条件的计算开始执行另一次循环。

for 语句的执行流程如图 3.10 所示。

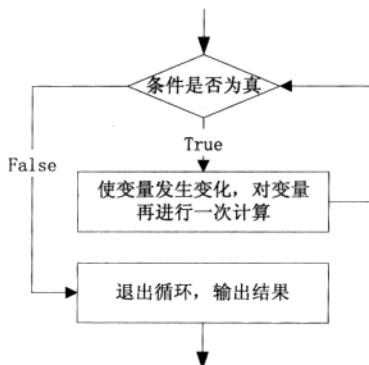


图 3.10 for 语句执行流程

**例 3.11** 创建一个控制台应用程序，首先声明一个 int 类型的数组，然后向数组中添加 10 个值，最后使用 for 循环语句遍历数组，并将数组中的值输出，代码如下。（实例位置：光盘\mr\03\sl\3.11）

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace UseFor
{
    class Program
    {
        static void Main(string[] args)
        {
            int[] myint = new int[10]; // 声明一个具有 10 个元素的整型数组
            myint[0] = 0; // 向数组中添加第 1 个元素
            myint[1] = 1; // 向数组中添加第 2 个元素
            myint[2] = 2; // 向数组中添加第 3 个元素
            myint[3] = 3; // 向数组中添加第 4 个元素
            myint[4] = 4; // 向数组中添加第 5 个元素
            myint[5] = 5; // 向数组中添加第 6 个元素
            myint[6] = 6; // 向数组中添加第 7 个元素
            myint[7] = 7; // 向数组中添加第 8 个元素
            myint[8] = 8; // 向数组中添加第 9 个元素
            myint[9] = 9; // 向数组中添加第 10 个元素
            for (int i = 0; i < myint.Length; i++) // 调用 for 循环语句
            {
                Console.WriteLine("myint[{0}] 的值是: {1}", i, myint[i]);
            }
            Console.ReadLine();
        }
    }
}
  
```

程序运行结果如图 3.11 所示。



图 3.11 使用 for 语句输出数组中的值

### 3.5.2 使用 foreach 语句遍历数据集合

foreach 语句用于枚举一个集合的元素，并对该集合中的每个元素执行一次嵌入语句，但 foreach 语句不应用于更改集合内容，以避免产生不可预知的错误。foreach 语句的基本格式如下。

```

foreach(【类型】 【迭代变量名】 in 【集合类型表达式】)
{
    【语句块】
}
  
```

其中，【类型】和【迭代变量名】用于声明迭代变量，迭代变量相当于一个范围覆盖整个语句块的局部变量，在 foreach 语句执行期间，迭代变量表示当前正在为其执行迭代的集合元素；【集合类型表达式】必须有一个从该集合的元素类型到迭代变量的类型的显示转换，如果【集合类型表达式】的值为 null，则会出现异常。

foreach 语句的执行流程如图 3.12 所示。

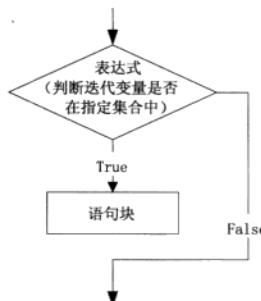


图 3.12 foreach 语句执行流程

**例 3.12** 创建一个控制台应用程序，同时创建一个 ArrayList 集合，并向该集合中添加值，然后通过使用 foreach 语句遍历整个集合，并输出集合中的值，代码如下。（实例位置：光盘\mr\03\sl\3.12）

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Collections;
namespace UseForeach
{
    class Program
    {
        static void Main()
        {
            ArrayList list = new ArrayList();
            list.Add("www.111.com");
            list.Add("www.222.com");
            list.Add("www.333.com");
            list.Add("www.444.com");
            list.Add("www.555.com");
            foreach (string item in list)
            {
                Console.WriteLine(item);
            }
        }
    }
}
  
```

```

static void Main(string[] args)
{
    ArrayList lists = new ArrayList();
    lists.Add("明日科技");
    lists.Add("C#编程词典");
    lists.Add("C#从基础到项目实战");
    Console.WriteLine("集合列表: ");
    foreach (string strName in lists)
    {
        Console.WriteLine(strName);
    }
    Console.ReadLine();
}
}

```

程序运行结果如图 3.13 所示。

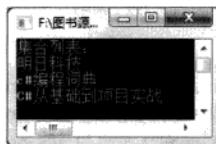


图 3.13 使用 foreach 语句遍历集合中的值

## 3.6 循环结构中的跳转语句

跳转语句主要用于无条件地转移控制，它会将控制转到某个位置，这个位置就成为跳转语句的目标。如果跳转语句出现在一个语句块内，而跳转语句的目标却在该语句块之外，则称该跳转语句退出该语句块。跳转语句主要包括 break、continue、goto 和 return 语句，本节将对这几种跳转语句分别进行介绍。

### 3.6.1 使用 break 语句跳出循环

break 语句只能应用在 switch、while、do…while、for 或 foreach 语句中，而且当多条 switch、while、do…while、for 或 foreach 语句互相嵌套时，break 语句只应用于最里层的语句。如果要穿越多个嵌套层，则必须使用 goto 语句。break 语句的执行流程如图 3.14 所示。

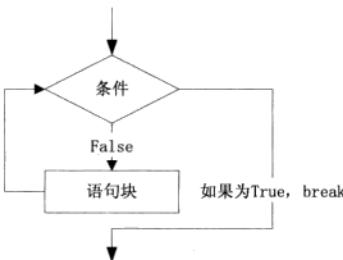


图 3.14 break 语句执行流程

下面主要举例说明 break 语句在 switch 语句和 for 语句中的使用。

### 1. break 语句在 switch 语句中的应用

**例 3.13** 创建一个控制台应用程序，声明一个 int 类型的变量 i，用于获取当前日期的返回值，然后通过使用 switch 语句根据变量 i 输出当前日期是星期几，代码如下。（实例位置：光盘\mr\03\sl\3.13）

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace BreakToSwitch
{
    class Program
    {
        static void Main(string[] args)
        {
            int i = Convert.ToInt32(DateTime.Today.DayOfWeek); // 获取当前日期的数值
            switch (i) // 调用 switch 语句
            {
                case 1: Console.WriteLine("今天是星期一"); break; // 如果 i 是 1，则输出今天是星期一
                case 2: Console.WriteLine("今天是星期二"); break; // 如果 i 是 2，则输出今天是星期二
                case 3: Console.WriteLine("今天是星期三"); break; // 如果 i 是 3，则输出今天是星期三
                case 4: Console.WriteLine("今天是星期四"); break; // 如果 i 是 4，则输出今天是星期四
                case 5: Console.WriteLine("今天是星期五"); break; // 如果 i 是 5，则输出今天是星期五
                case 6: Console.WriteLine("今天是星期六"); break; // 如果 i 是 6，则输出今天是星期六
                case 7: Console.WriteLine("今天是星期日"); break; // 如果 i 是 7，则输出今天是星期日
            }
            Console.ReadLine();
        }
    }
}
```

程序的运行结果为“今天是星期五”。

### 2. break 语句在 for 语句中的应用

**例 3.14** 创建一个控制台应用程序，使用两个 for 语句做嵌套循环，在内层的 for 语句中使用 break 语句实现当 int 类型变量 j 等于 12 时，跳出内循环，代码如下。（实例位置：光盘\mr\03\sl\3.14）

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace BreakToFor
{
    class Program
    {
        static void Main(string[] args)
        {
            for (int i = 0; i < 4; i++) // 调用 for 语句
            {
                Console.Write("\n 第{0}次循环： ", i); // 输出提示是第几次循环
            }
        }
    }
}
```

```

for (int j = 0; j < 200; j++)
{
    if (j == 12)
        break;
    Console.WriteLine(j + " ");
}
Console.ReadLine();
}
}
}

```

程序的运行结果为：

第0次循环：0 1 2 3 4 5 6 7 8 9 10 11  
 第1次循环：0 1 2 3 4 5 6 7 8 9 10 11  
 第2次循环：0 1 2 3 4 5 6 7 8 9 10 11  
 第3次循环：0 1 2 3 4 5 6 7 8 9 10 11

说明：从程序的运行结果中可以看出，使用 break 语句只终止了内层循环，而并没有影响到外部的循环，所以程序依然执行了 4 次循环。

### 3.6.2 使用 continue 语句继续程序的执行

continue 语句只能应用于 while、do…while、for 或 foreach 语句中，它用来忽略循环语句块内位于它后面的代码而直接开始一次新的循环。当多个 while、do…while、for 或 foreach 语句嵌套时，continue 语句只能使直接包含它的循环语句开始一次新的循环。continue 语句的执行流程如图 3.15 所示。

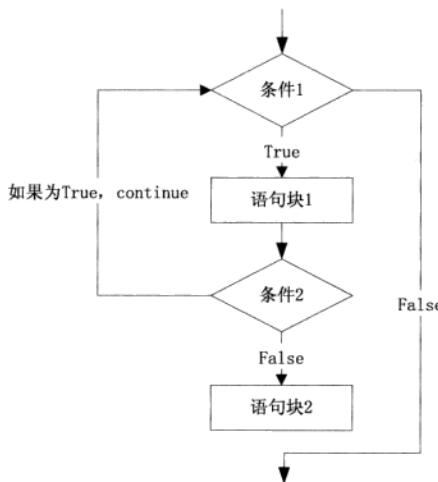


图 3.15 continue 语句执行流程

**例 3.15** 创建一个控制台应用程序，使用两个 for 语句做嵌套循环，在内层的 for 语句中使用 continue 语句，实现当 int 类型变量 j 为偶数时，重新开始内层的 for 循环，使其只输出 0~20 之间的所有奇数，代

码如下。（实例位置：光盘\mr\03\sl\3.15）

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace UseContinue
{
    class Program
    {
        static void Main(string[] args)
        {
            for (int i = 0; i < 4; i++) //调用 for 循环
            {
                Console.WriteLine("\n第{0}次循环：", i); //输出提示第几次循环
                for (int j = 0; j < 20; j++) //调用 for 循环
                {
                    if (j % 2 == 0) //调用 if 语句判断 j 是否是偶数
                        continue; //如果是偶数则使用 continue 语句继续下一循环
                    Console.Write(j + " "); //输出 j
                }
                Console.WriteLine();
            }
            Console.ReadLine();
        }
    }
}

```

程序的运行结果为：

第 0 次循环： 1 3 5 7 9 11 13 15 17 19

第 1 次循环： 1 3 5 7 9 11 13 15 17 19

第 2 次循环： 1 3 5 7 9 11 13 15 17 19

第 3 次循环： 1 3 5 7 9 11 13 15 17 19

**说明：**从程序的运行结果可以看出，当 int 类型的变量 j 为偶数时使用 continue 语句，忽略它后面的代码，而重新执行内层的 for 循环，输出 0~20 之间的奇数，在这期间，continue 语句并没有影响外部的 for 循环，所以程序依然执行了 4 次循环。

### 3.6.3 使用 goto 语句实现程序跳转

goto 语句用于将控制转移到由标签标记的语句，它可以被应用于 switch 语句中的 case 标签、default 标签以及标记语句所声明的标签。goto 语句的 3 种格式如下。

```

goto 【标签】
goto case 【参数表达式】
goto default

```

goto【标签】语句的目标是具有给定标签的标记语句；goto case 语句的目标是它所在的 switch 语句中的某个语句列表，此列表包含一个具有给定常数值的 case 标签；goto default 语句的目标是它所在的 switch 语句中的 default 标签。

goto 语句的执行流程如图 3.16 所示。

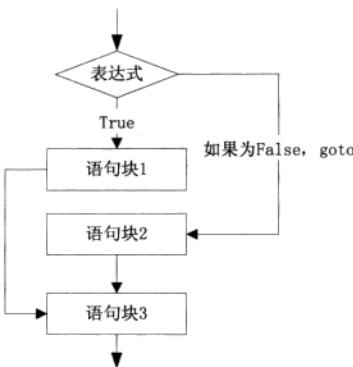


图 3.16 goto 语句执行流程

**例 3.16** 创建一个控制台应用程序，通过 goto 语句实现将程序跳转到指定语句的功能，代码如下。（实例位置：光盘\mr\03\sl\3.16）

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace UseGoto
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("请输入要查找的文字："); //输出提示信息
            string inputstr = Console.ReadLine(); //获取输入值
            string[] mystr = new string[3]; //创建一个字符串数组
            mystr[0] = "明日科技"; //向数组中添加第 1 个元素
            mystr[1] = "C#编程词典"; //向数组中添加第 2 个元素
            mystr[2] = "C#从基础到项目实战"; //向数组中添加第 3 个元素
            for (int i = 0; i < mystr.Length; i++) //调用 for 循环语句
            {
                if (mystr[i].Equals(inputstr)) //通过 if 语句判断是否存在输入的字符串
                {
                    goto Found; //调用 goto 语句跳转到 Found
                }
            }
            Console.WriteLine("您查找的{0}不存在！", inputstr); //输出信息
            goto Finish; //调用 goto 语句跳转到 Finish
        }

        Found:
        Console.WriteLine("您查找的{0}存在！", inputstr); //输出信息，提示存在输入的字符串

        Finish:
        Console.WriteLine("查找完毕！");
        Console.ReadLine();
    }
}
  
```

程序运行结果如图 3.17 所示。

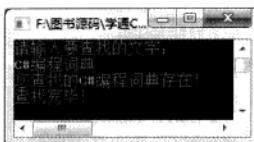


图 3.17 goto 语句的使用

**◆ 注意：** 虽然 goto 语句有一定的使用价值，但是目前对它的使用存在争议。有人建议避免使用它，有人建议把它用来作为排除错误的基本工具，各种观点截然不同。所以要小心使用，同时一定要确保程序是可维护的。

### 3.6.4 使用 return 语句使程序返回

return 语句用于退出类的方法，把控制返回方法的调用者，如果方法有返回类型，则 return 语句必须返回该类型的值；如果方法没有返回类型，则应使用没有表达式的 return 语句。return 语句的执行流程如图 3.18 所示。

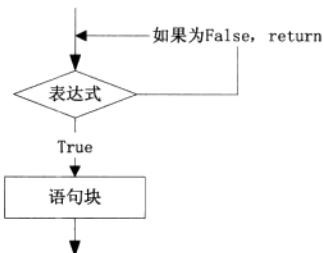


图 3.18 return 语句执行流程

**例 3.17** 创建一个控制台应用程序，建立一个返回类型为 string 类型的方法，利用 return 语句，返回一个 string 类型的值，然后在 Main 方法中调用这个自定义的方法，并输出该方法的返回值，代码如下。（实例位置：光盘\mr\03\s\3.17）

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace UseReturn
{
    class Program
    {
        static string MyStr(string str)           //创建一个 string 类型方法
        {
            string OutStr;                      //声明一个字符串变量
            OutStr = "您输入的数据是：" + str;     //为字符串变量赋值
            return OutStr;                      //使用 return 语句返回字符串变量
        }
        static void Main(string[] args)
    }
}
  
```

```

    {
        Console.WriteLine("请您输入内容:");           //输出提示信息
        string inputstr = Console.ReadLine();         //获取输入的数据
        Console.WriteLine(MyStr(inputstr));           //调用 MyStr 方法并将结果显示出来
        Console.ReadLine();
    }
}

```

程序运行结果如图 3.19 所示。

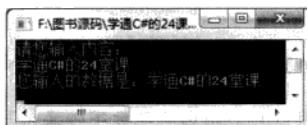


图 3.19 return 语句的应用

## 3.7 照猫画虎——基本功训练

### 3.7.1 基本功训练 1——循环向控制台中输出内容

**视频讲解：**光盘\mr\03\lx\循环向控制台中输出内容.exe

**实例位置：**光盘\mr\03\zmhh\01

循环语句在程序开发中经常被用到，可以使用 for 循环遍历数组，也可以使用 foreach 语句枚举出集合中的每一个元素。本实例使用 for 语句无限循环，在循环中每间隔 1 秒会向控制台中输出系统时间，然后再清空控制台信息，以实现动态显示系统时间的效果，实例运行效果如图 3.20 所示。

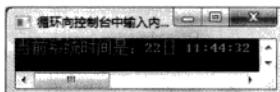


图 3.20 循环向控制台中输出内容

实现过程如下。

- (1) 打开 Visual Studio 2008 开发环境，新建一个控制台应用程序，并将其命名为 InputMessage。
- (2) 在 Main 方法中，首先通过 Console 类的 WindowWidth 属性和 WindowHeight 属性设置控制台窗口的宽度和高度，然后使用一个死循环的 for 语句，并调用 Console.WriteLine 方法循环输出当前的系统时间，程序主要代码如下。

```

class program
{
    static void Main()
    {
        System.Console.Title = "循环向控制台中输入内容";           //定义控制台标题
        System.Console.WindowWidth = 30;                                //设置控制台窗体宽度
        System.Console.WindowHeight = 2;                                 //设置控制台窗体高度
        for (; ; )                                                       //开始无限循环
        {
    }
}

```

```

        System.Console.WriteLine("当前系统时间是: {0}",  

            System.DateTime.Now.ToString("dd 日 hh:mm:ss"));  

        System.Threading.Thread.Sleep(1000);  

        System.Console.Clear();  

    }  

}  

}

```

**照猫画虎：**创建一个控制台应用程序，循环向控制台输出 0~20 之间的偶数。提示：判断某个数为偶数可以使用“%”运算符。（20 分）（实例位置：光盘\mr\03\zmhh\01\_zmhh）

### 3.7.2 基本功训练 2——使用 switch 语句实现数字转换大写

■ 视频讲解：光盘\mr\03\lx\使用 switch 语句实现数字转换大写.exe

■ 实例位置：光盘\mr\03\zmhh\02

在各种销售发票中，通常都会记录金额的小写和大写两种形式，这就存在着大小写金额之间相互转换的问题。本实例使用 switch 多分支选择语句实现将数字的小写形式转换成对应的大写形式，运行效果如图 3.21 所示。

实现过程如下。

(1) 创建一个控制台应用程序，并将其命名为 ConvertLowerToUpper。

(2) 在 Main 方法中，首先使用一个 while 死循环语句，并调用 Console.ReadLine 方法实现循环从控制台中读取数字，然后使用 switch 语句判断从控制台中读取的数字符合哪一个 case 分支，最后调用 Console.WriteLine 方法输出对应的大写形式，程序主要代码如下。

```

static void Main(string[] args)  

{
    while (true)                                //可以循环输入  

    {  

        Console.Write("请输入 0-9 中的任意一个数字: ");  

        int intNum = Convert.ToInt32(Console.ReadLine());  

        switch (intNum)                            //读取输入的数字  

        {  

            case 0:                                //判断输入的数字  

                Console.WriteLine("它的大写是 “零” ");  

                break;                               //若输入的是 0  

            case 1:  

                Console.WriteLine("它的大写是 “壹” ");  

                break;                               //输出大写 “零”  

            case 2:  

                Console.WriteLine("它的大写是 “贰” ");  

                break;                               //若输入的是 1  

            case 3:  

                Console.WriteLine("它的大写是 “叁” ");  

                break;                               //则输出大写 “壹”  

            case 4:  

                Console.WriteLine("它的大写是 “肆” ");  

                break;                               //若输入的是 2  

        }
    }
}

```



图 3.21 使用 switch 语句实现数字转换大写

```
        break;
    case 5:
        Console.WriteLine("它的大写是“伍”");
        break;
    case 6:
        Console.WriteLine("它的大写是“陆”");
        break;
    case 7:
        Console.WriteLine("它的大写是“柒”");
        break;
    case 8:
        Console.WriteLine("它的大写是“捌”");
        break;
    case 9:
        Console.WriteLine("它的大写是“玖”");
        break;
    default:
        Console.WriteLine("您输入的不是 0-9 中的数字");
        break;
    }
}
```

注意：在 switch 多路选择语句中，多个 case 标签可以使用一个 break 关键字，但是在这种情况下，只有最后一个 case 标签中可以带有语句块内容，而前面的 case 标签不可以带有语句块内容。

**熊猫画虎：**参照上面的实例，实现将数字的大写形式转换为对应的小写形式。提示：可使用 switch 语句判断从控制台中读取的数字大写形式符合哪一个 case 分支，最后调用 Console.WriteLine 方法输出对应的小写形式。（20 分）（实例位置：光盘\mr\03\zmhh\02\_zmhh）

### 3.7.3 基本功训练 3——鸡尾酒排序算法的实现

 视频讲解：光盘\mr\03\lx\鸡尾酒排序算法的实现.exe

 实例位置：光盘\mr\03\zmhh\03

鸡尾酒排序，也称双向冒泡排序、搅拌排序或涟漪排序，当使用鸡尾酒排序时，首先对要排序的内容从低到高排序，然后再从高到低排序。本实例使用鸡尾酒排序法对数据进行排序，原始数字的顺序为“3, 9, 27, 6, 18, 12, 21, 15”，运行本实例，这些数据排序后的效果如图 3.22 所示。

实现过程如下。

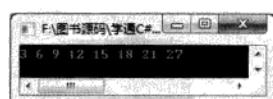


图 3.22 鸡尾酒排序算法的实现

(1) 创建一个控制台应用程序，并将其命名为 CockSort。在该程序中首先自定义一个静态的 int 类型的数组，用来表示要排序的数组，代码如下。

```
static int[] intArray; //定义要排序的数组
```

(2) 自定义一个静态的无返回值类型的 Change 方法，用来对两个数的位置进行互换。在该方法中有两个引用参数，分别用来表示要互换位置的两个数。Change 方法实现代码如下。

///两个数互换位置

```
//定义要排序的数组
```

```
///<param name="
```

```
///<param name="right">第二个数</param>
```

```

static void Change(ref int left, ref int right)
{
    int temp;
    temp = left; //临时变量
    left = right; //记录第一个数的值
    right = temp; //使第一个数的值等于第二个数的值
    //使第二个数的值等于临时变量记录的值
}

```

(3) 自定义一个静态的无返回值类型的 CockSorts 方法, 用来实现鸡尾酒排序算法。在该方法中有一个 int 类型的一维数组, 用来表示要判断的数组。CockSorts 方法实现代码如下。

```

///鸡尾酒排序算法
///<param name="intArray">要排序的数组</param>
static void CockSorts(int[] intArray)
{
    int low, up, index; //定义变量
    low = 0; //数组的开始索引
    up = intArray.Length - 1; //数组的结束索引
    index = low; //临时变量
    while (up > low) //判断数组中是否有多个元素
    {
        for (int i = low; i < up; i++) //从上向下扫描
        {
            if (intArray[i] > intArray[i + 1]) //比较前后两个数的大小
            {
                Change(ref intArray[i], ref intArray[i + 1]); //变换两个数的位置
                index = i; //记录当前索引
            }
        }
        up = index; //记录最后一个交换的位置
        for (int i = up; i > low; i--) //从最后一个交换位置处从下向上扫描
        {
            if (intArray[i] < intArray[i - 1]) //比较前后两个数的大小
            {
                Change(ref intArray[i], ref intArray[i - 1]); //变换两个数的位置
                index = i; //记录当前索引
            }
        }
        low = index; //记录最后一个交换的位置
    }
}

```

(4) 自定义一个静态的无返回值类型的 Sort 方法, 用来调用自定义方法 CockSorts 对指定的数组进行鸡尾酒排序。在该方法中有一个 int 类型的一维数组, 用来表示要排序的一维数组。Sort 方法实现代码如下。

```

///对指定数组使用鸡尾酒排序算法进行排序
///<param name="intArr">要排序的一维数组</param>
static void Sort(int[] intArr)
{
    intArray = intArr; //为数组赋值
    CockSorts(intArray); //使用鸡尾酒算法进行排序
}

```

(5) 在 Main 方法中, 首先定义一个 int 类型的一维数组并赋值, 然后调用自定义方法 Sort 对定义的一

维数组进行鸡尾酒排序，最后循环遍历该数组并输出数组中的每个元素。Main方法实现代码如下。

```
static void Main(string[] args)
{
    int[] arr = new int[] { 3, 9, 27, 6, 18, 12, 21, 15 };
    Sort(arr);
    for (int i = 0; i < arr.Length; i++)
    {
        Console.WriteLine(arr[i] + " ");
    }
    Console.ReadLine();
}
```

**熊猫画虎：**使用单项冒泡法排序“3, 9, 27, 6, 18, 12, 21, 15”这组数据。提示：单项冒泡法，即通常所说的冒泡排序法，它的实现过程很简单，首先将第一个记录的关键字和第二个记录的关键字进行比较，若为逆序，则将两个记录交换，然后比较第二个记录和第三个记录的关键字，依次类推，直至第n-1个记录和第n个记录的关键字进行过比较为止，上述过程称为第一趟冒泡排序，执行n-1次上述过程后，排序即可完成。（20分）（实例位置：光盘\mr\03\zmhh\03\_zmhh）

### 3.7.4 基本功训练4——判断用户登录身份

视频讲解：光盘\mr\03\lx\判断用户登录身份.exe

实例位置：光盘\mr\03\zmhh\04

在应用程序运行时，经常需要判断用户登录身份，根据用户登录身份的不同，给予相应的操作权限。本实例使用条件语句判断用户登录身份，实例运行效果如图3.23所示。

实现过程如下。

(1) 打开Visual Studio 2008开发环境，新建一个Windows窗体应用程序，并将其命名为Login。

(2) 更改默认窗体Form1的Name属性为Frm\_Main，向窗体中添加一个Combobox下拉列表控件，用于选择登录的用户；添加一个Button按钮，用于判断用户登录信息并使用消息框显示。

(3) 程序主要代码如下。

```
private void btn_login_Click(object sender, EventArgs e)
{
    if (cbox_select.SelectedItem.ToString() == "admin")           //判断用户登录信息
    {
        MessageBox.Show("管理员登录", "提示！");
    }
    else
    {
        MessageBox.Show("普通用户登录", "提示！");
    }
}
```



图3.23 判断用户登录身份

**技巧：**在程序运行时，如果不希望用户更改Combobox下拉列表中选中的内容，可以设置DropDownStyle属性为DropDownList。

**照猫画虎：**以上面的实例为基础，增加一种“开发人员”用户。提示：可以使用 if…else if…else 语句来实现。(20 分)(实例位置：光盘\mr\03\zmhh\04\_zmhh)

### 3.7.5 基本功训练 5——小明去学校和医院分别要走哪条路

■ 视频讲解：光盘\mr\03\lx\小明去学校和医院分别要走哪条路.exe

■ 实例位置：光盘\mr\03\zmhh\05

人类的大脑就好比是一部逻辑机器，每天都要处理大量的数据。那么，如何使编写的程序也同样具有逻辑处理能力呢？现在我们可以使用 if 语句，if 语句可以根据逻辑判断执行相应语句块，根据不同的条件做出不同的行为。例如，本实例用来判断小明去学校和医院分别要走哪条路，运行效果如图 3.24 所示。



图 3.24 小明去学校和医院分别要走哪条路

实现过程如下。

- (1) 打开 Visual Studio 2008 开发环境，新建一个 Windows 窗体应用程序，并将其命名为 WhichWay。
- (2) 更改默认窗体 Form1 的 Name 属性为 Frm\_Main，向窗体中添加一个 Button 按钮，用于计算“小明”应走哪条路；添加两个 RadioButton 单选按钮，分别用来指示小明要去学校还是医院。
- (3) 程序主要代码如下。

```
private void btn_go_Click(object sender, EventArgs e)
{
    if (rbtn_school.Checked)
        //判断小明去学校还是去医院
        MessageBox.Show("向左走", "提示！");
    else
        //如果去学校则向左走
        MessageBox.Show("向右走", "提示！");
}
```

技巧：在使用 if 语句进行判断并赋值时，可以适当地使用三元运算符（又称条件运算符）代替，适当地使用三元运算符会使代码更加清晰明了，它首先会判断布尔值或布尔表达式，并根据布尔结果返回表达式 1 的值或表达式 2 的值。

**照猫画虎：**以上面的实例为基础，若选择“去医院”，则还需进一步判断，是去“第一医院”，还是“第二医院”。若选择去“第一医院”，则提示“向前走”；否则，提示“向后走”。提示：可以使用 if…else 嵌套语句来实现。(20 分)(实例位置：光盘\mr\03\zmhh\05\_zmhh)

照猫画虎栏目分数统计：

照猫画虎题目	1	2	3	4	5	总分数	
分数							

## 3.8 情景应用——拓展与实践

### 3.8.1 情景应用 1——递归算法的经典面试题

视频讲解：光盘\mr\03\lx\递归算法的经典面试题.exe

实例位置：光盘\mr\03\qjyy\01

在编程的世界中充斥着大量的数据，如何方便有效地处理数据呢？这时算法就显得尤为重要了，本实例介绍了一个经典的面试题，有一组数 1、1、2、3、5、8、13、21、34…，要求用递归算法算出这组数的第 30 个数是多少？实例运行效果如图 3.25 所示。

实现过程如下。

(1) 打开 Visual Studio 2008 开发环境，新建一个 Windows 窗体应用程序，并将其命名为 Arithmetic。

(2) 更改默认窗体 Form1 的 Name 属性为 Frm\_Main，向窗体中添加一个 TextBox 控件，用于用户输入信息；添加一个 Button 按钮，用于计算结果；添加一个 Label 控件，用于显示计算结果。

(3) 程序主要代码如下。

```
private void btn_Get_Click(object sender, EventArgs e)
{
    int P_int_temp; //定义整型变量
    if (int.TryParse(txt_value.Text, out P_int_temp)) //为变量赋值
    {
        lb_result.Text = "计算结果为：" + Get(P_int_temp).ToString(); //输出计算结果
    }
    else
    {
        MessageBox.Show("请输入正确的数值！", "提示！"); //提示输入正确数值
    }
}
///递归算法
///<param name="i">参与计算的数值</param>
///<returns>计算结果</returns>
int Get(int i)
{
    if (i < 0) //判断数值是否小于 0
        return 0; //返回数值 0
    else if (i >= 0 && i <= 2) //判断位数是否大于等于 0 并且小于等于 2
        return 1; //返回数值 1
    else //如果不满足上述条件则执行下面语句
        return Get(i - 1) + Get(i - 2); //返回指定位数前两位数的和
}
```

DIY：求 10 的阶乘。提示：10 的阶乘，即  $1*2*3*...*10$  的值，可以采用递归的方法实现计算 n 的阶乘。  
(20 分)(实例位置：光盘\mr\03\qjyy\01\_diy)



图 3.25 递归算法的经典面试题

### 3.8.2 情景应用 2——使用流程控制语句报销业务花销

视频讲解：光盘\mr\03\lx\使用流程控制语句报销业务花销.exe

实例位置：光盘\mr\03\qjyy\02

流程控制语句一般用于进行逻辑判断后执行相应操作。

在下面实例中主要用来判断支出是否为业务花销，如果是业务花销，则弹出“正常报销！”提示框；如果不是业务花销，则弹出“不符合规定报销！”提示框。实例运行效果如图 3.26 所示。

实现过程如下。

(1) 打开 Visual Studio 2008 开发环境，新建一个 Windows 窗体应用程序，并将其命名为 IfThenElse。

(2) 更改默认窗体 Form1 的 Name 属性为 Frm\_Main，向窗体中添加两个 RadioButton 单选按钮，两个单选按钮的 Name 属性分别为 rbt\_true 和 rbt\_false，这两个单选按钮用于判断报销是否为业务花销；添加一个 Button 按钮，用于计算业务花销。

(3) 程序主要代码如下。

```
private void button1_Click(object sender, EventArgs e)
{
    if (rbtn_true.Checked)
    {
        MessageBox.Show("正常报销！");
    }
    else
    {
        MessageBox.Show("不符合规定报销！");
    }
}
```

//判断报销是否为业务花销  
//正常报销  
//不符合规定报销

说明：流程控制语句一般用来控制程序执行逻辑，所以熟练地使用流程控制语句是程序开发的基础，if 语句后面小括号内的布尔类型值或布尔类型表达式用于逻辑判断，根据判断的结果选择执行不同的语句块。

**DIY：** 创建一个控制台应用程序，使用 if…else 语句判断付款方式。提示：在 Main 方法中定义 3 种付款方式代码（A—现金、B—信用卡、C—普通银行卡），然后使用 Console.ReadLine 方法从控制台中读取一种付款方式代码，最后使用嵌套 if…else 语句判断输入的付款方式，并输出相应的汉字提示信息。（20 分）  
(实例位置：光盘\mr\03\qjyy\02\_diy)

### 3.8.3 情景应用 3——使用 switch 语句更改窗体颜色

视频讲解：光盘\mr\03\lx\使用 switch 语句更改窗体颜色.exe

实例位置：光盘\mr\03\qjyy\03

switch 语句是多分支选择语句，它根据表达式的值来使程序从多个分支中选择一个用于执行的分支。本实例使用 switch 语句来判断在下拉列表控件中所选择的颜色，然后程序根据所选择的颜色来设置窗体颜色。

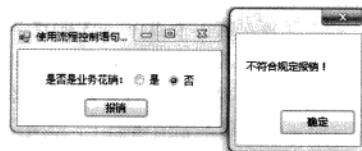


图 3.26 使用流程控制语句报销业务花销

实例运行效果如图 3.27 所示。

实现过程如下。

(1) 打开 Visual Studio 2008 开发环境，新建一个 Windows 窗体应用程序，并将其命名为 SelectColor。

(2) 更改默认窗体 Form1 的 Name 属性为 Frm\_Main，向窗体中添加一个 Combobox 下拉列表控件，用于选择窗体的颜色。

(3) 程序主要代码如下。

```
private void cbox_select_SelectedIndexChanged(object sender, EventArgs e)
{
    switch (cbox_select.SelectedIndex)
    {
        case 0:
            this.BackColor = Color.Orange; //窗体设置为橙色
            break;
        case 1:
            this.BackColor = Color.Green; //窗体设置为绿色
            break;
        case 2:
            this.BackColor = Color.Blue; //窗体设置为蓝色
            break;
    }
}
```

**DIY：**按照考试成绩的等级（例如，将考试成绩分为 A、B、C、D 4 个等级），要求在窗体上显示出对应的百分制分数段。提示：可以使用 switch 语句来判断考试等级，然后使用 Label 控件显示出对应分段。

(20 分) (实例位置：光盘\mr\03\qjyy\03\_diy)

### 3.8.4 情景应用 4——使用 goto 语句在数组中搜索指定图书

■**视频讲解：**光盘\mr\03\lx\使用 goto 语句在数组中搜索指定图书.exe

■**实例位置：**光盘\mr\03\qjyy\04

本实例将图书信息存储到字符串数组中，若要搜索指定的图书，通常可以使用 System.Array 的 IndexOf 静态方法搜索指定的图书，或者使用 for 循环遍历数组查找指定图书。但本实例使用了 goto 语句和一个计数器实现简单的循环，从字符串数组中搜索图书，实例运行效果如图 3.28 所示。

实现过程如下。

(1) 打开 Visual Studio 2008 开发环境，新建一个 Windows 窗体应用程序，并将其命名为 Goto。

(2) 更改默认窗体 Form1 的 Name 属性为 Frm\_Main，在该窗体中添加一个 ListBox 控件，用于显示数组中的图书信息；添加一个 GroupBox 控件，向 GroupBox 控件中添加一个 TextBox 控件，用于用户输入信息；向 GroupBox 控件中添加一个 Button 按钮，用于开始搜索图书信息。

(3) 程序主要代码如下。

```
private void btn_query_Click(object sender, EventArgs e)
{
}
```



图 3.27 使用 switch 语句更改窗体颜色



图 3.28 使用 goto 语句在数组中搜索指定图书

```

int i = 0;
label1:
if (G_str_array[i].Contains(txt_query.Text))
{
    lbox_str.SelectedIndex = i;
    MessageBox.Show(txt_query.Text + " 已经找到!", "提示!");
    return;
}
i++;
if (i < G_str_array.Length) goto label1;
MessageBox.Show(txt_query.Text + " 没有找到!", "提示!");
}

```

//定义计数器  
//定义标签  
//判断是否找到图书  
//选中查找到的结果  
//提示找到信息  
//条件满足则跳转到标签  
//提示未找到信息

注意：从上面的实例可以看出，使用 goto 语句可以很方便地跳转到定义标签所在的位置，但是在这里建议大家不要过多地使用 goto 语句，原因是 goto 语句过于灵活，或多或少地会干扰程序的执行逻辑。

DIY：修改上面的实例，不使用 goto 语句，同样实现在数组中搜索指定图书。提示：可以使用 `ListBox.Item.IndexOf` 方法获取列表中指定项的索引。（20 分）（实例位置：光盘\mr\03\qjyy\04\_diy）

### 3.8.5 情景应用 5——制作一个数字猜猜看小游戏

视频讲解：光盘\mr\03\lx\制作一个数字猜猜看小游戏.exe

实例位置：光盘\mr\03\qjyy\05

程序的世界中充斥着大量的数据与复杂的逻辑运算，如何才能处理好数据间的逻辑关系？如何才能锻炼我们的逻辑思维能力？开发一个简单的小游戏是不错的选择！游戏中首先使用随机对象产生一个 1~100 的整数，当用户单击“开始”按钮时，动态生成 100 个按钮，并开始计时，如果用户单击的数字小于随机数，那么被单击的按钮变为橙色，并显示字符串“小”；如果单击的数字等于随机数就会弹出消息框，提示已经猜对了数字，并显示用时及猜测次数。实例运行效果如图 3.29 所示。

实现过程如下。

(1) 打开 Visual Studio 2008 开发环境，新建一个 Windows 窗体应用程序，并将其命名为 NumGame。

(2) 更改默认窗体 Form1 的 Name 属性为 Frm\_Main，向窗体中添加一个 Button 按钮，用于控制游戏开始。

(3) 程序主要代码如下。

```

private void btn_begin_Click(object sender, EventArgs e)
{
    RemoveControl();
    int p_int_x = 10;
    int p_int_y = 60;
    for (int i = 0; i < 100; i++)
    {
        Button bt = new Button();
        bt.Text = (i + 1).ToString();
        //清空所有无用对象
        //x 坐标初始值为 10
        //y 坐标初始值为 60
        //添加 100 个按钮
        //创建 Button 按钮
        //设置 Button 按钮的文本值
    }
}

```



图 3.29 制作一个数字猜猜看小游戏

```

        bt.Name = (i + 1).ToString();
        bt.Width = 35;
        bt.Height = 35;
        bt.Location = new Point(p_int_x, p_int_y);
        bt.Click += new EventHandler(bt_Click);
        p_int_x += 36;
        if ((i + 1) % 10 == 0)
        {
            p_int_x = 10;
            p_int_y += 36;
        }
        Controls.Add(bt);
    }
    G_th = new System.Threading.Thread(delegate()
    {
        int P_int_count = 0;
        while (true)
        {
            P_int_count = ++P_int_count > 100000000 ? 0 : P_int_count; //计数器累加
            this.Invoke(
                (MethodInvoker)delegate
                {
                    lb_time.Text = P_int_count.ToString(); //窗体中显示计数
                });
            System.Threading.Thread.Sleep(1000); //线程睡眠 1 秒
        }
    });
    G_th.IsBackground = true; //设置线程为后台线程
    G_th.Start(); //开始执行线程
    G_int_num = G_random.Next(1, 100); //生成随机数
    btn_begin.Enabled = false; //停用“开始”按钮
}

```

说明：本实例用到了线程技术，适当地使用线程可以提高程序的运行效率，使程序的运行更加流畅。

DIY：编写一个实现哥德巴赫猜想的小程序。提示：哥德巴赫猜想的内容是——任意一个大于 6 的偶数都可以写成两个素数的和。判断素数的方法是对给出的数值进行开方，并利用这个值除以从 1 到开方或仅小于开方后的最大整数，如果不能被整除，则是素数，否则不是素数。（20 分）（实例位置：光盘\mr\03\qjyy\05\_diy）

情景应用 DIY 栏目分数统计：

DIY 题目	1	2	3	4	5	总分数	
分数							

### 3.9 自我测试

一、选择题（每题 10 分，5 道题）

1. 下列有关 break 语句的描述中，正确的是（ ）。

- A. 循环体内的 break 语句用于结束本次循环  
 B. 循环体内的 break 语句用于跳出循环  
 C. 在循环体内, break 语句可以被执行多次  
 D. 当嵌套循环时, break 语句可以退出最外层循环
2. 对于跳转语句 break 和 continue, 下列说法正确的是( )。  
 A. continue 语句只能用于循环体内  
 B. continue 和 break 语句的跳转范围不明确, 容易产生歧义  
 C. break 语句只能用于循环体内  
 D. break 语句是无条件跳转语句, 而 continue 不是
3. 下列有关 foreach 语句的描述中, 不正确的是( )。  
 A. foreach 语句是一种循环结构语句  
 B. 可以使用 foreach 语句读取集合或数组中的元素  
 C. 可以遍历数组或集合  
 D. 可以使用 foreach 语句给集合或数组中的元素赋值
4. 下列有关 while 和 do...while 语句的描述中, 不正确的是( )。  
 A. 都可以实现死循环  
 B. while 语句可以执行零次或多次  
 C. do...while 语句至少执行一次  
 D. while 语句与 do...while 语句可以相互替换

5. 仔细查看下面的这段代码:

```
static void Main(string[] args)
{
    int i = 0;
    int j = 0;
    while (i < 3)
    {
        i++;
        if (i > 2)
        {
            break;
        }
        ++j;
    }
    Console.WriteLine(i);
    Console.WriteLine(j);
}
```

程序运行后, 其输出结果为( )。

- A. 3, 3                    B. 2, 3                    C. 3, 2                    D. 2, 2

## 二、填空题 (每题 10 分, 5 道题)

1. 在实现多分支逻辑判断时, 除了可以使用 if...else if...else 语句之外, 最好使用( )语句来实现。
2. 在 switch 语句中, 若【表达式】的值与各个 case 分支的【常量表达式】都不符合, 则程序将执行

( ) 分支的语句块。

3. 在一个循环语句中，若要终止本次循环，可使用（ ）语句；若要跳出这个循环语句，可以使用（ ）语句。

4. 若要停止程序运行或者返回方法的返回值，通常可使用（ ）语句。

5. 在下面这段程序运行后，其控制台输出结果是（ ）。

```
static int GetIntValue(int paramInt)
```

```
{
    int j = 2;
    for (int i = 0; i < paramInt; i++)
    {
        j -= 1;
        if (i > j)
        {
            break;
        }
    }
    return j;
}
static void Main(string[] args)
{
    Console.WriteLine(GetIntValue(3));
}
```

测试分数统计：

类别	第1题	第2题	第3题	第4题	第5题
选择题分数					
填空题分数					
总分数					

### 3.10 行动指南

开始日期： 年 月 日

结束日期： 年 月 日

序号	内 容	行动指南	
1	照猫画虎栏目 分数（ ）	分数>75分	优秀，基本功掌握得不错，加油！
		75分>分数>50分	及格，知识掌握得不牢，重新做一遍照猫画虎。
		分数<50分	请使用光盘中提供的“实战能力测试系统”进行提高训练。
	情景应用栏目 分数（ ）	分数>75分	优秀，综合应用能力很强。
		75分>分数>50分	及格，综合应用能力需提高，再练一遍情景应用。
		分数<50分	请使用光盘中提供的“实战能力测试系统”进行提高训练。
	自我测试栏目 分数（ ）	分数>75分	优秀，有成为编程高手的潜质。
		分数<75分	请使用光盘中提供的“实战能力测试系统”进行提高训练。
	综合评价	反复训练后，以上各项分数都在75分以上，方可进入下一堂课学习。	

续表

序号	内 容	行 动 指 南
2	编程能力培养：本栏目提供了一些实践题目，对于培养编程能力很有效，要做好。	<p>1. 使用 while 语句计算前 N 个自然数之和。提示：可使用 while 语句循环 N 次，这样需要定义一个初始值为“1”的整数变量（并且该变量在 while 循环中要自增），接着让“该整数变量”与“自然数 N”进行“小于或等于”比较，若比较结果为 True，则计算本次的和并继续执行 while 循环，否则结束循环。</p> <p>2. 求最大公约数。提示：最大公约数是指能被两个整数整除的最大整数，例如，12 与 16 两个数的最大公约数为 4。求最大公约数可以用余法实现，即用两个数中最大的数除以最小的数求余，然后使用除数除以余数求余，直到余数为 0 时，之前的除数也就是两个数的最大公约数。</p> <p>3. 求最小公倍数。提示：最小公倍数可以通过两个数的乘积除以这两个数的最大公约数得到。例如，12 与 16 的最大公约数为 4，则最小公倍数的计算方法为 <math>12 * 16 / 4</math>，计算的结果就是这两个数的最小公倍数。</p>
3	编程习惯培养：本课堂培养读者在学习开发中记笔记的习惯，把开发中遇到的问题、总结的经验记录下来。	
4	创新能力培养：看看身边有没有可以用编程解决的问题，记录到右边的表格中。根据学习进度，尝试编写解决这些问题的小程序。	

### 3.11 成功可以复制——软件业的华人教父王嘉廉

王嘉廉（Charles Wang）1944 年出生于上海法租界内，父亲曾是最高法院的法官。1952 年，8 岁的王嘉廉随父母移民美国，先是进入布鲁科技高中就读，后在市立大学皇后学院毕业。那时生活异常艰辛，王嘉廉常做一些被人瞧不起的工作，经常连 32 美分一顿的午饭都吃不上。许多年后，回忆起早年的痛苦，他总是轻描淡写地带过，只是常说比尔·盖茨经历的磨难多。“我知道饥饿的滋味，而他却不知道”。

虽然与比尔·盖茨将近 1000 亿美元的个人财富相比，王嘉廉的 10 多亿个人资产不过是盖茨的一个零头，但是人们还是常常把他们放在一起，他们都是自己公司的中心和灵魂，都是凭借铁腕式的战略攻城略地，成为软件业头两号厂商。微软在 PC 领域一统天下，CA 的企业管理软件在大型机和客户机/服务器领域也是领袖群伦。

王嘉廉在皇后学院毕业后就结束了学习生涯。毕业后，王嘉廉进入哥伦比亚大学电子研究实验室当了一位程序员新手，结果一下子就爱上了编程。他在好几家公司从事软件工作，其中包括标准数据公司。在那里，他和大学的伙伴 Russ Artzt 编写并出售用于 IBM 大型机的系统软件。两人经常拜访客户，倾听企业



信息系统管理员反映的各种问题，从而看到了商机，萌生了创业的念头。1976年，他终于可以将想法付诸实践。一家瑞士的CA国际公司正在寻求一家美国公司，代理销售CA-EARL大型机软件。后来CA找到了标准数据公司，双方成立了一个合资公司。几个月后，标准数据公司准备放弃软件业务。王嘉廉、Artzt和另外两位朋友Sedino、Habermass就成立了CA国际公司，作为瑞士公司的子公司。在曼哈顿麦迪逊大街的一间办公室中，他们开始推销CA-SORT。当时条件十分艰难，他们是用服务换回了上机机室和办公室。Sedino为房东担任招待员，Artzt和Habermass负责产品开发，王嘉廉则是销售和市场部的光杆司令，负责销售业务。在一开始没有任何业绩时，王嘉廉就为大楼的休息厅铺地毯、在走廊中挂镜子，以抵消房租。这家位于纽约的软件公司从一开始可以说一文不名，但1980年4月，他们却以280万美元买下了瑞士的母公司，从此CA完全属于他们。也就是从这时开始，CA实施了一系列的购并举措，实现了高速发展。

### 经典语录

我们并不比其他公司的人更聪明，但不同的是，我们节省了许多相互指责的时间。从错误中学到教训，不断向上成长。

### 深度评价

尽管人们常说计算机行业是年轻人拼搏的战场，但也有宝刀不老者，55岁的王嘉廉就是例证。他领导的CA在国际软件业是知名的公司，他本人也因为华人却在白人领域中闯出一番天地而更具传奇色彩。王嘉廉的勤奋、刻苦、善于把握时机、勇往直前的精神，永远值得每一个渴望成功的人学习。



# 第 4 堂课

## 字符及字符串的使用

(  视频讲解：106分钟 )

C#中的字符和字符串分别用 Char 类和 string 类来表示，C#中对于文字的处理大多是通过字符和字符串实现的，本堂课将详细讲解字符与字符串的相关内容，讲解过程中为了便于读者理解结合了大量的实例。

学习摘要：

- ▶ 了解什么是字符类
- ▶ 掌握如何对字符进行操作
- ▶ 熟悉字符串的声明及使用
- ▶ 掌握常见的字符串操作方法
- ▶ 熟悉可变字符串类 StringBuilder
- ▶ 掌握可变字符串类的定义及使用
- ▶ 熟悉可变字符串类与字符串类的区别

## 4.1 字符操作

字符是组成字符串的基础，每个字符串都是由一个或多个字符组成的，那么在 C# 中如何使用字符呢？本节将对 C# 中字符类的使用进行详细讲解，并对 C# 中提供的一种特殊的转义字符进行介绍。

### 4.1.1 Char 类概述

Char 类在 C# 中表示一个 Unicode 字符，正是这些 Unicode 字符构成了字符串。Unicode 字符是目前计算机中通用的字符编码，它为不同语言中的每个字符设定了统一的二进制编码，用于满足跨语言、跨平台的文本转换和处理要求。字符的定义非常简单，可以通过下面的代码来完成。

**例 4.01** 定义两个字符，并将它们分别赋值为“L”和“1”，代码如下。

```
char ch1='L';
char ch2='1';
```

 注意：Char 类只定义一个 Unicode 字符。

### 4.1.2 使用 Char 类中的方法对字符进行操作

Char 类提供了许多种方法，程序开发人员可以通过这些方法对字符进行各种操作。Char 类的常用方法及说明如表 4.1 所示。

表 4.1 Char 类的常用方法及说明

方 法	说 明
IsControl	指示指定的 Unicode 字符是否属于控制字符类别
IsDigit	指示某个 Unicode 字符是否属于十进制数字类别
IsHighSurrogate	指示指定的 Char 对象是否为高代理项
IsLetter	指示某个 Unicode 字符是否属于字母类别
IsLetterOrDigit	指示某个 Unicode 字符是属于字母类别还是属于十进制数字类别
IsLower	指示某个 Unicode 字符是否属于小写字母类别
IsLowSurrogate	指示指定的 Char 对象是否为低代理项
IsNumber	指示某个 Unicode 字符是否属于数字类别
IsPunctuation	指示某个 Unicode 字符是否属于标点符号类别
IsSeparator	指示某个 Unicode 字符是否属于分隔符类别
IsSurrogate	指示某个 Unicode 字符是否属于代理项字符类别
IsSurrogatePair	指示两个指定的 Char 对象是否形成代理项对
IsSymbol	指示某个 Unicode 字符是否属于符号字符类别
IsUpper	指示某个 Unicode 字符是否属于大写字母类别
IsWhiteSpace	指示某个 Unicode 字符是否属于空白类别
Parse	将指定字符串的值转换为其等效 Unicode 字符
ToLower	将 Unicode 字符的值转换为其小写等效项
ToLowerInvariant	使用固定区域性的大小写规则，将 Unicode 字符的值转换为其小写等效项
Tostring	将此实例的值转换为其等效的字符串表示

续表

方 法	说 明
ToUpper	将 Unicode 字符的值转换为其大写等效项
ToUpperInvariant	使用固定区域性的大小写规则, 将 Unicode 字符的值转换为其大写等效项
TryParse	将指定字符串的值转换为它的等效 Unicode 字符。一个指示转换是否成功的返回代码

说明: 在 Char 类的方法中, 以 Is 和 To 开头的方法比较重要, 以 Is 开头的方法大多是判断 Unicode 字符是否为某个类别, 而以 To 开头的方法主要是转换为其他 Unicode 字符。

**例 4.02** 创建一个控制台应用程序, 该程序中使用 Char 类中的方法对字符进行各种操作, 代码如下。

(实例位置: 光盘\mr\04\sl\4.02)

```
static void Main(string[] args)
{
    char a = 'a';                                //声明字符 a
    char b = '8';                                //声明字符 b
    char c = 'L';                                //声明字符 c
    char d = ':';                                //声明字符 d
    char e = '|';                                //声明字符 e
    char f = ''';                               //声明字符 f

    //使用 IsLetter 方法判断 a 是否为字母
    Console.WriteLine("IsLetter 方法判断 a 是否为字母: {0}", Char.IsLetter(a));
    //使用 IsDigit 方法判断 b 是否为数字
    Console.WriteLine("IsDigit 方法判断 b 是否为数字: {0}", Char.IsDigit(b));
    //使用 IsLetterOrDigit 方法判断 c 是否为字母或数字
    Console.WriteLine("IsLetterOrDigit 方法判断 c 是否为字母或数字: {0}", Char.IsLetterOrDigit(c));
    //使用 IsLower 方法判断 a 是否为小写字母
    Console.WriteLine("IsLower 方法判断 a 是否为小写字母: {0}", Char.IsLower(a));
    //使用 IsUpper 方法判断 c 是否为大写字母
    Console.WriteLine("IsUpper 方法判断 c 是否为大写字母: {0}", Char.IsUpper(c));
    //使用 IsPunctuation 方法判断 d 是否为标点符号
    Console.WriteLine("IsPunctuation 方法判断 d 是否为标点符号: {0}", Char.IsPunctuation(d));
    //使用 IsSeparator 方法判断 e 是否为分隔符
    Console.WriteLine("IsSeparator 方法判断 e 是否为分隔符: {0}", Char.IsSeparator(e));
    //使用 IsWhiteSpace 方法判断 f 是否为空白
    Console.WriteLine("IsWhiteSpace 方法判断 f 是否为空白: {0}", Char.IsWhiteSpace(f));
    Console.ReadLine();
}
```

程序运行结果如图 4.1 所示。

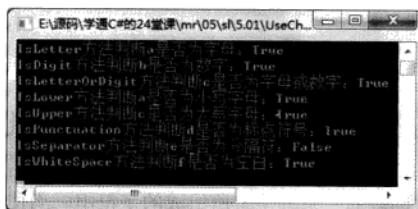


图 4.1 使用 Char 类中的方法对字符进行操作

### 4.1.3 认识并使用转义字符

C#采用字符“\”作为转义字符，其常用的转义字符及说明如表 4.2 所示。

表 4.2 转义字符及说明

转义字符	说 明
\n	回车换行
\t	横向跳到下一制表位置
\v	竖向跳格
\b	退格
\r	回车
\f	换页
\\\	反斜线符
\'	单引号符
\ddd	1~3位八进制数所代表的字符
\xhh	1~2位十六进制数所代表的字符

**技巧：**为了避免转义序列元素转义，可以通过以下两种方式避免。

- ① 通过@符实现。
- ② 通过逐字指定字符串字面值（两个反斜杠）实现。

**例 4.03** 创建一个控制台应用程序，通过转义字符使 Console.WriteLine 输出与 Console.ReadLine 相同的效果，代码如下。（实例位置：光盘\mr\04\sl\4.03）

```
static void Main(string[] args)
{
    Console.WriteLine("学通 C# 的 24 堂课"); //通过 Console.WriteLine 输出字符串
    Console.Write("学通 C# 的 24 堂课\n"); //通过使用转义字符输出字符串
    Console.Write("学通 C# 的 24 堂课\nC# 编程词典"); //通过使用转义字符输出字符串
    Console.ReadLine();
}
```

程序运行结果如图 4.2 所示。



图 4.2 转义字符的使用

## 4.2 字符串的声明及初始化

string 类表示由 Unicode 格式编码的一系列字符所组成的字符串。当使用 string 类时，一旦创建了其对象，就不能再进行修改。在实际使用 string 类时，表面看来能够修改字符串的所有方法实际上并不能修改，它们实际上返回一个根据所调用的方法修改的新的 string 对象，如果需要修改 string 字符串的实际内容，可

以使用 `StringBuilder` 类。

字符串是 Unicode 字符的有序集合，用于表示文本，当声明字符串时，可以使用 `string` 类型加变量名称来表示。

**例 4.04** 声明一个名称为 str 的字符串，代码如下。

```
string str;
```

声明完一个字符串后，需要对其进行初始化。当初始化字符串时，可以通过使用=运算符来实现。

**例 4.05** 声明一个名称为 str 的字符串，并将其初始化为“C#编程词典”，代码如下。

```
string str = "C#编程词典";
```

## 4.3 字符串操作

在开发程序时会经常遇到对字符串的操作，如比较字符串、格式化字符串、截取字符串、分割字符串、插入字符串、填充字符串、删除字符串、复制和替换字符串等，本节将对字符串的各种操作进行详细讲解。

### 4.3.1 比较字符串

C#中最常见的比较字符串的方法有 `Compare`、`CompareTo` 和 `Equals` 等，这些方法都归属于 `string` 类，下面对这 3 种方法进行详细介绍。

#### 1. Compare 方法

`Compare` 方法用来比较两个字符串是否相等，它有多种重载方法，其中最常用的两种重载格式如下。

语法：int `Compare(string strA, string strB)`

`int Compare(string strA, string strB, bool ignoreCase)`

说明：`strA` 和 `strB` 代表要比较的两个字符串。`ignoreCase` 是一个布尔类型的参数，如果该参数的值是 `True`，那么在比较字符串时就忽略大小写的差别。另外，它的返回值是一个 32 位有符号整数，指示两个比较数之间的词法关系。

 **说明：**`Compare` 方法是一个静态方法，使用时可以直接调用。

**例 4.06** 创建一个控制台应用程序，声明两个字符串变量，然后使用 `Compare` 方法比较两个字符串是否相等，代码如下。（实例位置：光盘\mr\04\sl\4.06）

```
static void Main(string[] args)
{
    string str1 = "学通 C# 的 24 堂课";
    string str2 = "C#编程词典";
    Console.WriteLine(string.Compare(str1, str2));
    Console.WriteLine(string.Compare(str1, str1));
    Console.WriteLine(string.Compare(str2, str1));
    Console.ReadLine();
}
```

程序运行结果为：1 0 -1。

 **注意：**比较字符串并非比较字符串长度的大小，而是比较字符串在英文字典中的位置。比较字符串时，按照字典排序的规则，判断两个字符串的大小，在英文字典中，前面的单词小于后面的单词。

## 2. CompareTo 方法

CompareTo 方法与 Compare 方法相似，都可以比较两个字符串是否相等，不同的是 CompareTo 方法以实例对象本身与指定的字符串作比较。

语法：public int CompareTo(string strB)

说明：strB 表示与此字符串相比较的字符串。它的返回值是一个 32 位有符号整数，指示两个比较数之间的语法关系。

**例 4.07** 创建一个控制台应用程序，使用 CompareTo 方法比较两个字符串是否相等，代码如下。（实例位置：光盘\mr\04\sl\4.07）

```
static void Main(string[] args)
{
    string str1 = "学通 C# 的 24 堂课";           // 声明一个字符串 str1
    string str2 = "C# 编程词典";                   // 声明一个字符串 str2
    Console.WriteLine(str1.CompareTo(str2));        // 输出 str1 与 str2 比较后的返回值
    Console.ReadLine();
}
```

程序运行结果为：1。

## 3. Equals 方法

Equals 方法主要用于比较两个字符串是否相同，如果相同，则返回值为 True，否则为 False，其常用的两种语法格式如下。

语法：public bool Equals(string value)

public static bool Equals(string a, string b)

说明：value 表示与实例比较的字符串。a 和 b 是要进行比较的两个字符串。另外，它的返回值是一个 bool 类型的值，如果两个值相同，则为 True，否则为 False。

**例 4.08** 创建一个控制台应用程序，声明两个字符串变量，然后使用 Equals 方法比较两个字符串是否相同，代码如下。（实例位置：光盘\mr\04\sl\4.08）

```
static void Main(string[] args)
{
    string str1 = "学通 C# 的 24 堂课";           // 声明一个字符串 str1
    string str2 = "C# 编程词典";                   // 声明一个字符串 str2
    Console.WriteLine(str1.Equals(str2));           // 用 Equals 方法比较字符串 str1 和 str2
    Console.WriteLine(string.Equals(str1, str2));   // 用 Equals 方法比较字符串 str1 和 str2
    Console.ReadLine();
}
```

程序运行结果为：False False。

### 4.3.2 格式化字符串

在 C# 中，string 类提供了一个静态的 Format 方法，用于将字符串数据格式化成指定的格式。

语法：Public static string Format(string format, object obj);

说明：format 用来指定字符串所要格式化的形式；obj 表示要被格式化的对象。另外，它的返回值是 format 的一个副本。

**例 4.09** 创建一个控制台应用程序，声明两个 string 类型的变量 strA 和 strB，然后使用 Format 方法格

式化这两个 string 类型变量，最后输出格式化后的字符串，代码如下。（实例位置：光盘\mr\04\sl\4.09）

```
static void Main(string[] args)
{
    string strA = "学通 C# 的 24 堂课";
    string strB = "经典图书";
    string newstr = string.Format("{0},{1}!!!", strA, strB);
    Console.WriteLine(newstr);
    Console.ReadLine();
}
```

程序运行结果为：学通 C# 的 24 堂课，经典图书！！！。

如果希望日期时间按照某种格式输出，则可以使用 Format 方法将日期时间格式化为指定的格式。C#中提供了一些用于日期时间的格式规范，具体描述如表 4.3 所示。

表 4.3 用于日期时间的格式规范

格 式 规 范	说 明
d	简短日期格式 (YYYY-MM-dd)
D	完整日期格式 (YYYY 年 MM 月 dd 日)
t	简短时间格式 (hh:mm)
T	完整时间格式 (hh:mm:ss)
f	简短的日期/时间格式 (YYYY 年 MM 月 dd 日 hh:mm)
F	完整的日期/时间格式 (YYYY 年 MM 月 dd 日 hh:mm:ss)
g	简短的可排序的日期/时间格式 (YYYY-MM-dd hh:mm)
G	完整的可排序的日期/时间格式 (YYYY-MM-dd hh:mm:ss)
M 或 m	月/日格式 (MM 月 dd 日)
Y 或 y	年/月格式 (YYYY 年 MM 月)

下面通过一个实例演示如何使用日期时间的格式规范（以格式规范 D 为例）。

**例 4.10** 创建一个控制台应用程序，声明一个 DateTime 类型的变量 dt，用于获取系统的当前日期时间，然后通过使用日期格式规范 D 将日期时间格式化为“YYYY 年 MM 月 dd 日”的格式，代码如下。（实例位置：光盘\mr\04\sl\4.10）

```
static void Main(string[] args)
{
    DateTime dt = DateTime.Now; // 获取系统当前日期
    string strB = string.Format("{0:D}", dt); // 格式化成短日期格式
    Console.WriteLine(strB); // 输出日期
    Console.ReadLine();
}
```

程序运行结果为：2010 年 11 月 16 日。

### 4.3.3 截取字符串

截取字符串时需要用到 string 类的 Substring 方法，该方法可以截取字符串中指定位置和指定长度的字符。

语法：public string Substring(int startIndex,int length)

说明：startIndex 表示子字符串的起始位置的索引；length 表示子字符串中的字符数。另外，其返回值是一个字符串，它等于此实例中从 startIndex 开始的长度为 length 的子字符串，如果 startIndex 等于此实例

的长度且 length 为零，则为空。

**例 4.11** 创建一个控制台应用程序，声明两个 string 类型的变量 strA 和 strB，并将 strA 初始化为“学通 C# 的 24 堂课”，然后使用 Substring 方法从 strA 字符串的索引 0 处开始截取 4 个字符，赋值给 strB，并输出 strB，代码如下。（实例位置：光盘\mr\04\sl\4.11）

```
static void Main(string[] args)
{
    string strA = "学通 C# 的 24 堂课";
    string strB = "";
    strB = strA.Substring(0, 4);
    Console.WriteLine(strB);
    Console.ReadLine();
}
```

程序运行结果为：学通 C#。

#### 4.3.4 分割字符串

分割字符串时需要用到 string 类的 Split 方法，该方法用于分割字符串，其返回值是包含所有分割子字符串的数组对象，可以通过数组获得所有分割的子字符串。

语法：public string[] Split(params char[] separator);

说明：separator 是一个数组，包含分隔符。其返回值是一个数组，其元素包含此实例中的子字符串，这些子字符串由 separator 中的一个或多个字符分隔。

**例 4.12** 创建一个控制台应用程序，声明一个 string 类型变量 strA，并将其初始化为“学通<C#>的 24 堂课”，然后通过 Split 方法分割变量 strA，并输出分割后的字符串，代码如下。（实例位置：光盘\mr\04\sl\4.12）

```
static void Main(string[] args)
{
    string strA = "学通<C#>的 24 堂课";
    char[] separator = {'<', '>'};
    string[] splitstrings = new string[100];
    splitstrings = strA.Split(separator);
    for (int i = 0; i < splitstrings.Length; i++)
    {
        Console.WriteLine("项{0}:{1}", i, splitstrings[i]);
    }
    Console.ReadLine();
}
```

程序运行结果如图 4.3 所示。



图 4.3 分割字符串

#### 4.3.5 插入和填充字符串

插入和填充字符串分别用到 string 类中的 Insert 方法和 PadLeft/PadRight 方法，下面分别对它们进行详

细讲解。

### 1. 插入字符串

Insert 方法用于向字符串的任意位置插入新元素。

语法：public string Insert(int startIndex, string value);

说明：startIndex 用于指定所要插入的位置，索引从 0 开始；value 用于指定所要插入的字符串。另外，其返回值是此字符串的一个新 String 等效项，但在位置 startIndex 处插入 value。

**例 4.13** 创建一个控制台应用程序，声明 3 个 string 类型的变量 str1、str2 和 str3。将变量 str1 初始化为“编程”，然后使用 Insert 方法在字符串 str1 的索引 0 处插入字符串“C#”，并将其赋值给字符串 str2，最后在字符串 str2 的索引 4 处插入字符串“词典”，同时赋值给字符串 str3，并输出 str3，代码如下。（实例位置：光盘\mr\04\sl\4.13）

```
static void Main(string[] args)
{
    string str1 = "编程";                                //声明字符串变量 str1 并赋值为“编程”
    string str2;                                         //声明字符串变量 str2
    str2 = str1.Insert(0,"C#");                          //使用 Insert 方法向字符串 str1 中插入字符串
    string str3 = str2.Insert(4,"词典");                  //使用 Insert 方法向字符串 str2 中插入字符串
    Console.WriteLine(str3);                            //输出字符串变量 str3
    Console.ReadLine();
}
```

程序运行结果为：C#编程词典。

### 2. 填充字符串

PadLeft/PadRight 方法用于填充字符串，其中，PadLeft 方法在字符串的左侧进行字符填充，而 PadRight 方法在字符串的右侧进行字符填充。PadLeft 方法的语法格式如下。

语法：public string PadLeft(int totalWidth,char paddingChar)

说明：totalWidth 表示结果字符串中的字符数，等于原始字符数加上任何其他填充字符。paddingChar 用来填充字符。另外，其返回值等效于此实例的一个新 String，但它是右对齐的，并在左边用达到 totalWidth 长度所需数目的 paddingChar 字符进行填充。如果 totalWidth 小于此实例的长度，则为与此实例相同的新 String。

PadRight 方法的语法格式如下。

语法：public string PadRight(int totalWidth,char paddingChar)

说明：totalWidth 表示结果字符串中的字符数，等于原始字符数加上任何其他填充字符。paddingChar 用来填充字符。另外，其返回值等效于此实例的一个新 String，但它是左对齐的，并在右边用达到 totalWidth 长度所需数目的 paddingChar 字符进行填充。如果 totalWidth 小于此实例的长度，则为与此实例相同的新 String。

**例 4.14** 创建一个控制台应用程序，声明 3 个 string 类型的变量 str1、str2 和 str3。将 str1 初始化为“\*^\_\_^\*”，然后使用 PadLeft 方法在 str1 的左侧填充字符“（”，并将其赋给字符串 str2，再使用 PadRight 方法在字符串 str2 的右侧填充字符“）”，最后得到字符串“(\*^\_\_^\*)”，赋给字符串 str3，并输出字符串 str3，代码如下。（实例位置：光盘\mr\04\sl\4.14）

```
static void Main(string[] args)
{
    string str1 = "*^__^*";                                //声明一个字符串变量 str1
```

```

//声明一个字符串变量 str2，并使用 PadLeft 方法在 str1 的左侧填充字符“（”
string str2 = str1.PadLeft(7, '(');
//声明一个字符串变量 str3，并使用 PadRight 方法在 str2 的右侧填充字符“）”
string str3 = str2.PadRight(8, ')';
Console.WriteLine("补充字符串之前: " + str1); //输出字符串 str1
Console.WriteLine("补充字符串之后: " + str3); //输出字符串 str2
Console.ReadLine();
}

```

程序运行结果如图 4.4 所示。



图 4.4 填充字符串

### 4.3.6 删除字符串

删除字符串时用到了 string 类提供的 Remove 方法，该方法用于从一个字符串的指定位置开始，删除指定数量的字符。

语法：public string Remove( int startIndex);

public string Remove( int startIndex, int count);

说明：startIndex 用于指定开始删除的位置，索引从 0 开始；count 用于指定删除的字符数量。另外，其返回值是一个新的 String 字符串。

说明：第一种语法格式删除字符串中从指定位置到最后位置的所有字符；第二种语法格式从字符串中指定位置开始删除指定数目的字符。

**例 4.15** 创建一个控制台应用程序，声明一个 string 类型的变量 str1，并初始化为“学通 C# 的 24 堂课”，然后分别使用 Remove 方法的第一种语法格式和第二种语法格式删除索引 4 后面的所有字符以及从字符串 str1 的索引 2 处删除一个字符，代码如下。（实例位置：光盘\mr\04\sl\4.15）

```

static void Main(string[] args)
{
    string str1 = "学通 C# 的 24 堂课"; //声明一个字符串变量 str1
    //声明一个字符串变量 str2，并使用 Remove 方法从字符串 str1 的索引 4 处开始删除
    string str2 = str1.Remove(4);
    Console.WriteLine(str2); //输出字符串 str2
    //声明一个字符串变量 str3，并使用 Remove 方法从字符串 str1 的索引 2 处删除一个字符
    string str3 = str1.Remove(2,1);
    Console.WriteLine(str3); //输出字符串 str3
    Console.ReadLine();
}

```

程序运行结果为：学通 C# 学通#的 24 堂课。

### 4.3.7 复制字符串

string 类提供了 Copy 和 CopyTo 方法，用于将字符串或子字符串复制到另一个字符串或 Char 类型的数

组中，下面分别对它们进行详细讲解。

### 1. Copy 方法

创建一个与指定的字符串具有相同值的字符串的新实例。

语法：public static string Copy(string str)

说明：str 表示要复制的字符串。其返回值是一个与 str 具有相同值的字符串。

**例 4.16** 创建一个控制台应用程序，声明一个 string 类型的变量 strA，并初始化为“学通 C# 的 24 堂课”，然后使用 Copy 方法复制字符串 strA，并赋值给字符串 strB，最后输出字符串 strB，代码如下。（实例位置：光盘\mr\04\sl\4.16）

```
static void Main(string[] args)
{
    string strA = "学通 C# 的 24 堂课"; //声明一个字符串变量 strA 并初始化
    string strB; //声明一个字符串变量 strB
    //使用 string 类的 Copy 方法，复制字符串 strA 并赋值给 strB
    strB = string.Copy(strA);
    Console.WriteLine(strB); //输出字符串 strB
    Console.ReadLine();
}
```

程序运行结果为：学通 C# 的 24 堂课。

### 2. CopyTo 方法

CopyTo 方法的功能与 Copy 方法基本相同，但是 CopyTo 方法可以将字符串的某一部分复制到另一个数组中。

语法：public void CopyTo(int sourceIndex,char[ ]destination,int destinationIndex,int count);

CopyTo 方法语法中的参数说明如表 4.4 所示。

表 4.4 CopyTo 方法语法中的参数说明

参 数	说 明
sourceIndex	需要复制的字符的起始位置
destination	目标字符数组
destinationIndex	指定目标数组中的开始存放位置
count	指定要复制的字符个数

**例 4.17** 创建一个控制台应用程序，声明一个 string 类型变量 str，并初始化为“C#编程词典”，然后声明一个 Char 类型的数组 myChar，使用 CopyTo 方法将 str 字符串中索引从 2 开始的 4 个字符赋值到 myChar 数组中，并输出，代码如下。（实例位置：光盘\mr\04\sl\4.17）

```
static void Main(string[] args)
{
    string str = "C#编程词典"; //声明一个字符串变量 str 并初始化
    char[] myChar = new char[100]; //声明一个字符数组 myChar
    //将字符串 str 中从索引 2 开始的 4 个字符串复制到字符数组 myChar 中
    str.CopyTo(2, myChar, 0, 4);
    Console.WriteLine(myChar); //输出字符数组中的内容
    Console.ReadLine();
}
```

程序运行结果为：编程词典。

### 4.3.8 替换字符串

替换字符串时用到了 `string` 类提供的 `Replace` 方法，该方法用于将字符串中的某个字符或字符串替换成其他的字符或字符串。

语法： `public string Replace(char OChar,char NChar)`

`public string Replace(string OValue,string NValue)`

`Replace` 方法语法中的参数说明如表 4.5 所示。

表 4.5 `Replace` 方法语法中的参数说明

参 数	说 明
<code>OChar</code>	待替换的字符
<code>NChar</code>	替换后的新字符
<code>OValue</code>	待替换的子字符串
<code>NValue</code>	替换后的新子字符串
返回值	替换子字符串之后的新字符串

 说明：第一种语法格式主要用于替换字符串中指定的字符，第二种语法格式主要用于替换字符串中指定的字符串。

**例 4.18** 创建一个控制台应用程序，声明一个 `string` 类型变量 `strA`，并初始化为“one world,one dream”，然后分别使用 `Replace` 方法的第一种语法格式和第二种语法格式将字符串 `strA` 中的“，”替换成“\*”和将字符串 `strA` 中的“one world”替换成“One World”，最后分别输出替换后的字符串，代码如下。（实例位置：光盘\mr\04\sl\4.18）

```
static void Main(string[] args)
{
    string strA = "one world,one dream"; //声明一个字符串变量 strA 并初始化
    //使用 Replace 方法将字符串 strA 中的“，”替换成“*”，并赋值给字符串变量 strB
    string strB = strA.Replace(',', '*');
    Console.WriteLine(strB); //输出字符串变量 strB
    //使用 Replace 方法将字符串 strA 中的“one world”替换成“One World”
    string strC = strA.Replace("one world", "One World");
    Console.WriteLine(strC); //输出字符串变量 strC
    Console.ReadLine();
}
```

程序运行结果为： one world\*one dream One World,one dream。

## 4.4 可变字符串类 `StringBuilder` 的使用

在程序中遇到对某个字符串的多种操作时，通常都考虑使用可变字符串，而 C# 中提供了 `StringBuilder` 类来表示可变字符串，本节将对其进行详细介绍。

### 4.4.1 `StringBuilder` 类概述

`StringBuilder` 类表示可变字符串，它位于 `System.Text` 命名空间下，之所以说 `StringBuilder` 对象的值是

可变的，是因为在通过追加、移除、替换或插入字符而创建它之后可以对它进行修改。

- 说明：**
- ① `StringBuilder` 的容量是对象在任何给定时间可存储的最大字符数，并且大于或等于对象值的字符串表示形式的长度。容量可通过 `Capacity` 属性或 `EnsureCapacity` 方法来增加或减少，但它不能小于 `Length` 属性的值。
  - ② 如果在初始化 `StringBuilder` 的对象时没有指定容量或最大容量，则使用 `StringBuilder` 对象的默认值。

#### 4.4.2 创建 `StringBuilder` 对象

创建 `StringBuilder` 对象时，需要使用 `StringBuilder` 类的构造函数，它有 6 种不同的构造函数，语法格式分别如下。

```
语法: public StringBuilder()
public StringBuilder(int capacity)
public StringBuilder(string value)
public StringBuilder(int capacity,int maxCapacity)
public StringBuilder(string value,int capacity)
public StringBuilder(string value,int startIndex,int length,int capacity)
```

`StringBuilder` 类的构造函数语法中的参数说明如表 4.6 所示。

表 4.6 `StringBuilder` 类的构造函数语法中的参数说明

参    数	说    明
<code>capacity</code>	<code>StringBuilder</code> 对象的建议起始大小
<code>value</code>	字符串，包含用于初始化 <code>StringBuilder</code> 对象的子字符串
<code>maxCapacity</code>	当前字符串可包含的最大字符数
<code>startIndex</code>	<code>value</code> 中子字符串开始的位置
<code>length</code>	子字符串中的字符数

**例 4.19** 创建一个 `StringBuilder` 对象，其初始引用的字符串为“Hello World!”，代码如下。

```
StringBuilder myStringBuilder = new StringBuilder("Hello World!");
```

#### 4.4.3 `StringBuilder` 类的使用

对 `StringBuilder` 对象进行操作时，需要用到 `StringBuilder` 类提供的方法，其常用方法及说明如表 4.7 所示。

表 4.7 `StringBuilder` 类中的常用方法及说明

方    法	说    明
<code>Append</code>	将文本或字符串追加到指定对象的末尾
<code>AppendFormat</code>	自定义变量的格式并将这些值追加到 <code>StringBuilder</code> 对象的末尾
<code>Insert</code>	将字符串或对象添加到当前 <code>StringBuilder</code> 对象中的指定位置
<code>Remove</code>	从当前 <code>StringBuilder</code> 对象中移除指定数量的字符
<code>Replace</code>	用另一个指定的字符来替换 <code>StringBuilder</code> 对象内的字符

下面通过实例来演示如何使用 `StringBuilder` 类中的这 5 种方法。

**例 4.20** 创建一个控制台应用程序，声明一个 `int` 类型的变量 `Num`，并初始化为 368，然后创建一个

StringBuilder 对象 SBuilder，其初始值为“明日科技”，初始大小为 100。之后分别使用 StringBuilder 类的 Append、AppendFormat、Insert、Remove 和 Replace 方法对 StringBuilder 对象进行操作，代码如下。（实例位置：光盘\mr\04\sI4.20）

```
static void Main(string[] args)
{
    int Num = 368; //声明一个 int 类型变量 Num 并初始化为 368
    StringBuilder SBuilder = new StringBuilder("明日科技", 100); //创建一个 StringBuilder 类，并初始化为“明日科技”
    SBuilder.Append("C#编程词典"); //使用 Append 方法将字符串追加到 SBuilder 的末尾
    Console.WriteLine(SBuilder); //输出 SBuilder
    //使用 AppendFormat 方法将字符串按照指定的格式追加到 SBuilder 的末尾
    SBuilder.AppendFormat("{0:C}", Num);
    Console.WriteLine(SBuilder); //输出 SBuilder
    SBuilder.Insert(0, "软件: "); //使用 Insert 方法将“软件: ”追加到 SBuilder 的开头
    Console.WriteLine(SBuilder); //输出 SBuilder
    SBuilder.Remove(14, SBuilder.Length - 14); //使用 Remove 方法从 SBuilder 中删除索引 14 以后的字符串
    Console.WriteLine(SBuilder); //输出 SBuilder
    SBuilder.Replace("软件", "软件工程师必备"); //使用 Replace 方法将“软件”替换成“软件工程师必备”
    Console.WriteLine(SBuilder); //输出 SBuilder
    Console.ReadLine();
}
```

程序运行结果如图 4.5 所示。

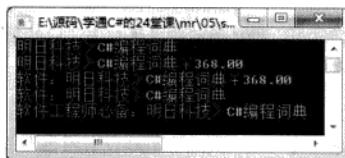


图 4.5 StringBuilder 类的应用

#### 4.4.4 StringBuilder 类与 string 类的区别

string 类是不可改变的，在每次使用 string 类中的方法时，都要在内存中创建一个新的字符串对象，这就需要为该新对象分配新的空间。在需要对字符串执行重复修改的情况下，与创建新的 string 对象相关的系统开销可能会非常昂贵。如果要修改字符串而不创建新的对象，则可以使用 StringBuilder 类。例如，当在一个循环中将许多字符串连接在一起时，使用 StringBuilder 类可以提升性能。

**例 4.21** 下面通过一个例子来具体看一下 StringBuilder 类与 String 类的区别，首先看下面一段代码。

```
string a="aa"+ "bb";
StringBuilder sb=new StringBuilder();
sb.append("aa");
sb.append("bb");
```

上面的代码分别使用 string 类和 StringBuilder 类连接字符串，但这两种方法在内存中的操作是不同的，使用第一种方法在内存中操作时有 3 个 string（分别为 aa、bb、aabb）变量；而使用第二种方法在内存中操作时只有一个（aabb）变量，所以它们的性能是完全不同的。下面通过两个图来形象地看一下 string 类和 StringBuilder 类在连接字符串时的具体过程。其中，图 4.6 表示 string 类示意图，图 4.7 表示 StringBuilder 类示意图。

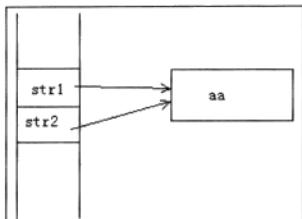


图 4.6 string 类示意图

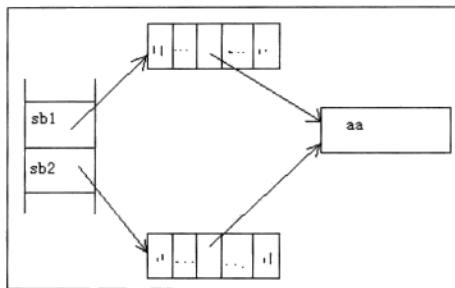


图 4.7 StringBuilder 类示意图

**技巧：**当程序中需要大量地对某个字符串进行操作时应考虑使用 `StringBuilder` 类处理该字符串，其设计目的就是针对大量 `string` 操作的一种改进办法，避免产生太多的临时对象。当程序中只是对某个字符串进行一次或几次操作时，采用 `string` 类即可。

## 4.5 照猫画虎——基本功训练

### 4.5.1 基本功训练 1——判断用户输入的用户名是否正确

视频讲解：光盘\mr\04\lx\判断用户输入的用户名是否正确.exe

实例位置：光盘\mr\04\zmhh\01

新建一个 Windows 窗体应用程序，在窗体中添加一个 `TextBox` 控件和一个 `Button` 控件，然后在 `Button` 控件的 `Click` 事件中判断用户输入的用户名是否正确，并弹出相应的信息提示，代码如下。

```
private void button1_Click(object sender, EventArgs e)
{
    if (txt_username.Text == "Admin") //判断是否输入正确用户名
    {
        MessageBox.Show("登录成功！", "提示！"); //提示登录成功
    }
    else
    {
        MessageBox.Show("用户名错误", "错误！"); //提示登录不成功
    }
}
```

程序运行结果如图 4.8 所示。



图 4.8 判断用户输入的用户名是否正确

**照猫画虎：**根据以上程序，制作一个判断用户输入的用户名和密码是否正确的程序。提示：在以上窗体中再添加一个 `TextBox` 控件，用来输入密码。(20 分)(实例位置：光盘\mr\04\zmhh\01\_zmhh)

### 4.5.2 基本功训练 2——将字符串的每个字符进行颠倒输出

视频讲解：光盘\mr\04\lx\将字符串的每个字符进行颠倒输出.exe

实例位置：光盘\mr\04\zmhh\02

新建一个 Windows 窗体应用程序，在窗体中添加两个 TextBox 控件，然后在第一个 TextBox 控件的 TextChanged 事件中通过调用 Array 类的 Reverse 方法对输入字符串中的每个字符进行颠倒输出，代码如下。

```
private void txt_input_TextChanged(object sender, EventArgs e)
{
    char[] P_chr = txt_input.Text.ToCharArray(); //从字符串中得到字节数组
    Array.Reverse(P_chr, 0, txt_input.Text.Length); //反转字节数组
    txt_output.Text = new StringBuilder().Append(P_chr).ToString(); //将字节数组转换为字符串并输出
}
```

程序运行结果如图 4.9 所示。



图 4.9 将字符串的每个字符进行颠倒输出

照猫画虎：根据以上程序，将字符串“吉林省明日科技有限公司”中的“明日科技”4个字符进行反转。  
(20 分) (实例位置：光盘\mr\04\zmhh\02\_zmhh)

### 4.5.3 基本功训练 3——去掉字符串中的所有空格

视频讲解：光盘\mr\04\lx\去掉字符串中的所有空格.exe

实例位置：光盘\mr\04\zmhh\03

新建一个 Windows 窗体应用程序，在窗体中添加两个 TextBox 控件和一个 Button 控件，然后在 Button 控件的 Click 事件中对第一个文本框中输入的字符串进行处理，去掉其中的所有空格，并将新得到的字符串显示在第一个文本框中，代码如下。

```
private void btn_RemoveBlank_Click(object sender, EventArgs e)
{
    char[] P_chr = txt_str.Text.ToCharArray(); //得到字符数组
    //得到枚举器
    IEnumerator P_ienumerator_chr = P_chr.GetEnumerator();
    //创建 stringbuilder 对象
    StringBuilder P_stringbuilder = new StringBuilder();
    while (P_ienumerator_chr.MoveNext()) //开始枚举
    {
        P_stringbuilder.Append( //向 stringbuilder 对象中添加非空格字符
            (char)P_ienumerator_chr.Current != ' ' ? P_ienumerator_chr.Current.ToString() : string.Empty);
    }
    txt_removeblank.Text = P_stringbuilder.ToString(); //得到没有空格的字符串
}
```

程序运行结果如图 4.10 所示。

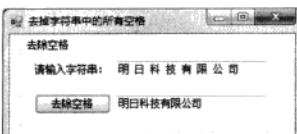


图 4.10 去掉字符串中的所有空格

**照猫画虎：**制作一个去掉字符串开始空格的程序。提示：使用 string 类的 TrimStart 方法实现。(20分)  
(实例位置：光盘\mr\04\zmhh\03\_zmhh)

#### 4.5.4 基本功训练 4——获取字符串中汉字的个数

■**视频讲解：**光盘\mr\04\lx\获取字符串中汉字的个数.exe

■**实例位置：**光盘\mr\04\zmhh\04

新建一个 Windows 窗体应用程序，在窗体中添加两个 TextBox 控件和一个 Button 控件，然后在 Button 控件的 Click 事件中通过使用正则表达式获取第一个文本框中输入的字符串中汉字的个数，代码如下。

```
private void btn_GetCount_Click(object sender, EventArgs e)
{
    int P_scalar = 0; //定义值类型变量并赋值为 0
    //创建正则表达式对象，用于判断字符是否为汉字
    Regex P_regex = new Regex("^\u4E00-\u9FA5{0,}$");
    for (int i = 0; i < txt_str.Text.Length; i++)
    {
        P_scalar = P_regex.IsMatch(txt_str.Text[i].ToString()) ? ++P_scalar : P_scalar;
        //如果检查的字符是汉字，则计数器加 1
    }
    txt_count.Text = P_scalar.ToString(); //显示汉字数量
}
```

程序运行结果如图 4.11 所示。

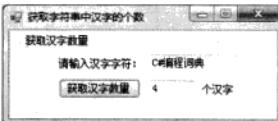


图 4.11 获取字符串中汉字的个数

**照猫画虎：**制作一个程序，用来获取字符串中的所有汉字。提示：在上面程序的基础上，使用一个字符串变量记录即可。(20分)(实例位置：光盘\mr\04\zmhh\04\_zmhh)

#### 4.5.5 基本功训练 5——从字符串中分离文件路径、文件名及扩展名

■**视频讲解：**光盘\mr\04\lx\从字符串中分离文件路径、文件名及扩展名.exe

■**实例位置：**光盘\mr\04\zmhh\05

新建一个 Windows 窗体应用程序，在窗体中添加一个 Button 控件、一个 OpenFileDialog 组件和 3 个 Label 控件，然后在 Button 控件的 Click 事件中，通过使用 OpenFileDialog 组件打开“打开”对话框。在该对话框中选择文件后，将选择的文件的文件路径、文件名及扩展名显示在相应的 Label 控件中，代码如下。

```

private void btn_Openfile_Click(object sender, EventArgs e)
{
    if (openFileDialog1.ShowDialog() == DialogResult.OK) //判断是否选择了文件
    {
        string P_str_all = openFileDialog1.FileName; //记录选择的文件全路径
        string P_str_path = P_str_all.Substring(0, P_str_all.LastIndexOf("\\") + 1); //获取文件路径
        string P_str_filename = P_str_all.Substring(P_str_all.LastIndexOf("\\") + 1, P_str_all.LastIndexOf(".") - (P_str_all.LastIndexOf("\\") + 1)); //获取文件名
        //获取文件扩展名
        string P_str_fileexc = P_str_all.Substring(P_str_all.LastIndexOf(".") + 1, P_str_all.Length - P_str_all.LastIndexOf(".")); //显示文件路径
        lb_filepath.Text = "文件路径: " + P_str_path; //显示文件名
        lb_filename.Text = "文件名称: " + P_str_filename; //显示扩展名
        lb_fileexc.Text = "文件扩展名: " + P_str_fileexc;
    }
}

```

程序运行结果如图 4.12 所示。

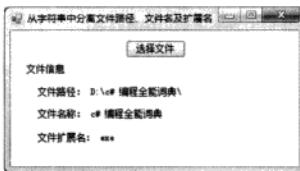


图 4.12 从字符串中分离文件路径、文件名及扩展名

**照猫画虎：**制作一个程序，用来从字符串中获取文件全名称。**(20 分)** (实例位置：光盘\mr\04\zmhh\05\_zmhh)

照猫画虎栏目分数统计：

照猫画虎题目	1	2	3	4	5	总分数	
分数							

## 4.6 情景应用——拓展与实践

### 4.6.1 情景应用 1——字母与 ASCII 码的转换

■ 视频讲解：光盘\mr\04\lx\字母与 ASCII 码的转换.exe

■ 实例位置：光盘\mr\04\qjyy\01

ASCII (American Standard Code for Information Interchange) 作为美国信息互换标准代码，它是基于拉丁字母的编码系统，也是现今最通用的单字节编码系统。在程序设计中，可以方便地将字母转换为 ASCII 码，也可以将 ASCII 码方便地转换为字母。新建一个 Windows 窗体应用程序，在窗体上添加 4 个 TextBox 文本框和两个 Button 控件，然后在窗体的代码页中编写如下代码。

```

private void btn_ToASCII_Click(object sender, EventArgs e)
{
}

```

```

if (txt_char.Text != string.Empty) //判断输入是否为空
{
    if ((Encoding.GetEncoding("unicode").GetBytes(new char[] { txt_char.Text[0] })[1] == 0)
        //判断输入是否为字母
    {
        txt_ASCII.Text = Encoding.GetEncoding("unicode").GetBytes(txt_char.Text)[0].ToString();
        //得到字符的 ASCII 码值
    }
    else
    {
        txt_ASCII.Text = string.Empty;
        MessageBox.Show("请输入字母！","提示！");
    }
}
private void btn_ToChar_Click(object sender, EventArgs e)
{
    if (txt_ASCII2.Text != string.Empty) //判断输入是否为空
    {
        int P_int_Num;
        if (int.TryParse(txt_ASCII2.Text, out P_int_Num))
        {
            txt_Char2.Text = ((char)P_int_Num).ToString(); //将 ASCII 码转换为字符
        }
        else
        {
            MessageBox.Show("请输入正确 ASCII 码值。","错误！"); //如果输入不符合要求，则弹出提示框
        }
    }
}
}

```

程序运行结果如图 4.13 所示。

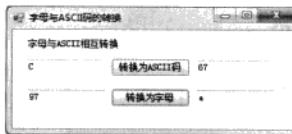


图 4.13 字母与 ASCII 码的转换

**DIY：**同时获取多个字母的 ASCII 码值。**提示：**遍历输入的多个字母，然后对遍历到的每一个字母获取 ASCII 码值，最后将获取到的 ASCII 码值连接起来并使用空格隔开。(20 分)(实例位置：光盘\mr\04\qjyy\01\_diy)

#### 4.6.2 情景应用 2——将汉字转换为拼音

■**视频讲解：**光盘\mr\04\lx\将汉字转换为拼音.exe

■**实例位置：**光盘\mr\04\qjyy\02

用户在使用计算机时经常使用拼音或五笔等输入法向文档中输入汉字，使用拼音输入法得到汉字的过

程是，通过用户输入的拼音，输入法会智能地匹配到相应的汉字或汉字中的词并输出，这里将实现将汉字转换为拼音的功能。运行效果如图 4.14 所示。

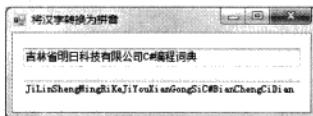


图 4.14 将汉字转换为拼音

实现过程如下。

(1) 创建一个 Windows 窗体应用程序，将其命名为 ChineseToABC，并将默认窗体 Form1.cs 修改为 Frm\_Main.cs。

(2) 向窗体中添加两个 TextBox 控件，分别用于输入汉字信息和输出拼音信息。

(3) 在当前项目中添加一个类，并将其命名为 PinYin.cs。该类用来定义将汉字转换为拼音的方法，主要代码如下。

```
public string GetABC(string str)
{
    Regex reg = new Regex("^[\u4e00-\u9fa5]$"); //验证输入是否为汉字
    byte[] arr = new byte[2]; //定义字节数组
    string pystr = ""; //定义字符串变量用于返回拼音
    char[] mChar = str.ToCharArray(); //获取汉字对应的字符数组
    return GetStr(mChar,pystr,reg,arr); //返回获取到的汉字拼音
}
```

(4) 在第一个 TextBox 控件的 TextChanged 事件中调用 PinYin 类的 GetABC 自定义方法，实现将汉字转换为拼音的功能，代码如下。

```
private void txt_Chinese_TextChanged(object sender, EventArgs e)
{
    txt_PinYin.Text = new PinYin().GetABC(txt_Chinese.Text); //调用拼音类的 GetABC 方法得到拼音字符串
}
```

**DIY：**制作程序实现获取汉字的区位码功能。提示：可以使用 Encoding 对象的 GetBytes 方法来实现。

**(20 分)** (实例位置：光盘\mr\04\qjyy\02\_diy)

### 4.6.3 情景应用 3——批量替换某一类字符串

视频讲解：光盘\mr\04\lx\批量替换某一类字符串.exe

实例位置：光盘\mr\04\qjyy\03

实际应用中经常遇到批量替换字符串的问题，如 Visual Studio 2008 开发环境中“编辑”菜单下的“全部替换”功能，这里将使用 C# 实现类似的功能。新建一个 Windows 窗体应用程序，在窗体上添加 3 个 TextBox 控件和一个 Button 控件，然后在窗体的代码页中编写如下代码。

```
private void btn_replace_Click(object sender, EventArgs e)
{
    txt_str.Text = txt_str.Text.Replace(txt_find.Text, txt_replace.Text);
    //使用字符串对象的 Replace 方法替换所有满足条件的字符串
}
```

程序运行结果如图 4.15 所示。



图 4.15 批量替换某一类字符串

**DIY：**制作程序实现在字符串中查找指定的子字符串。提示：可以使用 RichTextBox 类的 Find 方法实现。(20 分)(实例位置：光盘\mr\04\qjyy\03\_diy)

#### 4.6.4 情景应用 4——对字符串进行加密与解密

视频讲解：光盘\mr\04\lx\对字符串进行加密与解密.exe

实例位置：光盘\mr\04\qjyy\04

加密是指通过某种特殊的方法更改已有信息的内容，使得未授权的用户即使得到了加密信息，如果没有正确解密的方法，那么也无法得到信息的内容。这里将使用 CryptoStream 类对字符串进行加密和解密，使字符串中的信息更加安全。运行效果如图 4.16 所示。

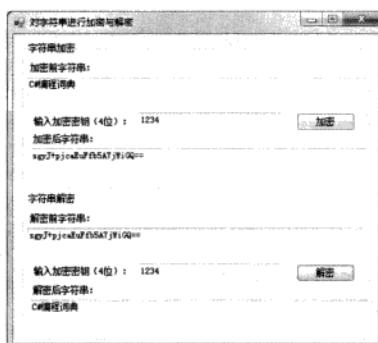


图 4.16 对字符串进行加密与解密

实现过程如下。

- (1) 创建一个 Windows 窗体应用程序，并将其命名为 StringEncrypt。
- (2) 更改默认窗体 Form1 的 Name 属性为 Frm\_Main，向窗体中添加 6 个 TextBox 控件，分别用于显示和输入加密解密信息；添加两个 Button 按钮，分别用于加密和解密字符串。
- (3) 在当前项目中添加一个类，并将其命名为 Encrypt.cs。该类主要用来定义加密字符串和解密字符串的方法，其中，ToEncrypt 方法用来对字符串进行加密；ToDecrypt 方法用来对加密后的字符串进行解密。ToEncrypt 方法实现代码如下。

```
internal string ToEncrypt(string encryptKey, string str)
{
    try
```

```

{
    byte[] P_byte_key = Encoding.Unicode.GetBytes(encryptKey);           //将密钥字符串转换为字节序列
    byte[] P_byte_data = Encoding.Unicode.GetBytes(str);                 //将字符串转换为字节序列
    MemoryStream P_Stream_MS = new MemoryStream();                      //创建内存流对象
    CryptoStream P_CryptStream_Stream =                                //创建加密流对象
        new CryptoStream(P_Stream_MS,new DESCryptoServiceProvider());
        CreateEncryptor(P_byte_key, P_byte_key),CryptoStreamMode.Write);
    P_CryptStream_Stream.Write(P_byte_data, 0, P_byte_data.Length);      //向加密流中写入字节序列
    P_CryptStream_Stream.FlushFinalBlock();                               //将数据写入基础流
    byte[] P_bt_temp = P_Stream_MS.ToArray();                           //从内存流中获取字节序列
    P_CryptStream_Stream.Close();                                       //关闭加密流
    P_Stream_MS.Close();                                              //关闭内存流
    return Convert.ToBase64String(P_bt_temp);                           //方法返回加密后的字符串
}
catch (CryptographicException ce)
{
    throw new Exception(ce.Message);
}
}

```

(4) ToDecrypt 方法实现代码如下。

```

internal string ToDecrypt(string encryptKey, string str)
{
    try
    {
        byte[] P_byte_key = Encoding.Unicode.GetBytes(encryptKey);           //将密钥字符串转换为字节序列
        byte[] P_byte_data = Convert.FromBase64String(str);                 //将加密后的字符串转换为字节序列
        MemoryStream P_Stream_MS = new MemoryStream(P_byte_data);          //创建内存流对象并写入数据
        CryptoStream P_CryptStream_Stream =                                //创建加密流对象
            new CryptoStream(P_Stream_MS,new DESCryptoServiceProvider());
            CreateDecryptor(P_byte_key, P_byte_key),CryptoStreamMode.Read);
        byte[] P_bt_temp = new byte[200];                                     //创建字节序列对象
        MemoryStream P_MemoryStream_temp = new MemoryStream();              //创建内存流对象
        int i = 0;                                                          //创建计数器
        while ((i = P_CryptStream_Stream.Read(P_bt_temp, 0, P_bt_temp.Length)) > 0)
        //使用 while 循环得到解密数据
        {
            P_MemoryStream_temp.Write(P_bt_temp, 0, i);                     //将解密后的数据放入内存流
        }
        return Encoding.Unicode.GetString(P_MemoryStream_temp.ToArray());   //方法返回解密后的字符串
    }
    catch (CryptographicException ce)
    {
        throw new Exception(ce.Message);
    }
}

```

(5) 在 Frm\_Main 窗体的后台代码中单击“加密”或“解密”按钮，分别调用自定义的 ToEncrypt 方法或 ToDecrypt 方法对输入的字符串进行加密或解密，代码如下。

```

private void btn_Encrypt_Click(object sender, EventArgs e)
{
    if (txt_password.Text.Length == 4)                                     //判断加密密钥长度是否正确

```

```

    {
        try
        {
            //调用实例 ToEncrypt 方法得到加密后的字符串
            txt_EncryptStr.Text = new Encrypt().ToEncrypt(txt_password.Text, txt_str.Text);
        }
        catch (Exception ex)                                //捕获异常
        {
            MessageBox.Show(ex.Message);                   //输出异常信息
        }
    }
    else
    {
        MessageBox.Show("密钥长度不符!", "提示");          //提示用户输入密钥长度不正确
    }
}

private void btn_UnEncrypt_Click(object sender, EventArgs e)
{
    if (txt_password2.Text.Length == 4)                //判断加密密钥长度是否正确
    {
        try
        {
            //调用 ToDecrypt 方法得到解密后的字符串
            txt_str2.Text = new Encrypt().ToDecrypt(txt_password2.Text, txt_EncryptStr2.Text);
        }
        catch (Exception ex)                                //捕获异常
        {
            MessageBox.Show(ex.Message);                   //输出异常信息
        }
    }
    else
    {
        MessageBox.Show("密钥长度不符!", "提示");          //提示用户输入密钥长度不正确
    }
}

```

**DIY:** 制作一个程序, 使用异或运算符“^”对数字进行加密与解密。提示: 异或运算符“^”用于比较两个二进制数的相应位。在执行按位“异或”运算时, 如果两个二进制数的相应位都为 1 或两个二进制数的相应位都为 0, 则返回 0, 如果两个二进制数的相应位的其中一个为 1, 另一个为 0, 则返回 1。(20 分)  
**(实例位置: 光盘\mr\04\qjyy\04\_diy)**

#### 4.6.5 情景应用 5——开发一个进制转换器

视频讲解: 光盘\mr\04\lx\开发一个进制转换器.exe

实例位置: 光盘\mr\04\qjyy\05

计算机中的 CPU(中央处理器)有着强大的处理能力, 它处理着大量的二进制信息, 用户可以使用相应的方法将十进制信息转换为二进制信息, 也可以使用相应的方法在二进制、八进制、十进制和十六进制之间方便地进行转换, 这里就实现了类似的功能。运行效果如图 4.17 所示。

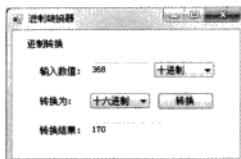


图 4.17 开发一个进制转换器

实现过程如下。

- (1) 创建一个 Windows 窗体应用程序，并将其命名为 Conversion。
- (2) 更改默认窗体 Form1 的 Name 属性为 Frm\_Main，向窗体中添加两个 TextBox 控件，分别用于输入数值和显示转换后数值；添加两个 Combobox 控件，分别用于设置输入数值的进制类型和转换为数值的进制类型；添加一个 Button 按钮，用于进制转换。
- (3) 在 Form1 窗体的后台代码中定义一个无返回值类型的 Action 方法，用来实现进制转换功能，代码如下。

```

private void Action()
{
    if (cbox_select.SelectedIndex != 3) //判断用户输入是否为十六进制数
    {
        long P_lint_value;
        if (long.TryParse(txt_value.Text, out P_lint_value)) //定义长整型变量
        {
            if (cbox_select.SelectedIndex == 0) //判断输入数值是否正确并赋值
            {
                switch (cbox_select2.SelectedIndex)
                {
                    case 0:
                        txt_result.Text = txt_value.Text; //将十进制转为十进制
                        break;
                    case 1:
                        txt_result.Text =
                            new Transform().TenToBinary(long.Parse(txt_value.Text)); //将十进制转为二进制
                        break;
                    case 2:
                        txt_result.Text =
                            new Transform().TenToEight(long.Parse(txt_value.Text)); //将十进制转为八进制
                        break;
                    case 3:
                        txt_result.Text =
                            new Transform().TenToSixteen(long.Parse(txt_value.Text)); //将十进制转为十六进制
                        break;
                }
            }
            else
            {
                if (cbox_select.SelectedIndex == 1) //判断用户输入的是否为二进制数
                {
                    switch (cbox_select2.SelectedIndex)
                    {
                        case 0:
                            txt_result.Text =
                                new Transform().TwoToOne(long.Parse(txt_value.Text)); //将二进制转为十进制
                            break;
                        case 1:
                            txt_result.Text =
                                new Transform().TwoToEight(long.Parse(txt_value.Text)); //将二进制转为八进制
                            break;
                        case 2:
                            txt_result.Text =
                                new Transform().TwoToSixteen(long.Parse(txt_value.Text)); //将二进制转为十六进制
                            break;
                    }
                }
            }
        }
    }
}

```

```

case 0:
    txt_result.Text =
        new Transform().BinaryToTen(long.Parse(txt_value.Text));
    break;
case 1:
    txt_result.Text = txt_value.Text;           //将二进制转为十进制
    break;
case 2:
    txt_result.Text =
        new Transform().BinaryToEight(long.Parse(txt_value.Text));
    break;
case 3:
    txt_result.Text =
        new Transform().BinaryToSixteen(long.Parse(txt_value.Text));
    break;
}
}
else
{
    if (cbox_select.SelectedIndex == 2)           //判断用户输入的是否为八进制数
    {
        switch (cbox_select2.SelectedIndex)
        {
            case 0:
                txt_result.Text =
                    new Transform().EightToTen(long.Parse(txt_value.Text));
                break;
            case 1:
                txt_result.Text =
                    new Transform().EightToBinary(long.Parse(txt_value.Text));
                break;
            case 2:
                txt_result.Text = txt_value.Text;      //将八进制转为八进制
                break;
            case 3:
                txt_result.Text =
                    new Transform().EightToSixteen(long.Parse(txt_value.Text));
                break;
        }
    }
}
else
{
    MessageBox.Show("请输入正确数值!", "提示!"); //提示错误信息
}
}
else
{

```

```

switch (cbox_select2.SelectedIndex)
{
    case 0:
        txt_result.Text = new Transform().SixteenToTen(txt_value.Text); //将十六进制转为十进制
        break;
    case 1:
        txt_result.Text = new Transform().SixteenToBinary(txt_value.Text); //将十六进制转为二进制
        break;
    case 2:
        txt_result.Text = new Transform().SixteenToEight(txt_value.Text); //将十六进制转为八进制
        break;
    case 3:
        txt_result.Text = txt_value.Text; //将十六进制转为十六进制
        break;
}
}
}

```

(4) 单击“转换”按钮，调用自定义方法 Action 实现进制转换功能，代码如下。

```

private void btn_transform_Click(object sender, EventArgs e)
{
    try
    {
        Action(); //调用 Action 方法进行转换操作
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message + " 请重新输入!" "出错!"); //如果出现异常，则提示错误信息
    }
}

```

**DIY：**根据以上程序，实现将指定的十进制数字 368 转换为对应的二进制数。(20 分)(实例位置：光盘\mr\04\qjyy\05\_diy)

情景应用 DIY 栏目分数统计：

DIY 题目	1	2	3	4	5	总分数	
分数							

## 4.7 自我测试

### 一、选择题(每题 10 分, 5 道题)

- 已知 x 为字符变量，则在下列赋值语句中，执行结果与其他 3 个不同的是（ ）。
 

A. x='a';      B. x='\\u0061';      C. x=(char)97;      D. x='\\x0097';
- 有两个 double 类型的变量 x 和 y，分别取值 8.8 和 4.4，则表达式(int)x-y/y 的值是（ ）。
 

A. 7      B. 7.0      C. 7.5      D. 8.0

3. 用户输入两个字符串后，如果想要知道第二个字符串在第一个字符串中的起始位置，应使用 String 类的（ ）方法。  
 A. Substring      B. ToLower      C. IndexOf      D. Insert
4. 如果系统当前日期是 2010 年 1 月 1 日，则表达式 DateTime.Now.AddDays(5).Day 的结果是（ ）。  
 A. 6      B. 5      C. 4      D. 1
5. 下面在字符串中表示回车的符号是（ ）。  
 A. Enter      B. \r      C. \e      D. \n

**二、填空题（每题 10 分，5 道题）**

1. 字符型为一个单 Unicode 字符，一个 Unicode 字符为（ ）位，它可以用来表示世界上的多种语言。
2. 已知大写字母 A 的 ASCII 码是 65，小写字母 a 的 ASCII 码是 97，则十六进制字符常量‘\u0042’表示（ ）。
3. 使用 MessageBox 类可以生成（ ）。
4. 下面代码运行后输出的结果是（ ）。

```
string str = "学通 C# 的 24 堂课";
string str1 = str.Substring(2, 5);
string str2 = str.Substring(5, 2);
string str3 = str.Replace("C#", "CSharp");
string str4 = str3.Substring(8);
Console.WriteLine(str1 + " " + str2 + " " + str4);
```

5. 下面代码运行后输出的结果是（ ）。

```
string str1 = "10";
string str2 = "20";
int result = Convert.ToInt32(str2) - Convert.ToInt32(str1);
if (result == Convert.ToInt32(str1))
    Console.WriteLine(result + Convert.ToInt32(str2));
else
    Console.WriteLine(Convert.ToInt32(str2) - result);
```

**测试分数统计：**

类别	第 1 题	第 2 题	第 3 题	第 4 题	第 5 题
选择题分数					
填空题分数					
总分数					

**4.8 行动指南**

开始日期：\_\_\_\_年\_\_\_\_月\_\_\_\_日

结束日期：\_\_\_\_年\_\_\_\_月\_\_\_\_日

序号	内 容	行动指南	
1	照猫画虎栏目	分数>75 分	优秀，基本功掌握得不错，加油！
		75 分>分数>50 分	及格，知识掌握得不牢，重新做一遍照猫画虎。
		分数<50 分	请使用光盘中提供的“实战能力测试系统”进行提高训练。

续表

序号	内 容	行 动 指 南	
1	情景应用栏目 分数( )	分数>75 分	优秀，综合应用能力很强。
		75 分>分数>50 分	及格，综合应用能力需提高，再练习一遍情景应用。
	自我测试栏目 分数( )	分数<50 分	请使用光盘中提供的“实战能力测试系统”进行提高训练。
		分数>75 分	优秀，有成为编程高手的潜质。
	综合评价	请使用光盘中提供的“实战能力测试系统”进行提高训练。 反复训练后，以上各项分数都在 75 分以上，方可进入下一堂课学习。	
2	编程能力培养：本栏目提供了一些实践题目，对于培养编程能力很有效，要做好。	1. 开发一个程序，用来将长字符串进行分段显示。 2. 开发一个程序，用来获取输入汉字的区位码。 3. 开发一个判断用户登录身份的程序。提示：使用 switch 语句并结合 goto 实现。	
3	编程习惯培养：本堂课培养读者在学习开发中记笔记的习惯，把开发中遇到的问题、总结的经验记录下来。		
4	创新能力培养：看看身边有没有可以用编程解决的问题，记录到右边的表格中。根据学习进度，尝试编写解决这些问题的小程序。		

## 4.9 成功可以复制——“杀毒王”王江民不可思议的传奇人生

王江民，1951 年出生于上海，3 岁时因患小儿麻痹后遗症而腿部残疾。初中毕业后，他回到老家山东烟台，从一名街道工厂的学徒工干起，刻苦自学，成长为拥有各种创造发明 20 多项的机械和光电类专家。作为所在国营企业的技术骨干，他先后被授予“全国新长征突击手标兵”、“全国青年自学成才标兵”称号。

改变王江民人生的阶段是 1987 年，那时王江民是所在国营企业的技术骨干。当时的国内计算机发展还处于预热期，但经常与高科打交道的王江民认定计算机必将伴随高科技的发展改变人类社会的生活。“如果不学，很可能跟不上科技发展的步伐。于是先买书，晚上熬夜，每天一点点地看”，开始了一个崭新领域的理论与实践探索之路。王江民说：“我 38 岁开始学计算机，没有感觉我老了，没有感觉我不行，只感到我的英语基础不好。再说，计算机是实践性非常强的学科。我搞计算机是用计算机，不是学计算机。”



机遇总是偏爱有准备肯钻研的人，1989年，王江民花1000多元自己买了一台中华学习机，第二年又买了一台8088PC机。王江民首先学的是Basic语言。当时，王江民的孩子正上小学一年级，王江民在辅导孩子功课时突发奇想，能否编一个程序代替家长辅导。没过多久，3个月前尚对软件领域一片空白的王江民就编好了一套数学语文教学软件，试过后发现效果奇佳，拿到电脑报参加了一个评奖，结果被誉为“教育软件第一”，并被第一个在全国隆重推广。王江民从一开始就是在用计算机，而不是在学计算机。

软件获奖激发了王江民对软件业的极大兴趣，使他从此把精力转到了软件领域。在开发工控软件时，王江民发现一些或明或暗的病毒。王江民先是用Debug手工杀病毒，接着是写一段程序杀一种病毒。王江民有一个很好的习惯，就是杀一种病毒就在报刊上发表一篇文章，公布这段杀病毒的程序。后来，王江民觉得这些各自独立的杀病毒程序用起来很麻烦，就把6个杀不同病毒的程序集成到了一起，命名为KV6，后来发展到KV8、KV12、KV18、KV20。随着时间的推移，KV系列杀毒软件在国内市场中一路高奏凯歌，先后囊括16项大奖，成为杀毒软件行业中无可置疑的领头羊。依靠KV系列软件，王江民也成为中关村的亿万富翁，并跻身“中国IT富豪榜50强”，成为新世纪“知识英雄”的典范，创造了中国软件行业的奇迹。



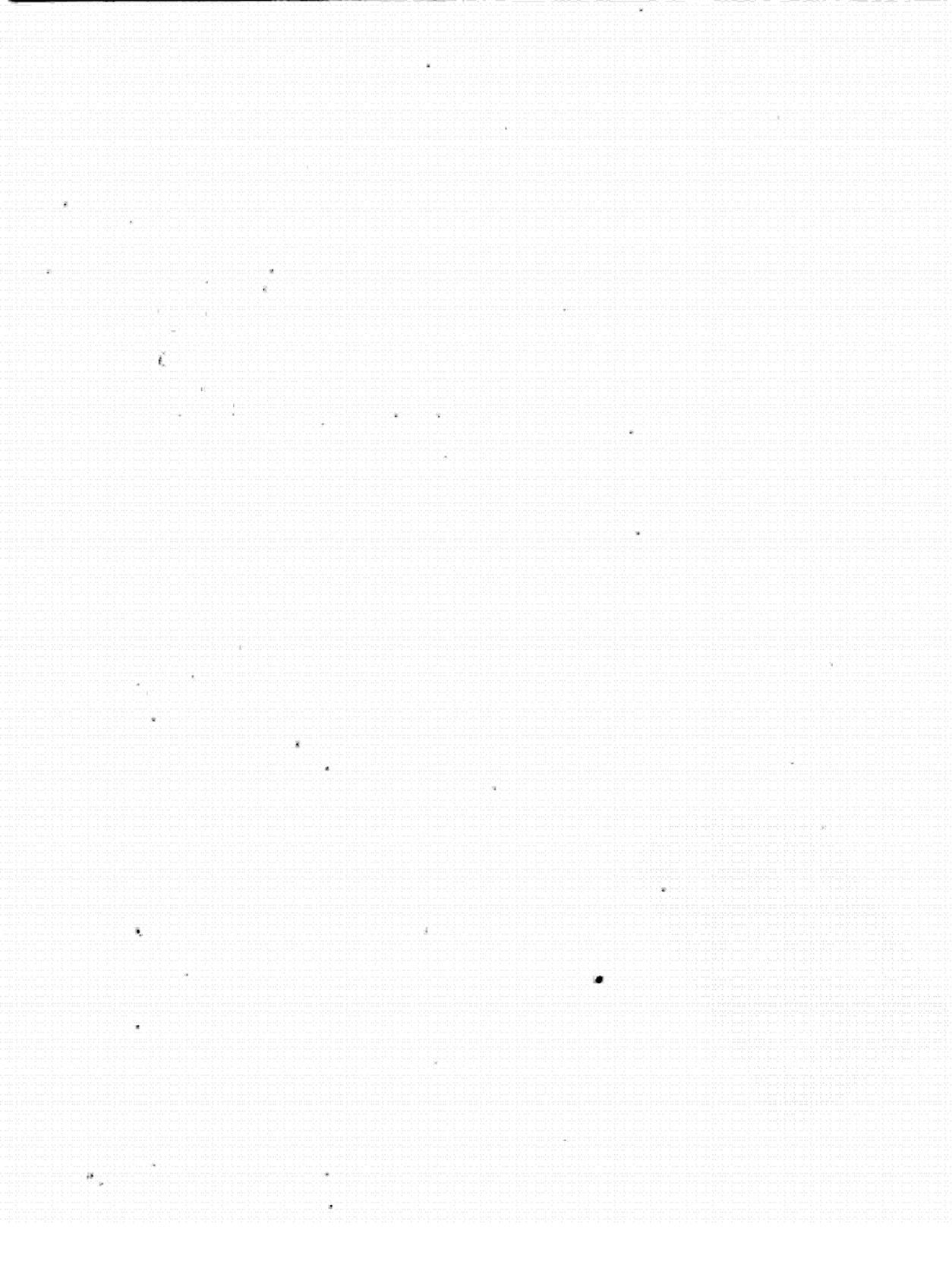
### 经典语录

我38岁开始学计算机，没有感觉我老了，没有感觉我不行，只感到我的英语基础不好。再说，计算机是实践性非常强的学科。我搞计算机是用计算机，不是学计算机。



### 深度评价

王江民的成功启示我们，每一个想追求成功和传奇的人都会在他身上看到自己的希望和信心，因为王江民各方面的起点都非常低，低到在外人看来凭着王江民的外在条件，他根本就没有任何成功的可能性。后来者们只要少一些浮躁，多一些热情和专注，多一些勤奋和执著，就一定能发现机会，并抓住机遇，书写一段属于你自己的编程传奇。



# 第 5 堂课

## 数组与集合

( 视频讲解：139分钟)

在 C# 程序设计中，数组与字符串一样也是最常用的类型之一，数组能够按一定规律把相关的数据组织在一起，并能通过“索引”或“下标”快速地管理这些数据；另外，C# 中还提供了 ArrayList 类用来表示集合，它也可以存储多个数据，本堂课也将对其进行详细讲解。

学习摘要：

- » 了解数组的基本概念
- » 掌握一维数组和二维数组的声明及使用
- » 掌握动态数组的声明及使用
- » 掌握数组的各种基本操作
- » 掌握 ArrayList 集合类的使用及操作

## 5.1 数组概述

数组是大部分编程语言均支持的一种数据类型，无论是 C 语言、C++、Java 还是 C# 均支持数组的概念。

数组是包含若干相同类型的变量，这些变量都可以通过索引进行访问。数组中的变量称为数组的元素，数组能够容纳元素的数量称为数组的长度。数组中的每个元素都具有唯一的索引与其相对应，其索引从零开始。

数组是通过指定数组的元素类型、数组的秩（维数）及数组每个维度的上限和下限来定义的，即一个数组的定义需要包含以下几个要素。

- 元素类型。
- 数组的维数。
- 每个维数的上下限。

数组的元素表示某一种确定的类型，如整数或字符串等，那么数组的确切含义是什么呢？数组类型值是对象，数组对象被定义为存储数组元素类型值的一系列位置。也就是说，数组是一个存储一系列元素位置的对象。数组中存储位置的数量由数组的秩和边界来确定。

数组类型是从抽象基类型 `Array` 派生的引用类型，通过 `new` 运算符创建数组并将数组元素初始化为它们的默认值。数组可以分为一维数组、二维数组和多维数组等，图 5.1 中演示了几种基本的数组，包括一维数组、二维数组及多维数组等。

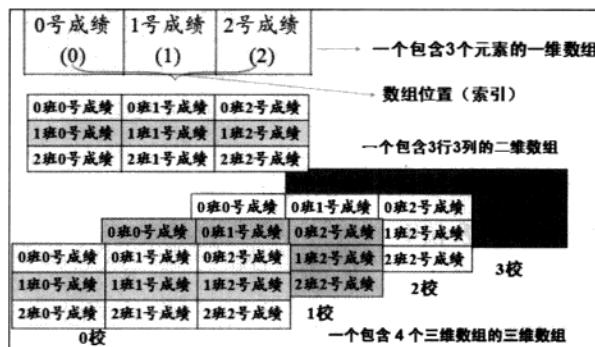


图 5.1 常见的几种数组

 **说明：**多维数组的使用与二维数组类似，实际上，二维数组也是一种简单的多维数组。

## 5.2 一维数组的声明和使用

一维数组是最常用的数组类型，本节将详细讲解一维数组的声明和使用方法。

### 5.2.1 一维数组的声明

一维数组即数组的维数为 1。

## 1. 声明

一维数组的声明语法如下。

语法: type[] arrayName;

说明: type 表示数组存储数据的数据类型; arrayName 表示数组名称。

**例 5.01** 声明一个 int 类型的一维数组 arr, 代码如下。

```
int[] arr;
```

**说明:** 需要注意的是, 数组的长度不是声明的一部分, 而且数组必须在访问前初始化。数组的类型可以是基本数据类型, 也可是枚举或其他类型。

## 2. 初始化

数组的初始化有多种形式, 开发人员可以通过 new 运算符创建数组并将数组元素初始化为它们的默认值。

**例 5.02** 声明一个 int 类型的一维数组 arr, 该数组中包含 5 个元素, 同时对该数组进行初始化, 代码如下。

```
int[] arr = new int[5];
```

可以在声明数组时将其初始化, 并且初始化的值为用户自定义的值。

**例 5.03** 声明一个 int 类型的一维数组 arr, 该数组中包含 5 个元素, 同时对该数组进行初始化, 代码如下。

```
int[] arr = new int[5]{1,2,3,4,5};
```

**说明:** 数组大小必须与大括号中的元素数量相匹配, 否则会产生编译错误。

可以声明一个数组变量时不对其初始化, 但在对数组初始化时必须使用 new 运算符。

**例 5.04** 声明一个 string 类型的一维数组 arrStr, 然后使用 new 运算符对其进行初始化, 代码如下。

```
string[] arrStr;
```

```
arrStr = new string[7]{"Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat"};
```

实际上, 在初始化数组时可以省略 new 运算符和数组的长度, 编译器将根据初始值的数量来计算数组长度, 并创建数组。

**例 5.05** 在声明一维数组 arrStr 时不使用 new 运算符, 而是直接对该数组进行初始化, 代码如下。

```
string[] arrStr = {"Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat"};
```

### 5.2.2 一维数组的使用

在需要存储多个值时可以使用一维数组, 而且可以通过使用 foreach 语句或数组的下标将数组中的元素值读出来。图 5.2 举例说明了包括 5 个元素的一维数组。

**例 5.06** 创建一个控制台应用程序, 其中定义了一个一维数组, 并通过使用 foreach 语句显示该数组的内容, 代码如下。(实例位置: 光盘\mr\05\s\5.06)

```
static void Main(string[] args)
{
    int[] arr = { 1, 2, 3, 4, 5 }; // 定义一个一维数组并为其赋值
    foreach (int n in arr) // 使用 foreach 语句循环遍历一维数组中的元素
        Console.WriteLine("{0}", n + " ");
    Console.ReadLine();
}
```

程序运行结果如图 5.3 所示。

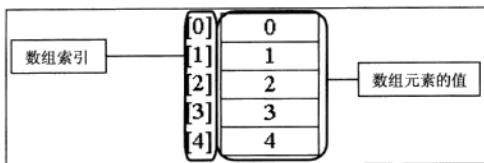


图 5.2 存储 5 个元素的一维数组结构



图 5.3 一维数组实例运行结果图

**说明:** foreach 语句声明一个迭代变量，其自动获取数组中每个元素的值，它是遍历一个数组的首选方式。如果要以其他方式遍历数组或修改数组中的元素，那么应采用 for 语句，因为 foreach 语句中的迭代变量是数组中每个元素的只读副本。

## 5.3 二维数组的声明和使用

二维数组即数组的维数为 2，它相当于一个表格。本节将对二维数组的声明和使用方法进行详细讲解。

### 5.3.1 二维数组的声明

#### 1. 声明

二维数组的声明语法如下。

语法：type[,] arrayName;

说明：type 表示数组存储数据的数据类型，arrayName 表示数组名称。

**例 5.07** 声明一个两行两列的二维数组，代码如下。

```
int[,] arr=new int[2,2];
```

#### 2. 初始化

二维数组的初始化有两种形式，可以通过 new 运算符创建数组并将数组元素初始化为它们的默认值。

**例 5.08** 声明一个两行两列的二维数组，同时使用 new 运算符对其进行初始化，代码如下。

```
int[,] arr=new int[2,2]{{1,2},{3,4}};
```

也可以在初始化数组时，不指定行数和列数，而是使用编译器根据初始值的数量来自动计算数组的行数和列数。

**例 5.09** 声明一个二维数组，声明时不指定该数组的行数和列数，然后使用 new 运算符对其进行初始化，代码如下。

```
int[,] arr=new int[]{{1,2},{3,4}};
```

### 5.3.2 二维数组的使用

当需要存储表格的数据时可以使用二维数组。如图 5.4 所示为 4 行 3 列的二维数组的存储结构。

**例 5.10** 创建一个控制台应用程序，其中定义了一个静态的二维数组，并使用数组的 GetLength 方法获取数组的行数和列数，然后通过遍历数组输出其元素值。代码如下。（实例位置：光盘\mr\05\sl\5.10）

```
static void Main(string[] args)
{
    int[,] arr = new int[3, 2] {{ 1, 2 }, { 3, 4 }, { 5, 6 }}; //自定义一个二维数组
```

```

Console.Write("数组的行数为: ");
Console.WriteLine(arr.GetLength(0));
Console.WriteLine("");
Console.Write("数组的列数为: ");
Console.WriteLine(arr.GetLength(1));
Console.WriteLine("");
for (int i = 0; i < arr.GetLength(0); i++)
{
    string str = "";
    for (int j = 0; j < arr.GetLength(1); j++)
    {
        str = str + Convert.ToString(arr[i, j]) + " ";
    }
    Console.WriteLine(str);
    Console.WriteLine("");
}
Console.ReadLine();
}

```

程序运行结果如图 5.5 所示。

数组索引	[0,0]	[0,1]	[0,2]
	[1,0]	[1,1]	[1,2]
	[2,0]	[2,1]	[2,2]
	[3,0]	[3,1]	[3,2]

图 5.4 二维数组的存储结构

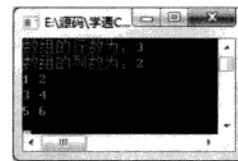


图 5.5 二维数组实例运行结果图

### 5.3.3 动态数组的声明及使用

动态数组的声明实际上就是将数组的定义部分和初始化部分分别写在不同的语句中，动态数组的初始化也需要使用 new 关键字为数组元素分配内存空间，并为数组元素赋初值。

动态数组的声明语法如下。

语法： type[] arrayName;

arrayName = new type[n1,n2,⋯]

说明： type 表示数组存储数据的数据类型； arrayName 表示数组名称； n1,n2 表示数组的长度，它们可以是整型常量或变量，分别表示一维数组和二维数组的长度； new 关键字仍然以默认值来初始化数组元素。

**例 5.11** 声明一个动态的二维数组，代码如下。

```

int m=2;
int n=2;
int[,] arr2 = new int[m,n];

```

说明：这里的 m 和 n 可以为任意整型类型的值。

**例 5.12** 创建一个控制台应用程序，其中将用户输入的行数和列数作为动态数组的行数和列数，然后将数组的行索引和列索引以字符串的形式合并在一起，作为动态数组的元素值输出。代码如下。（**实例位置：**光盘\mr\05\s\5.12）

```

static void Main(string[] args)
{
    Console.Write("请输入动态数组的行数: ");
    int row = Convert.ToInt32(Console.ReadLine());
    //定义动态数组的行数
    Console.Write("请输入动态数组的列数: ");
    int col = Convert.ToInt32(Console.ReadLine());
    //定义动态数组的列数
    int[,] arr = new int[row, col];
    //根据定义的行数和列数定义动态数组
    Console.WriteLine("结果: ");
    for (int i = 0; i < row; i++)
    {
        for (int j = 0; j < col; j++)
        {
            Console.Write(i + j.ToString() + " ");
            //输出动态数组中的元素
        }
        Console.WriteLine();
    }
    Console.ReadLine();
}

```

程序运行结果如图 5.6 所示。



图 5.6 动态数组实例运行结果图

## 5.4 数组的基本操作

C#中的数组是由 System.Array 类派生而来的引用对象，因此可以使用 Array 类中的各种方法对数组进行各种操作。对数组的操作可以分为静态操作和动态操作，静态操作主要包括查找、遍历等；动态操作主要包括插入、删除、合并和拆分等。本节将对数组的各种操作进行详细讲解。

### 5.4.1 遍历数组中的元素

使用 foreach 语句可以实现数组的遍历功能，开发人员可以使用该语句访问数组中的每个元素，而不需要确切地知道每个元素的索引号。

**例 5.13** 下面声明一个 int 类型的一维数组，该数组中包含 5 个元素，然后使用 foreach 语句遍历该数组中的元素，代码如下。

```

int[] arr = new int[10] { 10, 20, 30, 40, 50 };
//采用 foreach 语句对 arr 数组进行遍历
foreach (int number in arr)
{
    Console.WriteLine(number);
    Console.ReadLine();
}

```

### 5.4.2 添加和删除数组元素

添加和删除数组元素就是在数组中的指定位置对数组元素进行添加和删除操作。添加数组元素一般是通过使用 `ArrayList` 集合实现，该类将在 5.5 节中进行详细讲解。

开发人员可以利用数组的索引号对数组元素进行删除操作，但这种方法不能够真正实现对数组元素的删除，一般不推荐使用。

**例 5.14** 下面声明一个 `int` 类型的一维数组，然后使用索引号来删除数组中的第二个元素，代码如下。

```
int[] arr = new int[] { 1, 3, 5, 7, 9 };
//删除第二个数组元素
int n=1;
for (int i = n; i < arr.Length - n ; i++)
    arr[i] = arr[i + 1];
```

 **说明：**以上方法虽然删除了 `arr` 数组中的第二个数组元素，但数组的最后两位数组元素值却是相同的，它并没有完成真正地从数组中删除数组元素的操作。可以用 `ArrayList` 集合来真正地完成数组的删除操作，该集合将在 5.5 节中进行详细讲解。

### 5.4.3 数组的合并与拆分

数组的合并与拆分在很多情况下都会被使用，在对数组进行合并或拆分时，数组与数组之间的类型应一致。下面对数组的合并及拆分进行详细讲解。

#### 1. 数组的合并

数组的合并实际上就是将多个一维数组合并成一个一维数组，或者将多个一维数组合并成一个二维数组或多维数组。

**例 5.15** 创建一个控制台应用程序，首先将两个一维数组合并成一个新的二维数组，然后再将定义的两个一维数组合并为一个新的二维数组。代码如下。（实例位置：光盘\mr\05\s\5.15）

```
static void Main(string[] args)
{
    //定义两个一维数组
    int[] arr1 = new int[] { 1, 2, 3, 4, 5 };
    int[] arr2 = new int[] { 6, 7, 8, 9, 10 };
    int n = arr1.Length + arr2.Length;
    //根据定义的两个一维数组的长度定义一个新的二维数组
    int[,] arr3 = new int[n,];
    //将定义的两个一维数组中的元素添加到新的二维数组中
    for (int i = 0; i < arr3.Length; i++)
    {
        if (i < arr1.Length)
            arr3[i] = arr1[i];
        else
            arr3[i] = arr2[i - arr1.Length];
    }
    Console.WriteLine("合并后的一维数组：");
    foreach (int i in arr3)
```

```

Console.Write("{0}", i + " ");
Console.WriteLine();
//定义一个要合并的二维数组
int[,] arr4 = new int[2, 5];
//将两个一维数组循环添加到二维数组中
for (int i = 0; i < arr4.GetLength(0); i++)
{
    switch (i)
    {
        case 0:
        {
            for (int j = 0; j < arr1.Length; j++)
                arr4[i, j] = arr1[j];
            break;
        }
        case 1:
        {
            for (int j = 0; j < arr2.Length; j++)
                arr4[i, j] = arr2[j];
            break;
        }
    }
}
Console.WriteLine("合并后的二维数组: ");
//显示合并后的二维数组
for (int i = 0; i < arr4.GetLength(0); i++)
{
    for (int j = 0; j < arr4.GetLength(1); j++)
        Console.Write(arr4[i, j] + " ");
    Console.WriteLine();
}

}
Console.ReadLine();
}

```

程序运行结果如图 5.7 所示。

## 2. 数组的拆分

数组的拆分实际上就是将一个一维数组拆分成多个一维数组，或是将多维数组拆分成多个一维数组或多个多维数组。

**例 5.16** 创建一个控制台应用程序，首先将两个一维数组合并成一个新的二维数组，然后再将定义的两个一维数组合并为一个新的二维数组。代码如下。（实例位置：光盘\mr\05\sl\5.16）

```

static void Main(string[] args)
{
    int[,] arr1 = new int[2, 3] {{ 1, 3, 5 }, { 2, 4, 6 }}; //定义一个二维数组并赋值
    //定义两个一维数组，用来存储拆分的二维数组中的元素
    int[] arr2 = new int[3];
    int[] arr3 = new int[3];
    for (int i = 0; i < 2; i++)
    {

```

```

for (int j = 0; j < 3; j++)
{
    switch (i)
    {
        case 0: arr2[j] = arr1[i, j]; break; //判断如果是第一行中的元素，则添加到第一个一维数组中
        case 1: arr3[j] = arr1[i, j]; break; //判断如果是第二行中的元素，则添加到第二个一维数组中
    }
}
Console.WriteLine("数组一: ");
foreach (int n in arr2) //输出拆分后的第一个一维数组
    Console.Write(n + " ");
Console.WriteLine();
Console.WriteLine("数组二: ");
foreach (int n in arr3) //输出拆分后的第二个一维数组
    Console.Write(n + " ");
Console.ReadLine();
}

```

程序运行结果如图 5.8 所示。



图 5.7 数组合并实例运行结果图



图 5.8 数组拆分实例运行结果图

## 5.5 ArrayList 集合的使用

在面向对象程序设计中，集合类是一种将各相同类型的对象集合起来的类，数组实质上也是集合类型中的一种，从数据结构来讲，集合主要是以线性结构存储数据。C#中提供了几种集合类，如 ArrayList 类、Queue 类、Stack 类等，本节主要对 ArrayList 集合类进行详细讲解。

### 5.5.1 ArrayList 集合概述

ArrayList 类位于 System.Collections 命名空间下，它可以动态地添加和删除元素。ArrayList 类相当于一种高级的动态数组，它是 Array 类的升级版本，但它并不等同于数组。

与数组相比，ArrayList 类为开发人员提供了以下功能。

- 数组的容量是固定的，而 ArrayList 的容量可以根据需要自动扩充。
- ArrayList 提供添加、删除和插入某一范围元素的方法，但在数组中只能一次获取或设置一个元素的值。
- ArrayList 提供将只读和固定大小包装返回到集合的方法，而数组不提供。
- ArrayList 只能是一维形式，而数组可以是多维的。

ArrayList 提供了 3 个构造器，通过这 3 个构造器可以有 3 种声明方式，下面分别介绍。

(1) 默认的构造器，将会以默认的大小来初始化内部的数组。

语法：public ArrayList();

通过以上构造器声明 ArrayList 的语法格式如下。

语法：ArrayList List = new ArrayList();

说明：List 表示 ArrayList 对象名。

**例 5.17** 声明一个 ArrayList 对象，并为其添加 10 个 int 类型的元素值，代码如下。

```
ArrayList List = new ArrayList();
```

```
for (int i = 0; i < 10; i++)
```

//给 ArrayList 对象添加 10 个 int 类型元素

```
List.Add(i);
```

(2) 用一个 ICollection 对象来构造，并将该集合的元素添加到 ArrayList 中。

语法：public ArrayList(ICollection);

通过以上构造器声明 ArrayList 的语法格式如下。

语法：ArrayList List = new ArrayList(arryName);

说明：List 表示 ArrayList 对象名。arryName 表示要添加集合的数组名。

**例 5.18** 声明一个 int 类型的一维数组，然后声明一个 ArrayList 对象，同时将已经声明的一维数组中的元素添加到该对象中，代码如下。

```
int[] arr = new int[] { 1, 2, 3, 4, 5, 6, 7, 8, 9 };
```

```
ArrayList List = new ArrayList(arr);
```

//将声明的一维数组添加到 ArrayList 集合中

(3) 用指定的大小初始化内部的数组。

语法：public ArrayList(int);

通过以上构造器声明 ArrayList 的语法格式如下。

语法：ArrayList List = new ArrayList(n);

说明：List 表示 ArrayList 对象名；n 表示 ArrayList 对象的空间大小。

**例 5.19** 声明一个具有 10 个元素的 ArrayList 对象，并为其赋初始值，代码如下。

```
ArrayList List = new ArrayList(10);
```

```
for (int i = 0; i < List.Count; i++)
```

//为 ArrayList 对象添加 10 个 int 类型元素

```
List.Add(i);
```

ArrayList 集合类的常用属性及说明如表 5.1 所示。

表 5.1 ArrayList 集合类的常用属性及说明

属性	说明
Capacity	获取或设置 ArrayList 可包含的元素数
Count	获取 ArrayList 中实际包含的元素数
IsFixedSize	获取一个值，该值指示 ArrayList 是否具有固定大小
IsReadOnly	获取一个值，该值指示 ArrayList 是否为只读
IsSynchronized	获取一个值，该值指示是否同步对 ArrayList 的访问
Item	获取或设置指定索引处的元素
SyncRoot	获取可用于同步 ArrayList 访问的对象

## 5.5.2 添加 ArrayList 集合元素

向 ArrayList 集合中添加元素时，可以使用 ArrayList 类提供的 Add 方法和 Insert 方法，下面对这两种方

法进行详细介绍。

### 1. Add 方法

将对象添加到 ArrayList 集合的结尾处。

语法：public virtual int Add(Object value)

说明：value 表示要添加到 ArrayList 的末尾处的 Object，该值可以为空引用。其返回值为 ArrayList 索引，表示已在此处添加了 value。

**例 5.20** 声明一个包含 6 个元素的一维数组，并使用该数组实例化一个 ArrayList 对象，然后使用 Add 方法为该 ArrayList 对象添加元素，代码如下。

```
int[] arr = new int[] { 1, 2, 3, 4, 5, 6 };
ArrayList List = new ArrayList(arr);           //使用声明的一维数组实例化一个 ArrayList 对象
List.Add(7);                                  //为 ArrayList 对象添加元素
```

### 2. Insert 方法

将元素插入 ArrayList 集合的指定索引处。

语法：public virtual void Insert(int index, Object value)

说明：index 表示从零开始的索引，应在该位置插入 value；value 表示要插入的 Object，该值可以为空引用。

**例 5.21** 声明一个包含 6 个元素的一维数组，并使用该数组实例化一个 ArrayList 对象，然后使用 Insert 方法在该 ArrayList 对象的指定索引处添加一个元素，代码如下。

```
int[] arr = new int[] { 1, 2, 3, 4, 5, 6 };
ArrayList List = new ArrayList(arr);           //使用声明的一维数组实例化一个 ArrayList 对象
List.Insert(3, 7);                            //在 ArrayList 集合的指定位置添加一个元素
```

**例 5.22** 创建一个控制台应用程序，其中定义了一个 int 类型的一维数组，并使用该数组实例化一个 ArrayList 对象，然后分别使用 ArrayList 对象的 Add 方法和 Insert 方法向 ArrayList 集合的结尾处和指定索引处添加元素。代码如下。（实例位置：光盘\mr\05\s\5.22）

```
static void Main(string[] args)
{
    int[] arr = new int[] { 1, 2, 3, 4, 5, 6 };
    ArrayList List = new ArrayList(arr);           //使用声明的一维数组实例化一个 ArrayList 对象
    Console.WriteLine("原 ArrayList 集合：");
    foreach (int i in List)                      //遍历 ArrayList 集合并输出
    {
        Console.Write(i + " ");
    }
    Console.WriteLine();
    for (int i = 1; i < 5; i++)
    {
        List.Add(i + arr.Length);                //为 ArrayList 集合添加元素
    }
    Console.WriteLine("使用 Add 方法添加：");
    foreach (int i in List)                      //遍历添加元素后的 ArrayList 集合并输出
    {
        Console.Write(i + " ");
    }
    Console.WriteLine();
```

```

List.Insert(6, 6); //在 ArrayList 集合的指定位置添加元素
Console.WriteLine("使用 Insert 方法添加: ");
foreach (int i in List) //遍历最后的 ArrayList 集合并输出
{
    Console.Write(i + " ");
}
Console.ReadLine();
}

```

程序运行结果如图 5.9 所示。



图 5.9 ArrayList 元素的添加实例运行结果图

**说明:** 在 ArrayList 集合中插入一个数组时, 只需要在上面代码中重新声明一个一维数组, 并将 List.Insert(6, 6)语句改为 List.InsertRange(6,一维数组名)即可。

**注意:** 当程序中使用 ArrayList 类时, 需要在命名空间区域添加 using System.Collections;, 下面将不再提示。

### 5.5.3 删 除 ArrayList 集合元素

删除 ArrayList 集合中的元素时, 可以使用 ArrayList 类提供的 Clear 方法、Remove 方法、RemoveAt 方法和 RemoveRange 方法, 下面对这 4 种方法进行详细介绍。

#### 1. Clear 方法

从 ArrayList 中移除所有元素。

语法: public virtual void Clear()

**例 5.23** 声明一个包含 6 个元素的一维数组, 并使用该数组实例化一个 ArrayList 对象, 然后使用 Clear 方法清除 ArrayList 中的所有元素, 代码如下。

```

int[] arr = new int[] { 1, 2, 3, 4, 5, 6 };
ArrayList List = new ArrayList(arr);
List.Clear();

```

#### 2. Remove 方法

从 ArrayList 中移除特定对象的第一个匹配项。

语法: public virtual void Remove(Object obj)

说明: obj 表示要从 ArrayList 移除的 Object, 该值可以为空引用。

**例 5.24** 声明一个包含 6 个元素的一维数组, 并使用该数组实例化一个 ArrayList 对象, 然后使用 Remove 方法从声明的 ArrayList 对象中移除与“3”匹配的元素, 代码如下。

```

int[] arr = new int[] { 1, 2, 3, 4, 5, 6 };
ArrayList List = new ArrayList(arr);
List.Remove(3);

```

### 3. RemoveAt 方法

移除 ArrayList 的指定索引处的元素。

语法： public virtual void RemoveAt(int index)

说明： index 表示要移除的元素的从零开始的索引。

**例 5.25** 声明一个包含 6 个元素的一维数组，并使用该数组实例化一个 ArrayList 对象，然后使用 RemoveAt 方法从声明的 ArrayList 对象中移除索引为 3 的元素，代码如下。

```
int[] arr = new int[] { 1, 2, 3, 4, 5, 6 };
ArrayList List = new ArrayList(arr);
List.RemoveAt(3);
```

### 4. RemoveRange 方法

从 ArrayList 中移除一定范围的元素。

语法： public virtual void RemoveRange(int index,int count)

说明： index 表示要移除的元素的范围从零开始的起始索引； count 表示要移除的元素数。

**例 5.26** 声明一个包含 6 个元素的一维数组，并使用该数组实例化一个 ArrayList 对象，然后在该 ArrayList 对象中使用 RemoveRange 方法从索引 3 处删除两个元素，代码如下。

```
int[] arr = new int[] { 1, 2, 3, 4, 5, 6 };
ArrayList List = new ArrayList(arr);
List.RemoveRange(3,2);
```

**例 5.27** 创建一个控制台应用程序，使用 RemoveRange 方法删除 ArrayList 集合中的一批元素，代码如下。（实例位置：光盘\mr\05\sl\5.27）

```
static void Main(string[] args)
{
    int[] arr = new int[] { 1, 2, 3, 4, 5, 6, 7, 8, 9 };  

    ArrayList List = new ArrayList(arr);  

    Console.WriteLine("原 ArrayList 集合： ");  

    foreach (int i in List)  

        Console.WriteLine(i.ToString());  

    List.RemoveRange(0, 5);  

    Console.WriteLine("删除元素后的 ArrayList 集合： ");  

    foreach (int i in List)  

        Console.WriteLine(i.ToString());
}
```

程序运行结果如图 5.10 所示。

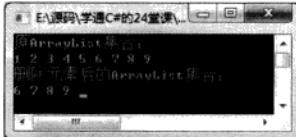


图 5.10 ArrayList 元素的删除实例运行结果图

### 5.5.4 遍历 ArrayList 集合

ArrayList 集合的遍历与数组类似，两者均可使用 foreach 语句，下面通过一个实例说明如何遍历 ArrayList 集合中的元素。

**例 5.28** 创建一个控制台应用程序，其中实例化了一个 ArrayList 对象，并使用 Add 方法向 ArrayList 集合中添加了两个元素，然后使用 foreach 语句遍历 ArrayList 集合中的各个元素并输出。代码如下。（实例位置：光盘\mr\05\s\5.28）

```
static void Main(string[] args)
{
    ArrayList list = new ArrayList();           //实例化一个 ArrayList 对象
    list.Add("C#编程词典");                    //向 ArrayList 集合中添加元素
    list.Add("学通 C# 的 24 堂课");
    foreach (string str in list)               //遍历 ArrayList 集合中的元素并输出
    {
        Console.WriteLine(str + " ");
    }
    Console.ReadLine();
}
```

程序运行结果为：C#编程词典 学通 C# 的 24 堂课。

### 5.5.5 查找 ArrayList 集合元素

在查找 ArrayList 集合中的元素时，可以使用 ArrayList 类提供的 Contains 方法、IndexOf 方法和 LastIndexOf 方法，下面对这 3 种方法进行详细介绍。

#### 1. Contains 方法

确定某元素是否在 ArrayList 集合中。

语法：public virtual bool Contains(Object item)

说明：item 表示要在 ArrayList 中查找的 Object，该值可以为空引用。其返回值表示：如果在 ArrayList 中找到 item，则为 True；否则为 False。

**例 5.29** 声明一个包含 6 个元素的一维数组，并使用该数组实例化一个 ArrayList 对象，然后使用 Contains 方法判断数字 2 是否在 ArrayList 集合中。代码如下。

```
int[] arr = new int[] { 1, 2, 3, 4, 5, 6 };
ArrayList list = new ArrayList(arr);
Console.WriteLine(list.Contains(2));           //判断 ArrayList 集合中是否包含指定的元素
```

程序运行结果为：True。

#### 2. IndexOf 方法

搜索指定的 Object，并返回整个 ArrayList 中第一个匹配项的从零开始的索引。

语法：public virtual int IndexOf(Object value)

说明：value 表示要在 ArrayList 中查找的 Object，该值可以为空引用。其返回值表示：如果在整个 ArrayList 中找到 value 的第一个匹配项，则为该项的从零开始的索引；否则为 -1。

**例 5.30** 声明一个包含 6 个元素的一维数组，并使用该数组实例化一个 ArrayList 对象，然后使用 IndexOf

方法在 ArrayList 集合中顺序查找数字 2 的索引位置，代码如下。

```
int[] arr = new int[] { 1, 2, 3, 4, 5, 6 };
ArrayList List = new ArrayList(arr);
Console.WriteLine(List.IndexOf(2)); //顺序查找数字 2 的索引位置
```

程序运行结果为：1。

### 3. LastIndexOf 方法

搜索指定的 Object，并返回整个 ArrayList 中最后一个匹配项的从零开始的索引。

语法：public virtual int LastIndexOf(Object value)

说明：value 表示要在 ArrayList 中查找的 Object，该值可以为空引用。其返回值表示：如果在整个 ArrayList 中找到 value 的最后一个匹配项，则为该项的从零开始的索引；否则为 -1。

**例 5.31** 声明一个包含 6 个元素的一维数组，并使用该数组实例化一个 ArrayList 对象，然后使用 LastIndexOf 方法在 ArrayList 集合中倒序查找数字 2 的索引位置。代码如下。

```
int[] arr = new int[] { 1, 2, 3, 4, 5, 6 };
ArrayList List = new ArrayList(arr);
Console.WriteLine(List.LastIndexOf(2)); //倒序查找数字 2 的索引位置
```

程序运行结果为：4。

## 5.6 照猫画虎——基本功训练

### 5.6.1 基本功训练 1——获取多维数组的行数与列数

■ 视频讲解：光盘\mr\05\lx\获取多维数组的行数与列数.exe

■ 实例位置：光盘\mr\05\zmhh\01

新建一个 Windows 窗体应用程序，在窗体中添加一个 Button 控件，用来生成随机数组；添加一个 TextBox 控件，用来显示生成的随机数组；添加一个 Label 控件，用来显示生成数组的行数和列数，然后在窗体的代码页中编写如下代码。

```
private string[,] G_str_array; //定义数组类型变量
private Random G_Random_Num = new Random(); //生成随机数对象
private void btn_GetArray_Click(object sender, EventArgs e)
{
    txt_display.Clear(); //清空控件中的字符串
    G_str_array = new string[ //随机生成二维数组
        G_Random_Num.Next(2, 10), G_Random_Num.Next(2, 10)];
    lab_Message.Text = string.Format(
        "生成了 {0} 行 {1} 列 的数组",
        G_str_array.GetUpperBound(0) + 1, //获取数组的行数
        G_str_array.GetUpperBound(1) + 1); //获取数组的列数
    DisplayArray(); //调用显示数组方法
}
private void DisplayArray()
{
    for (int i = 0; i < G_str_array.GetUpperBound(0) + 1; i++) //使用循环赋值
    {
        for (int j = 0; j < G_str_array.GetUpperBound(1) + 1; j++)
            txt_display.Text += G_str_array[i, j] + " ";
    }
}
```

```

        for (int j = 0; j < G_str_array.GetUpperBound(1) + 1; j++)
    {
        G_str_array[i, j] = i.ToString() + "," + j.ToString() + " ";
    }
}
for (int i = 0; i < G_str_array.GetUpperBound(0) + 1; i++) //使用循环输出
{
    for (int j = 0; j < G_str_array.GetUpperBound(1) + 1; j++)
    {
        txt_display.Text += G_str_array[i, j];
    }
    txt_display.Text += Environment.NewLine;
}
}

```

程序运行结果如图 5.11 所示。

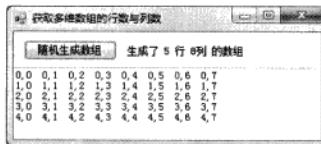


图 5.11 获取多维数组的行数与列数

**照猫画虎：**根据以上程序，制作一个获取数组元素个数的程序。提示：使用数组的 Length 属性实现。  
**(20 分)** (实例位置：光盘\mr\05\zmhh\01\_zmhh)

## 5.6.2 基本功训练 2——按指定条件在数组中检索元素

■**视频讲解：**光盘\mr\05\lx\按指定条件在数组中检索元素.exe

■**实例位置：**光盘\mr\05\zmhh\02

新建一个 Windows 窗体应用程序，在窗体中添加一个 Label 控件，用来显示原始数组；添加两个 TextBox 控件，分别用来输入查询条件和显示查询结果，然后在窗体的代码页中编写如下代码。

```

private string[] G_str_array; //定义字符串数组字段
private void Frm_Main_Load(object sender, EventArgs e)
{
    G_str_array = new string[] { //为字符串数组字段赋值
        "明日科技", "C#编程词典", "学通 C# 的 24 堂课", "C#范例宝典"};
    for (int i = 0; i < G_str_array.Length; i++) //循环输出字符串
    {
        lab_Message.Text += G_str_array[i] + "\n";
    }
}
private void txt_find_TextChanged(object sender, EventArgs e) //判断查找字符串是否为空
{
    if (txt_find.Text != string.Empty) //使用 FindAll 方法查找相应字符串
    {
        string[] P_str_temp = Array.FindAll
            (G_str_array, (s) => s.Contains(txt_find.Text));
    }
}

```

```

if (P_str_temp.Length > 0) //判断是否查找到相应字符串
{
    txt_display.Clear(); //清空控件中的字符串
    foreach (string s in P_str_temp)
    {
        txt_display.Text += s + Environment.NewLine;
    }
}
else
{
    txt_display.Clear(); //清空控件中的字符串
    txt_display.Text = "没有找到记录"; //提示没有找到记录
}
}
else
{
    txt_display.Clear(); //清空控件中的字符串
}
}

```

程序运行结果如图 5.12 所示。

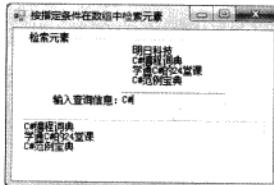


图 5.12 按指定条件在数组中检索元素

**熊猫画虎：**制作一个可以遍历所有数组元素的程序。提示：使用 `for` 语句实现。（20 分）（实例位置：光盘\mr\05\zmhh\02\_zmhh）

### 5.6.3 基本功训练 3——在数组中添加一个元素

■**视频讲解：**光盘\mr\05\lx\在数组中添加一个元素.exe

■**实例位置：**光盘\mr\05\zmhh\03

新建一个 Windows 窗体应用程序，在窗体中添加两个 TextBox 控件，分别用来显示随机生成的数组和输入要插入的数组元素；添加两个 Button 控件，分别用来执行随机生成数组操作和向数组中插入元素操作；添加一个 RichTextBox 控件，用来显示插入元素后的新数组，然后在窗体的代码页中编写如下代码。

```

private int[] G_int_array = new int[8]; //定义数组类型变量
///<summary>
///增加单个数组元素
///</summary>
///<param name="ArrayBorn">要向其中添加元素的一维数组</param>
///<param name="Index">添加索引</param>
///<param name="Value">添加值</param>
///<returns></returns>

```

```

static int[] AddArray(int[] ArrayBorn, int Index, int Value)
{
    if (Index >= (ArrayBorn.Length))
        Index = ArrayBorn.Length - 1;
    int[] TemArray = new int[ArrayBorn.Length + 1];
    for (int i = 0; i < TemArray.Length; i++)
    {
        if (Index >= 0)
            if (i < (Index + 1))
                TemArray[i] = ArrayBorn[i];
            else if (i == (Index + 1))
                TemArray[i] = Value;
            else
                TemArray[i] = ArrayBorn[i - 1];
        else
            if (i == 0)
                TemArray[i] = Value;
            else
                TemArray[i] = ArrayBorn[i - 1];
    }
    return TemArray;
}
private void btn_RArray_Click(object sender, EventArgs e)
{
    txt_RArray.Clear(); //清空文本框
    for (int i = 0; i < G_int_array.GetUpperBound(0) + 1; i++)
    {
        G_int_array[i] = i;
    }
    for (int i = 0; i < G_int_array.GetUpperBound(0) + 1; i++) //使用循环输出
    {
        txt_RArray.Text += G_int_array[i] + " ";
    }
}
private void btn_Sure_Click(object sender, EventArgs e)
{
    rtbox_NArray.Clear(); //清空文本框
    G_int_array = AddArray(G_int_array, 4, Convert.ToInt32(txt_Element.Text));
    //调用自定义方法向数组中插入单个元素
    for (int i = 0; i < G_int_array.GetUpperBound(0) + 1; i++) //使用循环输出
    {
        rtbox_NArray.Text += G_int_array[i] + " ";
    }
}

```

程序运行结果如图 5.13 所示。



图 5.13 在数组中添加一个元素

**熊猫画虎:** 根据以上程序, 实现在数组中添加一个数组的功能。(20分)(实例位置: 光盘\mr\05\zmhh\03\_zmhh)

#### 5.6.4 基本功训练4——不改变长度删除数组中的元素

视频讲解: 光盘\mr\05\lx\不改变长度删除数组中的元素.exe

实例位置: 光盘\mr\05\zmhh\04

新建一个 Windows 窗体应用程序, 在窗体中添加 3 个 TextBox 控件, 分别用来显示随机生成的数组和输入开始删除的索引、要删除的元素个数; 添加两个 Button 控件, 分别用来执行随机生成数组操作和删除数组中元素操作; 添加一个 RichTextBox 控件, 用来显示删除元素后的新数组, 然后在窗体的代码页中编写如下代码。

```

private int[] G_int_array = new int[8]; //定义数组类型变量
///<summary>
///删除数组中的元素
///</summary>
///<param name="ArrayBorn">要从中删除元素的数组</param>
///<param name="Index">删除索引</param>
///<param name="Len">删除的长度</param>
static void DeleteArray(int[] ArrayBorn, int Index, int Len)
{
    if (Len <= 0) //判断删除长度是否小于等于 0
        return; //返回
    if (Index == 0 && Len >= ArrayBorn.Length) //判断删除长度是否超出了数组范围
        Len = ArrayBorn.Length; //将删除长度设置为数组的长度
    else if ((Index + Len) >= ArrayBorn.Length) //判断删除索引和长度的和是否超出了数组范围
        Len = ArrayBorn.Length - Index - 1; //设置删除的长度
    int i = 0; //定义一个 int 型变量, 用来标识开始遍历的位置
    for (i = 0; i < ArrayBorn.Length - Index - Len; i++) //遍历删除的长度
        ArrayBorn[i + Index] = ArrayBorn[i + Len + Index]; //覆盖要删除的值
    //遍历删除长度后面的数组元素值
    for (int j = (ArrayBorn.Length - 1); j > (ArrayBorn.Length - Len - 1); j--) //设置数组为 0
        ArrayBorn[j] = 0;
}
private void btn_RArray_Click(object sender, EventArgs e)
{
    txt_RArray.Clear(); //清空文本框
    for (int i = 0; i < G_int_array.GetUpperBound(0) + 1; i++) //使用循环赋值
    {
        G_int_array[i] = i;
    }
    for (int i = 0; i < G_int_array.GetUpperBound(0) + 1; i++) //使用循环输出

```

```

    {
        txt_RArray.Text += G_int_array[i] + " ";
    }
}
private void btn_Sure_Click(object sender, EventArgs e)
{
    rtbox_NArray.Clear(); //清空文本框
    //删除数组中的元素
    DeleteArray(G_int_array, Convert.ToInt32(txt_Index.Text), Convert.ToInt32(txt_Num.Text));
    for (int i = 0; i < G_int_array.GetUpperBound(0) + 1; i++) //使用循环输出删除元素的数组
    {
        rtbox_NArray.Text += G_int_array[i] + " ";
    }
}

```

程序运行结果如图 5.14 所示。

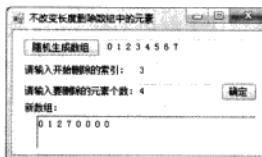


图 5.14 不改变长度删除数组中的元素

**照猫画虎：**根据以上程序，将删除后的数组元素用数字 98 替代。(20 分)(实例位置：光盘\mr\05\zmhh\04\_zmhh)

## 5.6.5 基本功训练 5——删除数组元素后改变其长度

视频讲解：光盘\mr\05\lx\删除数组元素后改变其长度.exe

实例位置：光盘\mr\05\zmhh\05

新建一个 Windows 窗体应用程序，在窗体中添加 3 个 TextBox 控件，分别用来显示随机生成的数组和输入开始删除的索引、要删除的元素个数；添加两个 Button 控件，分别用来执行随机生成数组操作和删除数组中元素操作；添加一个 RichTextBox 控件，用来显示删除元素后的新数组，然后在窗体的代码页中编写如下代码。

```

private int[] G_int_array = new int[8]; //定义数组类型变量
///<summary>
///删除数组中的元素，并改变数组的长度
///</summary>
///<param name="ArrayBorn">要从中删除元素的数组</param>
///<param name="Index">删除索引</param>
///<param name="Len">删除的长度</param>
///<returns>得到的新数组</returns>
static int[] DeleteArray(int[] ArrayBorn, int Index, int Len)
{
    if (Len <= 0) //判断删除长度是否小于等于 0
        return ArrayBorn; //返回源数组
    if (Index == 0 && Len >= ArrayBorn.Length) //判断删除长度是否超出了数组范围
        Len = ArrayBorn.Length; //将删除长度设置为数组的长度
}

```

```

else if ((Index + Len) >= ArrayBorn.Length)           //判断删除索引和长度的和是否超出了数组范围
    Len = ArrayBorn.Length - Index - 1;                //设置删除的长度
int[] temArray = new int[ArrayBorn.Length - Len];      //声明一个新的数组
for (int i = 0; i < temArray.Length; i++)             //遍历新数组
{
    if (i >= Index)
        temArray[i] = ArrayBorn[i + Len];
    else
        temArray[i] = ArrayBorn[i];
}
return temArray;                                       //返回得到的新数组
}

private void btn_RArray_Click(object sender, EventArgs e)
{
    txt_RArray.Clear();                                //清空文本框
    for (int i = 0; i < G_int_array.GetUpperBound(0) + 1; i++) //使用循环赋值
    {
        G_int_array[i] = i;
    }
    for (int i = 0; i < G_int_array.GetUpperBound(0) + 1; i++) //使用循环输出
    {
        txt_RArray.Text += G_int_array[i] + " ";
    }
}
private void btn_Sure_Click(object sender, EventArgs e)
{
    rtbox_NArray.Clear();                            //清空文本框
    //删除数组中的元素
    G_int_array = DeleteArray(G_int_array, Convert.ToInt32(txt_Index.Text), Convert.ToInt32(txt_Num.Text));
    for (int i = 0; i < G_int_array.GetUpperBound(0) + 1; i++) //使用循环输出删除元素的数组
    {
        rtbox_NArray.Text += G_int_array[i] + " ";
    }
}

```

程序运行结果如图 5.15 所示。

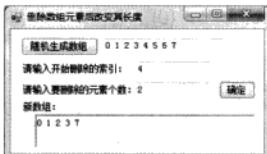


图 5.15 删除数组元素后改变其长度

**照猫画虎：**编写一个程序，使用 ArrayList 类的 RemoveRange 方法实现与以上程序相同的功能。(20 分)(实例位置：光盘\mr\05\zmhh\05\_zmhh)

照猫画虎栏目分数统计：

照猫画虎题目	1	2	3	4	5	总分数	
分数							

## 5.7 情景应用——拓展与实践

### 5.7.1 情景应用 1——操作便捷的简单电话簿

视频讲解：光盘\mr\05\lx\操作便捷的简单电话簿.exe

实例位置：光盘\mr\05\qjyy\01

使用集合制作一个简单的电话簿，新建一个控制台应用程序，然后在 Program.cs 文件中编写如下代码。

```

class program
{
    static void Main()
    {
        Console.Title = "简单电话簿"; //设置控制台标题
        ArrayList al = new ArrayList(); //创建集合对象
        string name = string.Empty; //定义字符串变量，用于存放姓名
        string phone = string.Empty; //定义字符串变量，用于存放电话号码
        while (name != "q") //开始 while 循环
        {
            Console.Write("请输入姓名(f 查看, q 退出): "); //提示用户输入姓名
            name = Console.ReadLine(); //等待用户输入姓名
            if (name == "q") break; //判断是否退出
            if (name != "f") //判断是否查看电话簿
            {
                Console.Write("请输入电话: "); //提示用户输入电话号码
                phone = Console.ReadLine(); //等待用户输入电话号码
                al.Add(new student() { Name = name, Phone = phone });//向集合中添加信息
            }
            else
            {
                Console.Clear(); //清空控制台
                Console.WriteLine("电话号码簿"); //输出字符串
                foreach (object o in al) //遍历集合输出电话信息
                {
                    Console.WriteLine("姓名: {0} 电话: {1}", ((student)o).Name, ((student)o).Phone);
                }
                Console.WriteLine(); //输出空行
            }
        }
        Console.Clear(); //清空控制台
        Console.WriteLine("欢迎使用本系统！"); //输出欢迎信息
        Console.ReadLine(); //等待回车继续
    }
}

class student //定义 student 类
{
    public string Name; //定义 Name 字段
    public string Phone; //定义 Phone 字段
}

```

程序运行结果如图 5.16 所示。



图 5.16 操作便捷的简单电话簿

**DIY:** 根据以上程序制作一个新的程序, 用来使用 ArrayList 集合记录用户的姓名和性别。(20 分)(实例位置: 光盘\mr\05\qjyy\01\_diy )

## 5.7.2 情景应用 2——使用数组解决约瑟夫环问题

视频讲解: 光盘\mr\05\lx\使用数组解决约瑟夫环问题.exe

实例位置: 光盘\mr\05\qjyy\02

约瑟夫环问题即设有 n 个人坐成一个圈, 从某个人开始报数, 数到 m 的人出列, 接着从出列的下一个人开始重新报数, 数到 m 的人再次出列, 如此反复循环, 直到所有人都出列为止, 最后按出列顺序输出。新建一个控制台应用程序, 然后在 Program.cs 文件中编写如下代码。

```
static int[] Jose(int total, int start, int alter)      //约瑟夫环问题算法
{
    int j, k = 0;
    int[] intCounts = new int[total + 1];           //intCounts 数组存储按出列顺序组成的数据, 以作为结果返回
    int[] intPers = new int[total + 1];           //intPers 数组存储初始数据
    //对数组 intPers 赋初值, 第一个人序号为 0, 第二个人为 1, 依次类推
    for (int i = 0; i < total; i++)
    {
        intPers[i] = i;
    }
    //按出列次序依次存于数组 intCounts 中
    for (int i = total; i >= 2; i--)
    {
        start = (start + alter - 1) % i;
        if (start == 0)
            start = i;
        intCounts[k] = intPers[start];
        k++;
        for (j = start + 1; j <= i; j++)
            intPers[j - 1] = intPers[j];
    }
    intCounts[k] = intPers[1];
}

return intCounts;                                //结果返回
}
#endregion
static void Main(string[] args)
```

```

{
    int[] intPers = Jose(12, 3, 4); //调用自定义方法解决约瑟夫环问题
    Console.WriteLine("约瑟夫环出列顺序: ");
    for (int i = 0; i < intPers.Length; i++)
    {
        Console.Write(intPers[i] + " "); //输出出列顺序
    }
    Console.ReadLine();
}

```

程序运行结果如图 5.17 所示。

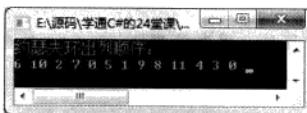


图 5.17 使用数组解决约瑟夫环问题

**DIY：**对一维数组使用鸡尾酒排序算法进行排序。提示：鸡尾酒排序，也称双向冒泡排序、搅拌排序或涟漪排序，使用鸡尾酒排序时，首先对要排序的内容从低到高排序，然后再从高到低排序。(20分)(实例位置：光盘\mr\05\qjyy\02\_diy)

### 5.7.3 情景应用 3——向班级集合中添加学生信息

■ 视频讲解：光盘\mr\05\lx\向班级集合中添加学生信息.exe

■ 实例位置：光盘\mr\05\qjyy\03

集合在程序开发中经常用到，如可以将学生信息、商品信息等存储到集合中，以便随时更新，这里将使用 ArrayList 集合实现存储学生信息的功能。新建一个 Windows 窗体应用程序，在窗体中添加一个 DataGridView 控件，用来显示 ArrayList 集合中存储的学生信息，然后在窗体的代码页中编写如下代码。

```

private void Frm_Main_Load(object sender, EventArgs e)
{
    ArrayList P_list_StudentInfo = new ArrayList(); //实例化集合对象
    string[] P_str_Students, P_str_Info; //定义两个字符串数组，分别用来记录学生整体信息和分解后的信息
    string P_str_Student = ""; //定义一个字符串变量，用来记录所有学生信息
    //向集合中添加学生信息
    P_list_StudentInfo.Add("小王 男 1980-01-01");
    P_list_StudentInfo.Add("小刘 女 1981-01-01");
    P_list_StudentInfo.Add("小赵 男 1990-01-01");
    P_list_StudentInfo.Add("小吕 男 1995-01-01");
    P_list_StudentInfo.Add("小梁 男 2000-01-01");
    foreach (string Pc_str_Student in P_list_StudentInfo) //遍历集合
    {
        P_str_Student += Pc_str_Student + ","; //记录所有学生信息
    }
    P_str_Students = P_str_Student.Split(','); //将学生信息存储在一个字符串数组中
    dgv_Info.Rows.Add(5); //为 DataGridView 控件添加 5 行
    for (int i = 0; i < P_str_Students.Length - 1; i++) //遍历存储学生整体信息的数组
    {
        P_str_Info = P_str_Students[i].Split(' ');
        dgv_Info.Rows[i].Cells[0].Value = P_str_Info[0]; //将学生整体信息进行分解
        dgv_Info.Rows[i].Cells[1].Value = P_str_Info[1]; //显示学生姓名
        dgv_Info.Rows[i].Cells[2].Value = P_str_Info[2];
    }
}

```

```

        dgv_Info.Rows[i].Cells[1].Value = P_str_Info[1]; //显示性别
        dgv_Info.Rows[i].Cells[2].Value = P_str_Info[2]; //显示出生年月
    }
}

```

程序运行结果如图 5.18 所示。



图 5.18 向班级集合中添加学生信息

**DIY：**在以上程序的基础上，将学生的出生年月显示为“\*\*\*\*年\*\*月\*\*日”这种格式。提示：可以使用 string 类的 Format 方法实现。(20 分)(实例位置：光盘\mr\05\qjyy\03\_diy)

#### 5.7.4 情景应用 4——使用哈希表对 XML 文件进行查询

■**视频讲解：**光盘\mr\05\lx\使用哈希表对 XML 文件进行查询.exe

■**实例位置：**光盘\mr\05\qjyy\04

在开发网络电台应用程序时，可以将网络电台的地址及名称存放到 XML 文件中，这时如果需要将 XML 文件中存储的所有电台地址及对应名称显示出来，就可以使用哈希表来实现。运行效果如图 5.19 所示。



图 5.19 使用哈希表对 XML 文件进行查询

实现过程如下。

- (1) 创建一个 Windows 窗体应用程序，并将其命名为 SelectXMLByHasTable。
- (2) 更改默认窗体 Form1 的 Name 属性为 Frm\_Main，并在窗体中添加两个 ComboBox 控件，分别用来显示 XML 文件中存储的网络电台名称和电台网址。
- (3) 在窗体的代码页中自定义一个 SelectXML 方法，该方法的返回值类型为 Hashtable，它主要用来从指定的 XML 文件中获取信息，并存储到 Hashtable 哈希表中。SelectXML 方法实现代码如下。

```

#region 在 XML 文件中查找电台地址及名称
///<summary>
///在 XML 文件中查找电台地址及名称
///</summary>
///<param name="strPath">XML 文件路径</param>
///<returns>Hashtable 对象，用来记录找到的电台地址及名称</returns>
static Hashtable SelectXML(string strPath)
{
    Hashtable HTable = new Hashtable(); //实例化哈希表对象
    XmlDocument doc = new XmlDocument(); //实例化 XML 文档对象
    doc.Load(strPath); //加载 XML 文档
    XmlNodeList xnl = doc.SelectSingleNode("BCastInfo").ChildNodes;//获取 NewDataSet 节点的所有子节点
}

```

```

string strVersion = "";
string strInfo = "";
foreach (XmlNode xn in xnl)
{
    XElement xe = (XElement)xn;
    if (xe.Name == "DInfo")
    {
        XmlNodeList xnChild = xe.ChildNodes;
        foreach (XmlNode xnChild in xnChild)
        {
            XElement xeChild = ( XElement)xnChild;
            if (xeChild.Name == "Address")
            {
                strVersion = xeChild.InnerText;           //记录电台地址
            }
            if (xeChild.Name == "Name")
            {
                strInfo = xeChild.InnerText;           //记录电台名称
            }
        }
        HTable.Add(strVersion, strInfo);           //向哈希表中添加键值
    }
}
return HTable;
}
#endregion

```

(4) Frm\_Main 窗体加载时，调用自定义的 SelectXML 方法获取 XML 文件中的信息，并存储到 Hashtable 哈希表中，然后对 Hashtable 哈希表进行遍历，并将遍历到的信息添加到相应的下拉列表中。代码如下。

```

private void Frm_Main_Load(object sender, EventArgs e)
{
    Hashtable myHashtable = SelectXML("BroadCastInfo.xml");           //使用自定义方法实例化哈希表对象
    IDictionaryEnumerator IDEnumerator = myHashtable.GetEnumerator();   //循环访问哈希表
    while (IDenumerator.MoveNext())
    {
        cbox_Name.Items.Add(IDenumerator.Value.ToString());           //显示电台名称
        cbox_NetAddress.Items.Add(IDenumerator.Key.ToString());         //显示电台网址
    }
    cbox_Name.SelectedIndex = cbox_NetAddress.SelectedIndex = 0;          //设置默认选项
}

```

**DIY：**在以上程序的基础上实现选择“电台名称”时，在“电台网址”下拉列表中自动显示对应的电台网址。(20 分)(实例位置：光盘\mr\05\qjyy\04\_diy)

## 5.7.5 情景应用 5——设计一个简单客车售票记录程序

视频讲解：光盘\mr\05\lx\设计一个简单客车售票记录程序.exe

实例位置：光盘\mr\05\qjyy\05

假设客车的座位数是 9 行 4 列，读者可以使用二维数组，在控制台应用程序中实现简单客车售票记录。新建一个控制台应用程序，然后在 Program.cs 文件中编写如下代码。

```

static void Main()
{
    Console.Title = "简单客车售票记录";
    string[,] zuo = new string[9, 4];
    for (int i = 0; i < 9; i++)
    {
        for (int j = 0; j < 4; j++)
        {
            zuo[i, j] = "【 空 】";
        }
    }
    string s = string.Empty;
    while (true)
    {
        System.Console.Clear();
        Console.WriteLine("简单客车售票记录" + "\n");
        for (int i = 0; i < 9; i++)
        {
            for (int j = 0; j < 4; j++)
            {
                System.Console.Write(zuo[i, j]);
            }
            System.Console.WriteLine();
        }
        //提示用户输入信息
        System.Console.Write("请输入座位行号和列号(如: 0,2)输入 q 键退出: ");
        s = System.Console.ReadLine();
        if (s == "q") break;
        string[] ss = s.Split(',');
        int one = int.Parse(ss[0]);
        int two = int.Parse(ss[1]);
        zuo[one, two] = "【 已售 】";
    }
}

```

//入口方法  
//设置控制台标题  
//定义二维数组  
//for 循环开始  
//for 循环开始  
//初始化二维数组  
//定义字符串变量  
//开始售票  
//清空控制台信息  
//输出字符串  
//输出售票信息  
//输出换行符  
//售票信息输入  
//输入字符串“q”退出系统  
//拆分字符串  
//得到座位行数  
//得到座位列数  
//标记售出票状态

程序运行结果如图 5.20 所示。



图 5.20 设计一个简单客车售票记录程序

**DIY：**在以上程序的基础上，将所有的行号和列号位置处都设置为“【 已售 】” 提示：在遍历输出每个行列位置之前为其重新赋值。(20 分)(实例位置：光盘\mr\05\qjyy\05\_diy)

## 情景应用 DIY 栏目分数统计：

DIY 题目	1	2	3	4	5	总分数	
分数							

## 5.8 自我测试

## 一、选择题（每题 10 分，5 道题）

1. 以下的数组声明语句中，正确的是（ ）。  
 A. int a[3];      B. int [3] a;      C. int[][] a=new int[][],;      D. int [] a={1,2,3};
2. 可以用来遍历数组或集合中所有元素的循环是（ ）。  
 A. while      B. do…while      C. foreach      D. if
3. 已知 int[][] arr=new int [3][] {new int[3]{5,6,2},new int[5]{6,9,7,8,3},new int[2]{3,2}}; 则 arr[2][2] 的值是（ ）。  
 A. 9      B. 1      C. 6      D. 越界
4. 下面代码运行后输出的结果是（ ）。  
 int[] names = new int[]{4,3,2,1};  
 Array.Sort(names,1,3);  
 foreach(int name in names)  
 {  
 Console.WriteLine(name);  
 }  
 A. 4321      B. 4123      C. 1234      D. 3214

5. 下面代码运行后输出的结果是（ ）。  
 int []num = new int[5]{1,3,2,0,0};  
 Array.Reverse(num);  
 foreach(int l in num)  
 {  
 Console.WriteLine(l);  
 }  
 A. 00123      B. 12300      C. 00132      D. 00231

## 二、填空题（每题 10 分，5 道题）

1. 数组的数组称为（ ）。  
 2. 如果二维数组 a 有 6 列，则在 a[1,5]前面有（ ）个元素。  
 3. 下面代码运行后输出的结果是（ ）。  
 ArrayList arrNumber=new ArrayList();  
 arrNumber.Capacity=2;  
 for(int i=0;i<5;i++)  
 {  
 arrNumber.Add(i);  
 }  
 Console.WriteLine(arrNumber.Count);

4. 下面代码运行后输出的结果是（ ）。

```
int []age=new int[]{16,18,20,14,22};
foreach(int i in age)
{
    if(i>18)
        continue;
    Console.WriteLine(i.ToString()+" ");
}
```

5. 下面代码运行后输出的结果是（ ）。

```
Hashtable hsStu=new Hashtable();
hsStu.Add(3,"甲");
hsStu.Add(2,"乙");
hsStu.Add(1,"丙");
Console.WriteLine(hsStu[3]);
```

测试分数统计：

类别	第1题	第2题	第3题	第4题	第5题
选择题分数					
填空题分数					
总分数					

## 5.9 行动指南

开始日期： 年 月 日      结束日期： 年 月 日

序号	内 容	行动指南	
1	照猫画虎栏目	分数>75 分	优秀，基本功掌握得不错，加油！
		75 分>分数>50 分	及格，知识掌握得不牢，重新做一遍照猫画虎。
	情景应用栏目	分数<50 分	请使用光盘中提供的“实战能力测试系统”进行提高训练。
		分数>75 分	优秀，综合应用能力很强。
	分数（ ）	75 分>分数>50 分	及格，综合应用能力需提高，再练一遍情景应用。
		分数<50 分	请使用光盘中提供的“实战能力测试系统”进行提高训练。
	自我测试栏目	分数>75 分	优秀，有成为编程高手的潜质。
		分数<75 分	请使用光盘中提供的“实战能力测试系统”，进行提高训练。
	综合评价	反复训练后，以上各项分数都在 75 分以上，方可进入下一堂课学习。	
2	编程能力培养：本栏目提供了一些实践题目，对于培养编程能力很有效，要做好。	1. 开发一个程序，主要使用冒泡排序法对一维数组进行排序。 2. 开发一个软件版本更新程序，要求使用哈希表记录软件的版本号及更新的功能。 3. 开发一个程序，用来计算矩形矩阵的乘积，提示：在执行两个矩阵的乘法运算时，需要将前面矩阵的第 i 行与后面矩阵的第 j 列对应的元素相乘，然后再相加，最后将得到的结果放到结果矩阵的第(i,j)这个位置上即可。	
3	编程习惯培养：本堂课培养读者在学习开发中记笔记的习惯，把开发中遇到的问题、总结的经验记录下来。		

续表

序号	内 容	行 动 指 南
4	创新能力培养：看看身边有没有可以用编程解决的问题，记录到右边的表格中。根据学习进度，尝试编写解决这些问题的小程序。	

## 5.10 成功可以复制——善于抓住时机的人徐少春

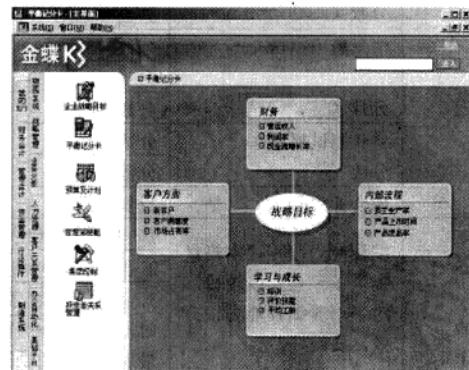
徐少春年轻时家境贫寒，以至于高中时只能用咸菜萝卜、萝卜咸菜的无穷反复来填饱自己的肚子。但他决心要改变现状，甚至他在课桌的左右两角分别刻下“金钱在向你呼唤”、“美人在向你招手”，当然这也是那时大多数学生求学的动力。通过刻苦努力学习，徐少春最终考上了大学，并读完了研究生。

1988 年，徐少春被分配到地方税务局工作。1990 年，徐少春放弃了人人羡慕的铁饭碗，来到改革开放的最前沿、创业的热土深圳，进了一家会计师事务所担任电脑部经理。但就是电脑部经理这个职务，让徐少春更加清楚地知道，中国是多么需要财务管理软件。徐少春决定抓住这个时机，于是他辞掉电脑部经理的工作，并向其岳父借了 5000 元买了一台 286 电脑，创办深圳爱普电脑技术有限公司，开始编写财务软件。那以后的两年间，徐少春带着独立开发的“爱普财务软件”，叩开一家又一家公司的门，亲自演示自己的软件产品。几个月后，他的用户增加到 30 多家，包括许多知名公司。凭借顽强的意志和坚定的信念，徐少春取得了最初的胜利。后来，徐少春先生与美籍华人赵西燕女士以及深圳市蛇口工业区社会保险公司合资成立金蝶软件科技（深圳）有限公司，至此金蝶软件的名称才得以问世。

20 世纪 90 年代中期，当时 PC 机的作业系统还是 DOS 版的天下，Windows 平台的财务软体还少有人问津，甚至许多人都认为 Windows 时代还有 2~3 年才会到来。但徐少春又一次感受到 Windows 时代是大势所趋、不久将至，一定要抓住这个时机。于是在 1994 年，徐少春就开始带领金蝶研发 Windows 版财务软件，他和 20 多个开发人员春节都没有休息，到 1995 年初徐少春就拿出了金蝶财务软件 For Windows 1.0 版。1996 年 4 月，金蝶 Windows 版财务软件被中国软件评测中心确认为中国首家优秀级 Windows 版财务软件。

20 世纪 90 年代末，在中国的南方，生产制造型企业不断兴起，企业的生产加工急需要软件来管理。在深圳的徐少春早就闻到企业 ERP 的气息，他又没错过这次时机，在 1999 年金蝶软件实现从财务软件向 ERP 软件的战略转移，这是一次革命性的转折点，为其以后成为中国软件产业的领导厂商奠定了基础。到了 2001 年，金蝶国际软件集团有限公司顺利在香港交易所挂牌上市。

2007 年，随着电子商务的不断兴起，徐少春又开始实施金蝶软件“全面进军全程电子商务的计划”，



于是推出了在线记账及商务管理平台——“友商网”。徐少春经过多年的拼搏与奋斗，时至今日，他所领导的金蝶国际软件集团有限公司已成为亚太地区领先的企业管理软件及电子商务应用解决方案供应商。

### ✓ 经典语录

在我的视线中，金蝶是不可挑战的。

### ✓ 深度评价

从徐少春的成功历程中可以看出，他的成功之处在于很好地抓住每一次时机，这最终使金蝶在众多的财务软件公司中成为佼佼者。所以我们大家在学习和工作中，也要善于抓住各种时机，使我们个人的事业迈向更大的成功。



# 第 6 堂课

## 程序设计中的算法

( 视频讲解：57分钟)

做任何事都有一定的步骤。例如，你想乘飞机去外地开会，首先需要购买机票，然后按时去飞机场，登上飞机，到达目标城市后，再乘车抵达会场参加会议。这些步骤都是按一定的顺序进行的，缺一不可，甚至顺序错了也不行。程序设计也是如此，而在进行程序设计前，还需要知道“应该做什么”和“怎么做”，算法就是解决“应该做什么”和“怎么做”的问题。因此，本堂课将对程序设计中最常用到的算法进行详细讲解。

学习摘要：

- ▶ 了解什么是算法
- ▶ 熟悉描述算法常用的两种流程图
- ▶ 掌握查找最大、最小值算法的实现
- ▶ 掌握杨辉三角算法的实现
- ▶ 掌握常用的 4 种排序算法
- ▶ 掌握算法在实际中的应用

## 6.1 算法基础

算法是解决问题的方法和步骤，对于计算机程序设计语言的初学者，掌握算法是必要的，因为它是学习程序设计的基础，也可以说是程序设计的入门知识。掌握算法可以帮助读者快速理清程序设计的思路，找出多种解决方法，从而选择最合适的解决方案。本节将对算法及描述算法的常用流程图进行详细讲解。

### 6.1.1 初识算法

算法是指为解决某一个问题而采取的步骤和方法的描述。程序设计中的算法是指对计算机工作步骤和方法的描述。

算法的每一个步骤都是严格规定好的，能够被计算机识别并正确执行，并且每一个步骤都能够被计算机理解为一个或一组唯一动作，而不使计算机产生歧义。算法必须有开始和结束，并且必须保证算法规定的每一个步骤最终都能够被完成。

下面通过一个例子对算法进行说明，即交换变量 intA 和 intB 中的内容。

计算机本身不能够直接执行这个操作，交换两个变量的最传统的方法就是借用第三方变量作中间量。具体算法描述如下：

- (1) 将变量 intA 的内容赋给中间变量。
- (2) 将变量 intB 的内容赋给变量 intA。
- (3) 将中间变量存放的内容赋给变量 intB。

最终算法可以写成：

- (1) `intC ← intA.`
- (2) `intA ← intB.`
- (3) `intB ← intC.`

从上面的例子可以看出，算法实际上就是用自然语言或其他方式描述的一个计算机程序，编写计算机程序也就是把用某种方式描述的算法通过程序设计语言重新对其进行描述。

一个算法应具有以下特点。

- 有穷性：一个算法必须在执行有穷多个计算步骤后终止。
- 确定性：一个算法给出的每个计算步骤必须是有精确定义的且无二义性。
- 有效性：算法中的每一个步骤必须有效地执行，并能得到确定结果。
- 输入：一个算法中可以没有输入信息，也可以有一个或多个输入信息。这些输入信息是算法所需的初始数据。
- 输出：一个算法应有一个或多个输出，一个算法得到的结果（中间结果或最后结果）就是算法的输出，没有输出的算法是没有意义的。

### 6.1.2 描述算法的两种常用流程图

为了让算法清晰易懂，需要选择一种好的描述方法。算法的描述方法有很多，有自然语言、传统流程图、N-S 结构化流程图等。自然语言就是用人们日常使用的语言描述解决问题的方法和步骤，这种描述方法通俗易懂，但比较繁琐，并且对程序流向等描述不明了、不直观，而针对这些缺点就产生了传统流程图和

N-S 结构化流程图，下面对这两种流程图进行详细讲解。

### 1. 传统流程图

传统流程图使用不同的几何图形来表示不同性质的操作，使用流程线来表示算法的执行方向，其具有直观形象、逻辑清楚和易于理解等特点。

**说明：**传统流程图占用的篇幅较大，而且流程随意转向，因此对于较大的流程图不易读懂。

传统流程图的基本流程图符号及说明如表 6.1 所示。

表 6.1 流程图符号及说明

流程图符号	名称	说 明
○	起止框	表示算法的开始和结束
□	处理框	表示完成某种操作，如初始化或运算赋值等
◇	判断框	表示根据一个条件成立与否，决定执行两种不同操作的其中一个
□ →	输入输出框	表示数据的输入输出操作
↓ →	流程线	用箭头表示程序执行的流向
○	连接点	用于流程分支的连接

例如，使用传统流程图描述最简单的 if 语句的流程如图 6.1 所示。

### 2. N-S 结构化流程图

N-S 结构化流程图是 1973 年美国学者 I.Nassi 和 B.Shneiderman 首次提出的一种描述算法的图形方法。N-S 结构化流程图将传统流程图中的流程线去掉，并将全部算法写在一个矩形框中，这个矩形框包含其他从属于它的从属框。当程序算法比较繁琐时，一般采用 N-S 结构化流程图。另外，N-S 结构化流程图更有利于程序设计的结构化。

例如，使用 N-S 结构化流程图描述最简单的 if 语句的流程如图 6.2 所示。



**说明：**以上介绍的两种流程图各有特点，但对于初学者和编写较小程序者，建议使用传统流程图来描述算法。

## 6.2 常用的算法

算法在程序开发中起着非常重要的作用，要开发项目，首先必须要掌握常用的算法，本节将对程序开发中最常用到的几种算法进行详细讲解，主要包括查找最大、最小值算法以及杨辉三角算法、冒泡排序算法、插入排序算法、选择排序算法和希尔排序算法等。

### 6.2.1 查找最大、最小值算法的实现

计算一组数的最大值和最小值是数值应用中经常用到的算法，也是程序设计中解决某些问题的基础，如排序问题就涉及比较数值大小的算法。下面给出查找最大值和最小值的算法思路：

(1) 假设一组数中的第一个数为最大数（或最小数）M。

(2) 将给定的这组数中的每个数依次与 M 进行比较，若某个数大于（或小于）M，则将这个数假设为新的最大数（或最小数）M'。

(3) 全部比较完毕后，M 即为所求。

查找最大值的算法流程图如图 6.3 所示。

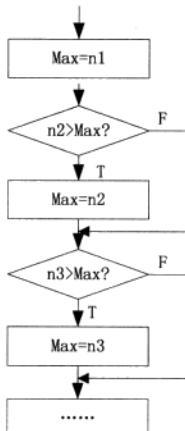


图 6.3 查找最大值算法流程图

说明：图 6.3 使用传统流程图描述了查找最大值的算法流程图，查找最小值的算法流程图与其类似，只需要将相应的>号修改为<号即可。

**例 6.01** 创建一个控制台应用程序，并将其命名为 FindMaxAndMin。该程序主要实现从一组数中查找最大值和最小值的功能，代码如下。（实例位置：光盘\mr\06\sl\6.01）

```

class Program
{
    static void Main(string[] args)
    {
        while (true) // 定义一个死循环，以便能够循环输入
        {
            Console.Write("请输入一组数（用,号隔开）：");
            string strNums = Console.ReadLine(); // 记录输入的一组数
            string[] Array_str = strNums.Split(',');
            int var_Max = Convert.ToInt32(Array_str[0]); // 获取要进行遍历的一组数
            int var_Min = Convert.ToInt32(Array_str[0]); // 记录数组中的第一个数
            for (int i = 1; i < Array_str.Length; i++) // 对数组进行遍历
            {
                // 获取最大的数值
            }
        }
    }
}
  
```

```
        var_Max = var_Max >= Convert.ToInt32(Array_str[i]) ? var_Max : Convert.ToInt32(Array_str[i]);
        //获取最小的数值
        var_Min = var_Min <= Convert.ToInt32(Array_str[i]) ? var_Min : Convert.ToInt32(Array_str[i]);
    }
    Console.WriteLine("最大值: " + var_Max);           //显示最大值
    Console.WriteLine("最小值: " + var_Min);           //显示最小值
}
}
```

程序运行结果如图 6.4 所示。

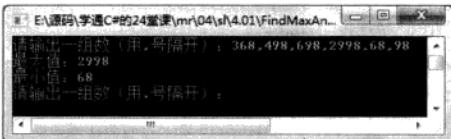


图 6.4 查找最大、最小值算法实例运行结果

### 6.2.2 杨辉三角算法的实现

杨辉三角是一个由数字排列成的三角形数表，其最本质的特征是它的两条边都是由数字 1 组成的，而其余的数则等于它上方的两个数之和。

⑧ 你问我答：杨辉三角的发展历史？

北宋人贾宪约于 1050 年首先使用“贾宪三角”进行高次开方运算。13 世纪中国宋代数学家杨辉在《详解九章算术》中讨论这种形式的数表，并说明此表引自 11 世纪上半叶贾宪的《释锁算术》，并绘画了“古法七乘方图”，因此，杨辉三角又被称为“贾宪三角”。元朝数学家朱世杰在《四元玉鉴》(1303 年)扩充“贾宪三角”成了“古法七乘方图”。意大利人称之为“塔塔利亚三角形”(Triangolo di Tartaglia)，以纪念在 16 世纪发现一元三次方程解的塔塔利亚。在欧洲直到 1623 年以后，法国数学家帕斯卡在 13 岁时发现了“帕斯卡三角”。

**例 6.02** 创建一个控制台应用程序，并将其命名为 YHSJ。该程序通过使用循环语句实现在控制台中显示一个杨辉三角，显示的杨辉三角行数可以由用户自己设定，代码如下。（实例位置：光盘\mr06\sl\6.02）

```
int[][] Array_int = new int[10][];
//向数组中记录杨辉三角形的值
for (int i = 0; i < Array_int.Length; i++)
{
    Array_int[i] = new int[i + 1];
    for (int j = 0; j < Array_int[i].Length; j++)
    {
        if (j <= 1)
        {
            Array_int[i][j] = 1;
            continue;
        }
        else
        {
            //定义一个 10 行的二维数组
            //遍历行数
            //定义二维数组的列数
            //遍历二维数组的列数
            //如果是数组的前两行
            //将其设置为 1
        }
    }
}
```

```

    {
        for (int i = 0; i < Array_int.Length; i++)
        {
            for (int j = 0; j < Array_int[i].Length; j++)
            {
                if (j == 0 || j == Array_int[i].Length - 1)
                    Array_int[i][j] = 1; //如果是行首或行尾
                else
                    Array_int[i][j] = Array_int[i - 1][j - 1] + Array_int[i - 1][j]; //根据杨辉三角算法进行计算
            }
        }
    }

    for (int i = 0; i < Array_int.Length; i++) //输出杨辉三角形
    {
        for (int j = 0; j < Array_int[i].Length; j++)
            Console.Write("{0}\t", Array_int[i][j]);
        Console.WriteLine();
    }
    Console.ReadLine();
}

```

程序运行结果如图 6.5 所示。

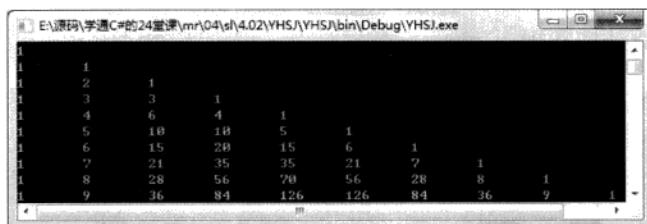


图 6.5 杨辉三角算法实例运行结果

### 6.2.3 冒泡排序法

冒泡排序法是最常用的排序方法之一，其过程很简单，就像气泡一样越往上升越大，因此被人们形象地称为冒泡排序法。冒泡排序法的过程很简单，首先将第一个记录的关键字和第二个记录的关键字进行比较，若为逆序，则将两个记录交换，然后比较第二个记录和第三个记录的关键字，依次类推，直至第  $n-1$  个记录和第  $n$  个记录的关键字进行过比较为止，上述过程称为第一趟冒泡排序，执行  $n-1$  次上述过程后，排序即可完成。

使用冒泡排序法排序的过程如图 6.6 所示。

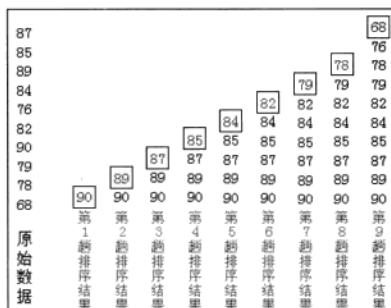


图 6.6 冒泡排序法排序过程

**例 6.03** 创建一个控制台应用程序，并将其命名为 BubbleUpSort。该程序主要使用冒泡排序算法对序列中的元素从小到大进行排序，代码如下。（实例位置：光盘\mr\06\s\6.03）

```

static void Main(string[] args)
{
    int[] arr = new int[] { 87, 85, 89, 84, 76, 82, 90, 79, 78, 68 }; //定义一个一维数组，并赋值
    Console.WriteLine("初始序列：");
    foreach (int m in arr) //循环遍历定义的一维数组，并输出其中的元素
    {
        Console.Write(m + " ");
    }
    //定义两个 int 类型的变量，分别用来表示数组下标和存储新的数组元素
    int j, temp;
    for (int i = 0; i < arr.Length - 1; i++) //根据数组下标的值遍历数组元素
    {
        j = i + 1;
        id:
        if (arr[i] > arr[j]) //定义一个标识，以便从这里开始执行语句
        {
            temp = arr[i];
            arr[i] = arr[j];
            arr[j] = temp;
            goto id; //判断前后两个数的大小
        }
        else
        {
            if (j < arr.Length - 1) //将比较后较大的元素赋值给定义的 int 型变量
            {
                j++;
                goto id; //将后一个元素的值赋值给前一个元素
            }
        }
    }
    Console.WriteLine("排序后的序列：");
    foreach (int n in arr) //将 int 型变量中存储的元素值赋值给后一个元素
    {
        Console.Write(n + " ");
    }
    Console.ReadLine();
}

```

程序运行结果如图 6.7 所示。

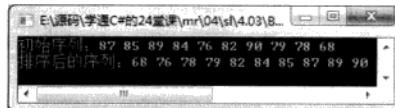


图 6.7 冒泡排序法实例运行结果

## 6.2.4 插入排序法

插入排序法是将一个记录插入到有序数列中，使得到的新的数列仍然有序。插入排序法的算法思想是：将  $n$  个有序数存放在数组  $a$  中，要插入的数为  $x$ ，首先确定  $x$  插在数组中的位置  $p$ ，数组中  $p$  之后的元素都向后移一个位置，空出  $a(p)$ ，将  $x$  放入  $a(p)$ ，这样即可实现插入后数列仍然有序。

使用插入排序法排序的过程如图 6.8 所示。

**例 6.04** 创建一个控制台应用程序，并将其命名为 InsertSort。该程序主要使用插入排序算法对序列中的元素从小到大进行排序，代码如下。（实例位置：光盘\mr\06\sl\6.04）

```

static void Main(string[] args)
{
    int[] arr = new int[] { 20, 40, 90, 30, 80, 70, 50 };           // 定义一个一维数组，并赋值
    Console.WriteLine("初始序列：");
    foreach (int n in arr)                                         // 循环遍历定义的一维数组，并输出其中的元素
    {
        Console.Write("{0} ", n);
    }
    for (int i = 0; i < arr.Length; ++i)                           // 循环访问数组中的元素
    {
        int temp = arr[i];                                         // 定义一个 int 型变量，并使用获得的数组元素值赋值
        int j = i;
        while ((j > 0) && (arr[j - 1] > temp))                  // 判断数组中的元素是否大于获得的值
        {
            arr[j] = arr[j - 1];                                    // 如果是，则将后一个元素的值提前
            --j;
        }
        arr[i] = temp;                                            // 最后将 int 型变量存储的值赋值给最后一个元素
    }
    Console.WriteLine("排序后的序列：");
    foreach (int n in arr)                                         // 循环访问排序后的数组元素并输出
    {
        Console.Write("{0} ", n);
    }
    Console.ReadLine();
}

```

程序运行结果如图 6.9 所示。

	0	1	2	3	4	5	6	
k=2	20	40	90	30	80	70	50	n=7
	0	1	2	3	4	5	6	
k=3	20	30	40	90	80	70	50	n=7
	0	1	2	3	4	5	6	
k=4	20	30	40	80	90	70	50	n=7
	0	1	2	3	4	5	6	
k=5	20	30	40	70	80	90	50	n=7
	0	1	2	3	4	5	6	
k=6	20	30	40	50	70	80	90	n=7

图 6.8 插入排序法排序过程

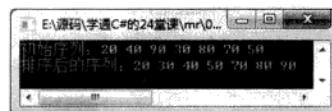


图 6.9 插入排序法实例运行结果

## 6.2.5 选择排序法

选择排序法的基本思想是：每一趟在  $n$  个记录中选取关键字最小的记录作为有序序列的第一个记录，并且令  $I$  从 1 至  $n-1$ ，进行  $n-1$  趟选择操作。

使用选择排序法排序的过程如图 6.10 所示。

**例 6.05** 创建一个控制台应用程序，并将其命名为 SelectSort。该程序主要使用选择排序算法对序列中的元素从小到大进行排序，代码如下。（实例位置：光盘\mr\06\sl\6.05）

```

static void Main(string[] args)
{
    int[] arr = new int[] { 94, 35, 61, 53, 77, 9, 12, 39 };           // 定义一个一维数组，并赋值
    Console.WriteLine("初始序列: ");
    foreach (int n in arr)
        Console.WriteLine("{0}", n + " ");
    int min;                                // 定义一个 int 型变量，用来存储数组下标
    for (int i = 0; i < arr.Length - 1; i++)
    {
        min = i;                            // 定义的数组下标赋值
        for (int j = i + 1; j < arr.Length; j++) // 循环访问数组中的元素值（除最后一个）
        {
            if (arr[j] < arr[min])          // 判断相邻两个元素值的大小
                min = j;                    // 将较小的数组元素值移动到前一位
        }
        int t = arr[min];                  // 定义一个 int 型变量，用来存储较大的数组元素值
        arr[min] = arr[i];                // 将较小的数组元素值移动到前一位
        arr[i] = t;                      // 将 int 型变量中存储的较大的数组元素值向后移
    }
    Console.WriteLine("排序后的序列: ");
    foreach (int n in arr)
        Console.WriteLine("{0}", n + " ");
    Console.ReadLine();
}

```

程序运行结果如图 6.11 所示。

原序列	94	35	61	53	77	9	12	39
第 1 遍选择	9	35	61	53	77	94	12	39
第 2 遍选择	9	12	61	53	77	94	35	39
第 3 遍选择	9	12	35	53	77	94	61	39
第 4 遍选择	9	12	35	39	77	94	61	53
第 5 遍选择	9	12	35	39	53	94	61	77
第 6 遍选择	9	12	35	39	53	61	94	77
第 7 遍选择	9	12	35	39	53	61	77	94

图 6.10 选择排序法排序过程

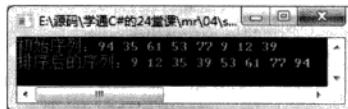


图 6.11 选择排序法实例运行结果

## 6.2.6 希尔排序法

希尔排序又称“缩小增量排序”，其基本思想是：先将整个待排序的一组序列分割成为若干子序列，然后分别进行直接插入排序，待整个序列中的数“基本有序”时再对全体记录进行一次直接插入排序。

**说明：**子序列的构成不是简单地“逐段分割”，而是将相隔某个“增量”的数组成一个子序列，这是希尔排序的特点。

使用希尔排序法排序的过程如图 6.12 所示。

**例 6.06** 创建一个控制台应用程序，并将其命名为 XESort。该程序中首先自定义一个静态的无返回值类型的 Sort 方法，用来使用希尔排序法对数组中的元素进行排序。该方法中使用一个 int 类型的一维数组作为参数，用来表示要排序的一维数组，然后在 Main 方法中定义一个 int 类型的一维数组，最后调用自定义方法 Sort 对其进行排序并输出。代码如下。（实例位置：光盘\mr\06\sl\6.06）

```

#region 希尔排序算法的实现
///<summary>
///希尔排序算法的实现
///</summary>
///<param name="arr">要排序的一维数组</param>
public static void Sort(int[] arr)
{
    int inc;
    for (inc = 1; inc <= arr.Length / 9; inc = 3 * inc + 1);
    for (; inc > 0; inc /= 3)
    {
        for (int i = inc + 1; i <= arr.Length; i += inc)
        {
            int t = arr[i - 1];
            int j = i;
            while ((j > inc) && (arr[j - inc - 1] > t))
            {
                arr[j - 1] = arr[j - inc - 1];
                j -= inc;
            }
            arr[j - 1] = t;
        }
    }
}
#endregion
static void Main(string[] args)
{
    int[] arr = new int[] { 21, 4, 26, 18, 32, 54, 47, 9, 15, 48 }; // 定义一个一维数组，并赋值
    Console.Write("初始序列: ");
    foreach (int n in arr)
        Console.Write("{0} ", n);
    Console.WriteLine();
    Program.Sort(arr); // 循环遍历定义的一维数组，并输出其中的元素
    Console.Write("排序后的序列: ");
    foreach (int m in arr)
        Console.Write("{0} ", m);
    Console.ReadLine();
}

```

程序运行结果如图 6.13 所示。

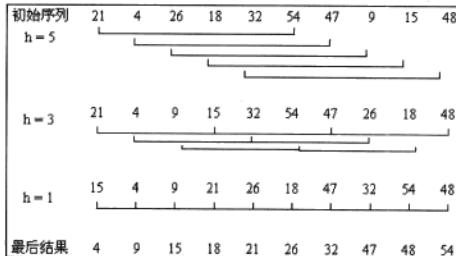


图 6.12 希尔排序法排序过程

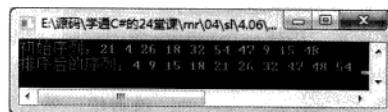


图 6.13 希尔排序法实例运行结果

## 6.3 照猫画虎——基本功训练

### 6.3.1 基本功训练1——计算 $1+2^2+3^3+4^4+\dots+n^n$ 的值

**视频讲解：**光盘\mr\06\lx\计算 $1+2^2+3^3+4^4+\dots+n^n$ 的值.exe

**实例位置：**光盘\mr\06\zmhh\01

新建一个控制台应用程序，在Program.cs文件中编写如下代码。

```
public string sum(int num)
{
    string P_str_Expression = "";
    double sum = 0;
    for (int i = 1; i <= num; i++)
    {
        sum += Convert.ToDouble(Math.Pow(i, i));
        P_str_Expression += i + "的" + i + "次幂" + " + ";
    }
    return P_str_Expression.Remove(P_str_Expression.Length - 3) + " = " + sum; //返回表达式和结果
}
static void Main(string[] args)
{
    while (true) //定义一个循环，以便循环输入数据
    {
        Console.Write("请输入一个整数：");
        int num = Convert.ToInt32(Console.ReadLine()); //记录要计算的数
        Program program = new Program(); //创建 Program 对象
        Console.WriteLine(program.sum(num)); //输出结果
    }
}
```

程序运行结果如图6.14所示。

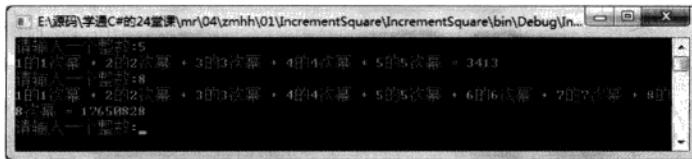


图6.14 计算 $1+2^2+3^3+4^4+\dots+n^n$ 的值

**照猫画虎：**根据以上程序，制作一个计算 $1+2^2+3^3+4^4+5^5+6^6+7^7+8^8+9^9+10^{10}$ 的程序。(20分)(实例位置：光盘\mr\06\zmhh\01\_zmhh)

### 6.3.2 基本功训练2——计算 $10!$ 的值

**视频讲解：**光盘\mr\06\lx\计算 $10!$ 的值.exe

**实例位置：**光盘\mr\06\zmhh\02

新建一个控制台应用程序，在Program.cs文件中编写如下代码。

```

public double factorial(int num)
{
    switch (num)
    {
        case 1:
            return 1;
        default:
            return num * factorial(num - 1);
    }
}
static void Main(string[] args)
{
    while (true)
    {
        Console.WriteLine("输入一个整数: ");
        int num = Convert.ToInt32(Console.ReadLine());
        Program program = new Program();
        Console.WriteLine("{0}! 的值为 {1}", num, program.factorial(num));
    }
}

```

程序运行结果为：3628800。

照猫画虎：根据以上程序，制作一个计算 5! 的程序。(20 分)(实例位置：光盘\mr\06\zmhh\02\_zmhh)

### 6.3.3 基本功训练 3——求最大公约数

视频讲解：光盘\mr\06\lx\求最大公约数.exe

实例位置：光盘\mr\06\zmhh\03

新建一个控制台应用程序，在 Program.cs 文件中编写如下代码。

```

public float maxGongYueShu(int n1, int n2)
{
    int temp = Math.Max(n1, n2);           //求两个数中的最大值
    n2 = Math.Min(n1, n2);                 //求两个数中的最小值
    n1 = temp;                            //记录临时值
    while (n2 != 0)
    {
        n1 = n1 > n2 ? n1 : n2;          //使 n1 中的数大于 n2 中的数
        int m = n1 % n2;                  //记录 n1 求余 n2 的结果
        n1 = n2;                          //交换两个数
        n2 = m;                           //记录求余结果
    }
    return n1;                            //得到最大公约数
}
static void Main(string[] args)
{
    while (true)
    {
        Console.Write("请输入第一个数: ");
        int n1 = Convert.ToInt32(Console.ReadLine()); //记录第一个数
    }
}

```

```

Console.WriteLine("请输入第二个数: ");
int n2 = Convert.ToInt32(Console.ReadLine());
if (n1 * n2 != 0)
{
    Program program = new Program(); //创建 Program 对象
    Console.WriteLine("最大公约数: " + program.maxGongYueShu(n1, n2)); //输出最大公约数
}
else
{
    Console.WriteLine("这两个数不能为 0。");
}
}

```

程序运行结果如图 6.15 所示。

照猫画虎：制作一个计算任意两个数最小公倍数的程序。（20 分）（实例位置：光盘\mr\06\zmhh\03\_zmhh）

#### 6.3.4 基本功训练 4——将 B 转换成 GB、MB 和 KB

视频讲解：光盘\mr\06\lx\将 B 转换成 GB、MB 和 KB.exe

实例位置：光盘\mr\06\zmhh\04

新建一个 Windows 窗体应用程序，在窗体中添加两个 TextBox 控件和一个 Button 控件，然后在 Button 控件的 Click 事件中通过调用自定义的方法实现将输入的 B 字节转换为 GB、MB 和 KB，代码如下。

```

const int GB = 1024 * 1024 * 1024; //定义 GB 的计算常量
const int MB = 1024 * 1024; //定义 MB 的计算常量
const int KB = 1024; //定义 KB 的计算常量
public string ByteConversionGBMBKB(Int64 KSize)
{
    if (KSize / GB >= 1) //如果当前 Byte 的值大于等于 1GB
        return (Math.Round(KSize / (float)GB, 2)).ToString() + "GB"; //将其转换成 GB
    else if (KSize / MB >= 1) //如果当前 Byte 的值大于等于 1MB
        return (Math.Round(KSize / (float)MB, 2)).ToString() + "MB"; //将其转换成 MB
    else if (KSize / KB >= 1) //如果当前 Byte 的值大于等于 1KB
        return (Math.Round(KSize / (float)KB, 2)).ToString() + "KB"; //将其转换成 KB
    else
        return KSize.ToString() + "Byte"; //显示 Byte 值
}
private void button1_Click(object sender, EventArgs e)
{
    textBox2.Text = ByteConversionGBMBKB(Convert.ToInt64(textBox1.Text));
}

```

程序运行结果如图 6.16 所示。



图 6.15 求最大公约数

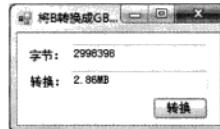


图 6.16 将 B 转换成 GB、MB 和 KB

**熊猫画虎：**制作一个程序，用来将 1024B 转换为对应的 KB 值。(20 分)(实例位置：光盘\mr\06\zmhh\04\_zmhh)

### 6.3.5 基本功训练 5——0~N 位数的任意组合

■**视频讲解：**光盘\mr\06\lx\0~N 位数的任意组合.exe

■**实例位置：**光盘\mr\06\zmhh\05

新建一个 Windows 窗体应用程序，在窗体中添加一个 ListBox 控件、一个 TextBox 控件和一个 Button 控件，然后在 Button 控件的 Click 事件中根据用户输入的位数组合数字，并将组合结果显示在 ListBox 控件中。代码如下。

```

static string[] a = new string[6]{ "0", "1", "2", "3", "4", "5" };
int num = 6;
int nIdx = 0, nSIdx = 0;
public void SetArbitrariness(int n, ArrayList List)
{
    ArrayList SList = new ArrayList(a);
    SList.Clear();
    try
    {
        if (n >= 1)
        {
            if (List.Count == 0)
            {
                for (nIdx = 0; nIdx <= num - 1; nIdx++)
                    SList.Add(a[nIdx]);
            }
            else
            {
                //添加多位数
                for (nIdx = 0; nIdx <= num - 1; nIdx++)
                    for (nSIdx = 0; nSIdx <= List.Count - 1; nSIdx++)
                        if (List[nSIdx].ToString().IndexOf(a[nIdx]) == -1)
                            SList.Add(a[nIdx] + List[nSIdx].ToString());
            }
            SetArbitrariness(n - 1, SList);
        }
        if (SList.Count > 0)
        {
            List.Clear();
            for (int i = 0; i < SList.Count; i++)
                List.Add(SList[i].ToString());
        }
    }
    catch
    {
        SList.Clear();
    }
}
private void button1_Click(object sender, EventArgs e)

```

```

{
    ArrayList List = new ArrayList(a);
    List.Clear();
    listBox1.Items.Clear();
    SetArbitrariness(Convert.ToInt32(textBox1.Text), List);
    //将不同的数值添加到 listBox1 控件上
    for (int i = 0; i < List.Count; i++)
        listBox1.Items.Add(List[i].ToString());
    listBox1.Items.Add("Total:" + listBox1.Items.Count.ToString());
}

```

//创建 ArrayList 对象  
//清空 ArrayList 类  
//清空 listBox1 控件  
//调用自定义方法  
//显示一共执行了几行

程序运行结果如图 6.17 所示。



图 6.17 0~N 位数的任意组合

**照猫画虎：**制作一个程序，用来任意组合两位数。(20 分)(实例位置：光盘\mr\06\zmhh\05\_zmhh)

照猫画虎栏目分数统计：

照猫画虎题目	1	2	3	4	5	总分数	
分数							

## 6.4 情景应用——拓展与实践

### 6.4.1 情景应用 1——身份证号从 15 位升到 18 位算法

视频讲解：光盘\mr\06\lx\身份证号从 15 位升到 18 位算法.exe

实例位置：光盘\mr\06\qjyy\01

在主窗体的“15 位身份证”文本框中输入 15 位身份证号码后，单击“转换”按钮，即可将输入的 15 位身份证号码转换为 18 位。运行结果如图 6.18 所示。



图 6.18 身份证号从 15 位升到 18 位算法

单击“转换”按钮，通过调用自定义方法 Shen 可实现将输入的 15 位身份证号码转换为 18 位的功能，代码如下。

```
public string Shen(string id)
{
    int[] w = new int[] { 7, 9, 10, 5, 8, 4, 2, 1, 6, 3, 7, 9, 10, 5, 8, 4, 2, 1 };
    char[] a = new char[] { '1', '0', 'x', '9', '8', '7', '6', '5', '4', '3', '2' }; //设置 18 位最后一位的值
    string newID = "";
    if (id.Length == 15) //判断位数
    {
        int s = 0;
        newID = id.Insert(6, "19"); //插入字符串
        for (int i = 0; i < 17; i++) //生成前 17 位
        {
            int k = Convert.ToInt32(newID[i]) * w[i];
            s = s + k;
        }
        int h = 0;
        Math.DivRem(s, 11, out h); //取余数
        newID = newID + a[h]; //生成 18 位
    }
    return newID;
}
private void button1_Click(object sender, EventArgs e)
{
    textBox2.Text = Shen(textBox1.Text); //转换身份证位数
}
```

**DIY：**为身份证号从 15 位升到 18 位的程序添加背景。提示：首先设置窗体的背景图片，然后设置 **GroupBox** 控件透明。(20 分)(实例位置：光盘\mr\06\qjyy\01\_diy)

#### 6.4.2 情景应用 2——韩信点兵的算法实现

■ 视频讲解：光盘\mr\06\lx\韩信点兵的算法实现.exe

■ 实例位置：光盘\mr\06\qjyy\02

韩信点兵是一道古代数学题，内容是：韩信带兵不足百人，3 人一行排列多一人，7 人一行排列少两人，5 人一行排列正好。新建一个 Windows 窗体应用程序，在窗体中添加一个 TextBox 文本框和一个 Button 控件，然后在窗体的代码页中编写如下代码。

```
private void button1_Click(object sender, EventArgs e)
{
    int a = 0, b = 0, c = 0; //定义变量
    for (int i = 1; i < 100; i++) //遍历
    {
        Math.DivRem(i, 3, out a); //3 行一列时取余
        Math.DivRem(i, 5, out b); //5 行一列时取余
        Math.DivRem(i, 7, out c); //7 行一列时取余
        if (a == 1 && b == 0 && c == 5) //如果 3 种方式的余数符合要求
        {
            textBox1.Text = i.ToString(); //显示人数
        }
    }
}
```

```
        return;  
    }  
}
```

程序运行结果如图 6.19 所示。

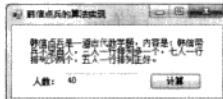


图 6.19 韩信点兵的算法实现

**DIY:** 制作程序实现百钱百鸡算法。提示：用 100 元钱买 100 只鸡，其中，公鸡 5 元一只，母鸡 3 元一只，小鸡 1 元三只。（20 分）（实例位置：光盘\mr\06\qjyy\02\_div）

### 6.4.3 情景应用 3——求水仙花数的算法实现

 视频讲解：光盘\mr\06\lx\求水仙花数的算法实现.exe

 实例位置：光盘\mr\06\qjyy\03

水仙花数就是一个 3 位数，每一位数的立方相加等于该数本身。新建一个 Windows 窗体应用程序，在窗体上添加一个 ListBox 控件和一个 Button 控件，然后在窗体的代码页中编写如下代码。

```

private void button1_Click(object sender, EventArgs e)
{
    int a = 0, b = 0, c = 0;
    listBox1.Items.Clear();
    for (int i = 100; i < 1000; i++)
    {
        a = i / 100;
        Math.DivRem(i, 100, out b);
        b = b / 10;
        Math.DivRem(i, 10, out c);
        a = a * a * a;
        b = b * b * b;
        c = c * c * c;
        if ((a + b + c) == i)
            listBox1.Items.Add(i.ToString());
    }
}

```

//定义变量  
//清空 listBox1 控件  
//遍历所有 3 位数  
  
//获取 3 位数中的第一位数  
//获取 3 位数中的后两位数  
//获取 3 位数中的第二位数  
//获取 3 位数中的第三位数  
//计算第一位数的立方  
//计算第二位数的立方  
//计算第三位数的立方  
//如果符合水仙花数  
//显示当前 3 位数

程序运行结果如图 6.20 所示。

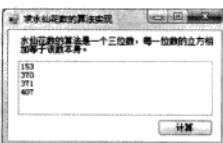


图 6.20 求水仙花数的算法实现

DIY：制作程序实现斐波纳契数列求和算法。提示：第一项和第二项均为 1，以后各项都为前两项之和。  
(20 分)(实例位置：光盘\mr\06\qjyy\03\_diy )

#### 6.4.4 情景应用 4——制作一个迷你星座查询器

视频讲解：光盘\mr\06\lx\制作一个迷你星座查询器.exe

实例位置：光盘\mr\06\qjyy\04

众所周知，传统的星座共有 12 个，分别是“白羊座”、“处女座”、“金牛座”、“巨蟹座”、“摩羯座”、“射手座”、“狮子座”、“双鱼座”、“双子座”、“水瓶座”、“天秤座”和“天蝎座”。每个人都有属于自己的星座，每种星座都有相应的诠释，其中包括运势、爱情、事业、性格和幸运颜色等，而迷你星座查询器就是一款根据用户的生日来查询所属星座及其相关信息的软件。运行效果如图 6.21 所示。

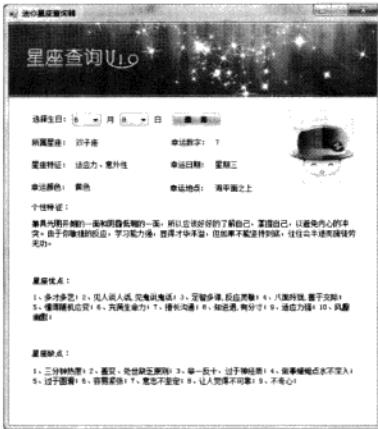


图 6.21 迷你星座查询器

实现过程如下。

- (1) 创建一个 Windows 窗体应用程序，命名为 ConstellationSearch，默认窗体为 Form1.cs。
- (2) 在默认窗体 Form1 中拖放程序需要用到的控件，如表 6.2 所示。

表 6.2 Form1 窗体中用到的主要控件

控件类型	控件 ID	主要属性设置	用途
ComboBox	cbbMonth	DropDownStyle 属性设为 DropDownList	选择出生月份
	cbbDay	DropDownStyle 属性设为 DropDownList	选择出生日
RichTextBox	rtbpersonality	BorderStyle 属性设为 None	显示个性特征
	rtbvirtue	BorderStyle 属性设为 None	显示星座优点
	rtbfaw	BorderStyle 属性设为 None	显示星座缺点
Label	lblName	无	显示所属星座名称
	lblTrait	无	显示星座特征
	lblColor	无	显示幸运色
	lblNum	无	显示幸运数字
	lblDate	无	显示幸运日
	lblPlace	无	显示幸运地点

续表

控件类型	控件ID	主要属性设置	用途
PictureBox	ptbConstellation	SizeMode 属性设为 StretchImage	显示与查询结果相对应的星座图片
	pictureBox1	无	查询
DateTimePicker	dateTimePicker1	无	选择出生日期

(3) 在窗体的后台代码中,首先定义一个 Search 类,其中定义两个静态的自定义方法 ConstellationDescription 和 GetConstellation, ConstellationDescription 方法用来根据星座返回相应的星座特色、幸运色、幸运数字、幸运日、幸运地、个性特征、优点和缺点;而 GetConstellation 方法用来根据生日查询所属的星座。代码如下。

```
public class Search
{
    /// <summary>
    /// 根据星座返回相应的星座特色、幸运色、幸运数字、幸运日、幸运地、个性特征、优点、缺点
    /// </summary>
    /// <param name="str">方法的参数, 代表某个星座名称</param>
    /// <returns></returns>
    public static string[] ConstellationDescription(string str)
    {
        string[] Description = new string[8];
        if (str != "")
        {
            switch (str)
            {
                case "摩羯座":
                    Description[0] = "现实、安全、平稳。"; //星座特色
                    Description[1] = "暗绿色"; //幸运色
                    Description[2] = "6"; //幸运数字
                    Description[3] = "星期三"; //幸运日
                    Description[4] = "远离嘈杂的地点"; //幸运地
                    Description[5] = "你深思慎、冷静而准确的判断力, 给人沉稳而严肃的印象。你有强烈的责任感和事业心, 时时鞭策自己努力实现理想。但是, 你凡事都太过认真, 乃至拘泥, 而显得过于刚强, 冥顽不灵。"; //个性特征
                    Description[6] = "1、有实际的人生观; 2、做事脚踏实地; 3、意志力强, 不容易受影响; 4、处处谨慎; 5、有克服困难的毅力; 6、坚守原则, 重视纪律; 7、有家庭观念; 8、对人谦逊; 9、有独树一帜的幽默感"; //优点
                    Description[7] = "1、太过现实; 2、固执; 3、不够乐观; 4、个人利己主义; 5、缺乏浪漫情趣; 6、过于压抑自己的欲望; 7、太专注于个人的目标; 8、缺乏对人群的关怀和热情; 9、不擅于沟通; 10、不能随机应变"; //缺点
                    break;
                //.....其他星座的相关信息
                default:
                    Description[0] = "星座输入错误";
                    break;
            }
        }
        return Description;
    }
    /// <summary>
```

```

///根据生日查询所属星座
///<summary>
///<param name="strBirthday">生日</param>
///<returns></returns>
public static string GetConstellation(DateTime strBirthday)
{
    string strConstellation = null; //定义一个字符串，用来记录星座
    float birthday = 0.00F; //定义一个 float 变量，用来记录使用生日组成的数字
    if (strBirthday.Month == 1 && strBirthday.Day < 20)
    {
        birthday = float.Parse(string.Format("13.{0}", strBirthday.Day)); //将生日格式化为数字
    }
    else
    {
        if (strBirthday.Day < 10) //判断生日中的天数是否小于 10
            birthday = float.Parse(string.Format("{0}.0{1}", strBirthday.Month, strBirthday.Day));
        else
            birthday = float.Parse(string.Format("{0}.{1}", strBirthday.Month, strBirthday.Day));
    }
    float[] atomBound = { 1.20F, 2.20F, 3.21F, 4.21F, 5.21F, 6.22F, 7.23F, 8.23F, 9.23F, 10.23F, 11.21F,
                         12.22F, 13.20F }; //定义星座间隔之间的数字
    string[] atoms = { "水瓶座", "双鱼座", "白羊座", "金牛座", "双子座", "巨蟹座", "狮子座",
                       "处女座", "天秤座", "天蝎座", "射手座", "摩羯座" }; //定义星座名称
    for (int i = 0; i < atomBound.Length - 1; i++) //遍历星座间隔之间的数字
    {
        if (atomBound[i] <= birthday && atomBound[i + 1] > birthday) //判断遍历到的数字与生日的大小关系
        {
            strConstellation = atoms[i]; //获取星座名称
            break;
        }
    }
    return strConstellation; //返回得到的星座名称
}
}

```

(4) 在 Form1 窗体的 Load 事件中，向 ComboBox 控件中添加月份和日期，以便选择出生日期，代码如下。

```

private void Form1_Load(object sender, EventArgs e)
{
    for (int i = 1; i <= 12; i++)
    {
        cbbMonth.Items.Add(i.ToString()); //初始化月份下拉列表
    }
    for (int j = 1; j <= 31; j++)
    {
        cbbDay.Items.Add(j.ToString()); //初始化日期下拉列表
    }
    cbbDay.SelectedIndex = cbbMonth.SelectedIndex = 0; //默认选择 1 月 1 日
}

```

(5) 选择出生日期后单击“查询”按钮，根据用户所选择的生日调用自定义类中的相关方法查询其星

座相关信息，并将查询结果显示在窗体的相应控件中，代码如下。

```

private void pictureBox1_Click(object sender, EventArgs e)
{
    try
    {
        DateTime dt = new DateTime(1, Convert.ToInt32(cbbMonth.SelectedItem.ToString()),
        Convert.ToInt32(cbbDay.SelectedItem.ToString())); //获取选择的生日
        lblName.Text = Search.GetConstellation(dt); //根据生日获取所属星座名称
        //获取其他信息
        string[] info = Search.ConstellationDescription(Search.GetConstellation(dt));
        lblTrait.Text = info[0]; //星座特征
        lblColor.Text = info[1]; //幸运色
        lblNum.Text = info[2]; //幸运数
        lblDate.Text = info[3]; //幸运日
        lblPlace.Text = info[4]; //幸运地点
        rtbpersonality.Text = info[5]; //个性特征
        rtbvirtue.Text = info[6]; //星座优点
        rtbflaw.Text = info[7]; //星座缺点
        switch (lblName.Text.Trim())
        {
            case "白羊座":
                ptbConstellation.Image = Properties.Resources.白羊座; break;
            case "处女座":
                ptbConstellation.Image = Properties.Resources.处女座; break;
            case "金牛座":
                ptbConstellation.Image = Properties.Resources.金牛座; break;
            case "巨蟹座":
                ptbConstellation.Image = Properties.Resources.巨蟹座; break;
            case "摩羯座":
                ptbConstellation.Image = Properties.Resources.摩羯座; break;
            case "射手座":
                ptbConstellation.Image = Properties.Resources.射手座; break;
            case "狮子座":
                ptbConstellation.Image = Properties.Resources.狮子座; break;
            case "双鱼座":
                ptbConstellation.Image = Properties.Resources.双鱼座; break;
            case "双子座":
                ptbConstellation.Image = Properties.Resources.双子座; break;
            case "水瓶座":
                ptbConstellation.Image = Properties.Resources.水瓶座; break;
            case "天秤座":
                ptbConstellation.Image = Properties.Resources.天秤座; break;
            case "天蝎座":
                ptbConstellation.Image = Properties.Resources.天蝎座; break;
        }
    }
    catch
    {
        MessageBox.Show("选择的日期有误！","警告",MessageBoxButtons.OK,MessageBoxIcon.Error);
    }
}

```

**DIY:** 制作一个程序，实现根据用户选择的出生年份查询其所属生肖的功能。(20 分)(实例位置：光盘\mr\06\qjyy\04\_diy)

### 6.4.5 情景应用 5——设计双色球彩票选号器

视频讲解：光盘\mr\06\lx\设计双色球彩票选号器.exe

实例位置：光盘\mr\06\qjyy\05

“彩票”是目前最为流行的一种博彩活动，这种活动大部分都是以一系列或一定位数的数字组成一个号码，如果彩民买中了中奖号码，就可以获得奖励。例如，中国福利彩票“双色球”就是一种由中国福利彩票发行管理中心统一组织发行，在全国销售联合发行的“乐透型”福利彩票，“双色球”彩票投注区分为红色球号码区和蓝色球号码区，每个彩民的投注号码都由 6 个红色球号码和 1 个蓝色球号码组成，其中，红色球号码从 1~33 中选择；蓝色球号码从 1~16 中选择。这里制作一个双色球彩票选号器，以便用来模拟双色球中奖号码的生成过程。运行效果如图 6.22 所示。

实现过程如下。

- (1) 创建一个 Windows 窗体应用程序，命名为 DbColorBall，默认窗体为 Form1.cs。
- (2) 在默认窗体 Form1 中拖放程序需要用到的控件，如表 6.3 所示。

表 6.3 Form1 窗体中用到的主要控件

控件类型	控件 ID	主要属性设置	用途
Button	button1	Text 属性设为“开始”，FlatStyle 属性设为 Flat，FlatAppearance.BorderSize 属性设为 0	开始、停止中奖号码的生成
Label	label1	ForeColor 属性设为 Red	显示生成的中奖号码
Timer	timer1	无	控制中奖号码的生成，并绘制双色球

(3) 在 Form1 窗体的后台代码中，首先定义 7 个 int 类型的变量，分别用来记录产生的 7 个中奖号码，代码如下。

```
int num1, num2, num3, num4, num5, num6, num7; //定义 7 个 int 型变量，分别用来记录 7 个中奖号码
```

(4) 在 Form1 窗体的后台代码中，自定义一个返回值类型为 int 类型的 DrawBall 方法，该方法主要用来随机生成中奖号码，并根据生成的中奖号码绘制双色球。代码如下。

```
//绘制双色球
private int DrawBall(int min,int max,Point pt)
{
    Graphics g = this.CreateGraphics(); //创建绘图对象
    Random rnd = new Random(); //创建随机对象
    int num = rnd.Next(min, max); //生成随机数字
    switch (num)
    {
        case 1:
            g.DrawImage(Image.FromFile("01.jpg"), pt); //以生成的数字为条件绘制双色球
            break;
        //.....其他数字的双色球绘制
    }
    return num; //在指定位置绘制双色球
}
```



图 6.22 双色球彩票选号器

(5) 单击“开始”按钮，判断 Button 按钮的 Text 文本值，如果为“开始”，则启动 Timer 计时器；如果为“停止”，则停止计时器，并在 Label 控件中显示生成的中奖号码。代码如下。

```
private void button1_Click(object sender, EventArgs e)
{
    if (button1.Text == "开始")
    {
        button1.Text = "停止";
        timer1.Start(); //启动计时器
    }
    else if (button1.Text == "停止")
    {
        button1.Text = "开始";
        timer1.Stop(); //停止计时器
        label1.Text = num1.ToString("00") + " " + num2.ToString("00") + " " + num3.ToString("00") + " " +
            num4.ToString("00") + " " + num5.ToString("00") + " " + num6.ToString("00") + " " +
            (num7 - 33).ToString("00"); //显示本期中奖号码
    }
}
```

(6) 在 Timer 计时器的 Tick 事件中，主要调用自定义的 DrawBall 方法来根据生成的中奖号码绘制双色球，代码如下。

```
private void timer1_Tick(object sender, EventArgs e)
{
    Point pt1 = new Point(61, 117); //定义第1个数字的位置
    Point pt2 = new Point(93, 117); //定义第2个数字的位置
    Point pt3 = new Point(125, 117); //定义第3个数字的位置
    Point pt4 = new Point(157, 117); //定义第4个数字的位置
    Point pt5 = new Point(189, 117); //定义第5个数字的位置
    Point pt6 = new Point(221, 117); //定义第6个数字的位置
    Point pt7 = new Point(273, 117); //定义第7个数字的位置
    num1 = DrawBall(1, 33, pt1); //抽取第1个数字
    Thread.Sleep(5); //线程休眠5毫秒
    num2 = DrawBall(1, 33, pt2); //抽取第2个数字
    Thread.Sleep(5); //线程休眠5毫秒
    num3 = DrawBall(1, 33, pt3); //抽取第3个数字
    Thread.Sleep(5); //线程休眠5毫秒
    num4 = DrawBall(1, 33, pt4); //抽取第4个数字
    Thread.Sleep(5); //线程休眠5毫秒
    num5 = DrawBall(1, 33, pt5); //抽取第5个数字
    Thread.Sleep(5); //线程休眠5毫秒
    num6 = DrawBall(1, 33, pt6); //抽取第6个数字
    Thread.Sleep(5); //线程休眠5毫秒
    num7 = DrawBall(34, 50, pt7); //抽取第7个数字
}
```

**DIY：**制作一个程序，实现根据手机号码尾数进行抽奖的功能。(20分)(实例位置：光盘\mr\06\qjyy\05\_diy)

情景应用 DIY 栏目分数统计：

DIY 题目	1	2	3	4	5	总分数	
分数							

## 6.5 自我测试

### 一、选择题（每题 10 分，5 道题）

1. 一位美国学者提出了一种用方框图来代替传统的程序流程图的想法，该图符合结构化程序设计原则，通常也把这种图称为（ ）。
  - A. 结构图
  - B. 数据流图
  - C. N-S 图
  - D. PAD 图
2. 根据数据结构中各数据元素之间前后关系的复杂程度，一般将数据结构分为（ ）。
  - A. 紧凑结构和非紧凑结构
  - B. 线性结构和非线性结构
  - C. 内部结构和外部结构
  - D. 动态结构和静态结构
3. 表达式 `Math.Abs(-3.5)+"1235".IndexOf('3')` 的值为（ ）。
  - A. -1.5
  - B. -0.5
  - C. 5.5
  - D. 6.5
4. 表达式 “`Math.Round(Math.Sqrt(-2.5))`” 的值为（ ）。
  - A. -2
  - B. -1
  - C. 0
  - D. 非数字
5. 已知 `Random r=new Random();`，则表达式 “`Math.Floor(r.NextDouble()*10+1)`” 的取值范围为（ ）。
  - A. (0,9)
  - B. [0,9]
  - C. (1,10)
  - D. [1,10]

### 二、填空题（每题 10 分，5 道题）

1. C# 可执行应用程序的入口点是（ ）。
2. 结束应用程序的语句是（ ）。
3. 队列在 C# 中使用（ ）类表示。
4. 描述算法的两种常用流程图分别为（ ）和（ ）。
5. 下面代码运行后输出的结果是（ ）。

```
class Test
{
    public enum WeekDays
    {
        Mon,Tue,Wed,Thur,Fri,Sat,Sun
    }
    static void Main()
    {
        WeekDays week=(WeekDays)2;
        Console.WriteLine(week);
    }
}
```

测试分数统计：

类别	第 1 题	第 2 题	第 3 题	第 4 题	第 5 题
选择题分数					
填空题分数					
总分数					

## 6.6 行动指南

开始日期: \_\_\_\_\_ 年 \_\_\_\_\_ 月 \_\_\_\_\_ 日

结束日期: \_\_\_\_\_ 年 \_\_\_\_\_ 月 \_\_\_\_\_ 日

序号	内 容	行动指南	
1	照猫画虎栏目 分数( )	分数>75 分	优秀, 基本功掌握得不错, 加油!
		75 分>分数>50 分	及格, 知识掌握得不牢, 重新做一遍照猫画虎。
	情景应用栏目 分数( )	分数<50 分	请使用光盘中提供的“实战能力测试系统”进行提高训练。
		分数>75 分	优秀, 综合应用能力很强。
	自我测试栏目 分数( )	75 分>分数>50 分	及格, 综合应用能力需提高, 再练一遍情景应用。
		分数<50 分	请使用光盘中提供的“实战能力测试系统”进行提高训练。
2	编程能力培养: 本栏目提供了一些实践题目, 对于培养编程能力很有效, 要做好。	分数>75 分	优秀, 有成为编程高手的潜质。
		分数<75 分	请使用光盘中提供的“实战能力测试系统”进行提高训练。
		综合评价 反复训练后, 上各项分数都在 75 分以上, 方可进入下一堂课学习。	
		1. 开发一个程序, 实现鸡尾酒排序算法。提示: 鸡尾酒排序也称双向冒泡排序、搅拌排序或涟漪排序, 当使用鸡尾酒排序时, 首先对要排序的内容从低到高排序, 然后再从高到低排序。 2. 使用 C#实现堆栈结构。 3. 使用 C#实现二叉树。	
3	编程习惯培养: 本课堂培养读者在学习开发中记笔记的习惯, 把开发中遇到的问题、总结的经验记录下来。		
4	创新能力培养: 看看身边有没有可以用编程解决的问题, 记录到右边的表格中。根据学习进度, 尝试编写解决这些问题的小程序。		

## 6.7 成功可以复制——缔造华人的硅谷传奇杨致远

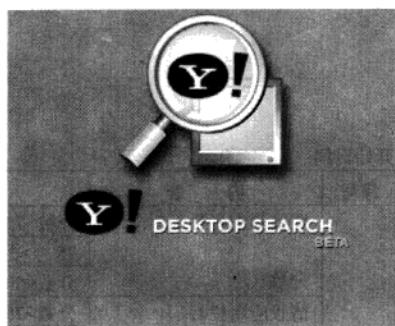
1994 年 4 月, 就读于斯坦福大学的杨致远与费洛为了完成论文, 在网络上寻找资料。在这个过程中, 他们收集了很多自己感兴趣的站点并加入到书签, 以便查找。可是当这些所收集的站点越来越多时, 管理和查找就成了大麻烦。于是他们把这些书签按类别整理好, 编制成了软件《Jerry's Guide to the World Wide Web》(杰瑞全球资讯网指南), 放到网络上让其他冲浪的人享用, 结果深受大家的欢迎与拥护。

随着《指南》越来越受欢迎与关注, 许多网友纷纷进入斯坦福大学电机系的工作站使用这套软件, 使校方大感困扰, 抱怨这项发明影响了学校计算机的正常运作。但是杨致远与费洛仍然积极为此努力, 并推出特色栏目 Cool Links 和 Hard to Believe, 并将网站改名为《Jerry and David's Guide to the World Wide Web》。但是由于条件限制, 网站的数据和搜索引擎只分别放在他们个人的计算机中。

在没日没夜、满目狼藉的工作室中，1994 年夏天的一个晚上，杨致远与费洛为他们的网站取了一个很特殊的名字——Yahoo！，于是神奇的 Yahoo！网站诞生了，当时是凌晨两点。由于 Yahoo！检索系统非常方便，前景被普遍看好，广告收入也相当乐观。结果，Yahoo！一上市就一鸣惊人，风头大出。

1996 年 4 月 12 日，Yahoo！股票首次上市，每股股价为 \$13，而开盘就达到 \$24.50，并持续飙升至 \$43。一下子给 Yahoo！公司带来了将近 8.48 亿美金的市场值，并创造了纳斯达克纪录。1998 年《福布斯》杂志推出高科技百名富翁，杨致远以 10 亿美元的财富跃居第 16 位，超过了冠群 CEO 王嘉廉，成为高科技中的华人首富。1999 年，杨致远的纸面财富更是达到 75 亿美元。

Yahoo！的成功与杨致远清晰快捷的思维和不知疲倦地为事业打拼是分不开的，在当今 IT 业，有 3 位没有上完大学就跑出去开公司、最后变成著名的亿万富翁的人，他们就是比尔·盖茨、迈克·戴尔和杨致远。



### 经典语录

如果只是为了成功和金钱去创业，能接受失败吗？不能。怎样才能接受失败？是因为能坚持，对所做的事情的热爱，一种固执的“笨”。在创业中，过程始终比终点更为重要。

### 深度评价

“我不怕输，即使我失败，我也有重新来过的勇气。”杨致远这种不怕输的劲头也衬托出一个传奇人物的精髓所在。引用杨致远的一句话：机会是偶然的，但取得成功并非如此，你把一种嗜好变成了一个事业，这需要巨大的努力。

# 第 7 堂课

## 面向对象程序设计

( 视频讲解：160分钟)

面向对象程序设计是在面向过程程序设计的基础上发展而来的，它将数据和对数据的操作看作是一个不可分割的整体，力求将现实问题简单化，因为这样不仅符合人们的思维习惯，同时也可以提高软件的开发效率，并方便后期的维护。本堂课将对面向对象程序设计中的基本知识进行详细讲解。

学习摘要：

- ▶ 了解面向对象程序设计的基本概念
- ▶ 掌握属性的定义及使用
- ▶ 掌握方法的声明及使用
- ▶ 掌握结构的定义与使用
- ▶ 掌握类与对象的使用
- ▶ 掌握面向对象的3个基本特性

## 7.1 面向对象编程概述

面向对象编程（Object-Oriented Programming）简称 OOP 技术，它是开发应用程序的一种新方法、新思想。过去的面向过程编程常常会导致所有的代码都包含在几个模块中，使程序难以阅读和维护，对软件做一些修改时常常牵一动百，使以后的开发和维护难以为继，而使用 OOP 技术，常常要使用许多代码模块，每个模块都只提供特定的功能，它们是彼此独立的，这样就提高了代码重用的几率，更加有利于软件的开发、维护和升级。

在面向对象编程中，算法与数据结构被看作是一个整体，称作对象，现实世界中任何类的对象都具有一定的属性和操作，也总能用数据结构与算法两者合而为一的来描述，所以可以用下面的等式来定义对象和程序。

对象 = (算法 + 数据结构)，程序 = (对象 + 对象 + ……)。

从上面的等式可以看出，程序就是许多对象在计算机中相继表现自己，而对象则是一个个程序实体。

面向对象的编程方式具有封装、继承和多态性等特点，下面进行详细介绍。

(1) 封装：类是属性和方法的集合，为了实现某项功能而定义类后，开发人员并不需要了解类体内每句代码的具体含义，只需通过对象来调用类内某个属性或方法即可实现某项功能，这就是类的封装性。

例如，在使用计算机时，并不需要将计算机拆开了解每个部件的具体用处，用户只需按下主机箱上的 Power 按钮就可以启动计算机，在键盘上敲打就可以将文字输入到计算机中，但对于计算机内部的构造，用户可能根本不了解，这就是封装的具体表现。

(2) 继承：通过继承可以创建子类（派生类）和父类之间的层次关系，子类（派生类）可以从其父类中继承属性和方法，通过这种关系模型可以简化类的操作。假如已经定义了 A 类，接下来准备定义 B 类，而 B 类中有很多属性和方法与 A 类相同，那么就可以使 B 类继承于 A 类，这样就无须再在 B 类中定义 A 类已有的属性和方法，从而可以在很大程度上提高程序的开发效率。

**例 7.01** 可以将水果看成一个父类，那么水果类具有颜色属性，然后再定义一个苹果类，在定义苹果类时完全可以不定义苹果类的颜色属性，通过如下继承关系完全可以使苹果类具有颜色属性。

```
class 水果类
{
    public 颜色；      //在水果类中定义颜色属性
}
class 苹果类：水果类
{
    //苹果类中其他的属性和方法
}
```

(3) 多态：类的多态性是指不同的类进行同一操作可以有不同的行为。例如，定义一个火车类和一个汽车类，火车和汽车都可以移动，说明两者在这方面可以进行相同的操作，然而，火车和汽车移动的行为是截然不同的，因为火车必须在铁轨上行驶，而汽车在公路上行驶，这就是类的多态性的形象比喻。

## 7.2 属性的定义及使用

属性提供功能强大的方法以将声明信息与 C# 代码（类型、方法、属性等）相关联，一旦属性与程序实

体关联，即可使用名为反射的技术对属性进行查询。本节将对属性进行详细讲解。

### 7.2.1 属性概述

属性是一种用于访问对象或类的特性的成员，它可以包括字符串的长度、字体的大小、窗体的标题和客户的名称等信息。属性是成员的自然扩展，两者都是关联类型的命名成员。

属性有访问器，这些访问器指定在它们的值被读取或写入时需要执行的语句，因此属性提供了一种机制，它把读取和写入对象的某些特性与一些操作关联起来。可以像使用公共数据成员一样使用属性，但实际上它们是称为“访问器”的特殊方法，这使得数据在可被轻松访问的同时，仍能提供方法的安全性和灵活性。

属性结合了字段和方法的多个方面。对于对象的用户，属性显示为字段，访问该属性需要完全相同的语法；对于类的实现者，属性是一个或两个代码块，表示一个 get 访问器和/或一个 set 访问器。当读取属性时，执行 get 访问器的代码块；当向属性分配一个新值时，执行 set 访问器的代码块。不具有 set 访问器的属性被视为只读属性，不具有 get 访问器的属性被视为只写属性，同时具有这两个访问器的属性为可读可写属性。

 注意：属性不能作为 ref 参数或 out 参数传递。

属性具有以下特点。

- 属性可向程序中添加元数据。元数据是指嵌入程序中的信息，如编译器指令或数据描述等。
- 程序可以使用反射检查自己的元数据。
- 通常使用属性与 COM 交互。

### 7.2.2 属性的定义

属性以两种形式存在：一种是在公共语言运行库的基类库中定义的属性，另一种是可以创建、可以向代码中添加附加信息的自定义属性。下面讲解如何对属性进行定义。

**例 7.02** 下面代码用来将 System.Reflection.TypeAttributes.Serializable 属性用于自定义类，以便使该类中的成员可以序列化。代码如下。

```
[System.Serializable]
public class MyClass
{
}
```

 说明：上面代码中的 Serializable 为 .Net Framework 类库中定义的属性。

自定义属性在类模块内是通过以下方式声明的：指定属性的访问级别，后面是属性的类型，接下来是属性的名称，然后是声明 get 访问器和/或 set 访问器的代码模块。

**例 7.03** 下面代码自定义了一个 Date 类，该类中有一个属性 day，因为该属性提供了 get 和 set 访问器，因此它是可读可写属性。代码如下。

```
public class Date
{
    private int Day = 7;
    public int day      //星期属性，该属性为可读可写
    {
        get
        {
```

```
    return Day;  
}  
set  
{  
    if ((value > 0) && (value < 8))  
    {  
        Day = value;  
    }  
}
```

**说明:** get 访问器与方法体相似, 它必须返回属性类型的值; 而 set 访问器类似于返回类型为 void 的方法, 它使用称为 value 的隐式参数, 此参数的类型是属性的类型。

### 7.2.3 屬性的使用

程序中调用属性的语法格式如下。

对象名.属性名

**注意:** ① 如果要在其他类中调用自定义属性, 必须将自定义属性的访问级别设置为 public。  
② 如果属性为只读属性, 则不能在调用时为其赋值, 否则将会产生异常。

**例 7.04** 创建一个控制台应用程序，其中定义了一个 MyClass 类，并在该类中定义了两个 string 类型的变量，分别用来记录用户的编号和姓名，然后在该类中自定义两个属性，用来表示用户编号和姓名。定义完成后，在 Program 主程序类中创建自定义类 MyClass 的一个对象，并分别为其中定义的用户编号和用户名属性赋值，最后调用 Console 类的 WriteLine 方法将赋值后的用户编号和用户名输出。程序代码如下。

(实例位置: 光盘\mr\07\s\7.04)

```
class MyClass
{
    private string id = "";
    private string name = "";
    public string ID
    {
        get
        {
            return id;
        }
        set
        {
            id = value;
        }
    }
    public string Name
    {
        get
        {
            return name;
        }
    }
}
```

//定义一个 string 类型的变量，用来记录用户编号  
//定义一个 string 类型的变量，用来记录用户姓名  
//定义用户编号属性，该属性为可读可写属性

//定义用户姓名属性，该属性为可读可写属性

```

set
{
    name = value;
}
}

class Program
{
    static void Main(string[] args)
    {
        MyClass myclass = new MyClass();           //创建 MyClass 对象
        myclass.ID = "BH001";                      //为用户编号属性赋值
        myclass.Name = "TM1";                      //为用户姓名属性赋值
        Console.WriteLine(myclass.ID + " " + myclass.Name); //输出用户编号和用户姓名
        myclass.ID = "BH002";                      //重新为用户编号属性赋值
        myclass.Name = "TM2";                      //重新为用户姓名属性赋值
        Console.WriteLine(myclass.ID + " " + myclass.Name); //再次输出用户编号和用户姓名
        Console.ReadLine();
    }
}

```

按 Ctrl+F5 键查看运行结果，如图 7.1 所示。



图 7.1 属性的使用实例运行结果图

## 7.3 方法的声明及使用

方法是编写程序代码过程中一个非常重要的部分，使用它可以提高代码的重用性，而且也可以方便后期代码的维护。本节将对方法进行详细讲解。

### 7.3.1 方法概述

方法是一种用于实现可以由对象或类执行的计算或操作的成员。类的方法主要是和类相关联的动作，它是类的外部界面，对于那些私有的字段来说，外部界面实现对它们的操作一般只能通过方法来实现。方法是包含一系列语句的代码块，在 C# 中，每个执行指令都是在方法的上下文中完成的。

### 7.3.2 方法修饰符

声明方法时，需要为其指定修饰符，以指定其访问级别或使用限制，C# 中常用的修饰符有 `private`、`public`、`protected`、`internal` 等 4 个访问修饰符和 `partial`、`new`、`static`、`virtual`、`override`、`sealed`、`abstract`、`extern` 等 8 个声明修饰符，下面分别对它们进行详细介绍。

- private:** 私有访问是允许的最低访问级别，私有成员只有在声明它们的类和结构体中才可以访问。

- public:** 公共访问是允许的最高访问级别，对访问公共成员没有限制。
  - protected:** 受保护成员在它的类中可访问并且可由派生类访问。
  - internal:** 只有在同一程序集的文件中，内部类型或成员才是可访问的。
  - partial:** 在整个同一程序集中定义分部类和结构。
  - new:** 从基类成员隐藏继承的成员。
  - static:** 声明属于类型本身而不是属于特定对象的成员。
  - virtual:** 在派生类中声明其实现可由重写成员更改的方法或访问器。
  - override:** 提供从基类继承的虚拟成员的新实现。
  - sealed:** 指定类不能被继承。
  - abstract:** 指示某个类只能是其他类的基类。
  - extern:** 指示在外部实现方法。
- 当声明方法时，需要遵循以下修饰符使用规则。
- 声明包含一个有效的访问修饰符组合。
  - 声明中所包含的修饰符彼此各不相同。
  - 声明最多包含下列修饰符中的一个，即 static、virtual 或 override。
  - 声明最多包含下列修饰符中的一个，即 new 或 override。
  - 如果声明包含 abstract 修饰符，则该声明不包含下列任何修饰符，即 static、virtual、sealed 或 extern。
  - 如果声明包含 private 修饰符，则该声明不包含下列任何修饰符，即 virtual、override 或 abstract。
  - 如果声明包含 sealed 修饰符，则该声明还包含 override 修饰符。

### 7.3.3 方法的声明

方法在类或结构中声明，声明时需要指定访问级别、返回值、方法名称及方法参数，方法参数放在括号中，并用逗号隔开。若括号中没有内容，则表示声明的方法没有参数。

方法声明的返回类型指定了由该方法计算和返回的值的类型，如果该方法并不返回值，则其返回类型为 void。

一个方法的名称和形参列表定义了该方法的签名，具体地讲，一个方法的签名由它的名称以及它的形参的个数、修饰符和类型组成。返回类型不是方法签名的组成部分，形参的名称也不是方法签名的组成部分。

 **注意：**一个方法的返回类型和它的形参列表中所引用的各个类型必须至少具有与该方法本身相同的可访问性。

对于 abstract 和 extern 方法，方法主体只包含一个分号，对于所有其他方法，方法主体由一个块组成，该块指定了在调用方法时要执行的语句。

方法的名称必须与在同一个类中声明的所有其他非方法成员的名称都不相同。此外，一个方法的签名必须与在同一个类中声明的所有其他方法的签名都不相同，并且在同一类中声明的两个方法的签名不能只有 ref 和 out 不同。

**例 7.05** 声明一个 public 类型的无返回值方法 method，代码如下。

```
public void method()
{
    Console.WriteLine("方法声明");
}
```

### 7.3.4 方法的分类

方法分为静态方法和非静态方法，如果方法声明中含有 static 修饰符，则称该方法为静态方法；如果方法声明中不含有 static 修饰符，则称该方法为非静态方法。下面分别对静态方法和非静态方法进行介绍。

#### 1. 静态方法

静态方法不对特定实例进行操作，在静态方法中引用 this 会导致编译错误。

**例 7.06** 创建一个控制台应用程序，其中定义了一个静态的方法 Add，该方法中有两个参数，其返回值类型为 int，它主要用来实现两个整数相加的功能，然后在主方法 Main 中使用类名直接调用自定义的静态方法，并传递两个参数。程序代码如下。（实例位置：光盘\mr\07\sl\7.06）

```
public static int Add(int x, int y)           //定义一个静态方法
{
    return x + y;
}
static void Main(string[] args)
{
    Console.WriteLine("结果为：" + Program.Add(3, 5));   //使用类名调用静态方法
    Console.ReadLine();
}
```

按 Ctrl+F5 键查看运行结果，如图 7.2 所示。

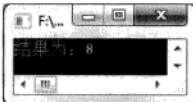


图 7.2 静态方法的使用实例运行结果图

#### 2. 非静态方法

非静态方法是指对类的某个给定的实例进行操作，而且可以用 this 来访问该方法。

**例 7.07** 创建一个控制台应用程序，其中定义了一个非静态的方法 Add，该方法中有两个参数，其返回值类型为 int，它主要用来实现两个整数相加的功能，然后在主方法 Main 中创建 Program 类的一个对象，并使用该对象名调用自定义的非静态方法，并向 Add 方法中传递两个参数。程序代码如下。（实例位置：光盘\mr\07\sl\7.07）

```
public int Add(int x, int y)           //定义一个非静态方法
{
    return x + y;
}
static void Main(string[] args)
{
    Program program = new Program();      //创建对象
    Console.WriteLine("结果为：" + program.Add(3, 5));   //使用对象调用定义的非静态方法
    Console.ReadLine();
}
```

说明：在调用非静态方法时，也可以使用 this 关键字。

按 Ctrl+F5 键查看运行结果，如图 7.3 所示。



图 7.3 非静态方法的使用实例运行结果图

### 7.3.5 重载方法的实现

方法重载是指调用同一方法名，但各方法中参数的数据类型、个数或顺序不同。只要类中有两个以上的同名方法，但是使用的参数类型、个数或顺序不同，调用时，编译器就可以判断在哪种情况下调用哪种方法。

**例 7.08** 创建一个控制台应用程序，其中定义了一个重载方法 Add，并在 Main 方法中分别调用其各种重载形式对传入的参数进行计算。程序代码如下。（实例位置：光盘\mr\07\sl\7.08）

```
public static int Add(int x, int y)           // 定义一个静态方法 Add，返回值为 int 类型，有两个 int 类型的参数
{
    return x + y;
}
public double Add(int x, double y)            // 重新定义方法 Add，它与第一个方法的返回值类型及参数类型不同
{
    return x + y;
}
public int Add(int x, int y, int z)           // 重新定义方法 Add，它与第一个方法的参数个数不同
{
    return x + y + z;
}
static void Main(string[] args)
{
    Program program = new Program(); // 创建对象
    int x = 3;
    int y = 5;
    int z = 7;
    double y2 = 5.5;
    // 根据传入的参数类型及参数个数的不同调用不同的 Add 重载方法
    Console.WriteLine(x + " + " + y + " = " + Program.Add(x, y));
    Console.WriteLine(x + " + " + y2 + " = " + program.Add(x, y2));
    Console.WriteLine(x + " + " + y + " + " + z + " = " + program.Add(x, y, z));
    Console.ReadLine();
}
```

按 Ctrl+F5 键查看运行结果，如图 7.4 所示。

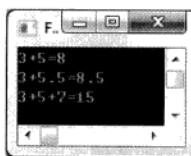


图 7.4 方法的重载实例运行结果图

## 7.4 枚举类型

枚举是用户定义的整型类型，其基础类型可以是除 Char 外的任何整型。枚举是值类型的一种特殊形式，它为基础基元类型的值提供替代名称，如果没有显式声明基础类型，则默认为 Int32 类型。

### 7.4.1 枚举类型概述

枚举类型由名称、基础类型和一组字段组成，基础类型必须是一个内置的有符号（或无符号）整数类型（如 Byte、Int32 或 UInt64 等）；字段为静态文本字段，其中的每一个字段都表示常数。同一个值可以分配给多个字段，当出现这种情况时，必须将其中某个值标记为主要枚举值，以便进行反射和字符串转换。

从宏观上来看，创建枚举可以节省大量的时间，减少许多麻烦。枚举的优点具体如下。

- ☒ 枚举可以使代码更容易维护，有助于确保为变量指定合法的、期望的值。
- ☒ 枚举使代码更加清晰，允许用描述性的名称表示整数值，而不是用含义模糊的数来表示。
- ☒ 枚举式的代码更容易输入。在为枚举类型的实例赋值时，Visual Studio 2008 开发环境会通过 IntelliSense 弹出一个包含可接受值的列表框，减少了按键次数，并能够让程序开发人员看到可能的值。

**技巧：**编写与日期相关的应用程序时，经常需要使用年、月、日、星期等日期数据，可以将这些数据组织成多个不同名称的枚举类型。

### 7.4.2 枚举类型的声明

在 C# 中使用关键字 enum 声明枚举，其形式如下。

```
enum 枚举名
{
    list1=value1,
    list2=value2,
    list3=value3,
    ...
    listN=valueN,
}
```

其中，大括号 {} 中的内容为枚举值列表，每个枚举值均对应一个枚举值名称，value1～valueN 为整型数据类型，list1～listN 则为枚举值的标识名称。

**说明：**在声明一个枚举时，需要指定该枚举中可以包含的一组可接受的实例值。

**例 7.09** 声明一个控制用户权限的枚举，其中包括总经理、部门经理和普通员工，代码如下。

```
enum POP
{
    总经理 = 0,
    部门经理 = 1,
    普通员工 = 2
}
```

注意：当程序中使用枚举时，通常有以下限制。

- ① 枚举不能定义自己的方法。
- ② 枚举中不能实现接口。
- ③ 枚举中不能定义属性或事件。

### 7.4.3 枚举类型的使用

下面通过一个实例来演示如何使用枚举类型。

**例 7.10** 创建一个控制台应用程序，程序首先通过 enum 关键字建立一个枚举，枚举值名称分别代表一周的 7 天，如果枚举值名称为 Sun，说明其代表的是一周中的星期日，设其枚举值为 0，依此类推；然后，声明一个 int 类型的变量 k，用于获取当前表示的日期是星期几；最后使用 switch 语句判断并输出当天是星期几。代码如下。（实例位置：光盘\mr\07\sl\7.10）

```

class Program
{
    enum MyDate //使用 enum 创建枚举
    {
        Sun = 0, //设置枚举值名称 Sun, 枚举值为 0
        Mon = 1, //设置枚举值名称 Mon, 枚举值为 1
        Tue = 2, //设置枚举值名称 Tue, 枚举值为 2
        Wed = 3, //设置枚举值名称 Wed, 枚举值为 3
        Thr = 4, //设置枚举值名称 Thr, 枚举值为 4
        Fri = 5, //设置枚举值名称 Fri, 枚举值为 5
        Sat = 6 //设置枚举值名称 Sat, 枚举值为 6
    }
    static void Main(string[] args)
    {
        int k = (int)DateTime.Now.DayOfWeek; //获取代表星期几的返回值
        switch (k)
        {
            //如果 k 等于枚举变量 MyDate 中的 Sun 的枚举值，则输出今天是星期日
            case (int)MyDate.Sun: Console.WriteLine("今天是星期日"); break;
            //如果 k 等于枚举变量 MyDate 中的 Mon 的枚举值，则输出今天是星期一
            case (int)MyDate.Mon: Console.WriteLine("今天是星期一"); break;
            //如果 k 等于枚举变量 MyDate 中的 Tue 的枚举值，则输出今天是星期二
            case (int)MyDate.Tue: Console.WriteLine("今天是星期二"); break;
            //如果 k 等于枚举变量 MyDate 中的 Wed 的枚举值，则输出今天是星期三
            case (int)MyDate.Wed: Console.WriteLine("今天是星期三"); break;
            //如果 k 等于枚举变量 MyDate 中的 Thr 的枚举值，则输出今天是星期四
            case (int)MyDate.Thr: Console.WriteLine("今天是星期四"); break;
            //如果 k 等于枚举变量 MyDate 中的 Fri 的枚举值，则输出今天是星期五
            case (int)MyDate.Fri: Console.WriteLine("今天是星期五"); break;
            //如果 k 等于枚举变量 MyDate 中的 Sat 的枚举值，则输出今天是星期六
            case (int)MyDate.Sat: Console.WriteLine("今天是星期六"); break;
        }
        Console.ReadLine();
    }
}

```

程序运行结果如图 7.5 所示。



图 7.5 运行结果

## 7.5 泛型及其使用

泛型在面向对象的程序开发中具有广泛的应用，是 C# 语言面向对象思想的一个重要体现，它其实就是一种将类型参数化以达到代码复用为目的的数据类型。泛型参数化类型，使类型抽象化，从而使其对外表现出更加灵活的功能。

### 7.5.1 泛型概述

泛型是一种用于处理算法、数据结构的编程方法，其目标是采用广泛适用和可交互性的形式来表示算法和数据结构，以使它们能够直接用于软件构造。泛型类、结构、接口、委托和方法可以根据它们存储和操作的数据类型来进行参数化。泛型能在编译时提供强大的类型检查，减少数据类型之间的显示转换、装箱操作和运行时的类型检查等。泛型类和泛型方法同时具备可重用性、类型安全和效率高等特性，这是非泛型类和非泛型方法无法具备的。

**技巧：**泛型通常用在集合和在集合上运行的方法中。

### 7.5.2 类型参数 T

泛型的类型参数 T 可以看作是一个占位符，它不是一种类型，而仅代表了某种可能的类型。在定义泛型时，T 出现的位置可以在使用时用任何类型来代替。类型参数 T 的命名规则如下。

- 使用描述性名称命名泛型类型参数，除非单个字母名称完全可以让人了解它表示的含义，而描述性名称不会有更多的意义。

**例 7.11** 使用代表一定意义的单词作为类型参数 T 的名称，代码如下。

```
public interface ISessionChannel<Session>
public delegate TOutput Converter<Input, Output>
```

- 将 T 作为描述性类型参数名的前缀。

**例 7.12** 使用 T 作为类型参数名的前缀，代码如下。

```
public interface ISessionChannel<TSession>
```

```
{
    TSession Session { get; }
}
```

### 7.5.3 泛型接口的声明及使用

泛型接口的声明形式如下。

```
interface 【接口名】<T>
```

```
{
```

**【接口体】**

}

在声明泛型接口时，与声明一般接口的唯一区别是增加了一个<T>。一般来说，声明泛型接口与声明非泛型接口遵循相同的规则。泛型类型声明所实现的接口必须对所有可能的构造类型都保持唯一，否则就无法确定该为某些构造类型调用哪个方法。

**例 7.13** 创建一个控制台应用程序，首先创建一个 Factory 类，在此类中建立一个 CreateInstance 方法；然后再创建一个泛型接口，在该泛型接口中调用 CreateInstance 方法。根据类型参数 T 获取其类型，代码如下。（实例位置：光盘\mr\07\sl\7.13）

```
public interface IGenericInterface<T> //创建一个泛型接口
{
    T CreateInstance(); //接口中声明 CreateInstance 方法
}

//实现上面泛型接口的泛型类
//派生约束 where T : TI (T 要继承自 TI)
//构造函数约束 where T : new() (T 可以创建)
public class Factory<T, TI> : IGenericInterface<TI> where T : TI, new()
{
    public TI CreateInstance() //创建一个公共方法 CreateInstance
    {
        return new T();
    }
}
class Program
{
    static void Main(string[] args)
    {
        //创建接口
        IGenericInterface<System.ComponentModel.IListSource> factory =
new Factory<System.Data.DataTable, System.ComponentModel.IListSource>();
        //输出指定泛型的类型
        Console.WriteLine(factory.CreateInstance().GetType().ToString());
        Console.ReadLine();
    }
}
```

程序运行结果如图 7.6 所示。



图 7.6 泛型接口的使用

## 7.6 结构的定义及使用

结构就是几个数据组成的数据结构，它与类共享几乎所有相同的语法，但结构比类受到的限制更多，本节将对结构进行详细讲解。

### 7.6.1 结构概述

结构是一种值类型，通常用来封装一组相关的变量，结构中可以包括构造函数、常量、字段、方法、属性、运算符、事件和嵌套类型等，但如果要同时包括上述几种成员，则应考虑使用类。

结构具有以下特点。

- 结构是值类型。
- 向方法传递结构时，结构是通过传值方式传递的，而不是作为引用传递的。
- 结构的创建可以不使用 new 运算符。
- 结构可以声明构造函数，但它们必须带参数。
- 一个结构不能从另一个结构或类继承。所有结构都直接继承自 System.ValueType，后者继承自 System.Object。
- 结构可以实现接口。
- 在结构中初始化实例字段是错误的。

 **说明：**由于结构的副本由编译器自动创建和销毁，因此不需要使用默认构造函数和析构函数。实际上，编译器通过为所有字段赋予默认值来实现默认构造函数。

### 7.6.2 结构的定义

在 C# 中，使用 struct 关键字来定义结构，语法如下。

结构修饰符 struct 结构名

```
{  
}
```

**例 7.14** 下面定义一个矩形结构，该结构中定义了矩形的宽和高，并自定义了一个 Area 方法，用来计算矩形的面积。程序代码如下。

```
public struct Rect //定义一个矩形结构  
{  
    public double width; //矩形的宽  
    public double height; //矩形的高  
    ///<summary>  
    ///计算矩形面积  
    ///</summary>  
    ///<returns>矩形面积</returns>  
    public double Area()  
    {  
        return width * height;  
    }  
}
```

### 7.6.3 结构的使用

结构通常用于较小的数据类型，下面通过一个实例说明如何在程序中使用结构。

**例 7.15** 创建一个控制台应用程序，其中定义一个矩形结构，在该结构中定义了矩形的宽和高，同时定义了一个构造函数，在该构造函数中有两个参数，用来初始化矩形的宽和高，接着自定义了一个 Area 方法，

用来计算矩形的面积。然后在 Main 方法中创建矩形结构的一个对象，并通过调用结构中的自定义方法计算矩形的面积。最后使用矩形结构的构造函数再次创建矩形结构的一个对象，并再次调用结构中的自定义方法计算矩形的面积。程序代码如下。（实例位置：光盘\mr\07\sl\7.15）

```

public struct Rect
{
    public double width; //矩形的宽
    public double height; //矩形的高
    public Rect(double x, double y) //构造函数，初始化矩形的宽和高
    {
        width = x;
        height = y;
    }
    public double Area() //计算矩形面积
    {
        return width * height;
    }
}
static void Main(string[] args)
{
    Rect rect1; //创建矩形结构
    rect1.width = 5; //为矩形的宽赋值
    rect1.height = 3; //为矩形的高赋值
    Console.WriteLine("矩形面积为：" + rect1.Area());
    Rect rect2 = new Rect(6, 4); //使用构造函数创建矩形结构
    Console.WriteLine("矩形面积为：" + rect2.Area());
    Console.ReadLine();
}

```

按 Ctrl+F5 键查看运行结果，如图 7.7 所示。

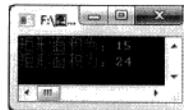


图 7.7 结构的使用实例运行结果图

## 7.7 类与对象详解

类是一种数据结构，它可以包含数据成员（常量和域）、函数成员（方法、属性、事件、索引器、运算符、构造函数和析构函数）和嵌套类型等。类支持继承，继承是一种使子类（派生类）可以对基类进行扩展和专用化的机制。本节将对类和对象进行详细讲解。

### 7.7.1 类的概念

类是对象在面向对象编程语言中的反映，是相同对象的集合，其描述了一系列在概念上有相同含义的对象，并为这些对象统一定义了编程语言上的属性和方法。例如，车是一个类，自行车、汽车、火车也是类，但是自行车、汽车、火车都属于车这个类的子类（派生类），因为它们有共同的特点——都是交通工具。

具、都有轮子、都可以运输；而汽车有发动机，需要通过燃烧汽油来提供动力，这是它和自行车不同的地方，是汽车类自己的属性，也是所有汽车共同的属性，所以汽车也是一个类；而具体到某个汽车就是一个对象了，例如，车牌照为吉 A2154\*\*\*\*的黑色奔驰车。简而言之，类是C#中功能最为强大的数据类型，像结构一样，类也定义了数据类型的数据和行为，然后，程序开发人员可以创建作为此类的实例对象。与结构不同，类支持继承，而继承是面向对象编程的基础部分。

## 7.7.2 类的声明

在C#中，类是使用**class**关键字来声明的，语法如下。

**类修饰符 class 类名**

```
{  
}
```

**例 7.16** 下面以汽车为例声明一个类，代码如下。

```
public class Car  
{  
    public int number;           //编号  
    public string color;         //颜色  
    private string brand;        //厂家  
}
```

**public** 是类的修饰符，下面介绍常用的几个类修饰符。

- new**: 仅允许在嵌套类声明时使用，表明类中隐藏了由基类中继承而来的、与基类中同名的成员。
- public**: 不限制对该类的访问。
- protected**: 只能从其所在类和所在类的子类（派生类）进行访问。
- internal**: 只有其所在类才能访问。
- private**: 只有.NET中的应用程序或库才能访问。
- abstract**: 抽象类，不允许建立类的实例。
- sealed**: 密封类，不允许被继承。

## 7.7.3 构造函数和析构函数

构造函数和析构函数是类中比较特殊的两种成员函数，主要用来对对象进行初始化和回收对象资源。一般来说，对象的生命周期从构造函数开始，到析构函数结束。如果一个类含有构造函数，则在创建该类的对象时就会调用；如果含有析构函数，则会在销毁对象时调用。构造函数的名字和类名相同，析构函数和构造函数的名字相同，但析构函数要在名字前加一个波浪号(~)。当退出含有该对象的成员时，析构函数将自动释放这个对象所占用的内存空间。本节将详细介绍如何在程序中使用构造函数和析构函数。

### 1. 构造函数的概念及使用

构造函数是在创建给定类型的对象时执行的类方法，其具有与类相同的名称，它通常初始化新对象的数据成员。

**例 7.17** 创建一个控制台应用程序，在**Program**类中定义了3个int类型的变量，分别用来表示加数、被加数和加法的和，然后声明**Program**类的一个构造函数，并在该构造函数中为加法的和赋值，最后在**Main**方法中创建**Program**类的对象，并输出加法的和。程序代码如下。（实例位置：光盘\mr\07\sl\7.17）

```
class Program
```

```
{
```

```

public int x = 3;           //定义 int 型变量，作为加数
public int y = 5;           //定义 int 型变量，作为被加数
public int z = 0;           //定义 int 型变量，记录加法运算的和
public Program()
{
    z = x + y;             //在构造函数中为和赋值
}
static void Main(string[] args)
{
    Program program = new Program();      //使用构造函数创建 Program 对象
    Console.WriteLine("结果：" + program.z); //使用创建的 Program 对象输出加法运算的和
    Console.ReadLine();
}
}

```

程序运行结果如图 7.8 所示。



图 7.8 运行结果

**说明：**不带参数的构造函数称为“默认构造函数”。无论何时，只要使用 new 运算符创建对象，并且不为 new 提供任何参数，就会调用默认构造函数。

## 2. 析构函数的概念及使用

析构函数是以类名加～来命名的。.NET Framework 类库具有垃圾回收功能，当某个类的实例被认为是不再有效，并符合析构条件时，.NET Framework 类库的垃圾回收功能就会调用该类的析构函数实现垃圾回收。

**例 7.18** 创建一个控制台应用程序，其中在 Program 类中声明了其析构函数，并在该析构函数中输出了一个字符串，这时在 Main 方法中创建 Program 类的对象，运行程序时，自动调用析构函数，并实现析构函数中的功能。程序代码如下。（实例位置：光盘\mr\07\sh\7.18）

```

class Program
{
    ~Program()           //析构函数
    {
        Console.WriteLine("析构函数自动调用"); //输出一个字符串
        Console.ReadLine();
    }
    static void Main(string[] args)
    {
        Program program = new Program(); //创建 Program 对象
    }
}

```

程序运行结果为“析构函数自动调用”。

**注意：**一个类中只能有一个析构函数，并且无法调用该析构函数，它是被自动调用的。

## 7.7.4 对象的声明和创建

对象是具有数据、行为和标识的编程结构，它是面向对象应用程序的一个重要组成部分，这个组成部

分封装了部分应用程序，这部分应用程序可以是一个过程、一些数据或一些更抽象的实体。

对象包含变量成员和方法类型，它所包含的变量组成了存储在对象中的数据，而其包含的方法可以访问对象的变量。略为复杂的对象可能不包含任何数据，而只包含方法，并使用方法表示一个过程。

C#中的对象是从类创建，这表示创建类的一个实例，“类的实例”和对象表示相同的含义，但需要注意的是，“类”和“对象”是完全不同的概念。

C#中，.NET Framework 类库中的所有类型都是对象，如 String 类型的实例就可以称之为对象。

用属性和字段可以访问对象中包含的数据。对象数据用来区分不同的对象，同一个类的不同对象可能在属性和字段中存储了不同的值。字段和属性都可以输入，通常把信息存储在字段和属性中，但是属性和字段是不同的，属性不能直接访问数据，而字段可以直接访问数据。在属性中可以添加对数据访问的限制，例如有一个 int 类型的属性，可以限制它只能存储 1~5 的数字，但如果用字段就可以存储任何 int 类型的数值。

**技巧：**最好在访问状态时提供属性，而不是字段，因为属性可以更好地控制访问过程和读写权限。除此之外，属性的可访问性确定了什么代码可以访问这些成员，可以声明为公有、私有或者其他更为复杂的方式。

**例 7.19** 创建一个控制台应用程序，其中定义了一个 MyClass 类，并在该类中定义了 3 个 int 类型的变量，分别用来记录加数、被加数和加法的和的初始值，然后使用这 3 个变量定义 3 个属性，分别用来表示加数、被加数和加法的和，这 3 个属性都设置为可读可写。在 Program 类中，定义两个 int 类型的变量，用来作为加数和被加数，然后创建 MyClass 类的一个对象，并通过该对象设置它的 3 个属性，最后定义一个 int 类型的变量，并使用声明的对象访问 MyClass 类中的属性。程序代码如下。(实例位置：光盘\mr\07\sl\7.19)

```
class MyClass
{
    private int x = 0;                                //定义 int 型变量，作为加数
    private int y = 0;                                //定义 int 型变量，作为被加数
    private int z = 0;                                //定义 int 型变量，记录加法运算的和
    public int X
    {
        get
        {
            return x;
        }
        set
        {
            x = value;
        }
    }
    public int Y                                     //被加数
    {
        get
        {
            return y;
        }
        set
        {
            y = value;
        }
    }
}
```

```

    }
}

public int Z //和
{
    get
    {
        return z;
    }
    set
    {
        z = value;
    }
}
public MyClass() //构造函数
{
}
}

class Program
{
    static void Main(string[] args)
    {
        int x = 3;
        int y = 5;
        MyClass myclass = new MyClass(); //创建 MyClass 对象
        myclass.X = x; //通过对象设置类中的属性 X
        myclass.Y = y; //通过对象设置类中的属性 Y
        myclass.Z = myclass.X + myclass.Y; //定义一个 int 型变量，并通过对象访问类中的属性 Z
        int z = myclass.Z;
        Console.WriteLine("运行结果为：" + z);
        Console.ReadLine();
    }
}

```

程序运行结果如图 7.9 所示。



图 7.9 运行结果

## 7.8 面向对象特性之封装

在 C# 中可使用类来实现数据封装的效果，这样就可以使数据与方法封装成单一元素，以便于通过方法存取数据。除此之外，还可以控制数据的存取方式。

### 7.8.1 封装概述

在面向对象编程中，大多数都是以类作为数据封装的基本单位。类将数据和操作数据的方法结合成一

个单位。设计类时，不希望直接存取类中的数据，而是希望通过方法来存取数据，这样就可以达到封装数据的目的，方便以后的维护升级，另外也可以在操作数据时多一层判断。

此外，封装还可以解决数据存取的权限问题，可以使用封装将数据隐藏起来，形成一个封闭的空间，然后可以设置哪些数据只能在这个空间中使用，哪些数据可以在空间外部使用。如果一个类中包含敏感数据，有些人可以访问，有些人不能访问，如果不对这些数据的访问加以限制，后果将会非常严重。所以在编写程序时，要对类的成员使用不同的访问修饰符，从而定义他们的访问级别。

## 7.8.2 封装的实现

在 C# 中可以使用类来实现数据封装的效果，这样就可以使数据与方法封装成单一元素，以便于通过方法存取数据。除此之外，还可以控制数据的存取方式。下面通过一个实例演示如何实现面向对象编程的封装性。

**例 7.20** 创建一个控制台应用程序，其中自定义了一个 MyClass 类，该类用来封装加数和被加数属性。然后自定义一个 Add 方法，该方法用来返回该类中两个 int 型属性的和。在 Program 主程序类中创建自定义类的对象，并分别为 MyClass 类中的两个属性赋值。最后调用 MyClass 类中的自定义方法 Add 返回两个属性的和。程序代码如下。（实例位置：光盘\mr\07\sl\7.20）

```
class MyClass
{
    private int x = 0; //定义 int 型变量，作为加数
    private int y = 0; //定义 int 型变量，作为被加数
    public int X //加数
    {
        get
        {
            return x;
        }
        set
        {
            x = value;
        }
    }
    public int Y //被加数
    {
        get
        {
            return y;
        }
        set
        {
            y = value;
        }
    }
    public int Add() //求和
    {
        return X + Y;
    }
}
```

```

class Program
{
    static void Main(string[] args)
    {
        MyClass myclass = new MyClass();           // 创建 MyClass 的对象
        myclass.X = 3;                            // 为 MyClass 类中的属性赋值
        myclass.Y = 5;                            // 为 MyClass 类中的属性赋值
        Console.WriteLine("运行结果为: " + myclass.Add()); // 调用 MyClass 类中的 Add 方法求和
        Console.ReadLine();
    }
}

```

程序运行结果如图 7.10 所示。



图 7.10 运行结果

## 7.9 面向对象特性之继承

继承是面向对象编程最重要的特性之一，任何类都可以从另外一个类继承。在面向对象编程中，被继承的类称为父类或基类，而派生出的新类称为子类。

### 7.9.1 继承概述

C# 中提供了类的继承机制，但只支持单继承，而不支持多重继承，即在 C# 中一次只允许继承一个类，而不能同时继承多个类；若要实现多重继承，则需要使用接口。

C# 中实现继承的语法格式如下。

```
class DerivedClass: BaseClass {}
```

**说明：**继承类时，必须在子类和基类之间用冒号（:）。

利用类的继承机制，程序开发人员可以在已有类的基础上构造新类，这一性质使得类支持分类的概念，例如用户可以通过增加、修改或替换类中的方法对这个类进行扩充，以适应不同的应用要求。在日常生活中很多事物都具有条理性，那是因为它们有着很好的层次分类，如果不用层次分类，则需要对每个对象都定义其所有的性质。使用继承后，每个对象就可以只定义自己的特殊性质，每一层的对象只需定义本身自身的性质，其他性质可以从上一层继承下来。

在继承一个类时，类成员的可访问性是一个重要的问题。子类（派生类）不能访问基类的私有成员，但是可以访问其公共成员，这就是说，只要使用 `public` 声明类成员，就可以让一个类成员被基类和子类（派生类）同时访问，同时也可以被外部的代码访问。

为了解决基类成员访问的问题，C# 还提供了另外一种可访问性，即只有子类（派生类）才能访问 `protected` 成员，而基类和外部代码均不能访问 `protected` 成员。

除了成员的保护级别外，还可以为成员定义其继承行为。基类的成员可以是虚拟的，成员可以由继承它的类重写。子类（派生类）可以提供成员的其他执行代码，这种执行代码不会删除原来的代码，仍可以

在类中访问原来的代码，但外部代码不能访问它们。如果没有提供其他执行方式，则外部代码就直接访问基类中成员的执行代码。

另外，基类还可以定义为抽象类。抽象类不能直接创建，要使用抽象类就必须继承这个类，然后再创建。

### 7.9.2 单继承的使用

单继承一般用于类之间的继承，C#中的类只支持单继承，实现单继承时需使用“子类：基类”格式。下面通过一个实例讲解如何实现单继承。

**例 7.21** 创建一个控制台应用程序，其中自定义了一个 MyClass 类。该类中定义了一个虚方法 Add，用来计算两个整数的和。在 Program 主程序类中，首先使该类继承于 MyClass 类，然后在主程序类中重写 MyClass 类中的虚方法 Add，并使其用来计算 3 个数的和，最后在 Main 方法中调用重写的 Add 方法计算 3 个数的和并输出。程序代码如下。（实例位置：光盘\mr\07\sl\7.21）

```
class MyClass
{
    public virtual int Add(int x, int y)          //计算两个数的和
    {
        return x + y;
    }
}
class Program:MyClass
{
    int z = 0;                                //定义一个 int 类型的变量，用来作为第二个被加数
    public override int Add(int x, int y)         //计算 3 个数的和
    {
        return base.Add(x, y) + z;                //调用基类中的 Add 方法
    }
    static void Main(string[] args)
    {
        Program program = new Program();          //创建 Program 对象
        program.z = 8;                            //为变量赋值
        Console.WriteLine("运行结果为：" + program.Add(3, 5)); //使用重写的方法计算 3 个数的和
        Console.ReadLine();
    }
}
```

程序运行结果如图 7.11 所示。

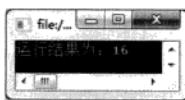


图 7.11 运行结果

技巧：在子类中重写基类中的虚方法时，可以使用 base 关键字调用基类中的虚方法。

### 7.9.3 多重继承的使用

如果要使用多重继承，则需要使用接口，因为 C#中的类只支持单继承，而接口支持多重继承。另外，

在实现多重继承时，继承的多个接口中间需用逗号（,）隔开。

 **说明：** 接口将在本书的第 16 章进行详细讲解，这里首先了解一下如何使用接口实现多重继承。

 **注意：** 实现多重继承时，也可以是继承一个类及多个接口。

下面通过一个实例讲解如何实现多重继承。

**例 7.22** 创建一个控制台应用程序，首先声明一个接口 **IPeople**，并在其中定义 Name 和 Sex 两个属性；然后声明一个 **MyClass** 类，该类中自定义一个 **Show** 方法，用来输出信息；最后使主程序类 **Program** 继承于自定义的接口和类，并在其中实现接口中定义的属性。在 **Main** 方法中创建 **Program** 类的一个对象，并使用该对象创建接口对象，分别为接口中的 Name 属性和 Sex 属性赋值，然后使用 **Program** 对象调用基类中的方法输出信息，并输出接口中两个属性的值。程序代码如下。（实例位置：光盘\mr\07\sl\7.22）

```
interface IPeople
{
    string Name          //姓名
    {
        get;
        set;
    }
    string Sex           //性别
    {
        get;
        set;
    }
}
class MyClass
{
    public void Show()      //输出信息
    {
        Console.WriteLine("人的信息：");
    }
}
class Program : MyClass,IPeople           //多重继承
{
    string name = "";
    string sex = "";
    public string Name          //姓名
    {
        get
        {
            return name;
        }
        set
        {
            name = value;
        }
    }
    public string Sex           //性别
}
```

```

{
    get
    {
        return sex;
    }
    set
    {
        sex = value;
    }
}
static void Main(string[] args)
{
    Program program = new Program();           //创建对象
    IPeople ipeople = program;                 //使用派生对象创建接口 IPeople
    ipeople.Name = "TM";
    ipeople.Sex = "男";
    program.Show();                          //调用基类中的方法
    Console.WriteLine(ipeople.Name + " " + ipeople.Sex);
    Console.ReadLine();
}
}

```

按 Ctrl+F5 键查看运行结果，如图 7.12 所示。



图 7.12 多重继承的使用

## 7.10 面向对象特性之多态

面向对象的编程方式具有封装、继承和多态性等特点。封装可以隐藏实现细节，使得代码模块化；继承可以扩展已经存在的代码模块，它们的目的都是为了代码重用；而多态是为了减少代码之间的紧密耦合，增加应用程序的灵活性。

### 7.10.1 多态概述

多态性可以简单地概括为“一个接口，多种方法”，其是在程序运行的过程中才决定调用的方法，它是面向对象编程的核心概念。

多态使得子类（派生类）的实例可以直接赋予基类的对象（这里不需要进行强制类型转换），然后直接就可以通过该对象调用子类（派生类）。

了解了多态性的基本概念后，那么多态性到底有什么作用呢？封装可以隐藏实现细节，使得代码模块化；继承可以扩展已经存在的代码模块，它们的目的都是为了代码重用；而多态则是为了实现另一个目的——接口重用，因为接口是最耗费时间的资源，实质上设计一个接口要比设计一堆类要显得更有效率。

**技巧：**在派生于同一个类的不同对象上执行任务时，多态是一种极为有效的技巧，使用的代码最少。可以首先将一组对象放到一个数组中，然后调用它们的方法，在这种情况下多态的作用就体现出来了，这些对象不必是相同类型的对象。当然，如果它们都继承自某个类，则可以把这些子类（派生类）都放到一个数组中。如果这些对象都有同名方法，就可以调用每个对象的同名方法。

### 7.10.2 多态的实现

在 C# 中，类的多态性是通过在子类（派生类）中重写基类的虚方法或函数成员来实现的，下面通过一个实例讲解如何实现多态性。

**例 7.23** 创建一个控制台应用程序，其中自定义了一个 MyClass1 类，该类中定义了一个虚方法 Add，用来计算两个整数的和，然后自定义一个 MyClass2 类，该类继承于 MyClass1 类，在 MyClass2 类中重写 MyClass1 类中的虚方法。在 Program 主程序类中，首先创建子类 MyClass2 的一个对象，然后使用该对象创建基类 MyClass1 的一个对象，这时，使用创建的 MyClass2 和 MyClass1 的两个对象都可以调用子类 MyClass2 中的重写方法。程序代码如下。（实例位置：光盘\mr\07\sl\7.23）

```

class MyClass1 //自定义类
{
    private int x = 0; //定义 int 型变量，作为加数
    private int y = 0; //定义 int 型变量，作为被加数
    public int X //加数
    {
        get
        {
            return x;
        }
        set
        {
            x = value;
        }
    }
    public int Y //被加数
    {
        get
        {
            return y;
        }
        set
        {
            y = value;
        }
    }
    public virtual int Add() //定义一个 virtual 类型的方法，以便在子类（派生类）中重写该方法
    {
        return X + Y;
    }
}
class MyClass2 : MyClass1 //自定义类，该类继承自 MyClass1
{
    public override int Add() //重写基类中的虚方法
}

```

```

    {
        int x=5;
        int y=7;
        return x + y;
    }
}
class Program
{
    static void Main(string[] args)
    {
        MyClass2 myclass2 = new MyClass2();           //创建 MyClass2 的对象
        //使用子类（派生类）MyClass2 的对象创建基类 MyClass1 的对象
        MyClass1 myclass1 = (MyClass1)myclass2;
        myclass1.X = 3;                            //为 MyClass1 类中的属性赋值
        myclass1.Y = 5;                            //为 MyClass1 类中的属性赋值
        Console.WriteLine(myclass2.Add());          //调用子类（派生类）中的方法
        Console.WriteLine(myclass1.Add());          //同样调用子类（派生类）中的方法
        Console.ReadLine();
    }
}

```

程序运行结果如图 7.13 所示。



图 7.13 运行结果

**说明：**当子类（派生类）从基类继承时，它会获得基类的所有方法、字段、属性和事件。若要更改基类的数据和行为，有两种选择：可以使用新的派生成员替换基成员，或者可以重写虚拟的基成员。上面的实例重写了基类中的虚方法，另外，开发人员还可以使用新的派生成员替换基类的成员，这时需要使用 new 关键字。如果基类定义了一个方法、字段或属性，则 new 关键字用于在子类（派生类）中创建该方法、字段或属性的新定义。new 关键字放置在要替换的类成员的返回类型之前，例如，上面实例中重写基类中虚方法的代码也可以替换为下面形式。

```

class MyClass2 : MyClass1
{
    public new int Add()      //重写基类中的虚方法
    {
        int x=5;
        int y=7;
        return x + y;
    }
}

```

**注意：**① virtual 修饰符不能与 private、static、abstract 或 override 修饰符同时使用。  
 ② override 修饰符不能与 new、static 或 virtual 修饰符同时使用，并且重写方法只能用于重写基类中的虚方法。  
 ③ 在 C# 中，继承、虚方法和重写方法组合在一起才能实现多态性。

## 7.11 照猫画虎——基本功训练

### 7.11.1 基本功训练 1——使用属性存储用户编号和姓名

视频讲解：光盘\mr\07\lx\使用属性存储用户编号和姓名.exe

实例位置：光盘\mr\07\zmhh\01

本实例使用属性来对用户编号和用户姓名进行存储，并将存储的用户编号和用户姓名显示到 Windows 窗体上。新建一个 Windows 应用程序，在默认窗体 Form1 中添加两个 Label 控件，分别用来显示用户编号和姓名，主要代码如下。

```

private string id = ""; //定义一个 string 类型的变量，用来记录用户编号
private string name = ""; //定义一个 string 类型的变量，用来记录用户姓名
public string ID //定义用户编号属性，该属性为可读可写属性
{
    get
    {
        return id;
    }
    set
    {
        id = value;
    }
}
public string Name //定义用户姓名属性，该属性为可读可写属性
{
    get
    {
        return name;
    }
    set
    {
        name = value;
    }
}
private void Form1_Load(object sender, EventArgs e) //窗体的 Load 事件，在窗体运行时触发
{
    ID = "BH001"; //为用户编号属性赋值
    Name = "MR1"; //为用户姓名属性赋值
    lab_First.Text = ID + " " + Name; //显示用户编号和用户姓名
    ID = "BH002"; //重新为用户编号属性赋值
    Name = "MR2"; //重新为用户姓名属性赋值
    lab_Second.Text = ID + " " + Name; //显示用户编号和用户姓名
}

```

实例运行效果如图 7.14 所示。

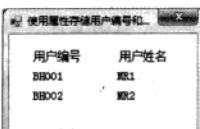


图 7.14 使用属性存储用户编号和姓名

**说明：**通常把信息存储在字段和属性中，但是属性和字段是不同的，体现在：属性不能直接访问数据，而字段可以直接访问数据。

**熊猫画虎：**以上面的实例为基础，添加两个属性，分别用来表示用户性别和用户年龄。(20分)(实例位置：光盘\mr\07\zmhh\01\_zmhh)

### 7.11.2 基本功训练 2——通过定义方法求一个数的平方

**视频讲解：**光盘\mr\07\lx\通过定义方法求一个数的平方.exe

**实例位置：**光盘\mr\07\zmhh\02

本实例将通过自定义的方法来计算一个数的平方。新建一个 Windows 窗体应用程序，在默认窗体 Form1 中添加两个 TextBox 控件，分别用来输入一个数和显示输入数的平方值；添加一个 Button 控件，用来对输入的数进行求平方运算，主要代码如下。

```
public double SquareNum(double num)
{
    return num * num;                                //求一个数的平方
}
private void button1_Click(object sender, EventArgs e)
{
    string strNum = textBox1.Text.Trim();            //记录 TextBox 文本框中的内容
    if (strNum != "")                                //判断是否输入了数据
    {
        try
        {
            textBox2.Text = SquareNum(double.Parse(strNum)).ToString(); //调用自定义方法进行求平方运算
        }
        catch
        {
            MessageBox.Show("请输入正确的数字！");
        }
    }
}
```

实例运行效果如图 7.15 所示。

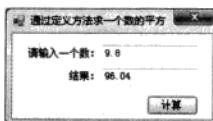


图 7.15 通过定义方法求一个数的平方

**说明：**在 C# 中计算一个数的平方时，也可以直接使用 Math 类的 Pow 方法来实现。

照猫画虎：自定义一个方法，用于实现计算一个数的立方。(20分)(实例位置：光盘\mr\07\zmhh\02\_zmhh)

### 7.11.3 基本功训练 3——使用重载方法实现不同类型数据的计算

视频讲解：光盘\mr\07\lx\使用重载方法实现不同类型数据的计算.exe

实例位置：光盘\mr\07\zmhh\03

在实际工作中，经常会遇到不同类型数据的计算问题，例如，当计算某公司某月的总工资时就有可能遇到这种情况，假设公司员工的工资都是整数，如果没有请假和迟到早退现象，那么就是所有人的工资相加，这时用到的就都是整数类型的数据相加；而如果这个月中出现了请假或迟到早退现象，那么有的员工的工资就可能是小数类型的，这时再计算总工资就有可能是整数和小数或小数和小数进行相加。在 C# 中实现以上功能最快捷的方式就是使用重载方法，通过设置重载方法中要计算的数据类型不同，就能够实现不同类型数据的计算功能。本实例运行效果如图 7.16、图 7.17 和图 7.18 所示。



图 7.16 两个整数类型数的和

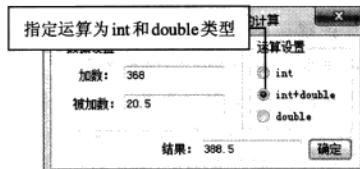


图 7.17 一个整数类型数和一个浮点类型数的和

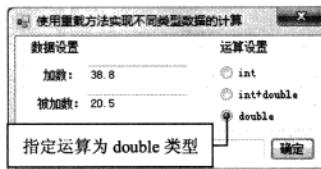


图 7.18 两个浮点类型数的和

实现过程如下。

(1) 新建一个 Windows 应用程序，在默认窗体 Form1 中添加 3 个文本框，分别用来输入加数、被加数和显示和；添加 3 个 RadioButton 控件，分别用来选择要计算的数据类型；添加一个按钮，用来对输入的两个数进行求和运算。

(2) 程序主要代码如下。

```
public static int Add(int x, int y)          // 定义一个静态方法 Add，返回值为 int 类型，有两个 int 类型的参数
{
    return x + y;
}
public double Add(int x, double y)           // 重新定义方法 Add，它与第一个方法的返回值类型及参数类型不同
{
    return x + y;
}
public double Add(double x, double y)         // 重新定义方法 Add，它与第一个方法的返回值类型及参数类型不同
{
    return x + y;
}
```

**照猫画虎：**自定义具有两种重载形式的方法 GetGirth，分别用来计算矩形周长和圆形周长（即圆周率的约等值）。(20分)(实例位置：光盘\mr\07\zmhh\03\_zmhh)

#### 7.11.4 基本功训练4——通过结构计算矩形的面积

■**视频讲解：**光盘\mr\07\lx\通过结构计算矩形的面积.exe

■**实例位置：**光盘\mr\07\zmhh\04

本实例中定义了一个矩形结构，其中定义了矩形的长和宽，并且自定义了一个 Area 方法，用来计算矩形的面积。新建一个 Windows 窗体应用程序，在默认窗体 Form1 中添加 3 个 TextBox 控件，分别用来输入矩形的长、宽和显示矩形的面积；添加一个 Button 控件，用来根据用户输入的长和宽计算矩形的面积，代码如下。

```
public struct Rect //定义一个矩形结构
{
    public double width; //矩形的宽
    public double height; //矩形的高
    public Rect(double x, double y) //构造函数，初始化矩形的宽和高
    {
        width = x;
        height = y;
    }
    public double Area() //计算矩形面积
    {
        return width * height;
    }
}
```

实例运行效果如图 7.19 所示。

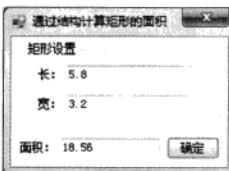


图 7.19 通过结构计算矩形的面积

**照猫画虎：**通过结构计算三角形的面积。提示：三角形的面积等于底乘高的积除以 2。(20分)(实例位置：光盘\mr\07\zmhh\04\_zmhh)

#### 7.11.5 基本功训练5——通过类继承计算梯形面积

■**视频讲解：**光盘\mr\07\lx\通过类继承计算梯形面积.exe

■**实例位置：**光盘\mr\07\zmhh\05

本实例通过类的继承来实现梯形面积的计算，首先定义一个父类，并声明 3 个属性，分别用来存储梯形的上底、下底和高；然后再定义一个继承自父类的子类，并在子类中定义一个计算梯形面积的方法。新建一个 Windows 应用程序，在窗体 Form1 中添加 4 个 TextBox 控件，分别用来输入梯形的上底、下底、高和显示梯形的面积；添加一个 Button 控件，用来根据用户输入的上底、下底和高计算梯形的面积，代码

如下。

```

class Trapezia
{
    private double sd = 0;           //自定义类
    private double xd = 0;           //定义 int 型变量，作为梯形的上底
    private double height = 0;        //定义 int 型变量，作为梯形的下底
    public double SD                //定义 int 型变量，作为梯形的高
    {
        get
        {
            return sd;
        }
        set
        {
            sd = value;
        }
    }
    public double XD                //上底
    {
        get
        {
            return xd;
        }
        set
        {
            xd = value;
        }
    }
    public double Height             //下底
    {
        get
        {
            return height;
        }
        set
        {
            height = value;
        }
    }
}
class TrapeziaArea : Trapezia      //自定义类，该类继承自 Trapezia
{
    public double Area()           //求梯形的面积
    {
        return (SD + XD) * Height / 2;
    }
}

```

实例运行效果如图 7.20 所示。

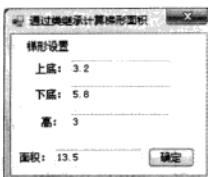


图 7.20 通过类继承计算梯形面积

**说明：**继承类时，必须在子类和基类之间用冒号（:）。

**照猫画虎：**通过继承计算圆形的面积。提示：定义一个基类，并在基类中封装表示圆形半径的属性，然后从基类派生一个子类，并在子类中定义计算圆形面积的方法。（20分）（实例位置：光盘\mr\07\zmhh\05\_zmhh）

照猫画虎栏目分数统计：

照猫画虎题目	1	2	3	4	5	总分数
分数						

## 7.12 情景应用——拓展与实践

### 7.12.1 情景应用 1——通过类的多态性确定人类的说话行为

**视频讲解：**光盘\mr\07\lx\通过类的多态性确定人类的说话行为.exe

**实例位置：**光盘\mr\07\qjyy\01

本实例通过使用类的多态性来确定人类的说话行为，运行本实例，首先在文本框中输入人的姓名，然后单击“确定”按钮，在 RichTextBox 控件中分别显示人的“说汉语”和“说英语”行为。实例运行效果如图 7.21 所示。

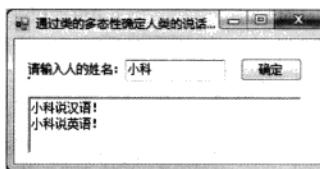


图 7.21 通过类的多态性确定人类的说话行为

实现过程如下。

- (1) 打开 Visual Studio 2008 开发环境，新建一个 Windows 窗体应用程序，并将其命名为 PeopleSpeakByMState。
- (2) 在默认窗体 Form1 中添加一个 TextBox 控件，用来输入人的姓名；添加一个 Button 控件，用来执行人说话操作；添加一个 RichTextBox 控件，用来显示人的说话行为。
- (3) 程序主要代码如下。

```
class People
{
    public virtual void Say(RichTextBox rtbox, string name) //定义一个虚方法，用来表示人的说话行为
```

```

    {
        rtbox.Text += name;                                //输出人的名字
    }
}
class Chinese : People                               //定义派生类，继承于 People 类
{
    public override void Say(RichTextBox rtbox, string name) //重写基类中的虚方法
    {
        base.Say(rtbox, name + "说汉语！\n");
    }
}
class American : People                            //定义派生类，继承于 People 类
{
    public override void Say(RichTextBox rtbox, string name) //重写基类中的虚方法
    {
        base.Say(rtbox, name + "说英语！");
    }
}
private void button1_Click(object sender, EventArgs e)
{
    if (textBox1.Text == "")                         //判断是否输入了姓名
    {
        Console.WriteLine("请输入人的姓名：");
        return;
    }
    richTextBox1.Clear();                           //清空文本框内容
    string strName = textBox1.Text;                //记录用户输入的名字
    People[] people = new People[2];               //声明 People 类型数组
    people[0] = new Chinese();                     //使用第一个派生类的对象初始化数组的第一个元素
    people[1] = new American();                   //使用第二个派生类的对象初始化数组的第二个元素
    for (int i = 0; i < people.Length; i++)        //遍历赋值后的数组
    {
        people[i].Say(richTextBox1,strName);       //根据数组元素调用相应派生类中的重写方法
    }
}

```

**说明：**在使用虚方法实现多态性时，首先在基类中定义虚方法，每一个子类都重写基类的虚方法，采用基类的对象引用子类实例的方法创建对象，这样会产生很多基类的对象，使用每一个基类的对象调用虚方法时会调用不同子类重写基类的方法，这样就实现了多态性。

**DIY：** 定义一个表示车辆的基类，并在该类中封装一个虚方法 Run；然后从该基类派生出一个轿车类和一个货车类，并在这两个子类中重写基类的虚方法。另外，当程序运行时，根据选择的不同车辆类型，程序会调用相应车辆类型的 Run 方法。（20 分）（实例位置：光盘\mr\07\qjyy\01\_diy）

### 7.12.2 情景应用 2——封装类实现一个简单的计算器

视频讲解：光盘\mr\07\lx\封装类实现一个简单的计算器.exe

实例位置：光盘\mr\07\qjyy\02

计算器是编程初学者开始学习编程时最常见的一一个例子，本实例将使用面向对象编程思想中的封装性

来编写一个简单的计算器。实例运行效果如图 7.22 所示。

实现过程如下。

(1) 打开 Visual Studio 2008 开发环境，新建一个 Windows 窗体应用程序，并将其命名为 CalcByClass。

(2) 在默认窗体 Form1 中添加一个 TextBox 控件，用来显示输入的数和计算结果；添加 16 个 Button 控件，分别用来表示相应的数字和对输入的数进行加、减、乘、除运算。

(3) 程序主要代码如下。

```
public class CCount
{
    public int Sum(int a, int b, string type)
    {
        switch (type)
        {
            case "+":
                return a + b; break; //计算两个数的和
            case "-":
                return a - b; break; //计算两个数的差
            case "*":
                return a * b; break; //计算两个数的积
            case "/":
                return a / b; break; //计算两个数的商
            default:
                return 0; break;
        }
    }
}
```



图 7.22 封装类实现一个简单的计算器

**DIY：**使用类来封装药品信息。提示：可创建一个控制台应用程序，首先定义一个描述药品信息（包括药品的名称、药品的单价、药品的用法）的类 Leechdom，然后在入口方法 Main 中创建 Leechdom 类的对象，接着设置药品的名称、单价，并调用描述药品用法的方法，最后输出某种药品的用法信息。（20 分）  
(实例位置：光盘\mr\07\qjyy\02\_diy)

### 7.12.3 情景应用 3——使用分部类记录学生信息

■ 视频讲解：光盘\mr\07\lx\使用分部类记录学生信息.exe

■ 实例位置：光盘\mr\07\qjyy\03

在定义类时，可以用分部类将公共部分和非公共部分分开定义。下面用分部类结构记录学生的信息，并进行显示。实例运行效果如图 7.23 所示。

实现过程如下。

(1) 打开 Visual Studio 2008 开发环境，新建一个 Windows 窗体应用程序，并将其命名为 StoreStuInfo。

(2) 在默认窗体 Form1 中添加 8 个 TextBox 控件，分别用来显示学生的编号、姓名、年龄、性别、出生年月、年



图 7.23 使用分部类记录学生信息

级、班级和班主任等信息；添加一个 Button 控件，用来获取分部类中存储的学生信息，并显示在 TextBox 控件中。

(3) 程序主要代码如下。

```
partial class CommInfo
{
    public object ID; //编号
    public object Name; //姓名
    object sex; //性别
    public object Sex
    {
        get
        {
            if ((bool)sex == true)
                sex = "男";
            else
                sex = "女";
            return sex;
        }
        set { sex = value; }
    }
    public object Age; //年龄
    public object Birthday; //出生年月
}
partial class CommInfo
{
    public object Grade; //年级
    public object Class; //班级
    public object Director; //班主任
}
```

**DIY：** 使用分部类实现两个数的加减运算。提示：定义一个封装了加减法运算的类，并将该类的定义分为两部分，在第一部分中定义执行两个数相加的方法，在第二部分中定义执行两个数相减的方法。（20 分）（实例位置：光盘\mr\07\qjyy\03\_diy）

#### 7.12.4 情景应用 4——使用泛型存储不同类型的数据列表

**视频讲解：**光盘\mr\07\lx\使用泛型存储不同类型的数据列表.exe

**实例位置：**光盘\mr\07\qjyy\04

泛型编程是一种编程范式，它利用“参数化类型”将类型抽象化，从而实现更为灵活的复用。本实例将使用泛型存储不同类型的数据，在实现时，首先定义一个泛型类，并在泛型类中定义多个泛型变量，然后使用这些变量记录不同类型的数据，这样就可以重复利用泛型变量来存储不同类型的数据。实例运行效果如图 7.24 所示。

实现过程如下。

(1) 打开 Visual Studio 2008 开发环境，新建一个

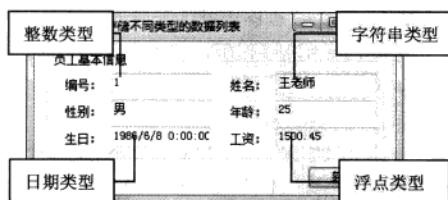


图 7.24 使用泛型存储不同类型的数据列表

Windows窗体应用程序，并将其命名为 ExtensiveList。

(2) 在默认窗体 Form1 中添加 6 个 TextBox 控件，分别用来显示员工的编号、姓名、年龄、性别、生日和工资等信息；添加一个 Button 控件，用来通过泛型获取员工信息并显示。

(3) 程序主要代码如下。

```
class Types<T>
{
    public T Num; //声明编号字段
    public T Name; //声明姓名字段
    public T Sex; //声明性别字段
    public T Age; //声明年龄字段
    public T Birthday; //声明生日字段
    public T Salary; //声明薪水字段
}
private void button1_Click(object sender, EventArgs e)
{
    Types<object> Exte = new Types<object>(); //创建泛型对象
    //为泛型类中声明的字段进行赋值，存储不同类型的值
    Exte.Num = 1;
    Exte.Name = "王老师";
    Exte.Sex = "男";
    Exte.Age = 25;
    Exte.Birthday = Convert.ToDateTime("1986-06-08");
    Exte.Salary = 1500.45F;
    //将泛型类中各字段的值显示在文本框中
    textBox1.Text = Exte.Num.ToString();
    textBox2.Text = Exte.Name.ToString();
    textBox3.Text = Exte.Sex.ToString();
    textBox4.Text = Exte.Age.ToString();
    textBox5.Text = Exte.Birthday.ToString();
    textBox6.Text = Exte.Salary.ToString();
}
```

**DIY：**使用 List<T>泛型存储学生信息。提示：可以使用 List<string>列表来存储学生信息（如学号、姓名及性别等）。(20分)(实例位置：光盘\mr\07\qjyy\04\_diy)

### 7.12.5 情景应用 5——使用泛型去掉数组中的重复数字

视频讲解：光盘\mr\07\lx\使用泛型去掉数组中的重复数字.exe

实例位置：光盘\mr\07\qjyy\05

本实例使用泛型去掉了数组中的重复数字，具体实现时，首先需要对数组进行排序，排序之后可以确定重复的数字是相邻的，这时只需比较邻近的数字是否相同即可。实例运行效果如图 7.25 所示。

实现过程如下。

(1) 打开 Visual Studio 2008 开发环境，新建一个 Windows 窗体应用程序，并将其命名为 RemoveSameNum。

(2) 在默认窗体 Form1 中添加两个 Label 控件，分别用来显示原始一维数组和去掉重复数字之后的一

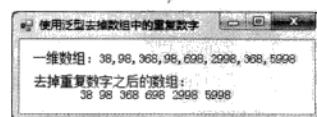


图 7.25 使用泛型去掉数组中的重复数字

维数组。

(3) 程序主要代码如下。

```
static int[] RemoveNum(int[] P_int_Data)
{
    List<int> P_list_Arrs = new List<int>(); //去掉数组中的重复数字
    for (int i = 0; i < P_int_Data.Length - 1; i++)
    {
        if (P_int_Data[i] != P_int_Data[i + 1]) //创建泛型集合对象
        {
            P_list_Arrs.Add(P_int_Data[i]); //循环访问源数组元素
        }
    }
    P_list_Arrs.Add(P_int_Data[P_int_Data.Length - 1]); //判断相邻的值是否相同
    return P_list_Arrs.ToArray(); //如果不同，则将值添加到泛型集合中
}
```

说明：List<T>泛型集合表示可通过索引访问的对象的强类型列表。

DIY：参考上面的实例，不对数组进行排序，尝试直接使用两个 for 循环进行比对，从而去掉数组中的重复数字。提示：可首先把数组通过 ToList 方法转换成 List<String> 实例。（20 分）（实例位置：光盘\mr\07\qjyy\05\_diy）

情景应用 DIY 栏目分数统计：

DIY 题目	1	2	3	4	5	总分数	
分数							

## 7.13 自我测试

### 一、选择题（每题 10 分，5 道题）

1. 下面有关重载方法的说法中，正确的是（ ）。  
 A. 重载函数的参数个数必须不同  
 C. 重载函数必须具有不同的返回值类型
2. 下面关于类和对象的说法中，不正确的是（ ）。  
 A. 类是一种系统提供的数据类型  
 C. 类和对象的关系是抽象和具体的关系
3. 下面关于析构函数的说法中，不正确的是（ ）。  
 A. 析构函数中不可以包含 return 语句  
 C. 用户可定义有参析构函数
4. 下面有关派生类的描述中，不正确的是（ ）。  
 A. 派生类可以继承基类的构造函数  
 C. 派生类不能访问基类的私有成员
5. 如果要从派生类中访问基类的成员，可以使用（ ）。

- A. this 关键字      B. me 关键字      C. override 关键字      D. base 关键字

## 二、填空题（每题 10 分，5 道题）

1. 面向对象编程具有（ ）、（ ）和（ ）3 大特点。
2. 在 C# 的继承中，派生出其他类的类称为（ ）或（ ），被派生的类称为（ ）或（ ）。
3. 在 C# 语言中，派生类继承基类中除（ ）和（ ）以外的全部成员。
4. 一个类可以包含（ ）个构造函数和（ ）个析构函数。
5. 下列代码的输出结果是（ ）。

```
class Test
{
    public enum WeekDays
    {
        Mon,Tue,Wed,Thur,Fri,Sta,Sun
    }
    static void Main()
    {
        WeekDays week=(WeekDays)2;
        Console.WriteLine(week);
    }
}
```

测试分数统计：

类别	第 1 题	第 2 题	第 3 题	第 4 题	第 5 题
选择题分数					
填空题分数					
总分数					

## 7.14 行动指南

开始日期： 年 月 日

结束日期： 年 月 日

序号	内 容	行动指南	
1	照猫画虎栏目 分数（ ）	分数>75 分	优秀，基本功掌握得不错，加油！
		75 分>分数>50 分	及格，知识掌握得不牢，重新做一遍照猫画虎。
	情景应用栏目 分数（ ）	分数<50 分	请使用光盘中提供的“实战能力测试系统”进行提高训练。
		分数>75 分	优秀，综合应用能力很强。
	自我测试栏目 分数（ ）	75 分>分数>50 分	及格，综合应用能力需提高，再练一遍情景应用。
		分数<50 分	请使用光盘中提供的“实战能力测试系统”进行提高训练。
综合评价		反复训练后，以上各项分数都在 75 分以上，方可进入下一堂课学习。	

续表

序号	内 容	行动指南
2	编程能力培养：本栏目提供了一些实践题目，对于培养编程能力很有效，要做好。	1. 使用面向对象的思想查找字符串中的数字。提示：自定义一个方法，在该方法中定义一个布尔型变量并初始化为 false，作为逻辑判断标记；定义一个由数字(0~9)组成的字符串数组，作为判断字符是否为数字的参照对象；然后通过 foreach 语句遍历数组中的元素，并与要判断的字符作比较，若在数组中可以找到该字符，则判定该字符为数字，否则不是数字。 2. 使用多态性计算多种几何图形的面积。提示：定义一个基类，并在该基类中声明若干个属性，用来描述几何图形边长、高度和半径等，还要在该基类中封装一个用于计算几何图形面积的虚方法，然后从该基类派生出若干个用于描述具体几何图形的子类，并在子类中重写基类的虚方法，从而实现计算具体几何图形的面积。
3	编程习惯培养：本堂课培养读者在学习开发中记笔记的习惯，把开发中遇到的问题、总结的经验记录下来。	
4	创新能力培养：看看身边有没有可以用编程解决的问题，记录到右边的表格中。根据学习进度，尝试编写解决这些问题的小程序。	

## 7.15 成功可以复制——百度 CEO 李彦宏

19岁的李彦宏离开山西阳泉到梦想中的北大就读信息管理专业，23岁远渡重洋赴美国布法罗纽约州立大学主攻计算机，正是北大的信息管理专业让他深谙搜索内涵，正是美国的计算机学业让他掌握了计算机工具。

在搜索引擎发展初期，李彦宏作为全球最早研究者之一，最先创建了 ESP 技术，并将它成功地应用于 Infoseek 的搜索引擎中，另外，图像搜索引擎是他的另一项极具应用价值的技术创新。

在李彦宏 31 岁时，他创建了中国最大的搜索引擎公司——百度网络技术有限公司。知识改变了命运！百度公司创始人、CEO 李彦宏十分相信这句话。

在美国的 8 年人生历程中，李彦宏先后担任了道·琼斯公司高级顾问、《华尔街日报》网络版实时金融信息系统设计者、国际知名互联网企业——Infoseek 资深工程师等职务。他为道·琼斯公司设计的实时金融系统，迄今仍被广泛地应用于华尔街各大公司的网站。

1999 年年底，李彦宏携 120 万美金的风险投资回国与好友徐勇先生共同创建百度网络技术有限公司，并在短短 6 个月的时间内完成目前中国最大、最好的中文搜索引擎的开发工作。

2005 年 8 月，百度在美国纳斯达克成功上市，成为全球资本



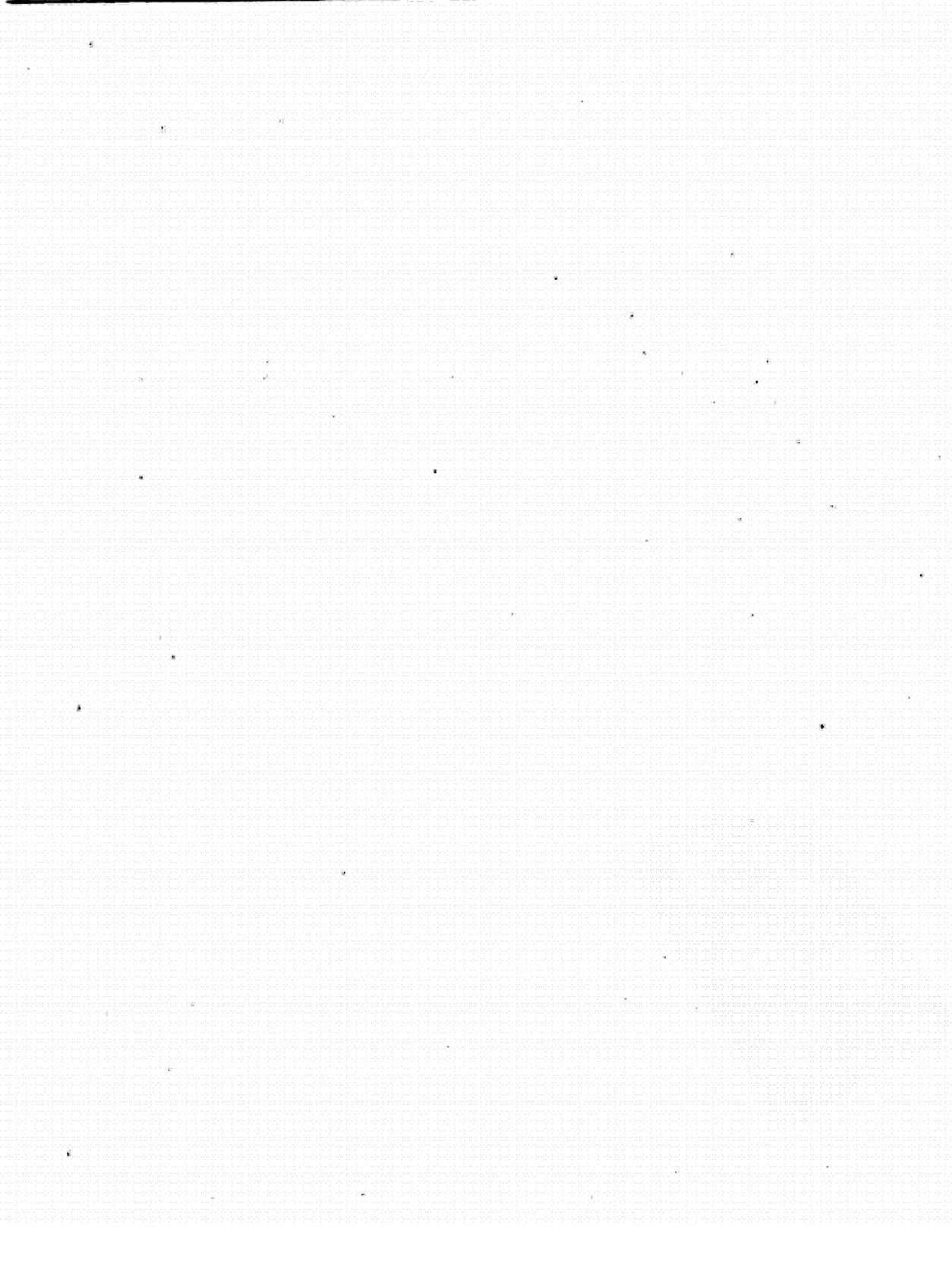
市场最受关注的上市公司之一。目前，百度也是全球跨国公司最多寻求合作的中国公司，随着百度日本公司的成立，百度加快了走向国际化的步伐。

 经典语录 -----

“活的搜索改变生活。”

 深度评价 -----

从李彦宏的求学、工作及创业经历中给我们这样的启示，知识可以改变命运，知识可以改变生活。作为程序设计人员，要多学习，刻苦钻研，掌握更多的知识和技能，然后把学到的知识和技能应用到实际工作和生活中，从而去改变我们的工作和生活。



# 第 2 部分

## 提高篇

- » 第 8 堂课 Windows 窗体设计
- » 第 9 堂课 Windows 应用程序常用控件
- » 第 10 堂课 Windows 应用程序高级控件
- » 第 11 堂课 ADO.NET 数据访问技术
- » 第 12 堂课 DataGridView 数据控件



# 第 8 堂课

## Windows 窗体设计

(  视频讲解：139分钟 )

Windows 环境中主流的应用程序都是窗体应用程序，Windows 窗体应用程序比控制台应用程序要复杂得多，理解其结构的基础是理解窗体，因此深刻认识 Windows 窗体变得尤为重要。本堂课将对 Windows 窗体的基本操作、分类及设计进行详细讲解。

学习摘要：

- » 了解什么是 Form 窗体
- » 掌握如何添加和删除新窗体
- » 了解如何设置启动窗体
- » 掌握窗体的基本属性和事件
- » 了解什么是 MDI 窗体
- » 掌握如何设置父窗体和子窗体
- » 掌握如何对子窗体进行排列
- » 了解什么是继承窗体
- » 掌握如何创建继承窗体

## 8.1 Form 窗体基础

Form 窗体也称为窗口，它是.NET 框架中的一种智能客户端技术，使用它可以显示信息、请求用户输入以及通过网络与远程计算机通信等。使用 Visual Studio 2008 可以轻松地创建 Form 窗体。下面将对 Form 窗体的相关内容进行详细介绍。

### 8.1.1 Form 窗体概述

在 Windows 窗体应用程序中，窗体是向用户显示信息的可视界面，它是 Windows 窗体应用程序的基本单元。窗体都具有自己的特征，开发人员可以通过编程来进行设置。窗体也是对象，窗体类定义了生成窗体的模板，每创建一个窗体类就产生一个窗体，.NET 框架类库的 System.Windows.Forms 命名空间中定义的 Form 类是所有窗体类的基类。在编写窗体应用程序时，首先需要设计窗体的外观和在窗体中添加控件或组件，虽然可以通过编写代码来实现，但是却不直观、也不方便，而且很难精确地控制界面。如果要编写窗体应用程序，推荐使用 Visual Studio 2008，Visual Studio 2008 提供了一个图形化的可视化窗体设计器，可以实现所见即所得的设计效果，以便快速开发窗体应用程序。

### 8.1.2 添加和删除 Form 窗体

在添加或删除窗体时，首先要创建一个 Windows 窗体应用程序，在本书的第 1 堂课中已经介绍过如何创建 Windows 窗体应用程序，此处不再赘述。如图 8.1 所示是一个新创建的 Windows 窗体应用程序。

如果要向项目中添加一个新窗体，可以在项目名称 NewForm 上单击鼠标右键，在弹出的快捷菜单中选择“添加”/“Windows 窗体”或“添加”/“新建项”命令，如图 8.2 所示。

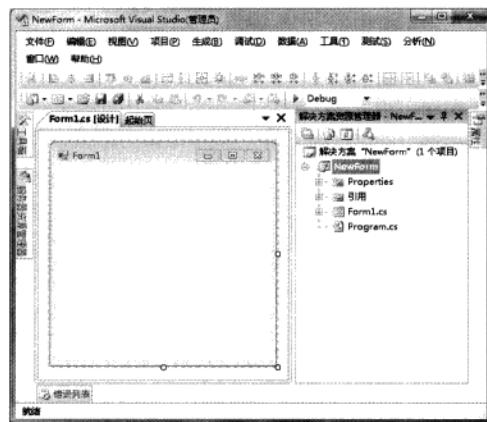


图 8.1 Windows 窗体应用程序

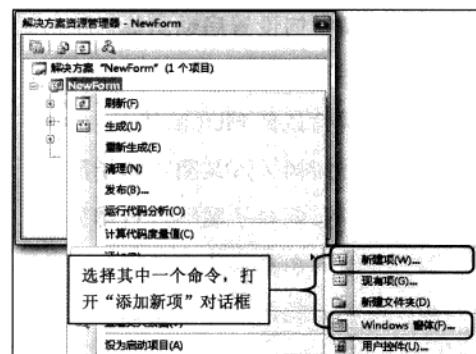


图 8.2 添加新窗体的右键菜单

选择“新建项”或“Windows 窗体”命令后，打开“添加新项”对话框，如图 8.3 所示。

选择“Windows 窗体”，输入窗体名称后单击“添加”按钮，即可向项目中添加一个新的窗体。

删除窗体的方法非常简单，只需在要删除的窗体名称上单击鼠标右键，在弹出的快捷菜单中选择“删

除”命令，即可将选中的窗体删除，如图 8.4 所示。

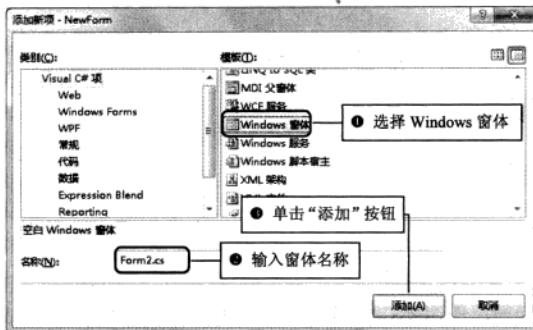


图 8.3 “添加新项”对话框



图 8.4 删除窗体

### 8.1.3 添加多窗体

一个完整的 Windows 窗体应用程序是由多个窗体组成的，此时就需要对多窗体设计有所了解。多窗体即向项目中添加多个窗体，在这些窗体中实现不同的功能，下面对多窗体的添加以及如何设置启动窗体进行讲解。

#### 1. 多窗体的添加

多窗体的添加即向某个项目中添加多个窗体，这里以 8.1.2 小节中的项目为例，演示如何添加多窗体。添加多窗体后的项目如图 8.5 所示。

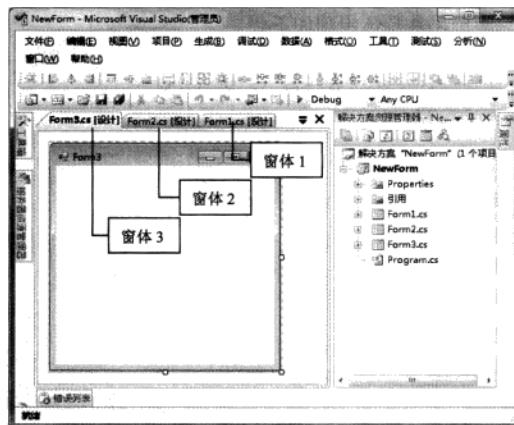


图 8.5 向项目中添加多个窗体

**说明：**添加窗体的方法在 8.1.2 小节中已经做了详细介绍，即添加多窗体时，只需重复执行 8.1.2 节中所述的添加窗体的步骤即可。

## 2. 设置启动窗体

向项目中添加了多个窗体以后，如果要调试程序，必须要设置首先运行的窗体，这时就需要设置项目的启动窗体。项目的启动窗体是在 Program.cs 文件中设置的，在 Program.cs 文件中改变 Run 方法的参数，即可实现设置启动窗体。

Run 方法用于在当前线程上开始运行标准应用程序，并使指定窗体可见。

语法：public static void Run(Form mainForm)

说明：mainForm 表示要设为启动窗体的窗体。

**例 8.01** 将 Form1 窗体设置为项目的启动窗体，代码如下。

```
Application.Run(new Form1());
```

### 8.1.4 设置窗体的属性

在新创建的窗体中包含一些基本的组成要素，如图标、标题、位置和背景等，设置这些要素可以通过窗体的属性面板实现，也可以通过代码实现。但是为了快速开发 Windows 窗体应用程序，通常都是通过属性面板进行设置，下面详细介绍窗体的常见属性设置。

#### 1. 更换窗体的图标

添加一个新的窗体后，窗体的图标是系统默认的图标。如果想更换窗体的图标，可以在属性面板中设置窗体的 Icon 属性。窗体的默认图标和更换后的图标如图 8.6 所示。

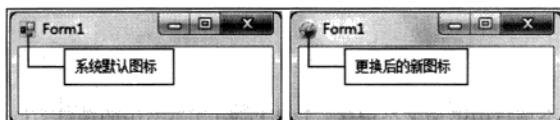


图 8.6 窗体的默认图标与更换后的图标

更换窗体图标的过程非常简单，其具体步骤如下。

- (1) 选中窗体，然后在窗体的属性面板中选中 Icon 属性，会出现  按钮，如图 8.7 所示。
- (2) 单击  按钮，打开选择图标文件的对话框，如图 8.8 所示。



图 8.7 窗体的 Icon 属性

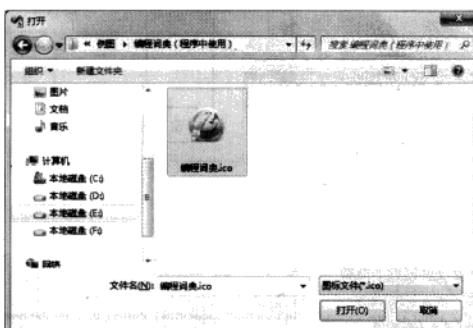


图 8.8 选择图标文件

- (3) 选择新的窗体图标文件之后单击“打开”按钮，即可完成窗体图标的更换。

## 2. 隐藏窗体的标题栏

在有些情况下需要隐藏窗体的标题栏，例如，软件的加载窗体大多数都采用无标题栏的窗体，开发人员可以通过设置窗体的 FormBorderStyle 属性为 None，实现隐藏窗体标题栏功能。FormBorderStyle 属性有 7 个属性值，其属性值及说明如表 8.1 所示。

表 8.1 FormBorderStyle 属性的属性值及说明

属性值	说明
Fixed3D	固定的三维边框
FixedDialog	固定的对话框样式的粗边框
FixedSingle	固定的单行边框
FixedToolWindow	不可调整大小的工具窗口边框
None	无边框
Sizable	可调整大小的边框
SizableToolWindow	可调整大小的工具窗口边框

## 3. 控制窗体的显示位置

设置窗体的显示位置时，可以通过设置窗体的 StartPosition 属性来实现。StartPosition 属性有 5 个属性值，其属性值及说明如表 8.2 所示。

表 8.2 StartPosition 属性的属性值及说明

属性值	说明
CenterParent	窗体在其父窗体中居中
CenterScreen	窗体在当前显示窗口中居中，其尺寸在窗体大小中指定
Manual	窗体的位置由 Location 属性确定
WindowsDefaultBounds	窗体定位在 Windows 默认位置，其边界也由 Windows 默认决定
WindowsDefaultLocation	窗体定位在 Windows 默认位置，其尺寸在窗体大小中指定

说明：设置窗体的显示位置时，只需根据不同的需要选择属性值即可。

## 4. 修改窗体的大小

在窗体的属性中，通过 Size 属性可以设置窗体的大小。双击窗体属性面板中的 Size 属性，可以看到其下拉菜单中有 Width 和 Height 两个属性，分别用于设置窗体的宽和高。修改窗体的大小，只需更改 Width 和 Height 属性的值即可。

## 5. 设置图像背景的窗体

为了使窗体设计更加美观，通常会设置窗体的背景，开发人员可以设置窗体的背景颜色，也可以设置窗体的背景图片。设置窗体的背景图片时可以通过设置窗体的 BackgroundImage 属性实现，其具体步骤如下。

(1) 选中窗体属性面板中的 BackgroundImage 属性，会出现 按钮，如图 8.9 所示。

(2) 单击 按钮，打开“选择资源”对话框，如图 8.10 所示。

(3) 在图 8.10 所示的“选择资源”对话框中有两个选项，一个



图 8.9 BackgroundImage 属性

是“本地资源”，另一个是“项目资源文件”，其差别是选择“本地资源”后，直接选择图片，保存的是图片的路径；而选择“项目资源文件”后，会将选择的图片保存到项目资源文件 Resources.resx 中。无论选择哪种方式，都需要单击“导入”按钮选择背景图片，选择完成后单击“确定”按钮完成窗体背景图片的设置，Form1 窗体背景图片设置前后对比如图 8.11 所示。



图 8.10 “选择资源”对话框

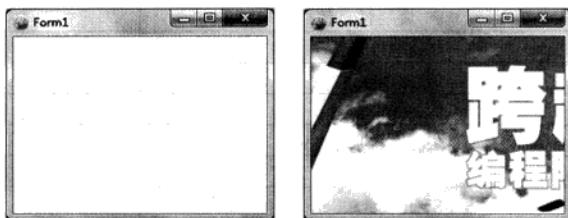


图 8.11 设置窗体背景图片前后对比

**技巧：**设置窗体背景图片时，窗体还提供了一个 `BackgroundImageLayout` 属性，该属性主要用来控制背景图片的布局，开发人员需要将该属性的属性值设置为 `Stretch`，以便能够使图片自动适应窗体的大小。

### 8.1.5 窗体的显示与隐藏

#### 1. 窗体的显示

如果要在一个窗体中通过按钮打开另一个窗体，就必须通过调用 `Show` 方法显示窗体。

语法： `public void Show()`

**例 8.02** 在 Form1 窗体中添加一个 Button 按钮，在按钮的 Click 事件中调用 `Show` 方法打开 Form2 窗体，代码如下。

```
Form2 frm2 = new Form2(); //创建 Form2
frm2.Show(); //调用 Show 方法显示 Form2 窗体
```

**说明：**由于 `Show` 方法为非静态方法，所以需要使用窗体对象进行调用。下面将要介绍到的 `Hide` 方法也是非静态方法，所以在使用它时也需要首先创建窗体对象。

#### 2. 窗体的隐藏

隐藏窗体时需要调用窗体的 `Hide` 方法。

语法： `public void Hide()`

**例 8.03** 在登录窗口登录系统时，输入用户名和密码后，单击“登录”按钮，隐藏登录窗口，显示主窗体，关键代码如下。

```
this.Hide(); //调用 Hide 方法隐藏当前窗体
frmMain frm = new frmMain();
frm.Show(); //调用 Show 方法打开窗体
```

**说明：**在使用 `Hide` 方法隐藏窗体之后，窗体所占用的资源并没有从内存中释放掉，而是继续存储在内存中，开发人员可以随时调用 `Show` 方法来显示隐藏的窗体。

### 8.1.6 触发窗体事件

Windows 是事件驱动的操作系统，对 Form 类的任何交互都是基于事件来实现的。Form 类提供了大量的事件用于响应执行窗体的各种操作，下面详细介绍 Form 类中最常用的 Click 事件、Load 事件和 FormClosing 事件。

**技巧：**在选择窗体事件时，可以通过选中控件，然后单击其“属性”窗口中的图标来实现。

#### 1. Click（单击）事件

当单击窗体时，将会触发窗体的 Click 事件。

语法：public event EventHandler Click

**例 8.04** 在窗体的 Click 事件中编写代码，实现当单击窗体时弹出对话框，代码如下。（实例位置：光盘\mr\08\sl\8.04）

```
private void Form1_Click(object sender, EventArgs e)           //窗体的 Click 事件
{
    MessageBox.Show("已经单击了窗体！");                      //弹出提示框
}
```

程序运行结果如图 8.12 所示。

#### 2. Load（加载）事件

窗体加载时将触发窗体的 Load 事件。

语法：public event EventHandler Load

**例 8.05** 当窗体加载时，弹出对话框，询问是否查看窗体，单击“是”按钮，查看窗体，代码如下。（实例位置：光盘\mr\08\sl\8.05）

```
private void Form1_Load(object sender, EventArgs e)           //窗体的 Load 事件
{
    //使用 if 语句判断是否单击了“是”按钮
    if (MessageBox.Show("是否查看窗体！", "", MessageBoxButtons.YesNo, MessageBoxIcon.Information) ==
        DialogResult.OK)
    {
    }
}
```

程序运行结果如图 8.13 所示。

#### 3. FormClosing（关闭）事件

窗体关闭时将触发窗体的 FormClosing 事件。

语法：public event FormClosingEventHandler FormClosing

**例 8.06** 创建一个 Windows 窗体应用程序，实现当关闭窗体之前，弹出提示框，询问是否关闭当前窗体，单击“是”按钮，关闭窗体，代码如下。（实例位置：光盘\mr\08\sl\8.06）

```
private void Form1_FormClosing(object sender, FormClosingEventArgs e)
{
    DialogResult dr = MessageBox.Show("是否关闭窗体", "提示", MessageBoxButtons.YesNo, MessageBoxIcon.Warning);
```

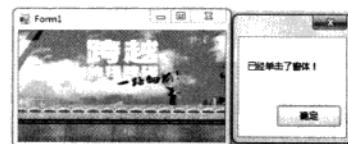


图 8.12 单击窗体触发 Click 事件

```

if (dr == DialogResult.Yes)
{
    e.Cancel = false; //使用 if 语句判断是否单击“是”按钮
}
else
{
    e.Cancel = true; //如果单击“是”按钮则关闭窗体
}
}
}

```

程序运行结果如图 8.14 所示。



图 8.13 窗体加载时弹出提示框



图 8.14 窗体关闭时显示提示框

## 8.2 MDI 窗体设计

窗体是所有界面的基础，这就意味着为了打开多个文档，需要具有能够同时处理多个窗体的应用程序，为了适应这个需求，Visual Studio 2008 中预置了 MDI 窗体，即多文档界面，本节将对 MDI 窗体设计进行详细讲解。

### 8.2.1 MDI 窗体概述

多文档界面（Multiple-Document Interface）简称 MDI 窗体，它主要用于同时显示多个文档，每个文档显示在各自的窗口中。MDI 窗体中通常有包含子菜单的窗口菜单，以便在窗口或文档之间进行切换。在实际应用中 MDI 窗体十分常见，如图 8.15 所示就是一个 MDI 窗体。



图 8.15 MDI 窗体

MDI 窗体的应用非常广泛，例如，如果某公司的库存管理系统需要实现自动化，需要使用窗体来输入客户和货品的数据、发出订单以及跟踪订单，而且这些窗体必须链接或从属于一个界面，并且必须能够同时处理多个文件，这时就需要建立 MDI 窗体以满足这些需求。

### 8.2.2 设置 MDI 窗体

在 MDI 窗体中，起到容器作用的窗体被称为“父窗体”，可放在父窗体中的其他窗体被成为“子窗体”，也称为“MDI 子窗体”。当 MDI 应用程序启动时，首先会显示父窗体。所有的子窗体都在父窗体中打开，在父窗体中可以在任何时候打开多个子窗体。需要注意的是，每个应用程序只能有一个父窗体，且其他子窗体不能移出父窗体的框架区域。下面介绍如何将窗体设置成父窗体或子窗体。

#### 1. 设置父窗体

如果想要将某个窗体设置成父窗体，只要在窗体的属性面板中将 IsMdiContainer 属性设置为 True 即可，如图 8.16 所示。

#### 2. 设置子窗体

设置完父窗体后，可以通过设置某个窗体的 MdiParent 属性来确定子窗体。

语法：public Form MdiParent { get; set; }

说明：它的属性值为 MDI 父窗体。

**例 8.07** 将 Form2、Form3、Form4 和 Form5 4 个窗体设置成当前主窗体的子窗体，并且在父窗体中打开这 4 个子窗体，代码如下。

```
Form2 frm2 = new Form2(); //创建 Form2 对象
frm2.Show(); //使用 Show 方法打开窗体
frm2.MdiParent = this; //设置 MdiParent 属性，将当前窗体作为父窗体
Form3 frm3 = new Form3(); //创建 Form3 对象
frm3.Show(); //使用 Show 方法打开窗体
frm3.MdiParent = this; //设置 MdiParent 属性，将当前窗体作为父窗体
Form4 frm4 = new Form4(); //创建 Form4 对象
frm4.Show(); //使用 Show 方法打开窗体
frm4.MdiParent = this; //设置 MdiParent 属性，将当前窗体作为父窗体
Form5 frm5 = new Form5(); //创建 Form5 对象
frm5.Show(); //使用 Show 方法打开窗体
frm5.MdiParent = this; //设置 MdiParent 属性，将当前窗体作为父窗体
```



图 8.16 设置父窗体

### 8.2.3 排列 MDI 子窗体

如果一个 MDI 窗体中有多个子窗体同时打开，则界面会显得非常混乱，而且不容易浏览，这时可以通过使用带有 MdiLayout 枚举的 LayoutMdi 方法来排列多文档界面父窗体中的子窗体。

语法：public void LayoutMdi(MdiLayout value)

说明：value 表示 MdiLayout 枚举值之一，它用来定义 MDI 子窗体的布局。MdiLayout 枚举用于指定 MDI 父窗体中子窗体的布局，其枚举成员及说明如表 8.3 所示。

表 8.3 MdiLayout 枚举成员及说明

枚举成员	说 明
Cascade	所有 MDI 子窗体均层叠在 MDI 父窗体的工作区内
TileHorizontal	所有 MDI 子窗体均水平平铺在 MDI 父窗体的工作区内
TileVertical	所有 MDI 子窗体均垂直平铺在 MDI 父窗体的工作区内

下面通过一个实例演示如何使用带有 MdiLayout 枚举的 LayoutMdi 方法来排列多文档界面父窗体中的子窗体。

**例 8.08** 创建一个 Windows 窗体应用程序，向项目中添加 4 个窗体，然后使用 LayoutMdi 方法及 MdiLayout 枚举设置窗体的排列，开发步骤如下。（实例位置：光盘\mr\08\sl\8.08）

(1) 新建一个 Windows 窗体应用程序，命名为 MDIForm，默认窗体为 Form1.cs。

(2) 将窗体 Form1 的 IsMdiContainer 属性设置为 True，以用作 MDI 父窗体，然后再添加 3 个 Windows 窗体，用作 MDI 子窗体。

(3) 在 Form1 窗体中添加一个 ToolStrip 控件，用作该父窗体的菜单项。

(4) 程序主要代码如下。

通过 ToolStrip 控件建立 4 个菜单项，分别为“加载子窗体”、“水平平铺”、“垂直平铺”和“层叠排列”。当运行程序并单击“加载子窗体”菜单后，可以加载所有的子窗体，代码如下。

```
private void 加载子窗体 ToolStripMenuItem_Click(object sender, EventArgs e)
{
    Form2 frm2 = new Form2();
    frm2.MdiParent = this;
    frm2.Show();
    Form3 frm3 = new Form3();
    frm3.MdiParent = this;
    frm3.Show();
    Form4 frm4 = new Form4();
    frm4.MdiParent = this;
    frm4.Show();
}
```

//创建 Form2 对象  
//设置 MdiParent 属性，将当前窗体作为父窗体  
//使用 Show 方法打开窗体  
//创建 Form3 对象  
//设置 MdiParent 属性，将当前窗体作为父窗体  
//使用 Show 方法打开窗体  
//创建 Form4 对象  
//设置 MdiParent 属性，将当前窗体作为父窗体  
//使用 Show 方法打开窗体

程序运行结果如图 8.17 所示。

加载所有的子窗体之后单击“水平平铺”菜单，使窗体中所有的子窗体水平排列，代码如下。

```
private void 水水平铺 ToolStripMenuItem_Click(object sender, EventArgs e)
{
    LayoutMdi(MdiLayout.TileHorizontal);
}
```

//使用 MdiLayout 枚举实现窗体的水平平铺

程序运行结果如图 8.18 所示。



图 8.17 加载所有子窗体

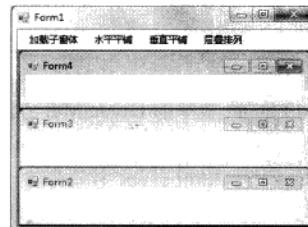


图 8.18 水水平铺子窗体

单击“垂直平铺”菜单，使窗体中所有的子窗体垂直排列，代码如下。

```
private void 垂直平铺 ToolStripMenuItem_Click(object sender, EventArgs e)
{
    LayoutMdi(MdiLayout.TileVertical); //使用 MdiLayout 枚举实现窗体的垂直平铺
}
```

程序运行结果如图 8.19 所示。

单击“层叠排列”菜单，使窗体中所有的子窗体层叠排列，代码如下。

```
private void 层叠排列 ToolStripMenuItem_Click(object sender, EventArgs e)
{
    LayoutMdi(MdiLayout.Cascade); //使用 MdiLayout 枚举实现窗体的层叠排列
}
```

程序运行结果如图 8.20 所示。



图 8.19 垂直平铺子窗体

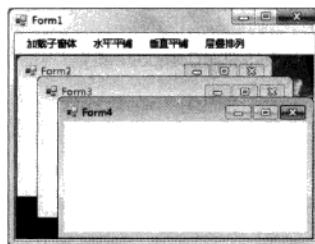


图 8.20 层叠排列子窗体

## 8.3 继承窗体设计

### 8.3.1 继承窗体概述

继承窗体就是根据现有窗体的结构创建一个与其相同的新窗体，这种从现有窗体继承的过程称为可视化继承。在某种情况下，项目可能需要一个与在以前的项目中创建的窗体类似的窗体，或者希望创建一个基窗体，其中含有随后将在项目中再次使用的控件布局之类的设置，每次重复使用时，都会对该原始窗体模板进行修改，这时就需要创建继承窗体。通过从基窗体继承来创建新 Windows 窗体是重复最佳工作成果的快捷方法，而不必每次需要窗体时都重新创建一个。

**注意：**为了从一个窗体继承，包含该窗体的文件或命名空间必须已编译成可执行文件或 DLL。

### 8.3.2 创建继承窗体

了解什么是继承窗体之后，接下来介绍如何创建继承窗体。创建继承窗体的方法有两种：一种是通过编程方式创建继承窗体，另一种是使用继承选择器创建继承窗体，下面对这两种方法分别进行介绍。

#### 1. 编程方式创建继承窗体

以编程方式创建继承窗体时，主要是在类中定义，并将引用添加到要从其继承的窗体。引用应包括包含该窗体的命名空间，后面跟一个句点，然后是基窗体本身的名字。

**例 8.09** 创建一个 Windows 窗体应用程序，以 Form1 为基窗体，将 Form2 设置为继承窗体。开发步骤

如下。（实例位置：光盘\mr\08\sl\8.09）

- (1) 创建一个 Windows 窗体应用程序， 默认窗体为 Form1.cs。
- (2) 在 Form1 窗体上添加一个 TextBox 控件、一个 Button 控件和一个 Label 控件。在 Button 控件的 Click 事件中添加代码， 实现 Label 控件显示 TextBox 控件中输入的内容。
- (3) 向项目中添加一个新的 Windows 窗体，并将其命名为 Form2。
- (4) 修改 Form2 窗体代码文件中 Form2 类所继承的基类。

Form2 窗体的原始代码如下。

```
namespace JCForm //项目名称
{
    public partial class Form2 : Form //表示当前窗体继承于 Form 类
    {
        public Form2()
        {
            InitializeComponent();
        }
    }
}
```

修改后的代码如下。

```
namespace JCForm //项目名称
{
    public partial class Form2:JCForm.Form1 //使 Form2 窗体继承 Form1 窗体
    {
        public Form2()
        {
            InitializeComponent();
        }
    }
}
```

运行 Form2 窗体， 其修改前后对比如图 8.21 和图 8.22 所示。

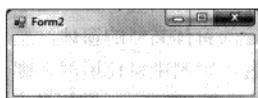


图 8.21 Form2 的起始运行结果



图 8.22 成为 Form1 窗体的继承窗体

**注意：**① 在修改 Form2 继承的基类时，首先需要生成项目，否则会出现设计器错误。

② 运行程序前，首先需要在 Program.cs 文件中将程序的启动窗体设置为 Form2。

## 2. 使用继承选择器创建继承窗体

继承窗体实现的最简便方法是使用“继承选择器”对话框，通过该对话框，不但可以继承当前项目中的窗体，还可以继承其他解决方案中的窗体。为了使用“继承选择器”对话框从某个窗体继承，包含该窗体的项目必须已经生成可执行文件或 DLL。

**说明：**如果要生成项目，可以通过选择“生成”菜单中的“生成解决方案”命令实现。

使用继承选择器创建继承窗体的步骤如下。

- (1) 在“解决方案资源管理器”面板中选中项目名称，单击鼠标右键，在弹出的快捷菜单中选择“添

加” / “新建项”命令，打开“添加新项”对话框，如图 8.23 所示。

(2) 在“添加新项”对话框中选择“继承的窗体”，并输入窗体名称，单击“添加”按钮，打开“继承选择器”对话框，如图 8.24 所示。

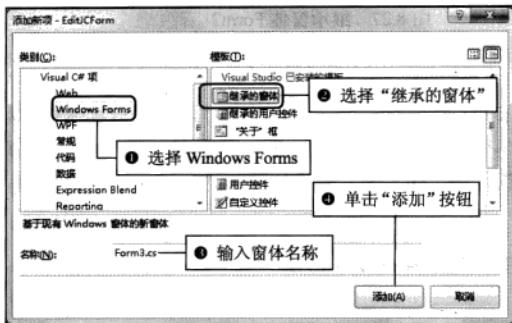


图 8.23 “添加新项”对话框



图 8.24 “继承选择器”对话框

(3) 在“继承选择器”对话框中单击“浏览”按钮，选择要继承的基窗体，单击“确定”按钮，完成继承窗体的创建。

### 8.3.3 在继承窗体中修改继承的控件属性

当向窗体中添加控件时，控件的 Modifiers 属性默认为 Private，因此，如果继承这样的窗体，则在继承窗体中控件的属性全部为不可编辑状态，这样就限制了继承窗体的扩展功能，开发人员可以通过将基窗体中控件的 Modifiers 属性设置为 Public，来实现在继承窗体中编辑控件属性的功能，从而也扩展了基窗体的使用。

**例 8.10** 创建一个 Windows 窗体应用程序，在继承窗体中修改基窗体中控件的属性，具体步骤如下。

(实例位置：光盘\mr\08\sl\8.10)

(1) 创建一个 Windows 窗体应用程序，默认窗体为 Form1.cs。

(2) 在 Form1 窗体中添加一个 Button 控件，设置其 Text 属性为“C#编程词典”，并将其 Modifiers 属性设置为 Public，如图 8.25 所示。



图 8.25 设置 Modifiers 属性

**说明：**这里也可以将 Button1 控件的 Modifiers 属性设置为 Protected 或 Protected Internal。

(3) 添加一个继承窗体，并将其命名为 Form2。在继承窗体 Form2 中可以修改 Button 控件的属性，这里将 Button 控件的 Text 属性重新设置为“学通 C# 的 24 堂课”，其前后对比如图 8.26 和图 8.27 所示。



图 8.26 基窗体 Form1

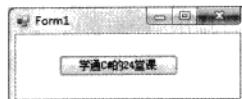


图 8.27 继承窗体 Form2

◆ 你问我答：什么时候应该使用继承窗体？

◆ 尽管继承窗体的出现非常讨人喜欢，但它并非没有局限性。例如，开发人员不可能在继承窗体中增加或减少基窗体中的菜单项，也不可能替换掉基窗体中的菜单事件处理程序。然而，作为一种避免复制控件和代码的模板机制，继承窗体很多时候都值得程序开发人员考虑使用。

## 8.4 照猫画虎——基本功训练

### 8.4.1 基本功训练 1——控制窗体加载时的位置

视频讲解：光盘\mr\08\lx\控制窗体加载时的位置.exe

实例位置：光盘\mr\08\zmhh\01

新建一个 Windows 窗体应用程序，在窗体的 Load 事件中通过设置当前窗体的 StartPosition 属性来控制窗体加载时的居中显示，代码如下。

```
private void Frm_Main_Load(object sender, EventArgs e)
{
    this.StartPosition = FormStartPosition.CenterScreen; //设置窗体居中显示
}
```

程序运行结果如图 8.28 所示。

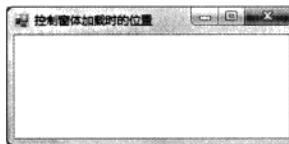


图 8.28 控制窗体加载时的位置

**照猫画虎：**制作一个程序，使窗体在桌面的左上角显示。提示：设置窗体的 Location 属性为(0,0)，然后将窗体的 StartPosition 属性设置为 Manual 枚举值。(20 分) (实例位置：光盘\mr\08\zmhh\01\_zmhh)

### 8.4.2 基本功训练 2——设置窗体在屏幕中的位置

视频讲解：光盘\mr\08\lx\设置窗体在屏幕中的位置.exe

实例位置：光盘\mr\08\zmhh\02

新建一个 Windows 窗体应用程序，在窗体中添加两个 TextBox 控件，分别用来输入窗体距屏幕左侧和上方的距离；添加一个 Button 控件，用来设置窗体在屏幕上的位置。然后在 Button 控件的 Click 事件中，通过设置窗体的 Left 属性和 Top 属性来设置窗体在屏幕中的位置，代码如下。

```
private void button1_Click(object sender, EventArgs e)
{
}
```

```

        this.Left = Convert.ToInt32(textBox1.Text);           //设置窗体左边缘与屏幕左边缘之间的距离
        this.Top = Convert.ToInt32(textBox2.Text);           //设置窗体上边缘与屏幕上边缘之间的距离
    }
}

```

运行程序，效果如图 8.29 所示。

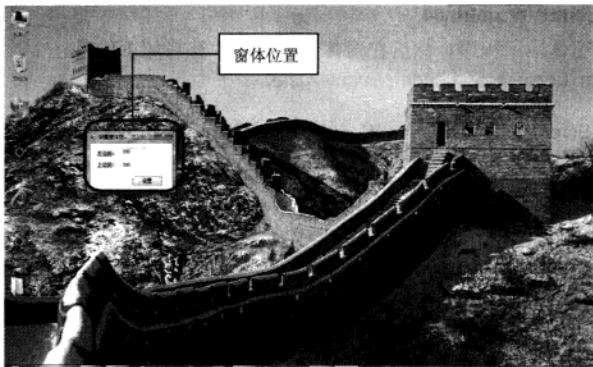


图 8.29 设置窗体在屏幕中的位置

**照猫画虎：**通过设置窗体的 Location 属性，使窗体在屏幕中居中显示。提示：首先获取屏幕的大小，然后根据屏幕的大小设置窗体的 Location 属性。(20 分)(实例位置：光盘\mr\08\zmhh\02\_zmhh)

#### 8.4.3 基本功训练 3——使窗体始终在桌面最顶层显示

**视频讲解：**光盘\mr\08\lx\使窗体始终在桌面最顶层显示.exe

**实例位置：**光盘\mr\08\zmhh\03

新建一个 Windows 窗体应用程序，并为窗体设置背景图片，然后在窗体的代码页中编写如下代码。

```

private void Frm_Main_Load(object sender, EventArgs e)
{
    this.TopMost = true;                                //设置在最顶层显示
}

```

程序运行结果如图 8.30 所示。

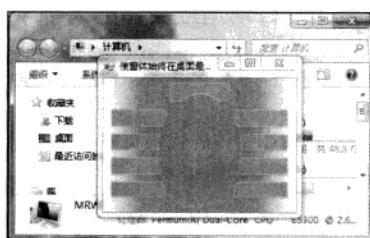


图 8.30 使窗体始终在桌面最顶层显示

**说明：**在图 8.30 中，“计算机”窗体获得了鼠标焦点，但是本实例的主窗体依然显示在最顶层。

**照猫画虎：**使用 API 函数 SetWindowPos 实现本程序的功能。(20 分)(实例位置：光盘\mr\08\zmhh\03\_zmhh)

#### 8.4.4 基本功训练 4——根据桌面大小调整窗体大小

视频讲解：光盘\mr\08\lx\根据桌面大小调整窗体大小.exe

实例位置：光盘\mr\08\zmhh\04

新建一个 Windows 窗体应用程序，然后在窗体的代码页中编写如下代码。

```
private void Frm_Main_Load(object sender, EventArgs e)
{
    int DeskWidth = Screen.PrimaryScreen.WorkingArea.Width;           //获取桌面宽度
    int DeskHeight = Screen.PrimaryScreen.WorkingArea.Height;         //获取桌面高度
    this.Width = Convert.ToInt32(DeskWidth * 0.8);                     //设置窗体宽度
    this.Height = Convert.ToInt32(DeskHeight * 0.8);                   //设置窗体高度
}
```

程序运行结果如图 8.31 所示。

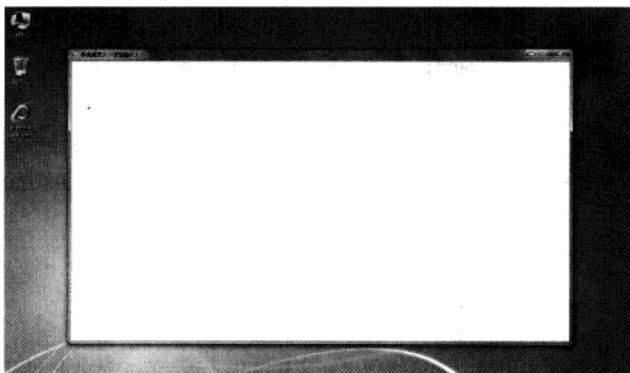


图 8.31 根据桌面大小调整窗体大小

**照猫画虎：**开发一个程序，用来手动改变无边框窗体的大小。(20 分)(实例位置：光盘\mr\08\zmhh\04\_zmhh)

#### 8.4.5 基本功训练 5——使背景图片自动适应窗体的大小

视频讲解：光盘\mr\08\lx\使背景图片自动适应窗体的大小.exe

实例位置：光盘\mr\08\zmhh\05

新建一个 Windows 窗体应用程序，然后在窗体的代码页中编写如下代码。

```
private void Frm_Main_Load(object sender, EventArgs e)
{
    this.BackgroundImage = Image.FromFile("test.jpg");           //设置窗体的背景图片
    this.BackgroundImageLayout = ImageLayout.Stretch;                  //使图片自动适应窗体大小
}
```

说明：test.jpg 图片文件应预先放置在程序的 Debug 文件夹下。

程序运行结果如图 8.32 所示。

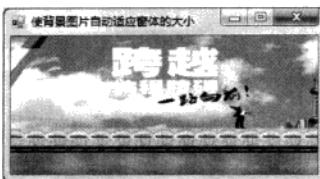


图 8.32 使背景图片自动适应窗体的大小

**照猫画虎：**根据以上程序，设置窗体的背景图片居中显示。提示：将窗体的 `BackgroundImageLayout` 属性设置为 `ImageLayout` 枚举的 `Center` 枚举值即可。(20 分)(实例位置：光盘\mr\08\zmhh\05\_zmhh)

照猫画虎栏目分数统计：

照猫画虎题目	1	2	3	4	5	总分数	
分数							

## 8.5 情景应用——拓展与实践

### 8.5.1 情景应用 1——从上次关闭位置启动窗体

视频讲解：光盘\mr\08\lx\从上次关闭位置启动窗体.exe

实例位置：光盘\mr\08\qjyy\01

本实例主要通过操作注册表实现从上次关闭位置启动窗体的功能。新建一个 Windows 窗体应用程序，并分别触发窗体的 Load 事件和 FormClosed 事件，然后在窗体的代码页中编写如下代码。

```
private void Frm_Main_Load(object sender, EventArgs e)
{
    RegistryKey myReg1, myReg2; //声明注册表对象
    myReg1 = Registry.CurrentUser; //获取当前用户注册表项
    try
    {
        myReg2 = myReg1.CreateSubKey("Software\\MySoft"); //在注册表项中创建子项
        //设置窗体的显示位置
        this.Location = new Point(Convert.ToInt16(myReg2.GetValue("1")), Convert.ToInt16(myReg2.
GetValue("2")));
    }
    catch {}
}
private void Frm_Main_FormClosed(object sender, FormClosedEventArgs e)
{
    RegistryKey myReg1, myReg2; //声明注册表对象
    myReg1 = Registry.CurrentUser; //获取当前用户注册表项
    myReg2 = myReg1.CreateSubKey("Software\\MySoft"); //在注册表项中创建子项
    try
    {
        myReg2.SetValue("1", this.Location.X.ToString()); //将窗体关闭位置的 x 坐标写入注册表
        myReg2.SetValue("2", this.Location.Y.ToString()); //将窗体关闭位置的 y 坐标写入注册表
    }
}
```

```

    }
    catch {}
}

```

程序运行结果如图 8.33 所示。



图 8.33 从上次关闭位置启动窗体

**DIY：**根据以上程序，开发一个将软件注册信息写入到注册表中的程序。提示：主要使用 RegistryKey 类的 CreateSubKey 方法和 SetValue 方法实现。(20 分)(实例位置：光盘\mr\08\qjyy\01\_diy)

### 8.5.2 情景应用 2——自定义最大化、最小化和关闭按钮

■ 视频讲解：光盘\mr\08\lx\自定义最大化、最小化和关闭按钮.exe

■ 实例位置：光盘\mr\08\qjyy\02

用户在制作应用程序时，为了使用户界面更加美观，一般都自己设计窗体的外观及窗体的最大化、最小化和关闭按钮。这里通过资源文件来存储窗体的外观及最大化、最小化和关闭按钮的图片，再通过鼠标移入、移出事件来实现按钮的动态效果。运行效果如图 8.34 所示。

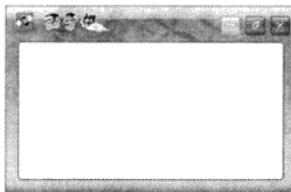


图 8.34 自定义最大化、最小化和关闭按钮

实现过程如下。

- (1) 创建一个 Windows 窗体应用程序，并将其命名为 ControlFormStatus。
- (2) 更改默认窗体 Form1 的 Name 属性为 Frm\_Main，并在该窗体中添加两个 Panel 控件，分别用来显示窗体标题栏和标题栏下面的窗体部分；添加 3 个 PictureBox 控件，分别用来表示“最大化”、“最小化”和“关闭”按钮。
- (3) 在 Frm\_Main 窗体加载时，将资源文件中的图片添加到相应的 PictureBox 和 Panel 控件中，代码如下。

```

private void Form1_Load(object sender, EventArgs e)
{
    this.Width = Properties.Resources.登录界面标题.Width; //设置窗体的宽度
    this.Height = Properties.Resources.登录界面标题.Height + Properties.Resources.登录界面下面.Height;
    //设置窗体的高度
    panel_Title.BackgroundImage = Properties.Resources.登录界面标题; //显示窗体标题栏的图片
    panel_ALL.BackgroundImage = Properties.Resources.登录界面下面; //显示窗体标题栏下面的图片
    pictureBox_Min.Image = null; //清空 PictureBox 控件
}

```

```

pictureBox_Min.Image = Properties.Resources.最小化按钮;           //显示最小化按钮的图片
pictureBox_Max.Image = null;                                     //清空 PictureBox 控件
pictureBox_Max.Image = Properties.Resources.最大化按钮;           //显示最大化按钮的图片
pictureBox_Close.Image = null;                                    //清空 PictureBox 控件
pictureBox_Close.Image = Properties.Resources.关闭按钮;           //显示关闭按钮的图片
}

```

(4) 在“关闭”按钮的鼠标移入、移出和单击事件中调用自定义方法 FrmClickMeans 和 ImageSwitch，以实现按钮的相关操作，代码如下。

```

private void pictureBox_Close_Click(object sender, EventArgs e)          //单击事件
{
    FrmClickMeans(this, Convert.ToInt16(((PictureBox)sender).Tag.ToString()));
}

private void pictureBox_Close_MouseEnter(object sender, EventArgs e)      //鼠标移入事件
{
    ImageSwitch(sender, Convert.ToInt16(((PictureBox)sender).Tag.ToString()), 0);
}

private void pictureBox_Close_MouseLeave(object sender, EventArgs e)        //鼠标移出事件
{
    ImageSwitch(sender, Convert.ToInt16(((PictureBox)sender).Tag.ToString()), 1);
}

```

 技巧：由于“最小化”、“最大化”和“关闭”按钮所使用的事件相同，所以将“最小化”和“最大化”按钮的事件指向“关闭”按钮的相关事件。

(5) 自定义方法 ImageSwitch 主要在鼠标移入和移出控件时对图片进行切换，代码如下。

```

#region 控制图片的切换状态
///<summary>
///控制图片的切换状态
///</summary>
///<param sender =" object ">要改变图片的对象</param>
///<param n="int">标识</param>
///<param ns="int">移入移出标识</param>
public static PictureBox Tem_PictB = new PictureBox();                  //创建 PictureBox 控件
public void ImageSwitch(object sender, int n, int ns)
{
    Tem_PictB = (PictureBox)sender;
    switch (n)                                         //获取标识
    {
        case 0:                                       //当前为最小化按钮
        {
            Tem_PictB.Image = null;                     //清空图片
            if (ns == 0)                               //鼠标移入
                Tem_PictB.Image = Properties.Resources.最小化变色;
            if (ns == 1)                               //鼠标移出
                Tem_PictB.Image = Properties.Resources.最小化按钮;
            break;
        }
        case 1:                                       //最大化按钮
    {

```

```

        Tem_PictB.Image = null;
        if (ns == 0)
            Tem_PictB.Image = Properties.Resources.最大化变色;
        if (ns == 1)
            Tem_PictB.Image = Properties.Resources.最大化按钮;
        break;
    }
    case 2: //关闭按钮
    {
        Tem_PictB.Image = null;
        if (ns == 0)
            Tem_PictB.Image = Properties.Resources.关闭变色;
        if (ns == 1)
            Tem_PictB.Image = Properties.Resources.关闭按钮;
        break;
    }
}
#endregion

```

(6) 自定义方法 FrmClickMeans 主要用于设置窗体的最大化、最小化和关闭按钮的操作，代码如下。

```

#region 设置窗体的最大化、最小化和关闭按钮的单击事件
///<summary>
///设置窗体的最大化、最小化和关闭按钮的单击事件
///</summary>
///<param Frm_Tem="Form">窗体</param>
///<param n="int">标识</param>
public void FrmClickMeans(Form Frm_Tem, int n)
{
    switch (n) //窗体的操作样式
    {
        case 0: //窗体最小化
            Frm_Tem.WindowState = FormWindowState.Minimized;
            break;
        case 1: //窗体最大化和还原的切换
            {
                if (Frm_Tem.WindowState == FormWindowState.Maximized)
                    Frm_Tem.WindowState = FormWindowState.Normal; //还原窗体大小
                else
                    Frm_Tem.WindowState = FormWindowState.Maximized; //窗体最大化
                break;
            }
        case 2: //关闭窗体
            Frm_Tem.Close();
            break;
    }
}
#endregion

```

**DIY：**完善以上程序，使窗体能够移动。提示：通过设置窗体的 DesktopLocation 属性实现。（20 分）  
 （实例位置：光盘\mr\08\qjyy\02\_diy）

### 8.5.3 情景应用3——磁性窗体的设计

视频讲解：光盘\mr\08\lx\磁性窗体的设计.exe

实例位置：光盘\mr\08\qjyy\03

在多媒体播放软件中经常会遇到一种情况，即当拖动其中一个窗体（主窗体）时，其他的窗体（子窗体）也会随着该窗体移动，但当拖动子窗体时，其他窗体将不跟随移动，这就是磁性窗体。这里将制作一个磁性窗体，当拖动主窗体移动时，两个子窗体如果相连，则跟随移动。运行效果如图8.35所示。

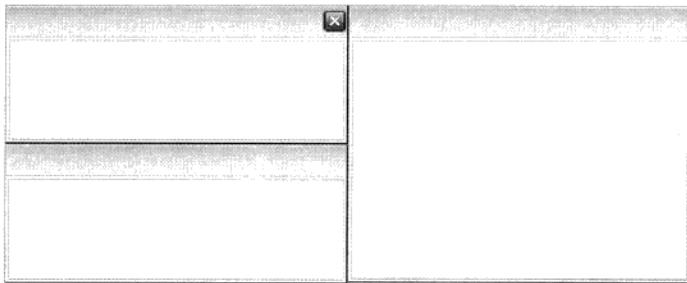


图8.35 磁性窗体的设计

实现过程如下。

(1) 创建一个Windows窗体应用程序，并将其命名为MagnetismForm。

(2) 更改默认窗体Form1的Name属性为Frm\_Play，并将该窗体的FormBorderStyle属性设置为None。在Frm\_Play窗体中添加两个Panel控件，分别命名为panel\_Title和panel\_Close，并将panel\_Title控件的Dock属性设置为Top。

(3) 在该项目中添加两个窗体，分别命名为Frm\_ListBox和Frm\_Libretto。

**说明：**磁性窗体的设置实际上就是将相关联的窗体位置和大小记录在公共类中，然后在移动窗体时，通过已记录的窗体信息，实现窗体的磁性效果。

(4) 在Frm\_Play窗体标题栏上按下鼠标时，获取其他两个窗体的相关信息（这两个窗体与该窗体具有磁性效果），代码如下。

```
private void panel_Title_MouseDown(object sender, MouseEventArgs e)
{
    int Tem_Y = 0;
    if (e.Button == MouseButtons.Left) //按下的是否为鼠标左键
    {
        Cla_FrmClass.FrmBackCheck(); //检测各窗体是否连在一起
        Tem_Y = e.Y; //获取鼠标在窗体上的位置，用于磁性窗体
        FrmClass.FrmPoint = new Point(e.X, Tem_Y); //获取鼠标在屏幕上的位置，用于窗体的移动
        FrmClass.CPoint = new Point(-e.X, -Tem_Y); //如果与Frm.ListBox窗体相连接
        if (FrmClass.Example_List_AdhereTo)
        {
            Cla_FrmClass.FrmDistanceJob(this, FrmClass.F_List); //计算窗体的距离差
            if (FrmClass.Example_Assistant_AdhereTo) //两个辅窗体是否连接在一起
            {
                ...
            }
        }
    }
}
```

```

        Cla_FrmClass.FrmDistanceJob(this, FrmClass.F_Libretto); //计算窗体的距离差
    }
}
if (FrmClass.Example_Libretto_AdhereTo) //如果与 Frm_Libretto 窗体相连接
{
    Cla_FrmClass.FrmDistanceJob(this, FrmClass.F_Libretto); //计算窗体的距离差
    if (FrmClass.Example_Assistant_AdhereTo) //两个辅窗体是否连接在一起
    {
        Cla_FrmClass.FrmDistanceJob(this, FrmClass.F_List); //计算窗体的距离差
    }
}
}
}
}

(5) 自定义方法 FrmBackCheck 的主要功能是通过各窗体的位置和大小，检测各窗体是否连接在一起，代码如下。

```

```

#region 检测各窗体是否连接在一起
///<summary>
///检测各窗体是否连接在一起
///</summary>
public void FrmBackCheck()
{
    bool Tem_Magnetism = false;
    Tem_Magnetism = false;
    if ((Example_Play_Top - Example_List_Top) == 0) //两个窗体是否顶对齐
        Tem_Magnetism = true;
    if ((Example_Play_Left - Example_List_Left) == 0) //两个窗体是否左对齐
        Tem_Magnetism = true;
    if ((Example_Play_Left - Example_List_Left - Example_List_Width) == 0) //两个窗体是否左右相连
        Tem_Magnetism = true;
    if ((Example_Play_Left - Example_List_Left + Example_List_Width) == 0) //两个窗体是否左右相连
        Tem_Magnetism = true;
    if ((Example_Play_Top - Example_List_Top - Example_List_Height) == 0) //两个窗体是否上下相连
        Tem_Magnetism = true;
    if ((Example_Play_Top - Example_List_Top + Example_List_Height) == 0) //两个窗体是否上下相连
        Tem_Magnetism = true;
    if (Tem_Magnetism) //如果连接
        Example_List_AdhereTo = true; //表示 Frm.ListBox 与主窗体是磁性窗体
    //Frm.Libretto 与主窗体
    Tem_Magnetism = false;
    if ((Example_Play_Top - Example.Libretto.Top) == 0) //两个窗体是否顶对齐
        Tem_Magnetism = true;
    if ((Example_Play_Left - Example.Libretto.Left) == 0) //两个窗体是否左对齐
        Tem_Magnetism = true;
    if ((Example_Play_Left - Example.Libretto.Left - Example.Libretto.Width) == 0)//两个窗体是否左右相连
        Tem_Magnetism = true;
    if ((Example_Play_Left - Example.Libretto.Left + Example.Libretto.Width) == 0)//两个窗体是否左右相连
        Tem_Magnetism = true;
    if ((Example_Play_Top - Example.Libretto.Top - Example.Libretto.Height) == 0)//两个窗体是否上下相连
        Tem_Magnetism = true;
    if ((Example_Play_Top - Example.Libretto.Top + Example.Libretto.Height) == 0)//两个窗体是否上下相连

```

```

    Tem_Magnetism = true;
    if (Tem_Magnetism)                                //如果连接
        Example_Libretto_AdhereTo = true;             //Frm_Libretto 与主窗体是磁性窗体
    //两个辅窗体

    Tem_Magnetism = false;
    if ((Example_List_Top - Example_Libretto_Top) == 0)
        Tem_Magnetism = true;
    if ((Example_List_Left - Example_Libretto_Left) == 0)
        Tem_Magnetism = true;
    if ((Example_List_Left - Example_Libretto_Left - Example_Libretto_Width) == 0)
        Tem_Magnetism = true;
    if ((Example_List_Left - Example_Libretto_Left + Example_Libretto_Width) == 0)
        Tem_Magnetism = true;
    if ((Example_List_Top - Example_Libretto_Top - Example_Libretto_Height) == 0)
        Tem_Magnetism = true;
    if ((Example_List_Top - Example_Libretto_Top + Example_Libretto_Height) == 0)
        Tem_Magnetism = true;
    if (Tem_Magnetism)                                //如果连接
        Example_Assistant_AdhereTo = true;            //两个辅窗体是磁性窗体
    }
#endifregion

```

(6) 自定义方法 FrmDistanceJob 用于计算 Frm\_Play 窗体与 Frm.ListBox 窗体、Frm.Libretto 窗体之间的距离，代码如下。

```

#region 计算窗体之间的距离差
///<summary>
///计算窗体之间的距离差
///</summary>
///<param Frm="Form">窗体</param>
///<param Follow="Form">跟随窗体</param>
public void FrmDistanceJob(Form Frm, Form Follow)
{
    switch (Follow.Name)
    {
        case "Frm.ListBox":                           //Frm.ListBox 子窗体
            {
                // “列表”窗体与“播放”窗体的位置差
                Example_List_space_Top = Follow.Top - Frm.Top;
                Example_List_space_Left = Follow.Left - Frm.Left;
                break;
            }
        case "Frm.Libretto":                         //Frm.Libretto 子窗体
            {
                //Frm.Libretto 子窗体与主窗体的位置差
                Example.Libretto_space_Top = Follow.Top - Frm.Top;
                Example.Libretto_space_Left = Follow.Left - Frm.Left;
                break;
            }
    }
#endifregion

```

(7) 当鼠标被按下后，通过鼠标的移动事件来移动窗体的位置，在移动窗体时，与该窗体相关联的磁性窗体也随之移动，代码如下。

```

private void panel_Title_MouseMove(object sender, MouseEventArgs e)
{
    if (e.Button == MouseButtons.Left) //当鼠标按下的是左键
    {
        Cla_FrmClass.FrmMove(this, e); //移动当前窗体
        if (FrmClass.Example_List_AdhereTo) //如果 Frm_ListBox 窗体是磁性窗体
        {
            Cla_FrmClass.ManyFrmMove(this, e, FrmClass.F_List); //移动相关联的磁性窗体
            Cla_FrmClass.FrmInitialize(FrmClass.F_List); //设置 Frm_ListBox 窗体的位置
            if (FrmClass.Example_Assistant_AdhereTo) //如果两个辅窗体是磁性窗体
            {
                Cla_FrmClass.ManyFrmMove(this, e, FrmClass.F_Libretto); //移动相关联的磁性窗体
                Cla_FrmClass.FrmInitialize(FrmClass.F_Libretto); //设置 Frm_Libretto 窗体的位置
            }
        }
        if (FrmClass.Example_Libretto_AdhereTo) //如果 Frm_Libretto 窗体是磁性窗体
        {
            Cla_FrmClass.ManyFrmMove(this, e, FrmClass.F_Libretto); //移动相关联的磁性窗体
            Cla_FrmClass.FrmInitialize(FrmClass.F_Libretto); //设置 Frm_Libretto 窗体的位置
            if (FrmClass.Example_Assistant_AdhereTo) //如果两个辅窗体是磁性窗体
            {
                Cla_FrmClass.ManyFrmMove(this, e, FrmClass.F_List); //移动相关联的磁性窗体
                Cla_FrmClass.FrmInitialize(FrmClass.F_List); //设置 Frm_ListBox 窗体的位置
            }
        }
        Cla_FrmClass.FrmInitialize(this); //设置当前窗体的位置
    }
}

```

(8) 自定义方法 ManyFrmMove 用于在主窗体移动时，获取鼠标在屏幕上的位置，然后计算出相关磁性窗体的移动位置，并实现磁性窗体的移动效果，代码如下。

```

#region 磁性窗体的移动
///<summary>
///磁性窗体的移动
///</summary>
///<param Frm="Form">窗体</param>
///<param e="MouseEventArgs">控件的移动事件</param>
///<param Follow="Form">跟随窗体</param>
//Form 或 MouseEventArgs 添加命名空间 using System.Windows.Forms;
public void ManyFrmMove(Form Frm, MouseEventArgs e, Form Follow)
{
    if (e.Button == MouseButtons.Left)
    {
        int Tem_Left = 0;
        int Tem_Top = 0;
        Point myPosition = Control.MousePosition; //获取当前鼠标的屏幕坐标
        switch (Follow.Name)
        {

```

```

case "Frm_ListBox": //Frm.ListBox子窗体
{
    //“列表”窗体与“播放”窗体的位置差
    Tem.Top = Example_List_space.Top - FrmPoint.Y;
    Tem.Left = Example_List_space.Left - FrmPoint.X;
    break;
}
case "Frm_Libretto": //Frm.Libretto子窗体
{
    //Frm.Libretto子窗体与主窗体的位置差
    Tem.Top = Example.Libretto_space.Top - FrmPoint.Y;
    Tem.Left = Example.Libretto_space.Left - FrmPoint.X;
    break;
}
}
myPosition.Offset(Tem.Left, Tem.Top); //获取设置后的坐标
Follow.DesktopLocation = myPosition; //设置跟随窗体的位置
}
}

#endregion

```

(9) 自定义方法 FrmInitialize 用于获取各窗体的初始化位置及移动后的位置，代码如下。

```

#region 对窗体的位置进行初始化
///<summary>
///对窗体的位置进行初始化
///</summary>
///<param Frm="Form">窗体</param>
public void FrmInitialize(Form Frm)
{
    switch (Frm.Name)
    {
        case "Frm_Play": //主窗体
        {
            Example_Play.Top = Frm.Top; //设置主窗体的上边距
            Example_Play.Left = Frm.Left; //设置主窗体的左边距
            Example_Play.Width = Frm.Width; //设置主窗体的宽度
            Example_Play.Height = Frm.Height; //设置主窗体的高度
            break;
        }
        case "Frm.ListBox": //Frm.ListBox子窗体
        {
            Example_List.Top = Frm.Top;
            Example_List.Left = Frm.Left;
            Example_List.Width = Frm.Width;
            Example_List.Height = Frm.Height;
            break;
        }
        case "Frm.Libretto": //Frm.Libretto子窗体
        {
            Example.Libretto.Top = Frm.Top;
            Example.Libretto.Left = Frm.Left;
        }
    }
}

```

```

        Example_Libretto_Width = Frm.Width;
        Example_Libretto_Height = Frm.Height;
        break;
    }
}
#endregion

```

(10) 当释放鼠标时存储各窗体的当前信息，以便于计算是否为磁性窗体，代码如下。

```

private void panel_Title_MouseUp(object sender, MouseEventArgs e)
{
    Cls_FrmClass.FrmPlace(this);
}

```

(11) 自定义方法 FrmPlace 用于存储当前窗体的位置和大小，并通过设置后的信息计算当前窗体是否与其他窗体成为磁性窗体，代码如下。

```

#region 存储各窗体的当前信息
///<summary>
///存储各窗体的当前信息
///</summary>
///<param Frm="Form">窗体</param>
public void FrmPlace(Form Frm)
{
    FrmInitialize(Frm);                                //对窗体的位置进行初始化
    FrmMagnetism(Frm);                               //磁性窗体的设置
}
#endregion

```

(12) 自定义方法 FrmMagnetism 用于根据指定的窗体名称，计算当前窗体与其他两个窗体是否能成为磁性窗体，代码如下。

```

#region 窗体的磁性设置
///<summary>
///窗体的磁性设置
///</summary>
///<param Frm="Form">窗体</param>
public void FrmMagnetism(Form Frm)
{
    if (Frm.Name != "Frm_Play")                      //如果不是主窗体
    {
        FrmMagnetismCount(Frm, Example_Play_Top, Example_Play_Left, Example_Play_Width,
Example_Play_Height, "Frm_Play");                  //计算当前窗体与主窗体是否为磁性窗体
    }
    if (Frm.Name != "Frm_ListBox")                    //如果不是 Frm_ListBox 子窗体
    {
        FrmMagnetismCount(Frm, Example_List_Top, Example_List_Left, Example_List_Width,
Example_List_Height, "Frm_ListBox");                //计算当前窗体与 Frm_ListBox 子窗体是否为磁性窗体
    }
    if (Frm.Name != "Frm.Libretto")                   //如果不是 Frm.Libretto 子窗体
    {
        FrmMagnetismCount(Frm, Example.Libretto_Top, Example.Libretto_Left, Example.Libretto_Width,
Example.Libretto_Height, "Frm.Libretto");           //计算当前窗体与 Frm.Libretto 子窗体是否为磁性窗体
    }
}

```

```

    FrmInitialize(Frm);
}
#endifregion

```

(13) 自定义方法 FrmMagnetismCount 用于计算两个窗体是否为磁性窗体，主要是当两个窗体上、下、左、右的间隔小于 10 时，将两个窗体自动粘贴到一起，代码如下。

```

#region 磁性窗体的计算
///<summary>
///磁性窗体的计算
///</summary>
///<param Frm="Form">子窗体</param>
///<param top="int">主窗体上边距</param>
///<param left="int">主窗体左边距</param>
///<param width="int">主窗体的宽度</param>
///<param height="int">主窗体的高度</param>
///<param Mforms="string">主窗体</param>
public void FrmMagnetismCount(Form Frm, int top, int left, int width, int height, string Mforms)
{
    bool Tem_Magnetism = false;                                //判断是否有磁性发生
    string Tem_MainForm = "";                                  //临时记录主窗体
    string Tem_AssistForm = "";                               //临时记录辅窗体
    bool TemUpDown = false;                                 //鼠标是否按下
    bool Tem_LeftRight = false;                            //下面进行磁性窗体计算
    if ((Frm.Top + Frm.Height - top) <= Example_FSpace && (Frm.Top + Frm.Height - top) >= -Example_FSpace)
    {
        //当一个主窗体不包含辅窗体时
        if ((Frm.Left >= left && Frm.Left <= (left + width)) || ((Frm.Left + Frm.Width) >= left && (Frm.Left + Frm.Width) <= (left + width)))
        {
            Frm.Top = top - Frm.Height;
            if ((Frm.Left - left) <= Example_FSpace && (Frm.Left - left) >= -Example_FSpace)
                Frm.Left = left;
            Tem_Magnetism = true;
        }
        //当一个主窗体包含辅窗体时
        if (Frm.Left <= left && (Frm.Left + Frm.Width) >= (left + width))
        {
            Frm.Top = top - Frm.Height;
            if ((Frm.Left - left) <= Example_FSpace && (Frm.Left - left) >= -Example_FSpace)
                Frm.Left = left;
            Tem_Magnetism = true;
        }
        if (Tem_Magnetism == true)
        {
            if (Frm.Left == left)
                Tem_Magnetism = false;
        }
    }
    else
        Tem_Magnetism = false;
}

```

```

//下面进行磁性窗体
if ((Frm.Top - (top + height)) <= Example_FSpace && (Frm.Top - (top + height)) >= -Example_FSpace)
{
    //当一个主窗体不包含辅窗体时
    if ((Frm.Left >= left && Frm.Left <= (left + width)) || ((Frm.Left + Frm.Width) >= left && (Frm.Left + Frm.Width) <= (left + width)))
    {
        Frm.Top = top + height;
        if ((Frm.Left - left) <= Example_FSpace && (Frm.Left - left) >= -Example_FSpace)
            Frm.Left = left;
        Tem_Magnetism = true;
    }
    //当一个主窗体包含辅窗体时
    if (Frm.Left <= left && (Frm.Left + Frm.Width) >= (left + width))
    {
        Frm.Top = top + height;
        if ((Frm.Left - left) <= Example_FSpace && (Frm.Left - left) >= -Example_FSpace)
            Frm.Left = left;
        Tem_Magnetism = true;
    }
    if (Tem_Magnetism == true)
    {
        if (Frm.Left == left)
            Tem_Magnetism = false;
    }
}
else
    Tem_Magnetism = false;
//左面进行磁性窗体
if ((Frm.Left + Frm.Width - left) <= Example_FSpace && (Frm.Left + Frm.Width - left) >= -Example_FSpace)
{
    //当一个主窗体不包含辅窗体时
    if ((Frm.Top > top && Frm.Top <= (top + height)) || ((Frm.Top + Frm.Height) >= top && (Frm.Top + Frm.Height) <= (top + height)))
    {
        Frm.Left = left - Frm.Width;
        if ((Frm.Top - top) <= Example_FSpace && (Frm.Top - top) >= -Example_FSpace)
            Frm.Top = top;
        Tem_Magnetism = true;
    }
    //当一个主窗体包含辅窗体时
    if (Frm.Top <= top && (Frm.Top + Frm.Height) >= (top + height))
    {
        Frm.Left = left - Frm.Width;
        if ((Frm.Top - top) <= Example_FSpace && (Frm.Top - top) >= -Example_FSpace)
            Frm.Top = top;
        Tem_Magnetism = true;
    }
}
else

```

```

    Tem_Magnetism = false;
    //右面进行磁性窗体
    if ((Frm.Left - (left + width)) <= Example_FSpace && (Frm.Left - (left + width)) >= -Example_FSpace)
    {
        //当一个主窗体不包含辅窗体时
        if ((Frm.Top > top && Frm.Top <= (top + height)) || ((Frm.Top + Frm.Height) >= top && (Frm.Top + Frm.Height) <= (top + height)))
        {
            Frm.Left = left + width;
            if ((Frm.Top - top) <= Example_FSpace && (Frm.Top - top) >= -Example_FSpace)
                Frm.Top = top;
            Tem_Magnetism = true;
        }
        //当一个主窗体包含辅窗体时
        if (Frm.Top <= top && (Frm.Top + Frm.Height) >= (top + height))
        {
            Frm.Left = left + width;
            if ((Frm.Top - top) <= Example_FSpace && (Frm.Top - top) >= -Example_FSpace)
                Frm.Top = top;
            Tem_Magnetism = true;
        }
    }
    else
    {
        Tem_Magnetism = false;
        //下面来判断两个辅窗体是否为磁性窗体
        if (Frm.Name == "Frm_Play")
            Tem_MainForm = "Frm_Play";
        else
            Tem_AssistForm = Frm.Name;
        if (Mforms == "Frm_Play")
            Tem_MainForm = "Frm_Play";
        else
            Tem_AssistForm = Mforms;
        if (Tem_MainForm == "")
            //如果是两个辅窗体间的移动
        {
            Example_Assistant_AdhereTo = Tem_Magnetism;
        }
        else
        {
            switch (Tem_AssistForm)
            {
                case "Frm_ListBox":
                    Example_List_AdhereTo = Tem_Magnetism; //移动的是 Frm_ListBox 子窗体
                    break;
                case "Frm_Libretto":
                    Example_Libretto_AdhereTo = Tem_Magnetism; //移动的是 Frm.Libretto 子窗体
                    break;
            }
        }
    }
    #endregion
}

```

**DIY：**完善以上程序，使制作完成的磁性窗体能够动态地显示和隐藏。提示：可以在原有主窗体上增加两个按钮，以便分别控制两个磁性窗体的显示和隐藏。(20 分)(实例位置：光盘\mr\08\qjyy\03\_diy)

### 8.5.4 情景应用 4——制作鼠标穿透窗体

视频讲解：光盘\mr\08\lx\制作鼠标穿透窗体.exe

实例位置：光盘\mr\08\qjyy\04

用户在对桌面进行操作时，为了使桌面更加美观，可以在桌面的上面加一层类似于玻璃的效果，用户可以用鼠标透过“玻璃”对桌面进行操作，这里通过使用鼠标穿透窗体来实现以上功能。运行效果如图 8.36 所示。

实现过程如下。

(1) 创建一个 Windows 窗体应用程序，并将其命名为 MouseThroughForm。

(2) 更改默认窗体 Form1 的 Name 属性为 Frm\_Main，并将该窗体的 FormBorderStyle 属性设置为 None。在 Frm\_Main 窗体中添加一个 NotifyIcon 控件，并设置其 Icon 属性为指定的图标，用来显示提示信息；添加一个 ContextMenuStrip 控件，用来作为程序的快捷菜单。

(3) 在 Frm\_Main 窗体中，首先自定义一个方法 CanPenetrate，该方法用来通过 API 函数 SetWindowLong 和 GetWindowLong 实现鼠标的穿透效果。CanPenetrate 方法实现代码如下。

```
private void CanPenetrate()
{
    uint intExTemp = GetWindowLong(this.Handle, GWL_EXSTYLE); //从当前窗口的结构中取得信息
    //在窗口结构中为当前窗口设置信息
    uint oldGWLEX = SetWindowLong(this.Handle, GWL_EXSTYLE, WS_EX_TRANSPARENT | WS_EX_LAYERED);
}
```

(4) Frm\_Main 窗体加载时，使窗体不出现在 Windows 任务栏中，并且使鼠标穿透窗体，代码如下。

```
private void Form1_Load(object sender, EventArgs e)
{
    this.ShowInTaskbar = false; //窗体不出现在 Windows 任务栏中
    CanPenetrate(); //使窗体始终在其他窗体之上
    this.TopMost = true;
}
```

**DIY：**根据以上程序的实现原理，制作一个带有穿透效果的日历。(20 分)(实例位置：光盘\mr\08\qjyy\04\_diy)

### 8.5.5 情景应用 5——窗体换肤程序

视频讲解：光盘\mr\08\lx\窗体换肤程序.exe

实例位置：光盘\mr\08\qjyy\05

在默认情况下，Windows 窗体的皮肤由操作系统的主题设置和外观设置来决定，但有些应用软件为了摆脱操作系统的这种束缚，已经开发出自定义窗体皮肤的功能，如常见的播放软件暴风影音、PPLive 等。

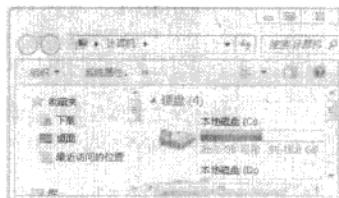


图 8.36 制作鼠标穿透窗体

在本实例的窗体中单击鼠标右键，将弹出一个用于更换窗体皮肤的快捷菜单，在该快捷菜单中选择“换皮肤”菜单下的任意命令，程序将为当前窗体更换皮肤。运行效果如图 8.37 所示。

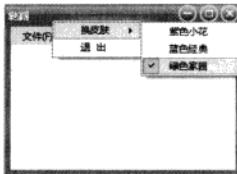


图 8.37 窗体换肤程序

实现过程如下。

- (1) 创建一个 Windows 窗体应用程序，并将其命名为 WinCusSkin。
- (2) 更改默认窗体 Form1 的 Name 属性为 Frm\_Main，在该窗体中添加 6 个 Panel 控件，分别用来作为窗体的标题栏、下边框、左边框、右边框、左下角边框和右下角边框；添加一个 ContextMenuStrip 控件，用来作为更换皮肤的快捷菜单；添加 3 个 PictureBox 控件，分别用来显示最大化、最小化和关闭图片。
- (3) 在 Frm\_Main 窗体中单击鼠标右键，将弹出一个用于更换窗体皮肤的快捷菜单，在该快捷菜单中选择“换皮肤”菜单下的任意命令，程序将为当前窗体更换皮肤。程序为窗体更换皮肤主要是通过设置 Panel 控件的 BackgroundImage 属性和设置 PictureBox 控件的 Image 属性来实现的，这里以单击“紫色小花”菜单项为例，单击该菜单项将触发它的 Click 事件，代码如下。

```
private void menItemSkin1_Click(object sender, EventArgs e)
{
    //设置窗体顶部 Panel 控件的 BackgroundImage 属性
    this.panel_Top.BackgroundImage = Image.FromFile(strImagePath + @"\images\purple\top.png");
    //设置窗体左侧 Panel 控件的 BackgroundImage 属性
    this.panel_Left.BackgroundImage = Image.FromFile(strImagePath + @"\images\purple\left.png");
    //设置窗体右侧 Panel 控件的 BackgroundImage 属性
    this.panel_Right.BackgroundImage = Image.FromFile(strImagePath + @"\images\purple\right.png");
    //设置窗体底部 Panel 控件的 BackgroundImage 属性
    this.panel_Bottom.BackgroundImage = Image.FromFile(strImagePath + @"\images\purple\bottom.png");
    //设置最小化图片控件的 Image 属性
    this.picMinimize.Image = Image.FromFile(strImagePath + @"\images\purple\min.png");
    if (bol == true)                                //若当前窗体处于最大化状态
    {
        //设置最大化图片控件的 Image 属性
        this.picMaximize.Image = Image.FromFile(strImagePath + @"\images\purple\max.png");
    }
    else                                              //若当前窗体处于普通状态
    {
        //设置最大化图片控件的 Image 属性
        this.picMaximize.Image = Image.FromFile(strImagePath + @"\images\purple\max_normal.png");
    }
    //设置关闭图片控件的 Image 属性
    this.picClose.Image = Image.FromFile(strImagePath + @"\images\purple\close.png");
    this.menItemSkin1.Checked = true;                  //设置菜单的选中标记
    this.menItemSkin2.Checked = false;                 //取消菜单的选中标记
    this.menItemSkin3.Checked = false;                 //取消菜单的选中标记
    //设置窗体主菜单的背景图像属性
```

```

this.menuStrip1.BackgroundImage = Image.FromFile(strImagesPath + @"\images\purple\menu.gif");
//设置窗体的背景图像属性
this.BackgroundImage = Image.FromFile(strImagesPath + @"\images\purple\background.gif");
}

```

(4) 在 Frm\_Main 窗体中单击“最大化”按钮，窗体的大小将会在最大化状态和普通状态之间切换。窗体实现这种大小状态的切换，主要是通过设置它的 Height 属性和 Width 属性来实现的，“最大化”按钮的 Click 事件代码如下。

```

private void picMaximize_Click(object sender, System.EventArgs e)
{
    if (bol)
    {
        top = this.Top;
        left = this.Left;
        hei = this.Height;
        wid = this.Width;
        this.Top = 0;
        this.Left = 0;
        int hg = SystemInformation.MaxWindowTrackSize.Height;
        int wh = SystemInformation.MaxWindowTrackSize.Width;
        this.Height = hg;
        this.Width = wh;
        bol = true;
        if (menuItemSkin1.Checked)
            //设置最大化图片的 Image 属性
            this.picMaximize.Image = Image.FromFile(strImagesPath + @"\images\purple\max.png");
        if (menuItemSkin2.Checked)
            //若选择“蓝色经典”菜单项
            this.picMaximize.Image = Image.FromFile(strImagesPath + @"\images\blue\max.png");
        if (menuItemSkin3.Checked)
            //若选择“绿色家园”菜单项
            this.picMaximize.Image = Image.FromFile(strImagesPath + @"\images\green\max.png");
    }
    else
    {
        this.Top = top;
        this.Left = left;
        this.Height = hei;
        this.Width = wid;
        bol = false;
        if (menuItemSkin1.Checked)
            this.picMaximize.Image = Image.FromFile(strImagesPath + @"\images\purple\max_Normal.png");
        if (menuItemSkin2.Checked)
            this.picMaximize.Image = Image.FromFile(strImagesPath + @"\images\blue\max_Normal.png");
        if (menuItemSkin3.Checked)
            this.picMaximize.Image = Image.FromFile(strImagesPath + @"\images\green\max_Normal.png");
    }
}

```

//若窗体处于普通状态

//获取窗体的 Top 属性值  
//获取窗体的 Left 属性值  
//获取窗体的 Height 属性值  
//获取窗体的 Width 属性值  
//设置窗体的 Top 属性值为零  
//设置窗体的 Left 属性值为零  
//获取窗口的默认最大高度  
//获取窗口的默认最大宽度  
//设置窗体的 Height 属性值  
//设置窗体的 Width 属性值  
//设置窗体标记表示最大化  
//若选择“紫色小花”菜单项

//若选择“蓝色经典”菜单项  
//若选择“绿色家园”菜单项

//若窗体处于最大化状态  
//设置窗体的 Top 属性值  
//设置窗体的 Left 属性值  
//设置窗体的 Height 属性值  
//设置窗体的 Width 属性值  
//设置窗体标记表示普通状态

**DIY：**完善以上程序，使其自定义的最小化按钮、最大化按钮和关闭按钮在鼠标滑过时具有动态效果。  
**提示：**可以通过图片控件的 MouseEnter 事件和 MouseLeave 事件实现。(20 分)(实例位置：光盘\mr\08\qjyy\05\_diy)

情景应用 DIY 栏目分数统计：

DIY 题目	1	2	3	4	5	总分数	
分数							

## 8.6 自我测试

### 一、选择题（每题 10 分，5 道题）

1. 如果不想让窗体显示最大化按钮，则应设置窗体的（ ）属性。  
A. MinimizeBox      B. MaximizeBox      C. AcceptButton      D. CancelButton
2. 下列描述正确的是（ ）。  
A. Form1.Hide()与 Form1.Visible=true 是等价的    B. Form1.Hide()与 Form1.Visible=false 是等价的  
C. Form1.Close()与 Form1.Visible=true 是等价的    D. Form1.Close()与 Form1.Visible=false 是等价的
3. 以下叙述错误的是（ ）。  
A. 无论项目中有多少个窗体，只能有一个启动窗体      B. 一个项目中只能有一个 Main 方法  
C. 窗体的 Show 方法和 ShowDialog 方法都能够显示窗体    D. 窗体间无法传递数据
4. 以下有关 MDI 应用程序的描述中，错误的是（ ）。  
A. MDI 子窗体最小化时，图标显示在任务栏上  
B. MDI 子窗体在 MDI 主窗体的内部区域显示  
C. MDI 主窗体可以容纳多个 MDI 子窗体  
D. 当 MDI 子窗体最大化时，其标题与 MDI 主窗体标题合并
5. 下面能够显示模式窗体的 C#语句是（ ）。  
A. PrintDialog.ShowDialog();                                  B. ColorDialog.ShowDialog();  
C. form.ShowDialog();                                         D. form.Show();

### 二、填空题（每题 10 分，5 道题）

1. 要想在窗体上添加控件，可以使用（ ）。
2. 通过把窗体的（ ）属性设置为 True，可以使一个窗体称为 MDI 主窗体。
3. 通过设置窗体的（ ）属性，可以使窗体成为顶层窗体。
4. Windows 窗体应用程序中的 resx 文件表示（ ）。
5. 有如下语句：  
`MessageBox.Show("显示按钮数","显示",MessageBoxButtons.YesNoCancel);`

使用上面语句弹出的消息框中有（ ）个按钮。

测试分数统计：

类别	第 1 题	第 2 题	第 3 题	第 4 题	第 5 题
选择题分数					
填空题分数					
总分数					

## 8.7 行动指南

开始日期： 年 月 日

结束日期： 年 月 日

序号	内 容	行动指南		
1	照猫画虎栏目 分数( )	分数>75 分	优秀，基本功掌握得不错，加油！	
		75 分>分数>50 分	及格，知识掌握得不牢，重新做一遍照猫画虎。	
		分数<50 分	请使用光盘中提供的“实战能力测试系统”进行提高训练。	
	情景应用栏目 分数( )	分数>75 分	优秀，综合应用能力很强。	
		75 分>分数>50 分	及格，综合应用能力需提高，再练习一遍情景应用。	
		分数<50 分	请使用光盘中提供的“实战能力测试系统”进行提高训练。	
	自我测试栏目 分数( )	分数>75 分	优秀，有成为编程高手的潜质。	
		分数<75 分	请使用光盘中提供的“实战能力测试系统”进行提高训练。	
	综合评价	反复训练后，以上各项分数都在 75 分以上，方可进入下一堂课学习。		
	编程能力培养：本栏目提供了一些实践题目，对于培养编程能力很有效，要做好。	1. 开发一个程序，使用树型列表动态显示窗体中的菜单。 2. 开发一个程序，使其实现类似 QQ 弹出窗口的 POP 提醒窗口。 3. 制作一个类似 Windows XP 的程序界面。		
2				
3	编程习惯培养：本课堂培养读者在学习开发中记笔记的习惯，把开发中遇到的问题、总结的经验记录下来。			
4	创新能力培养：看看身边有没有可以用编程解决的问题，记录到右边的表格中。根据学习进度，尝试编写解决这些问题的小程序。			

## 8.8 成功可以复制——迅雷创始人邹胜龙

“迅雷”于 2002 年年底由邹胜龙及程浩先生始创于美国硅谷。2003 年 1 月底，邹胜龙回国发展，在深圳市创办三代科技开发有限公司。2005 年 5 月，公司正式更名为深圳市迅雷网络技术有限公司，其寓意就是指下载速度可达到迅雷不及掩耳之势。

“迅雷”立足于为全球互联网提供最好的多媒体下载服务。经过艰苦创业，“迅雷”在大中华区域以领先的技术和诚信的服务，赢得了广大用户的深深喜爱和许多合作伙伴的认同与支持。公司旗舰产品——迅雷，已经成为中国互联网最流行的应用服务软件之一。作为中国最大的下载服务提供商，迅雷每天服务

来自几十个国家、超过数千万次的下载。如今迅雷每天为全球互联网传送 3200000GB 的资源，并以其下载软件的迅捷高速，每天为人类节省 1500 年时间。在本土，迅雷的市场覆盖率达 85%；迄今，全球已有 1.88 亿网民体验过了迅雷提供的服务。随着中国互联网宽带的普及，迅雷凭借“简单、高速”的下载体验，已经成为高速下载的代名词。此外，“迅雷”也获得了晨光科技和 IDGVC 等数家知名风险投资企业的认同和合资。2007 年 1 月迅雷宣布第三次融资成功，本次融资的领衔投资是联创策源（Ceyuan Ventures），参与投资的有晨光（Morningside Ventures）、IDGVC、Fidelity Asia Ventures，战略投资是 Google（谷歌）。这些投资合作伙伴除了给“迅雷”带来了更加雄厚的资金实力，也给“迅雷”带来了更丰富的行业资源和国际化公司运行环境。

### 经典语录

迅雷的团队是一只学习能力和执行能力都很强的队伍，他们是中国互联网最优秀的团队之一。

### 深度评价

在网易科技采访邹胜龙时说“您已经入选超过 5 万网友选出的互联网领袖扑克牌，并且进入了前十名，方便谈一下您的感受吗？”邹胜龙说：“非常荣幸！不过，我个人感觉受之有愧！其实，大家对我的厚爱主要是来自于对于迅雷的认识，而迅雷主要是由我的合伙人程浩和我们的团队一起建设的。相比之下，迅雷还非常年轻，还有很长的道路要走下去。”通过邹先生的这一番话，就可以了解到，任何一个成功的项目都不会是一个人完成的，想要在 IT 领域中获得成功，就要做好团队合作的准备，凝聚大家的力量，发挥大家的聪明才智，才能创造财富。





# 第 9 堂课

## Windows 应用程序常用控件

( 视频讲解：198分钟)

控件是窗体设计的基本组成单位，通过使用控件可以高效地开发 Windows 应用程序，所以，熟练掌握控件是合理、有效地进行程序开发的重要前提。本堂课详细地介绍了 Windows 应用程序的常用控件及使用，讲解过程中为了便于读者理解结合了大量的举例。

学习摘要：

- ▶ 了解控件的分类及作用
- ▶ 掌握控件的相关操作
- ▶ 掌握文本类控件的使用
- ▶ 掌握选择类控件的使用
- ▶ 掌握分组控件的使用
- ▶ 掌握对话框控件的使用
- ▶ 掌握菜单、工具栏和状态栏控件的使用

## 9.1 控件概述

Windows 窗体作为程序界面设计的主体框架，它完成的大部分功能都是由其上面排列的控件来实现的。控件直接或间接派生于 System.Windows.Forms.Control 基类，它负责在容器中（这个容器既可以是一个“普通”窗体，也可以是一个具有“容器性质”的控件）绘制操作界面和实现操作功能，并且使用控件可以极大地提高应用程序的开发效率。Visual Studio 开发环境中提供了许多控件，在默认情况下，可以从“工具箱”窗口中获得这些控件。

### 9.1.1 浏览常用控件

窗体是由控件有机构成的，所以熟悉控件是进行合理、有效程序开发的重要前提。Windows 应用程序中的控件大体分为常用控件和高级控件。Windows 应用程序中的常用控件如图 9.1 所示。

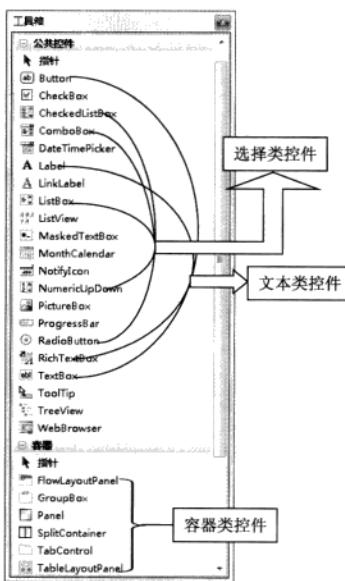


图 9.1 Windows 应用程序中的常用控件

**说明：**本堂课将对 Windows 常用控件进行介绍，Windows 应用程序的高级控件将会在第 10 堂课中进行介绍。

### 9.1.2 控件的分类及作用

在 Visual Studio 2008 开发环境中，常用的控件主要有文本类控件、选择类控件及分组控件等。Windows 应用程序控件的基类是位于 System.Windows.Forms 命名空间下的 Control 类，该类定义了控件类的公有属性、方法和事件，其他的控件类都直接或间接地派生自这个基类。几种常用控件的作用如表 9.1 所示。

表 9.1 常用控件的作用

控件分类	作用
文本类控件	文本类控件可以在控件上显示文本
选择类控件	主要为用户提供选择的项目
分组控件	使用分组控件可以将窗体中的其他控件进行分组处理

## 9.2 控件的相关操作

控件的相关操作主要包括添加控件、对齐控件、锁定控件和删除控件等，本节将对这些操作进行详细介绍。

### 9.2.1 添加控件

向窗体上添加控件时，主要通过“在窗体上绘制控件”和“将控件拖曳到窗体上”这两种方式进行添加，下面分别介绍。

#### 1. 在窗体上绘制控件

在“工具箱”中单击要添加到窗体的控件，然后在该窗体上单击希望控件左上角位于的位置，然后拖动到希望该控件右下角位于的位置，控件按指定的位置和大小添加到窗体中。例如，在窗体上绘制一个 Button 控件，如图 9.2 所示。

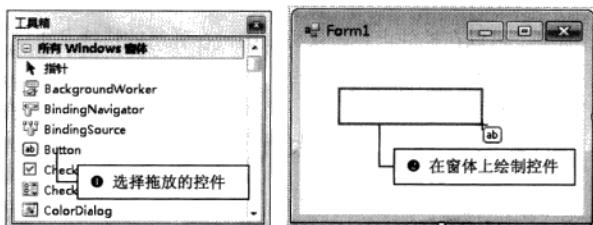


图 9.2 在窗体上绘制 Button 控件

#### 2. 将控件拖曳到窗体上

在“工具箱”中单击所需的控件并将其拖到窗体上，控件以其默认大小添加到窗体上的指定位置。例如，将一个 Button 控件拖曳到窗体上，如图 9.3 所示。

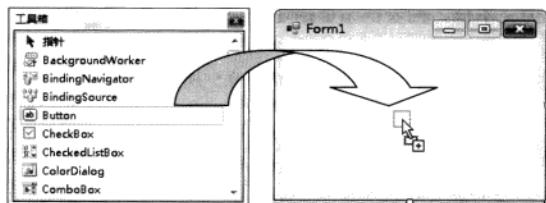


图 9.3 将一个 Button 控件拖曳到窗体上

### 9.2.2 对齐控件

选定一组控件，在执行对齐之前，首先选定主导控件（第一个被选定的控件就是主导控件），控件组的最终位置取决于主导控件的位置，然后选择菜单栏中的“格式”/“对齐”命令进行控件对齐设置，如图 9.4 所示。

常用的对齐方式及说明如下。

- 左对齐：将选定控件沿它们的左边对齐。
- 居中对齐：将选定控件沿它们的中心点水平对齐。
- 右对齐：将选定控件沿它们的右边对齐。
- 顶端对齐：将选定控件沿它们的顶边对齐。
- 中间对齐：将选定控件沿它们的中心点垂直对齐。
- 底端对齐：将选定控件沿它们的底边对齐。



图 9.4 选择“格式”/“对齐”命令

### 9.2.3 锁定控件

在锁定窗体中的控件时，需要在“属性”窗口中将其 Locked 属性设置为 True，此外，还可以通过右键单击控件选择“锁定控件”命令进行实现。如果要锁定窗体上的所有控件，则可以选择菜单栏中的“格式”/“锁定控件”命令。

### 9.2.4 删除控件

删除控件的方法非常简单，可以在控件上单击鼠标右键，在弹出的快捷菜单中选择“删除”命令进行删除，也可以选中控件，然后按 Delete 键进行删除。

## 9.3 文本类控件

文本类控件主要包括标签控件（Label 控件）、按钮控件（Button 控件）、文本框控件（TextBox 控件）和有格式文本控件（RichTextBox 控件）等，本节将对文本类控件及其使用进行详细讲解。

### 9.3.1 标签控件

标签控件（Label 控件）主要用于显示用户不能编辑的文本以及标识窗体上的对象（例如，为文本框、列表框添加描述信息等），另外，也可以通过编写代码来设置要显示的文本信息。如果添加一个标签控件，系统会自动创建标签控件的一个对象。如图 9.5 所示为标签控件（Label 控件）。

下面详细介绍标签控件（Label 控件）的常见用法。

#### 1. 设置标签文本

可以通过两种方法设置标签控件（Label 控件）显示的文本，第一种是直接在标签控件（Label 控件）

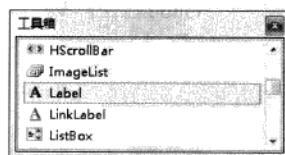


图 9.5 标签控件（Label 控件）

的属性面板中设置 Text 属性，第二种是通过代码设置 Text 属性。

**例 9.01** 向窗体中拖曳一个 Label 控件，然后将其显示文本设置为“C#编程词典”，如图 9.6 所示。也可以通过代码设置 Text 属性，代码如下。

```
label1.Text = "C#编程词典";
```

//设置 Label 控件的 Text 属性

## 2. 显示/隐藏标签控件

显示/隐藏标签控件（Label 控件）可以通过设置 Visible 属性来实现，即如果 Visible 属性的值为 True，则显示控件；如果 Visible 属性的值为 False，则隐藏控件。

**例 9.02** 显示标签控件（Label 控件），将其 Visible 属性设置为 True，实现的方法同样有两种，如图 9.7 所示。



图 9.6 设置 Text 属性



图 9.7 设置 Visible 属性

如果要显示标签控件（Label 控件），可通过代码将其 Visible 属性设置为 True，代码如下。

```
label1.Visible = true;
```

//设置 Label 控件的 Visible 属性

### 9.3.2 按钮控件

按钮控件（Button 控件）允许用户通过单击来执行操作。按钮控件（Button 控件）既可以显示文本，也可以显示图像。当该控件被单击时，它看起来像是被按下，然后被释放。如图 9.8 所示为按钮控件（Button 控件）。

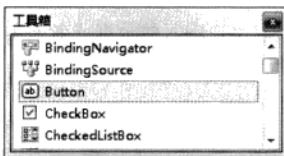


图 9.8 按钮控件（Button 控件）

下面详细介绍按钮控件（Button 控件）的常见用法。

#### 1. 响应按钮的单击事件

单击按钮控件（Button 控件）时将引发 Click 事件，执行 Click 事件中的代码。

**例 9.03** 创建一个 Windows 应用程序，单击按钮控件（Button 控件），引发 Click 事件，弹出提示框，代码如下。（实例位置：光盘\mr\09\s\l\9.03）

```
private void button1_Click(object sender, EventArgs e) //按钮的 Click 事件
{
    MessageBox.Show("单击了按钮，引发了 Click 事件"); //弹出提示框
}
```

程序运行结果如图 9.9 所示。

## 2. 将按钮设置为窗体的“接受”按钮

通过设置窗体的 `AcceptButton` 属性，可以设置窗体的“接受”按钮，如果设置了此按钮，则用户每次按下 Enter 键都相当于单击了该按钮。

**例 9.04** 创建一个 Windows 应用程序，将 `button1` 按钮设置为 `Form1` 窗体的“接受”按钮，运行程序，当按下 Enter 键时，就会触发 `button1` 按钮的 Click 事件，与单击 `button1` 按钮的结果相同。代码如下。（实例位置：光盘\mr\09\sl\9.04）

```
private void Form1_Load(object sender, EventArgs e)           //窗体的 Load 事件
{
    this.AcceptButton = button1;                            //将 button1 按钮设置为窗体的“接受”按钮
}
private void button1_Click(object sender, EventArgs e)        //按钮的 Click 事件
{
    MessageBox.Show("引发了接受按钮");                      //弹出提示框
}
```

运行程序，按下 Enter 键，触发 `button1` 按钮，弹出信息提示框，如图 9.10 所示。

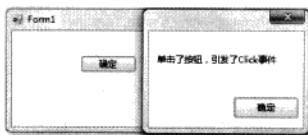


图 9.9 响应按钮的单击事件



图 9.10 将按钮设置为窗体的“接受”按钮

**说明：**通过设置窗体的 `CancelButton` 属性，可以设置窗体的“取消”按钮，如果设置了此按钮，则用户每次按下 Esc 键都相当于单击了该按钮。

### 9.3.3 文本框控件

文本框控件（`TextBox` 控件）用于获取用户输入的数据或者显示文本，它通常用于可编辑文本，也可以使其成为只读控件。文本框可以显示多行，开发人员可以使文本换行以便符合控件的大小。如图 9.11 所示为文本框控件（`TextBox` 控件）。

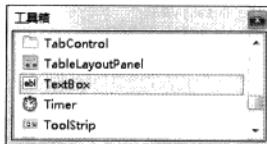


图 9.11 文本框控件（`TextBox` 控件）

下面详细介绍文本框控件（`TextBox` 控件）的常见用法。

#### 1. 创建只读文本框

通过设置文本框控件（`TextBox` 控件）的 `ReadOnly` 属性，可以设置文本框是否只读。如果 `ReadOnly` 属性为 `True`，则不能编辑文本框，而只能通过文本框显示数据；否则，可以编辑文本框中的数据。

**例 9.05** 创建一个 Windows 应用程序，将 `TextBox` 文本框设置为只读，并且在其中显示“C#编程词典”字样，代码如下。（实例位置：光盘\mr\09\sl\9.05）

```
private void Form1_Load(object sender, EventArgs e) //窗体的 Load 事件
{
    textBox1.ReadOnly = true; //将文本框设置为只读
    textBox1.Text = "C#编程词典"; //设置其 Text 属性
}
```

程序运行结果如图 9.12 所示。

## 2. 创建密码文本框

通过设置文本框的 PasswordChar 属性或 UseSystemPasswordChar 属性可以将文本框设置成密码文本框，使用 PasswordChar 属性可以设置输入密码时文本框中显示的字符（例如，将密码显示成“\*”或“#”等）；而如果将 UseSystemPasswordChar 属性设置为 True，则输入密码时，文本框中将密码显示为“\*”。

**例 9.06** 创建一个 Windows 应用程序，使用 PasswordChar 属性将密码文本框中字符自定义显示为“@”，同时将 UseSystemPasswordChar 属性设置为 True，使第二个密码文本框中的字符显示为“\*”，代码如下。

（实例位置：光盘\mr\09\sl\9.06）

```
private void Form1_Load(object sender, EventArgs e) //窗体的 Load 事件
{
    textBox1.PasswordChar = '@'; //设置文本框的 PasswordChar 属性为字符@
    textBox2.UseSystemPasswordChar = true; //设置文本框的 UseSystemPasswordChar 属性为 True
}
```

程序运行结果如图 9.13 所示。

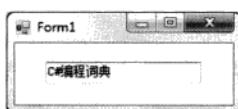


图 9.12 设置文本框为只读



图 9.13 设置 PasswordChar 属性

## 3. 创建多行文本框

默认情况下，文本框控件（TextBox 控件）只允许输入单行数据，如果将其 Multiline 属性设置为 True，文本框控件（TextBox 控件）就可以输入多行数据。

**例 9.07** 创建一个 Windows 应用程序，将文本框的 Multiline 属性设置为 True，使其能够输入多行数据，代码如下。（实例位置：光盘\mr\09\sl\9.07）

```
private void Form1_Load(object sender, EventArgs e) //窗体的 Load 事件
{
    textBox1.Multiline = true; //设置文本框的 Multiline 属性使其多行显示
    textBox1.Text = "昨夜星辰昨夜风，画楼西畔桂堂东。身无彩凤双飞翼，心有灵犀一点通。";
    textBox1.Height = 100; //设置文本框的高度
}
```

程序运行结果如图 9.14 所示。

## 4. 响应文本框的文本更改事件

当文本框中的文本发生更改时，将会引发文本框的 TextChanged 事件。

**例 9.08** 创建一个 Windows 应用程序，在文本框的 TextChanged 事件中编写代码，实现当文本框中的文本更改时，Label 控件实时显示更改后的文本，代码如下。（实例位置：光盘\mr\09\sl\9.08）

```

private void textBox1_TextChanged(object sender, EventArgs e) //文本框的 TextChanged 事件
{
    label1.Text = textBox1.Text;
}

```

程序运行结果如图 9.15 所示。

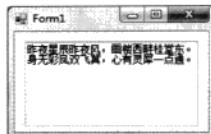


图 9.14 将 Multiline 属性设置为 True

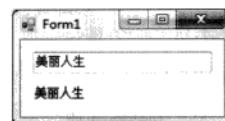


图 9.15 显示更改后的文本

### 9.3.4 有格式文本控件

有格式文本控件（RichTextBox 控件）用于显示、输入和操作带有格式的文本，例如它可以实现显示字体、颜色、链接、从文件加载文本及嵌入的图像、撤销和重复编辑操作以及查找指定的字符等功能。如图 9.16 所示为有格式文本控件（RichTextBox 控件）。

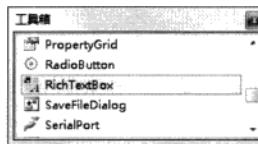


图 9.16 有格式文本控件（RichTextBox 控件）

下面详细介绍有格式文本控件（RichTextBox 控件）的常见用法。

#### 1. 在 RichTextBox 控件中显示滚动条

通过设置 RichTextBox 控件的 Multiline 属性，可以控制控件中是否显示滚动条。Multiline 属性设为 True，则显示滚动条，否则，不显示滚动条。默认情况下，此属性被设置为 True。滚动条分为水平滚动条和垂直滚动条，通过 ScrollBars 属性可以设置如何显示滚动条。ScrollBars 属性的属性值及说明如表 9.2 所示。

表 9.2 ScrollBars 属性的属性值及说明

属性值	说 明
Both	只有当文本超过控件的宽度或长度时，才显示水平滚动条或垂直滚动条，或两个滚动条都显示
None	从不显示任何类型的滚动条
Horizontal	只有当文本超过控件的宽度时，才显示水平滚动条。需要注意的是，此时必须将 WordWrap 属性设置为 False，才会出现这种情况
Vertical	只有当文本超过控件的高度时，才显示垂直滚动条
ForcedHorizontal	当 WordWrap 属性设置为 False 时，显示水平滚动条。在文本未超过控件的宽度时，该滚动条显示为浅灰色
ForcedVertical	始终显示垂直滚动条。在文本未超过控件的长度时，该滚动条显示为浅灰色
ForcedBoth	始终显示垂直滚动条。当 WordWrap 属性设置为 False 时，显示水平滚动条。在文本未超过控件的宽度或长度时，两个滚动条均显示为灰色

例 9.09 创建一个 Windows 应用程序，使 RichTextBox 控件只显示垂直滚动条，首先将 Multiline 属性

设为 True，然后设置 ScrollBars 属性的值为 Vertical，代码如下。（实例位置：光盘\mr\09\sl\9.09）

```
private void Form1_Load(object sender, EventArgs e)
{
    richTextBox1.Multiline = true; //将 Multiline 属性设为 True 实现多行显示
    richTextBox1.ScrollBars = RichTextBoxScrollBars.Vertical; //设置 ScrollBars 属性实现只显示垂直滚动条
}
```

运行程序，向控件中输入数据，如图 9.17 所示。

## 2. 在 RichTextBox 控件中设置字体属性

设置 RichTextBox 控件中的字体属性时可以使用 SelectionFont 属性和 SelectionColor 属性，其中 SelectionFont 属性用来设置字体的字体系列、大小和字样，而 SelectionColor 属性用来设置字体的颜色。

**例 9.10** 创建一个 Windows 应用程序，将 RichTextBox 控件中文本的字体设置为楷体，字体大小设置为 12，字样设置为粗体，文本的颜色设置为蓝色，代码如下。（实例位置：光盘\mr\09\sl\9.10）

```
private void Form1_Load(object sender, EventArgs e) //窗体的 Load 事件
{
    richTextBox1.Multiline = true; //将 Multiline 属性设为 True 实现多行显示
    richTextBox1.ScrollBars = RichTextBoxScrollBars.Vertical; //设置 ScrollBars 属性实现只显示垂直滚动条
    richTextBox1.SelectionFont = new Font("楷体", 12, FontStyle.Bold); //设置文本为楷体，大小为 12，字样是粗体
    richTextBox1.SelectionColor = System.Drawing.Color.Blue; //设置 SelectionColor 实现控件中的文本颜色为蓝色
}
```

程序运行结果如图 9.18 所示。

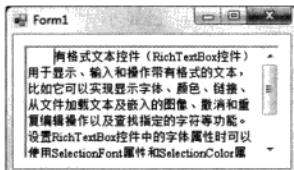


图 9.17 显示垂直滚动条

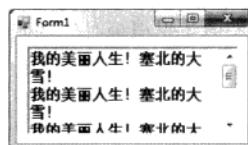


图 9.18 设置控件中文本的字体属性

## 3. 将 RichTextBox 控件显示为超链接样式

RichTextBox 控件可以将 Web 链接显示为彩色或下划线形式，然后通过编写代码，在单击链接时打开浏览器窗口，该窗口中显示链接文本中指定的网站。其设计思路是，首先通过 Text 属性设置控件中含有超链接的文本，然后在控件的 LinkClicked 事件中编写事件处理程序，将所需的文本发送到浏览器。

**例 9.11** 创建一个 Windows 应用程序，在 RichTextBox 控件的文本内容中含有超链接地址，超链接地址显示为彩色且带有下划线，单击该超链接地址后会打开相应的网站，代码如下。（实例位置：光盘\mr\09\sl\9.11）

```
private void Form1_Load(object sender, EventArgs e)
{
    richTextBox1.Multiline = true; //将 Multiline 属性设为 True，实现多行显示
    richTextBox1.ScrollBars = RichTextBoxScrollBars.Vertical; //设置 ScrollBars 属性，实现只显示垂直滚动条
    richTextBox1.Text = "欢迎登录 http://www.mrbccd.com 明日编程词典网"; //设置控件的 Text 属性
}

private void richTextBox1_LinkClicked(object sender, LinkClickedEventArgs e)
//当单击超链接地址时触发 LinkClicked 事件
{
```

```

    System.Diagnostics.Process.Start(e.LinkText); //将超链接地址发送到浏览器
}

```

程序运行结果如图 9.19 所示。

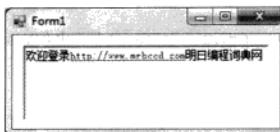


图 9.19 文本中含有超链接地址

## 9.4 选择类控件

选择类控件主要包括下拉组合框控件（ComboBox 控件）、复选框控件（CheckBox 控件）、单选按钮控件（RadioButton 控件）、数值选择控件（NumericUpDown 控件）和列表控件（ListBox 控件）等，本节将对选择类控件及其使用进行详细讲解。

### 9.4.1 下拉组合框控件

下拉组合框控件（ComboBox 控件）用于在下拉组合框中显示数据，该控件主要由两部分组成，其中，第一部分是一个允许用户输入列表项的文本框；第二部分是一个列表框，它显示一个选项列表，用户可以从中选择选项。如图 9.20 所示为 ComboBox 控件。

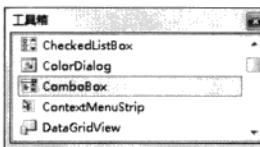


图 9.20 ComboBox 控件

下面详细介绍下拉组合框控件（ComboBox 控件）的常见用法。

#### 1. 创建只可以选择的下拉框

通过设置 ComboBox 控件的 DropDownStyle 属性，可以将其设置成只可以选择的下拉框。DropDownStyle 属性有 3 个属性值，分别对应不同的样式。

- Simple：使得 ComboBox 控件的列表部分总是可见的。
- DropDown：DropDownStyle 属性的默认值，使得用户可以编辑 ComboBox 控件的文本框部分，只有单击右侧的箭头才能显示列表部分。
- DropDownList：用户不能编辑 ComboBox 控件的文本框部分，呈现下拉框的样式。

将 ComboBox 控件的 DropDownStyle 属性设置为 DropDownList，它就只能是可选择的下拉框，而不能编辑文本框部分的内容。

**例 9.12** 创建一个 Windows 应用程序，将 ComboBox 控件的 DropDownStyle 属性设置为 DropDownList，并且向控件中添加 3 项数据，使其成为只可以进行选择操作的下拉框，代码如下。（**实例位置：**光盘\mr\09\sl\9.12）

```

private void Form1_Load(object sender, EventArgs e)           //窗体的 Load 事件
{
    comboBox1.DropDownStyle = ComboBoxStyle.DropDownList;
    //设置 DropDownList 属性，使控件呈现下拉列表的样式
    comboBox1.Items.Add("明日科技");
    comboBox1.Items.Add("C#编程词典");
    comboBox1.Items.Add("C#从基础到项目实战");
}

```

程序运行结果如图 9.21 所示。

## 2. 响应下拉组合框的选项值更改事件

当下拉列表的选择项发生改变时，将会引发控件的 SelectedValueChanged 事件。

**例 9.13** 创建一个 Windows 应用程序，当下拉列表的选择项发生改变时，引发 ComboBox 控件的 SelectedValueChange 事件。在 SelectedValueChange 事件中，使 Label 控件的 Text 属性等于控件的选择项，代码如下。（实例位置：光盘\mr\09\s\9.13）

```

private void Form1_Load(object sender, EventArgs e)
{
    comboBox1.DropDownStyle = ComboBoxStyle.DropDownList;
    //设置 DropDownList 属性，使控件呈现下拉列表的样式
    //向控件中添加项目
    comboBox1.Items.Add("明日科技");
    comboBox1.Items.Add("C#编程词典");
    comboBox1.Items.Add("C#从基础到项目实战");
}

private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)          //使 Label 控件的 Text 属性等于控件的选择项
{
    label1.Text = comboBox1.Text;
}

```

程序运行结果如图 9.22 所示。



图 9.21 只可以选择的下拉框样式

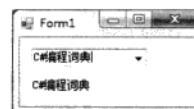


图 9.22 获取控件改变后的值

## 9.4.2 复选框控件

复选框控件（CheckBox 控件）用来表示是否选取了某个选项条件，它常用于为用户提供具有是/否或真/假值的选项。如图 9.23 所示为 CheckBox 控件。

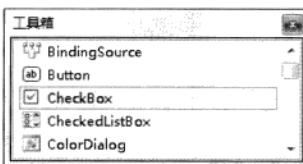


图 9.23 CheckBox 控件

下面详细介绍复选框控件（CheckBox 控件）的常见用法。

### 1. 判断复选框是否被选中

通过在控件的 Click(单击)事件中判断控件的 CheckState 属性，可以判断复选框是否被选中。CheckState 属性的返回值是 Checked 或 Unchecked，返回 Checked 表示控件处于选中状态，而返回 Unchecked 表示控件已经取消选中状态。

**例 9.14** 创建一个 Windows 应用程序，在 CheckBox 控件的 Click 事件中判断复选框是否被选中，并弹出相应的信息提示，代码如下。（实例位置：光盘\mr\09\sl\9.14）

```
private void checkBox1_Click(object sender, EventArgs e)
{
    if (checkBox1.CheckState == CheckState.Checked) //使用 if 语句判断控件是否被选中
    {
        MessageBox.Show("CheckBox 控件被选中"); //如果被选中，则弹出相应提示
    }
    else
    {
        MessageBox.Show("CheckBox 控件选择被取消"); //提示该控件的选择被取消
    }
}
```

程序运行结果如图 9.24 和图 9.25 所示。

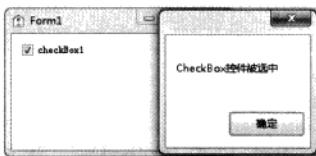


图 9.24 控件被选中

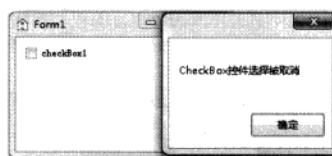


图 9.25 控件取消选择

### 2. 响应复选框的选中状态更改时事件

当控件的选择状态发生改变时，将会引发控件的 CheckStateChanged 事件。

**例 9.15** 创建一个 Windows 应用程序，在 CheckBox 控件的 CheckStateChanged 事件中编写代码，实现当复选框的选择状态发生改变时弹出提示框，代码如下。（实例位置：光盘\mr\09\sl\9.15）

```
private void checkBox1_CheckStateChanged(object sender, EventArgs e)
{
    MessageBox.Show("控件的选择状态发生改变"); //当控件的选择状态发生改变时，弹出提示框
}
```

程序运行结果如图 9.26 所示。

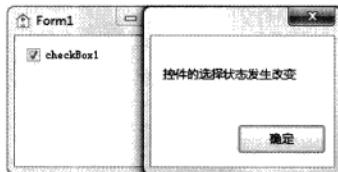


图 9.26 选中状态更改引发 CheckStateChanged 事件

### 9.4.3 单选按钮控件

单选按钮控件（RadioButton 控件）为用户提供由两个或多个互斥选项组成的选项集，当用户选中某个单选按钮时，同一组中的其他单选按钮不能同时被选中。如图 9.27 所示为 RadioButton 控件。

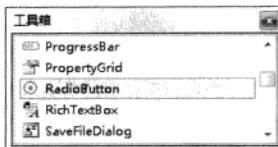


图 9.27 RadioButton 控件

下面详细介绍单选按钮控件（RadioButton 控件）的常见用法。

#### 1. 判断单选按钮是否被选中

判断单选按钮是否被选中时，可以使用 RadioButton 控件的 Checked 属性，Checked 属性的返回值如果为 True，则控件被选中；否则，控件选中状态被取消。

**例 9.16** 创建一个 Windows 应用程序，在窗体中添加两个 RadioButton 控件，并分别在两个控件的 Click 事件中通过 if 语句判断控件的 Checked 属性的返回值是否为 True，代码如下。（实例位置：光盘\mr\09\s\l\9.16）

```
private void Form1_Load(object sender, EventArgs e)
{
    radioButton1.Checked = false;           //设置单选按钮的 Checked 属性为 false
    radioButton2.Checked = false;           //设置单选按钮的 Checked 属性为 false
}
private void radioButton2_Click(object sender, EventArgs e)
{
    if (radioButton2.Checked == true)      //通过 if 语句判断控件的 Checked 属性的返回值是否为 True
    {
        MessageBox.Show("RadioButton2 控件被选中");
    }
}
private void radioButton1_Click(object sender, EventArgs e)
{
    if (radioButton1.Checked == true)      //通过 if 语句判断控件的 Checked 属性的返回值是否为 True
    {
        MessageBox.Show("RadioButton1 控件被选中");
    }
}
```

程序的运行结果如图 9.28 和图 9.29 所示。

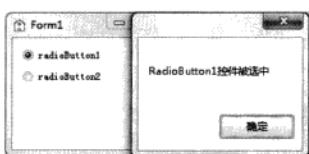


图 9.28 RadioButton1 控件被选中

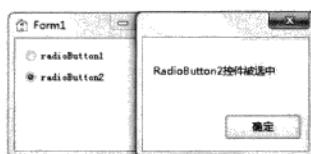


图 9.29 RadioButton2 控件被选中

## 2. 响应单选按钮选中状态更改时事件

当 RadioButton 控件的选中状态发生更改时，会引发控件的 CheckedChanged 事件。

**例 9.17** 创建一个 Windows 应用程序，在窗体中添加一个 RadioButton 控件和两个 Button 控件，单击 button1 按钮，选中 RadioButton 控件；单击 button2 按钮，取消 RadioButton 控件的选中状态，代码如下。

(实例位置：光盘\mr\09\sl\9.17)

```
private void radioButton1_CheckedChanged(object sender, EventArgs e)
{
    MessageBox.Show("RadioButton1 控件的选中状态被更改"); //实现当控件的选择状态改变时弹出提示
}
private void button1_Click(object sender, EventArgs e)
{
    radioButton1.Checked = true; //选中单选按钮
}
private void button2_Click(object sender, EventArgs e)
{
    radioButton1.Checked = false; //取消单选按钮的选中状态
}
private void Form1_Load(object sender, EventArgs e)
{
    radioButton1.Checked = false; //设置单选按钮的 Checked 属性为 false
}
```

程序的运行结果如图 9.30 和图 9.31 所示。

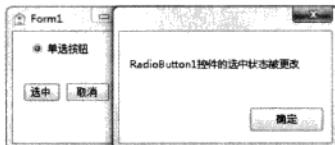


图 9.30 选中 RadioButton 控件

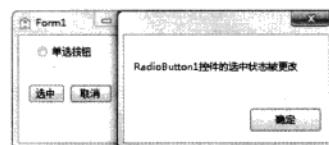


图 9.31 取消选中 RadioButton 控件

### 9.4.4 数值选择控件

数值选择控件（NumericUpDown 控件）是一个显示和输入数值的控件，该控件提供一对上下箭头，用户可以单击上下箭头选择数值，也可以直接输入。数值选择控件的 Maximum 属性可以设置数值的最大值，如果输入的数值大于这个属性的值，则自动把数值修改为设置的最大值；而其 Minimum 属性可以设置数值的最小值，如果输入的数值小于这个属性的值，则自动把数值修改为设置的最小值。如图 9.32 所示为 NumericUpDown 控件。

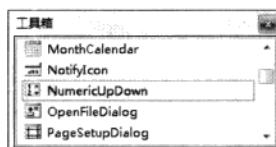


图 9.32 NumericUpDown 控件

下面详细介绍数值选择控件（NumericUpDown 控件）的常见用法。

### 1. 获取 NumericUpDown 控件中显示的数值

通过 Value 属性可以获取或设置 NumericUpDown 控件中显示的数值。

语法: public decimal Value { get; set; }

说明: 该属性值表示 NumericUpDown 控件的数值。

**例 9.18** 创建一个 Windows 应用程序, 向窗体中添加一个 NumericUpDown 控件和一个 Label 控件。在窗体的 Load 事件中, 首先设置 NumericUpDown 控件的 Maximum 属性为 20, Minimum 属性为 1, 当 NumericUpDown 控件的值发生改变时, 通过 Label 控件显示更改后的 NumericUpDown 控件中的数值, 代码如下。(实例位置: 光盘\mr\09\sl\9.18)

```
private void Form1_Load(object sender, EventArgs e)
{
    numericUpDown1.Maximum = 20; //设置控件的最大值为 20
    numericUpDown1.Minimum = 1; //设置控件的最小值为 1
}
private void numericUpDown1_ValueChanged(object sender, EventArgs e)
{
    label1.Text = "当前控件中显示的数值: " + numericUpDown1.Value; //实现当控件的值改变时, 显示当前的值
}
```

程序运行结果如图 9.33 所示。

### 2. 设置 NumericUpDown 控件中数值的显示方式

设置 NumericUpDown 控件中数值的显示方式时可以使用 DecimalPlaces 属性、ThousandsSeparator 属性和 Hexadecimal 属性, 其中 DecimalPlaces 属性用于确定在小数点后显示几位数, 默认为 0; ThousandsSeparator 属性用于确定是否每隔 3 个十进制数位就插入一个分隔符, 默认为 false; Hexadecimal 属性用于确定是否以十六进制显示数字, 默认为 false。

**例 9.19** 创建一个 Windows 应用程序, 通过设置 NumericUpDown 控件的 DecimalPlaces 属性为 2, 使控件中数值的小数点后显示两位小数, 代码如下。(实例位置: 光盘\mr\09\sl\9.19)

```
private void Form1_Load(object sender, EventArgs e)
{
    numericUpDown1.Maximum = 20; //设置控件的最大值为 20
    numericUpDown1.Minimum = 1; //设置控件的最小值为 1
    numericUpDown1.DecimalPlaces = 2; //设置控件的 DecimalPlaces 属性, 使控件中数值的小数点后显示两位小数
}
```

程序运行结果如图 9.34 所示。

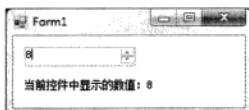


图 9.33 获取控件中显示的数值



图 9.34 将 DecimalPlaces 属性设置为 2

### 9.4.5 列表控件

列表控件(ListBox 控件)用于显示一个列表, 用户可以从中选择一项或多项, 如果选项总数超出可以显示的项数, 则控件会自动添加滚动条。如图 9.35 所示为 ListBox 控件。

下面详细介绍 ListBox 控件的几种常见用法。

## 1. 在 ListBox 控件中添加和移除项目

通过 ListBox 控件的 Items 属性的 Add 方法，可以向 ListBox 控件中添加项目；通过 ListBox 控件的 Items 属性的 Remove 方法，可以将 ListBox 控件中选中的项目移除。

**例 9.20** 创建一个 Windows 应用程序，通过 ListBox 控件的 Items 属性的 Add 方法和 Remove 方法，实现向列表中添加项目以及移除选中项目，代码如下。（实例位置：光盘\mr\09\sl\9.20）

```
private void button1_Click(object sender, EventArgs e)
{
    if (textBox1.Text == "") //使用 if 语句判断文本框中是否输入数据
    {
        MessageBox.Show("请输入要添加的数据"); //弹出提示
    }
    else //否则
    {
        listBox1.Items.Add(textBox1.Text); //使用 Add 方法向控件中添加数据
        textBox1.Text = ""; //清空文本框
    }
}
private void button2_Click(object sender, EventArgs e)
{
    if (listBox1.SelectedItems.Count == 0) //判断是否选择项目
    {
        MessageBox.Show("请选择要删除的项目"); //如果没有选择项目，则弹出提示
    }
    else //否则
    {
        listBox1.Items.Remove(listBox1.SelectedItem); //使用 Remove 方法移除选中项目
    }
}
```

程序运行结果如图 9.36 所示。

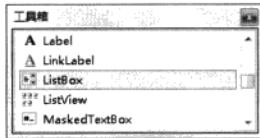


图 9.35 ListBox 控件

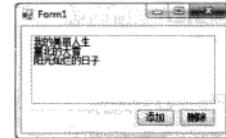


图 9.36 向 ListBox 控件中添加和移除项目

## 2. 在 ListBox 控件中选择多项

实现在 ListBox 控件中选择多项时需要设置其 SelectionMode 属性，SelectionMode 属性的属性值是 SelectionMode 枚举值之一，默认为 SelectionMode.One。SelectionMode 枚举成员及说明如表 9.3 所示。

表 9.3 SelectionMode 枚举成员及说明

枚举成员	说明
MultiExtended	可以选择多项，并且用户可使用 Shift 键、Ctrl 键和上下左右箭头键来进行选择
MultiSimple	可以选择多项
None	无法选择项
One	只能选择一项

**例 9.21** 创建一个 Windows 应用程序, 通过将 ListBox 控件的 SelectionMode 属性值设置为 SelectionMode 枚举成员 MultiExtended, 实现在列表中可以选择多项, 并且用户可以使用键盘上的 Shift 键、Ctrl 键和上下左右箭头键来进行选择, 代码如下。 (实例位置: 光盘\mr\09\sl\9.21)

```

private void Form1_Load(object sender, EventArgs e)
{
    listBox1.SelectionMode = SelectionMode.MultiExtended;           //实现在控件中可以选择多项
}
private void button2_Click(object sender, EventArgs e)
{
    if (textBox1.Text == "")
        MessageBox.Show("添加项目不能为空");                         //判断文本框中是否输入数据
    else
    {
        listBox1.Items.Add(textBox1.Text);                            //使用 Add 方法向控件中添加数据
        textBox1.Text = "";                                         //清空文本框
    }
}
private void button1_Click(object sender, EventArgs e)
{
    label1.Text = "共选择了: " + listBox1.SelectedItems.Count.ToString() + "项"; //显示选择项目的数量
}

```

程序运行结果如图 9.37 所示。

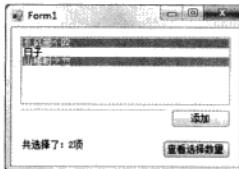


图 9.37 在 ListBox 控件中选择多项

## 9.5 分组控件

分组控件主要包括容器控件 (Panel 控件)、分组框控件 (GroupBox 控件) 和选项卡控件 (TabControl 控件) 等, 本节将对分组控件及其使用进行详细讲解。

### 9.5.1 容器控件

容器控件 (Panel 控件) 用于为其他控件提供可识别的分组, 它可以使窗体的分类更详细, 便于用户理解。如图 9.38 所示为 Panel 控件。

**技巧:** 在 Panel 控件中可以设置滚动条。

**例 9.22** 创建一个 Windows 应用程序, 在 TextBox 文本框中输入查找的软件名称, 如果输入的是 “C#

编程词典”，则调用 Show 方法显示 Panel 控件。在 Panel 控件中有一个 RichTextBox 控件，它用于显示“C# 编程词典”的相关信息，代码如下。（实例位置：光盘\mr\09\s\9.22）

```

private void Form1_Load(object sender, EventArgs e)
{
    panel1.Visible = false; //隐藏 Panel 控件
    //设置 RichTextBox 控件的 Text 属性
    richTextBox1.Text = "名称：C#编程词典\n类别：软件\n版权：明日科技\n版本：2009";
}
private void button1_Click(object sender, EventArgs e)
{
    if (textBox1.Text == "") //判断文本框中是否输入数据
    {
        panel1.Visible = false;
        MessageBox.Show("请输入名称"); //如果没有输入数据，则弹出提示
        textBox1.Focus(); //使光标焦点处于文本框中
    }
    else
    {
        if (textBox1.Text.Trim() == "C#编程词典") //判断文本框中是否输入“C#编程词典”
        {
            panel1.Show(); //如果输入“C#编程词典”，则显示 Panel 控件
        }
        else
        {
            panel1.Visible = false;
            MessageBox.Show("查无此软件"); //弹出提示
            textBox1.Text = ""; //清空文本框
        }
    }
}

```

程序运行结果如图 9.39 所示。

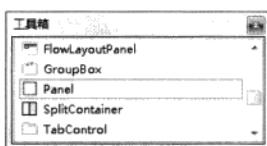


图 9.38 Panel 控件

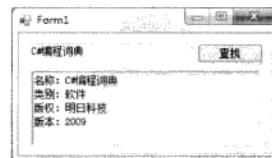


图 9.39 显示 Panel 控件

## 9.5.2 分组框控件

分组框控件（GroupBox 控件）主要为其他控件提供分组，并且按照控件的分组来细分窗体的功能，其在所包含的控件集周围总是显示边框，而且可以显示标题，但是该控件没有滚动条。如图 9.40 所示为GroupBox 控件。

开发人员可以通过GroupBox 控件的 Text 属性来设置要显示的标题。

**例 9.23** 创建一个 Windows 应用程序，通过设置GroupBox 控件的 Text 属性，使GroupBox 控件的标

题为“诗词”，代码如下。（实例位置：光盘\mr\09\sl\9.23）

```
private void Form1_Load(object sender, EventArgs e)
{
    groupBox1.Text = "诗词"; //设置控件的 Text 属性，使其显示“诗词”
}
```

程序运行结果如图 9.41 所示。

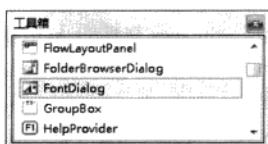


图 9.40 GroupBox 控件

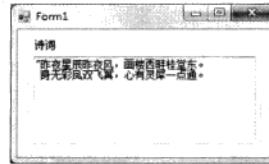


图 9.41 设置控件的标题

### 9.5.3 选项卡控件

在选项卡控件（TabControl 控件）中可以添加多个选项卡，同时也可以在选项卡中添加子控件，这样就可以把窗体设计成多页，并且使窗体的功能划分为多个部分。选项卡控件的选项卡中可以包含图片或其他控件。如图 9.42 所示为 TabControl 控件。

下面详细介绍 TabControl 控件的一些常见用法。

#### 1. 改变选项卡的显示样式

通过使用 TabControl 控件和组成控件上各选项卡的TabPage 对象的属性，可以更改 Windows 选项卡的外观。通过设置这些属性，可以实现在选项卡上显示图像、以垂直方式或水平方式显示选项卡、显示多行选项卡以及启用或禁用选项卡等功能。

**例 9.24** 创建一个 Windows 应用程序，向窗体中添加一个 ImageList 控件，然后将图像添加到 ImageList 控件的图像列表中；添加一个 TabControl 控件，将其 ImageList 属性设置为 ImageList 控件，并且将 tabPage 选项卡的 ImageIndex 属性设置为列表中相应图像的索引。代码如下。（实例位置：光盘\mr\09\sl\9.24）

```
private void Form1_Load(object sender, EventArgs e)
{
    tabControl1.ImageList = imageList1; //设置控件的 ImageList 属性为 imageList1
    tabPage1.ImageIndex = 0; //第一个选项卡的图标是 imageList1 中索引为 0 的图标
    tabPage1.Text = "选项卡 1"; //设置控件第一个选项卡的 Text 属性
    tabPage2.ImageIndex = 0; //第二个选项卡的图标是 imageList1 中索引为 0 的图标
    tabPage2.Text = "选项卡 2"; //设置控件第二个选项卡的 Text 属性
}
```

程序运行结果如图 9.43 所示。

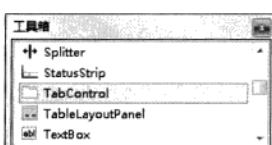


图 9.42 TabControl 控件

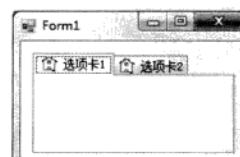


图 9.43 标签部位显示图标

## 2. 添加和移除选项卡

### 以编程方式添加选项卡

默认情况下 TabControl 控件包含两个TabPage 选项卡，开发人员可以使用 TabPages 属性的 Add 方法添加新的选项卡，该方法主要用于将TabPage 添加到集合。

语法：public void AddTabPage value)

说明：参数 value 表示要添加的TabPage。

**例 9.25** 创建一个 Windows 应用程序，使用TabControl 控件的 TabPages 属性的 Add 方法，向控件中添加新的选项卡，代码如下。（实例位置：光盘\mr\09\sl\9.25）

```
private void Form1_Load(object sender, EventArgs e)
{
    tabControl1.ImageList = imageList1; //设置控件的 ImageList 属性为 imageList1
    tabPage1.ImageIndex = 0; //第一个选项卡的图标是 imageList1 中索引为 0 的图标
    tabPage2.ImageIndex = 0; //第二个选项卡的图标是 imageList1 中索引为 0 的图标
    string Title = "新增选项卡 " + (tabControl1.TabCount + 1).ToString(); //声明一个字符串变量，用于生成新增选项卡的名称
   TabPage MyTabPage = newTabPage(Title); //创建TabPage
    tabControl1.TabPages.Add(MyTabPage); //使用 TabPages 属性的 Add 方法添加新的选项卡
}
```

程序运行结果如图 9.44 所示。

### 以编程方式移除选项卡

如果要移除 TabControl 控件中的某个选项卡，可以使用 TabPages 属性的 Remove 方法，该方法主要用于从集合中移除TabPage。

语法：public void RemoveTabPage value)

说明：参数 value 表示要移除的TabPage。

**例 9.26** 创建一个 Windows 应用程序，通过使用 TabPages 属性的 Remove 方法，删除 TabControl 控件中指定的选项卡，代码如下。（实例位置：光盘\mr\09\sl\9.26）

```
private void button1_Click(object sender, EventArgs e)
{
    string Title = "新增选项卡 " + (tabControl1.TabCount + 1).ToString(); //声明一个字符串变量，用于生成新增选项卡的名称
   TabPage MyTabPage = newTabPage(Title); //创建TabPage
    tabControl1.TabPages.Add(MyTabPage); //使用 TabPages 属性的 Add 方法添加新的选项卡
}
private void button2_Click(object sender, EventArgs e)
{
    if (tabControl1.SelectedIndex == 0) //判断是否选择了要删除的选项卡
    {
        MessageBox.Show("请选择要删除的选项卡"); //如果没有选择，则弹出提示
    }
    else
    {
        tabControl1.TabPages.Remove(tabControl1.SelectedTab); //使用 TabPages 属性的 Remove 方法删除指定的选项卡
    }
}
```

程序运行结果如图 9.45 所示。

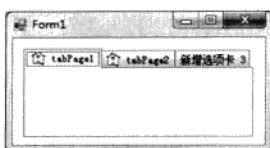


图 9.44 添加选项卡

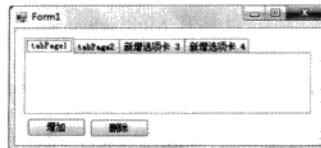


图 9.45 删除指定的选项卡

如果要删除 TabControl 控件中所有的选项卡，可以使用 TabPages 属性的 Clear 方法。

语法：public virtual void Clear()

说明：该方法主要用于从集合中移除所有的选项卡。

**例 9.27** 删除 tabControl1 控件中所有的选项卡，代码如下。

```
tabControl1.TabPages.Clear(); //使用 Clear 方法删除所有的选项卡
```

## 9.6 对话框控件

对话框控件主要包括打开对话框（OpenFileDialog 控件）、另存为对话框（SaveFileDialog 控件）、浏览文件夹对话框（FolderBrowserDialog 控件）、颜色对话框（ColorDialog 控件）和字体对话框（FontDialog 控件）等。C#中的 Windows 对话框控件如图 9.46 所示。

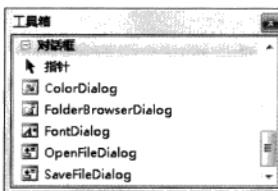


图 9.46 C#中的 Windows 对话框控件

### 9.6.1 对话框概述

如果一个窗体的弹出是为了对诸如打开文件之类的用户请求做出响应，并且同时停止了所有其他“用户与应用程序之间”的交互活动，那么它就是一个对话框。比较常用的对话框操作（如打开文件、选择字体和保存文件等）都是通过 Windows 提供的标准对话框实现的，C#也可以利用这些对话框来实现相应的功能。

另外，在 C#程序中还提供了两种对话框，分别为消息对话框和关于对话框。消息对话框主要用来向用户显示信息和消息，MessageBox 是一个预定义对话框，它主要用于向用户显示与应用程序相关的信息及请求来自用户的信息；而关于对话框则常用于显示软件产品的介绍信息。

### 9.6.2 打开对话框

OpenFileDialog 控件表示一个通用对话框，用户可以使用此对话框来指定一个或多个要打开的文件的文件名。如图 9.47 所示为 OpenFileDialog 控件。

**例 9.28** 创建一个 Windows 应用程序，主要用来实现在文本框中显示选择的多个文件。在该程序中添加一个 RichTextBox 控件，用来显示选择的文件名；添加一个 Button 控件，用来选择文件；添加一个

OpenFileDialog 控件，并将其 Multiselect 属性设置为 True，以便可以在打开的“打开”对话框中选择多个文件。代码如下。（实例位置：光盘\mr\09\sl\9.28）

```
private void button1_Click(object sender, EventArgs e)
{
    if(openFileDialog1.ShowDialog() == DialogResult.OK) //判断是否选择了文件
    {
        for (int i = 0; i < openFileDialog1.FileNames.Length; i++) //循环读取选择的多个文件
        {
            //将选择的文件显示在文本框中
            richTextBox1.Text += openFileDialog1.FileNames[i].ToString() + "\n";
        }
    }
}
```

程序运行结果如图 9.48 所示。



图 9.47 OpenFileDialog 控件

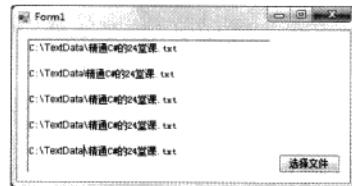


图 9.48 显示选择的多个文件

### 9.6.3 另存为对话框

SaveFileDialog 控件表示一个通用对话框，用户可以使用此对话框来指定一个要将文件另存为的文件名。如图 9.49 所示为 SaveFileDialog 控件。

**例 9.29** 创建一个 Windows 应用程序，主要用来实现通过另存为对话框保存文本文件功能。该程序中添加一个 RichTextBox 控件，用来显示选择的文本文件内容和编辑文本文件内容；添加两个 Button 控件，分别用来打开文件和另存文件功能；添加一个 OpenFileDialog 控件，用来显示“打开”对话框；添加一个 SaveFileDialog 控件，用来显示“另存为”对话框，以便选择文件的存放路径和输入另存为文件名。代码如下。（实例位置：光盘\mr\09\sl\9.29）

```
private void button1_Click(object sender, EventArgs e)
{
    openFileDialog1.Filter = "*.txt(文本文件)*.txt"; //设置打开文件格式
    if (openFileDialog1.ShowDialog() == DialogResult.OK) //判断是否选择文件
    {
        //创建读取数据流
        StreamReader SReader = new StreamReader(openFileDialog1.FileName, Encoding.Default);
        richTextBox1.Text = SReader.ReadToEnd(); //显示打开文件的内容
        SReader.Close(); //关闭读取数据流
    }
}

private void button2_Click(object sender, EventArgs e)
{
    saveFileDialog1.Filter = "*.txt(文本文件)*.txt"; //设置保存文件格式
    if (saveFileDialog1.ShowDialog() == DialogResult.OK) //判断是否输入了另存为文件名

```

```

    {
        //创建写入数据流
        StreamWriter SWriter = new StreamWriter(saveFileDialog1.FileName,true);
        SWriter.Write(richTextBox1.Text);           //向文件中写入数据
        SWriter.Close();                         //关闭写入数据流
    }
}

```

**说明：**由于本实例中用到读取数据流和写入数据流，所以首先需要添加 System.IO 命名空间。

程序运行结果如图 9.50 所示。

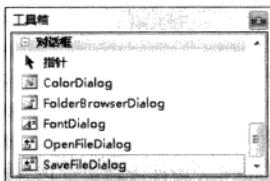


图 9.49 SaveFileDialog 控件

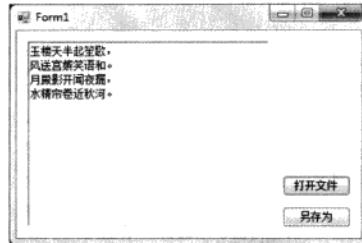


图 9.50 另存为文本文件

#### 9.6.4 浏览文件夹对话框

浏览文件夹对话框（FolderBrowserDialog 控件）主要用来提示用户选择文件夹。如图 9.51 所示为 FolderBrowserDialog 控件。

**例 9.30** 创建一个 Windows 应用程序，主要用来实现在文本框中显示选择的文件夹路径。在该程序中添加一个 RichTextBox 控件，用来显示选择的文件夹路径；添加一个 Button 控件，用来选择文件夹路径；添加一个 FolderBrowserDialog 控件，用来显示“浏览文件夹”对话框。代码如下。（实例位置：光盘\mr\09\sl\9.30）

```

private void button1_Click(object sender, EventArgs e)
{
    folderBrowserDialog1.RootFolder = Environment.SpecialFolder.Desktop; //设置浏览对话框的初始路径为桌面
    if(folderBrowserDialog1.ShowDialog() == DialogResult.OK)           //判断是否选择了文件
    {
        richTextBox1.Text += folderBrowserDialog1.SelectedPath;       //将选择的文件显示在文本框中
    }
}

```

程序运行结果如图 9.52 所示。

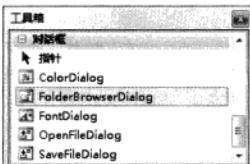


图 9.51 FolderBrowserDialog 控件

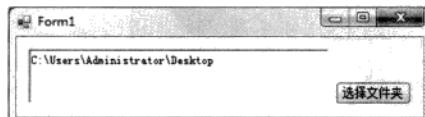


图 9.52 显示选择的文件夹路径

## 9.7 菜单、工具栏和状态栏控件

菜单是窗体应用程序主要的用户界面要素；工具栏为应用程序提供了操作系统各功能的快捷方式；状态栏显示系统的一些状态信息。C#中的菜单、工具栏和状态栏控件如图 9.53 所示。

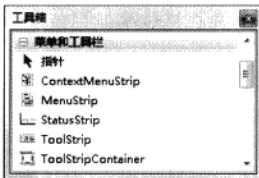


图 9.53 C# 中的菜单、工具栏和状态栏控件

### 9.7.1 菜单控件

菜单控件（MenuStrip 控件）用来设计程序的菜单栏，C#中的 ToolStrip 控件支持多文档界面、菜单合并、工具提示和溢出等功能，开发人员可以通过添加访问键、快捷键、选中标记、图像和分隔条来增强菜单的可用性和可读性。如图 9.54 所示为 ToolStrip 控件。



图 9.54 ToolStrip 控件

为了使读者更加直观地了解 ToolStrip 控件的用法，下面通过一个实例介绍如何设计菜单栏。

**例 9.31** 创建一个 Windows 应用程序，演示如何通过 ToolStrip 控件创建一个类似 Word 的“文件”菜单，具体步骤如下。（**实例位置：光盘\mr\09\sl\9.31**）

- (1) 创建一个 Windows 应用程序，从工具箱中将 ToolStrip 控件拖曳到窗体中，如图 9.55 所示。
- (2) 在输入菜单名称时，系统会自动产生输入下一个菜单名称的提示，如图 9.56 所示。



图 9.55 将 ToolStrip 控件拖曳到窗体中



图 9.56 输入菜单名称

(3) 在图 9.56 所示的输入框中输入“文件(&F)”后，菜单中会自动显示“文件(F)”，在此处，“&”被识别为确认快捷键的字符，例如，“文件(F)”菜单就可以通过键盘上的 Alt+F 键打开。同样，在“文件(F)”菜单下创建“新建(N)”、“打开(O)”、“关闭(C)”和“保存(S)”子菜单，如图 9.57 所示。

(4) 添加完毕后的最终效果如图 9.58 所示。



图 9.57 添加菜单内容

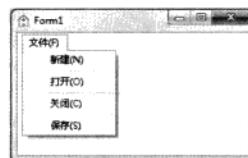


图 9.58 菜单示意图

### 9.7.2 工具栏控件

使用工具栏控件（ToolStrip 控件）可以创建具有 Windows XP、Office、Internet Explorer 或自定义外观和行为的工具栏及其他用户界面元素，这些元素支持溢出及运行时项重新排序。如图 9.59 所示为 ToolStrip 控件。

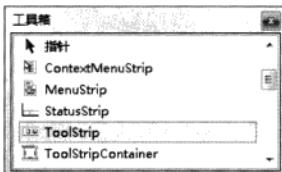


图 9.59 ToolStrip 控件

创建工具栏的过程比较简单，下面通过实例演示如何创建工具栏。

**例 9.32** 创建一个 Windows 应用程序，演示如何通过 ToolStrip 控件创建一个工具栏，具体步骤如下。  
(实例位置：光盘\mr\09\sl\9.32)

(1) 创建一个 Windows 应用程序，从工具箱中将 ToolStrip 控件拖曳到窗体中，如图 9.60 所示。

(2) 单击工具栏右侧下拉按钮的提示图标，如图 9.61 所示。从该图中可以看到，当单击工具栏右侧下拉按钮时，显示出 8 种不同类型的选项，下面分别进行介绍。

- Button**: 包含文本和图像中可让用户选择的项。
- Label**: 包含文本和图像的项，不可以让用户选择，可以显示超链接。
- SplitButton**: 在 Button 的基础上增加了一个下拉菜单。
- DropDownButton**: 用于下拉菜单选择项。

- Separator: 分隔符。
- ComboBox: 显示一个 ComboBox 的项。
- TextBox: 显示一个 TextBox 的项。
- ProgressBar: 显示一个 ProgressBar 的项。

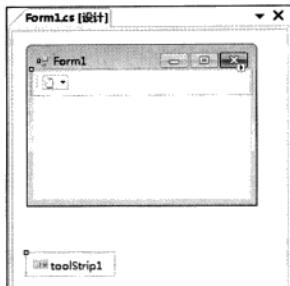


图 9.60 将 ToolStrip 控件拖曳到窗体中

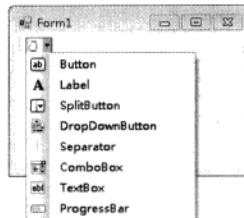


图 9.61 添加工具栏项目

(3) 添加相应的工具栏按钮后，可以设置其要显示的图像，如图 9.62 所示。

(4) 程序运行结果如图 9.63 所示。



图 9.62 设置按钮图像

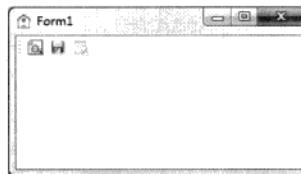


图 9.63 程序运行结果

### 9.7.3 状态栏控件

状态栏通常处于窗体的最底部，用于显示窗体上一些对象的相关信息，或者显示应用程序的信息。C# 中使用 StatusStrip 控件来设计状态栏，通常该控件由 ToolStripStatusLabel 对象组成，每个这样的对象都可以显示文本、图像或同时显示这两者，另外，该控件还可以包含 ToolStripDropDownButton、ToolStripSplitButton 和 ToolStripProgressBar 等控件。如图 9.64 所示为 StatusStrip 控件。

下面通过一个实例演示如何使用 StatusStrip 控件。

**例 9.33** 创建一个 Windows 应用程序，使用 StatusStrip 控件制作状态栏，当窗体加载时，在状态栏中显示当前日期；当单击“加载”按钮时，在状态栏中加载进度条，代码如下。（实例位置：光盘\mr\09\sl\9.33）

```
private void Form1_Load(object sender, EventArgs e)
```

```

        this.toolStripStatusLabel2.Text = DateTime.Now.ToString(); //在任务栏上显示系统的当前日期
    }
    private void button1_Click(object sender, EventArgs e)
    {
        this.toolStripProgressBar1.Value = 0; //设置进度条的初始值
        this.toolStripProgressBar1.Minimum = 0; //进度条的最小值
        this.toolStripProgressBar1.Maximum = 5000; //进度条的最大值
        this.toolStripProgressBar1.Step = 2; //进度条的增值
        for (int i = 0; i <= 4999; i++) //使用 for 循环读取数据
        {
            this.toolStripProgressBar1.PerformStep(); //按照 Step 属性的数量增加进度条的当前位置
        }
    }
}

```

程序运行结果如图 9.65 所示。

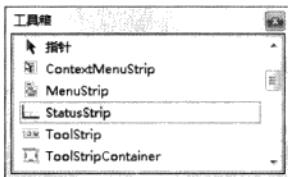


图 9.64 StatusStrip 控件



图 9.65 状态栏的应用

## 9.8 照猫画虎——基本功训练

### 9.8.1 基本功训练 1——在 ComboBox 下拉列表中显示图片

视频讲解：光盘\mr\09\lx\在 ComboBox 下拉列表中显示图片.exe

实例位置：光盘\mr\09\zmhh\01

ComboBox 控件可以方便地显示多条数据信息，本实例实现在 ComboBox 控件中显示图片信息。新建一个 Windows 窗体应用程序，在窗体中添加一个 ComboBox 控件和一个 Button 按钮，ComboBox 控件用于演示带有图片信息的下拉列表；Button 按钮用于向下拉列表中添加数据。程序主要代码如下。

```

private void cbox_DisplayPictures_DrawItem(object sender, DrawItemEventArgs e)
//当绘制特定项时触发 DrawItem 事件
{
    if (G_ImageList != null) //判断 ImageList 是否为空
    {
        Graphics g = e.Graphics; //得到绘图对象
        Rectangle r = e.Bounds; //得到绘图范围
        Size imageSize = G_ImageList.ImageSize; //获取图像大小
        if (e.Index >= 0) //判断是否有绘制项
        {
            Font fn = new Font("宋体", 10, FontStyle.Bold); //创建字体对象
            string s = cbox_DisplayPictures.Items[e.Index].ToString(); //得到绘制项的字符串
            DrawItemState dis = e.State;
            if (e.State == (DrawItemState.NoAccelerator | DrawItemState.NoFocusRect)) //判断是否是选中项
            {
                g.FillRectangle(Brushes.Blue, r); //填充矩形
                g.DrawString(s, fn, Brushes.White, r); //显示文本
            }
            else
            {
                g.FillRectangle(Brushes.White, r); //填充矩形
                g.DrawImage(G_ImageList.Images[e.Index], r); //显示图像
                g.DrawString(s, fn, Brushes.Black, r); //显示文本
            }
        }
    }
}

```

```

    {
        e.Graphics.FillRectangle(new SolidBrush(Color.LightYellow), r); //画条目背景
        G_ImageList.Draw(e.Graphics, r.Left, r.Top, e.Index); //绘制图像
        e.Graphics.DrawString(s, fn, new SolidBrush(Color.Black), //显示字符串
            r.Left + imageSize.Width, r.Top);
        e.DrawFocusRectangle(); //显示取得焦点时的虚线框
    }
    else
    {
        e.Graphics.FillRectangle(new SolidBrush(Color.LightGreen), r); //画条目背景
        G_ImageList.Draw(e.Graphics, r.Left, r.Top, e.Index); //绘制图像
        e.Graphics.DrawString(s, fn, new SolidBrush(Color.Black), //显示字符串
            r.Left + imageSize.Width, r.Top);
        e.DrawFocusRectangle(); //显示取得焦点时的虚线框
    }
}
}

```

实例运行效果如图 9.66 所示。

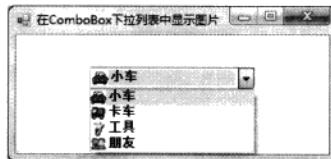


图 9.66 在 ComboBox 下拉列表中显示图片

**照猫画虎：**在上面的实例中，图片和文本显示到 ComboBox 控件的左侧，尝试修改上面的实例，让图片和文本显示到 ComboBox 控件的中间位置。(20 分)(实例位置：光盘\mr\09\zmhh\01\_zmhh)

### 9.8.2 基本功训练 2——实现带查询功能的 ComboBox 控件

 视频讲解：光盘\mr\09\lx\实现带查询功能的 ComboBox 控件.exe

 实例位置：光盘\mr\09\zmhh\02

通过设置 ComboBox 控件的 AutoCompleteSource 属性和 AutoCompleteMode 属性，可以实现从 ComboBox 控件中查询已存在的项，并自动完成控件内容的输入。新建一个 Windows 窗体应用程序，在窗体中添加一个 ComboBox 下拉列表控件和一个 Button 按钮：ComboBox 控件用于演示查询功能；Button 按钮用于设置 ComboBox 控件的查询功能。程序代码如下。

```
private void Frm_Main_Load(object sender, EventArgs e) //向 ComboBox 中添加元素的代码如下
{
    cbox_Find.Items.Clear(); //清空 ComboBox 集合
    cbox_Find.Items.Add("C#编程词典"); //向 ComboBox 集合添加元素
    cbox_Find.Items.Add("C#编程宝典"); //向 ComboBox 集合添加元素
    cbox_Find.Items.Add("C#视频学"); //向 ComboBox 集合添加元素
    cbox_Find.Items.Add("C#范例宝典"); //向 ComboBox 集合添加元素
    cbox_Find.Items.Add("C#从入门到精通"); //向 ComboBox 集合添加元素
    cbox_Find.Items.Add("C#范例大全"); //向 ComboBox 集合添加元素
}
```

```

private void btn_Begin_Click(object sender, EventArgs e)          //设置 ComboBox 控件查询功能的代码如下
{
    cbo_Find.AutoCompleteMode =                                     //设置自动完成的模式
        AutoCompleteMode.SuggestAppend;
    cbo_Find.AutoCompleteSource =                                    //设置自动完成字符串的源
        AutoCompleteSource.ListItems;
}

```

实例运行效果如图 9.67 所示。

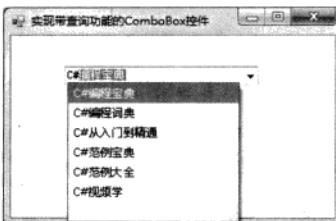


图 9.67 实现带查询功能的 ComboBox 控件

**照猫画虎：**修改上面的实例，实现自动查询任意磁盘下的所有文件夹。提示：可通过设置 ComboBox 控件的 AutoCompleteSource 属性来完成。（20 分）（实例位置：光盘\mr\09\zmhh\02\_zmhh）

### 9.8.3 基本功训练 3——在 RichTextBox 控件中实现关键字描红

■**视频讲解：**光盘\mr\09\lx\在 RichTextBox 控件中实现关键字描红.exe

■**实例位置：**光盘\mr\09\zmhh\03

本实例实现查找 RichTextBox 控件中指定的文本信息，并将找到的文本内容设置为红色。新建一个 Windows 窗体应用程序，在窗体中添加一个 RichTextBox 控件，此控件用于显示文本信息和被描红的文本信息；向窗体中添加一个 TextBox 控件，此控件用于添加查找字符串；向窗体中添加两个 Button 按钮，分别用于查找字符串和打开文本文件。程序代码如下。

```

private int flag = 0;                                         //定义一个 int 型的标识符
private void plotRed_Click(object sender,EventArgs e)
{
    if(flag == richTextBox1.Text.IndexOf(keyWord.Text,flag)) == -1   //当文件中不存在要搜索的关键字时
    {
        //弹出信息提示
        MessageBox.Show("没有要查找的结果","提示信息",MessageBoxButtons.OK,MessageBoxIcon.Asterisk);
        keyWord.Clear();                                            //清空文本框中的内容
        flag = 0;                                                   //重新为 flag 赋值
    }
    else
    {
        richTextBox1.Select(flag,keyWord.Text.Length);           //在 RichTextBox 控件中搜索关键字
        flag = flag + keyWord.Text.Length;                         //递增标识查询关键字的初始长度
        richTextBox1.SelectionColor = Color.Red;                  //设定关键字为红色
    }
}

```

实例运行效果如图 9.68 所示，关键字“明日”被描红。



图 9.68 在 RichTextBox 控件中实现关键字描红

**说明：**可以使用 SelectionFont 属性方便地设置当前选定文本或插入点的字体。

**熊猫画虎：**修改上面的实例，实现将 RichTextBox 控件中的关键字加粗。提示：可以通过设置 RichTextBox 控件的 SelectionFont 属性来实现。（20 分）（实例位置：光盘\mr\09\zmhh\03\_zmhh）

#### 9.8.4 基本功训练 4——对 ListBox 控件中的数据进行排序

**视频讲解：**光盘\mr\09\lx\对 ListBox 控件中的数据进行排序.exe

**实例位置：**光盘\mr\09\zmhh\04

本实例实现对 ListBox 控件中的数据项进行排序。新建一个 Windows 应用程序，在窗体中添加一个 ListBox 控件，用于显示多个数据项；向窗体中添加一个 Button 按钮，用于实现对 ListBox 控件中的数据项进行排序的操作。程序代码如下。

```
private void button1_Click(object sender, EventArgs e)
{
    listBox1.Sorted = true; //启用排序
}
```

实例运行效果如图 9.69 所示。



图 9.69 对 ListBox 控件中的数据进行排序

**熊猫画虎：**上面的实例实现对 ListBox 控件中的数据进行升序排列，尝试修改该实例，实现对 ListBox 控件中的数据进行降序排列。提示：可使用 String 类型的 CompareTo 方法比较两个字符串对象。（20 分）（实例位置：光盘\mr\09\zmhh\04\_zmhh）

#### 9.8.5 基本功训练 5——具有提示功能的工具栏

**视频讲解：**光盘\mr\09\lx\具有提示功能的工具栏.exe

**实例位置：**光盘\mr\09\zmhh\05

本实例通过设置工具栏的 ToolTipText 属性来实现工具栏具有的提示功能。实例运行效果如图 9.70 所示。

在窗体的左下角显示“工具栏提示”字样的提示信息。

实现过程如下。

(1) 打开 Visual Studio 2008 开发环境，新建一个 Windows 窗体应用程序，并将其命名为 MessageTool。

(2) 更改默认窗体 Form1 的 Name 属性为 Frm\_Main，更改 Text 属性为“具有提示功能的工具栏”，向窗体中添加一个 ToolStrip 控件，此控件用于显示菜单信息；向窗体中添加一个 ToolStrip 控件，此控件用于显示带有提示功能的工具栏。

(3) 为工具栏中的按钮添加提示信息非常简单，只需要设置工具栏按钮的 ToolTipText 属性即可。

**说明：**只有将 ToolStrip 控件的 ShowItemToolTips 设置为 true 时，ToolStrip 控件中 ToolStripButton 对象的 ToolTipText 属性才会生效。

**照猫画虎：**创建一个带有背景的工具栏。提示：可设置工具栏按钮的 BackgroundImage 属性。(20分)  
(实例位置：光盘\mr\09\zmhh\05\_zmhh)

照猫画虎栏目分数统计：

照猫画虎题目	1	2	3	4	5	总分数	
分数							

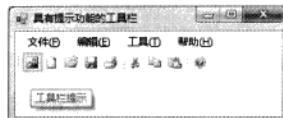


图 9.70 具有提示功能的工具栏

## 9.9 情景应用——拓展与实践

### 9.9.1 情景应用 1——只允许输入数字的 TextBox 控件

**视频讲解：**光盘\mr\09\lx\只允许输入数字的 TextBox 控件.exe

**实例位置：**光盘\mr\09\qjyy\01

我们可以在 TextBox 控件中轻松地输入文本信息，如字母、数字、汉字等。如果需要用户在 TextBox 控件中填写年龄信息，那么年龄信息应只允许为数字，怎样限制用户输入其他信息呢？本实例将会介绍一种方法，只允许用户在 TextBox 控件中输入数字，实例运行效果如图 9.71 所示。



图 9.71 只允许输入数字的 TextBox 控件

实现过程如下。

(1) 打开 Visual Studio 2008 开发环境，新建一个 Windows 窗体应用程序，并将其命名为 OnlyDigit。  
(2) 更改默认窗体 Form1 的 Name 属性为 Frm\_Main，向窗体中添加一个 TextBox 文本框控件，控件用于演示只允许向文本框中输入数字。

(3) 程序主要代码如下。

```
private void txt_Str_KeyPress(object sender, KeyPressEventArgs e)
{
    if (!char.IsDigit(e.KeyChar)) //判断是否为数字
    {
        MessageBox.Show("请输入数字！","提示！");
        MessageBoxButtons.OK,MessageBoxIcon.Information);
    }
}
```

```

        e.Handled = true;
    }
}

```

**说明：**把 KeyPressEventArgs 类型参数 e 的 Handled 属性设置为 true，可以取消用户在 TextBox 控件中的按键操作。

**DIY：**使 TextBox 控件只允许输入字母。提示：可使用 char.IsLetter 方法。（20 分）（实例位置：光盘\mr\09\qjyy\01\_diy）

## 9.9.2 情景应用 2——判断注册用户操作权限

**视频讲解：**光盘\mr\09\lx\判断注册用户操作权限.exe

**实例位置：**光盘\mr\09\qjyy\02

注册用户时，应当分配给用户一些相应的权限，以便于用户操作软件的相应模块。运行本实例，可以根据用户的职责选中相应模块前的复选框，如果取消选中相应模块前的复选框，则意味着取消该用户操作模块的权限。实例运行效果如图 9.72 所示。

实现过程如下。

(1) 打开 Visual Studio 2008 开发环境，新建一个 Windows 窗体应用程序，并将其命名为 Selected。

(2) 更改默认窗体 Form1 的 Name 属性为 Frm\_Main，向窗体中添加 5 个 TextBox 控件，用于输入用户信息；向窗体中添加 4 个 CheckBox 控件，用于选择用户操作权限。

(3) 程序主要代码如下。

```

private void ckShop_CheckedChanged(object sender, EventArgs e)//当 Checked 属性更改时触发 CheckedChanged
{
    if (ckShop.Checked == true)
    {
        ckShop.Visible = true;
        CheckAll(ckShop);
    }
    else
    {
        ckShop.Visible = false;
        CheckAllEsce(ckShop);
    }
}

private void ckSell_CheckedChanged(object sender, EventArgs e)
{
    if (ckSell.Checked == true)
    {
        ckSell.Visible = true;
        CheckAll(ckSell);
    }
    else

```

//判断是否选中进货管理

//显示进货管理信息

//选中所有进货管理

//隐藏进货管理信息

//取消选中所有进货管理

//判断是否选中销售管理

//显示销售管理信息

//选中所有销售管理



图 9.72 判断注册用户操作权限

```

    {
        ckISell.Visible = false;
        CheckAllEsce(ckISell);
    }
}

private void ckMange_CheckedChanged(object sender, EventArgs e)
{
    if (ckMange.Checked == true) //判断是否选中库存管理
    {
        ckIMange.Visible = true; //显示库存管理
        CheckAll(ckIMange); //选中所有库存管理
    }
    else
    {
        ckIMange.Visible = false; //隐藏库存管理
        CheckAllEsce(ckIMange); //取消选中所有库存管理
    }
}

```

**技巧：**通过设置 `CheckedListBox` 控件的 `Visible` 属性，可以方便地显示和隐藏 `CheckedListBox` 控件。`Visible` 属性为布尔值，当属性为 `true` 时，显示控件；当属性为 `false` 时，隐藏控件。

**DIY：**使用 `TreeView` 控件显示用户，使用 `CheckedListBox` 控件为用户授权。提示：用 `TreeView` 控件的节点文本显示用户名，关于 `TreeView` 控件可参见第 10 堂课中的相关内容。(20 分)(实例位置：光盘\mr\09\qjyy\02\_diy)

### 9.9.3 情景应用 3——实现类似 Word 的项目编号功能

**视频讲解：**光盘\mr\09\lx\实现类似 Word 的项目编号功能.exe

**实例位置：**光盘\mr\09\qjyy\03

在对大量数据进行统计时，Word 为用户提供了许多功能，如项目编号和项目符号等。在 C# 中，可以使用 `RichTextBox` 控件实现类似 Word 的项目编号功能，本实例运行效果如图 9.73 所示。

实现过程如下。

(1) 打开 Visual Studio 2008 开发环境，新建一个 Windows 窗体应用程序，并将其命名为 `DisplayNumber`。

(2) 更改默认窗体 `Form1` 的 `Name` 属性为 `Frm_Main`，向窗体中添加 4 个 `Button` 按钮，分别用于打开 RTF 文件；保存 RTF 文件；为 `RichTextBox` 控件中的文本添加项目符号；为 `RichTextBox` 控件中的文本添加数字编号。

(3) 程序主要代码如下。

```

private RichTextBoxEx richTextBox1 = new RichTextBoxEx(); //创建自定义的RichTextBox控件的实例
private void programNumeration_Click(object sender, EventArgs e) //单击“项目符号”按钮
{
    if(richTextBox1.SelectionBullet) //若项目符号样式已经应用到当前选定内容

```



图 9.73 实现类似 Word 的项目编号功能

```

    {
        richTextBox1.SelectionBullet = false; //则取消项目编号
    }
    else //若项目符号样式未应用到当前选定内容
    {
        richTextBox1.SelectionBullet = true; //将项目符号样式应用到当前选定内容
    }
}
private void figuresNumeration_Click(object sender, EventArgs e)
{
    richTextBox1.BulletType = RichTextBoxEx.AdvRichTextBulletType.Number;//在当前选定内容前应用数字编号
}

```

**DIY:** 设置 RichTextBox 控件中文本的对齐方式。提示：可使用 RichTextBox 控件的 SelectionAlignment 属性设置选定文本的对齐方式。（20 分）（实例位置：光盘\mr\09\qjyy\03\_diy）

#### 9.9.4 情景应用 4——制作带历史信息的菜单

视频讲解：光盘\mr\09\lx\制作带历史信息的菜单.exe

实例位置：光盘\mr\09\qjyy\04

很多软件会在程序菜单中记录最近打开文档的历史信息，当用户再次打开应用程序时可以方便地从历史信息中找到曾经处理过的文档。本实例将介绍如何创建带历史信息的菜单，实例运行效果如图 9.74 所示。

实现过程如下。

(1) 打开 Visual Studio 2008 开发环境，新建一个 Windows 窗体应用程序，并将其命名为 HistoryMenu。

(2) 更改默认窗体 Form1 的 Name 属性为 Frm\_Main，向窗体中添加一个 ToolStrip 控件。此控件用于打开图像文件，并记录打开文件的历史信息。

(3) 程序主要代码如下。

```

private void Form1_Load(object sender, EventArgs e)
{
    StreamReader sr = new StreamReader(address + "\\History.ini"); //创建流读取器对象
    int i = 文件 ToolStripMenuItem.DropDownItems.Count - 2; //得到菜单项索引
    while (sr.Peek() >= 0) //循环读取流中文本
    {
        ToolStripMenuItem menuitem = new ToolStripMenuItem(sr.ReadLine()); //创建菜单项对象
        this.文件 ToolStripMenuItem.DropDownItems.Insert(i, menuitem); //向菜单中添加新项
        i++; //向菜单中插入索引的位置
        menuitem.Click += new EventHandler(menuitem_Click); //添加点击事件
    }
    sr.Close(); //关闭流
}

```

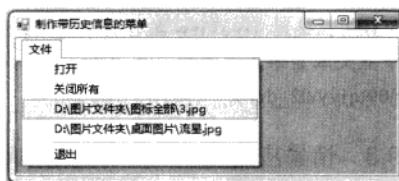


图 9.74 制作带历史信息的菜单

**◆ 注意：** 在 Insert 方法中，其 Index 参数的数值不能超出 DropDownItems 集合的索引范围，否则会抛出异常。

**DIY：** 制作可以伸缩的菜单。提示：要实现可以伸缩的菜单，关键是要使用一个控制菜单伸缩状态的变量，同时调用 ToolStripMenuItem 类的 ShowDropDown 方法。（20 分）（实例位置：光盘\mr\09\qjyy\04\_diy）

### 9.9.5 情景应用 5——制作仿 XP 系统的任务栏菜单

视频讲解：光盘\mr\09\lx\制作仿XP系统的任务栏菜单.exe

实例位置：光盘\mr\09\qjyy\05

在窗体应用程序开发过程中，为了使窗体更加美观、实用，我们可以模仿 XP 系统任务栏菜单的效果。那么，如何设计任务栏菜单呢？这正是本实例所要演示的内容，其运行效果如图 9.75 所示。

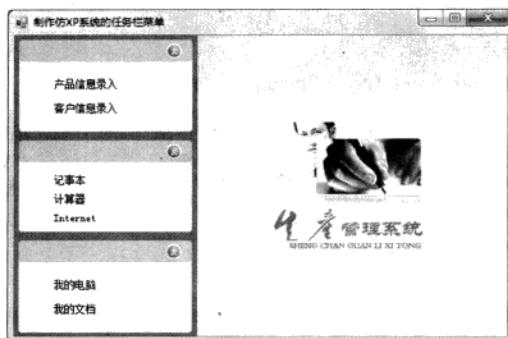


图 9.75 制作仿 XP 系统的任务栏菜单

实现过程如下。

- (1) 打开 Visual Studio 2008 开发环境，新建一个 Windows 窗体应用程序，并将其命名为 LikesXP。
- (2) 更改默认窗体 Form1 的 Name 属性为 Frm\_Main，更改 Text 属性为“仿 XP 系统的任务栏菜单”，向窗体中添加 4 个 Panel 控件，用于建立仿 XP 系统的任务栏菜单。

(3) 程序主要代码如下。

```
private void pictureBox_1_Click(object sender, EventArgs e) //当单击图片控件时触发 Click
{
    Var_i = Convert.ToInt16(((PictureBox)sender).Tag.ToString()); //得到控件中的数据
    switch (Var_i)
    {
        case 1:
        {
            Var_Panel = panel_Gut_1; //得到面板对象引用
            Var_Pict = pictureBox_1; //得到 PictureBox 对象引用
            break;
        }
        case 2:
        {
            Var_Panel = panel_Gut_2; //得到面板对象引用
            Var_Pict = pictureBox_2; //得到 PictureBox 对象引用
            break;
        }
        case 3:
        {
            Var_Panel = panel_Gut_3; //得到面板对象引用
            Var_Pict = pictureBox_3; //得到 PictureBox 对象引用
            break;
        }
    }
}
```

```
        break;
    }
}
if (Convert.ToInt16(Var_Panel.Tag.ToString()) == 0 || Convert.ToInt16(Var_Panel.Tag.ToString()) == 2)
{
    Var_Panel.Tag = 1;                                //设置为隐藏标识
    Var_Pict.Image = Properties.Resources.朝下按钮;   //设置图像属性
    Var_Panel.Visible = false;                         //隐藏面板
}
else
{
    if (Convert.ToInt16(Var_Panel.Tag.ToString()) == 1)
    {
        Var_Panel.Tag = 2;                            //设置为显示标识
        Var_Pict.Image = Properties.Resources.朝上按钮; //设置图像属性
        Var_Panel.Visible = true;                     //显示面板
    }
}
```

 说明：如果设置 Panel 控件的 Visible 属性为 false，则 Panel 面板及面板中的控件都会被隐藏。

**DIY:** 制作一个带导航菜单的主界面。提示：可考虑使用 Button 控件和 ListView 控件制作导航菜单界面。(20 分)(实例位置：光盘\mr\09\qjyy\05\div)

### 情景应用 DIY 栏目分数统计：

DIY 题目	1	2	3	4	5	总分数	
分数							

## 9.10 自我测试

### 一、选择题（每题 10 分，5 道题）

1. 如果要检测用户是否在 TextBox 控件内按了 Enter 键，可以通过触发该控件的（ ）事件来实现。  
A. Click                  B. KeyDown                  C. KeyPress                  D. MouseDown
  2. 通过把 RadioButton 控件的（ ）属性设置为 true，可以使用户只需单击一组单选按钮中的任意一个，即可自动清除同组其他单选按钮的选中状态。  
A. Checked                  B. CheckAlign                  C. AutoCheck                  D. TextAlign
  3. 在以下选项中，（ ）不是容器控件。  
A. Form 窗体                  B. GroupBox 控件                  C. ListBox 控件                  D. Panel 控件
  4. 通过把 Button 控件的（ ）属性设置为 false，可以使该控件变为不可用状态，即控件变灰色。  
A. Visible                  B. Enabled                  C. Dock                  D. ForeColor
  5. 通过把 ListBox 控件的 SelectionMode 属性设置为（ ），可以用 Shift+Ctrl 键选择列表框中的多个列表项。  
A. MultiExtended                  B. MultiSimple                  C. One                  D. None

## 二、填空题（每题 10 分，5 道题）

1. CheckBox 控件的 TextAlign 属性取值于（ ）。
2. TextBox 控件的 MultiLine 属性值的类型为（ ）。
3. 如果想获得 TextBox 中插入点的位置，首先应把该控件的（ ）属性设置为 0，然后通过文本框的（ ）属性获得插入点的位置。
4. 设置一个密码框，当用户输入文本时，它会显示“#”。此时应设置文本框的（ ）属性。
5. 当鼠标指针进入控件时，引发（ ）事件；当鼠标指针离开控件时，引发（ ）事件；当鼠标指针停留在控件上时，引发（ ）事件。

测试分数统计：

类别	第1题	第2题	第3题	第4题	第5题
选择题分数					
填空题分数					
总分数					

## 9.11 行动指南

开始日期： 年 月 日

结束日期： 年 月 日

序号	内 容	行动指南	
1	照猫画虎栏目 分数（ ）	分数>75 分	优秀，基本功掌握得不错，加油！
		75 分>分数>50 分	及格，知识掌握得不牢，重新做一遍照猫画虎。
		分数<50 分	请使用光盘中提供的“实战能力测试系统”进行提高训练。
	情景应用栏目 分数（ ）	分数>75 分	优秀，综合应用能力很强。
		75 分>分数>50 分	及格，综合应用能力需提高，再练一遍情景应用。
		分数<50 分	请使用光盘中提供的“实战能力测试系统”进行提高训练。
	自我测试栏目 分数（ ）	分数>75 分	优秀，有成为编程高手的潜质。
		分数<75 分	请使用光盘中提供的“实战能力测试系统”进行提高训练。
	综合评价	反复训练后，以上各项分数都在 75 分以上，方可进入下一堂课学习。	
2	编程能力培养：本栏目提供了一些实践题目，对于培养编程能力很有效，要做好。	1. 制作图片和文字混合编辑工具。提示：创建一个 Windows 应用程序，在默认窗体中添加两个 Button 控件和一个 RichTextBox 控件，其中 Button 控件用来执行打开文件、插入图片的操作；RichTextBox 控件用来显示文件和图片。 2. 用 ComboBox 控件制作浏览器网址输入框。提示：主要会用到 ComboBox 控件的 KeyDown 事件和TextChanged 事件。KeyDown 事件是在用户首次按下某个键时发生的事件；TextChanged 事件是当用户在 ComboBox 控件上更改 Text 属性值时引发的事件。	
3	编程习惯培养：本堂课培养读者在学习开发中记笔记的习惯，把开发中遇到的问题、总结的经验记录下来。		

续表

序号	内 容	行 动 指 南
4	创新能力培养：看看身边有没有可以用编程解决的问题，记录到右边的表格中。根据学习进度，尝试编写解决这些问题的小程序。	

## 9.12 成功可以复制——前微软 CEO 比尔·盖茨

比尔·盖茨出生于 1955 年 10 月 28 日，与两个姐姐一起在美国西雅图长大。他们的父亲是西雅图的一名律师，母亲是一名学校教师。比尔从小就精力过人，早在婴儿时期自己就能让摇篮晃动起来，从小就极爱思考，一迷上某事便能全身心投入。外祖母经常和比尔玩游戏，尤其是一些智力游戏，在外祖母的帮助下与指导下，比尔成了兴趣广泛、废寝忘食的读者——读书成了他打发精力的好方式。随着年龄的增长，家庭中的环境已无法满足比尔·盖茨天赋的进一步发挥。于是，他的父母把目光投向社会，积极为比尔寻找属于他的空间。

小学毕业后，父母在征求比尔·盖茨的意见后，送他进了湖滨中学。在湖滨中学，比尔痴迷上令他日后倾注毕生精力的计算机机。

比尔·盖茨在湖滨中学读书时，常按自己的兴趣爱好来安排学习。他在喜欢的课程上下功夫，学得非常棒，如数学和阅读方面。每次父母看到比尔拿回来的成绩单，尽管他们知道比尔在一些课程上会学得更好，但他们并没有责备他。

中学毕业后，比尔·盖茨很想到哈佛大学去读书，这也正是父母们最大的心愿。比尔·盖茨的父母是非常开明和民主的，经过冷静的思考之后，父母放弃了让儿子当律师的想法，让比尔·盖茨在大学领域里自由发展。这一点帮了比尔·盖茨的大忙。

1975 年，年仅 19 岁的盖茨就预言：“我们意识到软件时代到来了，并且对于芯片的长期潜能我们有足够的洞察力，这意味着什么？我现在不去抓住机会反而去完成我的哈佛学业，软件工业绝对不会原地踏步等着我。”所以比尔·盖茨决定离开哈佛大学，放弃锦绣的学业，与别人一起创办计算机公司！为了征得父母的同意，比尔与父母多次交谈，平静地表达了自己的想法。了解儿子秉性和志向的父母又能说什么呢！或许儿子的天赋与计算机事业是最佳的切合点吧！最终比尔·盖茨毅然离开了令亿万学子向往的哈佛大学，开始在软件领域大展鸿图。

实践证明，比尔·盖茨成就了非凡的事业，他使个人计算机成了日常生活用品，并因而改变了每一个现代人的工作、生活乃至交往的方式。



微软总部

因此有人说，比尔·盖茨对软件的贡献，就如同爱迪生发明了灯泡一样。

### 经典语录

不要让这个世界的复杂性阻碍你前进，要成为一个行动主义者。将解决人类的不平等视为己任，它将成为你生命中最重要的经历之一。

### 深度评价

比尔·盖茨在19岁时曾经预言：“我们意识到软件时代到来了，并且对于芯片的长期潜能我们有足够的洞察力，这意味着什么？我现在不去抓住机会反而去完成我的哈佛学业，软件工业绝对不会原地踏步等着我。”他的经历告诉我们：一个人想要成功，就要学会在机遇从头顶上飞过时跳起来抓住它，这样抓住机遇的机会就会大大增大。一旦做出决定就不要拖延。任何事情想到就去做！立即行动！



# 第 10 堂课

---

## Windows 应用程序高级控件

( 视频讲解：170 分钟)

在第 9 堂课中已经对 Windows 应用程序中的一些常用控件进行了讲解，本堂课将在此基础上对 Windows 应用程序中的一些高级控件进行讲解，这些高级控件在设计比较复杂的窗体界面时经常用到。

学习摘要：

- ▶ 熟悉存储图像控件的使用
- ▶ 掌握列表视图控件的使用
- ▶ 掌握树控件的使用
- ▶ 熟悉日期控件的使用
- ▶ 熟悉月历控件的使用
- ▶ 熟悉 ErrorProvider 控件和 HelpProvider 控件的使用
- ▶ 掌握 Timer 计时器的使用
- ▶ 掌握 ProgressBar 进度条控件的使用

## 10.1 存储图像控件

存储图像控件 (ImageList 控件) 用于存储图像资源，并在控件上显示出来，这样就简化了对图像的管理。ImageList 控件的主要属性是 Images，它包含关联控件将要使用的图片。每个单独的图像可以通过其索引值或键值来访问。另外，ImageList 控件中的所有图像都将以同样的大小显示，该大小由其 ImageSize 属性设置，较大的图像将缩小至适当的尺寸。如图 10.1 所示为 ImageList 控件。

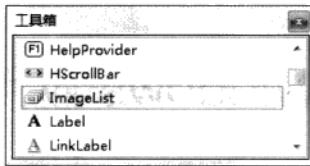


图 10.1 ImageList 控件

### 10.1.1 在 ImageList 控件中添加图像

向 ImageList 控件中添加图像时，需要用到其 Images 属性的 Add 方法。该方法主要用来将指定的图像添加到 ImageList 控件中，其语法如下。

语法：.public void Add(Image value)

说明：参数 value 表示要添加到列表中的图像。

**例 10.01** 创建一个 Windows 应用程序，首先获取图像的路径，然后通过 ImageList 控件的 Images 属性的 Add 方法向控件中添加图像，最后单击按钮显示 ImageList 控件中指定的图片，代码如下。（实例位置：光盘\mr\10\sl\10.01）

```
private void Form1_Load(object sender, EventArgs e)
{
    string Path = "01.jpg"; //设置要加载的第一张图片的路径
    string Path2 = "02.jpg"; //设置要加载的第二张图片的路径
    Image Mimg=Image.FromFile(Path,true); //创建一个 Image 对象
    imageList1.Images.Add(Mimg); //使用 Images 属性的 Add 方法向控件中添加图像
    Image Mimg2 = Image.FromFile(Path2, true); //创建一个 Image 对象
    imageList1.Images.Add(Mimg2); //使用 Images 属性的 Add 方法向控件中添加图像
    imageList1.ImageSize = new Size(200,165); //设置显示图片的大小
    pictureBox1.Width = 200; //设置 pictureBox1 控件的宽
    pictureBox1.Height = 165; //设置 pictureBox1 控件的高
}
private void button1_Click(object sender, EventArgs e)
{
    pictureBox1.Image = imageList1.Images[0]; //设置 pictureBox1 的图像索引是 imageList1 控件索引为 0 的图片
}
private void button2_Click(object sender, EventArgs e)
{
    pictureBox1.Image = imageList1.Images[1];//设置 pictureBox1 的图像索引是 imageList1 控件索引为 1 的图片
}
```

程序运行结果如图 10.2 所示。



图 10.2 显示添加后的图像

### 10.1.2 在 ImageList 控件中移除图像

在 ImageList 控件中可以使用 RemoveAt 方法移除单个图像，或者使用 Clear 方法清除图像列表中的所有图像。下面详细介绍 RemoveAt 方法和 Clear 方法。

RemoveAt 方法用于从列表中移除图像，其语法如下。

语法：public void RemoveAt(int index)

说明：参数 index 表示要移除的图像的索引。

Clear 方法主要用于从 ImageList 中移除所有图像，其语法如下。

语法：public void Clear()

**例 10.02** 创建一个 Windows 应用程序，在其默认窗体中添加两个 Button 控件，分别用来执行加载图像和移除图像功能。单击“加载图像”按钮，向 ImageList 控件中添加图像，并显示在 PictureBox 控件中；单击“移除图像”按钮，调用 ImageList 控件的 Images 属性的 RemoveAt 方法移除 ImageList 控件中的图像，并将 PictureBox 控件中的图像清空。代码如下。（实例位置：光盘\mr\10\sl\10.02）

```
private void button1_Click(object sender, EventArgs e)
{
    pictureBox1.Width = 200; //设置 pictureBox1 控件的宽
    pictureBox1.Height = 165; //设置 pictureBox1 控件的高
    string Path = "01.jpg"; //设置要加载图片的路径
    Image img = Image.FromFile(Path, true); //创建 Image 对象
    imageList1.Images.Add(img); //使用 Images 属性的 Add 方法向控件中添加图像
    imageSize = new Size(200,165); //设置显示图片的大小
    pictureBox1.Image = imageList1.Images[0];//使 pictureBox1 控件显示 imageList1 控件中索引为 0 的图像
}
private void button2_Click(object sender, EventArgs e)
{
    imageList1.Images.RemoveAt(0); //使用 RemoveAt 方法移除图像
    pictureBox1.Image = null; //清空图像
}
```

程序运行结果如图 10.3 所示。

另外，还可以使用 Clear 方法从 ImageList 控件中移除所有图像，代码如下。

```
imageList1.Images.Clear(); //使用 Clear 方法移除所有图像
```

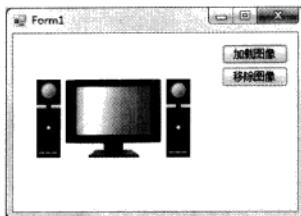


图 10.3 移除控件中的图像

## 10.2 列表视图控件

列表视图控件（ListView 控件）显示带图标的项列表，其中可以显示大图标、小图标和数据。使用该控件可以创建类似 Windows 资源管理器右边窗口的用户界面。如图 10.4 所示为 ListView 控件。

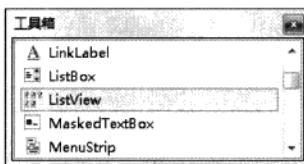


图 10.4 ListView 控件

ListView 控件可以通过 View 属性设置项在控件中显示的方式，View 属性的值及说明如表 10.1 所示。

表 10.1 View 属性的值及说明

属性值	说明
Details	每个项显示在不同的行上，并带有关于列中所排列的各项的进一步信息。最左边的列包含一个小图标和标签，后面的列包含应用程序指定的子项。列显示一个标头，它可以显示列的标题。用户可以在运行时调整各列的大小
LargeIcon	每个项都显示为一个最大的图标，在它的下面有一个标签。这是默认的视图模式
List	每个项都显示为一个小图标，在它右边带一个标签，各项排列在列中，没有列标头
SmallIcon	每个项都显示为一个小图标，在它右边带一个标签
Title	每个项都显示为一个完整大小的图标，在它的右边带项标签和子项信息。显示的子项信息由应用程序指定。此视图仅在 Windows XP 和 Windows Server 2003 系列平台上受支持。在之前的操作系统上，此值被忽略，并且 ListView 控件在 LargeIcon 视图中显示

### 10.2.1 在 ListView 控件中添加移除项

#### 1. 添加项

向 ListView 控件中添加项时需要用到其 Items 属性的 Add 方法。该方法主要用于将项添加至项的集合中，语法如下。

语法：public virtual ListViewItem Add(string text,int imageIndex)

说明：参数 text 表示项的文本；参数 imageIndex 表示要为该项显示的图像的索引。该方法返回已添加

到集合中的 ListViewItem。

**例 10.03** 创建一个 Windows 应用程序，通过使用 ListView 控件的 Items 属性的 Add 方法向控件中添加项，代码如下。（实例位置：光盘\mr\10\sl\10.03）

```
private void button1_Click(object sender, EventArgs e)
{
    if (textBox1.Text == "") //判断文本框中是否输入了数据
    {
        MessageBox.Show("项目不能为空"); //如果没有输入数据，则弹出提示
    }
    else
    {
        listView1.Items.Add(textBox1.Text.Trim()); //使用 ListView 控件的 Items 属性的 Add 方法向控件中添加项
    }
}
```

程序运行结果如图 10.5 所示。

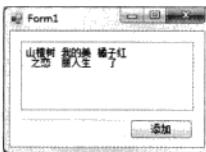


图 10.5 添加项目

## 2. 移除项

当移除 ListView 控件中的项时可以使用其 Items 属性的 RemoveAt 方法或 Clear 方法，其中 RemoveAt 方法用于移除指定的项，而 Clear 方法用于移除列表中的所有项。下面介绍 RemoveAt 和 Clear 方法。

**RemoveAt** 方法用于移除集合中指定索引处的项，其语法如下。

语法：public virtual void RemoveAt(int index)

说明：参数 index 表示从零开始的索引（属于要移除的项）。

**Clear** 方法用于从集合中移除所有项，其语法如下。

语法：public virtual void Clear()

**例 10.04** 创建一个 Windows 应用程序，向 ListView 控件中添加 5 项，然后选择要移除的项，单击“移除项”按钮，调用 ListView 控件的 Items 属性的 RemoveAt 方法移除选中的项；单击“清空”按钮，调用 Clear 方法清空所有的项，代码如下。（实例位置：光盘\mr\10\sl\10.04）

```
private void button1_Click(object sender, EventArgs e)
{
    if (textBox1.Text == "") //判断文本框中是否输入数据
    {
        MessageBox.Show("项目不能为空"); //如果没有输入数据，则弹出提示
    }
    else
    {
        listView1.Items.Add(textBox1.Text.Trim()); //使用 Add 方法向控件中添加数据
    }
}

private void button3_Click(object sender, EventArgs e)
```

```

{
    if (listView1.Items.Count == 0) //判断控件中是否存在项目
    {
        MessageBox.Show("项目中已经没有项目"); //如果没有项目，则弹出提示
    }
    else
    {
        listView1.Items.Clear(); //使用 Clear 方法移除所有项目
    }
}

private void button2_Click(object sender, EventArgs e)
{
    if (listView1.SelectedItems.Count == 0) //判断是否选择了要删除的项
    {
        MessageBox.Show("请选择要删除的项"); //如果没有选择，则弹出提示
    }
    else
    {
        listView1.Items.RemoveAt(listView1.SelectedItems[0].Index); //使用 RemoveAt 方法移除选择的项目
        listView1.SelectedItems.Clear(); //取消控件的选择
    }
}
}

```

程序运行结果如图 10.6 和图 10.7 所示。



图 10.6 移除项目之前



图 10.7 移除项目之后

### 10.2.2 选择 ListView 控件中的项

当选择 ListView 控件中的项时可以使用其 Selected 属性，该属性主要用于获取或设置一个值，该值指示是否选定此项，其语法如下。

语法：public bool Selected { get; set; }

说明：如果选定此项，则属性值为 true；否则为 false。

**例 10.05** 创建一个 Windows 应用程序，向 ListView 控件中添加 3 项，然后设置控件中第 3 项的 Selected 属性为 true，即设置为选中第 3 项，代码如下。（实例位置：光盘\mr\10\sl\10.05）

```

private void Form1_Load(object sender, EventArgs e)
{
    listView1.Items.Add("明日科技"); //使用 Add 方法向控件中添加项目
    listView1.Items.Add("C#编程词典"); //使用 Add 方法向控件中添加项目
    listView1.Items.Add("C#从基础到项目实战"); //使用 Add 方法向控件中添加项目
    listView1.Items[2].Selected = true; //使用 Selected 方法选中第 3 项
}

```

程序运行结果如图 10.8 所示。

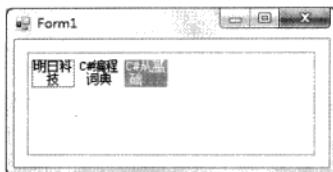


图 10.8 设置控件选择项

### 10.2.3 为 ListView 控件中的项添加图标

如果要为 ListView 控件中的项添加图标，需要使用 ImageList 控件设置 ListView 控件中项的图标。ListView 控件可显示 3 个图像列表中的图标，其中 List 视图、Details 视图和 SmallIcon 视图显示 SmallImageList 属性中指定的图像列表中的图像；LargeIcon 视图显示 LargeImageList 属性中指定的图像列表中的图像；列表视图在大图标或小图标旁显示 StateImageList 属性中设置的一组附加图标。实现步骤如下。

(1) 将相应的属性（SmallImageList、LargeImageList 或 StateImageList）设置为想要使用的现有 ImageList 控件。

(2) 为每个具有关联图标的列表项设置 ImageIndex 属性或 StateImageIndex 属性，这些属性可以在代码中设置，也可以在“ListViewItem 集合编辑器”中进行设置。若要在“ListViewItem 集合编辑器”中进行设置，需在“属性”窗口中单击 Items 属性旁的省略号按钮。

**例 10.06** 创建一个 Windows 应用程序，设置 ListView 控件的 LargeImageList 属性和 SmallImageList 属性为 imageList1 控件，然后通过编写代码向 ImageList 控件中添加图像，最后向 ListView 控件中添加两项，并设置这两项的 ImageIndex 属性分别为 0 和 1，代码如下。（实例位置：光盘\mr\10\sl\10.06）

```
private void Form1_Load(object sender, EventArgs e)
{
    listView1.LargeImageList = imageList1; //设置控件的 LargeImageList 属性
    imageList1.ImageSize = new Size(37, 36); //设置 ImageList 控件图标的大
    imageList1.Images.Add(Image.FromFile("01.png")); //向 imageList1 中添加图标
    imageList1.Images.Add(Image.FromFile("02.png")); //向 imageList1 中添加图标
    listView1.SmallImageList = imageList1; //设置控件的 SmallImageList 属性
    listView1.Items.Add("明日科技"); //向控件中添加两项
    listView1.Items.Add("C#编程词典");
    listView1.Items[0].ImageIndex = 0; //控件中第一项的图标索引为 0
    listView1.Items[1].ImageIndex = 1; //控件中第二项的图标索引为 1
}
```

程序运行结果如图 10.9 所示。



图 10.9 为控件中的项添加图标

### 10.2.4 在 ListView 控件中启用平铺视图

通过启用 ListView 控件的平铺视图功能，可以在图形信息和文本信息之间提供一种视觉平衡。在 ListView 控件中，平铺视图与分组功能或插入标记功能可以结合使用。如果要启用平铺视图，需要将 ListView 控件的 View 属性设置为 Tile，另外，还可以通过设置 TileSize 属性来调整平铺的大小。

**例 10.07** 创建一个 Windows 应用程序，将 ListView 控件的 View 属性设置为 Tile，以便启用平铺视图，然后为 ImageList 控件添加两张图片用来作为 ListView 控件中项的图标，向 ListView 控件中添加 5 项，并分别设置各项的图标，最后通过 ListView 控件的 TileSize 属性设置平铺的宽、高分别为 200 和 50。代码如下。（实例位置：光盘\mr\10\s\10.07）

```
private void Form1_Load(object sender, EventArgs e)
{
    listView1.View = View.Tile; //设置 listView1 控件的 View 属性
    listView1.LargeImageList = imageList1; //设置控件的 LargeImageList 属性，其大图标在 imageList1 控件中选择
    //向 imageList1 控件中添加两张图片
    imageList1.Images.Add(Image.FromFile("1.bmp"));
    imageList1.Images.Add(Image.FromFile("2.bmp"));
    //向控件中添加项目
    listView1.Items.Add("明日科技");
    listView1.Items.Add("C#编程词典");
    listView1.Items.Add("C#从基础到项目实战");
    listView1.Items.Add("C#项目开发全程实录");
    listView1.Items.Add("C#开发典型模块大全");
    //设置控件中项目的图标
    listView1.Items[0].ImageIndex = 0;
    listView1.Items[1].ImageIndex = 1;
    listView1.Items[2].ImageIndex = 0;
    listView1.Items[3].ImageIndex = 1;
    listView1.Items[4].ImageIndex = 0;
    listView1.TileSize = new Size(200,50); //设置 listView1 控件的 TileSize 属性
}
```

程序运行结果如图 10.10 所示。

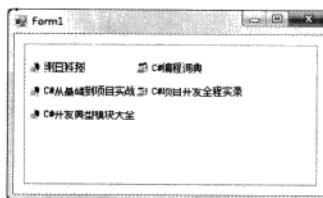


图 10.10 启用平铺视图

## 10.3 树控件

树控件（TreeView 控件）可以为用户显示节点层次结构，而每个节点又可以包含子节点。包含子节点的节点称为父节点，其效果就像在 Windows 操作系统的 Windows 资源管理器功能的左窗口中显示文件和文

件夹一样。如图 10.11 所示为 TreeView 控件。

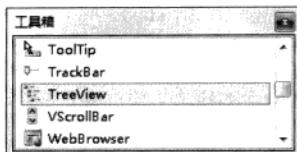


图 10.11 TreeView 控件

### 10.3.1 添加和删除树节点

#### 1. 添加节点

向 TreeView 控件中添加节点时，需要用到其 Nodes 属性的 Add 方法，其语法如下。

语法：public virtual int Add(TreeNode node)

说明：参数 node 表示要添加到集合中的 TreeNode，该方法返回被添加节点在 TreeNode 集合中的索引。

**例 10.08** 创建一个 Windows 应用程序，使用 TreeView 控件的 Nodes 属性的 Add 方法向树控件中添加两个父节点，然后再使用 Add 方法分别向两个父节点中添加 3 个子节点，代码如下。（实例位置：光盘\mr\10\10\10.08）

```
private void Form1_Load(object sender, EventArgs e)
{
    //为树控件建立两个父节点
    TreeNode tn1 = treeView1.Nodes.Add("名称");
    TreeNode tn2 = treeView1.Nodes.Add("类别");
    //建立 3 个子节点，用于显示名称
    TreeNode Ntn1 = new TreeNode("明日科技");
    TreeNode Ntn2 = new TreeNode("C#编程词典");
    TreeNode Ntn3 = new TreeNode("C#从基础到项目实战");
    //将以上 3 个子节点添加到第一个父节点中
    tn1.Nodes.Add(Ntn1);
    tn1.Nodes.Add(Ntn2);
    tn1.Nodes.Add(Ntn3);
    //然后再建立 3 个子节点，用于显示类别
    TreeNode Stn1 = new TreeNode("公司");
    TreeNode Stn2 = new TreeNode("软件");
    TreeNode Stn3 = new TreeNode("图书");
    //将以上 3 个子节点添加到第二个父节点中
    tn2.Nodes.Add(Stn1);
    tn2.Nodes.Add(Stn2);
    tn2.Nodes.Add(Stn3);
}
```

程序运行结果如图 10.12 所示。

#### 2. 移除节点

从 TreeView 控件中移除指定的树节点时，需要使用其 Nodes 属性的 Remove 方法，其语法如下。

语法：public void Remove(TreeNode node)

说明：node 表示要移除的 TreeNode。

**例 10.09** 创建一个 Windows 应用程序，通过 TreeView 控件的 Nodes 属性的 Remove 方法删除其选中的子节点，代码如下。（实例位置：光盘\mr\10\s\10.09）

```
private void Form1_Load(object sender, EventArgs e)
{
    TreeNode tn1 = treeView1.Nodes.Add("名称"); //建立一个父节点
    //建立 3 个子节点
    TreeNode Ntn1 = new TreeNode("明日科技");
    TreeNode Ntn2 = new TreeNode("C#编程词典");
    TreeNode Ntn3 = new TreeNode("C#从基础到项目实战");
    //将这 3 个子节点添加到父节点中
    tn1.Nodes.Add(Ntn1);
    tn1.Nodes.Add(Ntn2);
    tn1.Nodes.Add(Ntn3);
}
private void button1_Click(object sender, EventArgs e)
{
    //如果用户选择了“名称”，则证明没有选择要删除的子节点
    if (treeView1.SelectedNode.Text == "名称")
    {
        MessageBox.Show("请选择要删除的子节点");
    }
    else
    {
        treeView1.Nodes.Remove(treeView1.SelectedNode); //使用 Remove 方法移除选择项
    }
}
```

程序运行结果如图 10.13 所示。

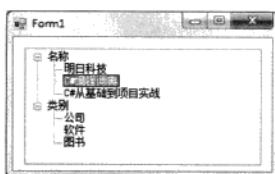


图 10.12 添加节点



图 10.13 删除子节点

### 10.3.2 获取树控件中选中的节点

获取 TreeView 树控件中选中的节点时，可以在该控件的 AfterSelect 事件中使用 EventArgs 对象返回对已选中节点对象的引用，其中通过检查 TreeViewEventArgs 类（它包含与事件有关的数据），确定单击了哪个节点。

**例 10.10** 创建一个 Windows 应用程序，在 TreeView 控件的 AfterSelect 事件中获取该控件中选中节点的文本，代码如下。（实例位置：光盘\mr\10\s\10.10）

```
private void Form1_Load(object sender, EventArgs e)
{
    TreeNode tn1 = treeView1.Nodes.Add("名称"); //建立一个父节点
    //建立 3 个子节点
```

```

TreeNode Ntn1 = new TreeNode("明日科技");
TreeNode Ntn2 = new TreeNode("C#编程词典");
TreeNode Ntn3 = new TreeNode("C#从基础到项目实战");
//将 3 个子节点添加到父节点中
tn1.Nodes.Add(Ntn1);
tn1.Nodes.Add(Ntn2);
tn1.Nodes.Add(Ntn3);
}
private void treeView1_AfterSelect(object sender, TreeViewEventArgs e)
{
    label1.Text = "当前选中的节点: " + e.Node.Text; //在 AfterSelect 事件中获取控件中选中节点显示的文本
}

```

程序运行结果如图 10.14 所示。

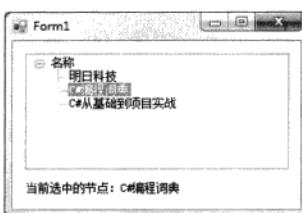


图 10.14 获取选中的节点

### 10.3.3 为树控件中的节点设置图标

在 TreeView 控件中可在每个节点的紧挨节点文本的左侧显示图标，但显示时必须使树视图与 ImageList 控件相关联。为 TreeView 控件中的节点设置图标的步骤如下。

(1) 设置 TreeView 控件的 ImageList 属性为想要使用的现有 ImageList 控件，该属性既可在设计器中使用“属性”窗口进行设置，也可在代码中设置。

**例 10.11** 设置 treeView1 控件的 ImageList 属性为 imageList1，代码如下。

```
treeView1.ImageList = imageList1;
```

(2) 设置树节点的 ImageIndex 和 SelectedImageIndex 属性，其中 ImageIndex 属性用来确定正常和展开状态下的节点显示的图像，而 SelectedImageIndex 属性用来确定选定状态下的节点显示图像。

**例 10.12** 设置 treeView1 控件的 ImageIndex 属性，确定正常或展开状态下的节点显示的图像的索引为 0，设置 SelectedImageIndex 属性，确定选定状态下的节点显示的图像的索引为 1，代码如下。

```
treeView1.ImageIndex = 0;
treeView1.SelectedImageIndex = 1;
```

**例 10.13** 创建一个 Windows 应用程序，向 TreeView 控件中添加一个父节点和 3 个子节点，设置 TreeView 控件的 ImageList 属性为 imageList1，并通过设置该控件的 ImageIndex 属性实现正常状况下节点显示的图像的索引为 0，然后设置该控件的 SelectedImageIndex 属性，实现选中某个节点后显示的图像的索引为 1，代码如下。（实例位置：光盘\mr\10\sl\10.13）

```

private void Form1_Load(object sender, EventArgs e)
{
    TreeNode tn1 = treeView1.Nodes.Add("组织结构"); //建立一个父节点
    //建立 3 个子节点
    TreeNode Ntn1 = new TreeNode("C#部门");

```

```

TreeNode Ntn2 = new TreeNode("ASP.NET 部门");
TreeNode Ntn3 = new TreeNode("VB 部门");
//将 3 个子节点添加到父节点中
tn1.Nodes.Add(Ntn1);
tn1.Nodes.Add(Ntn2);
tn1.Nodes.Add(Ntn3);
//设置 imageList1 控件中显示的图像
imageList1.Images.Add(Image.FromFile("1.png"));
imageList1.Images.Add(Image.FromFile("2.png"));
//设置 treeView1 的 ImageList 属性为 imageList1
treeView1.ImageList = imageList1;
imageList1.ImageSize = new Size(16,16);
treeView1.ImageIndex = 0;           //设置 treeView1 控件节点的图标在 imageList1 控件中的索引是 0
treeView1.SelectedImageIndex = 1;   //选择某个节点后显示的图标在 imageList1 控件中的索引是 1
}

```

程序运行结果如图 10.15 和图 10.16 所示。

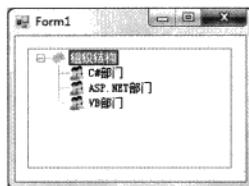


图 10.15 运行程序

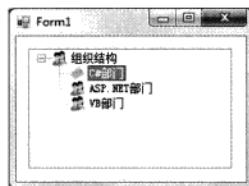


图 10.16 选中节点

## 10.4 日期控件

日期控件 (DateTimePicker 控件) 用于选择日期和时间, 但它只能选择一个时间, 而不是连续的时间段, 另外, 开发人员也可以直接输入日期和时间。如图 10.17 所示为 DateTimePicker 控件。

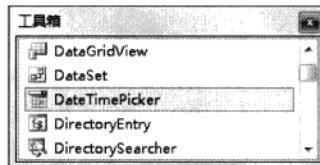


图 10.17 DateTimePicker 控件

### 10.4.1 使用 DateTimePicker 控件显示时间

通过将 DateTimePicker 控件的 Format 属性设置为 Time, 可以实现控件中只显示时间, Format 属性用于获取或设置控件中显示的日期和时间格式, 其语法如下。

语法: public DateTimeFormat Format { get; set; }

说明: 该属性值是 DateTimeFormat 枚举值之一, 默认为 Long。

DateTimeFormat 枚举值及说明如表 10.2 所示。

表 10.2 DateTimePickerFormat 枚举值及说明

枚举值	说明
Custom	DateTimePicker 控件以自定义格式显示日期/时间值
Long	DateTimePicker 控件以用户操作系统设置的长日期格式显示日期/时间值
Short	DateTimePicker 控件以用户操作系统设置的短日期格式显示日期/时间值
Time	DateTimePicker 控件以用户操作系统设置的时间格式显示日期/时间值

**例 10.14** 创建一个 Windows 应用程序，首先将 DateTimePicker 控件的 Format 属性设置为 Time，以便在控件中只显示时间，然后获取 DateTimePicker 控件中显示的时间，并显示到 TextBox 控件中，代码如下。

(实例位置：光盘\mr\10\sl\10.14)

```
private void Form1_Load(object sender, EventArgs e)
{
    dateTimePicker1.Format = DateTimePickerFormat.Time;
    //设置 dateTimePicker1 的 Format 属性为 Time 使其只显示时间
    textBox1.Text = dateTimePicker1.Text;           //使用文本框获取控件显示的时间
}
```

程序运行结果如图 10.18 所示。

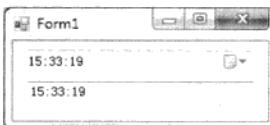


图 10.18 控件只显示时间

#### 10.4.2 使用 DateTimePicker 控件以自定义格式显示日期

以自定义格式在 DateTimePicker 控件中显示日期时，需要用到其 CustomFormat 属性，语法格式如下。

语法：public string CustomFormat { get; set; }

说明：该属性值表示自定义日期/时间格式的字符串。

**◆ 注意：**只有将 Format 属性设置为 DateTimePickerFormat.Custom，才能影响显示的日期和时间的格式设置。

通过组合格式字符串可以设置日期和时间格式，常用的有效日期格式字符串及其说明如表 10.3 所示。

表 10.3 有效日期格式字符串及其说明

格式字符串	说明
d	一位数或两位数的天数
dd	两位数的天数，一位数天数的前面加一个 0
ddd	3 个字符的星期几缩写
dddd	完整的星期几名称
h	12 小时格式的一位数或两位数小时数
hh	12 小时格式的两位数小时数，一位数数值前面加一个 0
H	24 小时格式的一位数或两位数小时数
HH	24 小时格式的两位数小时数，一位数数值前面加一个 0

续表

格式字符串	说 明
m	一位数或两位数分钟值
mm	两位数分钟值，一位数数值前面加一个 0
M	一位数或两位数月份值
MM	两位数月份值。一位数数值前面加一个 0
MMM	3 个字符的月份缩写
MMMM	完整的月份名
s	一位数或两位数秒数
ss	两位数秒数。一位数数值前面加一个 0
t	单字母 A.M./P.M 缩写 (A.M 将显示为 “A” )
tt	两字母 A.M./P.M 缩写 (A.M. 将显示为 “AM” )
y	一位数的年份 (2001 显示为 “1” )
yy	年份的最后两位数 (2001 显示为 “01” )
yyyy	完整的年份 (2001 显示为 “2001” )

**例 10.15** 创建一个 Windows 应用程序，首先将 DateTimePicker 控件的 Format 属性设置为 DateTimeFormat.Custom，然后再将其 CustomFormat 属性设置为自定义的日期格式，显示在 Label 控件中，代码如下。（实例位置：光盘\mr\10\sl\10.15）

```
private void Form1_Load(object sender, EventArgs e)
{
    dateTimePicker1.Format = DateTimePickerFormat.Custom;
    //设置 Format 属性为 Custom，使用户自定义的时间格式生效
    dateTimePicker1.CustomFormat = "MMMM dd, yyyy - dddd";
    //通过控件的 CustomFormat 属性设置自定义的格式
    label1.Text = dateTimePicker1.Text;
    //显示当前控件的自定义格式的日期
}
```

程序运行结果如图 10.19 所示。

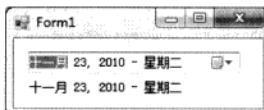


图 10.19 自定义时间格式

#### 10.4.3 返回 DateTimePicker 控件中选择的日期

获取 DateTimePicker 控件中选择的日期时，可以使用其 Text 属性。另外，开发人员还可以通过使用其 Value 属性的 Year、Month 和 Day 属性来获取选中日期的年、月、日等信息。

**例 10.16** 创建一个 Windows 应用程序，首先使用 DateTimePicker 控件的 Text 属性获取当前控件选择的日期，然后分别使用其 Value 属性的 Year、Month 和 Day 属性获取选择日期的年、月和日，并显示在相应的 TextBox 文本框中，代码如下。（实例位置：光盘\mr\10\sl\10.16）

```
private void Form1_Load(object sender, EventArgs e)
{
    textBox1.Text = dateTimePicker1.Text;
    //使用控件的 Text 属性获取当前控件选择的日期
```

```

    textBox2.Text = datePicker1.Value.Year.ToString(); //使用 Value 属性的 Year 方法获取选择日期的年
    textBox3.Text = datePicker1.Value.Month.ToString(); //使用 Value 属性的 Month 方法获取选择日期的月
    textBox4.Text = datePicker1.Value.Day.ToString(); //使用 Value 属性的 Day 方法获取选择日期的日
}

```

程序运行结果如图 10.20 所示。

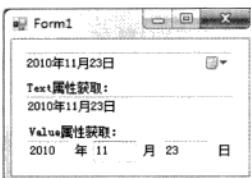


图 10.20 获取控件中选择的日期

## 10.5 月 历 控 件

月历控件（MonthCalendar 控件）提供了一个直观的图形界面，可以让用户查看和设置日期。另外，它还允许使用鼠标进行拖曳，以便选择一段连续的时间，此段连续的时间包括起始时间和结束时间。如图 10.21 所示为 MonthCalendar 控件。

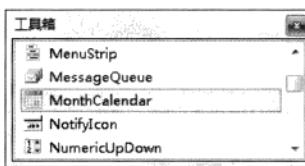


图 10.21 MonthCalendar 控件

### 10.5.1 在 MonthCalendar 控件中以粗体显示特定日期

MonthCalendar 控件能以粗体显示特殊的日期或重复出现的日子，这样做可以引起对特殊日期的注意。在实际应用时，可以使用该控件的 AddBoldedDate 方法在月历中添加以粗体显示的日期，然后调用 UpdateBoldedDates 方法重绘粗体格式的日期。

**例 10.17** 创建一个 Windows 应用程序，其中首先创建一个 DateTime 对象，该对象中指定需要以粗体显示的日期，然后使用 AddBoldedDate 方法在月历中添加以粗体显示的日期，最后调用 UpdateBoldedDates 方法重绘粗体格式的日期，代码如下。（实例位置：光盘\mr\10\s\10.17）

```

private void Form1_Load(object sender, EventArgs e)
{
    DateTime myVacation1 = new DateTime(2009, 3, 20); //创建 DateTime 类，使其值为“2009 年 3 月 20 号”
    monthCalendar1.AddBoldedDate(myVacation1);
    //使用 AddBoldedDate 方法在月历中将“2009 年 3 月 20 号”以粗体显示
    monthCalendar1.UpdateBoldedDates(); //调用 UpdateBoldedDates 方法重绘粗体格式的日期
}

```

程序运行结果如图 10.22 所示。

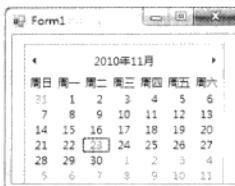


图 10.22 粗体显示特定日期

### 10.5.2 在 MonthCalendar 控件中选择日期范围

如果要在 MonthCalendar 控件中选择日期范围，需要设置其 SelectionStart 属性和 SelectionEnd 属性，这两个属性分别用于设置选择日期的起始时间和结束时间。

**例 10.18** 创建一个 Windows 应用程序，添加一个 MonthCalendar 控件。在该控件的 DateChanged 事件中获取 SelectionStart 属性和 SelectionEnd 属性的值，当 MonthCalendar 控件中选择的日期发生更改时触发 DateChanged 事件。运行程序，选择某个日期作为起始日期，然后按住 Shift 键再选择结束日期，这样即可实现日期范围的选择。代码如下。（实例位置：光盘\mr\10\sl\10.18）

```
private void Form1_Load(object sender, EventArgs e)
{
    textBox1.Text = monthCalendar1.TodayDate.ToString(); // 获取控件当前的日期和时间
}

private void monthCalendar1_DateChanged(object sender, DateRangeEventArgs e)
{
    textBox2.Text = monthCalendar1.SelectionStart.ToString(); // 通过 SelectionStart 属性获取用户选择的起始日期
    textBox3.Text = monthCalendar1.SelectionEnd.ToString(); // 通过 SelectionEnd 属性获取用户选择的结束日期
}
```

程序运行结果如图 10.23 所示。



图 10.23 设置控件中选择日期的范围

## 10.6 其他高级控件

在 Visual Studio 2008 开发环境中提供了很多 Windows 应用程序控件，除了前面所介绍的控件之外，还

包括很多其他的高级控件，如 ErrorProvider 控件、HelpProvider 控件、Timer 控件和 ProgressBar 控件等，本节将对 Windows 应用程序中的一些其他的高级控件进行详细讲解。

### 10.6.1 使用 ErrorProvider 控件验证文本框输入

ErrorProvider 控件可以在不影响用户操作的情况下向用户显示有错误发生，一般来说，在验证用户输入的数据时常用到该控件。如图 10.24 所示为 ErrorProvider 控件。

通过 ErrorProvider 控件的 SetError 方法可以设置指定控件的错误描述字符串，其语法如下。

语法：public void SetError(Control control, string value)

说明：参数 control 表示要为其设置错误描述字符串的控件；参数 value 表示描述错误信息的字符串。

**例 10.19** 创建一个 Windows 应用程序，在窗体中添加 3 个 TextBox 控件，在每个控件的 Validating 事件中判断是否输入了数据，如果输入的数据为空，则调用 ErrorProvider 控件的 SetError 方法设置错误描述字符串，而当“订货数量”文本框中没有输入数字时，会显示错误字符串“请输入一个数”，并在此文本框的后面显示错误图标；如果输入数字，则错误图标消失，代码如下。（实例位置：光盘\mr\10\sl\10.19）

```
private int a,b,c;
private void textBox1_Validating(object sender, System.ComponentModel.CancelEventArgs e)
{
    if (textBox1.Text == "")                                //判断是否输入了商品名称
    {
        errorProvider1.SetError(textBox1, "不能为空");      //如果没有输入，则激活 errorProvider1 控件
    }
    else
    {
        errorProvider1.SetError(textBox1,"");                //errorProvider1 控件不显示消息
        a = 1;                                              //将 a 赋值为 1
    }
}
private void textBox2_Validating(object sender, System.ComponentModel.CancelEventArgs e)
{
    if (textBox2.Text == "")                                //判断是否输入了订货数量
    {
        errorProvider2.SetError(textBox2, "不能为空");      //设置 errorProvider2 的错误提示
    }
    else
    {
        try
        {
            int x = Int32.Parse(textBox2.Text);              //判断是否输入了数字，如果不是数字会出现异常
            errorProvider2.SetError(textBox2,"");            //errorProvider2 控件不显示任何错误信息
            b = 1;                                         //将 b 赋值为 1
        }
        catch
        {
            errorProvider2.SetError(textBox2, "请输入一个数");
        }
    }
}
```

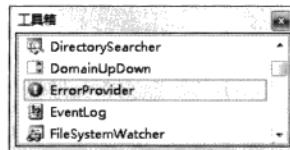


图 10.24 ErrorProvider 控件

```

        //如果出现异常，则设置 errorProvider2 控件的错误信息
    }
}
}

private void textBox3_Validating(object sender, System.ComponentModel.CancelEventArgs e)
{
    if (textBox3.Text == "")                                //判断是否输入了订货数量
    {
        errorProvider3.SetError(textBox3, "不能为空");      //设置 errorProvider3 显示的错误消息
    }
    else
    {
        errorProvider3.SetError(textBox3, "");                //errorProvider3 控件不显示任何消息
        c = 1;                                              //将 c 赋值为 1
    }
}

private void button2_Click(object sender, EventArgs e)
{
    //清空所有文本框
    textBox1.Text = "";
    textBox2.Text = "";
    textBox3.Text = "";
}

private void button1_Click(object sender, EventArgs e)
{
    if (a + b + c == 3)                                    //判断 a、b 和 c 的和是否等于 3
    {
        MessageBox.Show("数据录入成功", "提示", MessageBoxButtons.OK, MessageBoxIcon.Information);
    }
}
}

```

程序运行结果如图 10.25 所示。

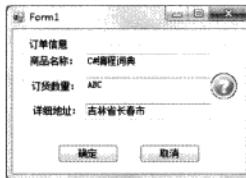


图 10.25 使用 ErrorProvider 验证文本框输入

## 10.6.2 使用 HelpProvider 控件调用帮助文件

使用 HelpProvider 控件可以将帮助文件 (.htm 文件或 .chm 文件) 与 Windows 应用程序相关联，为特定对话框或对话框中的特定控件提供区分上下文的帮助。如图 10.26 所示为 HelpProvider 控件。

通过设置 HelpProvider 控件的 HelpNamespace 属性及 SetShowHelp 方法，可以实现当按下 F1 键时打开指定的帮助文件。

HelpNamespace 属性用来设置一个值，该值指定与 HelpProvider 对象关联的帮助文件名，其语法如下。

语法：public virtual string HelpNamespace { get; set; }

说明：该属性值指定了帮助文件的名称。

SetShowHelp 方法用于指定是否显示指定控件的帮助信息，其语法如下。

语法：public virtual void SetShowHelp(Control ctl, bool value)

说明：参数 `ctl` 表示要打开或关闭帮助信息的控件，通常是指窗体；对于参数 `value`，如果显示控件的帮助信息，则为 `true`；否则为 `false`。

**例 10.20** 创建一个 Windows 应用程序，首先在程序的 bin 文件中创建一个名为 `helpPage.htm` 的帮助文件，然后设置 `HelpProvider` 控件的 `HelpNamespace` 属性为 `helpPage.htm` 文件路径，最后调用 `HelpProvider` 控件的 `SetShowHelp` 方法指定在窗体中显示帮助文件信息，代码如下。（实例位置：光盘\mr\10\sl\10.20）

```
private void Form1_Load(object sender, EventArgs e)
{
    string strPath ="helpPage.htm";           //设置帮助文件的位置
    helpProvider1.HelpNamespace = strPath;
    //设置 helpProvider1 控件的 HelpNamespace 属性，设置帮助文件的路径
    helpProvider1.SetShowHelp(this,true);      //设置 SetShowHelp 方法指定是否显示指定控件的帮助信息
}
```

程序运行结果如图 10.27 所示。

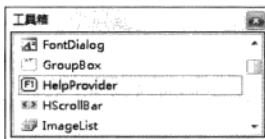


图 10.26 HelpProvider 控件



图 10.27 按 F1 键打开帮助文件

### 10.6.3 使用 Timer 控件设置时间间隔

`Timer` 控件可以定期引发事件，时间间隔的长度由其 `Interval` 属性定义，其属性值以毫秒为单位。若启用了该控件，则每个时间间隔引发一次 `Tick` 事件，开发人员可以在 `Tick` 事件中添加要执行操作的代码。如图 10.28 所示为 `Timer` 控件。

`Timer` 控件的 `Interval` 属性用于设置计时器开始计时的时间间隔，其语法如下。

语法：public int Interval { get; set; }

说明：该属性值表示计时器每次开始计时之间的间隔毫秒数，该值不小于 1。

当指定的计时器间隔已过去且计时器处于启用状态时会引发 `Timer` 控件的 `Tick` 事件，而其 `Start` 方法和 `Stop` 方法分别用来启动和停止计时器。

**例 10.21** 创建一个 Windows 应用程序，添加一个 `Timer` 控件，并在窗体加载时设置 `Timer` 控件的 `Interval` 属性为 1000 毫秒（1 秒）。然后在 `Timer` 控件的 `Tick` 事件中使文本框中显示当前的系统时间，单击 `Button` 控件，通过判断 `Button` 控件的文本值，调用 `Timer` 控件的 `Start` 方法和 `Stop` 方法启动和停止计时器。代码如下。（实例位置：光盘\mr\10\sl\10.21）

```
private void Form1_Load(object sender, EventArgs e)
{
    timer1.Interval = 1000;                      //设置 Interval 属性为 1000 毫秒
}
private void timer1_Tick(object sender, EventArgs e) //timer1 控件的 Tick 事件
{
    textBox1.Text = DateTime.Now.ToString();      //获取系统当前日期
}
```

```

}
private void button1_Click(object sender, EventArgs e)
{
    if (button1.Text == "开始")
    {
        timer1.Start(); //启动 timer1 控件
        button1.Text = "停止"; //设置按钮的 Text 属性为 "停止"
    }
    else
    {
        timer1.Stop(); //停止 timer1 控件
        button1.Text = "开始"; //设置按钮的 Text 属性为 "开始"
    }
}

```

程序运行结果如图 10.29 所示。

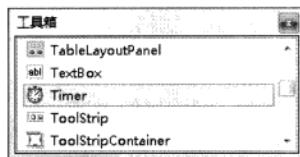


图 10.28 Timer 控件

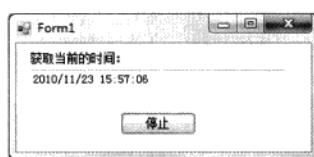


图 10.29 制作系统时钟

#### 10.6.4 使用 ProgressBar 控件显示程序运行进度条

ProgressBar 控件通过在水平放置的方框中显示适当数目的矩形块来指示工作的进度，工作完成后，进度条被填满，进度条通常用于帮助用户了解等待一项工作完成的进度。如图 10.30 所示为 ProgressBar 控件。

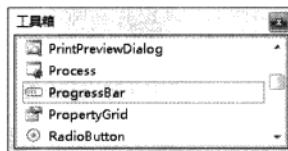


图 10.30 ProgressBar 控件

ProgressBar 控件中较为重要的属性有 Value、Minimum 和 Maximum，其中 Value 属性表示操作过程中已完成的进度；Minimum 和 Maximum 属性主要用于设置进度条的最小值和最大值。另外，还可以使用 ProgressBar 控件的 Step 属性指定 Value 属性递增的值，然后调用 PerformStep 方法来递增该值。

**例 10.22** 创建一个 Windows 应用程序，添加一个 ProgressBar 控件，并将其 Minimum 和 Maximum 属性分别设置为 0 和 500，以便确定进度条的最小值和最大值；然后设置 ProgressBar 控件的 Step 属性为 1，使其 Value 属性的递增值为 1；最后调用 PerformStep 方法递增 Step 值，使进度条不断前进，直到进度条达到最大值，代码如下。（实例位置：光盘\mr\10\sl\10.22）

```

private void button1_Click(object sender, EventArgs e)
{
    progressBar1.Value = 0; //设置进度条的初始值
    progressBar1.Minimum = 0; //设置 progressBar1 控件的 Minimum 值为 0
    progressBar1.Maximum = 500; //设置 progressBar1 控件的 Maximum 值为 500
}

```

```

progressBar1.Step = 1; //设置 progressBar1 控件的增值为 1
for (int i = 0; i < 500; i++)
{
    progressBar1.PerformStep(); //使用 PerformStep 方法按 Step 值递增
    textBox1.Text = "进度值: " + progressBar1.Value.ToString();
}
}

```

程序运行结果如图 10.31 所示。

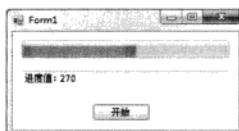


图 10.31 显示进度条

## 10.7 照猫画虎——基本功训练

### 10.7.1 基本功训练 1——在列表视图中拖动视图项

视频讲解：光盘\mr\10\lx\在列表视图中拖动视图项.exe

实例位置：光盘\mr\10\zmhh\01

本实例介绍一个有趣的功能，即在 ListView 控件中拖动视图项，移动视图项的位置。新建一个 Windows 窗体应用程序，在窗体中添加一个 ListView 控件，此控件用于显示可拖动的数据信息，程序主要代码如下。

```

private void listView1_DragDrop(object sender, DragEventArgs e)
{
    if (listView1.SelectedItems.Count == 0) //判断是否选择拖放的项
    {
        return; //退出方法
    }
    Point cp = listView1.PointToClient(new Point(e.X, e.Y)); //定义项的坐标点
    ListViewItem dragToltem = listView1.GetItemAt(cp.X, cp.Y); //得到指定位置的项
    if (dragToltem == null) //判断是否为空
    {
        return; //退出方法
    }
    List<ListViewItem> ls = new List<ListViewItem>(); //创建项集合
    foreach (ListViewItem lvi in listView1.SelectedItems) //遍历选中的项
    {
        ls.Add(lvi); //将选中项添加到集合中
    }
    for (int i = 0; i < ls.Count; i++) //移除数据项
    {
        listView1.Items.Remove(ls[i]);
    }
    for (int i = 0; i < ls.Count; i++) //遍历集合
    {

```

```

        listView1.Items.Insert(dragTotem.Index, ls[i]);
    }
    ls.Clear(); //清空数据集合
    for (int i = 0; i < listView1.Items.Count; i++)
    {
        ls.Add(listView1.Items[i]); //向数据集合中添加数据
    }
    listView1.Items.Clear(); //清空 ListView 中数据项
    for (int i = 0; i < ls.Count; i++)
    {
        listView1.Items.Add(ls[i]); //向 ListView 中添加数据项
    }
}

```

实例运行效果如图 10.32 所示。



图 10.32 在列表视图中拖动视图项

**说明：**在本实例中，设置 ListView 控件的 AllowDrop 属性值为 true，表示该控件允许拖放操作。

**熊猫画虎：**修改上面的实例，若列表视图中的项被拖出控件边界，则删除该项。提示：可以使用 ListView 控件的 DragLeave 事件来处理。（20 分）（实例位置：光盘\mr\10\zmhh\01\_zmhh）

### 10.7.2 基本功训练 2——制作带复选框的 ListView 控件

视频讲解：光盘\mr\10\lx\制作带复选框的 ListView 控件.exe

实例位置：光盘\mr\10\zmhh\02

ListView 控件中可以包含大量的数据项，用户可以对数据项进行标记。新建一个 Windows 窗体应用程序，在窗体中添加一个 ListView 控件，此控件用于显示带有复选框的文件信息；向窗体中添加两个 Button 按钮，这两个按钮分别用于全部选中、取消选中 ListView 中每一个数据项前的复选框。程序主要代码如下。

```

private void CheckBoxInListView_Load(object sender, EventArgs e)
{
    listView1.CheckBoxes = true; //设置 listView1 的复选框属性为真
    listView1.View = View.Details; //设置 listView1 的视图方式
    listView1.GridLines = true; //设置 listView1 显示网格线
    listView1.Columns.Add("文件名称", 150, HorizontalAlignment.Left); //向 listView1 中添加“文件名称”列
    listView1.Columns.Add("创建时间", 261, HorizontalAlignment.Left); //向 listView1 中添加“创建时间”列
    foreach (String fileName in Directory.GetFiles("C:\\")) //循环遍历 C 盘中的内容
    {
        FileInfo file = new FileInfo(fileName); //定义一个操作文件的实例
        ListViewItem OptionItem = new ListViewItem(file.Name); //定义一个.listView 选择项的实例
        OptionItem.SubItems.Add(file.CreationTime.ToString()); //添加文件创建时间列
    }
}

```

```

listView1.Items.Add(optionItem);
}
}

```

实例运行效果如图 10.33 所示。



图 10.33 制作带复选框的 ListView 控件

**说明：** ListView 控件的 Checkboxes 属性用来设置 ListView 控件中各数据项是否显示复选框。

**照猫画虎：**修改上面的实例，实现删除选定的项。(20 分)(实例位置：光盘\mr\10\zmhh\02\_zmhh)

### 10.7.3 基本功训练 3——使用 MaskedTextBox 控件实现输入验证

**视频讲解：**光盘\mr\10\lx\使用 MaskedTextBox 控件实现输入验证.exe

**实例位置：**光盘\mr\10\zmhh\03

新建一个 Windows 窗体应用程序，在窗体中添加 4 个 MaskedTextBox 控件，分别用来输入 18 位身份证号、15 位身份证号、邮政编码、出生日期。程序主要代码如下。

//在窗体的 Load 事件中，为 4 个 MaskedTextBox 控件设置掩码，即设置控件的 Mask 属性值

```
private void Form1_Load(object sender, EventArgs e)
{

```

```
    this.maskedTextBox1.Mask = "000000-00000000-000A";           //身份证号码 18 位
    this.maskedTextBox2.Mask = "000000-000000-000";             //身份证号码 15 位
    this.maskedTextBox3.Mask = "000000";                            //邮政编码
    this.maskedTextBox4.Mask = "0000 年 00 月 00 日";            //出生日期
}
```

实例运行效果如图 10.34 所示。



图 10.34 使用 MaskedTextBox 控件实现输入验证

**说明：**若使用 MaskedTextBox 控件实现输入验证，则需要组合使用该控件的 Mask 属性与 MaskInputRejected 事件。Mask 属性用于设置验证掩码；MaskInputRejected 事件用来提供自定义错误的处理逻辑。

**照猫画虎：**参照上面的实例，实现在 MaskedTextBox 控件中验证金额和手机号码的输入。(20 分)(实例位置：光盘\mr\10\zmhh\03\_zmhh)

### 10.7.4 基本功训练 4——使用 Timer 组件实现人物动画效果

**视频讲解：**光盘\mr\10\lx\使用 Timer 组件实现人物动画效果.exe

**实例位置：**光盘\mr\10\zmhh\04

本实例使用 Timer 组件实现人物动画效果。新建一个 Windows 应用程序，在窗体中添加一个 Timer 组件，用于间隔性地在窗体中绘制人物图像。程序主要代码如下。

```
private void tmr_Action_Tick(object sender, EventArgs e)
{
    //通过加载当前应用程序所在目录下的图片(jpg 类型)，在窗体上绘制图像
    CreateGraphics().DrawImage( Image.FromFile( index++ > 7 ? (index = 1) : index).ToString() + ".jpg",
new Point(0, 0));
}
```

实例运行效果如图 10.35 所示。



图 10.35 使用 Timer 组件实现人物动画效果

**说明：**Graphics 类的 DrawImage 方法用于在指定位置并且按原始大小绘制指定的 Image 对象。

**熊猫画虎：**使用 Timer 组件制作左右飘动的窗体。提示：通过窗体的 Left 和 Top 属性来设置窗体在桌面上的位置，使用 Timer 组件按照指定的时间间隔来更改窗体的 Left 和 Top 属性，从而实现窗体在桌面上左右飘动的效果。(20 分)(实例位置：光盘\mr\10\zmhh\04\_zmhh)

### 10.7.5 基本功训练 5——使用 ErrorProvider 组件验证文本框输入

**视频讲解：**光盘\mr\10\lx\使用 ErrorProvider 组件验证文本框输入.exe

**实例位置：**光盘\mr\10\zmhh\05

本实例使用 ErrorProvider 组件验证在文本框中的输入是否正确。新建一个 Windows 应用程序，在窗体中添加两个 ErrorProvider 组件，将其 BlinkRate 属性设置为 100，BlinkStyle 属性设置为 AlwaysBlink，该控件用于验证输入是否正确；添加两个 TextBox 控件，用于输入文本；添加两个 Button 控件，用于引发 ErrorProvider 组件验证，代码如下。

```
string strA = null;
string strB = null;
private void txtPasword_Validating(object sender, CancelEventArgs e)
{
    if (txtPasword.Text != "mrscoft")
    {
        errPassword.SetError(txtPasword, "密码错误");
    }
    //当密码输入不正确时
    //显示错误提示信息
}
```

```

else //当密码输入正确时
{
    errPassword.SetError(txtPassword, "");
    strA = txtPassword.Text;
}
}

private void txtUser_Validating(object sender, CancelEventArgs e) //验证登录名
{
    if (txtUser.Text != "mr") //当登录名输入错误时
    {
        errUser.SetError(txtUser, "登录名错误"); //显示错误的提示信息
    }
    else //当登录名输入正确时
    {
        errUser.SetError(txtUser, ""); //不显示任何内容
        strB = txtUser.Text; //得到用户名字符串
    }
}
}

```

本实例运行效果如图 10.36 所示。



图 10.36 使用 ErrorProvider 组件验证文本框输入

**说明：**ErrorProvider 组件的 SetError 方法用于将 Error 属性设置为指定的 TextWriter 对象。

**照猫画虎：**修改上面的实例，验证登录名和密码的长度都不允许超过 10。(20 分)(实例位置：光盘\mr\10\zmhh\05\_zmhh)

照猫画虎栏目分数统计：

照猫画虎题目	1	2	3	4	5	总分数	
分数							

## 10.8 情景应用——拓展与实践

### 10.8.1 情景应用 1——使用 TreeView 控件遍历磁盘目录

**视频讲解：**光盘\mr\10\lx\使用 TreeView 控件遍历磁盘目录.exe

**实例位置：**光盘\mr\10\qjyy\01

本实例将利用 TreeView 控件为用户显示磁盘目录信息，就如同在 Windows 操作系统的资源管理器的左窗格中显示文件和文件夹一样。实例运行效果如图 10.37 所示。

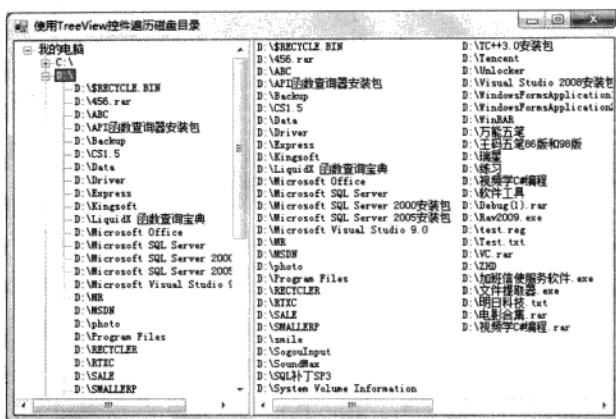


图 10.37 使用 TreeView 控件遍历磁盘目录

实现过程如下。

- (1) 打开 Visual Studio 2008 开发环境，新建一个 Windows 窗体应用程序，并将其命名为 Getdirectory。
- (2) 在默认窗体中添加一个 SplitContainer 面板，此面板用于装载 TreeView 和 ListView 控件；向窗体中添加一个 TreeView 控件，用于显示磁盘与文件夹信息；向窗体中添加一个 ListView 控件，用于显示文件信息。
- (3) 程序主要代码如下。

```
private void ListViewShow(TreeNode NodeDir) //初始化 ListView 控件
{
    ListViewFile.Clear(); //是否添加分区名称
    if (NodeDir.Parent == null)
    {
        foreach (string DrvName in Directory.GetLogicalDrives()) //获得硬盘分区名
        {
            ListViewItem ItemList = new ListViewItem(DrvName); //添加信息
            ListViewFile.Items.Add(ItemList);
        }
    }
    else
    {
        foreach (string DirName in Directory.GetDirectories((string)NodeDir.Tag)) //遍历所有目录
        {
            ListViewItem ItemList = new ListViewItem(DirName); //创建数据项
            ListViewFile.Items.Add(ItemList); //添加目录信息
        }
        foreach (string FileName in Directory.GetFiles((string)NodeDir.Tag)) //遍历所有目录的文件
        {
            ListViewItem ItemList = new ListViewItem(FileName); //创建数据项
            ListViewFile.Items.Add(ItemList); //添加文件信息
        }
    }
}
```

说明：ListView 控件中 Items 集合的 Add 方法用于将数据项添加到 ListView 数据项集合中。

**DIY：**使用 TreeView 控件遍历注册表 5 大基项的子项。提示：关于注册表的操作可查询 MSDN 的相关内容。(20 分)(实例位置：光盘\mr\10\qjyy\01\_diy)

### 10.8.2 情景应用 2——用树型列表动态显示菜单

■**视频讲解：**光盘\mr\10\lx\用树型列表动态显示菜单.exe

■**实例位置：**光盘\mr\10\qjyy\02

本实例演示将菜单中的内容动态地添加到树型列表中，并根据菜单中的用户权限，对树型列表中的相应项进行设置。实例运行效果如图 10.38 所示。

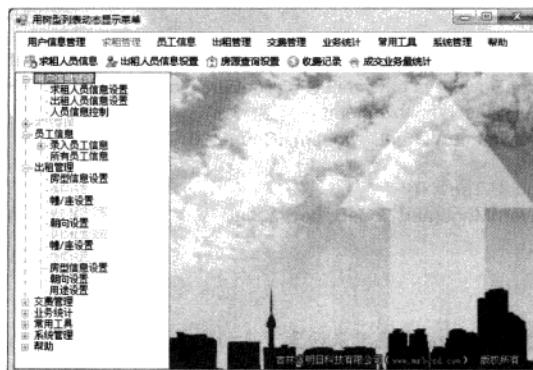


图 10.38 用树型列表动态显示菜单

实现过程如下。

- (1) 打开 Visual Studio 2008 开发环境，新建一个 Windows 窗体应用程序，并将其命名为 DisplayMenu。
- (2) 在默认窗体中添加一个 ToolStrip 控件，此控件用于在窗体中显示菜单项；向窗体中添加一个 TreeView 控件，用于显示菜单项的内容。
- (3) 程序主要代码如下。

```
public void GetCavernMenu(TreeNode newNodeA, ToolStripMenuItem newmenuA, bool BL)
{
    bool Var_Bool = true; //设置布尔值
    if (newmenuA.HasDropDownItems && newmenuA.DropDownItems.Count > 0) //遍历二级菜单项
    {
        //将二级菜单名称添加到 TreeView 组件的子节点 newNodeA 中，并设置当前节点的子节点 newNodeB
        TreeNode newNodeB = newNodeA.Nodes.Add(newmenuA.DropDownItems[0].Text);
        Var_Bool = true;
        if (BL == false) //判断一级命令是否可用
        {
            newNodeB.ForeColor = Color.Silver; //设置命令项的字体颜色
            newNodeB.Tag = 0; //标识，不显示相应的窗体
            Var_Bool = false;
        }
        else
        {
    
```

说明：TreeNode 对象的 ForeColor 属性用来设置节点的文本颜色。

**DIY:** 使用 TreeView 控件动态显示数据库中的商品信息。提示：关于数据库操作可参考本书第 11 堂课的相关内容。(20 分)(实例位置：光盘\mr\10\qivv02\div)

### 10.8.3 情景应用 3——设计一个电子万年历

 视频讲解：光盘\mr\10\x\设计一个电子万年历.exe

**案例位置:** 光盘\mr\10\qjxx\03

随着计算机的普及，人们对计算机软件的要求也越来越高，而电子万年历作为最近流行的一款工具软件，已经越来越受到用户的青睐。本节使用 C#语言制作一个电子万年历，程序运行结果如图 10.39 所示。

实现过程如下。

(1) 新建一个 Windows 应用程序，把默认窗体 Form1 的 FormBorderStyle 属性设置为 None。

(2) 在默认窗体中添加一个 MonthCalendar 控件，用来选择日期；添加一个 Label 控件，用来显示当前日期的农历表示法；添加一个 ContextMenuStrip 控件，用来作为窗体的

(3) 窗体加载时，在 Label 控件中显示当前日期的农历表示法，实现代码如下。

```
private void Form1_Load(object sender, EventArgs e)
```

{

```
string intmonth = monthCalendar1.TodayDate.Month.ToString();
string intday = monthCalendar1.TodayDate.Day.ToString();
```

//当前月份

//当前日期



图 10.39 设计一个电子万年历

```

if (monthCalendar1.TodayDate.Month < 10) //判断月份是否小于 10
{
    intmonth = "0" + monthCalendar1.TodayDate.Month.ToString(); //将月份格式化为两位
}
if (monthCalendar1.TodayDate.Day < 10) //判断天数是否小于 10
{
    intday = "0" + monthCalendar1.TodayDate.Day.ToString(); //将天数格式化为两位
}
string s = String.Format("{0}年{1}月{2}", GetStemBranch(monthCalendar1.TodayDate),
GetMonth(monthCalendar1.TodayDate), GetDay(monthCalendar1.TodayDate));
//把当前日期转化成中国传统日期
label1.Text = monthCalendar1.TodayDate.Year + "年" + intmonth + "月" + intday + "日" + " " + s + " " +
getReturnYear(monthCalendar1.TodayDate) + "年"; //在 Label 控件中显示中国传统日期
label1.ForeColor = Color.Green; //设置 Label 控件字体颜色
}

```

上面的代码中用到了 GetStemBranch、GetMonth、GetDay 和 getReturnYear 4 个自定义方法，它们的功能分别是把公历日期年份转换成农历年份的干支纪年、把公历日期月份转换成农历月份、把公历日期转换成农历天数、获取公历日期生肖年份，它们的代码详见源程序。

当用户在日历中选择窗体时，使用 ToolTip 控件弹出提示，显示选中日期的农历表示法，实现代码如下。

```

private void monthCalendar1_DateSelected(object sender, DateRangeEventArgs e)
{
    //将当期选中日期格式化为农历表示法
    string strYear = String.Format("{0}年{1}月{2}", GetStemBranch(monthCalendar1.SelectionStart),
GetMonth(monthCalendar1.SelectionStart), GetDay(monthCalendar1.SelectionStart));
    toolTip1.ToolTipTitle = monthCalendar1.SelectionStart.ToShortDateString(); //设置提示主题
    toolTip1.Show(strYear + " " + getReturnYear(monthCalendar1.SelectionStart) + "年", monthCalendar1,
monthCalendar1.Location, 5000); //显示当期日期的农历表示法
}

```

**DIY：制作日历计划任务。提示：首先在月历控件中选定未来的某个日期，然后在列表视图控件中添加对应的计划任务。（20 分）（实例位置：光盘\mr\10\qjyy\03\_diy）**

#### 10.8.4 情景应用 4——制作一个闹钟计时器

视频讲解：光盘\mr\10\lx\制作一个闹钟计时器.exe

实例位置：光盘\mr\10\qjyy\04

计时器在日常生活中经常用到，如短跑比赛需要计时等。本实例使用 C#制作了一个简单的闹钟计时器，当用户设置完定时时间之后，如果到了设置的时间，该软件会自动进行声音提示。实例运行效果如图 10.40 所示。

实现过程如下。

(1) 新建一个 Windows 窗体应用程序，在默认窗体中添加 3 个 NumericUpDown 控件，用于设置定时信息；添加两个按钮，分别用于确定和取消定时信息；添加两个 Timer 控件，分别用来实时显示系统时间和控制更新倒计时。

(2) 在设置完定时时间后单击“确定”按钮，启动 timer2 计时器，并初始化倒计时时间，显示在 Label 控件中，实现代码如下。



图 10.40 制作一个闹钟计时器

```

private void button1_Click(object sender, EventArgs e)
{
    DateTime get_time1 = Convert.ToDateTime(DateTime.Now.ToString()); // 获取当前时间
    // 获取定时时间
    DateTime sta_ontime1 = Convert.ToDateTime(Convert.ToDateTime(textBox2.Text.Trim().ToString()));
    long dat = DateAndTime.DateDiff("s", get_time1, sta_ontime1, FirstDayOfWeek.Sunday,
FirstWeekOfYear.FirstFourDays); // 计算倒计时
    if (dat > 0) // 判断倒计时是否为 0
    {
        if (timer2.Enabled != true)
        {
            timer2.Enabled = true; // 启动计时器
            label4.Text = "闹钟已启动"; // 显示倒计时
            label1.Text = "剩余" + dat.ToString() + "秒";
        }
        else
        {
            MessageBox.Show("时钟已经启动，请取消后，再启动");
        }
    }
    else
    {
        long hour = 24 * 3600 + dat; // 计算倒计时
        timer2.Enabled = true; // 启动计时器
        label4.Text = "闹钟已启动"; // 显示倒计时
        label1.Text = "剩余" + hour.ToString() + "秒";
    }
}

```

注意：在设置 Timer 组件的 Interval 属性值时，其值不能小于 1。

DIY：使用 Timer 组件实现世界杯倒计时。提示：首先设置世界杯的开幕时间，然后使用 Timer 组件实现世界杯倒计时的功能。（20 分）（实例位置：光盘\mr\10\qjyy\04\_diy）

### 10.8.5 情景应用 5——弹出模式窗口显示进度条

视频讲解：光盘\mr\10\lx\弹出模式窗口显示进度条.exe

实例位置：光盘\mr\10\qjyy\05

用户在复制文件夹时，为了便于观察文件夹的复制情况，Windows 系统提供了一个 ProgressBar 进度条。本实例在复制文件夹时，制作了一个弹出式窗口，在该窗口中将显示文件夹中各文件的复制进度。实例运行效果如图 10.41 所示。

实现过程如下。

(1) 新建一个 Windows 应用程序，在默认窗体中添加两个 TextBox 控件，分别用来显示源文件夹路径和目的文件夹路径；添加 3 个 Button 控件，分别用来打开源文件夹选择对话框、打开目的文件夹选择对话框和执行文件复制操作；添加一个 FolderBrowserDialog 控件，用来显示选择文件夹对话框。

(2) 在 ShowDialogBar 项目中添加一个 Frm\_Plan 窗体，然后在该窗体中添加一个 ProgressBar 控件。

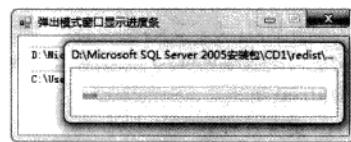


图 10.41 弹出模式窗口显示进度条

用来显示文件复制进度条。

(3) 在程序的代码中, 主要使用自定义方法 CopyFile 来实现复制文件操作, 并且该方法实现在复制文件的过程中实时显示进度条, 具体代码如下。

```

public void CopyFile(string FormerFile, string toFile, int SectSize, ProgressBar progressBar1)
{
    progressBar1.Value = 0;                                //设置进度条的当前位置为 0
    progressBar1.Minimum = 0;                             //设置进度条的最小值为 0
    FileStream fileToCreate = new FileStream(toFile, FileMode.Create); //创建目的文件, 如果已存在, 则将被覆盖
    fileToCreate.Close();                               //关闭所有资源
    fileToCreate.Dispose();                            //释放所有资源
    FormerOpen = new FileStream(FormerFile, FileMode.Open, FileAccess.Read); //以只读方式打开源文件
    ToFileOpen = new FileStream(toFile, FileMode.Append, FileAccess.Write); //以写方式打开目的文件
    //根据一次传输的大小, 计算传输的个数
    int max = Convert.ToInt32(Math.Ceiling((double)FormerOpen.Length / (double)SectSize));
    progressBar1.Maximum = max;                         //设置进度条的最大值
    int FileSize;                                     //要复制的文件的大小
    if (SectSize < FormerOpen.Length)                  //如果分段复制, 即每次复制内容小于文件总长度
    {
        byte[] buffer = new byte[SectSize];           //根据传输的大小, 定义一个字节数组
        int copied = 0;                                //记录传输的大小
        int tem_n = 1;                                //设置进度条中进度块的增加个数
        while (copied <= ((int)FormerOpen.Length - SectSize)) //复制主体部分
        {
            FileSize = FormerOpen.Read(buffer, 0, SectSize); //从 0 开始读, 每次最大读 SectSize
            FormerOpen.Flush();                           //清空缓存
            ToFileOpen.Write(buffer, 0, SectSize);        //向目的文件写入字节
            ToFileOpen.Flush();                           //清空缓存
            ToFileOpen.Position = FormerOpen.Position;   //使源文件和目的文件流的位置相同
            copied += FileSize;                          //记录已复制的大小
            progressBar1.Value = progressBar1.Value + tem_n; //增加进度条的进度块
        }
        int left = (int)FormerOpen.Length - copied;      //获取剩余大小
        FileSize = FormerOpen.Read(buffer, 0, left);     //读取剩余的字节
        FormerOpen.Flush();                           //清空缓存
        ToFileOpen.Write(buffer, 0, left);              //写入剩余的部分
        ToFileOpen.Flush();                           //清空缓存
    }
    else                                              //如果整体复制, 即每次复制内容大于文件总长度
    {
        byte[] buffer = new byte[FormerOpen.Length];   //获取文件的大小
        FormerOpen.Read(buffer, 0, (int)FormerOpen.Length); //读取源文件的字节
        FormerOpen.Flush();                           //清空缓存
        ToFileOpen.Write(buffer, 0, (int)FormerOpen.Length); //写入字节
        ToFileOpen.Flush();                           //清空缓存
    }
}

```

**DIY:** 制作一个可以调节速度的进度条。提示: 可以使用 TrackBar 控件来调节进度条的前进速度。(20 分)(实例位置: 光盘\mr\10\qjyy\05\_diy)

## 情景应用 DIY 栏目分数统计：

DIY 题目	1	2	3	4	5	总分数	
分数							

## 10.9 自我测试

## 一、选择题（每题 10 分，5 道题）

- 在 ImageList 控件的 Images 集合中可以存储大量的图像，下面哪种方法不可以移除 Images 集合中的图像（ ）。  
A. RemoveAt 方法      B. RemoveByKey 方法      C. Clear 方法      D. Dispose 方法
- ListView 控件可以通过 View 属性设置列表项在控件中的显示方式，若将每个项都显示为一个完整大小的图标，并在它的右边带项标签和子项信息，则需要设置 View 属性值为下面的那个选项（ ）。  
A. Details      B. LargeIcon      C. SmallIcon      D. Title
- 在下面的选项中，鼠标在 TreeView 控件的节点之间切换单击时，不会触发 TreeView 控件的那个事件（ ）。  
A. AfterSelect 事件      B. MouseClick 事件      C. CursorChanged 事件      D. Click 事件
- 若要在 DateTimePicker 控件中显示如“2010/11/23”格式的日期，需要设置该控件的 CustomFormat 属性值为以下哪种格式（ ）。  
A. yyyy-mm-dd      B. yyyyMMdd      C. yyyy/MM/dd      D. YYYY/MM/DD
- Timer 控件可以定期引发事件，那么可以通过该控件的（ ）属性来设置计时器开始计时的时间间隔。  
A. Tag 属性      B. Interval 属性      C. Enabled 属性      D. Modifiers 属性

## 二、填空题（每题 10 分，5 道题）

- 从 ListView 控件中移除视图项，可以使用（ ）和（ ）两种方法。
- 在 TreeView 控件中，每一个节点都是一个（ ）类型的实例。
- 在 TreeView 控件中，可在每个节点的紧挨节点文本的左侧显示图标，但在显示图标时，必须使 TreeView 控件与（ ）控件相关联。
- 通过 DateTimePicker 控件的 CustomFormat 属性可以设置自定义日期时间格式，但还需要设置该控件的（ ）属性值为（ ），这样才可以真正实现自定义格式。
- 在使用 ProgressBar 控件时，通过该控件的（ ）属性、（ ）属性来设置进度条的最大值和最小值。

## 测试分数统计：

类别	第 1 题	第 2 题	第 3 题	第 4 题	第 5 题
选择题分数					
填空题分数					
总分数					

## 10.10 行动指南

开始日期： 年 月 日

结束日期： 年 月 日

序号	内 容	行动指南	
1	照猫画虎栏目 分数( )	分数>75 分	优秀，基本功掌握得不错，加油！
		75 分>分数>50 分	及格，知识掌握得不牢，重新做一遍照猫画虎。
		分数<50 分	请使用光盘中提供的“实战能力测试系统”进行提高训练。
	情景应用栏目 分数( )	分数>75 分	优秀，综合应用能力很强。
		75 分>分数>50 分	及格，综合应用能力需提高，再练一遍情景应用。
		分数<50 分	请使用光盘中提供的“实战能力测试系统”进行提高训练。
2	自我测试栏目 分数( )	分数>75 分	优秀，有成为编程高手的潜质。
		分数<75 分	请使用光盘中提供的“实战能力测试系统”进行提高训练。
综合评价		反复训练后，以上各项分数都在 75 分以上，方可进入下一堂课学习。	
3	编程能力培养：本栏目提供了一些实践题目，对于培养编程能力很有效，要做好。	1. 使用 TreeView 控件显示公司的人员组织结构。提示：在窗体中添加一个 TreeView 控件和一个 ImageList 控件。其中，TreeView 控件用来显示部门结构；ImageList 控件用来存储 TreeView 控件中用到的图片文件。	
		2. 制作 Vista 风格的日历。提示：Vista 风格的电子日历外观与家庭常用的日历基本相似，需要显示当前系统日期的年、月、日以及当前日期所对应的甲子年和农历年等信息，可以通过调用 ChineseLunisolarCalendar 类的 GetSexagenaryYear 方法计算与指定日期对应的甲子循环中的年。	
4	编程习惯培养：本堂课培养读者在学习开发中记笔记的习惯，把开发中遇到的问题、总结的经验记录下来。		

## 10.11 成功可以复制——图文世界的缔造者约翰·沃洛克

1940 年，约翰·沃洛克（John Warnock）出生在美国犹他州的盐湖城。小时候，他并没有显示出任何过人的才华，直到读中学九年级时，代数考试仍不及格。有一次，学校组织智商测试，测试主持人询问他今后打算干什么工作？他想了想回答道：“或许，我想成为一名工程师。”主持者毫不客气地打断他的话：

“测试结果表明，在工程学领域，你成功的概率几乎是零。”幸运的是，他的数学老师帮助他在数学中找到了乐趣。之后他的数学成绩稳步上升，终于如愿地考上了犹他大学，并先后取得了数学学士学位、数学硕士学位、机电工程博士学位。

1965 年，约翰·沃洛克在犹他大学获得数学硕士学位，并开始在犹他大学计算中心工作。一次，一个正在研究图形程序的学生在开发中遇到了困难，他随意询问了正在工作的约翰·沃洛克，希望能找到一些灵感。没有想到，约翰·沃洛克提出了一种前所未有的解决方法，方法简单明了，但大学里所有的软件高手都没想到。这个小插曲改变了约翰·沃洛克的生活，他变成了学校里的软件高手，这也促使他把研究课题转到了图形处理领域。

约翰·沃洛克和同事们开发了一种定义三维数据库规则的程序——JAM 语言，在 JAM 的基础上，他又与同事开发出一种打印机标准 Interpress。约翰·沃洛克发现，Interpress 是一件很有发展潜力的产品，如果进行推广，将改变整个打印和出版业。他们想将 Interpress 发展成产品，但在其工作的施乐公司是不可能的，他们于是决定自己创业。

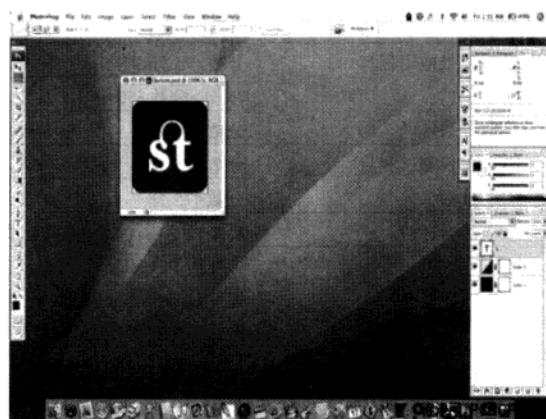
1982 年，约翰·沃洛克和 Chuck Geschke 共同创办了 Adobe 公司。如今，他已成为这个市值几百亿美元的软件企业的领袖，作为桌面出版、电子文档和图形处理软件市场先驱和领导者的 Adobe 公司，不但拥有 Photoshop、PageMaker、Acrobat Reader 软件，PDF 格式和 PostScript 语言也已是世界范围的标准。2005 年，美国《财富》杂志把 Adobe 公司评为硅谷最成功的两家公司之一。

### 经典语录

我想对孩子们说，除了我之外，爱因斯坦九年级数学考试也不及格，牛顿也是几何考试不及格。所以，考试不能说明发明者或成功者今后的前途，学生们完全可以怀疑别人告诉他们的事情。

### 深度评价

约翰·沃洛克之所以成功，是因为他能以开发改变世界的产品作为目标，并从中得到最大满足。他说：“金钱或者其他类似的东西是很好的，但这并不是他们创立 Adobe 公司的原因，金钱只是衡量经营公司的好坏程度和对市场的影响程度，开发伟大产品带来的乐趣和满足比任何其他东西给我的动力都大。”



苹果版 Adobe Photoshop CS3

# 第 11 堂课

---

## ADO.NET 数据访问技术

(  视频讲解：151分钟 )

在开发应用程序时，为了使客户端能够访问到服务器中的数据库，可以使用各种数据访问方法或技术，ADO.NET 就是这样一种技术。本堂课将对 ADO.NET 数据访问技术进行详细讲解。

学习摘要：

- ▶ 了解 ADO.NET 数据访问技术的基本概念
- ▶ 掌握数据库连接对象（Connection）
- ▶ 掌握执行 SQL 语句对象（Command）
- ▶ 掌握数据读取对象（DataReader）
- ▶ 掌握数据适配器对象（DataAdapter）
- ▶ 掌握数据集对象（DataSet）

## 11.1 ADO.NET 概述

ADO.NET 是一组向.NET 程序员公开数据访问服务的类，它为创建分布式数据共享应用程序提供了一组丰富的组件，同时它还提供了一系列的方法，用于支持对 Microsoft SQL Server 和 XML 等数据源进行访问，另外还提供了通过 OLE DB 和 XML 公开的数据源提供一致访问的方法。数据客户端应用程序可以使用 ADO.NET 来连接这些数据源，并查询、添加、删除和更新所包含的数据。

ADO.NET 支持两种访问数据的模型，即无连接模型和连接模型。无连接模型将数据下载到客户机器上，并在客户机上将数据封装到内存中，然后可以像访问本地关系数据库一样访问内存中的数据（如 DataSet）；连接模型则依赖于逐记录的访问，这种访问要求打开并保持与数据源的连接。

## 11.2 使用 Connection 对象连接数据库

### 11.2.1 Connection 对象概述

Connection 对象是一个连接对象，其主要功能是建立与物理数据库的连接。它主要包括 4 种访问数据库的对象类，也可称为数据提供程序，分别如下。

- SQL Server 数据提供程序，位于 System.Data.SqlClient 命名空间。
- ODBC 数据提供程序，位于 System.Data.Odbc 命名空间。
- OLEDB 数据提供程序，位于 System.Data.OleDb 命名空间。
- Oracle 数据提供程序，位于 System.Data.OracleClient 命名空间。

 **说明：**根据使用数据库的不同，引入不同的命名空间，然后通过命名空间中的 Connection 对象连接类连接数据库。例如，连接 SQL Server 数据库，首先要通过 using System.Data.SqlClient 命令引用 SQL Server 数据提供程序，然后才能调用空间下的 SqlConnection 类连接数据库。

### 11.2.2 连接数据库

以 SQL Server 数据库为例，如果要连接 SQL Server 数据库，必须使用 System.Data.SqlClient 命名空间下的 SqlConnection 类，所以首先要通过 using System.Data.SqlClient 命令引用命名空间。连接数据库之后，通过调用 SqlConnection 对象的 Open 方法打开数据库，另外，还可以通过 SqlConnection 对象的 State 属性判断数据库的连接状态。

语法：public override ConnectionState State { get; }

说明：属性值为 ConnectionState 枚举值之一， ConnectionState 枚举的值及说明如表 11.1 所示。

表 11.1 ConnectionState 枚举的值及说明

枚举值	说明
Broken	与数据源的连接中断。只有在连接打开之后才可能发生这种情况。可以关闭处于这种状态的连接，然后重新打开
Closed	连接处于关闭状态
Connecting	连接对象正在与数据源连接

续表

枚举值	说    明
Executing	连接对象正在执行命令
Fetching	连接对象正在检索数据
Open	连接处于打开状态

**例 11.01** 创建一个 Windows 应用程序，在 Form1 窗体中添加一个 TextBox 控件、一个 Button 控件和一个 Label 控件，分别用于输入要连接的数据库名称、执行连接数据库的操作以及显示数据库的连接状态，然后引入 System.Data.SqlClient 命名空间，使用 SqlConnection 类连接数据库，代码如下。（实例位置：光盘\mr\11\sl\11.01）

```

private void button1_Click(object sender, EventArgs e)
{
    if (textBox1.Text == "")
        {
            MessageBox.Show("请输入要连接的数据库名称");
        }
    else
    {
        try //调用 try...catch 语句
        {
            //声明一个字符串用于存储连接数据库字符串
            string ConStr = "server=MRWXK\WANGXIAOKE;database=" + textBox1.Text.Trim() + ";uid=sa;pwd=";

            SqlConnection conn = new SqlConnection(ConStr);           //创建一个 SqlConnection 对象
            conn.Open();                                         //打开连接
            if (conn.State == ConnectionState.Open)               //判断当前连接的状态
            {
                label2.Text = "数据库【" + textBox1.Text.Trim() + "】已经连接并打开"; //显示状态信息
            }
        }
        catch
        {
            MessageBox.Show("连接数据库失败");                  //出现异常，弹出提示
        }
    }
}

```

程序运行结果如图 11.1 所示。

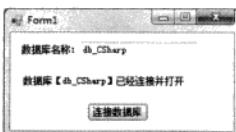


图 11.1 连接数据库

### 11.2.3 关闭连接

当对数据库操作完毕后，要关闭与数据库的连接，释放占用的资源，可以通过调用 SqlConnection 对象

的 Close 方法或 Dispose 方法来实现，这两种方法的主要区别是：Close 方法用于关闭一个连接，而 Dispose 方法不仅关闭一个连接，而且还清理连接所占用的资源。当使用 Close 方法关闭连接后，可以再调用 Open 方法打开连接，不会产生任何错误；而如果使用 Dispose 方法关闭连接，就不可以再次直接用 Open 方法打开连接，而必须重新初始化连接之后再打开。

**例 11.02** 创建一个 Windows 应用程序，首先向 Form1 窗体中添加一个 TextBox 控件和一个 RichTextBox 控件，分别用于输入连接的数据库名称和显示连接信息及错误提示；然后再添加 3 个 Button 控件，分别用于连接数据库、调用 Close 方法关闭连接再调用 Open 方法打开连接以及调用 Dispose 方法关闭并释放连接再调用 Open 方法打开连接，代码如下。（实例位置：光盘\mr\11\sl\11.02）

```

SqlConnection conn; //声明一个 SqlConnection 对象
private void button1_Click(object sender, EventArgs e)
{
    if (textBox1.Text == "") //判断是否输入数据库名称
    {
        MessageBox.Show("请输入数据库名称"); //如果没有输入，则弹出提示
    }
    else
    {
        try
        {
            //建立连接数据库字符串
            string str = "server=MRWXK\WANGXIAOKE;database=" + textBox1.Text.Trim() + ";uid=sa;pwd=";
            conn = new SqlConnection(str); //创建一个 SqlConnection 对象
            conn.Open(); //打开连接
            if (conn.State == ConnectionState.Open) //判断当前连接状态
            {
                MessageBox.Show("连接成功"); //弹出提示
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message); //出现异常，弹出错误信息
            textBox1.Text = ""; //清空文本框
        }
    }
}
private void button2_Click(object sender, EventArgs e)
{
    try
    {
        string str = ""; //声明一个字符串变量
        conn.Close(); //使用 Close 方法关闭连接
        if (conn.State == ConnectionState.Closed) //判断当前连接是否关闭
        {
            str = "数据库已经成功关闭\n"; //如果关闭，则弹出提示
        }
        conn.Open(); //重新打开连接
        if (conn.State == ConnectionState.Open) //判断连接是否打开
    }
}

```

```

        {
            str += "数据库已经成功打开\n";
            //弹出提示
        }
        richTextBox1.Text = str;
        //向 richTextBox1 中添加提示信息
    }
    catch (Exception ex)
    {
        richTextBox1.Text = ex.Message;
        //出现异常，将异常添加到 richTextBox1 中
    }
}

private void button3_Click(object sender, EventArgs e)
{
    try
    {
        conn.Dispose();
        conn.Open();
    }
    catch (Exception ex)
    {
        richTextBox1.Text = ex.Message;
    }
}

```

程序运行结果如图 11.2 和图 11.3 所示。



图 11.2 调用 Close 方法关闭连接

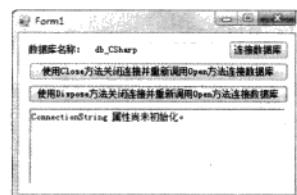


图 11.3 调用 Dispose 方法关闭并释放连接

## 11.3 使用 Command 对象执行 SQL 语句

### 11.3.1 Command 对象概述

Command 对象是一个数据命令对象，主要功能是向数据库发送查询、更新、删除、修改操作的 SQL 语句。Command 对象主要有以下几种方式。

- SqlCommand**: 用于向 SQL Server 数据库发送 SQL 语句，位于 System.Data.SqlClient 命名空间。
- OleDbCommand**: 用于向使用 OLEDB 公开的数据库发送 SQL 语句，位于 System.Data.OleDb 命名空间。例如，Access 数据库和 MySQL 数据库都是 OLEDB 公开的数据库。
- OdbcCommand**: 用于向 ODBC 公开的数据库发送 SQL 语句，位于 System.Data.Odbc 命名空间。若有些数据库没有提供相应的连接程序，则可以配置好 ODBC 连接后再使用 OdbcCommand。
- OracleCommand**: 用于向 Oracle 数据库发送 SQL 语句，位于 System.Data.OracleClient 命名空间。

**◆ 注意：**在使用 OracleCommand 向 Oracle 数据库发送 SQL 语句时，需要引入 System.Data.OracleClient 命名空间。但在默认情况下没有此命名空间，此时就需要将程序集 System.Data.OracleClient 引入到项目中。引入程序集的方法是在项目名称上单击鼠标右键，在弹出的快捷菜单中选择“添加引用”命令，打开“添加引用”对话框。在该对话框中选择 System.Data.OracleClient 程序集，单击“确定”按钮，即可将其添加到项目中。

### 11.3.2 设置数据源类型

Command 对象有 3 个重要的属性，分别是 Connection 属性、CommandText 属性和 CommandType 属性。Connection 属性用于设置 SqlCommand 使用的 SqlConnection 连接对象；CommandText 属性用于设置要对数据源执行的 SQL 语句或存储过程；CommandType 属性用于设置指定 CommandText 的类型。CommandType 属性的值是 CommandType 枚举值，CommandType 枚举有 3 个枚举成员，分别如下。

- StoredProcedure:** 存储过程的名称。
- TableDirect:** 表的名称。
- Text:** SQL 文本命令。

如果要设置数据源的类型，则可以通过设置 CommandType 属性来实现。下面通过一个实例演示如何使用 Command 对象的 3 个属性以及如何设置数据源类型。

**例 11.03** 创建一个 Windows 应用程序，向 Form1 窗体中添加一个 Button 控件、一个 TextBox 控件和一个 Label 控件，分别用于执行 SQL 语句、输入要查询的数据表名称和显示数据表中数据的数量，单击“查询数据表中数据”按钮，通过使用 Command 对象获取指定数据表中共有多少条数据，代码如下。（实例位置：光盘\mr\11\sl\11.03）

```

SqlConnection conn; //声明一个 SqlConnection 对象
private void Form1_Load(object sender, EventArgs e)
{
    //创建 SqlConnection 对象
    conn = new SqlConnection("server=MRWXK\WANGXIAOKE;database=db_CSharp;uid=sa;pwd=");
    conn.Open(); //打开连接
}
private void button1_Click(object sender, EventArgs e)
{
    try
    {
        if (conn.State == ConnectionState.Open || textBox1.Text != "") //判断是否打开连接或文本框不为空
        {
            SqlCommand cmd = new SqlCommand(); //创建一个 SqlCommand 对象
            cmd.Connection = conn; //设置 Connection 属性
            cmd.CommandText = "select count(*) from " + textBox1.Text.Trim(); //设置 CommandText 属性，设置 SQL 语句
            //设置 CommandType 属性为 Text，使其只执行 SQL 语句文本形式
            cmd.CommandType = CommandType.Text;
            //使用 ExecuteScalar 方法获取指定数据表中的数据数量
            int i = Convert.ToInt32(cmd.ExecuteScalar());
            label2.Text = "数据表中共有：" + i.ToString() + "条数据";
        }
    }
    catch (Exception ex)
}

```

```

    {
        MessageBox.Show(ex.Message);
    }
}

```

程序运行结果如图 11.4 所示。

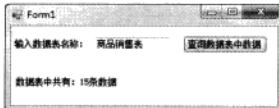


图 11.4 SqlCommand 对象执行查询语句

### 11.3.3 执行 SQL 语句

Command 对象提供了多种执行 SQL 语句的方法，下面以 SqlCommand 对象为例，介绍几种常用的执行 SQL 语句的方法。

#### 1. ExecuteNonQuery 方法

ExecuteNonQuery 方法执行 SQL 语句，并返回受影响的行数，在使用 SqlCommand 对象向数据库发送增、删、改命令时，通常使用该方法执行发送的 SQL 语句。

语法：public override int ExecuteNonQuery()

说明：它的返回值表示受影响的行数。

**例 11.04** 创建一个 Windows 应用程序，在“三八”妇女节这天，公司决定为每位女员工颁发奖金 50 元。这样，就需要向数据库发送更新命令，将数据库中所有女员工的奖金数额加上 50，这里使用 ExecuteNonQuery 方法执行发送的 SQL 语句，并获取受影响的行数，代码如下。（实例位置：光盘\mr\11\s\11.04）

```

SqlConnection conn;
private void button1_Click(object sender, EventArgs e)
{
    //创建 SqlConnection 变量 conn
    conn = new SqlConnection("server=MRWXK\WANGXIAOKE;database=db_CSharp;uid=sa;pwd=");
    conn.Open();                                         //打开连接
    SqlCommand cmd = new SqlCommand();                  //创建一个 SqlCommand 对象
    //设置 Connection 属性，指定其使用 conn 连接数据库
    cmd.Connection = conn;
    //设置 CommandText 属性，设置其执行的 SQL 语句
    cmd.CommandText = "update tb_command set 奖金=50 where 性别='女'";
    //设置 CommandType 属性为 Text，使其只执行 SQL 语句文本形式
    cmd.CommandType = CommandType.Text;
    int i = Convert.ToInt32(cmd.ExecuteNonQuery());      //使用 ExecuteNonQuery 方法执行 SQL 语句
    label2.Text = "共有" + i.ToString() + "名女员工获得奖金";
}

```

程序运行结果如图 11.5 所示。

#### 2. ExecuteReader 方法

ExecuteReader 方法执行 SQL 语句，并生成一个包含数据的 SqlDataReader 对象的实例。

语法：public SqlDataReader ExecuteReader()

说明：它的返回值为一个 SqlDataReader 对象。

**例 11.05** 创建一个 Windows 应用程序，使用 select \* from tb\_command 语句对数据库进行查询，并调用 SqlCommand 对象的 ExecuteReader 方法返回一个包含 tb\_command 表中所有数据的 SqlDataReader 对象，同时循环遍历该 SqlDataReader 对象，得到所有的员工姓名，代码如下。（实例位置：光盘\mr\11\sl\11.05）

```

SqlConnection conn;
private void button1_Click(object sender, EventArgs e)
{
    //创建 SqlConnection 变量 conn
    conn = new SqlConnection("server=MRWXK\WANGXIAOKE;database=db_CSharp;uid=sa;pwd=");
    conn.Open(); //打开连接
    //创建一个 SqlCommand 对象
    SqlCommand cmd = new SqlCommand();
    //设置 Connection 属性，指定其使用 conn 连接数据库
    cmd.Connection = conn;
    //设置 CommandText 属性，设置其执行的 SQL 语句
    cmd.CommandText = "select * from tb_command";
    //设置 CommandType 属性为 Text，使其只执行 SQL 语句文本形式
    cmd.CommandType = CommandType.Text;
    //使用 ExecuteReader 方法创建一个 SqlDataReader 对象
    SqlDataReader sdr = cmd.ExecuteReader();
    while (sdr.Read())
        //调用 while 语句，读取 SqlDataReader
        listView1.Items.Add(sdr[1].ToString()); //将内容添加到 listView1 控件中
    }
    conn.Dispose(); //释放连接
    button1.Enabled = false; //禁用按钮
}

```

程序运行结果如图 11.6 所示。

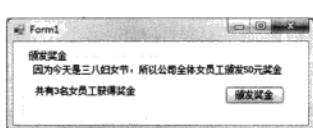


图 11.5 对数据表执行更新操作

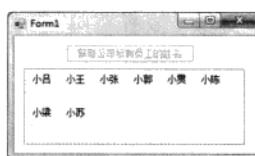


图 11.6 获取员工姓名

### 3. ExecuteScalar 方法

ExecuteScalar 方法执行 SQL 语句，并返回结果集中的第一行的第一列。

语法：public override Object ExecuteScalar()

说明：它的返回值表示结果集中第一行的第一列或空引用（如果结果集为空）。

**说明：** ExecuteScalar 方法的使用在例 11.03 中已经做过详细介绍，此处就不再赘述，需要注意的是，该方法通常与聚合函数一起使用，常见的聚合函数如表 11.2 所示。

表 11.2 常见的聚合函数及说明

聚 合 函 数	说 明
AVG(expr)	列平均值，该列只能包含数字数据
COUNT(expr)、COUNT(*)	列值的计数（如果将列名指定为 expr）或是表或分组中所有行的计数（如果指定*）忽略空值，但 COUNT(*) 在计数中包含空值

续表

聚合函数	说 明
MAX(expr)	列中最大值（文本数据类型中按字母顺序排在最后的值），忽略空值
MIN(expr)	列中最小值（文本数据类型中按字母顺序排在最前的值），忽略空值
SUM(expr)	列值的合计，该列只能包含数字数据

## 11.4 使用 DataReader 对象读取数据

### 11.4.1 DataReader 对象概述

DataReader 对象是数据读取器对象，它提供只读向前的游标，如果应用程序需要每次从数据库中取出最新的数据，或者只是需要快速读取数据，而并不需要修改数据，那么就可以使用 DataReader 对象进行读取。对于不同的数据库连接，有不同的 DataReader 类型。

- ☒ 在 System.Data.SqlClient 命名空间下可以调用 SqlDataReader 类。
- ☒ 在 System.Data.OleDb 命名空间下可以调用 OleDbDataReader 类。
- ☒ 在 System.Data.Odbc 命名空间下可以调用 OdbcDataReader 类。
- ☒ 在 System.Data.Oracle 命名空间下可以调用 OracleDataReader 类。

使用 DataReader 对象读取数据时，可以使用 SqlCommand 对象的 ExecuteReader 方法，根据 SQL 语句的结果创建一个 SqlDataReader 对象。

**例 11.06** 使用 SqlCommand 对象的 ExecuteReader 方法，创建一个读取 tb\_command 表中所有数据的 SqlDataReader 对象，代码如下。

```
conn = new SqlConnection("server=MRWXK\WANGXIAOKE;database=db_Csharp;uid=sa;pwd="); //连接数据库
conn.Open(); //打开数据库
SqlCommand cmd = new SqlCommand(); //创建 SqlCommand 对象
cmd.Connection = conn; //设置对象的连接
cmd.CommandText = "select * from tb_command"; //设置 SQL 语句
cmd.CommandType = CommandType.Text; //设置以文本形式执行 SQL 语句
//使用 ExecuteReader 方法创建 SqlDataReader 对象
SqlDataReader sdr = cmd.ExecuteReader();
```

### 11.4.2 判断查询结果中是否有值

可以通过 SqlDataReader 对象的 HasRows 属性获取一个值，该值指示 SqlDataReader 是否包含一行或多行，即判断查询结果中是否有值。

语法：public override bool HasRows { get; }

说明：它的属性值表示，如果 SqlDataReader 包含一行或多行，则为 true；否则为 false。

**例 11.07** 创建一个 Windows 应用程序，向 Form1 窗体中添加一个 TextBox 控件和一个 Button 控件，分别用于输入要查询的表名以及执行查询操作。然后通过 SqlDataReader 对象的 HasRows 属性进行判断，如果 SqlDataReader 包含一行或多行，则为 true；否则为 false，代码如下。（实例位置：光盘\mr\11\s\11.07）

```
private void button1_Click(object sender, EventArgs e)
{
    try
```

```

{
    //创建 SqlConnection 变量 conn
    SqlConnection conn = new SqlConnection("server=MRWXK\WANGXIAOKE;database=db_CSharp;
    uid=sa;pwd=");
    conn.Open();                                //打开连接
    //创建一个 SqlCommand 对象
    SqlCommand cmd = new SqlCommand("select * from " + textBox1.Text.Trim(), conn);
    //使用 ExecuteReader 方法创建 SqlDataReader 对象
    SqlDataReader sdr = cmd.ExecuteReader();
    sdr.Read();                                 //调用 Read 方法读取 SqlDataReader
    if (sdr.HasRows)                           //使用 HasRows 属性判断结果中是否有数据
    {
        MessageBox.Show("数据表中有值");          //弹出提示信息
    }
    else
    {
        MessageBox.Show("数据表中没有任何数据");
    }
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message);
}
}

```

程序运行结果如图 11.7 所示。



图 11.7 判断指定的数据表中是否有值

### 11.4.3 读取数据

如果要读取数据表中的数据，可以使用 SqlCommand 对象的 ExecuteReader 方法创建一个 SqlDataReader 对象，再调用 SqlDataReader 对象的 Read 方法读取数据。Read 方法使 SqlDataReader 前进到下一条记录，SqlDataReader 的默认位置在第一条记录前面，因此必须调用 Read 方法访问数据。

语法：public override bool Read()

说明：它的返回值表示，如果存在多个行，则为 true；否则为 false。

**注意：**对于每个关联的 SqlConnection，一次只能打开一个 SqlDataReader，在第一个关闭之前，打开另一个的任何尝试都将失败。

在使用完 SqlDataReader 对象后，要使用 Close 方法将其关闭。

语法：public override void Close()

**例 11.08** 关闭 SqlDataReader 对象，代码如下。

```
//创建 SqlConnection 变量 conn
SqlConnection conn = new SqlConnection("server=MRWXK\WANGXIAOKE;database=db_CSharp;uid=sa;pwd=");
```

```

conn.Open(); //打开连接
SqlCommand cmd = new SqlCommand("select * from "+textBox1.Text.Trim(), conn); //创建一个 SqlCommand 对象
//使用 ExecuteReader 方法创建 SqlDataReader 对象
SqlDataReader sdr = cmd.ExecuteReader();
sdr.Close();

```

 注意：在使用 SqlDataReader 对象之前必须打开数据库连接。如果针对一个 SqlConnection 创建多个 SqlDataReader 对象，在创建下一个 SqlDataReader 对象之前，要通过 Close 方法关闭上一个 SqlDataReader 对象。

## 11.5 数据适配器：DataAdapter 对象

### 11.5.1 DataAdapter 对象概述

DataAdapter 对象是一个数据适配器对象，它是 DataSet 与数据源之间的桥梁。DataAdapter 对象提供了 4 个属性，以实现与数据源之间的互通，具体如下。

- SelectCommand 属性：向数据库发送查询 SQL 语句。
- DeleteCommand 属性：向数据库发送删除 SQL 语句。
- InsertCommand 属性：向数据库发送插入 SQL 语句。
- UpdateCommand 属性：向数据库发送更新 SQL 语句。

在对数据库进行操作时，只要将上述 4 个属性设置成相应的 SQL 语句即可。另外，DataAdapter 对象中还有几个主要的方法，具体如下。

#### (1) Fill 方法

填充 DataSet 数据集。

语法：public int Fill(DataSet dataSet, string srcTable)

说明：dataSet 指定要用记录和架构（如果必要）填充的 DataSet；srcTable 用于表示表映射的源表的名称。另外，它的返回值表示已在 DataSet 中成功添加或刷新的行数，这不包括受不返回行的语句影响的行。

#### (2) Update 方法

更新数据库，这时 DataAdapter 将调用 DeleteCommand、InsertCommand 及 UpdateCommand 属性。

语法：public int Update(DataTable dataTable)

说明：dataTable 表示用于更新数据源的 DataTable。它的返回值表示 DataTable 中成功更新的行数。

例如，使用 DataAdapter 对象的 Fill 方法从数据源中提取数据并填充到 DataSet 时，就会用到 SelectCommand 属性中设置的命令对象。

### 11.5.2 填充 DataSet 数据集

通过 DataAdapter 对象的 Fill 方法可以填充 DataSet 数据集，该方法使用 Select 语句从数据源中检索数据，这里需要注意，与 Select 命令关联的 Connection 对象必须有效，但不需要将其打开，因为 DataAdapter 对象会自动打开和关闭数据库。

**例 11.09** 创建一个 Windows 应用程序，向 Form1 窗体中添加一个 Button 控件和一个 DataGridView 控件，分别用于执行数据绑定以及显示数据表中的数据。当单击 Button 控件后，程序首先连接数据库，然后

创建一个 SqlDataAdapter 对象，使用该对象的 Fill 方法填充 DataSet 数据集，最后设置 DataGridView 控件的数据源，显示查询的数据，代码如下。（实例位置：光盘\mr\11\sl\11.09）

```
SqlConnection conn;
private void button1_Click(object sender, EventArgs e)
{
    //创建 SqlConnection 变量 conn
    conn = new SqlConnection("server=MRWXK\WANGXIAOKE;database=db_CSharp;uid=sa;pwd=");
    //创建一个 SqlCommand 对象
    SqlCommand cmd = new SqlCommand("select * from tb_command", conn);
    SqlDataAdapter sda = new SqlDataAdapter(); //创建一个 SqlDataAdapter 对象
    sda.SelectCommand = cmd; //设置 SqlDataAdapter 对象的 SelectCommand 属性为 cmd
    DataSet ds = new DataSet(); //创建一个 DataSet 对象
    sda.Fill(ds, "tb_command"); //使用 SqlDataAdapter 对象的 Fill 方法填充 DataSet 数据集
    dataGridView1.DataSource = ds.Tables[0]; //设置 dataGridView1 控件的数据源
}
```

程序运行结果如图 11.8 所示。

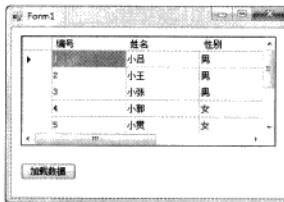


图 11.8 使用 Fill 方法填充 DataSet 数据集

### 11.5.3 更新数据源

使用 DataAdapter 对象的 Update 方法可以将 DataSet（在 11.6 节中将会详细介绍 DataSet 对象）中修改过的数据及时更新到数据库中。在调用 Update 方法之前，要创建一个 CommandBuilder 类，该类能自动根据 DataAdapter 的 SelectCommand 属性指定的 SQL 语句判断其他的 InsertCommand、UpdateCommand 和 DeleteCommand，这样就不用设置 DataAdapter 的 InsertCommand、UpdateCommand 和 DeleteCommand 属性，而是直接使用 DataAdapter 的 Update 方法来更新 DataSet、DataTable 或 DataRow 数组。

**例 11.10** 创建一个 Windows 应用程序，查询 tb\_command 表中的所有数据并显示在 DataGridView 控件中，单击某条数据，会显示其详细信息。当对某条数据进行修改后单击“修改”按钮，使用 DataAdapter 对象的 Update 方法更新数据源，代码如下。（实例位置：光盘\mr\11\sl\11.10）

```
SqlConnection conn; //声明一个 SqlConnection 变量
DataSet ds; //声明一个 DataSet 变量
SqlDataAdapter sda; //声明一个 SqlDataAdapter 变量
private void Form1_Load(object sender, EventArgs e)
{
    //创建 SqlConnection 变量 conn，连接数据库
    conn = new SqlConnection("server=MRWXK\WANGXIAOKE;database=db_CSharp;uid=sa;pwd=");
    //创建一个 SqlCommand 对象
    SqlCommand cmd = new SqlCommand("select * from tb_command", conn);
    sda = new SqlDataAdapter(); //创建 SqlDataAdapter 对象
    sda.SelectCommand = cmd; //设置 SqlDataAdapter 对象的 SelectCommand 属性为 cmd
```

```

ds = new DataSet();
sda.Fill(ds, "tb_command"); //创建 DataSet 对象
dataGridView1.DataSource = ds.Tables[0]; //使用 SqlDataAdapter 对象的 Fill 方法填充 DataSet
//设置 dataGridView1 控件的数据源
}

private void button1_Click(object sender, EventArgs e)
{
    DataTable dt = ds.Tables["tb_command"];
    sda.FillSchema(dt, SchemaType.Mapped); //创建一个 DataTable
    DataRow dr = dt.Rows.Find(txtNo.Text); //把表结构加载到 tb_command 表中
    //创建一个 DataRow
    //设置 DataRow 中的值
    dr["姓名"] = txtName.Text.Trim(); //设置姓名
    dr["性别"] = txtSex.Text.Trim(); //设置性别
    dr["年龄"] = txtAge.Text.Trim(); //设置年龄
    dr["奖金"] = txtJJ.Text.Trim(); //设置奖金
    //创建一个 SqlCommandBuilder
    SqlCommandBuilder cmdbuilder = new SqlCommandBuilder(sda);
    //调用其 Update 方法将 DataTable 更新到数据库中
    sda.Update(dt);
}

private void dataGridView1_CellClick(object sender, DataGridViewCellEventArgs e)
{
    //在 dataGridView1 控件的 CellClick 事件中实现单击某条数据显示详细信息
    txtNo.Text = dataGridView1.SelectedCells[0].Value.ToString();
    txtName.Text = dataGridView1.SelectedCells[1].Value.ToString();
    txtSex.Text = dataGridView1.SelectedCells[2].Value.ToString();
    txtAge.Text = dataGridView1.SelectedCells[3].Value.ToString();
    txtJJ.Text = dataGridView1.SelectedCells[4].Value.ToString();
}
}

```

 注意：在使用 DataTable 对象的 Rows.Find 方法时，所操作的表必须设置主键。

程序运行结果如图 11.9 所示。

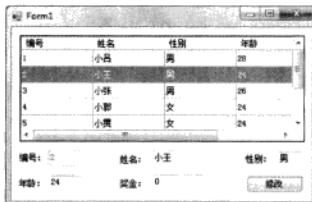


图 11.9 更新数据源

## 11.6 数据集：DataSet 对象

### 11.6.1 DataSet 对象概述

DataSet 对象就像是存放于内存中的一个小型数据库，它可以包含数据表、数据列、数据行、视图、约束及关系等。通常，DataSet 的数据来源于数据库或 XML，为了从数据库中获取数据，需要使用数据适配器

(DataAdapter) 从数据库中得到数据。

 注意：关于数据适配器（DataAdapter）在第 11.5 节已经做过介绍，此处不再赘述。

**例 11.11** 使用数据适配器（DataAdapter）从数据库中查询数据，并调用其 Fill 方法填充 DataSet 对象，代码如下。

```
conn = new SqlConnection("server=MRWXK\WANGXIAOKE;database=db_CSharp;uid=sa;pwd="); //连接数据库
DataSet ds = new DataSet(); //创建一个 DataSet 对象
SqlDataAdapter sda = new SqlDataAdapter("select * from tb_test", conn); //创建一个 SqlDataAdapter 对象
sda.Fill(ds); //使用 Fill 方法填充 DataSet 对象
```

## 11.6.2 合并 DataSet 内容

使用 DataSet 的 Merge 方法可以将 DataSet、DataTable 或 DataRow 中的内容并入现有 DataSet，该方法主要用来将指定的 DataSet 及其架构与当前的 DataSet 进行合并，在此过程中，将根据给定的参数保留或放弃在当前 DataSet 中的更改并处理不兼容的架构。

语法：public void Merge(DataSet dataSet, bool preserveChanges, MissingSchemaAction missingSchemaAction)

说明：dataSet 表示其数据和架构将被合并的 DataSet；preserveChanges 表示若要保留当前 DataSet 中的更改，则为 true，否则为 false，MissingSchemaAction 表示 MissingSchemaAction 枚举值之一。

MissingSchemaAction 枚举成员及说明如表 11.3 所示。

表 11.3 MissingSchemaAction 枚举成员及说明

枚举值	说明
Add	添加必需的列以完成架构
AddWithKey	添加必需的列和主键信息以完成架构，用户可以在每个 DataTable 上显式设置主键约束。这将确保对与现有记录匹配的传入记录进行更新，而不是追加
Error	如果缺少指定的列映射，则生成 InvalidOperationException
Ignore	忽略额外列

**例 11.12** 创建一个 Windows 应用程序，向 Form1 窗体中添加一个 DataGridView 控件。首先获取数据表 tb\_test 中的数据，并存储在 DataSet 对象 ds 中，然后再获取数据表 tb\_man 中的数据，存储在另一个 DataSet 对象 ds1 中，最后调用 DataSet 对象的 Merge 方法将 ds 与 ds1 合并，并将合并后的 DataSet 作为 DataGridView 控件的数据源。代码如下。（实例位置：光盘\mr\11\sl\11.12）

```
SqlConnection conn;
private void Form1_Load(object sender, EventArgs e)
{
    //创建 SqlConnection 对象 conn，连接数据库
    conn = new SqlConnection("server=MRWXK\WANGXIAOKE;database=db_CSharp;uid=sa;pwd=");
    //创建两个 DataSet 对象
    DataSet ds = new DataSet();
    DataSet ds1 = new DataSet();
    SqlDataAdapter sda = new SqlDataAdapter("select * from tb_test", conn); //创建一个 SqlDataAdapter 对象
    sda.Fill(ds); //使用 Fill 方法填充 DataSet
    SqlDataAdapter sda1 = new SqlDataAdapter("select * from tb_man", conn); //创建一个 SqlDataAdapter 对象
    SqlCommandBuilder sbl = new SqlCommandBuilder(sda1); //创建一个 SqlCommandBuilder 对象
    sda1.Fill(ds1); //使用 Fill 方法填充 DataSet
    ds1.Merge(ds, true, MissingSchemaAction.AddWithKey); //使用 Merge 方法将 da 合并到 ds1 中
```

```

    dataGridView1.DataSource = ds1.Tables[0]; //设置 dataGridView1 控件的数据源
}
}

```

程序运行结果如图 11.10 所示。

职工年龄	职工原籍	职工工资	商品名称	商品价格	商品类型
24	吉林省	1000			
24	吉林省	1000			
24	吉林省	1000			
27	吉林省	1000			
28	吉林省	1000			
			电动自行车	300	交通工具
			手机	1300	家电
			电脑	9000	家电
			背包	350	服饰
			MP4	299	家电

图 11.10 合并 DataSet

### 11.6.3 复制 DataSet 内容

为了在不影响原始数据的情况下使用数据或使用 DataSet 中数据的子集，可以创建 DataSet 的副本。在复制 DataSet 时，可以指定以下内容。

- 创建 DataSet 的原样副本，其中包含架构、数据、行状态信息和行版本。
- 创建包含现有 DataSet 的架构但仅包含已修改行的 DataSet。可以返回已修改的所有行或指定特定的 DataRowState。有关行状态的更多信息，可参见行状态与行版本。
- 仅复制 DataSet 的架构(即关系结构)，而不复制任何行。可以使用 ImportRow 将行导入现有 DataSet 数据集的 DataTable 中。

使用 DataSet 对象的 Copy 方法可以创建包含其架构和数据的 DataSet 的原样副本，该方法的作用是复制指定 DataSet 的结构和数据。

语法：public DataSet Copy()

说明：它的返回值表示新的 DataSet，具有与该 DataSet 相同的结构（表架构、关系和约束）和数据。

**例 11.13** 创建一个 Windows 应用程序，向 Form1 窗体中添加两个 DataGridView 控件和一个 Button 控件，其中第一个 DataGridView 控件用于显示数据表 tb\_test 中的数据，当单击 Button 控件后，通过 DataSet 对象的 Copy 方法复制第一个 DataGridView 控件的 DataSet 数据源，并作为第二个 DataGridView 控件的数据源。代码如下。（实例位置：光盘\mr\11\s\11.13）

```

SqlConnection conn; //声明一个 SqlConnection 变量
DataSet ds; //声明一个 DataSet 变量
private void Form1_Load(object sender, EventArgs e)
{
    //创建 SqlConnection 变量 conn，连接数据库
    conn = new SqlConnection("server=MRWXK\WANGXIAOKE;database=db_CSharp;uid=sa;pwd=");
    SqlCommand cmd = new SqlCommand("select * from tb_test", conn); //创建一个 SqlCommand 对象
    SqlDataAdapter sda = new SqlDataAdapter(); //创建一个 SqlDataAdapter 对象
    //设置 SqlDataAdapter 对象的 SelectCommand 属性，设置执行的 SQL 语句
    sda.SelectCommand = cmd;
    ds = new DataSet(); //创建 DataSet 对象
    sda.Fill(ds, "test"); //使用 SqlDataAdapter 对象的 Fill 方法填充 DataSet
}

```

```

    dataGridView1.DataSource = ds.Tables[0]; //设置 dataGridView1 的数据源
}
private void button1_Click(object sender, EventArgs e)
{
    DataSet ds1 = ds.Copy(); //调用 DataSet 的 Copy 方法复制 ds 中的内容
    dataGridView2.DataSource = ds1.Tables[0]; //将 ds1 作为 dataGridView2 的数据源
}

```

程序运行结果如图 11.11 所示。



图 11.11 复制 DataSet

## 11.7 照猫画虎——基本功训练

### 11.7.1 基本功训练 1——连接加密的 Access 数据库

视频讲解：光盘\mr\11\lx\连接加密的 Access 数据库.exe

实例位置：光盘\mr\11\zmhh\01

新建一个 Windows 窗体应用程序，向窗体中添加一个 TextBox 控件，用于用户输入连接 Access 数据库的密码；添加一个 Button 控件，用于根据用户填写的密码连接 Access 数据库并查询数据信息；添加一个 DataGridView 控件，用于显示 Access 数据库中指定数据表的数据信息。然后在 Button 控件的 Click 事件中编写如下代码。

```

private void btn_Select_Click(object sender, EventArgs e)
{
    try
    {
        string ConStr = string.Format(
            @"Provider=Microsoft.Jet.OLEDB.4.0;Jet OLEDB:DataBase Password='{0}'; //设置数据库连接字符串
User Id=admin;Data source={1}\test.mdb", txt_Password.Text, Application.StartupPath);
        OleDbConnection oleCon = new OleDbConnection(ConStr); //创建数据库连接对象
        OleDbDataAdapter oleDap = new OleDbDataAdapter("select * from 账目", oleCon); //创建数据适配器对象
        DataSet ds = new DataSet(); //创建数据集
        oleDap.Fill(ds, "账目"); //填充数据集
        this.dataGridView1.DataSource = ds.Tables[0].DefaultView; //设置数据源
    }
}

```

```

        oleCon.Close();
        oleCon.Dispose();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}

```

程序运行结果如图 11.12 所示。



图 11.12 连接加密的 Access 数据库

**照猫画虎：**开发一个程序，使其能够连接网络上共享的 Access 数据库。提示：将数据库连接字符串中的数据源指定为指定的网络共享 Access 数据库路径。(20 分)(实例位置：光盘\mr\11\zmhh\01\_zmhh)

### 11.7.2 基本功训练 2——连接文本文件并显示其内容

视频讲解：光盘\mr\11\lx\连接文本文件并显示其内容.exe

实例位置：光盘\mr\11\zmhh\02

新建一个 Windows 窗体应用程序，在窗体中添加两个 TextBox 控件，分别用于显示文本文件路径和文本文件内容；添加一个 Button 控件，用于选择文本文件并提取文本文件内容。然后在 Button 控件的 Click 事件中打开文本文件，并将其内容显示在 TextBox 控件中，代码如下。

```

private void btn_Browser_Click(object sender, EventArgs e)
{
    //创建打开文件对话框对象
    OpenFileDialog P_open = new OpenFileDialog();
    P_open.Filter = "文本文件|*.txt|所有文件|*.*";
    if (P_open.ShowDialog() == DialogResult.OK)
    {
        txt_Path.Text = P_open.FileName;
        string conn = string.Format(
            @"Provider=Microsoft.Jet.OLEDB.4.0;Data Source={0};Extended Properties='text',
            P_open.FileName.Substring(0,P_open.FileName.LastIndexOf(@"\")));
        OleDbConnection P_OleDbConnection = new OleDbConnection(conn);
        try
        {
            P_OleDbConnection.Open();
            OleDbCommand cmd = new OleDbCommand(
                string.Format("select * from {0}", 

```

```

        P_open.FileName.Substring(P_open.FileName.LastIndexOf(@"\"),  

        P_open.FileName.Length - P_open.FileName.LastIndexOf(@"\"))),P_OleDbConnection);  

        OleDbDataReader oda = cmd.ExecuteReader(); //得到数据读取器对象  

        while (oda.Read())  

        {  

            txt_Message.Text += oda[0].ToString(); //得到文本文件中的字符串  

        }  

    }  

    catch (Exception ex)  

    {  

        MessageBox.Show(ex.Message); //弹出消息对话框  

    }  

    finally  

    {  

        P_OleDbConnection.Close(); //关闭连接  

    }
}

```

程序运行结果如图 11-13 所示。

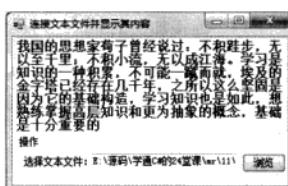


图 11-13 连接文本文件并显示其内容

**熊猫画虎：**根据以上程序，制作一个可以连接 Excel 文件的数据库。提示：将上面程序中连接文本文件的字符串修改为连接 Excel 文件的字符串。(20分)(实例位置：光盘\mr\11\zmhh\02\_zmhh)

### 11.7.3 基本功训练 3——读取 SQL Server 数据库结构

 视频讲解：光盘\mr\11\x\读取 SQL Server 数据库结构.exe

 实例位置：光盘\mr\11\zmhh\03

新建一个 Windows 窗体应用程序，在窗体中添加一个 ComboBox 控件，用于显示数据库信息；添加一个 ListBox 控件，用于显示数据表；添加一个 DataGridView 控件，用于显示数据表结构。读取 SQL Server 数据库结构的功能是在 ListBox 控件的 Click 事件中实现的，代码如下。

```
private void listBox1_Click(object sender, EventArgs e)
{
    string strTableName = this.listBox1.SelectedValue.ToString(); //获取数据表名称
    using (SqlConnection con = new SqlConnection(@"server=MRWXK\WANGXIAOKE;uid=sa;pwd=;database=
"" + strDatabase + "")) //创建数据库连接对象
    {
        string strSql =
            @"select name 字段名, xusertype 类型编号, length 长度 into hy_Linshibiao
from syscolumns where id=object_id('" + strTableName + "');";
        //创建 SQL 字符串
    }
}
```

```

strSql += " //添加字符串信息
@"select name 类型,xusertype 类型编号 into angel_Linshibiao from
sysatypes where xusertype in (select xusertype from syscolumns where id=object_id(" + strTableName + "))";
con.Open(); //打开数据库连接
SqlCommand cmd = new SqlCommand(strSql, con); //创建命令对象
cmd.ExecuteNonQuery(); //执行 SQL 命令
con.Close(); //关闭数据库连接
SqlDataAdapter da = new SqlDataAdapter( //创建数据适配器
    @"select 字段名,类型,长度 from
hy_Linshibiao t,angel_Linshibiao b where t.类型编号=b.类型编号", con);
DataTable dt = new DataTable(); //创建数据表
da.Fill(dt); //填充数据表
this.dataGridView1.DataSource = dt.DefaultView; //设置数据源
SqlCommand cmdnew = new SqlCommand( //创建命令对象
    "drop table hy_Linshibiao,angel_Linshibiao", con);
con.Open(); //打开数据库连接
cmdnew.ExecuteNonQuery(); //执行 SQL 命令
con.Close(); //关闭数据库连接
}
}

```

程序运行结果如图 11.14 所示。

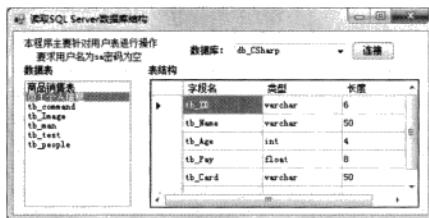


图 11.14 读取 SQL Server 数据库结构

**照猫画虎：**根据以上程序的实现原理，制作一个可以获取 SQL Server 数据库中指定视图结构的程序。  
**(20 分)** (实例位置：光盘\mr\11\zmhh\03\_zmhh)

#### 11.7.4 基本功训练 4——备份指定的 SQL Server 数据库

**视频讲解：**光盘\mr\11\lx\备份指定的 SQL Server 数据库.exe

**实例位置：**光盘\mr\11\zmhh\04

新建一个 Windows 窗体应用程序，在窗体中添加一个 ComboBox 控件，用于用户选择数据库；添加一个 TextBox 控件，用于显示备份文件的路径信息；添加一个 Button 控件，用于备份数据库。备份数据库功能主要是通过自定义的 beifenInfo 方法实现的，该方法代码如下。

```

public void beifenInfo()
{
    try
    {
        sd.InitialDirectory = Application.StartupPath + "\\";
        //设置默认路径
        sd.FilterIndex = 1;
        //默认值为第一个
        sd.RestoreDirectory = true;
        //重新定位保存路径
    }
}

```

```

sd.Filter = "备份文件 (*.bak)|*.bak|所有文件 (*.*)|*.*";
//设置筛选文件类型
if (sd.ShowDialog() == DialogResult.OK)
{
    if (!File.Exists(sd.FileName.ToString()))
    {
        SqlConnection con = new SqlConnection(); //创建数据库连接对象
        con.ConnectionString = @"server=MRWXK\WANGXIAOKE;uid=sa;pwd=" + this.comboBox1.Text + ""; //显示文件路径信息
        con.Open(); //打开数据库连接
        SqlCommand com = new SqlCommand(); //创建命令对象
        this.textBox1.Text = sd.FileName.ToString(); //显示文件路径信息
        com.CommandText = "BACKUP DATABASE " + this.comboBox1.Text + //设置要执行的 SQL 语句
            " TO DISK = '" + sd.FileName.ToString() + "'";
        com.Connection = con; //设置连接属性
        com.ExecuteNonQuery(); //执行 SQL 语句
        con.Close(); //关闭数据库连接
        MessageBox.Show("数据备份成功!"); //弹出消息对话框
    }
    else
    {
        MessageBox.Show("请重新命名!"); //弹出消息对话框
    }
}
catch (Exception k)
{
    MessageBox.Show(k.Message); //弹出消息对话框
}
}

```

程序运行结果如图 11.15 所示。



图 11.15 备份指定的 SQL Server 数据库

**照猫画虎：**完善以上程序，以便在备份完数据库后，能够通过备份文件还原指定的数据库。提示：还原数据库可以使用 restore 语句来实现。(20 分)(实例位置：光盘\mr\11\zmhh\04\_zmhh)

### 11.7.5 基本功训练 5——判断计算机中是否安装了 SQL 软件

■ 视频讲解：光盘\mr\11\x\判断计算机中是否安装了 SQL 软件.exe

■ 实例位置：光盘\mr\11\zmhh\05

新建一个 Windows 窗体应用程序，在窗体中添加一个 Button 控件，用来检查计算机中是否安装了 SQL 软件，然后在窗体的代码页中编写如下代码。

```

public bool ExitSQL()
{
}

```

```

bool sqlFlag = false; //定义布尔类型变量
ServiceController[] services = ServiceController.GetServices(); //得到系统服务集合
for (int i = 0; i < services.Length; i++)
{
    if (services[i].DisplayName.ToString() == "MSSQLSERVER") //方法返回布尔值
        sqlFlag = true;
}
return sqlFlag; //方法返回布尔值
}

private void button1_Click(object sender, EventArgs e)
{
    if (ExitSQL())
    {
        label1.Text = "本地计算机中已经安装 SQL 软件"; //显示字符串信息
    }
    else
    {
        label1.Text = "本地计算机中没有安装 SQL 软件"; //显示字符串信息
    }
}

```

**说明：**程序中使用 `ServiceController` 类时，首先需要添加 `System.ServiceProcess` 命名空间。

程序运行结果如图 11.16 所示。

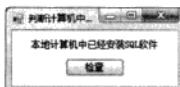


图 11.16 判断计算机中是否安装了 SQL 软件

**照猫画虎：**完善以上程序，在判断完如果安装了 SQL 软件之后，开启 SQL 服务。提示：可以使用 `Process` 类实现。（20 分）（实例位置：光盘\mr\11\zmhh\05\_zmhh）

照猫画虎栏目分数统计：

照猫画虎题目	1	2	3	4	5	总分数	
分数							

## 11.8 情景应用——拓展与实践

### 11.8.1 情景应用 1——向 SQL Server 数据库中批量写入海量数据

**视频讲解：**光盘\mr\11\lx\向 SQL Server 数据库中批量写入海量数据.exe

**实例位置：**光盘\mr\11\qjyy\01

通过 `INSERT` 语句可以向数据库中写入数据记录，但是每执行一次 `INSERT INTO` 语句只可以写入一条数据记录，那么如何实现批量写入数据呢？这里通过在 `INSERT INTO` 语句中嵌入 `SELECT` 语句，并将 `SELECT` 语句的查询结果写入到指定的数据表，从而实现批量向 SQL Server 数据库中写入海量数据的功能。

运行效果如图 11.17 所示。



图 11.17 向 SQL Server 数据库中批量写入海量数据

实现过程如下。

- (1) 创建一个 Windows 窗体应用程序，并将其命名为 UseInsertSelect。
- (2) 更改默认窗体 Form1 的 Name 属性为 Frm\_Main，在该窗体中添加一个 Button 控件，用于向数据库中批量写入数据；添加一个 DataGridView 控件，用于显示数据表中的信息。
- (3) 批量写入海量数据的功能主要是通过一个自定义的方法 InsertData 实现的，该方法代码如下。

```
private void InsertData()
{
    string P_Str_ConnectionStr = string.Format(
        @"server=MRWXKXWANGXIAOKE;database=db_CSharp;uid=sa;pwd=");
    string P_Str_SqlStr = string.Format(
        @"INSERT INTO tb_Student_Copy(学生姓名,学生年龄,性别,家庭住址)
    SELECT 学生姓名,年龄,性别,家庭住址 FROM tb_Student");
    SqlConnection P_con = new SqlConnection(P_Str_ConnectionStr);
    try
    {
        P_con.Open();
        SqlCommand P_cmd = new SqlCommand(P_Str_SqlStr, P_con);
        if (P_cmd.ExecuteNonQuery() != 0)
        {
            MessageBox.Show("成功写入数据", "提示!");
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "提示!");
    }
    finally
    {
        P_con.Close();
    }
}
```

**DIY：**制作一个程序，用来使用事务批量提交两个数据表。提示：主要使用 SqlTransaction 类实现，用到 tb\_Author 和 tb\_AuthorsBook 两个数据表。(20 分)(实例位置：光盘\mr\11\qjyy\01\_diy)

### 11.8.2 情景应用 2——使用断开式连接批量更新数据库中数据

■**视频讲解：**光盘\mr\11\lx\使用断开式连接批量更新数据库中数据.exe

■**实例位置：**光盘\mr\11\qjyy\02

本实例主要使用断开式连接批量更新数据库中数据。新建一个 Windows 窗体应用程序，在窗体中添加一个 DataGridView 控件，用于显示数据库中的数据记录和修改数据信息；添加一个 Button 控件，用于提交用户在 DataGridView 控件中对数据所做的修改。然后在 Button 控件的 Click 事件中实现批量更新数据库中数据的功能，代码如下。

```
private void btn_Submit_Click(object sender, EventArgs e)
{
    SqlDataAdapter P_SqlDataAdapter = new SqlDataAdapter(); //创建数据适配器
    SqlCommand P_cmd = new SqlCommand() //创建命令对象
    @ "UPDATE tb_Student_Copy SET 学生姓名=@name,学生年龄=@age,性别=@sex,家庭住址=@address
    WHERE id=@id",
    new SqlConnection(@"server=MRWXK\WANGXIAOKE;database=db_CSharp;uid=sa;pwd="));
    P_cmd.Parameters.Add("@id", SqlDbType.Int, 10, "id"); //设置参数
    P_cmd.Parameters.Add("@name", SqlDbType.VarChar, 10, "学生姓名"); //设置参数
    P_cmd.Parameters.Add("@age", SqlDbType.Int, 10, "学生年龄"); //设置参数
    P_cmd.Parameters.Add("@sex", SqlDbType.NChar, 2, "性别"); //设置参数
    P_cmd.Parameters.Add("address", SqlDbType.VarChar, 50, "家庭住址"); //设置参数
    P_SqlDataAdapter.UpdateCommand = P_cmd; //设置 UpdateCommand 属性
    P_SqlDataAdapter.Update(G_st.Tables[0]); //更新数据库中数据
    G_st.AcceptChanges(); //提交修改
    MessageBox.Show("更改成功！","提示！");
    GetMessage(); //弹出消息对话框
    dgv_Message.DataSource = G_st.Tables[0]; //填充表
    dgv_Message.Columns[0].Visible = false; //设置数据源
    } //隐藏主键列
}
```

程序运行结果如图 11.18 所示。

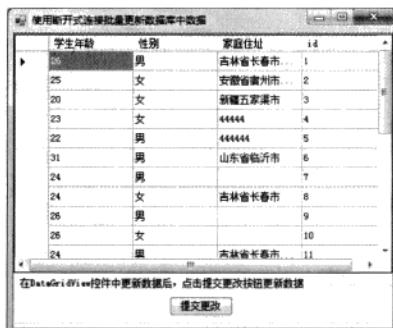


图 11.18 使用断开式连接批量更新数据库中数据

**DIY：**通过使用 SqlCommandBuilder 类实现与上面程序相同的功能。提示：要更新的数据表中必须设置主键。(20 分)(实例位置：光盘\mr\11\qjyy\02\_diy)



```

{
    if (validateNum(txt_Age.Text) && validateNum(txt_Age2.Text))
        FindValue += "(Age between " + Convert.ToInt32(txt_Age.Text) +      //组合 SQL 字符串
                    " and " + Convert.ToInt32(txt_Age2.Text) + ") and";
    else
    {
        MessageBox.Show("年龄必须为数字！");
        txt_Age.Text = txt_Age2.Text = "";
        txt_Age.Focus();
    }
}
else
{
    if (txt_Age.Text != "")
    {
        if (validateNum(txt_Age.Text))
            FindValue += "(Age = " + Convert.ToInt32(txt_Age.Text) + ") and"; //组合 SQL 字符串
        else
        {
            MessageBox.Show("年龄必须为数字！");
            txt_Age.Text = "";
            txt_Age.Focus();
        }
    }
    else if (txt_Age2.Text != "")
    {
        if (validateNum(txt_Age2.Text))
            FindValue += "(Age = " + Convert.ToInt32(txt_Age2.Text) + ") and"; //组合 SQL 字符串
        else
        {
            MessageBox.Show("年龄必须为数字！");
            txt_Age2.Text = "";
            txt_Age2.Focus();
        }
    }
}
if (txt_QQ.Text != "")
{
    if (validateNum(txt_QQ.Text) && txt_QQ.Text.Length >= 4 && txt_QQ.Text.Length <= 9)
        FindValue += "(QQ = " + Convert.ToInt32(txt_QQ.Text) + ") and"; //组合 SQL 字符串
    else
    {
        MessageBox.Show("QQ 号码必须为 4 到 9 位以内的数字！");
        txt_QQ.Text = "";
        txt_QQ.Focus();
    }
}
if (txt_Phone.Text != "")
{
    if (validatePhone(txt_Phone.Text))
        FindValue += "(Tel=" + txt_Phone.Text + ") and"; //组合 SQL 字符串
    else
}

```

```

    {
        MessageBox.Show("请输入正确的电话号码！");
        txt_Phone.Text = "";
        txt_Phone.Focus();
    }
}

if (txt_Email.Text != "")
{
    if (validateEmail(txt_Email.Text))
        FindValue += "(Email=" + txt_Email.Text + ") and";
    else
    {
        MessageBox.Show("请输入正确的 Email 地址！");
        txt_Email.Text = "";
        txt_Email.Focus();
    }
}

if (txt_Address.Text != "")
    FindValue += "(Address=" + txt_Address.Text + ") and";
if (FindValue.Length > 0)
{
    if (FindValue.IndexOf("and") > -1)
        FindValue = FindValue.Substring(0, FindValue.Length - 4);
}
else
{
    FindValue = "";
    if (FindValue != "")
        Find_SQL = Find_SQL + " where " + FindValue;
    GetAllInfo(Find_SQL);
}
}

```

//弹出消息对话框  
//引用空字符串  
//得到焦点  
//组合 SQL 字符串  
//弹出消息对话框  
//引用空字符串  
//得到焦点  
//组合 SQL 字符串  
//删除 AND 运算符  
//如果 FindValue 字段不为空  
//组合 SQL 字符串  
//按照 SQL 字符串进行查询

**DIY：**完善以上程序，在查询完职工信息之后，通过分页形式查看查询完的用户信息。（20 分）（实例位置：光盘\mr\11\qjyy\03\_diy）

#### 11.8.4 情景应用 4——使用二进制存取用户头像

■ 视频讲解：光盘\mr\11\lx\使用二进制存取用户头像.exe

■ 实例位置：光盘\mr\11\qjyy\04

在一个完善的客户管理系统中，图像数据的存取是必不可少的，如用户头像等信息，这里使用 C#实现了使用二进制存取用户头像的功能。运行效果如图 11.20 所示。

实现过程如下。

(1) 创建一个 Windows 窗体应用程序，并将其命名为 SaveBinary。

(2) 更改默认窗体 Form1 的 Name 属性为 Frm\_Main。在该窗体中添加一个 TextBox 控件，用于填写用户名；添加一个 PictureBox 控件，用于显示用户头像；添加两个 Button 控件，分别用于选择和添加用户头像信息；添



图 11.20 使用二进制存取用户头像

加一个 DataGridView 控件，用于显示用户信息。

(3) 在 Frm\_Main 窗体的后台代码中，首先创建程序所需要的 ADO.NET 对象及公共变量，代码如下。

```
SqlConnection sqlcon; //声明数据库连接对象
SqlDataAdapter sqlda; //声明数据桥接器对象
DataSet myds; //声明数据集对象
string strCon = @"Data Source=MRWXK\WANGXIAOKE;Database=db_CSharp;uid=sa;pwd=";
//定义数据库连接字符串
```

(4) 在 Frm\_Main 窗体中单击“选择”按钮，打开“选择头像”对话框，选择用户头像，并将选择的头像显示在 PictureBox 控件中。“选择”按钮的 Click 事件代码如下。

```
private void button1_Click(object sender, EventArgs e)
{
    //定义可选择的头像类型
    openFileDialog1.Filter = "*.*";
    openFileDialog1.Title = "选择头像";
    if (openFileDialog1.ShowDialog() == DialogResult.OK) //判断是否选择了头像
    {
        pictureBox1.Image = Image.FromFile(openFileDialog1.FileName); //显示选择的用户头像
    }
}
```

(5) 如果已经选择用户头像，并输入用户名，则单击“添加”按钮，将用户名和头像保存到数据库中，同时刷新 DataGridView 中的数据显示。“添加”按钮的 Click 事件代码如下。

```
private void button2_Click(object sender, EventArgs e)
{
    if (openFileDialog1.FileName != "" && textBox1.Text != "")
    {
        if (AddInfo(textBox1.Text, openFileDialog1.FileName)) //添加用户信息
        {
            MessageBox.Show("用户信息添加成功");
        }
    }
    ShowInfo();
}
```

(6) 在上面的代码中用到了 AddInfo 和 ShowInfo 两个方法，下面分别进行介绍。AddInfo 方法为自定义的返回值类型为 bool 类型的方法，主要用来添加用户信息，它有两个参数，分别表示用户名和选择的头像名称；AddInfo 方法实现代码如下。

```
private bool AddInfo(string strName, string strImage)
{
    sqlcon = new SqlConnection(strCon); //创建数据库连接对象
    FileStream FStream = new FileStream(strImage, FileMode.Open, FileAccess.Read); //创建 FileStream 对象
    BinaryReader BReader = new BinaryReader(FStream); //创建 BinaryReader 读取对象
    byte[] bytelImage = BReader.ReadBytes((int)FStream.Length); //读取二进制图片
    SqlCommand sqlcmd = new SqlCommand("insert into tb_Image(name,photo) values(@name,@photo)",sqlcon);
    sqlcmd.Parameters.Add("@name", SqlDbType.VarChar, 50).Value = strName;//为 SQL 语句添加@name 参数
    sqlcmd.Parameters.Add("@photo", SqlDbType.Image).Value = bytelImage;//为 SQL 语句添加@Photo 参数
    sqlcon.Open(); //打开数据库连接
    sqlcmd.ExecuteNonQuery(); //执行用户信息添加操作
    sqlcon.Close(); //关闭数据库连接
```

```
    return true;
}
```

(7) ShowInfo 方法为自定义的无返回值类型方法，主要用来在 DataGridView 控件中显示用户名。该方法实现代码如下。

```
private void ShowInfo()
{
    sqlcon = new SqlConnection(strCon); //创建数据库连接对象
    sqlda = new SqlDataAdapter("select name as 用户名称 from tb_Image", sqlcon); //创建数据库桥接器对象
    myds = new DataSet(); //创建数据集对象
    sqlda.Fill(myds); //填充数据集
    dataGridView1.DataSource = myds.Tables[0]; //为 DataGridView 设置数据源
}
```

(8) 当在 DataGridView 控件中选择某用户名时，程序会自动从数据库中查找该用户的详细信息，并将用户名显示在窗体左侧的“用户名”文本框中，同时将用户头像显示在窗体左侧的 PictureBox 控件中。实现代码如下。

```
private void dataGridView1_CellClick(object sender, DataGridViewCellEventArgs e)
{
    string strName = dataGridView1.Rows[e.RowIndex].Cells[0].Value.ToString(); //记录选择的用户名
    if (strName != "")
    {
        sqlcon = new SqlConnection(strCon); //创建数据库连接对象
        //创建数据桥接器对象
        sqlda = new SqlDataAdapter("select * from tb_Image where name=" + strName + "", sqlcon); //创建数据集对象
        myds = new DataSet(); //创建数据集
        sqlda.Fill(myds); //填充数据集
        textBox1.Text = myds.Tables[0].Rows[0][1].ToString(); //显示用户名
        //使用数据库中存储的二进制头像创建内存数据流
        MemoryStream MStream = new MemoryStream((byte[])myds.Tables[0].Rows[0][2]); //显示用户头像
        pictureBox1.Image = Image.FromStream(MStream);
    }
}
```

**DIY：**制作一个程序，通过存储文件名的方式存储用户头像信息。(20 分)(实例位置：光盘\mr\11\qjyy\04\_diy)

### 11.8.5 情景应用 5——使用存储过程实现员工自动编号

■**视频讲解：**光盘\mr\11\lx\使用存储过程实现员工自动编号.exe

■**实例位置：**光盘\mr\11\qjyy\05

本实例主要使用存储过程实现员工自动编号的功能。新建一个 Windows 窗体应用程序，在窗体中添加一个 Label 控件，用于显示自动生成的员工编号；添加 4 个 TextBox 控件，分别用来输入员工的信息；添加 3 个 Button 控件，分别用来执行自动生成编号、添加信息和关闭程序操作。自动生成员工编号的功能主要是通过一个自定义的方法 strid 实现的，该方法代码如下。

```
private string strid()
{
    //创建数据库连接对象
    using (SqlConnection con = new SqlConnection("server=MRWXK\WANGXIAOKE;pwd=;uid=sa;database=db_CSharp"))
```

```

{
    try
    {
        SqlCommand cmd = new SqlCommand();
        cmd.CommandType = CommandType.StoredProcedure;
        cmd.Connection = con;
        cmd.Connection.Open();
        cmd.CommandText = "Proc_IdS";
        SqlParameter par = new SqlParameter("@Id", SqlDbType.VarChar, 8); //创建 SqlCommand 对象
        par.Direction = ParameterDirection.Output; //指定执行存储过程
        cmd.Parameters.Add(par); //指定数据库连接对象
        cmd.ExecuteNonQuery(); //打开连接
        con.Close(); //指定存储过程名称
        this.groupBox1.Enabled = true; //添加参数
        return par.Value.ToString(); //执行存储过程
    }
    catch (Exception ey)
    {
        MessageBox.Show(ey.Message); //关闭数据库连接
        return null; //启用控件
    }
}
}

```

本实例中用到的 Proc\_IdS 存储过程代码如下。

```

CREATE proc [dbo].[Proc_IdS]
@Id varchar(8) output
as
begin
declare @TempString varchar(8)
select @TempString=Max(tb_ID) from 员工个人信息
        set
@TempString=Substring( @TempString,1,1)+cast((CAST(Substring( @TempString,2,len(@TempString)) as
int)+1) as varchar)
        set @Id=@TempString
end

```

程序运行结果如图 11.21 所示。



图 11.21 使用存储过程实现员工自动编号

**DIY：**仿照以上程序制作一个程序，主要实现自动生成职工编号、查询职工信息和添加职工信息的功能。**提示：**涉及数据库的操作都使用存储过程实现。(20 分)(实例位置：光盘\mr\11\qjyy\05\_diy)

情景应用 DIY 栏目分数统计：

DIY 题目	1	2	3	4	5	总分数	
分数							

## 11.9 自我测试

### 一、选择题（每题 10 分，5 道题）

1. 在 ADO.NET 中，用来与数据源建立连接的对象是（ ）。  
A. Connection 对象      B. Command 对象      C. DataAdapter 对象      D. DataSet 对象
2. 如果想使用 SqlCommand 对象对 SQL Server 数据库进行操作，应引入（ ）命名空间。  
A. System.Data.OleDb      B. System.Data.SqlClient  
C. System.Data.Odbc      D. System.Data.OracleClient
3. 数据库连接字符串 “Server=(local);Database=Northwind;Uid=sa;Pwd=” 中，Server 指的是（ ）。  
A. 数据库名      B. 数据库服务器名      C. 数据表名      D. 用户计算机名
4. 使用 Command 对象的（ ）方法可以创建一个 DataReader 对象。  
A. ExecuteReader      B. ExecuteScalar  
C. ExecuteNonQuery      D. ExecuteXmlReader
5. “DataRelation relation=new DataRelation(r1,r2,r3);” 这条语句中的 r1、r2 和 r3 分别表示（ ）。  
A. 父列、子列、关系名      B. 关系名、子列、父列  
C. 子列、父列、关系名      D. 关系名、父列、子列

### 二、填空题（每题 10 分，5 道题）

1. 如果要执行一条更新的 SQL 语句，应使用 Command 对象的（ ）方法。
2. 如果要把数据集中的数据更新回数据源，应使用（ ）对象的 Update 方法。
3. 如果要使用存储过程对数据源进行操作，应设置 Command 对象的（ ）属性。
4. 通常情况下，DataReader 对象在内存中保留（ ）行数据。
5. OleDbDataAdapter 对象通过（ ）对象在数据库中执行 SQL 语句。

测试分数统计：

类别	第 1 题	第 2 题	第 3 题	第 4 题	第 5 题
选择题分数					
填空题分数					
总分数					

## 11.10 行动指南

开始日期：\_\_\_\_\_ 年 \_\_\_\_\_ 月 \_\_\_\_\_ 日      结束日期：\_\_\_\_\_ 年 \_\_\_\_\_ 月 \_\_\_\_\_ 日

序号	内 容	行动指南	
		分数>75 分	优秀，基本功掌握得不错，加油！
1	照猫画虎栏目	75 分>分数>50 分	及格，知识掌握得不牢，重新做一遍照猫画虎。
	分数（ ）	分数<50 分	请使用光盘中提供的“实战能力测试系统”进行提高训练。

续表

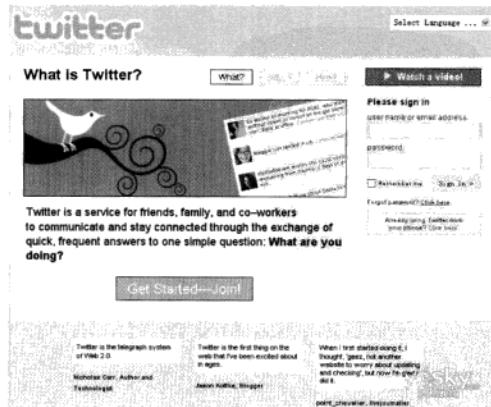
序号	内 容	行 动 指 南	
1	情景应用栏目 分数( )	分数>75 分	优秀, 综合应用能力很强。
		75 分>分数>50 分 分数<50 分	及格, 综合应用能力需提高, 再练一遍情景应用。 请使用光盘中提供的“实战能力测试系统”进行提高训练。
	自我测试栏目 分数( )	分数>75 分	优秀, 有成为编程高手的潜质。
		分数<75 分	请使用光盘中提供的“实战能力测试系统”进行提高训练。
综合评价		反复训练后, 以上各项分数都在 75 分以上, 方可进入下一堂课学习。	
2	编程能力培养: 本栏目提供了一些实践题目, 对于培养编程能力很有效, 要做好。	1. 开发一个程序, 用来将 Access 数据库中的数据导入到 Excel 文件中。 2. 制作一个数据库通用连接器, 可以根据用户的选择自动生成连接 Access、Excel 或 SQL Server 数据库的 SQL 语句。 3. 开发一个程序, 主要通过存储过程对职工信息进行管理。	
3	编程习惯培养: 本堂课培养读者在学习开发中记笔记的习惯, 把开发中遇到的问题、总结的经验记录下来。		
4	创新能力培养: 看看身边有没有可以用编程解决的问题, 记录到右边的表格中。根据学习进度, 尝试编写解决这些问题的小程序。		

## 11.11 成功可以复制——微型博客 Twitter 创始人埃文·威廉姆斯

Twitter, 这个发布于 2006 年的微型博客网站, 目前已成为互联网时代最新最热的明星。对很多人来说, Twitter 的迅速崛起实在令人意外, 对 Twitter 的创始人埃文·威廉姆斯来说, 同样颇为感慨。

要知道, 他当初推出 Twitter, 主要是为了方便交流和共享。

埃文·威廉姆斯, 1970 年 8 月 20 生在内布拉斯加州的一个农场, 高中毕业后, 考入了内布拉斯加州大学, 中途认为“上大学是浪费时间”而退学。1994 年, 埃文·威廉姆斯创办了一家互联网公司。虽然埃文·威廉姆斯当时并不懂互联网, 但他认为互联网最终将非常



Twitter 首页

重要。通过自学，埃文·威廉姆斯知道了如何创建网站，并给一些企业做一些项目。但因没有管理经验，企业没经营多久就倒闭了。回忆过去，埃文·威廉姆斯说“当时我赔了很多钱，包括我父亲的投资。公司因为拖欠国税局的税款而关门，当时我的很多员工都要疯掉了。”

在第一次创业失败后，埃文·威廉姆斯并没有放弃，他决定去硅谷闯荡。威廉姆斯进入一家媒体公司，一开始做销售，后来开始编写计算机程序，很快，他就开始承接硅谷巨头英特尔公司和惠普公司的业务。但他厌倦了公司的环境，决定辞职继续创业。

1999 年，埃文·威廉姆斯与人合伙成立了 Pyra 实验室，主要制作管理软件。在这期间，为了方便交流，埃文·威廉姆斯偶然发明了博客。后来，博客变成了公司的主业，博客网站（Blogger.com）也随之上线。前两年大火特火的博客一词，就是埃文·威廉姆斯发明的。2001 年，《财富》杂志曾将“博客网”评选为最佳创新网站。2000 年，互联网泡沫破灭，埃文·威廉姆斯和他的公司也未能幸免。他别无选择，只有裁掉公司所有员工，最后只剩下他一人独自苦撑。不过，“博客网”不仅改变了网络世界，也改变了埃文·威廉姆斯的人生。2003 年，谷歌公司决定收购“博客网”，濒临绝境的埃文·威廉姆斯把公司卖给了谷歌，这笔交易让威廉姆斯捞到了他的第一桶金。直到今天，埃文·威廉姆斯仍然认为，“博客网”是他最大的创造。

2005 年，埃文·威廉姆斯成立了一家播客公司，又“偶然”地创造了 Twitter，谱写了新的互联网奇迹。



### 经典语录

企业家分为两种类型，一种是把赚钱当作目标，另一种是为了创造新事物，创造从未有过的产品或服务。我属于第二种，并热衷于此。这是世界上出现新事物的源泉，而我非常幸运能够创造出一个改变世界的产品。



### 深度评价

从埃文·威廉姆斯身上，我们学到了 3 点经验：1. 积极地思考，创新地工作，想办法让工作更高效、更便捷、更快速，或许，你就发现了机会。2. 不断尝试才能创造价值，坚持才能收获成功。3. 专注于为用户创造最大的价值，而不是其他，如金钱等。

# 第 12 堂课

## DataGridView 数据控件

(  视频讲解：103分钟 )

在开发 WinForms 应用程序时，经常需要使用数据库存储数据，那么如何将数据库中存储的数据呈现在用户面前呢？使用 DataGridView 控件可以快速地将数据库中的数据显示给用户，并且可以直接对数据进行操作，从而大大增强了操作数据库的效率，本堂课将对 DataGridView 数据控件及其使用进行详细讲解。

学习摘要：

- ▶ 熟悉 DataGridView 控件的概念及作用
- ▶ 掌握如何通过 DataGridView 控件显示数据
- ▶ 掌握如何获取 DataGridView 控件中的单元格
- ▶ 掌握如何通过 DataGridView 控件批量修改数据
- ▶ 掌握如何设置选中控件中某行时显示不同颜色
- ▶ 掌握如何禁止在 DataGridView 控件中添加和删除行
- ▶ 掌握如何禁用 DataGridView 控件的自动排序功能
- ▶ 掌握如何合并 DataGridView 控件的单元格

## 12.1 DataGridView 控件概述

DataGridView 控件提供一种强大而灵活的、以表格形式显示数据的方式，可以使用它来显示少量数据的只读视图，也可以对其进行缩放以显示特大数据集的可编辑视图，使用该控件还可以显示和编辑来自多种不同类型的数据源的表格数据。将数据绑定到 DataGridView 控件非常简单和直观，在大多数情况下，只需设置 DataSource 属性即可。另外，DataGridView 控件具有极高的可配置性和可扩展性，它提供有大量的属性、方法和事件，可以用来对该控件的外观和行为进行自定义。当需要在 Windows 窗体应用程序中显示表格数据时，应首先考虑使用 DataGridView 控件。若要以小型网格显示只读值或者使用户能够编辑具有数百万条记录的表，DataGridView 控件将提供可以方便地进行编程以及有效地利用内存的解决方案。

## 12.2 在 DataGridView 控件中显示数据

通过 DataGridView 控件显示数据表中的数据，首先需要使用 DataAdapter 对象查询指定的数据，然后通过该对象的 Fill 方法填充 DataSet 数据集，最后设置 DataGridView 控件的 DataSource 属性为 DataSet 数据集即可。DataSource 属性用于获取或设置 DataGridView 控件所显示数据的数据源。

语法：public Object DataSource { get; set; }

说明：它的属性值包含 DataGridView 控件要显示的数据的对象。

**例 12.01** 创建一个 Windows 应用程序，向窗体中添加一个 DataGridView 控件，然后将数据表 tb\_emp 中的数据显示到 DataGridView 控件中，代码如下。（实例位置：光盘\mr\12\sl\12.01）

```
private void Form1_Load(object sender, EventArgs e)
{
    //创建 SqlConnection 变量 conn，连接数据库
    SqlConnection conn = new SqlConnection("server=MRWXK\WANGXIAOKE;database=db_CSharp;uid=sa;
    pwd=");
    SqlDataAdapter sda = new SqlDataAdapter("select * from tb_emp", conn); //创建一个 SqlDataAdapter 对象
    DataSet ds = new DataSet(); //创建一个 DataSet 对象
    sda.Fill(ds); //使用 SqlDataAdapter 对象的 Fill 方法填充 DataSet
    dataGridView1.DataSource = ds.Tables[0]; //设置 dataGridView1 控件数据源
}
```

程序运行结果如图 12.1 所示。

编号	姓名	职称	部门
3	小大	高级工程师	C#
4	小王	高级工程师	ASP.NET
5	笑笑	高级工程师	ASP.NET
6	小刘	高级工程师	C#
7	小强	工程师	C#

图 12.1 显示 tb\_emp 数据表中的数据

### 12.3 获取 DataGridView 控件中的当前单元格

如果要与 DataGridView 控件进行交互，则通常需要通过编程方式发现其中哪个单元格处于活动状态，然后通过 DataGridView 控件的 CurrentCell 属性来获取当前单元格信息。

CurrentCell 属性用于获取当前处于活动状态的单元格。

语法：public DataGridViewCell CurrentCell { get; set; }

说明：它的属性值表示当前单元格的 DataGridViewCell，如果没有当前单元格，则为空引用。默认值是第一列中的第一个单元格。

**例 12.02** 创建一个 Windows 应用程序，向窗体中添加一个 DataGridView 控件、一个 Button 控件和一个 Label 控件，分别用于显示数据、获取指定单元格信息以及显示单元格信息。当单击 Button 控件时，通过 DataGridView 控件的 CurrentCell 属性来获取当前单元格信息。代码如下。（实例位置：光盘\mr\12\s\12.02）

```
SqlConnection conn; //声明一个 SqlConnection 变量
SqlDataAdapter sda; //声明一个 SqlDataAdapter 变量
DataSet ds = null; //声明一个 DataSet 变量
private void Form1_Load(object sender, EventArgs e)
{
    //创建 SqlConnection 变量 conn，连接数据库
    conn = new SqlConnection("server=MRWXK\WANGXIAOKE;database=db_CSharp;uid=sa;pwd=");
    sda = new SqlDataAdapter("select * from tb_teacher", conn); //创建 SqlDataAdapter 对象
    ds = new DataSet(); //创建 DataSet 对象
    sda.Fill(ds); //使用 SqlDataAdapter 对象的 Fill 方法填充 DataSet
    dataGridView1.DataSource = ds.Tables[0]; //设置 dataGridView1 控件的数据源
}
private void button1_Click(object sender, EventArgs e)
{
    //使用 CurrentCell.RowIndex 和 CurrentCell.ColumnIndex 获取数据的行和列坐标
    string msg = String.Format("第{0}行,第{1}列", dataGridView1.CurrentCell.RowIndex, dataGridView1.CurrentCell.ColumnIndex);
    label1.Text = "选择的单元格为：" + msg;
}
```

程序运行结果如图 12.2 所示。

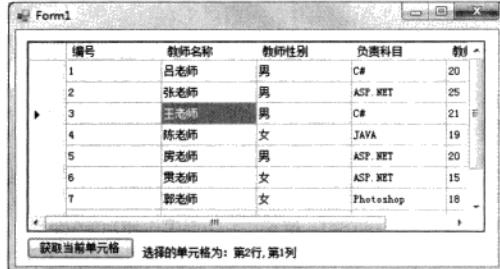


图 12.2 获取单元格的信息

## 12.4 在 DataGridView 控件中修改数据

在 DataGridView 控件中修改数据主要用到 DataTable 的 ImportRow 方法和 DataAdapter 对象的 Update 方法，其实现的过程是通过 DataTable 的 ImportRow 方法将更改后的数据复制到一个 DataTable 中，然后通过 DataAdapter 对象的 Update 方法将 DataTable 中的数据更新到数据库中。

ImportRow 方法用于将 DataRow 复制到 DataTable 中，保留任何属性设置以及初始值和当前值。

语法：public void ImportRow(DataRow row)

说明：row 表示要导入的 DataRow。

**说明：**DataAdapter 对象的 Update 方法在第 11 堂课中已经做过详细介绍，此处不再赘述。

**注意：**在默认情况下，用户可以通过在当前 DataGridView 文本框单元格中输入或按 F2 键来编辑该单元格的内容。在 DataGridView 控件单元格中编辑内容的前提是该控件已启用以及单元格、行、列和控件的 ReadOnly 属性都设置为 false。ReadOnly 属性用于指示用户是否可以编辑 DataGridView 控件的单元格。

**例 12.03** 创建一个 Windows 应用程序，向默认窗体中添加一个 DataGridView 控件和两个 Button 控件。DataGridView 控件用于显示和修改数据；而这两个 Button 控件分别用于加载数据和将修改后的数据更新到数据库中。代码如下。（实例位置：光盘\mr\12\s\12.03）

```

int intIndex = 0;                                //记录行索引
SqlConnection conn;                             //声明一个 SqlConnection 变量
SqlDataAdapter adapter;                         //声明一个 SqlDataAdapter 变量
private void button1_Click(object sender, EventArgs e)
{
    //创建 SqlConnection 变量 conn，连接数据库
    conn = new SqlConnection("server=MRWXK\WANGXIAOKE;database=db_CSharp;uid=sa;pwd=");
    //创建 SqlDataAdapter 对象
    SqlDataAdapter sda = new SqlDataAdapter("select * from tb_emp", conn);
    DataSet ds = new DataSet();                    //创建 DataSet 对象
    sda.Fill(ds);                               //使用 SqlDataAdapter 对象的 Fill 方法填充 DataSet
    dataGridView1.DataSource = ds.Tables[0];        //设置 dataGridView1 控件的数据源
    dataGridView1.RowHeadersVisible = false;         //禁止显示行标题
    for (int i = 0; i < dataGridView1.ColumnCount; i++) //使用 for 循环设置控件的列宽
    {
        dataGridView1.Columns[i].Width = 84;
    }
    button1.Enabled = false;                      //禁用按钮
    dataGridView1.Columns[0].ReadOnly = true;        //将控件设置为只读
}
private DataTable dbconn(string strSql)           //建立一个 DataTable 类型的方法
{
    this.adapter = new SqlDataAdapter(strSql, conn); //创建 SqlDataAdapter 对象
    DataTable dtSelect = new DataTable();            //创建 DataTable 对象
    int rnt = this.adapter.Fill(dtSelect);          //使用 Fill 方法填充 DataTable 对象
    return dtSelect;                               //返回 DataTable 对象
}

```

```

private void button2_Click(object sender, EventArgs e)
{
    if (dbUpdate()) //判断 dbUpdate 方法返回的值是否为 true
    {
        MessageBox.Show("修改成功！"); //弹出提示
    }
    button1_Click(sender, e);
}
private Boolean dbUpdate() //建立一个 Boolean 类型的方法 dbUpdate
{
    string strSql = "select * from tb_emp"; //声明 SQL 语句
    DataTable dtUpdate = new DataTable(); //创建 DataTable
    dtUpdate = this.dbconn(strSql); //调用 dbconn 方法
    dtUpdate.Rows.Clear(); //调用 Clear 方法
    DataTable dtShow = new DataTable(); //创建 DataTable
    dtShow = (DataTable)this.dataGridView1.DataSource; //使用 ImportRow 方法复制 dtShow 中的值
    dtUpdate.ImportRow(dtShow.Rows[intindex]); //使用 ImportRow 方法复制 dtShow 中的值
    try
    {
        SqlCommandBuilder CommandBuiler; //声明 SqlCommandBuilder 变量
        CommandBuiler = new SqlCommandBuilder(this.adapter); //调用 Update 方法更新数据
        this.adapter.Update(dtUpdate);
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message.ToString()); //出现异常，弹出提示
        return false;
    }
    dtUpdate.AcceptChanges(); //提交更改
    return true;
}
private void dataGridView1_CellClick(object sender, DataGridViewCellEventArgs e)
{
    intindex = e.RowIndex; //记录当前行号
}

```

程序运行结果如图 12.3 所示。



图 12.3 在 DataGridView 控件中修改数据

## 12.5 选中 DataGridView 控件中的行时显示不同颜色

可以利用 DataGridView 控件的 SelectionMode 属性、ReadOnly 属性和 SelectionBackColor 属性实现当选

中 DataGridView 控件中的行时显示不同颜色，下面对这 3 个属性进行详细介绍。

(1) SelectionMode 属性用于设置如何选择 DataGridView 控件的单元格。

语法： public DataGridViewSelectionMode SelectionMode { get; set; }

说明：它的属性值为 DataGridViewSelectionMode 枚举值之一，默认为 RowHeaderSelect。DataGridViewSelectionMode 枚举值及说明如表 12.1 所示。

表 12.1 DataGridViewSelectionMode 枚举值及说明

枚举值	说明
CellSelect	可以选定一个或多个单元格
ColumnHeaderSelect	可以通过单击列的标头单元格选定此列，通过单击某个单元格可以单独选定此单元格
FullColumnSelect	通过单击列的标头或该列所包含的单元格选定整个列
FullRowSelect	通过单击行的标头或该行所包含的单元格选定整个行
RowHeaderSelect	通过单击行的标头单元格选定此行，通过单击某个单元格可以单独选定此单元格

(2) ReadOnly 属性用于设置是否可以编辑 DataGridView 控件的单元格。

语法： public bool ReadOnly { get; set; }

说明：它的属性值表示，如果用户不能编辑 DataGridView 控件的单元格，为 true，否则为 false，默认为 false。

**例 12.04** 禁止用户编辑 dataGridView1 控件的单元格，代码如下。

```
dataGridView1.ReadOnly = true;
```

(3) SelectionBackColor 属性用于设置 DataGridView 单元格在被选定时的背景色。

语法： public Color SelectionBackColor { get; set; }

说明：它的属性值为一个 Color 结构，表示选定单元格的背景色，默认为 Empty。

 注意：由于 SelectionBackColor 属性包含在 DataGridViewCellStyle 类中，所以调用此属性之前要调用 DataGridViewCellStyle 属性。

**例 12.05** 创建一个 Windows 应用程序，向默认窗体中添加一个 DataGridView 控件，用于显示 tb\_emp 表中的所有数据，然后通过 DataGridView 控件的 SelectionMode 属性、ReadOnly 属性和 SelectionBackColor 属性实现选中某行时，改变选中行的背景颜色，代码如下。（实例位置：光盘\mr\12\sl\12.05）

```
SqlConnection conn; //声明 SqlConnection 对象
private void Form1_Load(object sender, EventArgs e)
{
    //创建 SqlConnection 对象 conn，连接数据库
    conn = new SqlConnection("server=MRWXK\WANGXIAOKE;database=db_CSharp;uid=sa;pwd=");
    //创建 SqlDataAdapter 对象
    SqlDataAdapter sda = new SqlDataAdapter("select * from tb_emp", conn);
    DataSet ds = new DataSet(); //创建 DataSet 对象
    sda.Fill(ds); //使用 SqlDataAdapter 对象的 Fill 方法填充 DataSet
    dataGridView1.DataSource = ds.Tables[0]; //设置 dataGridView1 控件的数据源
    //设置 SelectionMode 属性为 FullRowSelect 使控件能够整行选择
    dataGridView1.SelectionMode = DataGridViewSelectionMode.FullRowSelect;
    dataGridView1.ReadOnly = true; //设置 dataGridView1 控件的 ReadOnly 属性，使其为只读
    //设置 dataGridView1 控件的 DefaultCellStyle.SelectionBackColor 属性，使其选择行为黄绿色
    dataGridView1.DefaultCellStyle.SelectionBackColor = Color.YellowGreen;
}
```

程序运行结果如图 12.4 所示。



图 12.4 选中某行时显示不同颜色

## 12.6 禁止在 DataGridView 控件中添加和删除行

通过设置 DataGridView 控件的公共属性 AllowUserToAddRows 属性、AllowUserToDeleteRows 属性和 ReadOnly 属性可以禁止在 DataGridView 控件中添加和删除行。AllowUserToAddRows 属性设置一个值，该值指示是否向用户显示添加行的选项；AllowUserToDeleteRows 属性设置一个值，该值指示是否允许用户从 DataGridView 中删除行；ReadOnly 属性设置一个指示网格是否处于只读模式的值。

**例 12.06** 禁止在 dataGridView1 控件中添加和删除行，可以通过如下代码实现。

```
dataGridView1.AllowUserToAddRows = false; //禁止添加行
dataGridView1.AllowUserToDeleteRows = false; //禁止删除行
dataGridView1.ReadOnly = true; //控件中的数据为只读
```

## 12.7 禁用 DataGridView 控件的自动排序功能

禁用 DataGridView 控件的自动排序功能需要用到其 Columns 对象的 SortMode 属性，该属性用来获取或设置列的排序模式。

语法：public DataGridViewColumnSortMode SortMode { get; set; }

说明：它的属性值为 DataGridViewColumnSortMode 枚举值，指定根据列中单元格的值对行进行排序的条件。DataGridViewColumnSortMode 枚举值及说明如表 12.2 所示。

表 12.2 DataGridViewColumnSortMode 枚举值及说明

枚举值	说明
NotSortable	此列仅能以编程方式进行排序，但由于它原本并不打算排序，所以列标头将不包含排序标志符号的空间
Automatic	除非列标头用于进行选择，否则用户可以通过单击列标头对列进行排序。排序标志符号将自动显示出来
Programmatic	此列仅能以编程方式进行排序，并且列标头将包含排序标志符号的空间

**例 12.07** 创建一个 Windows 应用程序，向默认窗体中添加一个 DataGridView 控件，并在 DataGridView 控件中显示数据表中的数据，然后通过设置其 Columns 对象的 SortMode 属性来禁用 DataGridView 控件的自动排序功能。代码如下。（实例位置：光盘\mr\12\sl\12.07）

```
private void Form1_Load(object sender, EventArgs e)
{
}
```

```

//创建 SqlConnection 连接对象
SqlConnection sqlcon = new SqlConnection("server=MRWXK\WANGXIAOKE;database=db_CSharp;uid=
sa;pwd=");
SqlDataAdapter sqlda = new SqlDataAdapter("select * from tb_emp", sqlcon);
//创建 SqlDataAdapter 桥接器对象
DataSet myds = new DataSet();                                //创建 DataSet 数据集对象
sqlda.Fill(myds);                                         //填充 DataSet 数据集
dataGridView1.DataSource = myds.Tables[0];                   //为 DataGridView 控件设置数据源
//禁用 DataGridView 控件的排序功能
for (int i = 0; i < dataGridView1.Columns.Count; i++)
    dataGridView1.Columns[i].SortMode = DataGridViewColumnSortMode.NotSortable;
}

```

程序运行结果如图 12.5 所示。



图 12.5 禁用 DataGridView 控件的自动排序功能

## 12.8 合并 DataGridView 控件中的单元格

合并 DataGridView 控件中的单元格时，需要用到其 CellPainting 事件，该事件在绘制单元格时触发。本节在合并 DataGridView 控件中的单元格时，主要通过在其 CellPainting 事件中对具有相同数据的单元格进行重绘来实现。

**例 12.08** 创建一个 Windows 应用程序，向默认窗体中添加一个 DataGridView 控件，在该控件中显示数据表中的数据时，通过在其 CellPainting 事件中对表格进行重绘实现数据相同的单元格合并功能，代码如下。（实例位置：光盘\mr\12\sl\12.08）

```

//连接数据库字符串
static string connectionString = "server=MRWXK\WANGXIAOKE;database=db_CSharp;uid=sa;pwd=";
SqlConnection conn = new SqlConnection(connectionString);           //连接数据库
SqlDataAdapter Adapter;                                         //声明 Adapter 对象
DataSet dataSet = new DataSet();                                //声明 dataSet 数据集对象
private void Form1_Load(object sender, EventArgs e)
{
    if (conn.State == ConnectionState.Closed)                      //判断数据库是否关闭
        conn.Open();                                            //打开数据库
    string selectString = "select * from tb_emp";               //定义查询字符串
    Adapter = new SqlDataAdapter(selectString, conn);          //创建填充数据集和更新数据库的对象
    Adapter.Fill(dataSet);                                       //填充 dataSet 数据集
    dataGridView1.DataSource = dataSet.Tables[0];                //为 dataGridView1 设置数据源
    conn.Close();                                              //关闭数据库
}
private void dataGridView1_CellPainting(object sender, DataGridViewCellPaintingEventArgs e)
{

```

```

if (e.ColumnIndex == 2 && e.RowIndex != -1 || e.ColumnIndex == 3 && e.RowIndex != -1 || e.ColumnIndex
== 4 && e.RowIndex != -1) //对第 1 列相同单元格进行合并
{
    Brush datagridBrush = new SolidBrush(dataGridView1.GridColor);
    SolidBrush groupLineBrush = new SolidBrush(e.CellStyle.BackColor);
    using (Pen datagridLinePen = new Pen(datagridBrush))
    {
        e.Graphics.FillRectangle(groupLineBrush, e.CellBounds); //清除单元格
        if (e.RowIndex < dataGridView1.Rows.Count - 1 && dataGridView1.Rows[e.RowIndex + 1].Cells[e.
ColumnIndex].Value != null && dataGridView1.Rows[e.RowIndex + 1].Cells[e.ColumnIndex].Value.ToString() !=
e.Value.ToString())
        {
            e.Graphics.DrawLine(datagridLinePen, e.CellBounds.Left, e.CellBounds.Bottom - 1, e.CellBounds.
Right, e.CellBounds.Bottom - 1); //绘制底边线
            e.Graphics.DrawLine(datagridLinePen, e.CellBounds.Right - 1, e.CellBounds.Top, e.CellBounds.
Right - 1, e.CellBounds.Bottom); //绘制右边线
        }
        else
        {
            e.Graphics.DrawLine(datagridLinePen, e.CellBounds.Right - 1, e.CellBounds.Top, e.CellBounds.
Right - 1, e.CellBounds.Bottom); //绘制右边线
        }
        if (e.RowIndex == dataGridView1.Rows.Count - 1) //对最后一条记录只绘制底边线
        {
            e.Graphics.DrawLine(datagridLinePen, e.CellBounds.Left, e.CellBounds.Bottom - 1, e.CellBounds.
Right, e.CellBounds.Bottom - 1); //绘制底边线
        }
        //填写单元格内容，相同的内容的单元格只填写第一个
        if (e.Value != null)
        {
            if (!(e.RowIndex > 0 && dataGridView1.Rows[e.RowIndex - 1].Cells[e.ColumnIndex].Value.ToString()
== e.Value.ToString()))
            {
                e.Graphics.DrawString(e.Value.ToString(), eCellStyle.Font, Brushes.Black, e.CellBounds.X + 2,
e.CellBounds.Y + 5, StringFormat.GenericDefault); //绘制单元格内容
            }
        }
        e.Handled = true;
    }
}
}

```

程序运行结果如图 12.6 所示。



图 12.6 合并 DataGridView 控件中的单元格

## 12.9 照猫画虎——基本功训练

### 12.9.1 基本功训练 1——设置 DataGridView 控件中网格线的样式

**视频讲解：**光盘\mr\12\lx\设置 DataGridView 控件中网格线的样式.exe

**实例位置：**光盘\mr\12\zmhh\01

新建一个 Windows 窗体应用程序，在窗体中添加一个 DataGridView 控件，用来显示绑定的数据。然后在窗体的代码页中通过设置 DataGridView 控件的 GridColor 属性来设置其网格线样式，代码如下。

```
class Student
{
    public string Name { get; set; } //定义姓名字段
    public int Age { get; set; } //定义年龄字段
}
private void Frm_Main_Load(object sender, EventArgs e)
{
    dgv_Message.GridColor = Color.Blue; //设置网格颜色
    dgv_Message.DataSource = new List<Student>() { //绑定到数据集合
        new Student(){Name="小明",Age=30},
        new Student(){Name="老张",Age=40},
        new Student(){Name="小李",Age=33},
        new Student(){Name="小王",Age=31}};
    dgv_Message.Columns[0].Width = 200; //设置列宽
    dgv_Message.Columns[1].Width = 170; //设置列宽
}
```

程序运行结果如图 12.7 所示。

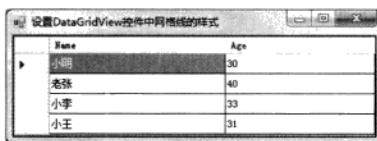


图 12.7 设置 DataGridView 控件中网格线的样式

**照猫画虎：**制作一个程序，将 DataGridView 控件的网格线颜色设置为红色。提示：将 DataGridView 控件的 GridColor 属性设置为 Color.Red 即可。(20 分)(实例位置：光盘\mr\12\zmhh\01\_zmhh)

### 12.9.2 基本功训练 2——在 DataGridView 控件中设置数据显示格式

**视频讲解：**光盘\mr\12\lx\在 DataGridView 控件中设置数据显示格式.exe

**实例位置：**光盘\mr\12\zmhh\02

新建一个 Windows 窗体应用程序，在窗体中添加一个 DataGridView 控件，用来显示绑定的数据。然后在窗体的代码页中通过设置 DataGridView 控件的 DefaultCellStyle.Format 属性来设置指定列的数据显示格式，代码如下。

```
class Fruit
{
}
```

```

public string Name { get; set; } //定义名称字段
public float Price { get; set; } //定义价格字段
}
private void Frm_Main_Load(object sender, EventArgs e)
{
    dgv_Message.DataSource = new List<Fruit>() { //绑定数据集合
        new Fruit(){Name="苹果",Price=30},
        new Fruit(){Name="橘子",Price=40},
        new Fruit(){Name="鸭梨",Price=33},
        new Fruit(){Name="水蜜桃",Price=31}};
    dgv_Message.Columns[0].Width = 200; //设置列宽度
    dgv_Message.Columns[1].Width = 170; //设置列宽度
    dgv_Message.Columns[1].DefaultCellStyle.Format = "c"; //设置内容格式
}

```

程序运行结果如图 12.8 所示。

Name	Price
苹果	\$30.00
橘子	\$40.00
鸭梨	\$33.00
水蜜桃	\$31.00

图 12.8 在 DataGridView 控件中设置数据显示格式

**熊猫画虎：**根据以上程序制作一个程序，将 DataGridView 控件中显示日期的数据列格式化为长日期格式。(20 分)(实例位置：光盘\mr\12\zmhh\02\_zmhh)

### 12.9.3 基本功训练 3——设置 DataGridView 控件单元格的文本对齐方式

视频讲解：光盘\mr\12\lx\设置 DataGridView 控件单元格的文本对齐方式.exe

实例位置：光盘\mr\12\zmhh\03

新建一个 Windows 窗体应用程序，在窗体中添加一个 DataGridView 控件，用来显示绑定的数据。然后在窗体的代码页中通过设置 DataGridView 控件的 DefaultCellStyle.Alignment 属性来设置指定列的文本对齐方式，代码如下。

```

class Fruit
{
    public string Name { get; set; } //定义名称字段
    public float Price { get; set; } //定义价格字段
}
private void Form1_Load(object sender, EventArgs e)
{
    dgv_Message.DataSource = new List<Fruit>() { //绑定数据集合
        new Fruit(){Name="苹果",Price=30},
        new Fruit(){Name="橘子",Price=40},
        new Fruit(){Name="鸭梨",Price=33},
        new Fruit(){Name="水蜜桃",Price=31}};
    dgv_Message.Columns[0].Width = 200; //设置列宽度
    dgv_Message.Columns[1].Width = 170; //设置列宽度
    dgv_Message.Columns[0].DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter; //设置文本对齐方式
}

```

```
//设置对齐方式
}
```

程序运行结果如图 12.9 所示。



图 12.9 设置 DataGridView 控件单元格的文本对齐方式

**照猫画虎：**完善以上程序，将 DataGridView 控件中所有列的数据都设置为居中对齐。(20 分)(实例位置：光盘\mr\12\zmhh\03\_zmhh)

#### 12.9.4 基本功训练 4——在 DataGridView 控件中实现下拉列表

■ 视频讲解：光盘\mr\12\lx\在 DataGridView 控件中实现下拉列表.exe

■ 实例位置：光盘\mr\12\zmhh\04

新建一个 Windows 窗体应用程序，在窗体中添加一个 DataGridView 控件，用来显示绑定的数据。然后在窗体的代码页中创建一个 DataGridViewComboBoxColumn 对象，并使用 DataGridView 控件的 Columns.Add 方法将其添加到 DataGridView 控件中，从而实现在 DataGridView 控件中显示下拉列表的功能。代码如下。

```
class Fruit
{
    public string Name { get; set; } //定义名称字段
    public float Price { get; set; } //定义价格字段
}
private void Frm_Main_Load(object sender, EventArgs e)
{
    DataGridViewComboBoxColumn dgvc = new DataGridViewComboBoxColumn(); //创建列对象
    dgvc.Items.Add("苹果"); //向集合中添加元素
    dgvc.Items.Add("芒果"); //向集合中添加元素
    dgvc.Items.Add("鸭梨"); //向集合中添加元素
    dgvc.Items.Add("橘子"); //向集合中添加元素
    dgvc.HeaderText = "水果"; //设置列标题文本
    dgv_Message.Columns.Add(dgvc); //将列添加到集合
}
```

程序运行结果如图 12.10 所示。

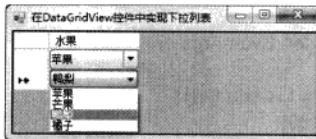


图 12.10 在 DataGridView 控件中实现下拉列表

**照猫画虎：**使用窗体设计器实现与本实例一样的功能。提示：单击 DataGridView 控件右上角的小箭头，然后在弹出的对话框中进行添加，注意选择 DataGridViewComboBoxColumn。(20 分)(实例位置：光盘\mr\12\zmhh\04\_zmhh)

### 12.9.5 基本功训练 5——为 DataGridView 控件实现复选功能

视频讲解：光盘\mr\12\lx\为 DataGridView 控件实现复选功能.exe

实例位置：光盘\mr\12\zmhh\05

新建一个 Windows 窗体应用程序，在窗体中添加一个 DataGridView 控件，用来显示绑定的数据。然后在窗体的代码页中创建一个 DataGridViewCheckBoxColumn 对象，并使用 DataGridView 控件的 Columns.Add 方法将其添加到 DataGridView 控件中，从而实现在 DataGridView 控件中显示复选框的功能。代码如下。

```

class Fruit
{
    public string Name { get; set; } //定义名称字段
    public float Price { get; set; } //定义价格字段
    public bool ft; //定义删除标识
}

private List<Fruit> P_Fruit;
private void Frm_Main_Load(object sender, EventArgs e)
{
    DataGridViewCheckBoxColumn dgvc = new DataGridViewCheckBoxColumn(); //创建列对象
    dgvc.HeaderText = "状态"; //设置列标题文本
    dgv_Message.Columns.Add(dgvc); //添加列
    P_Fruit = new List<Fruit>(); //创建数据集合
    new Fruit(){Name="苹果",Price=30};
    new Fruit(){Name="橘子",Price=40};
    new Fruit(){Name="鸭梨",Price=33};
    new Fruit(){Name="水蜜桃",Price=31};
    dgv_Message.DataSource = P_Fruit; //绑定数据集合
    dgv_Message.Columns[0].Width = 50; //设置列宽度
    dgv_Message.Columns[1].Width = 170; //设置列宽度
    dgv_Message.Columns[2].Width = 150; //设置列宽度
}

```

说明：本实例还实现了删除选中复选框记录的功能，其详细代码可参见本书配套光盘中的源代码。

程序运行结果如图 12.11 所示。



图 12.11 为 DataGridView 控件实现复选功能

照猫画虎：使用窗体设计器实现与本实例一样的功能。提示：单击 DataGridView 控件右上角的小箭头，然后在弹出的对话框中进行添加，注意选择 DataGridViewCheckBoxColumn。(20 分)(实例位置：光盘\mr\12\zmhh\05\_zmhh)

照猫画虎栏目分数统计：

照猫画虎题目	1	2	3	4	5	总分数	
分数							

## 12.10 情景应用——拓展与实践

### 12.10.1 情景应用 1——在 DataGridView 控件中验证数据输入

■ 视频讲解：光盘\mr\12\lx\在 DataGridView 控件中验证数据输入.exe

■ 实例位置：光盘\mr\12\qjyy\01

在 DataGridView 控件的单元格中可以编辑数据，但有时需要控制单元格输入的数据类型，这里主要实现验证 DataGridView 控件的第二列的单元格中只允许输入数字的功能。新建一个 Windows 窗体应用程序，在窗体中添加一个 DataGridView 控件，用来显示绑定的数据和编辑单元格的数据。验证单元格中的输入是否为数字的功能是在 DataGridView 控件的 CellValidating 事件中实现的，代码如下。

```
private void dataGridView1_CellValidating(object sender, DataGridViewCellValidatingEventArgs e)
{
    if (e.ColumnIndex == 1) //验证指定列
    {
        float result = 0;
        if (!float.TryParse(e.FormattedValue.ToString(), out result)) //判断数据是否为数值类型
        {
            dgv_Message.Rows[e.RowIndex].ErrorText = "内容必须为数值类型"; //提示错误信息
            e.Cancel = true; //取消事件的值
        }
    }
}
```

程序运行结果如图 12.12 所示。

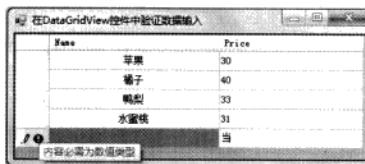


图 12.12 在 DataGridView 控件中验证数据输入

DIY：根据以上程序，制作一个验证 DataGridView 控件的指定单元格中数据是否为电话号码格式的程序。提示：可以使用 Regex 类的 IsMatch 方法，并结合相应的正则表达式实现。(20 分)(实例位置：光盘\mr\12\qjyy\01\_diy)

### 12.10.2 情景应用 2——在 DataGridView 控件中添加“合计”和“平均值”

■ 视频讲解：光盘\mr\12\lx\在 DataGridView 控件中添加“合计”和“平均值”.exe

■ 实例位置：光盘\mr\12\qjyy\02

本实例实现为 DataGridView 控件中的第一列的所有行求和和第二个列的所有行求平均数的功能。新建

一个 Windows 窗体应用程序，在窗体中添加一个 DataGridView 控件，用来显示绑定的数据和计算所得的合计值、平均值，然后在窗体的代码页中编写如下代码。

```

class Fruit
{
    public string Name { get; set; }
    public float Price { get; set; }
}

private List<Fruit> G_Fruit;
private void Frm_Main_Load(object sender, EventArgs e)
{
    G_Fruit = new List<Fruit>()
    {
        new Fruit(){Name="苹果",Price=30}, //定义名称字段
        new Fruit(){Name="橘子",Price=40}, //定义价格字段
        new Fruit(){Name="鸭梨",Price=33},
        new Fruit(){Name="水蜜桃",Price=31}};
        dgv_Message.Columns.Add("Fruit", "水果"); //添加列
        dgv_Message.Columns.Add("Pric", "价格"); //添加列
        foreach (Fruit f in G_Fruit) //添加元素
        {
            dgv_Message.Rows.Add(new string[]
            {
                f.Name,
                f.Price.ToString()
            });
        }
        dgv_Message.Columns[0].Width = 200; //设置列宽度
        dgv_Message.Columns[1].Width = 170; //设置列宽度
        float sum = 0; //定义 float 类型变量
        G_Fruit.ForEach(
            (pp) =>
            {
                sum += pp.Price; //求和
            });
        dgv_Message.Rows.Add(new string[] //在新列中显示平均值及合计信息
        {
            "合计: "+sum.ToString()+" 元",
            "平均价格: "+(sum/G_Fruit.Count).ToString()+" 元"
        });
}

```

程序运行结果如图 12.13 所示。

在 DataGridView 控件中添加“合计”和“平均值”	
水果	价格
苹果	30
橘子	40
鸭梨	33
水蜜桃	31
合计: 134 元	平均值: 33.5 元

图 12.13 在 DataGridView 控件中添加“合计”和“平均值”

**DIY:** 完善以上程序，使计算完成的“合计”和“平均值”显示为货币格式。提示：使用 `ToString("C")` 方法实现。(20 分)(实例位置：光盘\mr\12\qjyy\02\_diy)

### 12.10.3 情景应用 3——使用交叉表实现商品销售统计

视频讲解：光盘\mr\12\lx\使用交叉表实现商品销售统计.exe

实例位置：光盘\mr\12\qjyy\03

在进行数据分析时，使用交叉表可以清晰、直观地反映出数据之间的关系，这里使用交叉表实现了统计商品销售信息的功能。新建一个 Windows 窗体应用程序，在窗体中添加两个 ComboBox 控件，分别用来选择表头字段和分组字段；添加一个 Button 控件，用来使用交叉表统计商品销售信息；添加一个 DataGridView 控件，用来显示商品销售统计信息。使用交叉表实现商品销售统计的功能主要是在自定义方法 bindInfo 中实现的，代码如下。

```
protected void bindInfo()
{
    //创建数据库连接对象
    SqlConnection sqlcon = new SqlConnection("Data Source=MRWXK\WANGXIAOKE;Database=db_CSharp;
    Uid=sa;Pwd=");
    SqlCommand sqlcom = new SqlCommand("proc_across_table", sqlcon);           //创建 SqlCommand 对象
    sqlcom.CommandType = CommandType.StoredProcedure;                            //指定执行存储过程
    //为存储过程添加参数
    sqlcom.Parameters.Add("@TableName", SqlDbType.VarChar, 50).Value = "商品销售表";
    if (comboBox1.Text == comboBox2.Text)
    {
        MessageBox.Show("表头字段和分组字段不能相同！", "警告", MessageBoxButtons.OK, MessageBoxIcon.Warning);
        return;
    }
    sqlcom.Parameters.Add("@NewColumn", SqlDbType.VarChar, 50).Value = comboBox1.Text;
    sqlcom.Parameters.Add("@GroupColumn", SqlDbType.VarChar, 50).Value = comboBox2.Text;
    sqlcom.Parameters.Add("@StatColumn", SqlDbType.VarChar, 50).Value = "订货数量";
    sqlcom.Parameters.Add("@Operator", SqlDbType.VarChar, 10).Value = "SUM";
    SqlDataAdapter myda = new SqlDataAdapter();                                //创建 SqlDataAdapter 对象
    myda.SelectCommand = sqlcom;
    DataSet myds = new DataSet();                                              //创建数据集对象
    myda.Fill(myds);                                                        //填充数据集
    dataGridView1.DataSource = myds.Tables[0];                                  //为 DataGridView 设置数据源
    dataGridView1.Columns[1].Width = 120;                                       //设置列宽
}
```

本实例中用到的 proc\_across\_table 存储过程代码如下。

```
CREATE procedure proc_across_table
@TableName as varchar(50),
@NewColumn as varchar(50),
@GroupColumn as varchar(50),
@StatColumn as varchar(50),
@Operator as varchar(10)
AS
DECLARE @SQL as varchar(1000), @Column as varchar(50)
EXECUTE ('DECLARE cursor_new_column CURSOR FOR SELECT DISTINCT ' + @NewColumn + ' from ' +
@TableName + ' for read only')
begin
```

--生成交叉表依据的表名

--生成表头依据的字段名

--分组依据的字段名

--欲统计的字段名

--统计的运算方式

--定义参数

--定义游标

```

SET nocount ON
--定义 SQL 语句头
SET @SQL='select ' + @GroupColumn + ',' + @Operator + '[' + @StatColumn + ') AS [' + @Operator + 'of' +
@StatColumn + ']'
OPEN cursor_new_column
while (0=0)
BEGIN
    FETCH NEXT FROM cursor_new_column INTO @Column
    if (@@fetch_status<>0) break
        SET @SQL = @SQL + ',' + @Operator + '(CASE ' + @NewColumn + ' WHEN "' + @Column + '" THEN
' + @StatColumn + ' ELSE Null END) AS [' + @Column + ']'
    END
    SET @SQL = @SQL + ' from ' + @TableName + ' group by ' + @GroupColumn --定义 SQL 语句尾
    EXECUTE(@SQL) --执行 SQL 语句
    PRINT @SQL --输出 SQL 语句
    IF @@error <>0 RETURN @@error --如果出错，则返回错误代码
    CLOSE cursor_new_column --关闭游标
    DEALLOCATE cursor_new_column RETURN 0 --释放游标，若释放成功，则返回 0
end
GO

```

**说明：**这是一个通用的存储过程，只要正确地传入生成交叉表依据的表名（@TableName）、生成表头依据的字段名（@NewColumn）、生成主键列依据的字段名（@GroupColumn）、将统计的字段名（@StatColumn）和统计的运算方式（@Operator），就可以成功地将其应用到任何数据表中。

程序运行结果如图 12.14 所示。

商品名	STW	订货数量	100001	100002	100003
沙发	10		10		
茶几	4				
床	20		20		
电视	200	100	100		
音响	40	30	10		
风扇	45			45	
玻璃	34		34		
沙发	140	80	80		
西餐	54			54	

图 12.14 使用交叉表实现商品销售统计

**DIY：**根据上面程序的实现原理制作一个程序，用来使用动态交叉表实现公司员工的出生月份统计。

**提示：**通过调用 db\_CSharp 数据库中的 proc\_across\_table\_new 存储过程实现。（20 分）（实例位置：光盘\mr\12\qjyy\03\_diy）

#### 12.10.4 情景应用 4——将 DataGridView 中数据导出到 Word

**视频讲解：**光盘\mr\12\lx\将 DataGridView 中数据导出到 Word.exe

**实例位置：**光盘\mr\12\qjyy\04

Microsoft Word 具有很强的文档处理功能，将 C# 程序与 Word 文档相结合就能够设计出更加强大的应用

程序。本实例通过使用 Microsoft Word 自动化对象模型中的 Cell 对象，将 DataGridView 控件中的数据导出到 Word 文档表格中。运行效果如图 12.15 所示。

实现过程如下。

(1) 创建一个 Windows 窗体应用程序，并将其命名为 GridToWord。

(2) 更改默认窗体 Form1 的 Name 属性为 Frm\_Main，在该窗体中添加一个 DataGridView 控件，用来显示数据；添加一个 Button 控件，用来把 DataGridView 控件中的数据导出到 Word 文档。

(3) 在窗体的 Load 事件中，首先绑定 DataGridView 控件到数据集合，代码如下。

```
private void Frm_Main_Load(object sender, EventArgs e)
{
    dgv_Message.DataSource = new List<Fruit>() { //绑定数据集合
        new Fruit(){Name="苹果",Price=30},
        new Fruit(){Name="橘子",Price=40},
        new Fruit(){Name="鸭梨",Price=33},
        new Fruit(){Name="水蜜桃",Price=31}};
    dgv_Message.Columns[0].Width = 200; //设置列宽度
    dgv_Message.Columns[1].Width = 170; //设置列宽度
}
```

(4) 单击窗体中的“导出到 Word 文档”按钮，将 DataGridView 控件中的数据导出到 Word 文档，代码如下。

```
private void btn_OutPut_Click(object sender, EventArgs e)
{
    List<Fruit> P_Fruit = new List<Fruit>(); //创建数据集合
    foreach (DataGridViewRow dgvr in dgv_Message.Rows)
    {
        P_Fruit.Add(new Fruit()); //向数据集合中添加数据
        {
            Name = dgvr.Cells[0].Value.ToString(),
            Price = Convert.ToSingle(dgvr.Cells[1].Value.ToString())
        };
    }
    SaveFileDialog P_SaveFileDialog = new SaveFileDialog(); //创建保存文件对话框对象
    P_SaveFileDialog.Filter = "*.doc|*.doc";
    if (DialogResult.OK == P_SaveFileDialog.ShowDialog()) //确认是否保存文件
    {
        ThreadPool.QueueUserWorkItem( //开始线程池
            (pp) => //使用 Lambda 表达式
            {
                G_wa = new Microsoft.Office.Interop.Word.Application(); //创建应用程序对象
                object P_obj = "Normal.dot"; //定义文档模板
                Word.Document P_wd = G_wa.Documents.Add( //向 Word 应用程序中添加文档
                    ref P_obj, ref G_missing, ref G_missing, ref G_missing);
                Word.Range P_Range = P_wd.Range(ref G_missing, ref G_missing); //得到文档范围
                object o1 = Word.WdDefaultTableBehavior.wdWord8TableBehavior; //设置文档中表格格式
                object o2 = Word.WdAutoFitBehavior.wdAutoFitWindow; //设置文档中表格格式
            });
    }
}
```

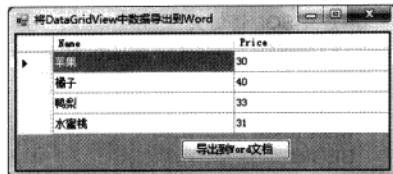


图 12.15 将 DataGridView 中数据导出到 Word

```

Word.Table P_WordTable = P_Range.Tables.Add(P_Range, //在文档中添加表格
    P_Fruit.Count ,2, ref o1, ref o2);
P_WordTable.Cell(1, 1).Range.Text = "水果"; //向表格中添加信息
P_WordTable.Cell(1, 2).Range.Text = "价格"; //向表格中添加信息
for (int i = 2; i < P_Fruit.Count + 1; i++)
{
    P_WordTable.Cell(i, 1).Range.Text = P_Fruit[i - 2].Name; //向表格中添加信息
    P_WordTable.Cell(i, 2).Range.Text = P_Fruit[i - 2].Price.ToString(); //向表格中添加信息
}
object P_Path = P_SaveFileDialog.FileName;
P_wd.SaveAs( //保存 Word 文件
    ref P_Path,
    ref G_missing, ref G_missing, ref G_missing, ref G_missing,
    ref G_missing, ref G_missing, ref G_missing, ref G_missing,
    ref G_missing, ref G_missing, ref G_missing, ref G_missing,
    ref G_missing, ref G_missing, ref G_missing);
(Word.Application)G_wa.Application.Quit( //退出应用程序
    ref G_missing, ref G_missing, ref G_missing);
this.Invoke( //调用窗体线程
    (MethodInvoker)((() =>
    {
        MessageBox.Show("成功创建 Word 文档！", "提示！"); //弹出消息对话框
    })));
}
}

```

 **说明:** 本实例实现时, 首先需要添加 Microsoft Word 11.0 Object Library 引用。

DIY：根据以上程序的实现原理，实现将 DataGridView 控件中数据导出到 Excel 中的功能。提示：通过添加 Microsoft Excel 11.0 Object Library 引用实现。（20 分）（实例位置：光盘\mr\12\qjv\04\div）

#### 12.10.5 情景应用 5——通过 DataGridView 分页查看用户信息

 视频讲解：光盘\mr\12\lx\通过 DataGridView 分页查看用户信息.exe

 实例位置：光盘\mr\12\qjyy\05

上网的读者可能经常在网络中搜索信息，搜索过程中由于信息量很大，所以查询是以分页的形式呈现出来的，同时会将“总页数”及“当前页”提示给用户，这样可以使用户更快速地查找到相关信息。这里将通过 DataGridView 分页查看用户信息，运行效果如图 12.16 所示。

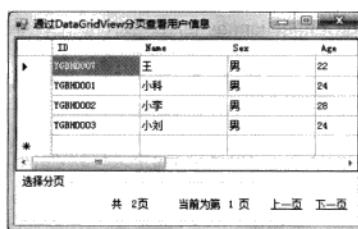


图 12.16 通过 DataGridView 分页查看用户信息

实现过程如下。

(1) 创建一个 Windows 窗体应用程序，并将其命名为 UsePagination。

(2) 更改默认窗体 Form1 的 Name 属性为 Frm\_Main，在该窗体中添加 6 个 Label 控件，分别用于显示页数索引、总页数和移动到指定分页；添加一个 DataGridView 控件，用于显示分页信息。

(3) 在窗体代码页中，首先定义 3 个 int 类型的变量，分别用来记录当前页、总页数和每页记录数，代码如下。

```
public static int lNum = 0, AllCount = 0; //定义静态字段
int Sizes = 4; //每页记录数量

(4) 在窗体加载时，获取数据库中的所有记录，并根据每页显示的记录数计算总页数，代码如下。
private void Form1_Load(object sender, EventArgs e)
{
    //创建数据库连接对象
    using (SqlConnection con = new SqlConnection(@"server=MRWXKWANGXIAOKE;pwd=;uid=sa;database=db_CSharp"))
    {
        SqlDataAdapter da = new SqlDataAdapter("select * from tb_Employee", con); //创建数据适配器对象
        DataTable dt = new DataTable(); //创建数据表对象
        da.Fill(dt); //填充数据表
        int i = dt.Rows.Count; //得到记录数量
        AllCount = i; //得到记录数量
        int m = i % Sizes; //取模运算
        if (m == 0)
        {
            m = i / Sizes; //计算记录页数
        }
        else
        {
            m = i / Sizes + 1; //计算记录页数
        }
        this.label3.Text = m.ToString(); //显示记录页数
        show(0, 4); //显示数据记录
        this.label4.Text = "1"; //显示当前页数
    }
}
```

(5) 在上面的代码中用到了 show 方法，该方法为自定义的无返回值类型方法，用来查询并显示指定的数据记录，代码如下。

```
private void show(int i, int j)
{
    //创建数据库连接对象
    SqlConnection con = new SqlConnection(@"server=MRWXKWANGXIAOKE;pwd=;uid=sa;database=db_CSharp");
    SqlDataAdapter daone = new SqlDataAdapter("select * from tb_Employee", con); //创建数据适配器对象
    DataSet dsone = new DataSet(); //创建数据集
    daone.Fill(dsone, i, j, "one"); //填充数据集
    this.dataGridView1.DataSource = dsone.Tables["one"].DefaultView; //设置数据源
}
```

(6) 单击“上一页”和“下一页”超链接，分别根据当前页的索引计算上一页或下一页的页数，并调用 show 方法显示相应的记录，代码如下。

```

private void linkLabel3_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)
{
    this.dataGridView1.DataSource = null; //设置数据源
    int m = Convert.ToInt32(this.label4.Text) - 1; //得到页数索引
    if (m < 1)
    {
        this.label4.Text = "1"; //显示当前页数
    }
    else
    {
        this.label4.Text = m.ToString(); //显示当前页数
    }
    int a = Convert.ToInt32(this.label4.Text) * 4 - 4; //得到记录数索引
    show(a, 4); //显示数据记录
}
private void linkLabel4_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)
{
    this.dataGridView1.DataSource = null; //设置数据源
    int m = Convert.ToInt32(this.label4.Text) + 1; //得到页数索引
    if (m > Convert.ToInt32(this.label3.Text))
    {
        this.label4.Text = this.label3.Text; //显示当前页数
    }
    else
    {
        this.label4.Text = m.ToString(); //显示当前页数
    }
    int a = Convert.ToInt32(this.label4.Text) * 4 - 4; //得到记录数索引
    show(a, 4); //显示数据记录
}

```

**DIY:** 完善以上程序，为其增加“首页”和“末尾页”功能。(20分)(实例位置：光盘\mr\12\qjyy\05\_diy)

情景应用 DIY 栏目分数统计：

DIY 题目	1	2	3	4	5	总分数	
分数							

## 12.11 自我测试

### 一、选择题(每题 10 分, 5 道题)

- 以下表示 DataGridView 控件的是( )。
 

A. B. C. D.
- 如果要在 DataGridView 控件中显示超链接，则需要使用( )对象。
 

A. DataGridViewComboBoxColumn B. DataGridViewCheckBoxColumn  
C. DataGridViewLinkColumn D. DataGridViewButtonColumn

3. 使 DataGridView 控件同时显示垂直和水平滚动条时, 需要将其 ScrollBars 属性设置为( )。  
 A. Both      B. Horizontal      C. Vertical      D. None
4. 设置 DataGridView 控件的背景颜色时, 需要设置其( )属性。  
 A. BackImage      B. BackColor      C. BackgroundColor      D. BackgroundImage
5. 如果要使 DataGridView 控件中能够选择多行, 则( )。  
 A. 将其 MultiSelect 属性设置为 False      B. 将其 MultiSelect 属性设置为 True  
 C. 将其 Enabled 属性设置为 False      D. 将其 Enabled 属性设置为 True

## 二、填空题 (每题 10 分, 5 道题)

1. 如果要为 DataGridView 控件设置数据源, 则需要设置其( )属性。  
 2. 如果要在 DataGridView 控件中显示下拉列表, 则需要使用( )对象。  
 3. 如果要在 DataGridView 控件中显示复选框, 则需要使用( )对象。  
 4. 如果要将 DataGridView 控件中的数据设置为只读, 则需要设置其 ReadOnly 属性为( )。  
 5. 当选择 DataGridView 控件中的某个单元格时, 如果要将其整行选中, 则应将其 SelectionMode 属性设置为( )。

测试分数统计:

类别	第 1 题	第 2 题	第 3 题	第 4 题	第 5 题
选择题分数					
填空题分数					
总分数					

## 12.12 行动指南

开始日期: \_\_\_\_\_ 年 \_\_\_\_\_ 月 \_\_\_\_\_ 日

结束日期: \_\_\_\_\_ 年 \_\_\_\_\_ 月 \_\_\_\_\_ 日

序号	内 容	行动指南	
1	照猫画虎栏目	分数>75 分	优秀, 基本功掌握得不错, 加油!
		75 分>分数>50 分	及格, 知识掌握得不牢, 重新做一遍照猫画虎。
	情景应用栏目	分数<50 分	请使用光盘中提供的“实战能力测试系统”进行提高训练。
		分数>75 分	优秀, 综合应用能力很强。
	分数 ( )	75 分>分数>50 分	及格, 综合应用能力需提高, 再练一遍情景应用。
		分数<50 分	请使用光盘中提供的“实战能力测试系统”进行提高训练。
	自我测试栏目	分数>75 分	优秀, 有成为编程高手的潜质。
		分数<75 分	请使用光盘中提供的“实战能力测试系统”进行提高训练。
2	综合评价	反复训练后, 上以上各项分数都在 75 分以上, 方可进入下一堂课学习。	
	编程能力培养: 本栏目提供了一些实践题目, 对于培养编程能力很有效, 要做好。	1. 美化 DataGridView 控件, 使得其在显示数据时, 奇偶行显示不同的颜色。 2. 开发一个程序, 主要实现从 DataGridView 控件中拖放数据到 TreeView 控件进行显示的功能。 3. 制作一个 DataGridView 数据复制器, 使得 DataGridView 控件中的数据能够像 Excel 一样进行复制。	

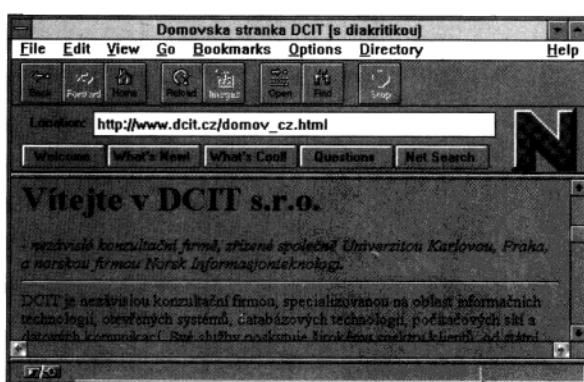
续表

序号	内 容	行动指南
3	编程习惯培养：本堂课培养读者在学习开发中记笔记的习惯，把开发中遇到的问题、总结的经验记录下来。	
4	创新能力培养：看看身边有没有可以用编程解决的问题，记录到右边的表格中。根据学习进度，尝试编写解决这些问题的小程序。	

## 12.13 成功可以复制——因特网的点火人马克·安德森

安德森出生在威斯康星州一个小镇的普通家庭，9岁开始接触计算机，在图书馆自学Basic语言。当升入六年级时，安德森已经创造出一个替他做数学家庭作业的虚拟计算器。七年级时，他自编出一个在自家计算机上玩的游戏程序。1993年，刚刚大学毕业的安德森还没有找到更好的工作，于是就和几个志同道合的朋友一起编写Internet浏览软件。早期的万维网非常简陋，没有图像、声音和色彩，只是一个基于文本之上、仅仅通过原始界面才能进入的网络。安德森在使用万维网颇感不便的同时，敏锐地察觉到开发一个带有图形、易于客户使用的网页浏览器的潜在市场。经过不断探索和坚持，1994年4月，安德森带领他的工作组开发成功“Navigator（领航员）”浏览器，随后该浏览器的销售在Internet上如“凯歌般地行进”，一下就占据了80%以上的份额。

1995年8月9日，互联网时代到来了，成立还不到16个月、从未赢利过的Netscape公司在纽约上市。投资银行事先估计每股仅能卖14美元左右，然而开盘后股价一路飙升，最高时竟达到让全美经纪人都目瞪口呆的71美元，两个小时内500万股被抢购一空，这家创始资金只有400



Navigator 1.0 浏览器

万美元的小公司一夜之间便成为20亿美元的巨人。华尔街日报评论说，通用动力公司花了43年才使市值达到27亿美元，而网景只花了1分钟。年仅24岁的安德森也仿佛神话般地从一文不名到拥有1.74亿美元身价的“Internet富翁”。1997年7月的美国《旗帜》周刊更是把安德森们称为“无限制资本家”，他们正从根本上重塑着美国社会，推动着从工业经济向信息经济的过渡。

 经典语录

在互联网的光芒下，操作系统只不过是一套“漏洞重重的设备驱动器”。

 深度评价

马克·安德森的成功启示我们，作为编程开发人员，要善于从生活、工作中发现机会，并坚持把一个小机会做到完美，就可以成就大事业。编程知识不在于多，而在于把学到的知识应用于实践，改善我们工作、学习中的低效率或不便。

# 第3部分

## 高级篇

- ▶ 第 13 堂课 面向对象编程高级技术
- ▶ 第 14 堂课 LINQ 技术的使用
- ▶ 第 15 堂课 文件及 IO
- ▶ 第 16 堂课 GDI+绘图技术
- ▶ 第 17 堂课 水晶报表与打印
- ▶ 第 18 堂课 网络编程
- ▶ 第 19 堂课 线程的使用
- ▶ 第 20 堂课 异常处理与程序调试
- ▶ 第 21 堂课 Windows 应用程序打包部署



# 第13堂课

## 面向对象编程高级技术

( 视频讲解：100分钟)

本堂课将介绍面向对象编程中的几种比较高级的技术，主要包括接口、抽象类和密封类等，这些内容相对于前面章节中所讲的内容更复杂，但为了能够使开发人员开发出结构良好、组织严密、扩展性好及运行稳定的程序，它们又是必不可少的。

学习摘要：

- ▶ 了解接口的基本概念
- ▶ 掌握接口的声明及使用
- ▶ 掌握接口的多重继承的实现
- ▶ 熟悉显式接口成员的实现方法
- ▶ 了解抽象类的基本概念
- ▶ 掌握抽象方法的使用
- ▶ 掌握抽象类的声明及使用方法
- ▶ 了解抽象类与接口的区别
- ▶ 了解密封类的基本概念
- ▶ 掌握密封方法的使用
- ▶ 掌握密封类的声明及使用方法

## 13.1 接口的声明及实现

由于 C# 中的类不支持多重继承，但在客观世界中出现多重继承的情况又比较多，所以为了避免传统的多重继承给程序带来的复杂性等问题，同时保证多重继承带给程序员的诸多好处，C# 中提出了接口概念，通过接口可以实现多重继承的功能。本节将对接口的声明及实现进行详细讲解。

### 13.1.1 接口概述

接口是一种用来定义程序的协议，其描述可属于任何类或结构的一组相关行为。它由方法、属性、事件和索引器或这 4 种成员类型的任何组合构成，但不能包含字段。

类和结构可以像类继承基类或结构一样从接口继承，但是可以继承多个接口。当类或结构继承接口时，它继承成员定义但不继承实现。若要实现接口成员，类中的对应成员必须是公共的、非静态的，并且与接口成员具有相同的名称和签名。类的属性和索引器可以为接口中定义的属性或索引器定义额外的访问器。例如，接口可以声明一个带有 get 访问器的属性，而实现该接口的类可以声明同时带有 get 和 set 访问器的同一属性。但是，如果属性或索引器使用显式实现，则访问器必须匹配。

接口可以继承其他接口，类可以通过其继承的基类或接口多次继承某个接口，在这种情况下，如果将该接口声明为新类的一部分，则类只能实现该接口一次。如果没有将继承的接口声明为新类的一部分，其实现将由声明它的基类提供。基类可以使用虚拟成员实现接口成员；在这种情况下，继承接口的类可通过重写虚拟成员来更改接口行为。

综上所述，接口具有以下特征。

- 接口类似于抽象基类：继承接口的任何非抽象类型都必须实现接口的所有成员。
- 不能直接创建接口。
- 接口可以包含事件、索引器、方法和属性。
- 接口不包含方法的实现。
- 类和结构可从多个接口继承。
- 接口自身可从多个接口继承。

### 13.1.2 接口的声明

在 C# 中声明接口时使用 interface 关键字，其语法格式如下。

修饰符 interface 接口名称: 继承的接口列表

```
{
    接口内容;
}
```

 说明：① 声明接口时，除 interface 关键字和接口名称外，其他的都是可选项。

② 可以使用 new、public、protected、internal 和 private 等修饰符声明接口，但接口成员必须是公共的。

**例 13.01** 下面代码声明了一个接口，该接口中包含编号和姓名两个属性，还包含一个自定义方法 ShowInfo，该方法用来显示定义的编号和属性。代码如下。

```
interface IMyInterface
{
    string ID           //编号（可读可写）
    {
        get;
        set;
    }
    string Name         //姓名（可读可写）
    {
        get;
        set;
    }
    void ShowInfo();    //显示定义的编号和姓名
}
```

### 13.1.3 接口的实现与继承

接口的实现通过类继承来实现，一个类虽然只能继承一个基类，但可以继承任意多个接口。声明实现接口的类时，需要在基类列表中包含类所实现的接口的名称。

**例 13.02** 创建一个控制台应用程序，该程序在例 13.01 的基础上实现，Program 类继承自接口 IMyInterface，并实现了该接口中的所有属性和方法，然后在 Main 方法中创建 Program 类的一个对象，并使用该对象创建 IMyInterface 接口，最后通过创建的接口对象访问派生类中的属性和方法。程序代码如下。（实例位置：光盘\mr\13\s\13.02）

```
class Program:IMyInterface           //继承自接口
{
    string id = "";
    string name = "";
    public string ID                  //表示编号的属性
    {
        get
        {
            return id;
        }
        set
        {
            id = value;
        }
    }
    public string Name                //表示姓名的属性
    {
        get
        {
            return name;
        }
        set
        {
            name = value;
        }
    }
}
```

```

public void ShowInfo()           //显示定义的编号和姓名
{
    Console.WriteLine("编号\t      姓名");
    Console.WriteLine(ID + "\t" + Name);
}
static void Main(string[] args)
{
    Program program = new Program(); //创建 Program 对象
    IMyInterface imyinterface = program; //使用派生对象创建接口 IMyInterface
    imyinterface.ID = "明日科技"; //为派生类中的 ID 属性赋值
    imyinterface.Name = "C#从基础到项目实战"; //为派生类中的 Name 属性赋值
    imyinterface.ShowInfo(); //调用派生类中方法显示定义的属性值
    Console.ReadLine();
}
}

```

按 Ctrl+F5 键查看运行结果，如图 13.1 所示。

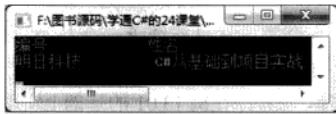


图 13.1 接口的实现实例运行结果图

#### 13.1.4 显式接口成员实现

如果类实现两个接口，并且这两个接口包含具有相同签名的成员，那么在类中实现该成员将导致两个接口都使用该成员作为它们的实现。然而，如果两个接口成员实现不同的功能，那么这可能会导致其中一个接口的实现不正确或两个接口的实现都不正确，这时可以显式地实现接口成员，即创建一个仅通过该接口调用并且特定于该接口的类成员。显式接口成员实现是使用接口名称和一个句点命名该类成员来实现的。

**例 13.03** 创建一个控制台应用程序，其中声明了两个接口 `IMyInterface1` 和 `IMyInterface2`，在这两个接口中声明了一个同名方法 `Add`，然后定义一个类 `MyClass`，该类继承自己已经声明的两个接口，在 `MyClass` 类中实现接口中的方法时，由于 `IMyInterface1` 和 `IMyInterface2` 接口中声明的方法名相同，所以这里使用了显式接口成员实现，最后在主程序类 `Program` 的 `Main` 方法中使用接口对象调用接口中定义的方法。程序代码如下。（实例位置：光盘\mr\13\sl\13.03）

```

interface IMyInterface1
{
    int Add(); //求和方法
}
interface IMyInterface2
{
    int Add(); //求和方法
}
class MyClass : IMyInterface1, IMyInterface2
{
    //接口 IMyInterface1 的求和方法
    int IMyInterface1.Add() //显式接口成员实现
    {

```

```

int x = 3;
int y = 5;
return x + y;
}
//接口 IMyInterface2 的求和方法
int IMyInterface2.Add() //显式接口成员实现
{
    int x = 3;
    int y = 5;
    int z = 7;
    return x + y + z;
}
}
class Program
{
    static void Main(string[] args)
    {
        myClass myclass = new myClass(); //创建接口继承类的对象
        IMyInterface1 imyinterface1 = myclass; //使用接口继承类的对象创建接口
        Console.WriteLine(imyinterface1.Add()); //使用接口对象调用接口中方法
        IMyInterface2 imyinterface2 = myclass; //使用接口继承类的对象创建接口
        Console.WriteLine(imyinterface2.Add()); //使用接口对象调用接口中方法
        Console.ReadLine();
    }
}

```

程序运行结果如图 13.2 所示。

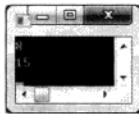


图 13.2 运行结果

注意：① 显式接口成员实现中不能包含访问修饰符、abstract、virtual、override 或 static 修饰符。

② 由于显式接口成员是属于接口的成员，而不是类的成员，因此不能使用对象直接访问，而只能通过接口对象来访问。

## 13.2 抽象类的声明及使用

如果一个类不与具体的事物相联系，而只是表达一种抽象的概念，仅仅是作为其派生类的一个基类，则这样的类就是抽象类。在抽象类中声明方法时，如果加上 abstract 关键字，则为抽象方法。本节将对抽象类的声明及使用进行详细介绍。

### 13.2.1 抽象类概述

抽象类主要用来提供多个派生类可共享的基类的公共定义，它与非抽象类的主要区别如下。

- 抽象类不能直接创建。
- 抽象类中可以包含抽象成员，但非抽象类中不可以。
- 抽象类不能被密封。

### 13.2.2 抽象类的声明

在 C# 中声明抽象类时需要使用 `abstract` 关键字，具体语法格式如下。

访问修饰符 `abstract class` 类名:基类或接口

```
{
    //类成员
}
```

说明：在声明抽象类时，除了 `abstract` 关键字、`class` 关键字和类名外，其他的都是可选项。

**例 13.04** 下面代码声明一个抽象类，在该抽象类中包含一个 `int` 类型的变量和一个无返回值类型方法，实现代码如下。

```
public abstract class myClass
{
    public int i;
    public void method()
    {}
}
```

### 13.2.3 抽象方法的声明

抽象方法就是指在声明方法时加上 `abstract` 关键字，声明抽象方法时需要注意以下几点。

- 抽象方法必须声明在抽象类中。
- 当声明抽象方法时，不能使用 `virtual`、`static` 和 `private` 修饰符。
- 抽象方法声明引入了一个新方法，但不提供该方法的实现，由于抽象方法不提供任何实际实现，因此抽象方法的方法体只包含一个分号。

当从抽象类派生一个非抽象类时，需要在非抽象类中重写抽象方法，以提供具体的实现，重写抽象方法时使用 `override` 关键字。

**例 13.05** 下面代码声明一个抽象类，该抽象类中声明一个抽象方法，实现代码如下。

```
public abstract class myClass
{
    public abstract void method(); //抽象方法
}
```

### 13.2.4 抽象类的使用

本节通过一个实例介绍如何在程序中使用抽象类。

**例 13.06** 创建一个控制台应用程序，其中声明一个抽象类 `myClass`。在该抽象类中，声明了两个属性和一个方法，其中，为两个属性提供了具体实现，方法为抽象方法，然后声明一个派生类 `DriveClass`，继承自 `myClass`，在 `DriveClass` 派生类中重写 `myClass` 抽象类中的抽象方法，并提供具体的实现，最后在主程序类 `Program` 的 `Main` 方法中创建 `DriveClass` 派生类的一个对象，使用该对象创建抽象类，并使用抽象对象访问

抽象类中的属性和派生类中重写的方法。程序代码如下。（实例位置：光盘\mr\13\sl\13.06）

```

public abstract class myClass
{
    private string id = "";
    private string name = "";
    public string ID //编号属性及实现
    {
        get
        {
            return id;
        }
        set
        {
            id = value;
        }
    }
    public string Name //姓名属性及实现
    {
        get
        {
            return name;
        }
        set
        {
            name = value;
        }
    }
    public abstract void ShowInfo(); //抽象方法，用来输出信息
}
public class DriveClass:myClass //继承抽象类
{
    public override void ShowInfo() //重写抽象类中输出信息的方法
    {
        Console.WriteLine(ID + " " + Name);
    }
}
class Program
{
    static void Main(string[] args)
    {
        DriveClass driveclass = new DriveClass();
        myClass myclass = driveclass; //创建派生类
        myclass.ID = "BH0001"; //使用派生对象创建抽象类
        myclass.Name = "TM"; //使用抽象对象访问抽象类中的编号属性
        myclass.ShowInfo(); //使用抽象对象访问抽象类中的姓名属性
        Console.ReadLine(); //使用抽象对象调用派生类中的方法
    }
}

```

程序运行结果如图 13.3 所示。



图 13.3 运行结果

### 13.2.5 抽象类与接口

抽象类和接口都包含可以由派生类继承的成员，它们都不能直接创建，但可以声明它们的变量。如果这样做，就可以使用多态性把继承这两种类型的对象指定给它们的变量，接着通过这些变量来使用这些类型的成员，但不能直接访问派生类中的其他成员。

抽象类和接口的区别主要有以下几点。

- 它们的派生类只能继承一个基类，即只能直接继承一个抽象类，但可以继承任意多个接口。
- 抽象类中可以定义成员的实现，但接口中不可以。
- 抽象类中可以包含字段、构造函数、析构函数、静态成员或常量等，但接口中不可以。
- 抽象类中的成员可以是私有的（只要它们不是抽象的）、受保护的、内部的或受保护的内部成员（受保护的内部成员只能在应用程序的代码或派生类中访问），但接口中的成员必须是公共的。

 **说明：**抽象类和接口这两种类型用于完全不同的目的。抽象类主要用作对象系列的基类，共享某些主要特性，如共同的目的和结构；接口主要用于类，这些类在基础水平上有所不同，但仍可以完成某些相同的服务。

## 13.3 密封类的声明及使用

如果所有的类都可以被继承，那么很容易导致继承的滥用，进而使类的层次结构体系变得十分复杂，这样就使得开发人员对类的理解和使用变得十分困难。为了避免滥用继承，C#中提出了密封类的概念，本节将对密封类的声明及使用进行详细介绍。

### 13.3.1 密封类概述

密封类可以用来限制扩展性，如果密封了某个类，则其他类不能从该类继承；如果密封了某个成员，则派生类不能重写该成员的实现。默认情况下，不应密封类型和成员。密封可以防止对库的类型和成员进行自定义，但同时也会影响某些开发人员对可用性的认识。

在 C# 中使用密封类时，如果类满足如下条件，则应将其密封。

- 类是静态类。
- 类包含带有安全敏感信息的继承的受保护成员。
- 类继承多个虚成员，并且密封每个成员的开发和测试开销明显大于密封整个类。
- 类是一个要求使用反射进行快速搜索的属性，密封属性可提高反射在检索属性时的性能。

### 13.3.2 密封类的声明

在 C# 中声明密封类时需要使用 sealed 关键字，具体语法格式如下。

```
访问修饰符 sealed class 类名:基类或接口
{
    //类成员
}
```

- 说明:** ① 密封类不能作为基类被继承，但它可以继承其他类或接口。  
 ② 在密封类中不能声明受保护成员或虚成员，因为受保护成员只能从派生类进行访问，而虚成员只能在派生类中重写。  
 ③ 由于密封类的不可继承性，因此密封类不能声明为抽象的，即 sealed 修饰符不能与 abstract 修饰符同时使用。

**例 13.07** 下面代码声明一个密封类，在该密封类中包含一个 int 类型的变量和一个无返回值类型方法，它们只能通过创建密封类的对象来访问，而不能被继承。实现代码如下。

```
public sealed class myClass          //声明密封类
{
    public int = 0;
    public void method()
    {
        Console.WriteLine("密封类");
    }
}
```

### 13.3.3 密封方法的声明

并不是每个方法都可以声明为密封方法，密封方法只能用于对基类的虚方法进行重写，并提供具体的实现，所以，当声明密封方法时，sealed 修饰符总是和 override 修饰符同时使用。

**例 13.08** 下面代码声明一个类 MyClass1，在该类中声明一个虚方法 Method，然后声明一个密封类 MyClass2，该类继承自 MyClass1 类，在密封类 MyClass2 中密封并重写 MyClass1 类中的虚方法 Method。实现代码如下。

```
public class MyClass1
{
    public virtual void Method()
    {
        Console.WriteLine("基类中的虚方法");
    }
}
public sealed class MyClass2:myClass1
{
    public sealed override void Method()          //密封并重写基类中的虚方法 Method
    {
        base.Method();
        Console.WriteLine("密封类中重写后的方法");
    }
}
```

**技巧:** 在上面代码中，密封并重写基类中的虚方法 Method 时用到了 base.Method();语句，该语句表示调用基类中 Method 方法。base 关键字主要是为派生类调用基类成员提供一种简写的方法。

### 13.3.4 密封类的使用

密封类除了不能被继承外，与非密封类的用法大致相同。密封方法必须通过重写基类中的虚方法来实现，下面通过一个实例讲解如何在程序中使用密封类。

**例 13.09** 创建一个控制台应用程序，其中声明一个类 myClass1，在该类中声明了一个虚方法 ShowInfo，用来显示信息，然后声明一个密封类 MyClass2，继承自 myClass1 类，在 MyClass2 密封类中声明两个公共属性，分别用来表示用户编号和名称，然后密封并重写 myClass1 基类中的虚方法 ShowInfo，并提供具体的实现，最后在主程序类 Program 的 Main 方法中创建 MyClass2 密封类的一个对象，并使用该对象访问 MyClass2 密封类中的公共属性和密封方法。程序代码如下。（实例位置：光盘\mr\13\sl\13.09）

```

public class myClass1
{
    public virtual void ShowInfo()           //虚方法，用来显示信息
    {
    }

}

public sealed class MyClass2 : myClass1      //密封类，继承自 MyClass1
{
    private string id = "";                  //string 类型变量，用来记录编号
    private string name = "";                //string 类型变量，用来记录名称
    public string ID                         //编号属性
    {
        get
        {
            return id;
        }
        set
        {
            id = value;
        }
    }

    public string Name                      //名称属性
    {
        get
        {
            return name;
        }
        set
        {
            name = value;
        }
    }

    public sealed override void ShowInfo()    //密封并重写基类中的 ShowInfo 方法
    {
        Console.WriteLine(ID + " " + Name);
    }
}

class Program

```

```

{
    static void Main(string[] args)
    {
        myClass2 myclass2 = new myClass2(); //创建密封对象
        myclass2.ID = "BH0001";           //为密封类中的编号属性赋值
        myclass2.Name = "TM";            //为密封类中的名称属性赋值
        myclass2.ShowInfo();             //调用密封类中的密封方法
        Console.ReadLine();
    }
}

```

程序运行结果如图 13.4 所示。



图 13.4 运行结果

## 13.4 照猫画虎——基本功训练

### 13.4.1 基本功训练 1——自定义抽象类计算圆形的面积

视频讲解：光盘\mr\13\lx\自定义抽象类计算圆形的面积.exe

实例位置：光盘\mr\13\zmhh\01

本实例将通过自定义的抽象类来计算圆形的面积。新建一个 Windows 窗体应用程序，在默认窗体 Form1 中添加两个 TextBox 控件，分别用来输入圆半径和显示圆面积；添加一个 Button 控件，用来根据输入的圆半径计算圆面积。程序主要代码如下。

```

public abstract class Roll
{
    private int r = 0;
    public int R //圆半径
    {
        get
        {
            return r;
        }
        set
        {
            r = value;
        }
    }
    public abstract double Area(); //抽象方法，用来计算圆面积
}
public class CalcArea : Roll //继承抽象类
{
    public override double Area() //重写抽象类中计算圆面积的方法
    {

```

```

        return Math.PI * R * R;      //返回圆面积
    }
}

```

实例运行效果如图 13.5 所示。

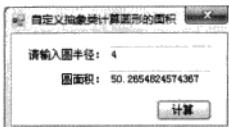


图 13.5 自定义抽象类计算圆形的面积

**说明：**计算圆面积时需要用到  $\pi$ ， $\pi$  在 C# 中使用 `Math.PI` 常量字段来表示，它的值为 3.14159265358979323846。

**熊猫画虎：**自定义抽象类计算矩形的面积。(20 分)(实例位置：光盘\mr\13\zmhh\01\_zmhh)

### 13.4.2 基本功训练 2——利用接口实现选择不同的语言

**视频讲解：**光盘\mr\13\lx\利用接口实现选择不同的语言.exe

**实例位置：**光盘\mr\13\zmhh\02

当一个中国人和一个美国人交流时，要么美国人对中国人说汉语，要么中国人对美国人说英语。因此，要在程序中模拟这种场景，首先应建立一个接口，并在该接口中定义一个用于对话的方法；然后分别创建一个中国人的类和一个美国人的类，这两个类都继承自该接口，当和不同国家的人交流时，只需声明该接口的引用，并调用相应类中的方法即可。新建一个 Windows 窗体应用程序，在默认窗体 Form1 中添加一个 ComboBox 控件，用来选择对话方式；添加一个 TextBox 控件，并将其 Multiline 属性设置为 True，用来输入对话内容；添加一个 Button 控件，用来执行对话操作。程序代码如下。

```

interface ISelectLanguage //声明一个接口，用于定义 Speak 方法，而具体 Speak 方法功能的实现是在类中进行的
{
    void Speak(string str);
}
class C_SpeakChinese : IselectLanguage //如果和中国人对话，则说汉语
{
    public void Speak(string str)           //显示对中国说的话
    {
        MessageBox.Show("您对中国友人说：" + str, "提示", MessageBoxButtons.OK, MessageBoxIcon.Information);
    }
}
class C_SpeakEnglish : IselectLanguage //如果和美国人对话，则说英语
{
    public void Speak(string str)           //显示对美国人说的话
    {
        MessageBox.Show("您对美国友人说：" + str, "提示", MessageBoxButtons.OK, MessageBoxIcon.Information);
    }
}

```

实例运行效果如图 13.6 和图 13.7 所示。



图 13.6 选择“中国人”对话方式



图 13.7 选择“美国人”对话方式

**熊猫画虎：**参照上面的实例，利用接口实现输出不同类型动物的叫声，如输出鸟儿吱吱叫、猫喵喵叫等。(20 分)(实例位置：光盘\mr\13\zmhh\02\_zmhh)

### 13.4.3 基本功训练 3——使用接口作为方法参数进行编程

**视频讲解：**光盘\mr\13\lx\使用接口作为方法参数进行编程.exe

**实例位置：**光盘\mr\13\zmhh\03

依赖倒置原则是面向对象设计原则中一条非常重要的原则，它描述在进行业务设计时，要依赖于抽象而不依赖于具体实现。若使用接口或抽象类而不是它的实现类来编程，则会使代码变得更加稳定，这样大大提高了程序应对业务变化的能力。创建一个 Windows 应用程序，在默认窗体 Form1 中添加一个 ComboBox 控件和一个 Button 控件，分别用来显示列表项和实现向 ComboBox 控件添加项的操作。在 Form1 窗体的后台代码中自定义 InitComboBox 方法，该方法实现向 ComboBox 控件添加项，其第一个参数的类型是 ICollection，该参数表示向 ComboBox 控件中添加的项的集合。主要代码如下。

```
private void InitComboBox(ICollection items, ComboBox cbx) //实现向 ComboBox 控件中添加项
{
    foreach (object item in items)
        cbx.Items.Add(item.ToString()); //遍历 items 参数中的元素
    //向 ComboBox 控件中添加项
}
private void button1_Click(object sender, EventArgs e)
{
    List<string> list = new List<string>(); //单击“添加项”按钮，向 ComboBox 控件添加项
    list.Add("C# 编程词典"); //创建一个字符串列表
    list.Add("JAVA 编程词典"); //向字符串列表中添加项
    list.Add("VB 编程词典"); //向字符串列表中添加项
    InitComboBox(list, comboBox1); //向 ComboBox 控件添加项
}
```

程序运行结果如图 13.8 所示。



图 13.8 使用接口作为方法参数进行编程

**说明：**从上面的 InitComboBox 方法可以看出，对于任何实现 ICollection 接口的类型它都能很好地工作，如数组、字符串列表 (List<string>) 等，这样的程序具有较强的应对变换的能力，使代码变得比较稳定，从而体现出使用接口作为方法参数进行编程的优越性。

**照猫画虎：**修改上面的实例，向 InitComboBox 方法的第一个参数传入数组类型的实参。(20 分)(实例位置：光盘\mr\13\zmhh\03\_zmhh)

#### 13.4.4 基本功训练 4——通过重写虚方法实现加法运算

**视频讲解：**光盘\mr\13\lx\通过重写虚方法实现加法运算.exe

**实例位置：**光盘\mr\13\zmhh\04

新建一个 Windows 窗体应用程序，在默认窗体 Form1 中添加 3 个 TextBox 控件，分别用来输入要计算的两个数和显示计算结果；添加一个 ComboBox 控件，用来选择计算方式；添加一个 Button 控件，用来按指定方式对输入的数进行计算。程序主要代码如下。

```
class Operation
{
    public virtual double operation(int x,int y) //建立的虚方法，可以在子类中被重写
    {
        return x * y; //计算两个数的乘积
    }
}
class Addition : Operation //在子类中重写虚方法
{
    public override double operation(int x, int y) //重写虚方法
    {
        return (x + y); //计算两个数的和
    }
}
```

程序运行结果如图 13.9 所示。

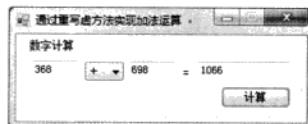


图 13.9 通过重写虚方法实现加法运算

**说明：**virtual 修饰符不能与 static、abstract 或 override 修饰符同时使用。另外，由于虚方法不能是私有的，所以 virtual 修饰符不能与 private 修饰符同时使用。

**照猫画虎：**修改上面的实例，通过重写虚方法实现计算指定数字的平方根。提示：计算某个数字的平方根，可使用 Math 类的 Sqrt 方法。(20 分)(实例位置：光盘\mr\13\zmhh\04\_zmhh)

#### 13.4.5 基本功训练 5——使用多重继承实现教师和学生信息的输出

**视频讲解：**光盘\mr\13\lx\使用多重继承实现教师和学生信息的输出.exe

**实例位置：**光盘\mr\13\zmhh\05

创建一个 Windows 应用程序，在默认窗体 Form1 的后台代码中声明了 3 个接口 IPeople、ITeacher 和 IStudent，其中，ITeacher 和 IStudent 继承自 IPeople；然后使用 People 类继承这 3 个接口，并分别实现这 3 个接口中的属性和方法；最后在 Form1 窗体的 Load 事件中创建 People 类的对象，并调用该对象的相关属性

和方法。程序代码如下。

```

interface Ipeople //自定义描述人的接口
{
    string Name //描述姓名
    {
        get;
        set;
    }
    string Sex //描述性别
    {
        get;
        set;
    }
}
interface ITteacher : Ipeople //继承公共接口
{
    string teach(); //具有教学方法
}
interface Istudent : Ipeople //继承公共接口
{
    string study(); //具有学习的行为
}
class People : Ipeople, ITteacher, Istudent //实现这3个接口
{
    string strName = ""; //定义名称字段
    string strSex = ""; //定义性别字段
    public string Name //实现描述姓名的属性
    {
        get
        {
            return strName;
        }
        set
        {
            strName = value;
        }
    }
    public string Sex //实现描述性别的属性
    {
        get
        {
            return strSex;
        }
        set
        {
            strSex = value;
        }
    }
    public string teach() //实现教学这个方法
}

```

```

    {
        return Name + ":" + Sex + ", 是一名教师";
    }
    public string study() //实现学习这个方法
    {
        return Name + ":" + Sex + ", 是一名学生";
    }
}
private void Form1_Load(object sender, EventArgs e)
{
    People people = new People(); //创建对象
    ITeacher iteacher = people; //使用派生对象创建接口 ITeacher
    iteacher.Name = "东*方"; //设置姓名
    iteacher.Sex = "男"; //设置性别
    richTextBox1.Text = iteacher.teach() + Environment.NewLine; //输出老师信息
    IStudent istudent = people; //使用派生对象创建接口 IStudent
    istudent.Name = "小*科"; //设置姓名
    istudent.Sex = "女"; //设置性别
    richTextBox1.Text += istudent.study(); //输出学生信息
}

```

程序运行结果如图 13.10 所示。

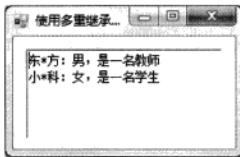


图 13.10 使用多重继承实现教师和学生信息的输出

**照猫画虎：**修改上面的程序，增加输出教师和学生的年龄信息。提示：在 Ipeople 接口增加一个表示年龄的属性。(20 分)(实例位置：光盘\mr\13\zmhh\05\_zmhh)

**照猫画虎栏目分数统计：**

照猫画虎题目	1	2	3	4	5	总分数	
分数							

## 13.5 情景应用——拓展与实践

### 13.5.1 情景应用 1——使用迭代器显示公交车站点

视频讲解：光盘\mr\13\lx\使用迭代器显示公交车站点.exe

实例位置：光盘\mr\13\qjyy\01

在人们去某地时，有时只知道地方的大概位置，并不知道具体怎么走，这就可以通过查找公交车的站点来明确坐几路公交车，本实例将使用 C# 中的迭代器依次显示公交车的所有站点，实例运行效果如图 13.11 所示。

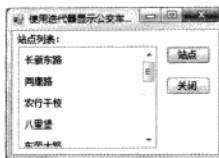


图 13.11 使用迭代器显示公交车站点

实现过程如下。

- (1) 打开 Visual Studio 2008 开发环境，新建一个 Windows 窗体应用程序，并将其命名为 ShowBusStation。
- (2) 在默认窗体 Form1 中添加一个 ListView 控件，用来显示公交车站点列表；添加两个 Button 控件，分别用来获取公交车站点和关闭当前窗体。

(3) 程序主要代码如下：

```
public static IList<object> items = new List<object>();           // 定义一个泛型对象，用于存储对象
public static IEnumerable<object> GetValues()                      // 通过迭代器获取泛型中的所有对象值
{
    if (items != null)                                            // 如果泛型不为空
    {
        foreach (object i in items)                                // 遍历泛型中的对象
            yield return i;
    }
}
private void button1_Click(object sender, EventArgs e)
{
    foreach (object i in GetValues())                            // 遍历泛型集合
        listView1.Items.Add(i.ToString());                         // 在列表视图中显示公交车站点
}
```

**DIY：使用迭代器显示中国四大银行。**提示：中国四大银行包括中国工商银行、中国建设银行、中国银行、中国农业银行。(20 分)(实例位置：光盘\mr\13\qjyy\01\_diy)

### 13.5.2 情景应用 2——通过迭代器实现文字的动态效果

■ 视频讲解：光盘\mr\13\lx\通过迭代器实现文字的动态效果.exe

■ 实例位置：光盘\mr\13\qjyy\02

本实例主要使用迭代器遍历文本字符串中的每一个文字，然后使用 GDI+技术在窗体上以不同的字体样式依次绘制每一个文字，以便实现文字的动态效果。实例运行效果如图 13.12 所示。

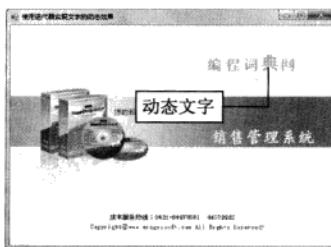


图 13.12 通过迭代器实现文字的动态效果

实现过程如下。

(1) 打开 Visual Studio 2008 开发环境，新建一个 Windows 窗体应用程序，并将其命名为 DynamicLetter。

(2) 在默认窗体 Form1 中添加一个 Panel 控件，用来显示动态文字。

(3) 程序主要代码如下。

```

/// <param string="n">文字字符串</param>
/// <returns>返回单个文字</returns>
public static IEnumerable<object> Transpose(string n)                                //通过迭代器实现字符串的遍历
{
    if (n.Length > 0)                                                               //如果泛型不为空
    {
        foreach (object i in n)                                                     //对字符串进行遍历
            yield return i;
    }
}
/// <param string="C_Str">绘制的文字字符串</param>
public void DynamicText(Object C_Str)                                              //绘制动态文字
{
    float tem_left = 0;                                                            //获取当前文字的左端位置
    float tem_top = 0;                                                             //获取当前文字的顶端位置
    float tem_w = 0;                                                               //获取文字的宽度
    float tem_h = 0;                                                               //获取文字的高度
    float tem_place = Str_Width;                                                 //获取起始文字的位置
    Font Tem_Font = new Font("黑体", FSize[0], FontStyle.Bold);                  //定义字体样式
    int p = 0;                                                                    //记录字符串中文字的索引号
    int Str_Index = 0;
    try
    {
        foreach (object s in Transpose(C_Str.ToString()))                         //遍历字符串
        {
            for (int i = 1; i < 5; i++)                                          
            {
                if (i >= 3)
                    p = Convert.ToInt16(Math.Floor(i / 2F));
                else
                    p = i;
                ProtractText(C_Str.ToString(), Str_Index);                         //定义字体样式
                Tem_Font = new Font("黑体", FSize[p], FontStyle.Bold);             //将绘制的单个文字进行格式化
                SizeF TitSize = g.MeasureString(s.ToString(), Str_Font);          //获取文字的宽度
                tem_w = TitSize.Width;                                               //获取文字的高度
                tem_h = TitSize.Height;                                              //获取文字改变大小后的左端位置
                tem_left = tem_place - (tem_w - Str_Odd_Width) / 2F;                //获取文字改变大小后的顶端位置
                tem_top = Str_Height - (Str_Height - tem_h) / 2F;
                ProtractOddText(s.ToString(), Tem_Font, tem_left, tem_top);         //绘制单个文字
                Thread.Sleep(200);                                                   //等待 0.2 秒
                g.FillRectangle(new SolidBrush(Panel_C), 0, 0, Panel_W, Panel_H);   //清空绘制的文字
            }
            tem_place += Str_Odd_Width + Str_block;                             //计算下一个文字的左端位置
            Str_Index += 1;                                                       //将索引号定位到下一个文字
        }
    }
}

```

```

        ProtractText(C_Str.ToString(), -1); //恢复文字的原始绘制样式
        th = new Thread(new ParameterizedThreadStart(DynamicText)); //创建 ParameterizedThreadStart 委托线程
        th.Start(C_Str); //传递一个字符串的参数
    }
    catch
    {
        th.Abort(); //在关闭窗体时关闭线程
    }
}
}

```

**技巧：**如果一个任务执行时间过长，会导致程序主窗体处于“假死”状态，为了避免这种情况的发生，可以通过使用 Thread 类来创建多线程，即每一个线程完成一个功能，这样就可以有效地避免程序出现“假死”现象。

**DIY：**使用迭代器实现倒序输出字符串。提示：在迭代器的 for 语句中，设置迭代表达式为 i--，以便从末尾开始遍历字符串中的字符。（20分）（实例位置：光盘\mr\13\qjyy\02\_diy）

### 13.5.3 情景应用 3——使用分部类实现多种计算方法

**视频讲解：**光盘\mr\13\lx\使用分部类实现多种计算方法.exe

**实例位置：**光盘\mr\13\qjyy\03

本实例使用分部类来制作一个计算器，例如，将实现加、减、乘和除的方法放在一个分部类中，而将实现开方、百分比和倒数的方法放在另一个分部类中。实例运行效果如图 13.13 所示。

实现过程如下。

(1) 打开 Visual Studio 2008 开发环境，新建一个 Windows 窗体应用程序，并将其命名为 BranchMeans。

(2) 在默认窗体 Form1 中添加一个 TextBox 控件，用来显示输入的数和计算结果；添加 23 个 PictureBox 控件，分别用来表示相应的数字和对输入的数进行加、减、乘、除、求倒数、开方、相反数和百分比运算。

(3) 程序主要代码如下。

```

public partial class Account
{
    public double Addition(double Former, double After) //执行两个数的加法运算
    {
        return Former + After; //返回相加后的结果
    }
    public double Subtraction(double Former, double After) //执行两个数的减法运算
    {
        return Former - After; //返回相减后的结果
    }
    public double Multiplication(double Former, double After) //执行两个数的乘法运算
    {
        return Former * After; //返回相乘后的结果
    }
    public double Division(double Former, double After) //执行两个数的除法运算
    {
    }
}

```



图 13.13 使用分部类实现多种计算方法

```

if (After == 0)                                //判断被除数是否为 0
{
    MessageBox.Show("被除数不能为 0。");
    return 0;
}
return Former / After;                         //返回相除后的结果
}

public partial class Account
{
    public double Molecule(double num)           //计算一个数的倒数
    {
        return 1 / num;                          //返回倒数值
    }
    public double Evolution(double num)          //计算一个数的开方
    {
        return Math.Sqrt(num);                  //返回开方后的值
    }
    public double Opposition(double num)         //求一个数的相反数
    {
        return -num;                           //返回相反数
    }
    public double Par(double num)                //计算一个数的百分比
    {
        return num / 100 * num;                 //返回百分比
    }
}
}

```

**技巧：**C# 编码规范中规定，一个类中的代码最好不要超过 500 行，但在实际开发中，可能需要在一个类中编写超过 500 行以上的代码，该怎么办呢？这时可考虑使用分部类将该类分成几部分。

**DIY：** 使用分部类显示员工信息。提示：把员工的基本信息放入一个分部类中，把员工的岗位信息放入另一个分部类中。（20 分）（实例位置：光盘\mr\13\qjyy\03\_diy）

#### 13.5.4 情景应用 4——通过继承泛型类实现输出学生信息

**视频讲解：**光盘\mr\13\lx\通过继承泛型类实现输出学生信息.exe

**实例位置：**光盘\mr\13\qjyy\04

在学生成绩管理系统中显示学生信息时，有时需要将学生信息分块进行存储，这样在使用时，将会在不同情况下显示不同范围的学生信息。本实例通过继承泛型类来分块显示学生的个人信息及成绩信息。实例运行效果如图 13.14 所示。

实现过程如下。

(1) 打开 Visual Studio 2008 开发环境，新建一个 Windows 窗体应用程序，并将其命名为 ShowStuInfo。

(2) 在默认窗体 Form1 中添加 9 个 TextBox 控件，



图 13.14 通过继承泛型类实现输出学生信息

分别用来显示学生的编号、姓名、性别、年龄、生日、班级、语文成绩、数学成绩和英语成绩等信息；添加一个 Button 控件，用来通过访问继承的泛型派生类获取学生信息并显示。

(3) 程序主要代码如下。

```
class BStuInfo<T>
{
    public T ID;                                //声明学生编号字段
    public T Name;                             //声明姓名字段
    public T Sex;                               //声明性别字段
    public T Age;                               //声明年龄字段
    public T Birthday;                         //声明生日字段
    public T Grade;                            //声明班级字段
}
class HStuInfo<T> : BStuInfo<T>           //继承自 BStuInfo 泛型类
{
    public T Chinese;                          //声明语文成绩字段
    public T Math;                            //声明数学成绩字段
    public T English;                         //声明英语成绩字段
}
```

**DIY：**通过实现泛型接口输出轿车信息。提示：定义一个泛型接口，并在其中声明若干用于描述车辆信息的属性和方法，然后定义一个轿车类，并实现该泛型接口。(20 分)(实例位置：光盘\mr\13\qjyy\04\_diy)

### 13.5.5 情景应用 5——使用密封类密封用户信息

■**视频讲解：**光盘\mr\13\lx\使用密封类密封用户信息.exe

■**实例位置：**光盘\mr\13\qjyy\05

密封类的突出特点是不能被继承，通过密封类封装用户信息可以增加用户信息的安全性。本实例将使用密封类密封用户的登录用户名和密码，以保证其安全性。实例运行效果如图 13.15 所示。



图 13.15 使用密封类密封用户信息

实现过程如下。

- (1) 打开 Visual Studio 2008 开发环境，新建一个 Windows 窗体应用程序，并将其命名为 SealedUserInfo。
- (2) 在默认窗体 Form1 中添加两个 TextBox 控件，分别用来输入用户名和密码；添加两个 Button 控件，分别用来执行用户登录和退出窗体功能。

(3) 程序主要代码如下。

```
public sealed class myClass          //通过 sealed 关键字声明密封类，防止类被继承，有效保护重要信息
{
    private string name = "";        //string 类型变量，用来记录用户名
    private string pwd = "";         //string 类型变量，用来记录密码
    public string Name               //用户名
```

```

    {
        get
        {
            return name;
        }
        set
        {
            name = value;
        }
    }
    public string Pwd          //密码
    {
        get
        {
            return pwd;
        }
        set
        {
            pwd = value;
        }
    }
}

```

**DIY：** 使用密封类封装个人身份证件信息。提示：密封的个人信息至少应包括姓名、出生日期、身份证号。(20分)(实例位置：光盘\mr\13\qjyy\05\_diy)

情景应用 DIY 栏目分数统计：

DIY 题目	1	2	3	4	5	总分数	
分数							

## 13.6 自我测试

### 一、选择题(每题 10 分, 5 道题)

1. 下面关于抽象类的描述中, 正确的是( )。
  - A. 因为抽象类不能创建, 所以抽象类不能包含构造函数
  - B. 基类是抽象类, 该基类的派生类可以是抽象类, 也可以不是抽象类
  - C. 抽象类中, 只能包含抽象方法, 不能包含实例方法
  - D. 抽象类中的抽象方法可以具有公有、私有和保护访问权限
2. 关于继承和接口, 以下说法正确的是( )。
 

A. C#既允许多接口实现, 也允许多重继承	B. C#允许多接口实现, 但不允许多重继承
C. C#不允许多接口实现, 但允许多重继承	D. C#既不允许多接口实现, 也不允许多重继承
3. 有关 Sealed 修饰符, 描述正确的是( )。
 

A. 密封类可以被继承	B. abstract 修饰符可以和 sealed 修饰符一起使用
C. 密封类不能实例化	D. 使用 sealed 修饰符可以保证此类不能被派生

4. 下列方法中，哪个是抽象方法（ ）。  
 A. static void Func()  
 C. abstract void Func()
- B. virtual void Func()  
 D. override void Func()
5. 下面关于抽象类和接口论述不正确的是（ ）。  
 A. 它们的派生类只能继承一个基类，即只能直接继承一个抽象类，但可以继承任意多个接口  
 B. 抽象类中可以定义成员的实现，但接口中不可以  
 C. 抽象类中可以包含字段、构造函数、析构函数、静态成员或常量等，接口中不可以  
 D. 抽象类不可以继承自接口

**二、填空题（每题 10 分，5 道题）**

1. 若要实现接口成员，类中的对应成员必须是（ ）、（ ），并且与接口成员具有相同的（ ）和（ ）。
2. 抽象类和接口都包含可以由派生类继承的成员，它们都不能直接（ ），但可以声明它们的（ ）。
3. 密封类可以用来限制扩展性，如果密封了某个类，则其他类不能从该类（ ）；如果密封了某个成员，则派生类不能重写该成员的（ ）。
4. 可以使用 new、public、protected、internal 和 private 等修饰符声明接口，但接口成员必须是（ ）。
5. 在密封类中不能声明（ ）或（ ），因为受保护成员只能从派生类进行访问，而虚成员只能在派生类中重写。

**测试分数统计：**

类别	第 1 题	第 2 题	第 3 题	第 4 题	第 5 题
选择题分数					
填空题分数					
总分数					

**13.7 行 动 指 南**

开始日期： 年 月 日 结束日期： 年 月 日

序号	内 容	行 动 指 南	
1	照猫画虎栏目 分数（ ）	分数>75 分	优秀，基本功掌握得不错，加油！
		75 分>分数>50 分	及格，知识掌握得不牢，重新做一遍照猫画虎。
		分数<50 分	请使用光盘中提供的“实战能力测试系统”进行提高训练。
	情景应用栏目 分数（ ）	分数>75 分	优秀，综合应用能力很强。
		75 分>分数>50 分	及格，综合应用能力需提高，再练一遍情景应用。
	自我测试栏目 分数（ ）	分数<50 分	请使用光盘中提供的“实战能力测试系统”进行提高训练。
		分数>75 分	优秀，有成为编程高手的潜质。
	综合评价	反复训练后，以上各项分数都在 75 分以上，方可进入下一堂课学习。	

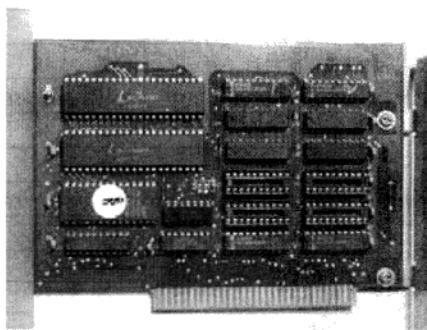
续表

序号	内 容	行动指南
2	编程能力培养：本栏目提供了一些实践题目，对于培养编程能力很有效，要做好。	1. 通过实现接口计算矩形面积。提示：首先定义一个能够描述计算矩形面积的接口，并在该接口中定义两个未实现的属性，分别用来表示矩形长度和宽度，然后定义一个未实现的方法，用来表示计算矩形的面积。 2. 通过泛型实现子窗体的不同操作。提示：实际开发项目时，有时会因为调用窗体或提示窗体过多而难于管理，这时可以通过泛型方法的重载将调用窗体与提示窗体分开编写。这样，当在程序中使用调用窗体或提示窗体时，只需调用指定的泛型方法即可。
3	编程习惯培养：本堂课培养读者在学习开发中记笔记的习惯，把开发中遇到的问题、总结的经验记录下来。	
4	创新能力培养：看看身边有没有可以用编程解决的问题，记录到右边的表格中。根据学习进度，尝试编写解决这些问题的小程序。	

## 13.8 成功可以复制——征途巨人史玉柱

1980 年，史玉柱以优异的成绩考入浙江大学数学系。大学期间，史玉柱坚持每天长跑 18 公里，以此磨练意志和精神，这也为他日后做事百折不挠打下了坚实的基础。1984 年毕业后，史玉柱被分配到了安徽省统计局农村抽样调查队工作。利用业余时间，史玉柱编写了一个程序，使得过去二三十个人忙活一年的工作，一两天就干完了，搞得很多人从此没事干。干得起兴，史玉柱又编写了一个分析软件，能分析出不同层次收入的农民可能会买什么东西、消费特征是什么，史玉柱的软件不断完善，以致得到了国家统计局的认可，要求全国各地的农村抽样调查都使用史玉柱的软件。史玉柱因此得到了一二十元奖金和一个进步奖。相比当时每月 54 元的工资，史玉柱挺知足。

1988 年，史玉柱提交了辞职报告，在家潜心编写汉字文字处理软件。1989 年 7 月，史玉柱怀揣独立开发的汉卡软件和“M-6401 桌面排版印刷系统”软盘南下深圳。当时，除了 4000 元钱，史玉柱一无所有。为了买到当时深圳最便宜的计算机（8500 元），他以加价 1000 元为条件，向计算机销售商获得推迟付款半个月的“优惠”，赊账得到了平生第一台计算机。为了推广产品，他用同样赊账的办法在《计算机世界》连续做了 3 期 1/4 版的广告。《计算机世界》给史玉柱的付款期限只有 15 天，可一直到广告见报后的第 12 天，史玉柱分文未进。就在关键时刻，第 13 天出现了转机：他一下子收到 3 张邮局汇



巨人汉卡图片

款单，总金额为 1.582 万元！先人一步的思维方式，让史玉柱迎来最初的成功。两个月后，他账上的金额竟达到了 10 万元。他再把钱投入广告中，扩大影响，4 个月后，仅靠卖 M-6401 产品就回款 100 万元，半年之后回款 400 万元。

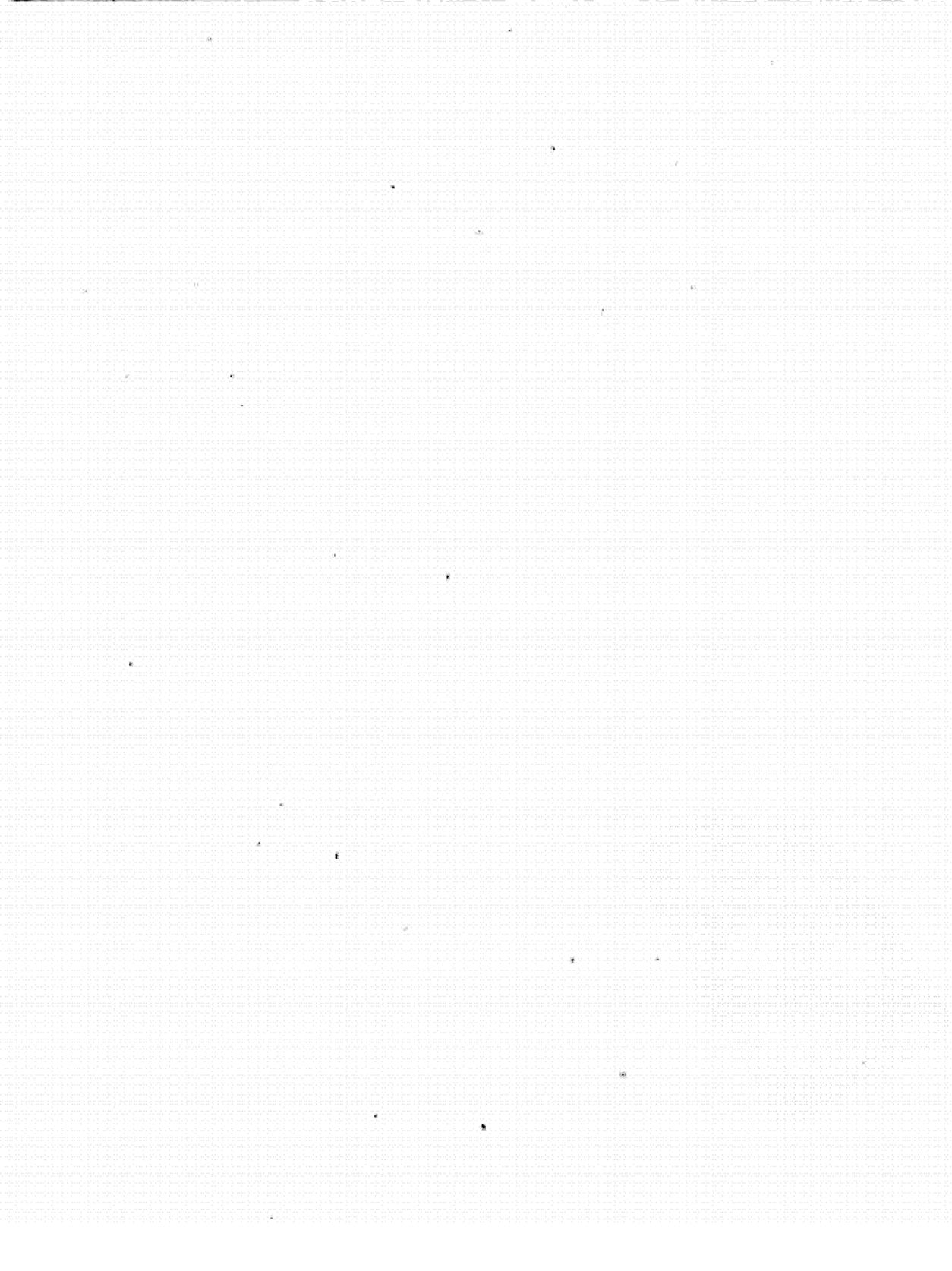
1991 年 4 月，史玉柱带着汉卡软件和 100 多名员工来到珠海，注册成立珠海巨人新技术公司。但是刚刚把企业做大的史玉柱感受到了市场的巨大压力，其 M-6402 系列产品受到了来自香港金山电脑的强烈冲击。为了迅速打开市场，史玉柱又做了一次大胆的豪赌——向全国各地的计算机销售商发出邀请，只要订购 10 块巨人汉卡，史玉柱为他们报销路费，让他们前来珠海参加巨人汉卡的全国订货会，这种大胆的营销方式快速建立了巨人汉卡的营销网络。1991 年，巨人汉卡的销量一跃成为全国同类产品之首，公司获纯利 1000 多万元。1992 年，巨人集团的资本超过 1 亿元，迎来第一个事业高峰，史玉柱也收获了人生的第一桶金。

### ✓ 经典语录

不要认为自己初中水平怎么样，初中水平跟博士后没啥区别。只要能干就行，我一直是这个观点，不在乎学历，只要能干能做出贡献就行。

### ✓ 深度评价

从巨人汉卡到巨人大厦，从脑白金到黄金搭档，史玉柱是具有传奇色彩的创业者之一。他曾经在 5 年时间内跻身财富榜第 8 位；也曾一夜之间负债 2.5 亿；而如今他又是一个著名的东山再起者，保健巨鳄、网游新锐，身家数百亿的企业家。史玉柱的成功凸显出“执著与毅力”的魅力与价值。事业的跌宕起伏、世间的是非议论，唯有敢与苦难作伴的人，才能从跌倒的阴影中爬起来，迈向成功。



# 第 14 堂课

## LINQ 技术的使用

( 视频讲解：124分钟)

LINQ (Language-Integrated Query，语言集成查询) 是微软公司提供的一项新技术，它能够将查询功能直接引入到.NET Framework 3.5 所支持的编程语言中。查询操作可以通过编程语言自身来传达，而不是以字符串形式嵌入到应用程序代码中。LINQ 主要包括 LINQ To SQL、LINQ To DataSet、LINQ To Objects 和 LINQ To XML 4 种关键技术，本堂课将对 LINQ 技术进行详细讲解。

学习摘要：

- ▶ 了解 LINQ
- ▶ 熟悉 var 关键字的使用
- ▶ 掌握 Lambda 表达式的使用
- ▶ 掌握 LINQ 查询表达式的使用
- ▶ 掌握使用 LINQ 操作 SQL Server 数据库
- ▶ 掌握使用 LINQ 操作数组及集合
- ▶ 掌握使用 LINQ 操作 DataSet 数据集
- ▶ 掌握使用 LINQ 操作 XML

## 14.1 LINQ 基础

### 14.1.1 LINQ 概述

语言集成查询 (LINQ) 是 Visual Studio 2008 中的一组新增功能，可为 C# 和 Visual Basic 语言语法提供强大的查询功能。LINQ 引入了标准的、易于学习的查询和更新数据模式，可以对其技术进行扩展以支持几乎任何类型的数据存储。Visual Studio 2008 包含 LINQ 提供程序的程序集，这些程序集支持将 LINQ 与 .NET Framework 集合、SQL Server 数据库、ADO.NET 数据集和 XML 文档一起使用，它在对象领域和数据领域之间架起了一座桥梁。

### 14.1.2 使用 var 创建隐型局部变量

在 C# 1.0、1.1 及 2.0 版本中，如果要声明一个变量，必须指定变量的类型，但在 C# 3.5 中声明变量时，可以不明确指定它的数据类型，而使用关键字 var 来声明，该关键字用来创建隐型局部变量，它指示编译器根据初始化语句右侧的表达式推断变量的类型。推断类型可以是内置类型、匿名类型、用户定义类型、.NET Framework 类库中定义的类型或任何表达式。

**例 14.01** 使用关键字 var 声明一个隐型局部变量，并赋值为 2009，代码如下。

```
var number = 2010; //声明隐型局部变量
```

在很多情况下，var 是可选的，它只是提供了语法上的便利。但在使用匿名类型初始化变量时需要使用它，这在 LINQ 查询表达式中很常见。由于只有编译器知道匿名类型的名称，因此必须在源代码中使用 var。如果已经使用 var 初始化了查询变量，则还必须使用 var 作为对查询变量进行循环访问的 foreach 语句中迭代变量的类型。

**例 14.02** 创建一个控制台应用程序，首先定义一个字符串数组，然后通过定义隐型查询表达式将字符串数组中的单词分别转换为大写和小写，最后循环访问隐型查询表达式，并输出相应的大小写单词。代码如下。（实例位置：光盘\mr\14\sl\14.02）

```
static void Main(string[] args)
{
    string[] strWords = { "MingRi", "XiaoKe", "MRBccd" }; //定义字符串数组
    //定义隐型查询表达式
    var ChangeWord =
        from word in strWords
        select new { Upper = word.ToUpper(), Lower = word.ToLower() };
    foreach (var vWord in ChangeWord) //循环访问隐型查询表达式
    {
        Console.WriteLine("大写: {0}, 小写: {1}", vWord.Upper, vWord.Lower); //输出转换后的单词
    }
    Console.ReadLine();
}
```

程序运行结果如图 14.1 所示。

使用隐式类型的变量时，需要遵循以下规则。

- 只有在同一语句中声明和初始化局部变量时才能使用 var，注意不能将该变量初始化为 null。
- 不能将 var 用于类范围的域。
- 由 var 声明的变量不能用在初始化表达式中。例如 var v = v++；，这样会在编译时产生错误。

- 不能在同一语句中初始化多个隐式类型的变量。
- 如果一个名为 var 的类型位于范围中，则当尝试用 var 关键字初始化局部变量时将产生编译错误。

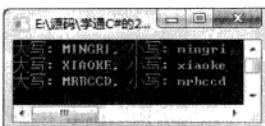


图 14.1 关键字 var 的使用

### 14.1.3 Lambda 表达式的使用

Lambda 表达式是一个匿名函数，它可以包含表达式和语句，并且可用于创建委托或表达式目录树类型。所有 Lambda 表达式都使用 Lambda 运算符=>，该运算符读为“goes to”。Lambda 运算符的左边是输入参数（如果有），右边包含表达式或语句块，例如，Lambda 表达式  $x \Rightarrow x * x$  读作“ $x$  goes to  $x$  times  $x$ ”。Lambda 表达式的基本形式如下。

语法：(input parameters) => expression

说明：input parameters 表示输入参数；expression 表示表达式。

**技巧：**① Lambda 表达式用在基于方法的 LINQ 查询中，作为诸如 Where 和 Where(IQueryable, String, Object[]) 等标准查询运算符方法的参数。

② 使用基于方法的语法在 Enumerable 类中调用 Where 方法时（像在 LINQ to Objects 和 LINQ to XML 中那样），参数是委托类型 Func<T, TResult>，使用 Lambda 表达式创建委托最为方便。

**注意：**在 is 或 as 运算符的左侧不允许使用 Lambda 表达式。

**例 14.03** 创建一个控制台应用程序，首先定义一个字符串数组，然后通过使用 Lambda 表达式查找数组中包含“C#”的字符串。代码如下。（实例位置：光盘\mr\14\sl\14.03）

```
static void Main(string[] args)
{
    //声明一个数组并初始化
    string[] strLists = new string[] { "明日科技", "C#编程词典", "学通 C# 的 24 堂课" };
    //使用 Lambda 表达式查找数组中包含“C#”的字符串
    string[] strList = Array.FindAll(strLists, s => (s.IndexOf("C#") >= 0));
    //使用 foreach 语句遍历输出
    foreach (string str in strList)
    {
        Console.WriteLine(str);
    }
    Console.ReadLine();
}
```

程序运行结果如图 14.2 所示。

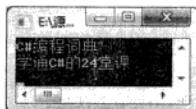


图 14.2 Lambda 表达式的使用

下列规则适用于 Lambda 表达式中的变量范围。

- 捕获的变量将不会被作为垃圾回收，直至引用变量的委托超出范围为止。
- 在外部方法中看不到 Lambda 表达式内引入的变量。
- Lambda 表达式无法从封闭方法中直接捕获 ref 或 out 参数。
- Lambda 表达式中的返回语句不会导致封闭方法返回。
- Lambda 表达式不能包含其目标位于所包含匿名函数主体外部或内部的 goto 语句、break 语句或 continue 语句。

#### 14.1.4 LINQ 查询表达式

语言集成查询 (LINQ) 是一组技术的名称，这些技术建立在将查询功能直接集成到 C# 语言（或 Visual Basic 和可能的任何其他.NET 语言）的基础上。借助于 LINQ，目前查询已是高级语言构造，就如同类、方法和事件等一样。

对于编写查询的开发人员来说，LINQ 最明显的“语言集成”部分是查询表达式。查询表达式是使用 C# 3.5 中引入的声明性查询语法编写的。通过使用查询语法，开发人员可以使用最少的代码对数据源执行复杂的筛选、排序和分组操作。可以使用相同的基本查询表达式模式来查询和转换 SQL 数据库、ADO.NET 数据集、XML 文档和流以及.NET 集合中的数据等。

使用 LINQ 查询表达式时，需要注意以下几点。

- 查询表达式可用于查询和转换来自任意支持 LINQ 的数据源中的数据。例如，单个查询可以从 SQL 数据库检索数据，并生成 XML 流作为输出。
- 查询表达式容易掌握，因为它们使用许多常见的 C# 语言构造。
- 查询表达式中的变量都是强类型的，但许多情况下不需要显式提供类型，因为编译器可以推断类型。
- 在循环访问 foreach 语句中的查询变量之前，不会执行查询。
- 在编译时，根据 C# 规范中设置的规则将查询表达式转换为“标准查询运算符”方法调用。任何可以使用查询语法表示的查询也可以使用方法语法表示。但是在大多数情况下，查询语法更易读和简洁。
- 作为编写 LINQ 查询的一项规则，建议尽量使用查询语法，只在必需的情况下才使用方法语法。
- 一些查询操作，如 Count 或 Max 等，由于没有等效的查询表达式子句，因此必须表示为方法调用。
- 查询表达式可以编译为表达式目录树或委托，具体取决于查询所应用到的类型。其中，`IEnumerable<T>` 查询编译为委托，`IQueryable` 和 `IQueryable<T>` 查询编译为表达式目录树。

LINQ 查询表达式包含 8 个基本子句，分别为 from、select、group、where、orderby、join、let 和 into，它们的说明如表 14.1 所示。

表 14.1 LINQ 查询表达式子句及说明

子句	说明
from	指定数据源和范围变量
select	指定当执行查询时返回的序列中的元素将具有的类型和形式
group	按照指定的键值对查询结果进行分组
where	根据一个或多个由逻辑“与”和逻辑“或”运算符 ( <code>&amp;&amp;</code> 或 <code>  </code> ) 分隔的布尔表达式筛选源元素
orderby	基于元素类型的默认比较器按升序或降序对查询结果进行排序
join	基于两个指定匹配条件之间的相等比较来连接两个数据源
let	引入一个用于存储查询表达式中的子表达式结果的范围变量
into	提供一个标识符，它可以充当对 join、group 或 select 子句的结果的引用

**例 14.04** 创建一个控制台应用程序，首先定义一个字符串数组，然后使用 LINQ 查询表达式查找数组中长度大于 7 的所有项并输出。代码如下。（实例位置：光盘\mr\14\sl\14.04）

```
static void Main(string[] args)
{
    //定义一个字符串数组
    string[] strName = new string[] { "明日科技", "C#编程词典", "学通 C# 的 24 堂课", "C# 开发实战 1200 例" };
    //定义 LINQ 查询表达式，从数组中查找长度小于 7 的所有项
    I Enumerable<string> selectQuery =
        from Name in strName
        where Name.Length <
        select Name;
    //执行 LINQ 查询，并输出结果
    foreach (string str in selectQuery)
    {
        Console.WriteLine(str);
    }
    Console.ReadLine();
}
```

程序运行结果如图 14.3 所示。



图 14.3 LINQ 查询表达式的使用

## 14.2 LINQ 操作 SQL Server 数据库

### 14.2.1 使用 LINQ 查询 SQL Server 数据库

使用 LINQ 查询 SQL 数据库时，首先需要创建 LinqToSql 类文件，具体步骤如下。

(1) 选中当前项目，单击鼠标右键，在弹出的快捷菜单中选择“添加”/“新建项”命令，如图 14.4 所示。

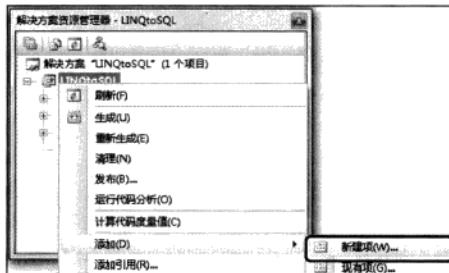


图 14.4 选择“添加”/“新建项”命令

(2) 弹出“添加新项”对话框，如图 14.5 所示。在该对话框中选择“LINQ to SQL 类”，并输入名称，

单击“添加”按钮，添加一个 LinqToSql 类文件。

(3) 在弹出的“服务器资源管理器”对话框的“数据连接”上单击鼠标右键，在弹出的快捷菜单中选择“添加连接”命令，如图 14.6 所示。

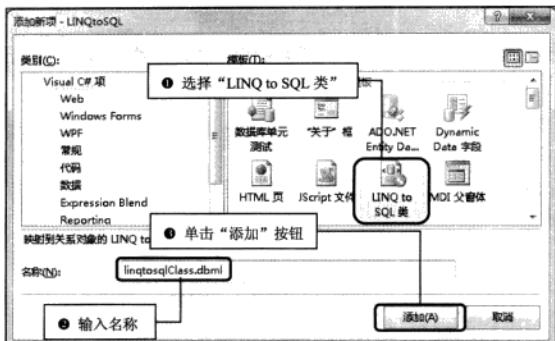


图 14.5 “添加新项”对话框

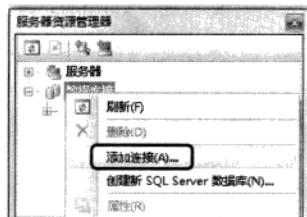


图 14.6 选择“添加连接”命令

(4) 弹出如图 14.7 所示的“添加连接”对话框，在该对话框中选择服务器名，并选择要连接的数据。

(5) 在图 14.7 所示的对话框中单击“确定”按钮，返回到“服务器资源管理器”对话框中，如图 14.8 所示，展开新建的连接对象。

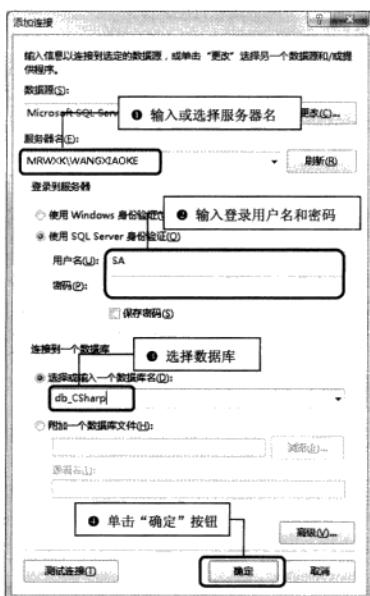


图 14.7 “添加连接”对话框

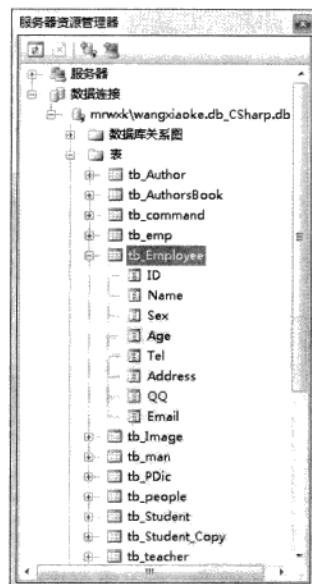


图 14.8 “服务器资源管理器”对话框

(6) 在“服务器资源管理器”对话框中选中要连接的表，并将其拖放到 linqtosqlClass.dbml 设计界面中，如图 14.9 所示。

(7) 单击工具栏中的 $\text{保存}$ 按钮，可以看到“解决方案资源管理器-解决方案”对话框中存在有 linqtosqlClass.dbml 文件，如图 14.10 所示。双击 linqtosqlClass.designer.cs，即可查看其详细代码。



图 14.9 linqtosqlClass.dbml 设计界面



图 14.10 “解决方案资源管理器”对话框

创建完 LinqToSql 类文件之后，接下来就可以使用它了，下面通过一个例子讲解如何使用 LINQ 查询 SQL Server 数据库。

**例 14.05** 创建一个 Windows 应用程序，在 Form1 窗体中添加一个 ComboBox 控件，用来选择查询条件；添加一个 TextBox 控件，用来输入查询关键字；添加一个 Button 控件，用来执行查询操作；添加一个 DataGridView 控件，用来显示数据库中的数据。（实例位置：光盘\mr\14\sl\14.05）

首先在当前项目中依照上面所讲的步骤创建一个 LinqToSql 类文件，然后在 Form1 窗体中定义一个 string 类型的变量，用来记录数据库连接字符串，并声明 Linq 连接对象，代码如下。

```
string strCon = "Data Source=MRWXK\WANGXIAOKE;Database=db_CSharp;Uid=sa;Pwd='';" //定义数据库连接字符串
//声明 Linq 连接对象
linqtosqlClassDataContext linq;
```

Form1 窗体加载时，首先将数据库中的所有员工信息显示到 DataGridView 控件中，实现代码如下。

```
private void Form1_Load(object sender, EventArgs e)
{
    BindInfo();
}
```

上面的代码中用到了 BindInfo 方法，该方法为自定义的无返回值类型方法，主要用来使用 LinqToSql 技术根据指定条件查询员工信息，并将查询结果显示在 DataGridView 控件中。BindInfo 方法实现代码如下。

```
#region 查询员工信息
///<summary>
///查询员工信息
///</summary>
private void BindInfo()
{
    linq = new linqtosqlClassDataContext(strCon); //创建 Linq 连接对象
    if (txtKeyWord.Text == "")
    {
        //获取所有员工信息
        var result = from info in linq.tb_Employee
                    select new
                    {
                        员工编号 = info.ID,
                        员工姓名 = info.Name,
                        性别 = info.Sex,
                        年龄 = info.Age,
```

```

        电话 = info.Tel,
        地址 = info.Address,
        QQ = info.QQ,
        Email = info.Email
    );
    dgvInfo.DataSource = result; //对 DataGridView 控件进行数据绑定
}
else
{
    switch (cboxCondition.Text)
    {
        case "员工编号":
            var resultid = from info in linq.tb_Employee
                           where info.ID == txtKeyWord.Text
                           select new
                           {
                               员工编号 = info.ID,
                               员工姓名 = info.Name,
                               性别 = info.Sex,
                               年龄 = info.Age,
                               电话 = info.Tel,
                               地址 = info.Address,
                               QQ = info.QQ,
                               Email = info.Email
                           };
            dgvInfo.DataSource = resultid;
            break;
        case "员工姓名": //根据员工姓名查询员工信息
            var resultname = from info in linq.tb_Employee
                             where info.Name.Contains(txtKeyWord.Text)
                             select new
                             {
                                 员工编号 = info.ID,
                                 员工姓名 = info.Name,
                                 性别 = info.Sex,
                                 年龄 = info.Age,
                                 电话 = info.Tel,
                                 地址 = info.Address,
                                 QQ = info.QQ,
                                 Email = info.Email
                           };
            dgvInfo.DataSource = resultname;
            break;
        case "性别": //根据员工性别查询员工信息
            var resultsex = from info in linq.tb_Employee
                           where info.Sex == txtKeyWord.Text
                           select new
                           {
                               员工编号 = info.ID,
                               员工姓名 = info.Name,
                               性别 = info.Sex,
                           };
    }
}

```

```

        年龄 = info.Age,
        电话 = info.Tel,
        地址 = info.Address,
        QQ = info.QQ,
        Email = info.Email
    );
    dgvInfo.DataSource = resultsex;
    break;
}
}
#endregion

```

单击“查询”按钮，调用 BindInfo 方法查询员工信息，并将查询结果显示到 DataGridView 控件中。“查询”按钮的 Click 事件代码如下。

```

private void btnQuery_Click(object sender, EventArgs e)
{
    BindInfo();
}

```

程序运行结果如图 14.11 所示。



图 14.11 使用 LINQ 查询 SQL Server 数据库

## 14.2.2 使用 LINQ 管理 SQL Server 数据库

使用 LINQ 管理 SQL Server 数据库时，主要有添加、修改和删除 3 种操作，本节将分别进行详细讲解。

### 1. 添加数据

使用 LINQ 向 SQL Server 数据库中添加数据时，需要用到 InsertOnSubmit 方法和 SubmitChanges 方法，其中 InsertOnSubmit 方法用来将处于 pending insert 状态的实体添加到 SQL 数据表中。

语法：void InsertOnSubmit(Object entity)

说明：entity 表示要添加的实体。

SubmitChanges 方法用来记录要插入、更新或删除的对象，并执行相应命令以实现对数据库的更改。

语法：public void SubmitChanges()

**例 14.06** 创建一个 Windows 应用程序，Form1 窗体设计为如图 14.12 所示的界面。首先在当前项目中创建一个 LinqToSql 类文件，然后在 Form1 窗体中定义一个 string 类型的变量，用来记录数据库连接字符串，并声明 Linq 连接对象，代码如下。（实例位置：光盘\mr\14\s\14.06）

```

string strCon = "Data Source=MRWXK\WANGXIAOKE;Database=db_CSharp;Uid=sa;Pwd=";
//定义数据库连接字符串

```

```
linqtosqlClassDataContext linq; //声明 Linq 连接对象
```

在 Form1 窗体中单击“添加”按钮，首先创建 Linq 连接对象，然后创建 tb\_Employee 对象，该类为对应的 tb\_Employee 数据表类，为 tb\_Employee 对象中的各个属性赋值，最后调用 Linq 连接对象中的 InsertOnSubmit 方法添加员工信息，并调用其 SubmitChanges 方法将添加员工操作提交服务器。“添加”按钮的 Click 事件代码如下。

```
private void btnAdd_Click(object sender, EventArgs e)
{
    linq = new linqtosqlClassDataContext(strCon); //创建 Linq 连接对象
    tb_Employee employee = new tb_Employee(); //创建 tb_Employee 对象
    //为 tb_Employee 类中的员工实体赋值
    employee.ID = txtID.Text;
    employee.Name = txtName.Text;
    employee.Sex = cboxSex.Text;
    employee.Age = Convert.ToInt32(txtAge.Text);
    employee.Tel = txtTel.Text;
    employee.Address = txtAddress.Text;
    employee.QQ = Convert.ToInt32(txtQQ.Text);
    employee.Email = txtEmail.Text;
    linq.tb_Employee.InsertOnSubmit(employee); //添加员工信息
    linq.SubmitChanges(); //提交操作
    MessageBox.Show("数据添加成功");
    BindInfo();
}
```

上面的代码中用到了 BindInfo 方法，该方法为自定义的无返回值类型方法，主要用来获取所有员工信息，并绑定到 DataGridView 控件上。BindInfo 方法实现代码如下。

```
#region 显示所有员工信息
///<summary>
///显示所有员工信息
///</summary>
private void BindInfo()
{
    linq = new linqtosqlClassDataContext(strCon); //创建 Linq 连接对象
    //获取所有员工信息
    var result = from info in linq.tb_Employee
                select new
                {
                    员工编号 = info.ID,
                    员工姓名 = info.Name,
                    性别 = info.Sex,
                    年龄 = info.Age,
                    电话 = info.Tel,
                    地址 = info.Address,
                    QQ = info.QQ,
                    Email = info.Email
                };
    dgvInfo.DataSource = result; //对 DataGridView 控件进行数据绑定
}
#endregion
```

程序运行结果如图 14.12 所示。



图 14.12 添加数据

## 2. 修改数据

在使用 LINQ 修改 SQL Server 数据库中的数据时需要用到 SubmitChanges 方法，该方法在“添加数据”中已经做过详细介绍，此处不再赘述。

**例 14.07** 创建一个 Windows 应用程序，Form1 窗体设计为如图 14.13 所示的界面。首先在当前项目中创建一个 LinqToSql 类文件，然后在 Form1 窗体中定义一个 string 类型的变量，用来记录数据库连接字符串，并声明 Linq 连接对象，代码如下。（实例位置：光盘\mr\14\sl\14.07）

```
string strCon = "Data Source=MRWXK\WANGXIAOKE;Database=db_CSharp;Uid=sa;Pwd=";
//定义数据库连接字符串
linqtosqlClassDataContext linq; //声明 Linq 连接对象
```

当在 DataGridView 控件中选中某条记录时，根据选中记录的员工编号查找其详细信息，并显示在对应的文本框中，实现代码如下。

```
private void dgvInfo_CellClick(object sender, DataGridViewCellEventArgs e)
{
    linq = new linqtosqlClassDataContext(strCon); //创建 Linq 连接对象
    //获取选中的员工编号
    txtID.Text = Convert.ToString(dgvInfo[0, e.RowIndex].Value).Trim();
    //根据选中的员工编号获取其详细信息，并重新生成一个表
    var result = from info in linq.tb_Employee
                where info.ID == txtID.Text
                select new
                {
                    ID = info.ID,
                    Name = info.Name,
                    Sex = info.Sex,
                    Age = info.Age,
                    Tel = info.Tel,
                    Address = info.Address,
                    QQ = info.QQ,
                    Email = info.Email
                };
    //在相应的文本框及下拉列表中显示选中员工的详细信息
    foreach (var item in result)
    {
        txtName.Text = item.Name;
        cboxSex.Text = item.Sex;
    }
}
```

```

        txtAge.Text = item.Age.ToString();
        txtTel.Text = item.Tel;
        txtAddress.Text = item.Address;
        txtQQ.Text = item.QQ.ToString();
        txtEmail.Text = item.Email;
    }
}
}

```

在 Form1 窗体中单击“修改”按钮，首先判断是否选择了要修改的记录，如果没有，则弹出提示信息，否则创建 Linq 连接对象，并从该对象中的 tb\_Employee 表中查找是否有相关记录；如果有，则为 tb\_Employee 表中的字段赋值，并调用 Linq 连接对象中的 SubmitChanges 方法修改指定编号的员工信息。“修改”按钮的 Click 事件代码如下。

```

private void btnEdit_Click(object sender, EventArgs e)
{
    if (txtID.Text == "")
    {
        MessageBox.Show("请选择要修改的记录");
        return;
    }
    linq = new linqtosqlClassDataContext(strCon); //创建 Linq 连接对象
    //查找要修改的员工信息
    var result = from employee in linq.tb_Employee
                 where employee.ID == txtID.Text
                 select employee;
    //对指定的员工信息进行修改
    foreach (tb_Employee tbemployee in result)
    {
        tbemployee.Name = txtName.Text;
        tbemployee.Sex = cboxSex.Text;
        tbemployee.Age = Convert.ToInt32(txtAge.Text);
        tbemployee.Tel = txtTel.Text;
        tbemployee.Address = txtAddress.Text;
        tbemployee.QQ = Convert.ToInt32(txtQQ.Text);
        tbemployee.Email = txtEmail.Text;
        linq.SubmitChanges();
    }
    MessageBox.Show("员工信息修改成功");
    BindInfo();
}

```

上面的代码中用到了 BindInfo 方法，该方法为自定义的无返回值类型方法，主要用来获取所有员工信息，并绑定到 DataGridView 控件上。BindInfo 方法实现代码如下。

```

#region 显示所有员工信息
///<summary>
///显示所有员工信息
///</summary>
private void BindInfo()
{
    linq = new linqtosqlClassDataContext(strCon); //创建 Linq 连接对象
    //获取所有员工信息
    var result = from info in linq.tb_Employee
                select new

```

```

{
    员工编号 = info.ID,
    员工姓名 = info.Name,
    性别 = info.Sex,
    年龄 = info.Age,
    电话 = info.Tel,
    地址 = info.Address,
    QQ = info.QQ,
    Email = info.Email
};

dgvInfo.DataSource = result; //对 DataGridView 控件进行数据绑定
}
#endifregion

```

程序运行结果如图 14.13 所示。



图 14.13 修改数据

### 3. 删除数据

在使用 LINQ 删 SQL Server 数据库中的数据时需要用到 DeleteAllOnSubmit 方法和 SubmitChanges 方法，其中 SubmitChanges 方法在“添加数据”中已经做过详细介绍，这里主要讲解 DeleteAllOnSubmit 方法。DeleteAllOnSubmit 方法用来将集合中的所有实体置于 pending delete 状态。

语法：void DeleteAllOnSubmit(IEnumerable entities)

说明：entities 表示要移除所有项的集合。

**例 14.08** 创建一个 Windows 应用程序，在 Form1 窗体中添加一个 ContextMenuStrip 控件，用来作为“删除”快捷菜单；添加一个 DataGridView 控件，用来显示数据库中的数据，并将 DataGridView 控件的 ContextMenuStrip 属性设置为 contextMenuStrip1。（实例位置：光盘\mr\14\sl\14.08）

首先在当前项目中依照上面所讲的步骤创建一个 LinqToSql 类文件，然后在 Form1 窗体中定义一个 string 类型的变量，用来记录数据库连接字符串，并声明 Linq 连接对象，再声明一个 string 类型的变量，用来记录选中的员工编号。代码如下。

```

string strCon = "Data Source=MRWXK\WANGXIAOKE;Database=db_CSharp;Uid=sa;Pwd=";
//定义数据库连接字符串
linqtosqlClassDataContext linq; //声明 Linq 连接对象
string strID = ""; //记录选中的员工编号

```

在 DataGridView 控件中选择行时，记录当前选中行的员工编号，并赋值给定义的全局变量，代码如下。

```
private void dgvInfo_CellClick(object sender, DataGridViewCellEventArgs e)
```

```
{
```

```
strID = Convert.ToString(dgvInfo[0, e.RowIndex].Value).Trim(); //获取选中的员工编号
}
```

在 DataGridView 控件上单击鼠标右键，在弹出的快捷菜单中选择“删除”命令，首先判断要删除的员工编号是否为空，如果为空，则弹出提示信息，否则创建 Linq 连接对象，并从该对象中的 tb\_Employee 表中查找是否有相关记录；如果有，则调用 Linq 连接对象中的 DeleteAllOnSubmit 方法删除员工信息，并调用其 SubmitChanges 方法将删除员工操作提交服务器。“删除”菜单项的 Click 事件代码如下。

```
private void 删除 ToolStripMenuItem_Click(object sender, EventArgs e)
{
    if (strID == "")
    {
        MessageBox.Show("请选择要删除的记录");
        return;
    }
    linq = new linqtosqlClassDataContext(strCon); //创建 Linq 连接对象
    //查找要删除的员工信息
    var result = from employee in linq.tb_Employee
                 where employee.ID == strID
                 select employee;
    linq.tb_Employee.DeleteAllOnSubmit(result); //删除员工信息
    linq.SubmitChanges(); //提交操作
    MessageBox.Show("员工信息删除成功");
    BindInfo();
}
```

上面的代码中用到了 BindInfo 方法，该方法为自定义的无返回值类型方法，主要用来获取所有员工信息，并绑定到 DataGridView 控件上。BindInfo 方法实现代码如下。

```
#region 显示所有员工信息
///<summary>
///显示所有员工信息
///</summary>
private void BindInfo()
{
    linq = new linqtosqlClassDataContext(strCon); //创建 Linq 连接对象
    //获取所有员工信息
    var result = from info in linq.tb_Employee
                select new
                {
                    员工编号 = info.ID,
                    员工姓名 = info.Name,
                    性别 = info.Sex,
                    年龄 = info.Age,
                    电话 = info.Tel,
                    地址 = info.Address,
                    QQ = info.QQ,
                    Email = info.Email
                };
    dgvInfo.DataSource = result; //对 DataGridView 控件进行数据绑定
}
#endregion
```

程序运行结果如图 14.14 所示。

员工编号	员工姓名	性别	年龄	电话
TGBH0001	王	男	22	22222
TGBH0001	李	男	24	1310000000
TGBH0002	小李	男	28	1300000000
TGBH0003	小刘	男	24	1380000000
TGBH0004	小吕	男	28	1300000000
TGBH0005	小王	男	28	1300000000

图 14.14 删除数据

### 14.3 LINQ 操作其他数据

#### 14.3.1 使用 LINQ 操作数组和集合

对数组和集合进行操作时可以使用 LinqToObjects 技术，它是一种新的处理集合的方法，如果采用旧方法，程序开发人员必须编写指定如何从集合检索数据的复杂的 foreach 循环，而采用 LinqToObjects 技术，只需编写描述要检索的内容的声明性代码即可。LinqToObjects 能够直接使用 LINQ 查询 IEnumerable 或 IEnumerable<T> 集合，而不需要使用 LINQ 提供程序或 API，可以说，使用 LINQ 能够查询任何可枚举的集合，如数组、泛型列表等。

下面通过一个实例讲解如何使用 LINQ 技术操作数组和集合。

**例 14.09** 创建一个控制台应用程序，在 Main 方法中定义一个一维数组，然后使用 LINQ 技术从该数组中查找及格范围内的分数，最后循环访问查询结果并输出。(实例位置：光盘\mr\14\sl\14.09)

```
static void Main(string[] args)
{
    int[] intScores = { 45, 68, 80, 90, 75, 76, 32 }; // 定义 int 类型的一维数组
    // 使用 LINQ 技术从数组中查找及格范围内的分数
    var score = from hgScroe in intScores
                where hgScroe >= 60
                orderby hgScroe ascending
                select hgScroe;
    Console.WriteLine("及格的分数：");
    foreach (var v in score) // 循环访问查询结果并显示
    {
        Console.WriteLine(v.ToString());
    }
    Console.ReadLine();
}
```

程序运行结果如图 14.15 所示。



图 14.14 删除数据

### 14.3.2 使用 LINQ 操作 DataSet 数据集

对 DataSet 数据集进行操作时可以使用 LINQ to DataSet 技术，它是 LINQ to ADO.NET 中的独立技术，使用该技术查询 DataSet 对象更加方便快捷。下面对 LINQ to DataSet 技术中常用到的方法进行详细讲解。

#### (1) AsEnumerable 方法

AsEnumerable 方法可以将 DataTable 对象转换为 EnumerableRowCollection<DataRow>对象。

语法：public static EnumerableRowCollection<DataRow> AsEnumerable(this DataTable source)

说明：source 表示可枚举的源 DataTable。它的返回值是一个 IEnumerable<T>对象，其泛型参数 T 为 DataRow。

#### (2) CopyToDataTable 方法

CopyToDataTable 方法用来将 IEnumerable<T>对象中的数据赋值到 DataTable 对象中。

语法：public static DataTable CopyToDataTable<T>(this IEnumerable<T> source) where T : DataRow

说明：source 表示源 IEnumerable<T>序列。它的返回值为一个 DataTable，其中包含作为 DataRow 对象的类型的输入序列。

#### (3) AsDataView 方法

AsDataView 方法用来创建并返回支持 LINQ 的 DataView 对象。

语法：public static DataView AsDataView<T>(this EnumerableRowCollection<T> source) where T : DataRow

说明：source 指定从中创建支持 LINQ 的 DataView 的源 LINQ to DataSet 查询。它的返回值表示支持 LINQ 的 DataView 对象。

#### (4) Take 方法

Take 方法用来从序列的开头返回指定数量的连续元素。

语法：public static IEnumerable<TSource> Take<TSource>(this IEnumerable<TSource> source,int count)

说明：source 表示要从其返回元素的序列；count 表示要返回的元素数量。另外，它的返回值是一个 IEnumerable<T>，包含输入序列开头的指定数量的元素。

#### (5) Sum 方法

Sum 方法用来计算数值序列之和。

语法：public static decimal Sum(this IEnumerable<decimal> source)

说明：source 表示一个要计算和的 Decimal 值序列。它的返回值表示序列值之和。

 说明：上面介绍的几种方法都有多种重载形式，这里只介绍其常用到的重载形式。

下面通过一个实例讲解如何使用 LINQ 技术在 DataGridView 控件中显示 DataSet 数据集中的数据。

**例 14.10** 创建一个 Windows 应用程序，在 Form1 窗体中添加一个 DataGridView 控件，用来显示 DataSet 数据集中的数据。窗体加载时，首先将数据库中的数据填充到 DataSet 数据集中，然后使用 LINQ 技术从 DataSet 数据集中查找信息并显示在 DataGridView 控件中。代码如下。（实例位置：光盘\mr\14\sl\14.10）

```
private void Form1_Load(object sender, EventArgs e)
{
    string strCon = "Data Source=MRWXK\WANGXIAOKE;Database=db_CSharp;Uid=sa;Pwd='';";
    //定义数据库连接字符串
    SqlConnection sqlcon;
    SqlDataAdapter sqlda;
    DataSet myds;
    //声明 SqlConnection 对象
    //声明 SqlDataAdapter 对象
    //声明 DataSet 数据集对象
```

```

sqlcon = new SqlConnection(strCon); //创建数据库连接对象
//创建数据库桥接器对象
sqlda = new SqlDataAdapter("select * from tb_Salary", sqlcon);
myds = new DataSet();
sqlda.Fill(myds, "tb_Salary");
//从数据集中查询所有数据
var query = from salary in myds.Tables["tb_Salary"].AsEnumerable()
            select salary;
//将查询结果转化为 DataTable
DataTable myDTable = query.CopyToDataTable<DataRow>();
dataGridView1.DataSource = myDTable; //显示查询到的信息
}

```

程序运行结果如图 14.16 所示。

ID	Name	Salary
TGBH0001	小王	1500
TGBH0002	小李	1600
TGBH0003	小刘	1500
TGBH0004	小科	1500
TGBH0005	小美	1500

图 14.16 使用 LINQ 操作 DataSet 数据集

### 14.3.3 使用 LINQ 操作 XML

对 XML 文件进行操作时可以使用 LINQ to XML 技术，它是 LINQ 技术中的一种，其提供了修改文档对象模型的内存文档，并支持 LINQ 查询表达式等功能。下面对 LINQ to XML 技术中常用到的方法进行详细讲解。

#### (1) XElement 类的 Load 方法

XElement 类表示一个 XML 元素，其 Load 方法用来从文件加载 XElement。

语法：public static XElement Load(string uri)

说明：uri 表示一个 URI 字符串，用来引用要加载到新 XElement 中的文件。它的返回值为一个包含所指定文件的内容的 XElement。

#### (2) XElement 类的 SetAttributeValue 方法

SetAttributeValue 方法用来设置属性的值、添加属性或移除属性。

语法：public void SetAttributeValue(XName name, Object value)

说明：name 表示一个 XName，其中包含要更改的属性的名称；Value 表示分配给属性的值，如果该值为 null，则移除该属性；否则会将值转换为其字符串表示形式，并分配给该属性的 Value 属性。

#### (3) XElement 类的 Add 方法

Add 方法用来将指定的内容添加为此 XContainer 的子级。

语法：public void Add(Object content)

说明：content 表示要添加的包含简单内容的对象或内容对象集合。

#### (4) XElement 类的 ReplaceNodes 方法

ReplaceNodes 方法用来使用指定的内容替换此文档或元素的子节点。

语法：public void ReplaceNodes(Object content)

说明：content 表示一个用于替换子节点的包含简单内容的对象或内容对象集合。

#### (5) XElement 类的 Save 方法

Save 方法用来序列化此元素的基础 XML 树，可以将输出保存到文件、XmlTextWriter、TextWriter 或 XmlWriter。

语法：public void Save(string fileName)

说明：fileName 表示一个包含文件名称的字符串。

#### (6) XDocument 类的 Save 方法

XDocument 类表示 XML 文档，其 Save 方法用来将此 XDocument 序列化为文件、TextWriter 或 XmlWriter。

语法：public void Save(string fileName)

说明：fileName 表示一个包含文件名称的字符串。

#### (7) XDeclaration 类

XDeclaration 类表示一个 XML 声明，其构造函数语法格式如下。

语法：public XDeclaration(string version, string encoding, string standalone)

说明：version 指定 XML 的版本，通常为“1.0”；encoding 指定 XML 文档的编码；standalone 包含“yes”或“no”的字符串，用来指定 XML 是独立的还是需要解析外部实体。

说明：使用 LINQ to XML 技术中的类时，需要添加 System.Linq.Xml 命名空间。

下面通过一个实例讲解如何使用 LINQ 技术对 XML 文件进行操作。

**例 14.11** 创建一个 Windows 应用程序，Form1 窗体设计为如图 14.17 所示的界面。在 Form1 窗体中先定义两个字符串类型的全局变量，分别用来记录 XML 文件路径及选中的 ID 编号，代码如下。（实例位置：光盘\mr\14\sl\14.11）

```
static string strPath = "Employee.xml"; //记录 XML 文件路径
static string strID = ""; //记录选中的 ID 编号
```

Form1 窗体加载时，将 XML 文件中的数据显示在 DataGridView 控件中。Form1 窗体的 Load 事件代码如下。

```
private void Form1_Load(object sender, EventArgs e)
{
    getXmlInfo(); //窗体加载时加载 XML 文件
}
```

上面的代码中用到了 getXmlInfo 方法，该方法为自定义的无返回值类型方法，它主要用来将 XML 文件中的内容绑定到 DataGridView 控件中。getXmlInfo 方法实现代码如下。

```
#region 将 XML 文件内容绑定到 DataGridView 控件
///<summary>
///将 XML 文件内容绑定到 DataGridView 控件
///</summary>
private void getXmlInfo()
{
    DataSet myds = new DataSet(); //创建 DataSet 数据集对象
    myds.ReadXml(strPath); //读取 XML 结构
    dataGridView1.DataSource = myds.Tables[0]; //在 DataGridView 中显示 XML 文件中的信息
}
#endregion
```

单击“添加”按钮，使用 LINQ to XML 技术向指定的 XML 文件中插入用户输入的数据，并重新保存 XML 文件。“添加”按钮的 Click 事件代码如下。

```

private void button2_Click(object sender, EventArgs e)
{
    XElement xe = XElement.Load(strPath); //加载 XML 文档
    //创建 IEnumerable 泛型接口
    IEnumerable<XElement> elements1 = from element in xe.Elements("People")
                                         select element;
    //生成新的编号
    string str = (Convert.ToInt32(elements1.Max(element => element.Attribute("ID").Value)) + 1).ToString("000");
    XElement people = new XElement("People", new XAttribute("ID", str), //创建 XML 元素
                                   new XElement("Name", textBox11.Text), //为 XML 元素设置属性
                                   new XElement("Sex", comboBox1.Text),
                                   new XElement("Salary", textBox12.Text));
    xe.Add(people);
    xe.Save(strPath); //添加 XML 元素
    getXmlInfo(); //保存 XML 元素到 XML 文件
}

```

当用户在 DataGridView 控件中选择某记录时，使用 LINQ to XML 技术在 XML 文件中查找选中记录的详细信息，并显示到相应的文本框和下拉列表中。实现代码如下。

```

private void dataGridView1_CellClick(object sender, DataGridViewCellEventArgs e)
{
    strID = dataGridView1.Rows[e.RowIndex].Cells[3].Value.ToString(); //记录选中的 ID 编号
    XElement xe = XElement.Load(strPath); //加载 XML 文档
    //根据编号查找信息
    IEnumerable<XElement> elements = from PInfo in xe.Elements("People")
                                         where PInfo.Attribute("ID").Value == strID
                                         select PInfo;
    foreach ( XElement element in elements) //遍历查找到的所有信息
    {
        textBox11.Text = element.Element("Name").Value; //显示员工姓名
        comboBox1.SelectedItem = element.Element("Sex").Value; //显示员工性别
        textBox12.Text = element.Element("Salary").Value; //显示员工薪水
    }
}

```

单击“修改”按钮，首先判断是否选定要修改的记录，如果已经选定，则使用 LINQ to XML 技术修改 XML 文件中的指定记录，并重新保存 XML 文件。“修改”按钮的 Click 事件代码如下。

```

private void button3_Click(object sender, EventArgs e)
{
    if (strID != "") //判断是否选择了编号
    {
        XElement xe = XElement.Load(strPath); //加载 XML 文档
        //根据编号查找信息
        IEnumerable<XElement> elements = from element in xe.Elements("People")
                                         where element.Attribute("ID").Value == strID
                                         select element;
        if (elements.Count() > 0) //判断是否找到了信息
        {
            XElement newXE = elements.First(); //获取找到的第一条记录

```

```

newXE.SetAttributeValue("ID", strID); //为 XML 元素设置属性值
newXE.ReplaceNodes(
    new XElement("Name", textBox11.Text),
    new XElement("Sex", comboBox1.Text),
    new XElement("Salary", textBox12.Text)
);
}
xe.Save(strPath); //保存 XML 元素到 XML 文件
}
getXmlInfo();
}

```

单击“删除”按钮，首先判断是否选定要删除的记录，如果已经选定，则使用 LINQ to XML 技术删除 XML 文件中的指定记录，并重新保存 XML 文件。“删除”按钮的 Click 事件代码如下。

```

private void button4_Click(object sender, EventArgs e)
{
    if (strID != "") //判断是否选择了编号
    {
        XElement xe = XElement.Load(strPath); //加载 XML 文档
        //根据编号查找信息
        IEnumerable<XElement> elements = from element in xe.Elements("People")
                                         where element.Attribute("ID").Value == strID
                                         select element;
        if (elements.Count() > 0) //判断是否找到了信息
            elements.First().Remove(); //删除找到的 XML 元素信息
        xe.Save(strPath); //保存 XML 元素到 XML 文件
    }
    getXmlInfo();
}

```

程序运行结果如图 14.17 所示。

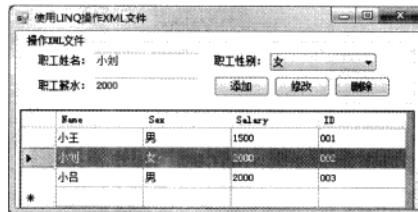


图 14.17 使用 LINQ 操作 XML 文件

## 14.4 照猫画虎——基本功训练

### 14.4.1 基本功训练 1——检查序列中是否包含指定元素

视频讲解：光盘\mr\14\lx\检查序列中是否包含指定元素.exe

实例位置：光盘\mr\14\zmhh\01

新建一个 Windows 窗体应用程序，并向窗体中添加 3 个 Label 控件，分别用来显示数据源、查询表达

式和查询结果。然后在窗体的 Load 加载事件中编写如下代码。

```
private void Frm_Main_Load(object sender, EventArgs e)
{
    List<Person> People = new List<Person>(); //创建人员列表
    for (int i = 1; i < 10; i++)
    {
        People.Add(new Person(i, "User0" + i.ToString()));
    }
    bool result = People.Contains(new Person(3, "User03")); //检查序列是否包含指定元素
    //显示查询结果
    label1.Text = "数据源: People(包含 9 个元素, 元素的 ID 值分别为 1、2、3、.....、9)"; //数据源
    label2.Text = "查询表达式: Contains(new Person(3, "User03"))"; //查询表达式/操作
    label3.Text = "查询结果: " + result.ToString(); //查询结果
}
```

程序运行结果如图 14.18 所示。

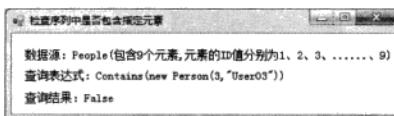


图 14.18 检查序列中是否包含指定元素

**熊猫画虎：**制作一个程序，用来检查序列中的所有元素是否都满足指定条件。提示：使用 Enumerable 类的 All 方法实现。（20 分）（实例位置：光盘\mr\14\zmhh\01\_zmhh）

#### 14.4.2 基本功训练 2——使用 LINQ 生成随机序列

■ 视频讲解：光盘\mr\14\lx\使用 LINQ 生成随机序列.exe

■ 实例位置：光盘\mr\14\zmhh\02

新建一个控制台应用程序，然后在 Program.cs 文件中编写如下代码。

```
static void Main(string[] args)
{
    Random rand = new Random(); //创建一个随机数生成器
    Console.WriteLine("请输入一个整数: ");
    try
    {
        int intCount = Convert.ToInt32(Console.ReadLine()); //输入要生成随机数的组数
        //生成一个包含指定个数的重复元素值的序列
        //由于 Linq 的延迟性，所以此时并不产生随机数，而是在枚举 randomSeq 时生成随机数
        IEnumerable<int> randomSeq = Enumerable.Repeat<int>(1, intCount).Select(i => rand.Next());
        Console.WriteLine("将产生 " + intCount.ToString() + " 个随机数: ");
        foreach (int item in randomSeq) //通过枚举序列来生成随机数
        {
            Console.WriteLine(item.ToString()); //输出若干组随机数
        }
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
    }
}
```

```

    }
    Console.Read();
}

```

程序运行结果如图 14.19 所示。



图 14.19 使用 LINQ 生成随机序列

**照猫画虎：**根据以上程序，制作一个生成随机字符串的程序。(20 分)(实例位置：光盘\mr\14\zmhh\02\_zmhh)

#### 14.4.3 基本功训练 3——统计每种商品的销售次数

■ 视频讲解：光盘\mr\14\lx\统计每种商品的销售次数.exe

■ 实例位置：光盘\mr\14\zmhh\03

新建一个 Windows 窗体应用程序，在窗体中添加一个 DataGridView 控件，用来显示每种商品的销售次数。然后按照 14.2.1 节中讲解的“创建 LinqToSql 类文件”步骤创建 LINQ to SQL 的 dbml 文件，并将 V\_SaleDetail 视图添加到 dbml 文件中。最后在窗体的 Load 加载事件中编写如下代码。

```

private void Frm_Main_Load(object sender, EventArgs e)
{
    DataClassesDataContext dc = new DataClassesDataContext(); //创建数据上下文类的实例
    //统计每种商品的销售次数
    var query = from det in dc.V_SaleDetail
                group det by det.ProductCode into g //按商品代码分组
                select new
                {
                    商品代码 = g.Key,
                    商品名称 = g.Max(itm => itm.ProductName),
                    //使用 Count 方法统计销售次数
                    销售次数 = g.Count()
                };
    //将查询结果集绑定到 dataGridView1
    dataGridView1.DataSource = query;
}

```

程序运行结果如图 14.20 所示。

统计每种商品的销售次数		
商品代码	商品名称	销售次数
200911170935...	液晶电视	2
200911170940...	液晶200	1
200911170941...	洗衣机50	5

图 14.20 统计每种商品的销售次数

**照猫画虎：**根据以上程序的实现原理，制作一个可以统计每种商品销售总额的程序。提示：使用 **IEnumerable** 泛型接口的 **Sum** 方法实现。（20 分）（实例位置：光盘\mr\14\zmhh\03\_zmhh）

#### 14.4.4 基本功训练 4——统计每种商品的销售均价

■**视频讲解：**光盘\mr\14\lx\统计每种商品的销售均价.exe

■**实例位置：**光盘\mr\14\zmhh\04

新建一个 Windows 窗体应用程序，在窗体中添加一个 **DataGridView** 控件，用来显示每种商品的销售均价。然后按照 14.2.1 节中讲解的“创建 **LinqToSql** 类文件”步骤创建 LINQ to SQL 的 **dbml** 文件，并将 **V\_SaleDetail** 视图添加到 **dbml** 文件中。最后在窗体的 **Load** 加载事件中编写如下代码。

```
private void Frm_Main_Load(object sender, EventArgs e)
{
    DataClassesDataContext dc = new DataClassesDataContext(); //创建数据上下文类的实例
    //统计每种商品的销售平均价
    var query = from det in dc.V_SaleDetail
                group det by det.ProductCode into g //按商品代码分组
                select new
                {
                    商品代码 = g.Key,
                    商品名称 = g.Max(item => item.ProductName),
                    //使用 Average 统计每种商品的销售平均价
                    销售平均价 = g.Average(item => item.Price)
                };
    //将查询的结果集绑定到 dataGridView1
    dataGridView1.DataSource = query;
}
```

程序运行结果如图 14.21 所示。

商品代码	商品名称	销售平均价
200911170940...	液晶200	3200
200911170937...	液晶电视	3750
200911170941...	洗衣机50	520

图 14.21 统计每种商品的销售均价

**照猫画虎：**根据以上程序的实现原理，制作一个可以统计每种商品最高销售价的程序。提示：使用 **IEnumerable** 泛型接口的 **Max** 方法实现。（20 分）（实例位置：光盘\mr\14\zmhh\04\_zmhh）

#### 14.4.5 基本功训练 5——获取有过返货记录的商品列表

■**视频讲解：**光盘\mr\14\lx\获取有过返货记录的商品列表.exe

■**实例位置：**光盘\mr\14\zmhh\05

新建一个 Windows 窗体应用程序，在窗体中添加一个 **DataGridView** 控件，用来显示有过返货记录的商品列表。然后按照 14.2.1 节中讲解的“创建 **LinqToSql** 类文件”步骤创建 LINQ to SQL 的 **dbml** 文件，并将 **V\_SaleInfo** 视图和 **V\_SaleReturnInfo** 视图添加到 **dbml** 文件中。最后在窗体的 **Load** 加载事件中编写如下代码。

```
private void Frm_Main_Load(object sender, EventArgs e)
{
```

```

//创建数据上下文类的实例
DataClassesDataContext dc = new DataClassesDataContext();
var saleInfo = from si in dc.V_SaleInfo
               select new
               {
                   商品代码 = si.ProductCode,
                   商品名称 = si.ProductName
               };
var saleRetu = from sr in dc.V_SaleReturnInfo
               select new
               {
                   商品代码 = sr.ProductCode,
                   商品名称 = sr.ProductName
               };
var query = saleInfo.Intersect(saleRetu);
//取交集
//将查询的结果集绑定到 dataGridView1
dataGridView1.DataSource = query;
}

```

程序运行结果如图 14.22 所示。

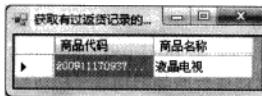


图 14.22 获取有过返货记录的商品列表

**照猫画虎：**制作一个程序，主要用来获取从未返过货的商品列表。提示：可使用 `IEnumerable` 泛型接口的 `Except` 方法实现。（20 分）（实例位置：光盘\mr\14\zmhh\05\_zmhh）

照猫画虎栏目分数统计：

照猫画虎题目	1	2	3	4	5	总分数	
分数							

## 14.5 情景应用——拓展与实践

### 14.5.1 情景应用 1——使用存储过程查询单表数据

视频讲解：光盘\mr\14\lx\使用存储过程查询单表数据.exe

实例位置：光盘\mr\14\qjyy\01

本实例中使用 LINQ to SQL 调用存储过程，并根据传入的仓库助记码查询仓库信息表。新建一个 Windows 窗体应用程序，在窗体中添加一个 TextBox 控件，用来输入仓库助记码；添加一个 Button 控件，用来调用存储过程执行查询操作；添加一个 DataGridView 控件，用来显示查询结果。然后按照 14.2.1 节中讲解的“创建 LinqToSql 类文件”步骤创建 LINQ to SQL 的 dbml 文件，并将 `P_queryWarehouseInfo` 存储过程添加到 dbml 文件中。最后在 Button 控件的 Click 事件中编写如下代码。

```

private void button1_Click(object sender, EventArgs e)
{
}

```

```

DataClassesDataContext dc = new DataClassesDataContext();
var query = dc.P_queryWarehouseInfo(textBox1.Text);
dataGridView1.DataSource = query;
}

```

**说明：**在 LINQ 中调用存储过程时，只需要将要调用的存储过程添加到 LINQ to SQL 对应的 dbml 文件中即可。

程序运行结果如图 14.23 所示。

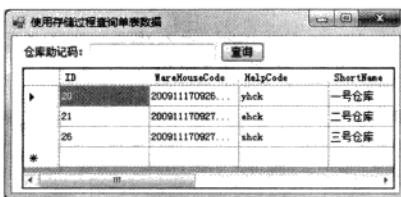


图 14.23 使用存储过程查询单表数据

**DIY：**制作一个程序，使用 LINQ 技术实现关联查询多表数据的功能。提示：在 LINQ 查询表达式中使用 join 子句实现多表连接查询。(20 分)(实例位置：光盘\mr\14\qjyy\01\_diy)

#### 14.5.2 情景应用 2——使用 LINQ 技术防止 SQL 注入式攻击

**视频讲解：**光盘\mr\14\lx\使用 LINQ 技术防止 SQL 注入式攻击.exe

**实例位置：**光盘\mr\14\qjyy\02

本实例使用 LINQ 技术实现防止 SQL 注入式攻击的功能。新建一个 Windows 窗体应用程序，在窗体中添加两个 TextBox 控件，分别用来输入用户名和密码；添加一个 Button 控件，用来执行用户登录操作。然后按照 14.2.1 节中讲解的“创建 LinqToSql 类文件”步骤创建 LINQ to SQL 的 dbml 文件，并将 tb\_Login 表添加到 dbml 文件中。最后在 Button 控件的 Click 事件中编写如下代码。

```

private void btnLogin_Click(object sender, EventArgs e)
{
    string name = txtName.Text;                                //获取用户名
    string pwd = txtPwd.Text;                                 //获取密码
    LinqClassDataContext linqDataContext = new LinqClassDataContext(); //创建 LINQ 对象
    //创建 LINQ 查询语句，查询满足指定用户名和密码的用户
    var result = from v in linqDataContext.tb_Login
                where v.Name == name && v.Pwd == pwd
                select v;
    if (result.Count() > 0)
    {
        //输出相应信息
        MessageBox.Show("登录成功");
    }
    else
    {
        MessageBox.Show("登录失败！");
    }
}

```

程序运行结果如图 14.24 所示。

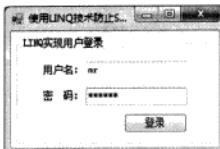


图 14.24 使用 LINQ 技术防止 SQL 注入式攻击

**技巧：**与一般的情况相比，使用 LINQ 防止 SQL 注入攻击，省去了很多代码，操作性更强。其实，只要按照 LINQ 的语法操作数据库，就能防止 SQL 注入攻击。

**DIY：**通过在 SQL 语句中使用 @ 参数实现与上面程序相同的功能。(20 分)(实例位置：光盘\mr\14\qjyy\02\_diy)

### 14.5.3 情景应用 3——使用 LINQ 技术实现数据分页

**视频讲解：**光盘\mr\14\lx\使用 LINQ 技术实现数据分页.exe

**实例位置：**光盘\mr\14\qjyy\03

数据的分页查看在 Windows 应用程序中经常遇到，但是 Visual Studio 2008 开发环境自带的数据控件 DataGridView 并没有此项功能，那么这时就需要开发人员自己编写代码来实现数据分页功能。本实例将演示如何使用 LINQ 技术来方便地实现数据分页功能。运行效果如图 14.25 所示。



图 14.25 使用 LINQ 技术实现数据分页

实现过程如下。

- (1) 创建一个 Windows 窗体应用程序，并将其命名为 LinqPages。
- (2) 更改默认窗体 Form1 的 Name 属性为 Frm\_Main。在窗体中添加一个 DataGridView 控件，显示数据库中的数据；添加两个 Button 控件，分别用来执行上一页和下一页操作。
- (3) 按照 14.2.1 节中讲解的“创建 LinqToSql 类文件”步骤创建 LINQ to SQL 的 dbml 文件，并将 Address 表添加到 dbml 文件中。
- (4) 在窗体的代码页中，首先创建 LINQ 对象，并定义两个 int 类型的变量，分别用来记录每页显示的记录数和当前页数，代码如下。

```
LinqClassDataContext linqDataContext = new LinqClassDataContext(); //创建 LINQ 对象
int pageSize = 7; //设置每页显示 7 条记录
int page = 0; //记录当前页
```

- (5) 自定义一个 getCount 方法，用来根据数据库中的记录计算总页数，代码如下。

```

protected int getCount()
{
    int sum = linqDataContext.Address.Count(); //设置总数据行数
    int s1 = sum / pageSize; //获取可以分的页面
    //总行数对页数求余后是否大于 0，如果大于，则获取 1；否则获取 0
    int s2 = sum % pageSize > 0 ? 1 : 0;
    int count = s1 + s2; //计算出总页数
    return count;
}

```

(6) 自定义一个 bindGrid 方法，用来根据当前页获取指定区间的记录，并显示在 DataGridView 控件中，代码如下。

```

protected void bindGrid()
{
    int pageIndex = Convert.ToInt32(page); //获取当前页数
    //使用 LINQ 查询，并对查询的数据进行分页
    var result = (from v in linqDataContext.Address
                 select new
                 {
                     地址编号 = v.AddressID,
                     城市 = v.City,
                     邮政编码 = v.PostalCode,
                     省份编号 = v.StateProvinceID
                 }).Skip(pageSize * pageIndex).Take(pageSize);
    dgvInfo.DataSource = result; //设置 DataGridView 控件的数据源
    btnBack.Enabled = btnNext.Enabled = true;
    //判断是否为第一页，如果为第一页，则禁用首页和上一页按钮
    if (page == 0)
    {
        btnBack.Enabled = false;
    }
    //判断是否为最后一页，如果为最后一页，则禁用尾页和下一页按钮
    if (page == getCount() - 1)
    {
        btnNext.Enabled = false;
    }
}

```

(7) 窗体加载时，设置当前页为第一页，并调用 bindGrid 方法显示指定的记录，代码如下。

```

private void Form1_Load(object sender, EventArgs e)
{
    page = 0; //设置当前页面
    bindGrid(); //调用自定义的 bindGrid 方法绑定 DataGridView 控件
}

```

(8) 单击“上一页”按钮，使用当前页的索引减 1 作为将要显示的页，并调用 bindGrid 方法显示指定的记录，代码如下。

```

private void btnBack_Click(object sender, EventArgs e)
{
    page = page - 1; //设置当前页数为当前页数减 1
    bindGrid(); //调用自定义的 bindGrid 方法绑定 DataGridView 控件
}

```

(9) 单击“下一页”按钮，使用当前页的索引加 1 作为将要显示的页，并调用 bindGrid 方法显示指定的记录，代码如下。

```
private void btnNext_Click(object sender, EventArgs e)
{
    page = page + 1;                                //设置当前页数为当前页数加 1
    bindGrid();                                     //调用自定义的 bindGrid 方法绑定 DataGridView 控件
}
```

**DIY：**完善以上程序，为其增加“首页”和“尾页”功能。(20 分)(实例位置：光盘\mr\14\qjyy\03\_diy)

#### 14.5.4 情景应用 4——从头开始提取满足指定条件的记录

■ 视频讲解：光盘\mr\14\lx\从头开始提取满足指定条件的记录.exe

■ 实例位置：光盘\mr\14\qjyy\04

本实例使用 LINQ to DataSet 技术实现从头开始提取所有生日小于 2009-7-1 的员工信息的功能。新建一个 Windows 窗体应用程序，在窗体中添加一个 DataGridView 控件，用来显示生日小于 2009-7-1 的所有员工信息。然后在窗体的 Load 加载事件中编写如下代码。

```
private void Frm_Main_Load(object sender, EventArgs e)
{
    string conStr = "Data Source=MRWXK\WANGXIAOKE;Database=db_CSharp;UID=sa;Pwd='';//定义连接字符串
    string sql = "select * from EmployeeInfo";                                         //构造 SQL 语句
    DataSet ds = new DataSet();                                                       //创建数据集
    using (SqlConnection con = new SqlConnection(conStr))                            //创建数据库连接
    {
        //创建 Command 对象
        SqlCommand cmd = new SqlCommand(sql, con);
        //创建 DataAdapter 对象
        SqlDataAdapter sda = new SqlDataAdapter(cmd);
        //填充数据集
        sda.Fill(ds, "EmployeeInfo");
    }
    //从头开始提取生日小于 2009-7-1 的员工信息
    IEnumerable<DataRow> query = ds.Tables["EmployeeInfo"].AsEnumerable().TakeWhile(itm => itm.Field<DateTime>("Birthday") < Convert.ToDateTime("2009-7-1"));
    //设置 dataGridView1 数据源
    dataGridView1.DataSource = query.CopyToDataTable();
}
```

程序运行结果如图 14.26 所示。

Sex	Birthday	Handset	Age
0	2009/1/2	13944196422	87
0	2009/2/2	13756146555	76
0	2009/3/2	13944448888	87
0	2009/4/2	13944196422	76
0	2009/5/2	13756146555	87
0	2009/6/2	13944448888	76

图 14.26 从头开始提取满足指定条件的记录

**DIY：**根据以上程序的实现原理制作一个程序，主要实现从头开始提取指定数量记录的功能。提示：可使用 `Enumerable` 类的 `Take` 方法实现。(20 分)(实例位置：光盘\mr\14\qjyy\04\_diy)

### 14.5.5 情景应用 5——读取 XML 文件并更新到数据库

视频讲解：光盘\mr\14\lx\读取 XML 文件并更新到数据库.exe

实例位置：光盘\mr\14\qjyy\05

XML 是一种类似于 HTML 的标记语言，它以简易而标准的方式保存各种信息（如文字和数字等信息），它适用于不同应用程序间的数据交换。本实例将通过 LINQ 技术实现将 XML 文件中的数据更新到 SQL Server 数据库的功能。运行本实例，首先将 XML 文件中的数据显示在 DataGridView 控件中，如图 14.27 所示，然后单击“更新”按钮，即可将 DataGridView 控件中显示的 XML 数据更新到 SQL Server 数据库的 `tb_XML` 表中。`tb_XML` 表中数据更新前和更新后的效果如图 14.28 所示。

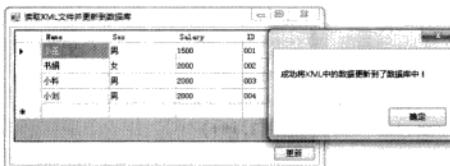


图 14.27 读取 XML 文件并更新到数据库

表 - dbo.tb_XML			
ID	Name	Sex	Salary
001	小王	男	1500
002	书娴	女	2000
003	小科	男	2000
004	小刘	男	2000
NULL	NULL	NULL	NULL

更新前

ID	Name	Sex	Salary
001	小王	男	1500
002	书娴	女	2000
003	小科	男	2000
004	小刘	男	2000
NULL	NULL	NULL	NULL

更新后

图 14.28 `tb_XML` 表中数据更新前和更新后的效果

实现过程如下。

- (1) 创建一个 Windows 窗体应用程序，并将其命名为 `XmLToDatabase`。
- (2) 更改默认窗体 `Form1` 的 `Name` 属性为 `Frm_Main`。在窗体中添加一个 `DataGridView` 控件，用来显示 XML 文件中的内容；添加一个 `Button` 控件，用来执行将 XML 文件中的数据更新到 SQL Server 数据库的操作。
- (3) 按照 14.2.1 节中讲解的“创建 `LinqToSql` 类文件”步骤创建 LINQ to SQL 的 `dbml` 文件，并将 `tb_XML` 表添加到 `dbml` 文件中。

(4) 在窗体的代码页中，首先创建 LINQ 对象，并定义两个 `int` 类型的变量，分别用来记录每页显示的记录数和当前页数，代码如下。

```
static string strPath = "Employee.xml"; //记录 XML 文件路径
string strCon = "Data Source=MRWXK\WANGXIAOKE;Database=db_CSharp;Uid=sa;Pwd=";
//定义数据库连接字符串
linqtosqlDataContext linq; //创建 Linq 连接对象
```

(5) 自定义一个 `getXmlInfo` 方法，用来将 XML 文件内容绑定到 `DataGridView` 控件上，代码如下。

```
private void getXmlInfo()
{
}
```

```

DataSet myds = new DataSet();
myds.ReadXml(strPath);
//在 DataGridView 中显示 XML 文件中的信息
dataGridView1.DataSource = myds.Tables[0];
}

```

(6) 窗体加载时，首先判断指定的 XML 文件是否存在，然后调用自定义方法 getXmlInfo 显示 XML 文件中的内容，代码如下。

```

private void Form1_Load(object sender, EventArgs e)
{
    if (File.Exists(strPath)) //判断 XML 文件是否存在
    {
        getXmlInfo(); //窗体加载时加载 XML 文件
    }
}

```

(7) 单击“更新”按钮，遍历 DataGridView 控件中显示的 XML 文件数据，并调用 InsertOnSubmit 方法和 SubmitChanges 方法将遍历到的数据更新到指定的数据表中。“更新”按钮的 Click 事件代码如下。

```

private void btn_Edit_Click(object sender, EventArgs e)
{
    for (int i = 0; i < dataGridView1.Rows.Count - 1; i++) //遍历所有行
    {
        linq = new linqtosqlDataContext(strCon); //创建 linq 连接对象
        tb_XML xml = new tb_XML(); //创建 tb_XML 对象
        xml.ID = dataGridView1.Rows[i].Cells[3].Value.ToString(); //为 ID 赋值
        xml.Name = dataGridView1.Rows[i].Cells[0].Value.ToString(); //为 Name 赋值
        xml.Sex = dataGridView1.Rows[i].Cells[1].Value.ToString(); //为 Sex 赋值
        xml.Salary = Convert.ToInt32(dataGridView1.Rows[i].Cells[2].Value); //为 Salary 赋值
        linq.tb_XML.InsertOnSubmit(xml); //提交数据
        linq.SubmitChanges(); //执行对数据库的修改
        linq.Dispose(); //释放 linq 对象
    }
    MessageBox.Show("成功将 XML 中的数据更新到了数据库中！"); //弹出提示
}

```

**DIY：**制作一个程序，用来使用 LINQ 技术实现将 SQL Server 数据库中的数据转换为 XML 格式的功能。**提示：**通过向 LINQ to SQL 的 dbml 文件中添加 tb\_Employee 表实现。(20 分)(实例位置：光盘\mr\14\qjyy\05\_diy)

情景应用 DIY 栏目分数统计：

DIY 题目	1	2	3	4	5	总分数	
分数							

## 14.6 自我测试

### 一、选择题（每题 10 分，5 道题）

1. LINQ 查询表达式中必须包含（ ）子句。

- A. from                  B. where                  C. orderby                  D. groupby  
2. 在 LINQ 查询表达式中，（ ）子句用于指定筛选元素的逻辑条件。  
A. from                  B. where                  C. orderby                  D. select  
3. LINQ 查询表达式必须以（ ）子句结束。  
A. where                  B. orderby                  C. groupby                  D. select 或 group  
4. orderby 子句降序排序时使用（ ）关键字。  
A. ascending              B. descending              C. ASC                  D. DESC  
5. 使用 LINQ 操作 SQL Server 时，调用（ ）方法将最终操作结果更新到数据库中。  
A. InsertOnSubmit          B. DeleteOnSubmit          C. DeleteAllOnSubmit          D. SubmitChanges

## 二、填空题（每题 10 分， 5 道题）

1. 判断序列中所有元素是否都满足指定条件时使用（`Where`）操作符。
  2. 获取序列中的第一个元素时使用（`First`）操作符。
  3. 查询非泛型集合（如 `ArrayList`）需要使用（`Cast`）操作符将其转换为 `IEnumerable<T>` 类型。
  4. 筛选指定类型的元素时应使用（`OfType`）操作符。
  5. 查找字符串中属于标点类型的字符时应使用（`Where`）方法。

测试分数统计:

类别	第 1 题	第 2 题	第 3 题	第 4 题	第 5 题
选择题分数					
填空题分数					
总分数					

## 14.7 行动指南

开始日期： 年 月 日

结束日期： 年 月 日

序号	内 容	行 动 指 南	
1	照猫画虎栏目	分数>75 分	优秀, 基本功掌握得不错, 加油!
		75 分>分数>50 分	及格, 知识掌握得不牢, 重新做一遍照猫画虎。
	分数( )	分数<50 分	请使用光盘中提供的“实战能力测试系统”进行提高训练。
	情景应用栏目	分数>75 分	优秀, 综合应用能力很强。
		75 分>分数>50 分	及格, 综合应用能力需提高, 再练一遍情景应用。
	分数( )	分数<50 分	请使用光盘中提供的“实战能力测试系统”进行提高训练。
	自我测试栏目	分数>75 分	优秀, 有成为编程高手的潜质。
		分数<75 分	请使用光盘中提供的“实战能力测试系统”进行提高训练。
	综合评价	反复训练后, 上各项分数都在 75 分以上, 方可进入下一堂课学习。	
2	编程能力培养: 本栏目提供了一些实践题目, 对于培养编程能力很有效, 要做好。	1. 开发一个程序, 主要通过使用 LINQ 技术统计每个单词在文章中出现的次数。 2. 开发一个程序, 主要使用 LINQ 技术统计每种商品的最高销售价。 3. 开发一个程序, 使用 LINQ 技术统计员工的工资总额。	

续表

序号	内 容	行 动 指 南
3	编程习惯培养：本堂课培养读者在学习开发中记笔记的习惯，把开发中遇到的问题、总结的经验记录下来。	
4	创新能力培养：看看身边有没有可以用编程解决的问题，记录到右边的表格中。根据学习进度，尝试编写解决这些问题的小程序。	

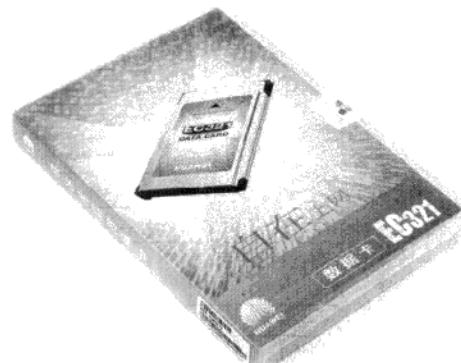
## 14.8 成功可以复制——中国通信设备行业的领跑者任正非

任正非，祖籍浙江浦江县，出生在一个贫苦山区的小村庄，任正非的爷爷是一个做火腿的大师傅，任正非父亲的兄弟姊妹都没有读过书。由于爷爷的“良心发现”，也由于爸爸的执著要求，爸爸才读了书。任母虽然只有高中文化程度，但是受丈夫影响，通过自修，当上了中学教员。虽然是农村，却是一个知识分子家庭。家庭背景是任正非一生第一个决定性因素。中国的知识分子对知识的重视和追求，可谓“贫贱不能移”。在 3 年自然灾害时期，父母仍然坚持从牙缝里挤出粮食来让孩子读书。任正非凭借其才智和能力以及对知识的追求，进入了一个科技密集型行业。在文化大革命时期，任正非不仅没有放弃学业，而且还自学了高等数学、逻辑、哲学和 3 门外语。任正非的知识渊博、见解独到，在他的讲话中体现为旁征博引、一针见血。后来任正非参军了，作为通讯兵的他被抽调到一个飞机制造厂，参与一项代号为 011 的军事通讯系统工程。当时中央军委提出要重视高科技的作用。任正非上进好学，有多项发明创造，两次填补了国家空白。因技术方面的多次突破，被选为军方代表，到北京参加全国科学大会，时年 33 岁。

任正非从部队转业后首先选择了一家电子公司，而在创办华为时选择了高科技含量的电信行业。1992 年，任正非孤注一掷投入 C&C08 机的研发，虽然是形势所逼，也可以看出他对技术的重视。当时身处房地产热和股票热的核心地带，任正非不仅不为所动，而且对于股票和泡沫深恶痛绝，其实他在内心更多的是对知识、技术和真才实学的尊重。

2006 年美国《新闻周刊》说，尽管创立者任正非一直保持低调，但华为已经与电讯业的几个国际巨头北方电讯、朗讯科技、阿尔卡特、思科系统站在同一水平线展开竞争，而且华为常常可以从他们之间赢得更多网络运营业务。

如今，华为公司在全球设立了包括印度、美国、瑞典、俄罗斯以及中国的北京、上海、南京等多个研究所，100000 名员工中的 48% 从事研发工作，截至 2008 年 6 月，华为已累计申请专利超过 29666 件，已连续数年成为中国申请专利最多的单位。同时华为还在全球建立了 100 多个分支机构，营销及服务网络遍及全球，为客户



提供快速、优质的服务。

 经典语录 -----

世界上一切资源都可能枯竭，只有一种资源可以生生不息，那就是文化。

 深度评价 -----

对于人生而言，贫穷是老师，他可以教会人如何去生存。贫穷有压力，可以使你的脊梁比别人更硬，坦然吃苦、不屈不挠。在我国许许多多的民营企业家身上都可以找到贫穷的影子，而他们的吃苦耐劳、坚忍不拔的品质，往往就是创业的精神所在。因此有人将贫穷当作是成功者的“财富”。



# 第 15 堂课

## 文件及 I/O

( 视频讲解：171分钟)

在软件开发过程中经常需要对文件及文件夹进行操作，如读写、移动、复制、删除文件及创建、移动、删除、遍历文件夹等，C#中与文件、文件夹及文件读写有关的类都位于 System.IO 命名空间下。本堂课将详细介绍如何在 C#中对文件、文件夹进行操作，以及如何对文件进行 I/O 数据流读写。

学习摘要：

- ▶ 了解文件及 I/O 操作的常用类
- ▶ 掌握 File 类和 FileInfo 类的使用
- ▶ 掌握 Directory 类和 DirectoryInfo 类的使用
- ▶ 掌握文件的基本操作
- ▶ 掌握文件夹的基本操作
- ▶ 掌握文件 I/O 流类的使用
- ▶ 掌握如何对文本文件进行写入与读取
- ▶ 掌握如何对二进制文件进行写入与读取

## 15.1 文件操作基础

对文件进行操作时需要用到 System.IO 命名空间下提供的各种类，该命名空间包含允许在数据流和文件上进行同步和异步读取及写入的类型。这里需要注意文件和 I/O 流的差异，文件是一些具有永久存储及特定顺序的字节组成的一个有序的、具有名称的集合，因此，关于文件，人们常会想到目录路径、磁盘存储、文件和目录名等方面内容。相反，I/O 流提供一种向后备存储写入字节和从后备存储读取字节的方式。后备存储可以为多种存储媒介之一，正如除磁盘外存在多种后备存储一样，除文件流之外也存在多种流，如网络流、内存流等。

System.IO 命名空间中的常用类及说明如表 15.1 所示。

表 15.1 System.IO 命名空间中的类及说明

类	说 明
BinaryReader	用特定的编码将基元数据类型读作二进制值
BinaryWriter	以二进制形式将基元类型写入流，并支持用特定的编码写入字符串
Directory	公开用于创建、移动和枚举通过目录和子目录的静态方法。无法继承此类
DirectoryInfo	公开用于创建、移动和枚举目录和子目录的实例方法。无法继承此类
File	提供用于创建、复制、删除、移动和打开文件的静态方法，并协助创建 FileStream 对象
FileInfo	提供创建、复制、删除、移动和打开文件的实例方法，并且帮助创建 FileStream 对象
FileStream	公开以文件为主的 Stream，既支持同步读写操作，也支持异步读写操作
FileSystemInfo	为 FileInfo 和 DirectoryInfo 对象提供基类
StreamReader	实现一个 TextReader，使其以一种特定的编码从字节流中读取字符
StreamWriter	实现一个 TextWriter，使其以一种特定的编码向流中写入字符

### 15.1.1 File 类和 FileInfo 类介绍

File 类和 FileInfo 类都用来对文件进行操作，本节将对这两个类的用法及区别进行详细介绍。

#### 1. File 类

File 类支持对文件的基本操作，它包括用于创建、复制、删除、移动和打开文件的静态方法，并协助创建 FileStream 对象。File 类中共包含 40 多个方法，这里只列出其常用的几种方法，如表 15.2 所示。

表 15.2 File 类的常用方法及说明

方 法	说 明
Copy	将现有文件复制到新文件
Create	在指定路径中创建文件
Delete	删除指定的文件。如果指定的文件不存在，则不引发异常
Exists	确定指定的文件是否存在
Move	将指定文件移到新位置，并提供指定新文件名的选项
Open	打开指定路径上的 FileStream
GetCreationTime	返回指定文件或目录的创建日期和时间
OpenRead	打开现有文件以进行读取
OpenText	打开现有 UTF-8 编码文本文件以进行读取
OpenWrite	打开现有文件以进行写入

**说明:** ① 由于 File 类中的所有方法都是静态的, 所以如果只想执行一个操作, 那么使用 File 类中方法的效率比使用相应的 FileInfo 类中的方法可能更高。

② File 类中的方法都是静态方法, 在使用时需要对所有方法都执行安全检查, 因此如果打算多次重用某个对象, 可考虑改用 FileInfo 类中的相应方法, 因为并不总是需要安全检查。

**例 15.01** 下面演示如何使用 File 类中的方法, 程序开发步骤如下。(实例位置: 光盘\mr\15\sl\15.01)

(1) 新建一个 Windows 应用程序, 命名为 UseFile, 默认窗体为 Form1.cs。

(2) 在 Form1 窗体中添加一个 TextBox 控件和一个 Button 控件。其中, TextBox 控件用来输入要创建的文件路径及名称; Button 控件用来执行创建文件操作。

(3) 程序主要代码如下。

```
private void button1_Click(object sender, EventArgs e)
{
    if (textBox1.Text == string.Empty) //判断输入的文件名是否为空
    {
        MessageBox.Show("文件名不能为空!");
    }
    else
    {
        if (File.Exists(textBox1.Text)) //使用 File 类的 Exists 方法判断要创建的文件是否存在
        {
            MessageBox.Show("该文件已经存在!");
        }
        else
        {
            File.Create(textBox1.Text); //使用 File 类的 Create 方法创建文件
        }
    }
}
```

程序运行结果如图 15.1 所示。

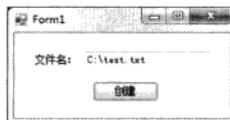


图 15.1 File 类的使用

**注意:** 使用与文件、文件夹及流相关的类时, 首先需要添加 System.IO 命名空间。

## 2. FileInfo 类

在 FileInfo 类和 File 类之间有许多方法调用都是相同的, 但是 FileInfo 类没有静态方法, 该类中的方法仅可用于创建的对象。File 类是静态类, 它的调用需要字符串参数为每一个方法调用规定文件位置, 因此如果要在对象上进行单一方法调用, 则可以使用静态 File 类, 在这种情况下静态调用速度要快一些, 因为.NET 框架不必执行创建新对象并调用其方法的过程。如果要在文件上执行几种操作, 则创建 FileInfo 对象使用其方法就更好一些, 这样会提高效率, 因为对象将在文件系统上引用正确的文件, 而静态类就必须每次都寻找文件。

FileInfo 类的常用属性及说明如表 15.3 所示。

表 15.3 FileInfo 类的常用属性及说明

属性	说 明
Directory	获取父目录的实例
Exists	获取指示文件是否存在值
FullName	获取目录或文件的完整目录
Length	获取当前文件的大小
Name	获取文件名

**例 15.02** 下面演示如何使用 FileInfo 类中的属性及方法，程序开发步骤如下。（实例位置：光盘\mr\15\sl\15.02）

(1) 新建一个 Windows 应用程序，命名为 UseFileInfo， 默认窗体为 Form1.cs。

(2) 在 Form1 窗体中添加一个 TextBox 控件和一个 Button 控件。其中， TextBox 控件用来输入要创建的文件路径及名称； Button 控件用来执行创建文件操作。

(3) 程序主要代码如下。

```
private void button1_Click(object sender, EventArgs e)
{
    if (textBox1.Text == string.Empty)           //判断输入的文件名称是否为空
    {
        MessageBox.Show("文件名称不能为空！");
    }
    else
    {
        FileInfo finfo = new FileInfo(textBox1.Text); //创建 FileInfo 对象
        if (finfo.Exists)                         //使用 FileInfo 对象的 Exists 属性判断要创建的文件是否存在
        {
            MessageBox.Show("该文件已经存在");
        }
        else
        {
            finfo.Create();                      //使用 FileInfo 对象的 Create 方法创建文件
        }
    }
}
```

程序运行结果如图 15.2 所示。

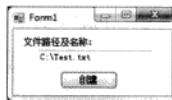


图 15.2 FileInfo 类的使用

### 15.1.2 Directory 类和 DirectoryInfo 类介绍

使用 Directory 类和 DirectoryInfo 类可以方便地对文件夹（即目录）进行操作，本节将对这两个类的用法及区别进行详细介绍。

#### 1. Directory 类

Directory 类公开了用于创建、移动、枚举、删除目录和子目录的静态方法，这里介绍一些该类中的常

用方法，如表 15.4 所示。

表 15.4 Directory 类的常用方法及说明

方 法	说 明
CreateDirectory	创建指定路径中的所有目录
Delete	删除指定的目录
Exists	确定给定路径是否引用磁盘上的现有目录
GetFiles	返回指定目录中的文件的名称
Move	将文件或目录及其内容移到新位置
SetCurrentDirectory	将应用程序的当前工作目录设置为指定的目录
SetLastAccessTime	设置上次访问指定文件或目录的日期和时间
SetLastWriteTime	设置上次写入目录的日期和时间

**例 15.03** 下面演示如何使用 Directory 类中的方法，程序开发步骤如下。（实例位置：光盘\mr\15\s\15.03）

- (1) 新建一个 Windows 应用程序，命名为 UseDirectory，默认窗体为 Form1.cs。
- (2) 在 Form1 窗体中添加一个 TextBox 控件和一个 Button 控件。其中，TextBox 控件用来输入要创建的文件夹路径及名称；Button 控件用来执行创建文件夹操作。
- (3) 程序主要代码如下。

```
private void button1_Click(object sender, EventArgs e)
{
    if (textBox1.Text == string.Empty)          //判断输入的文件夹名称是否为空
    {
        MessageBox.Show("文件夹名称不能为空！");
    }
    else
    {
        if (Directory.Exists(textBox1.Text))      //使用 Directory 类的 Exists 方法判断要创建的文件夹是否存在
        {
            MessageBox.Show("该文件夹已经存在");
        }
        else
        {
            //使用 DirectoryInfo 类的 CreateDirectory 方法创建文件夹
            Directory.CreateDirectory(textBox1.Text);
        }
    }
}
```

程序运行结果如图 15.3 所示。

## 2. DirectoryInfo 类

在 DirectoryInfo 类和 Directory 类之间有许多方法调用都是相同的，但是 DirectoryInfo 类没有静态方法，该类中的方法仅可用于创建的对象。Directory 类是静态类，所以它的调用需要字符串参数为每一个方法调用规定文件夹路径，因此如果要在对象上进行单一方法调用，则可以使用静态 Directory 类，在这种情况下静态调用速度要快一些，因为.NET 框架不必执行创建新对象并调用其方法的过程。

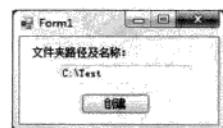


图 15.3 Directory 类的使用

程。如果要在文件夹上执行几种操作，则创建 DirectoryInfo 对象使用其方法就更好一些，这样会提高效率，因为对象将在文件夹系统上引用正确的文件夹，而静态类就必须每次都寻找文件夹。

DirectoryInfo 类的常用属性及说明如表 15.5 所示。

表 15.5 DirectoryInfo 类的常用属性及说明

属性	说 明
Exists	获取指示目录是否存在值
Extension	获取表示文件扩展名部分的字符串
FullName	获取目录或文件的完整目录
Name	获取 DirectoryInfo 实例的名称
Parent	获取指定子目录的父目录
Root	获取路径的根部分

**例 15.04** 下面演示如何使用 DirectoryInfo 类中的属性及方法，程序开发步骤如下。（实例位置：光盘\mr\15\sl\15.04）

(1) 新建一个 Windows 应用程序，并将其命名为 Use DirectoryInfo，默认窗体为 Form1.cs。

(2) 在 Form1 窗体中添加一个 TextBox 控件和一个 Button 控件。其中，TextBox 控件用来输入要创建的文件夹路径及名称；Button 控件用来执行创建文件夹操作。

(3) 程序主要代码如下。

```
private void button1_Click(object sender, EventArgs e)
{
    if (textBox1.Text == string.Empty) //判断输入的文件夹名称是否为空
    {
        MessageBox.Show("文件夹名称不能为空！");
    }
    else
    {
        //创建 DirectoryInfo 对象
        DirectoryInfo dinfo = new DirectoryInfo(textBox1.Text);
        if (dinfo.Exists) //使用 DirectoryInfo 对象的 Exists 属性判断要创建的文件夹是否存在
        {
            MessageBox.Show("该文件夹已经存在");
        }
        else
        {
            dinfo.Create(); //使用 DirectoryInfo 对象的 Create 方法创建文件夹
        }
    }
}
```

程序运行结果如图 15.4 所示。

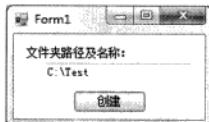


图 15.4 DirectoryInfo 类的使用

## 15.2 文件基本操作

对文件的基本操作大体可以分为：判断文件是否存在、创建文件、复制或移动文件、删除文件以及获取文件基本信息等，本节将对这些基本操作进行详细讲解。

### 15.2.1 判断文件是否存在

判断文件是否存在可以使用 `File` 类的 `Exists` 方法或者 `FileInfo` 类的 `Exists` 属性来实现，下面分别对它们进行介绍。

#### 1. `File` 类的 `Exists` 方法

确定指定的文件是否存在，语法如下。

语法：`public static bool Exists(string path)`

说明：参数 `path` 表示要检查的文件。如果调用方具有要求的权限并且 `path` 包含现有文件的名称，则该方法返回 `true`；否则返回 `false`。如果 `path` 为空引用或零长度字符串，则此方法返回 `false`。如果调用方不具有读取指定文件所需的足够权限，则不引发异常并且该方法返回 `false`，这与 `path` 是否存在无关。

**例 15.05** 下面代码使用 `File` 类的 `Exists` 方法判断 C 盘根目录下是否存在 `Test.txt` 文件。

```
File.Exists("C:\Test.txt");
```

#### 2. `FileInfo` 类的 `Exists` 属性

获取指示文件是否存在的值，语法如下。

语法：`public override bool Exists { get; }`

说明：如果该文件存在，则该属性值为 `true`；如果该文件不存在或该文件是目录，则该属性值为 `false`。

**例 15.06** 下面代码首先创建一个 `FileInfo` 对象，然后使用该对象调用 `FileInfo` 类中的 `Exists` 属性判断 C 盘根目录下是否存在 `Test.txt` 文件。

```
FileInfo finfo = new FileInfo("C:\Test.txt");
if (finfo.Exists)
{
}
```

### 15.2.2 创建文件

创建文件可以使用 `File` 类的 `Create` 方法或者 `FileInfo` 类的 `Create` 方法来实现，下面分别对它们进行介绍。

#### 1. `File` 类的 `Create` 方法

该方法为可重载方法，它有以下 4 种重载形式。

```
public static FileStream Create(string path)
public static FileStream Create(string path,int bufferSize)
public static FileStream Create(string path,int bufferSize,FileOptions options)
public static FileStream Create(string path,int bufferSize,FileOptions options,FileSecurity fileSecurity)
```

参数说明如表 15.6 所示。

表 15.6 File 类的 Create 方法参数说明

参    数	说    明
path	文件名
bufferSize	用于读取和写入文件的已放入缓冲区的字节数
options	FileOptions 值之一，它描述如何创建或改写该文件
fileSecurity	FileSecurity 值之一，它确定文件的访问控制和审核安全性

**例 15.07** 下面代码调用 File 类的 Create 方法在 C 盘根目录下创建一个 Test.txt 文本文件。

```
File.Create("C:\Test.txt");
```

## 2. FileInfo 类的 Create 方法

语法： public FileStream Create()

说明： 该方法返回文件流，默认情况下，该方法将向所有用户授予对新文件的完全读写访问权限。

**例 15.08** 下面代码首先创建了一个 FileInfo 对象，然后使用该对象调用 FileInfo 类的 Create 方法在 C 盘根目录下创建一个 Test.txt 文本文件。

```
FileInfo finfo = new FileInfo("C:\Test.txt");
finfo.Create();
```

## 15.2.3 复制文件

复制文件可以使用 File 类的 Copy 方法或者 FileInfo 类的 CopyTo 方法来实现，下面分别对它们进行介绍。

### 1. File 类的 Copy 方法

该方法为可重载方法，它有以下两种重载形式。

```
public static void Copy(string sourceFileName,string destFileName)
public static void Copy(string sourceFileName,string destFileName,bool overwrite)
```

说明： 参数 sourceFileName 表示要复制的文件；参数 destFileName 表示目标文件的名称，不能是目录，如果是第一种重载形式，则该参数不能是现有文件；参数 overwrite 表示如果可以改写目标文件，则为 true，否则为 false。

**例 15.09** 下面代码调用 File 类的 Copy 方法将 C 盘根目录下的 Test.txt 文本文件复制到 D 盘根目录下。

```
File.Copy("C:\Test.txt","D:\Test.txt");
```

### 2. FileInfo 类的 CopyTo 方法

该方法为可重载方法，它有以下两种重载形式。

```
public FileInfo CopyTo(string destFileName)
public FileInfo CopyTo(string destFileName,bool overwrite)
```

说明： 参数 destFileName 表示要复制到的新文件的名称。如果参数 overwrite 为 true，则允许改写现有文件；否则为 false。第一种重载形式的返回值为带有完全限定路径的新文件；第二种重载形式的返回值为新文件，或者如果 overwrite 为 true，则为现有文件的改写，如果文件存在，且 overwrite 为 false，则会发生 IOException。

**例 15.10** 下面代码首先创建了一个 FileInfo 对象，然后使用该对象调用 FileInfo 类的 CopyTo 方法将 C

盘根目录下的 Test.txt 文本文件复制到 D 盘根目录下，如果 D 盘根目录下已经存在 Test.txt 文本文件，则将其替换。

```
FileInfo finfo = new FileInfo("C:\\Test.txt");
finfo. CopyTo("D:\\Test.txt",true);
```

## 15.2.4 移动文件

移动文件可以使用 File 类的 Move 方法或者 FileInfo 类的 MoveTo 方法来实现，下面分别对它们进行介绍。

### 1. File 类的 Move 方法

将指定文件移到新位置，并提供指定新文件名的选项。

语法：public static void Move(string sourceFileName, string destFileName)

说明：参数 sourceFileName 表示要移动的文件的名称；参数 destFileName 表示文件的新路径。

**例 15.11** 下面代码调用 File 类的 Move 方法将 C 盘根目录下的 Test.txt 文本文件移动到 D 盘根目录下。  
File.Move("C:\\Test.txt","D:\\Test.txt");

### 2. FileInfo 类的 MoveTo 方法

将指定文件移到新位置，并提供指定新文件名的选项。

语法：public void MoveTo(string destFileName)

说明：参数 destFileName 表示要将文件移动到的路径，可以指定另一个文件名。

**例 15.12** 下面代码首先创建了一个 FileInfo 对象，然后使用该对象调用 FileInfo 类的 MoveTo 方法将 C 盘根目录下的 Test.txt 文本文件移动到 D 盘根目录下。

```
FileInfo finfo = new FileInfo("C:\\Test.txt");
finfo. MoveTo("D:\\Test.txt");
```

## 15.2.5 删 除文件

删除文件可以使用 File 类的 Delete 方法或者 FileInfo 类的 Delete 方法来实现，下面分别对它们进行介绍。

### 1. File 类的 Delete 方法

删除指定的文件，语法如下。

语法：public static void Delete(string path)

说明：参数 path 表示要删除的文件的名称。

**例 15.13** 下面代码调用 File 类的 Delete 方法删除 C 盘根目录下的 Test.txt 文本文件。

```
File.Delete("C:\\Test.txt");
```

### 2. FileInfo 类的 Delete 方法

永久删除文件，语法如下。

语法：public override void Delete()

说明：该方法无返回值。

**例 15.14** 下面代码首先创建了一个 FileInfo 对象，然后使用该对象调用 FileInfo 类的 Delete 方法删除 C 盘根目录下的 Test.txt 文本文件。

```
FileInfo finfo = new FileInfo("C:\\Test.txt");
finfo. Delete();
```

## 15.3 文件夹基本操作

对文件夹的基本操作大体可以分为：判断文件夹是否存在、创建文件夹、移动文件夹、删除文件夹以及遍历文件夹中的文件，本节将对这些基本操作进行详细讲解。

### 15.3.1 判断文件夹是否存在

判断文件夹是否存在可以使用 `Directory` 类的 `Exists` 方法或者  `DirectoryInfo` 类的 `Exists` 属性来实现，下面分别对它们进行介绍。

#### 1. `Directory` 类的 `Exists` 方法

确定给定路径是否引用磁盘上的现有目录，语法如下。

语法：`public static bool Exists(string path)`

说明：参数 `path` 表示要测试的路径。如果 `path` 引用现有目录，则该方法返回 `true`；否则返回 `false`。

**例 15.15** 下面代码使用 `Directory` 类的 `Exists` 方法判断 C 盘根目录下是否存在 Test 文件夹。

```
Directory.Exists("C:\\Test");
```

#### 2. `DirectoryInfo` 类的 `Exists` 属性

获取指示目录是否存在的值，语法如下。

语法：`public override bool Exists { get; }`

说明：如果目录存在，则属性值为 `true`；否则为 `false`。

**例 15.16** 下面代码首先创建一个  `DirectoryInfo` 对象，然后使用该对象调用  `DirectoryInfo` 类中的 `Exists` 属性判断 C 盘根目录下是否存在 Test 文件夹。

```
DirectoryInfo dinfo = new DirectoryInfo ("C:\\Test");
if (dinfo.Exists)
{
}
```

### 15.3.2 创建文件夹

创建文件夹可以使用 `Directory` 类的 `CreateDirectory` 方法或者  `DirectoryInfo` 类的 `Create` 方法来实现，下面分别对它们进行介绍。

#### 1. `Directory` 类的 `CreateDirectory` 方法

该方法为可重载方法，它有以下两种重载形式。

`public static DirectoryInfo CreateDirectory(string path)`

`public static DirectoryInfo CreateDirectory(string path,DirectorySecurity directorySecurity)`

说明：参数 `path` 表示要创建的目录路径，参数 `directorySecurity` 表示要应用于此目录的访问控制。第一种重载形式的返回值为由 `path` 指定的  `DirectoryInfo`；第二种重载形式的返回值为新创建的目录的  `DirectoryInfo` 对象。

**例 15.17** 下面代码调用 `Directory` 类的 `CreateDirectory` 方法在 C 盘根目录下创建一个 Test 文件夹。

```
Directory.CreateDirectory ("C:\\Test ");
```

## 2. DirectoryInfo 类的 Create 方法

该方法为可重载方法，它有以下两种重载形式。

```
public void Create()
```

```
public void Create(DirectorySecurity directorySecurity)
```

说明：参数 directorySecurity 表示要应用于此目录的访问控制。

**例 15.18** 下面代码首先创建了一个 DirectoryInfo 对象，然后使用该对象调用 DirectoryInfo 类的 Create 方法在 C 盘根目录下创建一个 Test 文件夹。

```
DirectoryInfo dinfo = new DirectoryInfo ("C:\\Test ");
dinfo.Create();
```

### 15.3.3 移动文件夹

移动文件夹可以使用 Directory 类的 Move 方法或者 DirectoryInfo 类的 MoveTo 方法来实现，下面分别对它们进行介绍。

#### 1. Directory 类的 Move 方法

将文件夹或目录及其内容移到新位置，语法如下。

```
语法： public static void Move(string sourceDirName,string destDirName)
```

说明：参数 sourceDirName 表示要移动的文件夹或目录的路径；参数 destDirName 表示指向 sourceDirName 的新位置的路径。

**例 15.19** 下面代码调用 Directory 类的 Move 方法将 C 盘根目录下的 Test 文件夹移动到 C 盘根目录下的“新建文件夹”文件夹中。

```
Directory.Move("C:\\Test ","C:\\新建文件夹\\Test");
```

 **注意：**使用 Move 方法移动文件夹时需要统一磁盘根目录，例如，C 盘下的文件夹只能移动到 C 盘中的某个文件夹下，同样，使用 MoveTo 方法移动文件夹时也是如此，下面不再强调。

#### 2. DirectoryInfo 类的 MoveTo 方法

将 DirectoryInfo 对象及其内容移动到新路径，语法如下。

```
语法： public void MoveTo(string destDirName)
```

说明：参数 destDirName 表示要将此目录移动到的目标位置的名称和路径。目标不能是另一个具有相同名称的磁盘卷或目录，它可以是要将此目录作为子目录添加到其中的一个现有目录。

**例 15.20** 下面代码首先创建了一个 DirectoryInfo 对象，然后使用该对象调用 DirectoryInfo 类的 MoveTo 方法将 C 盘根目录下的 Test 文件夹移动到 C 盘根目录下的“新建文件夹”文件夹中。

```
DirectoryInfo dinfo = new DirectoryInfo ("C:\\Test ");
dinfo.MoveTo("C:\\新建文件夹\\Test");
```

### 15.3.4 删 除文件夹

删除文件夹可以使用 Directory 类的 Delete 方法或者 DirectoryInfo 类的 Delete 方法来实现，下面分别对它们进行介绍。

### 1. Directory 类的 Delete 方法

该方法为可重载方法，它有以下两种重载形式。

```
public static void Delete(string path)
public static void Delete(string path, bool recursive)
```

说明：参数 path 表示要移除的空目录/目录的名称。若要移除 path 中的目录、子目录和文件，则参数 recursive 为 true；否则为 false。

**例 15.21** 下面代码调用 Directory 类的 Delete 方法删除 C 盘根目录下的 Test 文件夹。

```
Directory.Delete("C:\Test");
```

### 2. DirectoryInfo 类的 Delete 方法

永久删除文件，它有以下两种重载形式。

```
public override void Delete()
public void Delete(bool recursive)
```

说明：参数 recursive 的值若为 true，则删除此目录、其子目录及所有文件；否则为 false。

 说明：第一种重载形式，如果 DirectoryInfo 为空，则删除它；第二种重载形式，删除 DirectoryInfo 对象，并指定是否要删除子目录和文件。

**例 15.22** 下面代码首先创建了一个 DirectoryInfo 对象，然后使用该对象调用 DirectoryInfo 类的 Delete 方法删除 C 盘根目录下的 Test 文件夹。

```
DirectoryInfo dinfo = new DirectoryInfo ("C:\Test");
dinfo.Delete();
```

## 15.4 I/O 输入输出

I/O 数据流提供了一种向后备存储写入字节和从后备存储读取字节的方式，它是在.NET Framework 中执行读写文件操作时的一种非常重要的介质，下面对 I/O 数据流进行详细讲解。

### 15.4.1 流概述

.NET Framework 使用流来支持读取和写入文件，开发人员可以将流视为一组连续的一维数据，包含开头和结尾，并且其中的游标指示了流中的当前位置。

#### 1. 流操作

流中包含的数据可能来自内存、文件或 TCP/IP 套接字。流包含以下几种可应用于自身的操作。

- 读取。将数据从流传输到数据结构（如字符串或字节数组）中。
- 写入。将数据从数据源传输到流中。
- 查找。查询和修改在流中的位置。

#### 2. 流的类型

在.NET Framework 中，流由 Stream 类来表示，该类构成了所有其他流的抽象类。不能直接创建 Stream 类的实例，但是必须使用它实现某个 I/O 流类。

C#中有许多类型的流，但在处理文件输入/输出（I/O）时，最重要的类型为 FileStream 类，它提供读取

和写入文件的方式。可在处理文件 I/O 时使用的其他流主要包括 `BufferedStream`、`CryptoStream`、`MemoryStream` 和 `NetworkStream` 等。

### 15.4.2 文件 I/O 流介绍

C#中，文件 I/O 流使用 `FileStream` 类实现，该类公开以文件为主的 Stream，它表示在磁盘或网络路径上指向文件的流。一个 `FileStream` 类的实例实际上代表一个磁盘文件，它通过 `Seek` 方法进行对文件的随机访问，也同时包含了流的标准输入、标准输出和标准错误等。`FileStream` 默认对文件的打开方式是同步的，但它同样很好地支持异步操作。

#### 1. `FileStream` 类的常用属性

`FileStream` 类的常用属性及说明如表 15.7 所示。

表 15.7 `FileStream` 类的常用属性及说明

属性	说 明
<code>Length</code>	获取用字节表示的流长度
<code>Name</code>	获取传递给构造函数的 <code>FileStream</code> 的名称
<code>Position</code>	获取或设置此流的当前位置
<code>ReadTimeout</code>	获取或设置一个值，该值确定流在超时前尝试读取多长时间
<code>WriteTimeout</code>	获取或设置一个值，该值确定流在超时前尝试写入多长时间

#### 2. `FileStream` 类的常用方法

`FileStream` 类的常用方法及说明如表 15.8 所示。

表 15.8 `FileStream` 类的常用方法及说明

方 法	说 明
<code>Close</code>	关闭当前流并释放与之关联的所有资源
<code>Read</code>	从流中读取字节块并将该数据写入给定缓冲区中
<code>ReadByte</code>	从文件中读取一个字节，并将读取位置提升一个字节
<code>Seek</code>	将该流的当前位置设置为给定值
<code>SetLength</code>	将该流的长度设置为给定值
<code>Write</code>	使用从缓冲区读取的数据将字节块写入该流
<code>WriteByte</code>	将一个字节写入文件流的当前位置

#### 3. 使用 `FileStream` 类操作文件

若要使用 `FileStream` 类操作文件就要先创建一个 `FileStream` 对象，`FileStream` 类的构造函数具有许多不同的重载形式，其中包括了一个最重要的参数，即  `FileMode` 枚举。

`FileMode` 枚举规定了如何打开或创建文件，其包括的枚举成员及说明如表 15.9 所示。

表 15.9  `FileMode` 类的枚举成员及说明

枚 举 成 员	说 明
<code>Append</code>	打开现有文件并查找到文件尾或创建新文件。 <code> FileMode.Append</code> 只能与 <code> FileAccess.Write</code> 一起使用。任何读尝试都将失败并引发 <code> ArgumentException</code>

续表

枚举成员	说    明
Create	指定操作系统应创建新文件。如果文件已存在，则它将被改写。这要求 FileIOPermissionAccess.Write。System.IO.FileMode.Create 等效于这样的请求：如果文件不存在，则使用 CreateNew；否则使用 Truncate
CreateNew	指定操作系统应创建新文件。此操作需要 FileIOPermissionAccess.Write。如果文件已存在，则将引发 IOException
Open	指定操作系统应打开现有文件。打开文件的能力取决于 FileAccess 所指定的值。如果该文件不存在，则引发 System.IO.FileNotFoundException
OpenOrCreate	指定操作系统应打开文件（如果文件存在）；否则，应创建新文件。如果用 FileAccess.Read 打开文件，则需要 FileIOPermissionAccess.Read。如果文件访问为 FileAccess.Write 或 FileAccess.ReadWrite，则需要 FileIOPermissionAccess.Write。如果文件访问为 FileAccess.Append，则需要 FileIOPermissionAccess.Append
Truncate	指定操作系统应打开现有文件。文件一旦打开，就将被截断为零字节大小。此操作需要 FileIOPermissionAccess.Write。试图从使用 Truncate 打开的文件中进行读取将导致异常

**例 15.23** 下面代码通过使用 FileStream 对象打开 Test.txt 文本文件并对其进行读写访问。

```
FileStream aFile = new FileStream("Test.txt", FileMode.OpenOrCreate, FileAccess.ReadWrite)
```

 注意：文件要放在程序运行目录下，否则就需要给构造函数传递绝对路径。

### 15.4.3 使用 I/O 流操作文本文件

使用 I/O 流操作文本文件时主要用到 StreamWriter 类和 StreamReader 类，下面对这两个类进行详细讲解。

#### 1. StreamWriter 类

StreamWriter 类是专门用来处理文本文件的类，可以方便地向文本文件中写入字符串，同时它也负责重要的转换和处理向 FileStream 对象写入工作。

StreamWriter 类的常用属性及说明如表 15.10 所示。

表 15.10 StreamWriter 类的常用属性及说明

属    性	说    明
Encoding	获取将输出写入到其中的 Encoding
Formatprovider	获取控制格式设置的对象
NewLine	获取或设置由当前 TextWriter 使用的行结束符字符串

StreamWriter 类的常用方法及说明如表 15.11 所示。

表 15.11 StreamWriter 类的常用方法及说明

方    法	说    明
Close	关闭当前的 StringWriter 和基础流
Write	写入到 StringWriter 的此实例中
WriteLine	写入重载参数指定的某些数据，后跟行结束符

#### 2. StreamReader 类

StreamReader 类是专门用来读取文本文件的类，它可以从底层 Stream 对象创建 StreamReader 对象的实

例，而且也能指定编码规范参数。创建 `StreamReader` 对象后，它还提供了许多用于读取和浏览字符数据的方法。

`StreamReader` 类的常用方法及说明如表 15.12 所示。

表 15.12 `StreamReader` 类的常用方法及说明

方 法	说 明
<code>Close</code>	关闭 <code>StringReader</code>
<code>Read</code>	读取输入字符串中的下一个字符或下一组字符
<code>ReadLine</code>	从基础字符串中读取一行
<code>ReadToEnd</code>	将整个流或从流的当前位置到流的结尾作为字符串读取

**例 15.24** 下面演示如何对文本文件进行写入与读取，程序开发步骤如下。（实例位置：光盘\mr\15\sl\15.24）

(1) 新建一个 Windows 应用程序，命名为 RWTxtByIO，默认窗体为 Form1.cs。

(2) 在 Form1 窗体中添加一个 `SaveFileDialog` 控件、一个 `OpenFileDialog` 控件、一个 `TextBox` 控件和两个 `Button` 控件。其中，`SaveFileDialog` 控件用来显示“另存为”对话框；`OpenFileDialog` 控件用来显示“打开”对话框；`TextBox` 控件用来输入要写入文本文件的内容和显示选中文本文件的内容；`Button` 控件分别用来打开“另存为”对话框并执行文本文件写入操作和打开“打开”对话框并执行文本文件读取操作。

(3) 程序主要代码如下。

```
private void button1_Click(object sender, EventArgs e)
{
    if (textBox1.Text == string.Empty)
    {
        MessageBox.Show("要写入的文件内容不能为空");
    }
    else
    {
        //设置保存文件的格式
        saveFileDialog1.Filter = "文本文件(*.txt)|*.txt";
        if (saveFileDialog1.ShowDialog() == DialogResult.OK)
        {
            //使用“另存为”对话框中输入的文件名创建 StreamWriter 对象
            StreamWriter sw = new StreamWriter(saveFileDialog1.FileName, true);
            sw.WriteLine(textBox1.Text); //向创建的文件中写入内容
            sw.Close(); //关闭当前文件写入流
            textBox1.Text = string.Empty;
        }
    }
}

private void button2_Click(object sender, EventArgs e)
{
    openFileDialog1.Filter = "文本文件(*.txt)|*.txt"; //设置打开文件的格式
    if (openFileDialog1.ShowDialog() == DialogResult.OK)
    {
        textBox1.Text = string.Empty;
        //使用“打开”对话框中选择的文件创建 StreamReader 对象
        StreamReader sr = new StreamReader(openFileDialog1.FileName);
        textBox1.Text = sr.ReadToEnd(); //调用 ReadToEnd 方法读取选中文件的全部内容
    }
}
```

```
sr.Close(); //关闭当前文件读取流
```

```
}
```

```
}
```

运行程序，单击“写入”按钮，弹出“另存为”对话框，输入要保存的文件名，单击“保存”按钮，将文本框中的内容写入到文件中；单击“读取”按钮，弹出“打开”对话框，选择要读取的文件，单击“打开”按钮，将选择的文件的内容显示在文本框中。程序运行结果如图 15.5 所示。



图 15.5 使用 I/O 流操作文本文件

#### 15.4.4 使用 I/O 流操作二进制文件

使用 I/O 流操作二进制文件时主要用到 `BinaryWriter` 类和 `BinaryReader` 类，下面对这两个类进行详细讲解。

##### 1. BinaryWriter 类

`BinaryWriter` 类以二进制形式将基元类型写入流，并支持用特定的编码写入字符串，其常用方法及说明如表 15.13 所示。

表 15.13 `BinaryWriter` 类的常用方法及说明

方 法	说 明
<code>Close</code>	关闭当前的 <code>BinaryWriter</code> 和基础流
<code>Seek</code>	设置当前流中的位置
<code>Write</code>	将值写入当前流

##### 2. BinaryReader 类

`BinaryReader` 类用特定的编码将基元数据类型读作二进制值，其常用方法及说明如表 15.14 所示。

表 15.14 `BinaryReader` 类的常用方法及说明

方 法	说 明
<code>Close</code>	关闭当前阅读器及基础流
<code>Read</code>	从基础流中读取字符，并提升流的当前位置
<code>ReadBytes</code>	从当前流中将 count 个字节读入字节数组，并使当前位置提升 count 个字节
<code>ReadInt32</code>	从当前流中读取 4 个字节有符号整数，并使流的当前位置提升 4 个字节
<code>ReadString</code>	从当前流中读取一个字符串。字符串有长度前缀，一次 7 位地被编码为整数

**例 15.25** 下面演示如何对二进制文件进行写入与读取，程序开发步骤如下。（实例位置：光盘\mr\15\sl\15.25）

(1) 新建一个 Windows 应用程序，命名为 `RWBFileByIO`，默认窗体为 `Form1.cs`。

(2) 在 `Form1` 窗体中添加一个 `SaveFileDialog` 控件、一个 `OpenFileDialog` 控件、一个 `TextBox` 控件和

两个 Button 控件。其中，SaveFileDialog 控件用来显示“另存为”对话框；OpenFileDialog 控件用来显示“打开”对话框；TextBox 控件用来输入要写入二进制文件的内容和显示选中二进制文件的内容；Button 控件分别用来打开“另存为”对话框并执行二进制文件写入操作和打开“打开”对话框并执行二进制文件读取操作。

(3) 程序主要代码如下。

```

private void button1_Click(object sender, EventArgs e)           //写入二进制文件
{
    if (textBox1.Text == string.Empty)
    {
        MessageBox.Show("要写入的文件内容不能为空");
    }
    else
    {
        saveFileDialog1.Filter = "二进制文件(*.dat)|*.dat";      //设置保存文件的格式
        if (saveFileDialog1.ShowDialog() == DialogResult.OK)
        {
            //使用“另存为”对话框中输入的文件名创建 FileStream 对象
            FileStream myStream = new FileStream(saveFileDialog1.FileName, FileMode.OpenOrCreate,
FileAccess.ReadWrite);
            BinaryWriter myWriter = new BinaryWriter(myStream); //创建 BinaryWriter 二进制写入流对象
            myWriter.Write(textBox1.Text);                      //以二进制方式向创建的文件中写入内容
            myWriter.Close();                                //关闭当前二进制写入流
            myStream.Close();                               //关闭当前文件流
            textBox1.Text = string.Empty;
        }
    }
}

private void button2_Click(object sender, EventArgs e)           //读取二进制文件
{
    openFileDialog1.Filter = "二进制文件(*.dat)|*.dat";          //设置打开文件的格式
    if (openFileDialog1.ShowDialog() == DialogResult.OK)
    {
        textBox1.Text = string.Empty;
        //使用“打开”对话框中选择的文件名创建 FileStream 对象
        FileStream myStream = new FileStream(openFileDialog1.FileName, FileMode.Open, FileAccess.Read);
        //使用 FileStream 对象创建 BinaryReader 二进制写入流对象
        BinaryReader myReader = new BinaryReader(myStream);
        if (myReader.PeekChar() != -1)
        {
            textBox1.Text = Convert.ToString(myReader.ReadInt32()); //以二进制方式读取文件中的内容
        }
        myReader.Close();                                         //关闭当前二进制读取流
        myStream.Close();                                       //关闭当前文件流
    }
}

```

 说明：本实例的运行结果图与例 15.24 中的运行结果图类似，只是写入和读取文件的方式不同，这里不再给出实例运行结果图。

## 15.5 照猫画虎——基本功训练

### 15.5.1 基本功训练 1——获取文件基本信息

视频讲解：光盘\mr\15\lx\获取文件基本信息.exe

实例位置：光盘\mr\15\zmhh\01

新建一个 Windows 应用程序，在窗体中添加一个 OpenFileDialog 控件、一个 TextBox 控件和一个 Button 控件。其中， OpenFileDialog 控件用来显示“打开”对话框； TextBox 控件用来显示选择的文件名； Button 控件用来打开“打开”对话框并获取选择文件的基本信息。代码如下。

```
private void button1_Click(object sender, EventArgs e)
{
    if (openFileDialog1.ShowDialog() == DialogResult.OK)
    {
        textBox1.Text = openFileDialog1.FileName;
        FileInfo finfo = new FileInfo(textBox1.Text); // 创建 FileInfo 对象
        string strCTime, strLTime, strLWTime, strName, strFName, strDName, strISRead;
        long lgLength;
        strCTime = finfo.CreationTime.ToString(); // 获取文件创建时间
        strLTime = finfo.LastAccessTime.ToString(); // 获取上次访问该文件的时间
        strLWTime = finfo.LastWriteTime.ToString(); // 获取上次写入文件的时间
        strName = finfo.Name; // 获取文件名称
        strFName = finfo.FullName; // 获取文件的完整目录
        strDName = finfo.DirectoryName; // 获取文件的完整路径
        strISRead = finfo.IsReadOnly.ToString(); // 获取文件是否只读
        lgLength = finfo.Length; // 获取文件长度
        MessageBox.Show("文件信息：\n 创建时间：" + strCTime + " 上次访问时间：" + strLTime + "\n 上次写入时间：" + strLWTime + " 文件名称：" + strName + "\n 完整目录：" + strFName + "\n 完整路径：" + strDName + "\n 是否只读：" + strISRead +
        " 文件长度：" + lgLength); // 显示文件基本信息对话框
    }
}
```

运行程序，单击“浏览”按钮，选择指定文件，并弹出显示该文件基本信息的对话框，运行结果如图 15.6 所示。

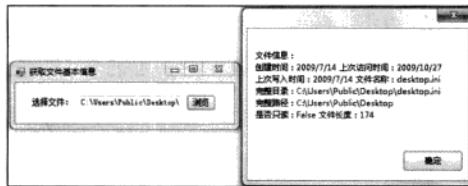


图 15.6 获取文件基本信息

**照猫画虎：获取指定文件的扩展名。提示：可使用 String 类的 Substring 方法从文件名中截取相应的字符串来实现。（20 分）（实例位置：光盘\mr\15\zmhh\01\_zmhh）**

### 15.5.2 基本功训练 2——遍历文件夹

视频讲解：光盘\mr\15\lx\遍历文件夹.exe

实例位置：光盘\mr\15\zmhh\02

新建一个 Windows 应用程序，在窗体中添加一个 FolderBrowserDialog 控件、一个 TextBox 控件、一个 Button 控件和一个 ListView 控件。其中，FolderBrowserDialog 控件用来显示“浏览文件夹”对话框；TextBox 控件用来显示选择的文件夹；Button 控件用来打开“浏览文件夹”对话框；ListView 控件用来显示选择的文件夹中的子文件夹及文件信息。主要代码如下。

```
private void button1_Click(object sender, EventArgs e)
{
    listView1.Items.Clear()                                //清空 ListView 控件中的项
    if (folderBrowserDialog1.ShowDialog() == DialogResult.OK)
    {
        textBox1.Text = folderBrowserDialog1.SelectedPath;
        DirectoryInfo dinfo = new DirectoryInfo(textBox1.Text);      //创建 DirectoryInfo 对象
        FileSystemInfo[] fsinfos = dinfo.GetFileSystemInfos();       //获取指定目录下的所有子目录及文件类型
        foreach (FileSystemInfo fsinfo in fsinfos)
        {
            if (fsinfo is DirectoryInfo)                         //判断是否为文件夹
            {
                DirectoryInfo dirinfo = new DirectoryInfo(fsinfo.FullName);
                //使用获取的文件夹名称创建 DirectoryInfo 对象
                listView1.Items.Add(dirinfo.Name);               //为 ListView 控件添加文件夹信息
                listView1.Items[listView1.Items.Count - 1].SubItems.Add(dirinfo.FullName);
                listView1.Items[listView1.Items.Count - 1].SubItems.Add("");
                listView1.Items[listView1.Items.Count - 1].SubItems.Add(dirinfo.CreationTime.ToString());
            }
            else
            {
                FileInfo finfo = new FileInfo(fsinfo.FullName);   //使用获取的文件名称创建 FileInfo 对象
                listView1.Items.Add(finfo.Name);                  //为 ListView 控件添加文件信息
                listView1.Items[listView1.Items.Count - 1].SubItems.Add(finfo.FullName);
                listView1.Items[listView1.Items.Count - 1].SubItems.Add(finfo.Length.ToString());
                listView1.Items[listView1.Items.Count - 1].SubItems.Add(finfo.CreationTime.ToString());
            }
        }
    }
}
```

运行程序，单击“浏览”按钮，弹出“浏览文件夹”对话框，选择文件夹，单击“确定”按钮，将选择的文件夹中所包含的子文件夹及文件信息显示在 ListView 控件中。程序运行结果如图 15.7 所示。

说明：在上面的代码中，GetDirectories 方法用来返回当前目录的子目录；GetFiles 方法返回当前目录的文件列表；GetFileSystemInfos 方法检索表示当前目录的文件和子目录的强类型 FileSystemInfo 对象的数组。



图 15.7 遍历文件夹

**照猫画虎：使用递归法删除文件夹中的所有文件。提示：可调用 DirectoryInfo 类的 GetFileSystemInfos 方法获取指定文件夹中的所有子文件夹及文件。（20分）（实例位置：光盘\mr\15\zmhh\02\_zmhh）**

### 15.5.3 基本功训练 3——使用 C#操作 INI 文件

 视频讲解：光盘\mr\15\x\使用 C#操作INI文件.exe

 实例位置：光盘\mr\15\zmhh\03

本实例讲解 C#中有关 INI 文件的操作。新建一个 Windows 窗体应用程序，在窗体中添加 4 个 TextBox 控件，分别用来显示 INI 文件中对应的 4 个节点的内容；添加一个 Button 控件，用来对 INI 文件执行修改操作。程序主要代码如下。

//读取INI文件的节点中的内容

```

private void Form1_Load(object sender, EventArgs e)
{
    str = Application.StartupPath + "\\ConnectionString.ini";
    strOne = System.IO.Path.GetFileNameWithoutExtension(str);
    if (File.Exists(str))
    {
        server.Text = ContentReader(strOne, "Data Source", "");
        database.Text = ContentReader(strOne, "DataBase", "");
        uid.Text = ContentReader(strOne, "Uid", "");
        pwd.Text = ContentReader(strOne, "Pwd", "");
    }
}

//修改完成后单击“修改”按钮，可将编辑的内容写入到INI文件中
private void button1_Click(object sender, EventArgs e)
{
    if (File.Exists(str)) //判断是否存在INI文件
    {
        WritePrivateProfileString(strOne, "Data Source", server.Text, str); //修改INI文件中服务器节点的内容
        WritePrivateProfileString(strOne, "DataBase", database.Text, str); //修改INI文件中数据库节点的内容
        WritePrivateProfileString(strOne, "Uid", uid.Text, str); //修改INI文件中用户节点的内容
        WritePrivateProfileString(strOne, "Pwd", pwd.Text, str); //修改INI文件中密码节点的内容
        MessageBox.Show("恭喜你，修改成功！", "提示信息", MessageBoxButtons.OK, MessageBoxIcon.Information);
    }
    else
    {
        MessageBox.Show("对不起，你所要修改的文件不存在，请确认后再进行修改操作！", "提示信息",
        MessageBoxButtons.OK, MessageBoxIcon.Information);
    }
}

```

程序运行结果如图 15.8 所示。



图 15.8 使用 C# 操作 INI 文件

说明：系统 API 函数 WritePrivateProfileString 用于向 INI 文件写入数据。

照猫画虎：参照上面的实例，读取 Windows 操作系统中 ODBC.INI 文件的 Excel Files 节点的内容。提示：可使用系统 API 函数的 GetPrivateProfileString 方法。(20 分)(实例位置：光盘\mr\15\zmhh\03\_zmhh)

#### 15.5.4 基本功训练 4——按行读取文本文件中数据

视频讲解：光盘\mr\15\lx\按行读取文本文件中数据.exe

实例位置：光盘\mr\15\zmhh\04

新建一个 Windows 窗体应用程序，在窗体中添加两个文本框控件，分别用来显示文件路径和文件中的全部数据。程序主要代码如下。

```
private void button1_Click(object sender, EventArgs e)
{
    try
    {
        openFileDialog1.Filter = "文本文件 (*.txt)|*.txt"; //设置选择的文件类型
        openFileDialog1.ShowDialog(); //打开对话框
        textBox1.Text = openFileDialog1.FileName; //设置文件路径
        StreamReader SReader = new StreamReader(textBox1.Text, Encoding.Default); //创建 StreamReader 对象
        string strLine = string.Empty;
        while ((strLine = SReader.ReadLine()) != null) //逐行读取文本文件
        {
            textBox2.Text += strLine + Environment.NewLine; //在文本框中显示读取内容
        }
    }
    catch {}
}
```

程序运行结果如图 15.9 所示。



图 15.9 按行读取文本文件中数据

说明：可使用静态类 Environment 的 NewLine 属性实现文本的换行显示。

照猫画虎：参照上面的实例，读取文本文件中的第一行数据。提示：调用 StreamReader 类的 ReadLine 方法仅读取一次即可。(20 分)(实例位置：光盘\mr\15\zmhh\04\_zmhh)

### 15.5.5 基本功训练 5——获取指定文件夹的上级目录

视频讲解：光盘\mr\15\lx\获取指定文件夹的上级目录.exe

实例位置：光盘\mr\15\zmhh\05

新建一个 Windows 窗体应用程序，在窗体中添加一个 Button 控件，用来选择文件夹；添加一个 TextBox 控件，用来显示选择的文件夹；添加一个 Label 控件，用来显示选择的文件夹的当前目录和上级目录。程序主要代码如下。

```
private void button1_Click(object sender, EventArgs e)
{
    FolderBrowserDialog FBDialo = new FolderBrowserDialog();
    if (FBDialo.ShowDialog() == DialogResult.OK)
    {
        textBox1.Text = FBDialo.SelectedPath; //显示选择的文件夹
        string str1 = textBox1.Text;
        string str2 = Directory.GetParent(str1).FullName; //记录选择的文件夹
        string myInfo = "当前目录是：" + str1; //获取上级目录的全名
        myInfo += "\n 上级目录是：" + str2; //显示当前文件夹
        myInfo += "\n 上级目录是：" + str1; //显示上级文件夹
        label2.Text = myInfo;
    }
}
```

程序运行结果如图 15.10 所示。

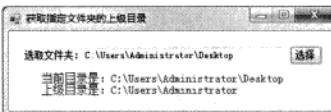


图 15.10 获取指定文件夹的上级目录

说明：Directory 类的 GetParent 方法用来检索指定路径的父文件夹，包括绝对路径和相对路径。

**照猫画虎：**获取指定文件夹所在的磁盘分区。提示：可以使用 Directory.GetDirectoryRoot 方法获取指定文件夹的根目录。(20 分)(实例位置：光盘\mr\15\zmhh\05\_zmhh)

照猫画虎栏目分数统计：

照猫画虎题目	1	2	3	4	5	总分数	
分数							

## 15.6 情景应用——拓展与实践

### 15.6.1 情景应用 1——根据日期动态建立文件

视频讲解：光盘\mr\15\lx\根据日期动态建立文件.exe

实例位置：光盘\mr\15\qjyy\01

本实例主要实现以当前日期时间为依据创建文件的功能，运行本实例，单击“根据系统日期建立文件”

按钮，以当前的日期时间为名称在指定位置创建一个文件。实例运行效果如图 15.11 所示，创建的文件名称如图 15.12 所示。

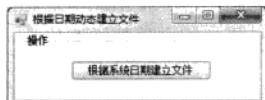


图 15.11 根据日期动态建立文件



图 15.12 创建的文件名称

实现过程如下。

(1) 新建一个 Windows 窗体应用程序，在窗体中添加一个 Button 控件，用来根据当前日期时间动态地创建文件。

(2) 程序主要代码如下。

```
private void btn_Create_Click(object sender, EventArgs e)
{
    FolderBrowserDialog P_FolderBrowserDialog = new FolderBrowserDialog(); //创建浏览文件夹对话框对象
    if (P_FolderBrowserDialog.ShowDialog() == DialogResult.OK) //判断是否选择了文件夹
    {
        //创建文件
        File.Create(P_FolderBrowserDialog.SelectedPath + "\\\" + DateTime.Now.ToString("yyyyMMddHHmmss")
        + ".txt");
    }
}
```

**DIY:** 以当前日期时间为依据创建文件夹。提示：可使用 DirectoryInfo 类的 Create 方法并配合 DateTime.Now 属性创建文件夹。(20 分)(实例位置：光盘\mr\15\qjyy\01\_diy)

## 15.6.2 情景应用 2——文件批量更名

视频讲解：光盘\mr\15\lx\文件批量更名.exe

实例位置：光盘\mr\15\qjyy\02

本实例实现批量修改文件的名称，并在程序中提供了更名的模板，选择后可以快速更名，也可以自定义更名模板，对于文件名还可以进行简体和繁体的相互转换。实例运行效果如图 15.13 所示。



图 15.13 文件批量更名

实现过程如下。

(1) 打开 Visual Studio 2008 开发环境，新建一个 Windows 窗体应用程序，并将其命名为 FileBatchChangeName。

(2) 在窗体中添加一个 OpenFileDialog 控件，用来选择要批量更名的文件；添加一个 ComboBox 控件，用来选择预设模板；添加一个 TextBox 控件，用来设置模板；添加两个 NumericUpDown 控件，分别用来选择起始数字和增量值；添加一个 ListView 控件，用来显示要批量更名的文件的详细信息。

(3) 程序主要代码如下。

```
private void ChangeName()
{
    int flag = 0; //记录出现错误的文件数量
    try
    {
        toolStripProgressBar1.Minimum = 0; //设置进度条的起始值为 0
        toolStripProgressBar1.Maximum = listView1.Items.Count - 1; //设置进度条的最大值为文件数量
        for (int i = 0; i < listView1.Items.Count; i++) //开始读取每一项
        {
            string path = listView1.Items[i].SubItems[4].Text; //获取文件所在目录
            string sourcePath = path + listView1.Items[i].SubItems[0].Text; //获取文件的完整路径
            string newPath = path + listView1.Items[i].SubItems[1].Text; //设置更名后的文件的完整路径
            File.Copy(sourcePath, newPath); //将更名后的文件复制到原目录下
            File.Delete(sourcePath); //删除原目录下的原始文件
            toolStripProgressBar1.Value = i; //设置进度条进度
            listView1.Items[i].SubItems[0].Text = listView1.Items[i].SubItems[1].Text; //设置新的文件名，并显示在 ListView 中
            listView1.Items[i].SubItems[6].Text = "√ 成功"; //设置更改成功后的提示信息
        }
    }
    catch (Exception ex)
    {
        flag++; //如果发生异常，则使 flag 加 1
        MessageBox.Show(ex.Message);
    }
    finally
    {
        tsslError.Text = flag.ToString() + " 个错误"; //显示出错的数量
    }
}
```

**DIY：批量复制文件。** 提示：可使用 File 类的 Copy 方法并结合 for 或 foreach 语句循环复制多个文件。  
**(20 分)** (实例位置：光盘\mr\15\qjyy\02\_diy)

### 15.6.3 情景应用 3——复制文件时显示复制进度

■ 视频讲解：光盘\mr\15\lx\复制文件时显示复制进度.exe

■ 实例位置：光盘\mr\15\qjyy\03

复制文件时显示复制进度实际上就是用文件流来复制文件，并在每一块文件复制后，用进度条来显示文件的复制进度情况。本实例实现了复制文件时显示复制进度的功能，实例运行效果如图 15.14 所示。



图 15.14 复制文件时显示复制进度

实现过程如下。

(1) 新建一个 Windows 窗体应用程序，在窗体中添加一个 OpenFileDialog 控件，用来选择源文件；添加一个 FolderBrowserDialog 控件，用来选择目的文件的路径；添加两个 TextBox 控件，分别用来显示源文件与目的文件的路径；添加 3 个 Button 控件，分别用来选择源文件和目的文件的路径以及实现文件的复制功能；添加一个 ProgressBar 控件，用来显示复制进度条。

(2) 程序主要代码如下。

```
public void CopyFile(string FormerFile, string toFile, int SectSize, ProgressBar progressBar1)
{
    progressBar1.Value = 0; //设置进度栏的当前位置为 0
    progressBar1.Minimum = 0; //设置进度栏的最小值为 0
    FileStream fileToCreate = new FileStream(toFile, FileMode.Create); //创建目的文件, 如果已存在, 则将被覆盖
    fileToCreate.Close(); //关闭所有资源
    fileToCreate.Dispose(); //释放所有资源
    FormerOpen = new FileStream(FormerFile, FileMode.Open, FileAccess.Read); //以只读方式打开源文件
    ToFileOpen = new FileStream(toFile, FileMode.Append, FileAccess.Write); //以写方式打开目的文件
    //根据一次传输的大小计算传输的个数
    int max = Convert.ToInt32(Math.Ceiling((double)FormerOpen.Length / (double)SectSize));
    progressBar1.Maximum = max; //设置进度栏的最大值
    int FileSize; //要复制的文件的大小
    if (SectSize < FormerOpen.Length) //如果分段复制, 即每次复制内容小于文件总长度
    {
        byte[] buffer = new byte[SectSize]; //根据传输的大小, 定义一个字节数组
        int copied = 0; //记录传输的大小
        int tem_n = 1; //设置进度栏中进度块的增加个数
        while (copied <= ((int)FormerOpen.Length - SectSize)) //复制主体部分
        {
            FileSize = FormerOpen.Read(buffer, 0, SectSize); //从 0 开始读, 每次最大读 SectSize
            FormerOpen.Flush(); //清空缓存
            ToFileOpen.Write(buffer, 0, SectSize); //向目的文件写入字节
            ToFileOpen.Flush(); //清空缓存
            ToFileOpen.Position = FormerOpen.Position; //使源文件和目的文件流的位置相同
            copied += FileSize; //记录已复制的大小
            progressBar1.Value = progressBar1.Value + tem_n; //增加进度栏的进度块
        }
        int left = (int)FormerOpen.Length - copied; //获取剩余大小
        FileSize = FormerOpen.Read(buffer, 0, left); //读取剩余的字节
        FormerOpen.Flush(); //清空缓存
        ToFileOpen.Write(buffer, 0, left); //写入剩余的部分
        ToFileOpen.Flush(); //清空缓存
    }
    else //如果整体复制, 即每次复制内容大于文件总长度
    {
```

```
byte[] buffer = new byte[FormerOpen.Length];           //获取文件的大小
FormerOpen.Read(buffer, 0, (int)FormerOpen.Length);    //读取源文件的字节
FormerOpen.Flush();                                  //清空缓存
ToFileOpen.Write(buffer, 0, (int)FormerOpen.Length);   //把缓存中的字节写入文件流
ToFileOpen.Flush();                                //清空缓存
}
FormerOpen.Close();                                //释放所有资源
ToFileOpen.Close();                                //释放所有资源
if (MessageBox.Show("复制完成") == DialogResult.OK) //显示“复制完成”提示对话框
{
    progressBar1.Value = 0;                          //设置进度栏的当前位置为 0
    textBox1.Clear();                             //清空文本
    textBox2.Clear();
    str = "";
}
```

**DIY:** 使用 FileStream 复制大容量文件。提示：使用 FileStream 类将文件以指定的文件流大小进行分割，然后在指定的位置创建一个同名的空文件，将分割后的文件流追加到空文件中，以实现大文件的复制。  
**(20 分)** (实例位置: 光盘\mr\15\qjyy\03\_diy )

#### 15.6.4 情景应用 4——伪装文件夹

 视频讲解：光盘\mr\15\lx\伪装文件夹.exe

 实例位置：光盘\mr\15\qjyy\04

出于安全性的考虑，很多人喜欢通过加密软件对文件夹进行加密，但本实例通过对文件夹进行伪装来提高文件夹的安全性。例如，将文件夹伪装成回收站，那么当双击伪装后的文件夹后就会进入回收站，而不是文件夹。实例运行效果如图 15.15 所示。

实现过程如下。

(1) 打开 Visual Studio 2008 开发环境，新建一个 Windows 窗体应用程序，并将其命名为 CamouflageFolder。

(2) 在窗体中添加一个 FolderBrowserDialog 控件, 用来选择要伪装或还原的文件夹; 添加两个 TextBox 控件, 分别用来显示选择的文件夹和输入类标识符; 添加一个 ComboBox 控件, 用件, 分别用来执行打开浏览文件夹对话框、伪装文件夹和还原原

(3) 程序主要代码如下。

在窗体的后台代码中，首先自定义一个 `GetFolType` 方法，用于根据选择的伪装方式获取其对应的 Windows 文件标识符。代码如下。

```
private string GetFoType()
{
    int Tid = comboBox1.SelectedIndex; //获取选择项索引
    switch (Tid) //根据索引设置 Windows 文件标识符
    {
        case 0: return @"[20D04FE0-3AEA-1069-A2D8-08002B30309D]"; //我的电脑的 Windows 文件标识符
        case 1: return @"[450D8FB0-AD25-11D0-98A8-0800361B1103]"; //我的文档的 Windows 文件标识符
    }
}
```



图 15.15 伪装文件夹

```

        case 2: return @"{992CFFA0-F557-101A-88EC-00DD010CCC48}"; //拨号网络的 Windows 文件标识符
        case 3: return @"{21EC2020-3AEA-1069-A2DD-08002B30309D}"; //控制面板的 Windows 文件标识符
        case 4: return @"{D6277990-4C6A-11CF-8D87-00AA0060F5BF}"; //计划任务的 Windows 文件标识符
        case 5: return @"{2227A280-3AEA-1069-A2DE-08002B30309D}"; //打印机的 Windows 文件标识符
        case 6: return @"{208D2C60-3AEA-1069-A2D7-08002B30309D}"; //网络邻居的 Windows 文件标识符
        case 7: return @"{645FF040-5081-101B-9F08-00AA002F954E}"; //回收站的 Windows 文件标识符
        case 8: return @"{85BBD920-42A0-1069-A2E4-08002B30309D}"; //公文包的 Windows 文件标识符
        case 9: return @"{BD84B380-8CA2-1069-AB1D-08000948F534}"; //字体的 Windows 文件标识符
        case 10: return @"{BDEAD00-C265-11d0-BCED-00A0C90AB50F}"; //Web 文件夹的 Windows 文件标识符
    }
    return @"{20D04FE0-3AEA-1069-A2D8-08002B30309D}"; //若不符合则返回我的电脑 Windows 文件标识符
}

```

自定义一个 Camouflage 方法，用于在指定的文件夹下创建一个名为 desktop.ini 的文件，在此文件中写入伪装类型的 Windows 文件标识符。例如，如果想将文件夹伪装成“我的电脑”，那么在调用此方法后，desktop.ini 文件中就被写入“我的电脑”的 Windows 文件标识符“CLSID={20D04FE0-3AEA-1069-A2D8-08002B30309D}”。Camouflage 方法实现代码如下。

```

private void Camouflage(string str) //用于创建 desktop.ini 文件
{
    StreamWriter sw = File.CreateText(txtFolderPath.Text.Trim() + @"\desktop.ini");
    //用 desktop.ini 文件创建 StreamWriter 对象
    sw.WriteLine(@"[.ShellClassInfo]"); //写入 “[.ShellClassInfo]”
    sw.WriteLine("CLSID=" + str); //写入 Windows 文件标识符
    sw.Close(); //关闭对象
    File.SetAttributes(txtFolderPath.Text.Trim() + @"\desktop.ini", FileAttributes.Hidden); //设置 desktop.ini 文件为隐藏
    File.SetAttributes(txtFolderPath.Text.Trim(), FileAttributes.System); //设置文件夹属性为系统属性
    MessageBox.Show("伪装成功！", "提示", MessageBoxButtons.OK, MessageBoxIcon.Information);
}

```

程序中提供了预置伪装和自定义伪装两种类型，如果选择自定义伪装类型，需要手动输入要伪装类型的 Windows 文件标识符，然后再调用 Camouflage 方法进行伪装，代码如下：

```

private void button2_Click(object sender, EventArgs e)
{
    if (this.txtFolderPath.Text == "") //如果没有选择文件
    {
        MessageBox.Show("请选择文件夹路径！", "提示信息", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    else
    {
        try
        {
            if (txtID.ReadOnly == false) //如果自定义 Windows 文件标识符
            {
                string str = txtID.Text.Trim();
                if (str.StartsWith(".")) //获取选择的 Windows 文件标识符
                    str = str.Substring(1); //如果以 “.” 开头
                if (!str.StartsWith("{") || str.Trim().Length != 38) //则去掉 “.”
                {
                    MessageBox.Show("自定义类型错误！", "提示信息", MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
            }
        }
    }
}

```

```

        else
    {
        Camouflage(str);
    }
}
else
{
    Camouflage(GetFoType());
}
}
catch
{
    MessageBox.Show("不要进行多次伪装！","提示",MessageBoxButtons.OK,MessageBoxIcon.Error);
}
}
}

```

**DIY：**获取文件夹中的图标资源。提示：可使用 API 函数 SHGetFileInfo 获得文件的基本信息及其图标，然后用 API 函数 ExtractIconEx 获取图标句柄并进行相应的显示。(20 分)(实例位置：光盘\mr\15\qjyy\04\_diy)

### 15.6.5 情景应用 5——对指定文件夹中的文件进行分类存储

■ 视频讲解：光盘\mr\15\lx\对指定文件夹中的文件进行分类存储.exe

■ 实例位置：光盘\mr\15\qjyy\05

本实例实现对指定文件夹中的文件进行分类存储（例如，将 txt 类型的文件放在一个文件夹中，将 doc 类型的文件放在另一个文件夹中）。实例运行效果如图 15.16 所示，单击“选择”按钮，选择要整理的文件夹；单击“整理”按钮，对选择的文件夹中的文件进行分类存储；单击“查看”按钮，可以查看整理后的文件夹，如图 15.17 所示。



图 15.16 对指定文件夹中的文件进行分类存储

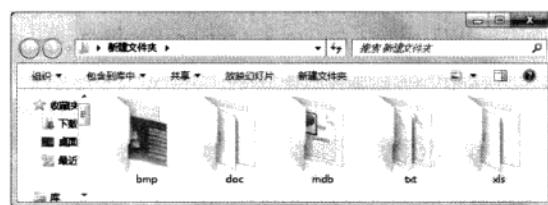


图 15.17 查看整理后的文件夹

实现过程如下。

(1) 新建一个 Windows 窗体应用程序，在窗体中添加一个 TextBox 控件，用来显示选择的文件夹；添加 3 个 Button 控件，分别用来执行选择文件夹、整理文件夹和查看整理后文件夹的操作。

(2) 程序主要代码如下。

```

Private void button2_Click(object sender, EventArgs e)
{
    List<string> listExten = new List<string>(); // 创建泛型集合对象
    DirectoryInfo Dinfo = new DirectoryInfo(textBox1.Text); // 创建 DirectoryInfo 对象
    FileInfo[] Finfos = Dinfo.GetFiles(); // 获取文件夹中的所有文件
    string strExten = " "; // 定义一个变量，用来存储文件扩展名
}

```

```

foreach (FileInfo Finfo in Finfos)           //遍历所有文件
{
    strExten = Finfo.Extension;
    if (!listExten.Contains(strExten))
    {
        listExten.Add(strExten.TrimStart(' ')); //将扩展名去掉之后添加到泛型集合中
    }
}
for (int i = 0; i < listExten.Count; i++)      //遍历泛型集合
{
    Directory.CreateDirectory(textBox1.Text + listExten[i]); //创建文件夹
}
foreach (FileInfo Finfo in Finfos)           //遍历所有文件
{
    //将文件移动到对应的文件夹中
    Finfo.MoveTo(textBox1.Text + Finfo.Extension.TrimStart('.') + "\\" + Finfo.Name);
}
MessageBox.Show("整理完毕！", "提示", MessageBoxButtons.OK, MessageBoxIcon.Information);
}

```

**DIY:** 将多个小文件合并为一个文件。提示：首先要以合并后的文件名和 FileMode.Append 方式来创建 FileStream 流，以 FileStream 流来创建 BinaryWriter 文件书写器，该文件书写器用来合并分割的文件。

(20 分)(实例位置：光盘\mr\15\qjyy\05\_diy)

情景应用 DIY 栏目分数统计：

DIY 题目	1	2	3	4	5	总分数	
分数							

## 15.7 自我测试

### 一. 选择题（每题 10 分，5 道题）

1. 使用 FileInfo 类可以实现以下哪些功能（ ）。  
A. 创建文件      B. 删除文件      C. 复制文件      D. 以上均正确
2. 关于文件路径“C:\data\text.doc”的描述正确是（ ）。  
A. 路径中的文件名是 text.doc      B. 目录路径为 C:\data\text.doc  
C. 目录路径为 C:\data\text      D. 路径中的文件名是 text
3. 用于按二进制方式读写文件的类是（ ）。  
A. BinaryReader 和 BinaryWriter      B. BinaryReader  
C. BinaryWriter      D. System.IO
4. 用于对文件进行创建、删除、复制、移动、打开等操作的类是（ ）。  
A. File 和 FileStream      B. File      C. Stream      D. System.IO
5. 用于对目录进行管理的类是（ ）。  
A. System.IO      B. Directory      C. File      D. Stream

## 二、填空题（每题 10 分，5 道题）

1. 对于文件系统的操作，相关类都在（ ）命名空间中。
2. （ ）类是提供操作整个文件和目录的类，它不涉及文件内部数据的访问。
3. （ ）方法可以将一整行文本读取到字符串中，但不包括行尾符。
4. FileStream 对象支持使用（ ）方法对文件进行随机访问，它允许将读/写位置移动到文件的任意位置。
5. （ ）类是 FileInfo 和 DirectoryInfo 类的抽象基类。

测试分数统计：

类别	第 1 题	第 2 题	第 3 题	第 4 题	第 5 题
选择题分数					
填空题分数					
总分数					

## 15.8 行动指南

开始日期： 年 月 日

结束日期： 年 月 日

序号	内 容	行动指南
1	照猫画虎栏目 分数（ ）	分数>75 分 优秀，基本功掌握得不错，加油！
		75 分>分数>50 分 及格，知识掌握得不牢，重新做一遍照猫画虎。
		分数<50 分 请使用光盘中提供的“实战能力测试系统”进行提高训练。
	情景应用栏目 分数（ ）	分数>75 分 优秀，综合应用能力很强。
		75 分>分数>50 分 及格，综合应用能力需提高，再练一遍情景应用。
		分数<50 分 请使用光盘中提供的“实战能力测试系统”进行提高训练。
	自我测试栏目 分数（ ）	分数>75 分 优秀，有成为编程高手的潜质。
		分数<75 分 请使用光盘中提供的“实战能力测试系统”进行提高训练。
	综合评价	反复训练后，以上各项分数都在 75 分以上，方可进入下一堂课学习。
2	编程能力培养：本栏目提供了一些实践题目，对于培养编程能力很有效，要做好。	1. 设计一个文件压缩软件。提示：可考虑使用 Gzip 类，该类位于 System.IO.Compression 命名空间下，提供用于压缩和解压缩流的方法和属性。当使用 GzipStream 类构造一个压缩流后，就可以使用该类的 Write 方法写数据，从而实现压缩文件的功能。 2. 设计一个文件解压缩软件。提示：可考虑使用 Gzip 类，首先使用 GzipStream 类构造一个解压缩流，然后调用该类的 Read 方法读取数据，从而实现解压缩文件的功能。
3	编程习惯培养：本堂课培养读者在学习开发中记笔记的习惯，把开发中遇到的问题、总结的经验记录下来。	

续表

序号	内 容	行 动 指 南
4	创新能力培养：看看身边有没有可以用编程解决的问题，记录到右边的表格中。根据学习进度，尝试编写解决这些问题的小程序。	

## 15.9 成功可以复制——中国网络游戏产业的领军人陈天桥

1994 年，陈天桥毕业于复旦大学经济学专业，毕业后的最初几年，先后在上海陆家嘴集团和金信证券公司工作。

1999 年，陈天桥以 50 万元启动资金和 20 名员工为基础，在上海市的一所民宅里创立了盛大网络有限责任公司，并推出网络虚拟社区“天堂归谷”。

2000 年，盛大网络获得了中华网 300 万美元的注资。于是很快，初出茅庐的盛大广泛涉足了网上互动娱乐社区的开发经营、即时通信软件的开发和服务以及网上动画、漫画的开发经营，而这种多元化的经营使盛大网络进入了迷茫而无序的发展状态。在功成名就之后，陈天桥告诉前来取经的创业者：“当你认准一个方向的时候全力以赴，只有专注的企业才能成功，多元化的企业可以存活但是很难成功。”而这正是他的切肤之痛，盲目发展很快结出了恶果，盛大网络陷入困顿中。

在总结了自己的失败原因之后，陈天桥开始寻求改变。这时《传奇》进入了他的视野并把他深深吸引了，他立刻给中华网写了厚厚一叠项目建议书，并告知投资方这是一个非常好的商业领域，认为到年底不但能赚钱而且能赚大钱，但是投资方觉得陈天桥在讲一个神话，于是对他说“你可以一个人走，但是我们不陪你走。”最终，陈天桥与中华网分手，中华网按股份留给陈天桥 30 万美元。

2001 年 7 月 14 日，盛大和《传奇》海外版权持有商 Actoz 以每年 30 万美元的价格签约。陈天桥把剩余的 30 万美元全部交给了 Actoz。合同签完后，盛大资金就不足了，但游戏运营才刚刚开始，光服务器和网络带宽就需要一大笔钱，形势十分危险。为了解决硬件设备问题，他就拿着与韩国方面签订的合约找到商家，告诉他们盛大要运作韩国人的游戏，申请试用机器两个月。对方一看是国际正规合同，于是就同意了。然后，拿着服务器的合约，以同样的方式找通信运营商谈，通信运营商最终给了盛大两个月测试期免费的带宽试用。有了韩方的合同，再加上服务器厂家和中国电信的支持，陈天桥又取得了游戏分销商的信任，代销盛大游戏点卡，分成 33%。

2001 年 9 月 28 日，《传奇》开始公测，两个月后正式收费，不久在线人数迅速突破 40 万大关，全国点卡集体告罄，资金迅速回笼，盛大安然渡过了这场生死玄关。此时离盛大和 Actoz 签约仅仅 4 个月。

2004 年，专注于游戏产业而获得巨额财富的盛大在纳斯达克上市，陈天桥所持有的盛大网络股票市值



就高达 50 亿元。时至今日，陈天桥一直把游戏产业作为盛大公司的主营业务，一举成为中国网络游戏业的领头羊。

 经典语录

当你认准一个方向的时候全力以赴，只有专注的企业才能成功，多元化的企业可以存活但是很难成功。

 深度评价

从陈天桥的成功历程我们可以看出专注的重要性，由于他一时的疏忽，在企业实力还不是很强大的情况下涉足过多的产业，差一点断送了盛大的“前途”。后来盛大专注于网络游戏产业的经营，终于走出困境而获得巨大成功，所以我们学习编程也一样，不要三心二意，今天想把 C# 学好，过几天又想把 VB 学好，这样只能是一事无成，最终哪个也没有学好。

# 第 16 堂课

## GDI+绘图技术

( 视频讲解：145分钟)

用户界面上的窗体和控件非常引人注目，但有时还需要在屏幕上使用颜色和图形对象。例如，可能需要使用线条或弧线来开发游戏，或者需要使用多个移动的图形来开发屏保程序，在这种情况下，只使用 WinForms 控件是不够的，还需要使用图形绘制功能。C# 中的图形绘制通过 GDI+ 技术实现，GDI+ 是图形设备接口的高级版本。本堂课将对 GDI+ 绘图技术进行详细讲解。

学习摘要：

- ▶ 了解什么是 GDI+ 技术
- ▶ 熟悉 GDI+ 操作的基础类（Graphics 类）的使用
- ▶ 熟悉 Pen 类的使用
- ▶ 熟悉 Brush 类的使用
- ▶ 掌握如何绘制直线和矩形
- ▶ 掌握如何绘制椭圆、弧形和扇形
- ▶ 掌握如何绘制多边形
- ▶ 掌握如何绘制文本和图形
- ▶ 掌握 GDI+ 技术在实际中的应用

## 16.1 GDI+绘图基础

### 16.1.1 GDI+概述

GDI+是 GDI(即 Windows 早期版本中附带的 Graphics Device Interface)的后继者, 它是一种构成 Windows XP 操作系统的子系统的应用程序编程接口 (API)。

一般来说, 目前有 3 种基本类型的绘图界面, 分别为 Windows 窗体上的控件、要发送给打印机的页面和内存中的位图图像。GDI+将应用程序与图形硬件分隔, 使程序员能够创建与设备无关的应用程序。GDI+主要用于在窗体上绘制各种图形图像, 可用于绘制各种数据图形及数学仿真等。另外, GDI+还可以在窗体程序中产生很多自定义的图形, 便于开发人员展示各种图形化的数据。

### 16.1.2 创建 Graphics 对象

Graphics 类是 GDI+的核心, Graphics 对象表示 GDI+绘图表面, 它提供将对象绘制到显示设备的方法。Graphics 类封装了绘制直线、曲线、图形、图像和文本的方法, 它是进行一切 GDI+操作的基础类。创建 Graphics 对象有以下 3 种方法。

- 在窗体或控件的 Paint 事件中创建, 将其作为 PaintEventArgs 的一部分。

例 16.01 在 Paint 事件中创建 Graphics 对象, 代码如下。

```
private void Form1_Paint(object sender, PaintEventArgs e) //窗体的 Paint 事件
{
    Graphics g = e.Graphics; //创建 Graphics 对象
}
```

技巧: 在为控件创建绘制代码时, 通常会使用此方法来获取对图形对象的引用。

- 调用控件或窗体的 CreateGraphics 方法以获取对 Graphics 对象的引用, 该对象表示控件或窗体的绘图画面。

例 16.02 在窗体的 Load 事件中, 通过 CreateGraphics 方法创建 Graphics 对象, 代码如下。

```
private void Form1_Load(object sender, EventArgs e)
{
    Graphics g; //声明一个 Graphics 对象
    g = this.CreateGraphics(); //使用 CreateGraphics 方法创建 Graphics 对象
}
```

技巧: 如果在已存在的窗体或控件上绘图, 应使用此方法。

- 由从 Image 继承的任何对象创建 Graphics 对象。

例 16.03 在窗体的 Load 事件中, 通过 FromImage 方法创建 Graphics 对象, 代码如下。

```
private void Form1_Load(object sender, EventArgs e)
{
    Bitmap mbit = new Bitmap(@"C:\ls.bmp"); //创建 Bitmap 类
    Graphics g = Graphics.FromImage(mbit); //通过 FromImage 方法创建 Graphics 对象
}
```

技巧: 此方法在需要更改已存在的图像时十分有用。

### 16.1.3 创建 Pen 对象

Pen 类主要用于绘制线条或者线条组合成的其他几何形状。其构造函数如下。

语法：public Pen(Color color, float width)

说明：color 用来设置 Pen 的颜色；width 用来设置 Pen 的宽度。

例 16.04 创建一个 Pen 对象，使其颜色为蓝色，宽度为 2，代码如下。

```
Pen mypen1 = new Pen(Color.Blue, 2); //创建一个 Pen 类，并设置其颜色和宽度
```

### 16.1.4 创建 Brush 对象

Brush 类主要用于填充几何图形，如将正方形和圆形填充其他颜色。该类是一个抽象基类，不能进行实例化，若要创建一个画笔对象，需要使用从 Brush 派生出的类，如 SolidBrush 类和 HatchBrush 类等，下面对这些派生出的类进行详细介绍。

#### 1. SolidBrush 类

SolidBrush 类定义单色画笔，画笔用于填充图形形状，如矩形、椭圆、扇形、多边形和封闭路径等，其构造函数如下。

语法：public SolidBrush(Color color)

说明：color 表示画笔的颜色。

例 16.05 创建一个 Windows 应用程序，通过使用 SolidBrush 对象将绘制的矩形填充为红色，代码如下。

(实例位置：光盘\mr\16\s\16.05)

```
private void button1_Click(object sender, EventArgs e)
{
    Graphics ghs = this.CreateGraphics(); //创建 Graphics 对象
    Brush mybs = new SolidBrush(Color.Red); //使用 SolidBrush 类创建一个 Brush 对象
    Rectangle rt = new Rectangle(10,10,100,100); //绘制一个矩形
    ghs.FillRectangle(mybs,rt); //用 Brush 填充 Rectangle
}
```

程序运行结果如图 16.1 所示。

#### 2. HatchBrush 类

HatchBrush 类提供了一种特定样式的图形，用来制作填满整个封闭区域的绘图效果，其构造函数如下。

语法：public HatchBrush(HatchStyle hatchstyle, Color foreColor)

说明：hatchstyle 为 HatchStyle 值之一，表示此 HatchBrush 所绘制的图案；foreColor 为 Color 结构，表示此 HatchBrush 所绘制线条的颜色。

说明： HatchBrush 类位于 System.Drawing.Drawing2D 命名空间下。

例 16.06 创建一个 Windows 应用程序，利用 HatchBrush 对象填充 5 个长条图示的封闭区域，代码如下。(实例位置：光盘\mr\16\s\16.06)

```
private void button1_Click(object sender, EventArgs e)
{
    Graphics ghs = this.CreateGraphics(); //创建 Graphics 对象
    for (int i = 1; i < 6; i++) //使用 for 循环
    {
```

```

        HatchStyle hs=(HatchStyle)(5+i);
        HatchBrush hb = new HatchBrush(hs,Color.White);
        Rectangle rtl = new Rectangle(10,50*i,50*i,50);
        ghs.FillRectangle(hb,rtl);
    }
}

```

程序运行结果如图 16.2 所示。

### 3. LinearGradientBrush 类

LinearGradientBrush 类提供一种渐变色彩的特效，用于填满图形的内部区域，其构造函数如下。

语法：public LinearGradientBrush(Point point1, Point point2, Color color1, Color color2)

LinearGradientBrush 类的构造函数语法中的参数说明如表 16.1 所示。

表 16.1 LinearGradientBrush 类构造函数语法中的参数说明

参 数	说 明	参 数	说 明
point1	表示线形渐变的开始点	color1	表示线形渐变的开始色彩
point2	表示线形渐变的结束点	color2	表示线形渐变的结束色彩

说明：LinearGradientBrush 类位于 System.Drawing.Drawing2D 命名空间下。

**例 16.07** 创建一个 Windows 应用程序，通过 LinearGradientBrush 类绘制线形渐变图形，代码如下。（实例位置：光盘\mr\16\sl\16.07）

```

private void button1_Click(object sender, EventArgs e)
{
    //创建两个 Point 类
    Point p1 = new Point(100,100);
    Point p2 = new Point(150,150);
    //创建 LinearGradientBrush 类，设置其使用黑色和白色进行渐变
    LinearGradientBrush lgb = new LinearGradientBrush(p1,p2,Color.Black,Color.White);
    Graphics ghs = this.CreateGraphics(); //创建 Graphics 类
    //设置 WrapMode 属性指示该 LinearGradientBrush 的环绕模式
    lgb.WrapMode = WrapMode.TileFlipX;
    ghs.FillRectangle(lgb,15,15,150,150); //填充绘制矩形
}

```

程序运行结果如图 16.3 所示。



图 16.1 填充矩形

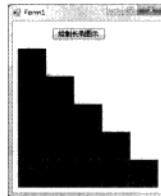


图 16.2 绘制长条图示



图 16.3 绘制渐变

## 16.2 基本图形绘制

介绍完 GDI+绘图技术的几个基本对象后，下面讲解如何通过这些基本对象绘制常见的几何图形，包括

直线、矩形、椭圆、多边形、文本及图形等。

### 16.2.1 绘制直线和矩形

#### 1. 绘制直线

调用 Graphics 类中的 DrawLine 方法,结合 Pen 对象可以绘制直线,DrawLine 方法有以下两种重载形式。

(1) 第一种用于绘制一条连接两个 Point 结构的线

语法: public void DrawLine(Pen pen,Point pt1,Point pt2)

说明: pen 表示 Pen 对象,它确定线条的颜色、宽度和样式; pt1 是一个 Point 结构,表示要连接的第一个点; pt2 是一个 Point 结构,表示要连接的第二个点。

(2) 第二种用于绘制一条连接由坐标指定的两个点的线条。

语法: public void DrawLine(Pen pen,int x1,int y1,int x2,int y2)

DrawLine 方法语法中的参数说明如表 16.2 所示。

表 16.2 DrawLine 方法语法中的参数说明

参 数	说 明
pen	Pen 对象, 它确定线条的颜色、宽度和样式
x1	第一个点的 x 坐标
y1	第一个点的 y 坐标
x2	第二个点的 x 坐标
y2	第二个点的 y 坐标

**例 16.08** 创建一个 Windows 应用程序,向窗体中添加两个 Button 控件,分别用于绘制横向和纵向的直线,代码如下。(实例位置: 光盘\mr\16\sl\16.08)

```
private void button1_Click(object sender, EventArgs e)
{
    Pen blackPen = new Pen(Color.Black, 3);           //创建 Pen 类
    Point point1 = new Point(10, 50);                 //创建一个 Point 类
    Point point2 = new Point(100, 50);                //创建一个 Point 类
    Graphics g = this.CreateGraphics();               //创建一个 Graphics 类
    g.DrawLine(blackPen, point1, point2);             //调用 DrawLine 方法绘制直线
}

private void button2_Click(object sender, EventArgs e)
{
    Graphics graphics = this.CreateGraphics();         //创建 Graphics 类
    Pen myPen = new Pen(Color.Black, 3);               //创建 Pen 类
    graphics.DrawLine(myPen, 150, 30, 150, 100);       //调用 DrawLine 方法绘制直线
}
```

程序运行结果如图 16.4 所示。

#### 2. 绘制矩形

通过 Graphics 类中的 DrawRectangle 方法可以绘制矩形图形,该方法可以绘制由一对坐标、宽度和高度指定的矩形。

语法: public void DrawRectangle(Pen pen,int x,int y,int width,int height)

DrawRectangle 方法语法中的参数说明如表 16.3 所示。

表 16.3 DrawRectangle 方法语法中的参数说明

参 数	说 明
pen	Pen 对象, 它确定矩形的颜色、宽度和样式
x	要绘制矩形的左上角的 x 坐标
y	要绘制矩形的左上角的 y 坐标
width	要绘制矩形的宽度
height	要绘制矩形的高度

**例 16.09** 创建一个 Windows 应用程序, 向窗体中添加一个 Button 控件, 用于调用 Graphics 类中的 DrawRectangle 方法绘制矩形, 代码如下。(实例位置: 光盘\mr\16\sl\16.09)

```
private void button1_Click(object sender, EventArgs e)
{
    Graphics graphics = this.CreateGraphics();           //声明一个 Graphics 对象
    Pen myPen = new Pen(Color.Black, 8);                 //创建 Pen 类
    //调用 Graphics 对象的 DrawRectangle 方法, 绘制矩形
    graphics.DrawRectangle(myPen, 10, 10, 150, 100);
}
```

程序运行结果如图 16.5 所示。

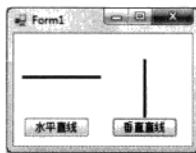


图 16.4 绘制直线



图 16.5 绘制矩形

## 16.2.2 绘制椭圆、弧和扇形

### 1. 绘制椭圆

通过 Graphics 类中的 DrawEllipse 方法可以轻松地绘制椭圆, 此方法可以绘制一对坐标、高度和宽度指定的椭圆。

语法: public void DrawEllipse(Pen pen,int x,int y,int width,int height)

DrawEllipse 方法语法中的参数说明如表 16.4 所示。

表 16.4 DrawEllipse 方法语法中的参数说明

参 数	说 明
pen	Pen 对象, 它确定曲线的颜色、宽度和样式
x	定义椭圆边框的左上角的 x 坐标
y	定义椭圆边框的左上角的 y 坐标
width	定义椭圆边框的宽度
height	定义椭圆边框的高度

**例 16.10** 创建一个 Windows 应用程序，通过 Graphics 类中的 DrawEllipse 方法绘制一个线条宽度为 3 的黑色椭圆，代码如下。（实例位置：光盘\mr\16\sl\16.10）

```
private void button1_Click(object sender, EventArgs e)
{
    Graphics graphics = this.CreateGraphics();           //创建 Graphics 对象
    Pen myPen = new Pen(Color.Black, 3);                //创建 Pen 对象
    graphics.DrawEllipse(myPen, 100, 50, 100, 50);       //绘制椭圆
}
```

程序运行结果如图 16.6 所示。

### 2. 绘制圆弧

通过 Graphics 类中的 DrawArc 方法可以绘制圆弧，此方法可以绘制由一对坐标、宽度和高度指定的椭圆。

语法：public void DrawArc(Pen pen, Rectangle rect, float startAngle, float sweepAngle)

DrawArc 方法语法中的参数说明如表 16.5 所示。

表 16.5 DrawArc 方法语法中的参数说明

参    数	说    明
pen	Pen 对象，它确定弧线的颜色、宽度和样式
rect	Rectangle 结构，它定义椭圆的边界
startAngle	从 x 轴到弧线的起始点沿顺时针方向度量的角（以度为单位）
sweepAngle	从 startAngle 参数到弧线的结束点沿顺时针方向度量的角（以度为单位）

**例 16.11** 创建一个 Windows 应用程序，使用 Graphics 类中的 DrawArc 方法绘制一条线条宽度为 3 的黑色圆弧，代码如下。（实例位置：光盘\mr\16\sl\16.11）

```
private void button1_Click(object sender, EventArgs e)
{
    Graphics ghs = this.CreateGraphics();           //创建 Graphics 类
    Pen myPen = new Pen(Color.Black, 3);             //创建 Pen 类
    Rectangle myRectangle = new Rectangle(70, 20, 100, 60); //定义一个 Rectangle 结构
    //调用 Graphics 对象的 DrawArc 方法绘制圆弧
    ghs.DrawArc(myPen, myRectangle, 210, 120);
}
```

程序运行结果如图 16.7 所示。

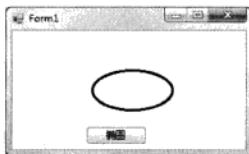


图 16.6 绘制椭圆

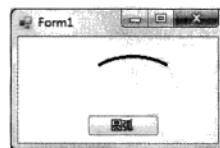


图 16.7 绘制圆弧

### 3. 绘制扇形

通过 Graphics 类中的 DrawPie 方法可以绘制扇形，此方法可以绘制由一对坐标、宽度、高度以及两条射线所指定的椭圆。

语法：public void DrawPie(Pen pen, float x, float y, float width, float height, float startAngle, float sweepAngle)

DrawPie 方法语法中的参数说明如表 16.6 所示。

表 16.6 DrawPie 方法语法中的参数说明

参 数	说 明
pen	Pen，它确定扇形的颜色、宽度和样式
x	边框的左上角的 x 坐标，该边框定义扇形所属的椭圆
y	边框的左上角的 y 坐标，该边框定义扇形所属的椭圆
width	边框的宽度，该边框定义扇形所属的椭圆
height	边框的高度，该边框定义扇形所属的椭圆
startAngle	从 x 轴到扇形的第一条边沿顺时针方向度量的角（以度为单位）
sweepAngle	从 startAngle 参数到扇形的第二条边沿顺时针方向度量的角（以度为单位）

**例 16.12** 创建一个 Windows 应用程序，通过 Graphics 类中的 DrawPie 方法绘制一个线条宽度为 3 的黑色扇形，它的起始坐标分别为 50 和 50，代码如下。（实例位置：光盘\mr\16\sl\16.12）

```
private void button1_Click(object sender, EventArgs e)
{
    Graphics ghs = this.CreateGraphics(); //创建 Graphics 类
    Pen mypen = new Pen(Color.Black,3); //创建 Pen 类
    ghs.DrawPie(mypen,50,50,120,100,210,120); //绘制扇形
}
```

程序运行结果如图 16.8 所示。



图 16.8 绘制扇形

### 16.2.3 绘制多边形

多边形是有 3 条或更多直边的闭合图形。例如，三角形是有 3 条边的多边形，矩形是有 4 条边的多边形，五边形是有 5 条边的多边形。若要绘制多边形，需要使用 Graphics 对象提供的 DrawPolygon 方法，该方法用于绘制由一组 Point 结构定义的多边形。

语法：public void DrawPolygon(Pen pen,Point[] points)

说明：pen 表示 Pen 对象，用于确定多边形的颜色、宽度和样式；points 是一个 Point 结构数组，用来表示多边形的顶点。

**例 16.13** 创建一个 Windows 应用程序，通过 Graphics 类中的 DrawPolygon 方法绘制多边形，其参数分别是 Pen 对象和 Point 对象数组，绘制一个线条宽度为 3 的黑色多边形，代码如下。（实例位置：光盘\mr\16\sl\16.13）

```
private void button1_Click(object sender, EventArgs e)
{
    Graphics ghs = this.CreateGraphics(); //创建 Graphics 类
    Pen myPen = new Pen(Color.Black,3); //创建 Pen 类
```

```

Point point1 = new Point(80, 20);           //创建 Point 类
Point point2 = new Point(40, 50);           //创建 Point 类
Point point3 = new Point(80, 80);           //创建 Point 类
Point point4 = new Point(160, 80);          //创建 Point 类
Point point5 = new Point(200, 50);          //创建 Point 类
Point point6 = new Point(160, 20);          //创建 Point 类
//创建 Point 结构数组
Point[] myPoints = { point1, point2, point3, point4, point5, point6 };
//调用 Graphics 对象的 DrawPolygon 方法绘制一个多边形
ghs.DrawPolygon(myPen, myPoints);
}

```

程序运行结果如图 16.9 所示。

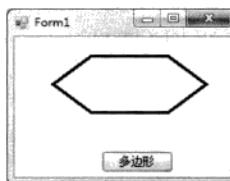


图 16.9 绘制多边形

#### 16.2.4 绘制文本

通过 Graphics 类中的 DrawString 方法可以绘制文本，此方法在指定位置并且用指定的 Brush 和 Font 对象绘制指定的文本字符串。

语法：public void DrawString(string s,Font font,Brush brush,float x,float y)

DrawString 方法语法中的参数说明如表 16.7 所示。

表 16.7 DrawString 方法语法中的参数说明

参 数	说 明
s	要绘制的字符串
font	Font，它定义字符串的文本格式
brush	Brush，它确定所绘制文本的颜色和纹理
x	所绘制文本的左上角的 x 坐标
y	所绘制文本的左上角的 y 坐标

**例 16.14** 创建一个 Windows 应用程序，通过 Graphics 类中的 DrawString 方法在窗体上绘制“明日科技 C# 编程词典”文本，代码如下。（实例位置：光盘\mr\16\sl\16.14）

```

private void button1_Click(object sender, EventArgs e)
{
    string str = "明日科技 C# 编程词典";           //定义绘制的字符串
    Font myFont = new Font("华文行楷", 16);          //创建 Font 对象
    SolidBrush myBrush = new SolidBrush(Color.Black); //创建画刷对象
    Graphics myGraphics = this.CreateGraphics();      //创建 Graphics 对象
    myGraphics.DrawString(str, myFont, myBrush,20,20); //绘制文本
}

```

程序运行结果如图 16.10 所示。

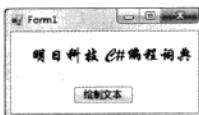


图 16.10 绘制文本

### 16.2.5 绘制图形

通过 Graphics 类中的 DrawImage 方法可以绘制图形，此方法实现在指定的坐标位置按照图像的原始物理大小绘制指定的新图像。

语法：public void DrawImage(Image image,int x,int y)

说明：image 表示要绘制的 Image；x 表示所绘制图像的左上角的 x 坐标；y 表示所绘制图像的左上角的 y 坐标。

**例 16.15** 创建一个 Windows 应用程序，通过 Graphics 类中的 DrawImage 方法在指定位置绘制一个图形，代码如下。（实例位置：光盘\mr\16\sl\16.15）

```
private void button1_Click(object sender, EventArgs e)
{
    Image myImage = Image.FromFile("明日科技.bmp");
    Graphics myGraphics = this.CreateGraphics();
    myGraphics.DrawImage(myImage, 50, 20);
}
```

程序运行结果如图 16.11 所示。



图 16.11 绘制图形

## 16.3 照猫画虎——基本功训练

### 16.3.1 基本功训练 1——绘制公章

■ 视频讲解：光盘\mr\16\lx\绘制公章.exe

■ 实例位置：光盘\mr\16\zmhh\01

新建一个 Windows 窗体应用程序，向窗体中添加一个 Button 控件，用来在窗体上绘制公章。然后在 Button 控件的 Click 事件中编写如下代码。

```
private void button1_Click(object sender, EventArgs e)
{
    int tem_Line = 0;
    int circularity_W = 4;
```

//记录圆的直径  
//设置圆画笔的粗细

```

if (panel1.Width >= panel1.Height)
    tem_Line = panel1.Height;
else
    tem_Line = panel1.Width; //设置宽度为圆的直径
Font Var_Font = new Font("Arial", 12, FontStyle.Bold); //定义字符串的字体样式
Rectangle rect = new Rectangle(10, 10, 160, 160); //实例化 Rectangle 类
//设置圆的绘制区域
rect = new Rectangle(circularity_W, circularity_W, tem_Line - circularity_W * 2, tem_Line - circularity_W * 2);
Font star_Font = new Font("Arial", 30, FontStyle.Regular); //设置星号的字体样式
string star_Str = "★";
Graphics g = this.panel1.CreateGraphics(); //实例化 Graphics 类
g.SmoothingMode = SmoothingMode.AntiAlias; //消除绘制图形的锯齿
g.Clear(Color.White); //以白色清空 panel1 控件的背景
Pen myPen = new Pen(Color.Red, circularity_W); //设置画笔的颜色
g.DrawEllipse(myPen, rect); //绘制圆
SizeF Var_Size = new SizeF(rect.Width, rect.Width); //实例化 SizeF 类
Var_Size = g.MeasureString(star_Str, star_Font); //对指定字符串进行测量
g.DrawString(star_Str, star_Font, myPen.Brush, new PointF((rect.Width / 2F) + circularity_W - Var_Size.Width / 2F, rect.Height / 2F - Var_Size.Width / 2F)); //在指定的位置绘制星号
Var_Size = g.MeasureString("专用章", Var_Font); //对指定字符串进行测量
g.DrawString("专用章", Var_Font, myPen.Brush, new PointF((rect.Width / 2F) + circularity_W - Var_Size.Width / 2F, rect.Height / 2F + Var_Size.Height * 2)); //绘制文字
string tempStr = "吉林省明日科技有限公司";
int len = tempStr.Length; //获取字符串的长度
float angle = 180 + (180 - len * 20) / 2; //设置文字的旋转角度
for (int i = 0; i < len; i++) //将文字以指定的弧度进行绘制
{
    //将指定的平移添加到 g 的变换矩阵前
    g.TranslateTransform((tem_Line + circularity_W / 2) / 2, (tem_Line + circularity_W / 2) / 2);
    g.RotateTransform(angle); //将指定的旋转用于 g 的变换矩阵
    Brush myBrush = Brushes.Red; //定义画刷
    g.DrawString(tempStr.Substring(i, 1), Var_Font, myBrush, 60, 0); //显示旋转文字
    g.ResetTransform(); //将 g 的全局变换矩阵重置为单位矩阵
    angle += 20; //设置下一个文字的角度
}
}

```

程序运行结果如图 16.12 所示。



图 16.12 绘制公章

**照猫画虎：**仿照以上程序，制作一个绘制“C#编程词典专用章”的程序。(20分)(实例位置：光盘\mr\16\zmhh\01\_zmhh)

### 16.3.2 基本功训练 2——波形图的绘制

■ 视频讲解：光盘\mr\16\lx\波形图的绘制.exe

■ 实例位置：光盘\mr\16\zmhh\02

新建一个 Windows 窗体应用程序，在窗体中添加一个 Button 控件，用来在窗体上绘制一段波形图。然后在 Button 控件的 Click 事件中编写如下代码。

```
private void button1_Click(object sender, EventArgs e)
{
    Graphics graphics = this.CreateGraphics(); //实例化窗体的 Graphics 类
    Pen myPen = new Pen(Color.Black, 1); //设置画笔
    int beginX = 50; //定义变量
    int beginY = 65;
    int height = 35;
    int width = 50;
    Point pointX1 = new Point(beginX, beginY);
    Point pointY1 = new Point(beginX + 210, beginY);
    Point pointX2 = new Point(beginX, beginY - 45);
    Point pointY2 = new Point(beginX, beginY + 45);
    //调用 DrawLine 方法绘制两条垂直相交的直线，用来作为波形图的横纵坐标
    graphics.DrawLine(myPen, pointX1, pointY1);
    graphics.DrawLine(myPen, pointX2, pointY2);
    graphics.DrawBezier(myPen, beginX, beginY, beginX + 15, beginY - height, beginX + 40, beginY - height,
    beginX + width, beginY); //绘制上半区域交错连接的贝塞尔曲线
    //绘制下半区域交错连接的贝塞尔曲线
    graphics.DrawBezier(myPen, beginX + width, beginY, beginX + width + 15, beginY + height, beginX + width +
    + 40, beginY + height, beginX + width * 2, beginY);
    //绘制上半区域交错连接的贝塞尔曲线
    graphics.DrawBezier(myPen, beginX + width * 2, beginY, beginX + width * 2 + 15, beginY - height, beginX +
    width * 2 + 40, beginY - height, beginX + width * 3, beginY);
    //绘制下半区域交错连接的贝塞尔曲线
    graphics.DrawBezier(myPen, beginX + width * 3, beginY, beginX + width * 3 + 15, beginY + height, beginX +
    width * 3 + 40, beginY + height, beginX + width * 4, beginY);
}
```

程序运行结果如图 16.13 所示。

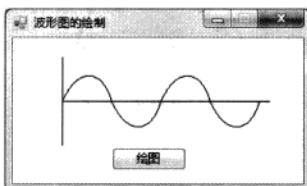


图 16.13 波形图的绘制

**照猫画虎：**将以上程序中绘制的波形图进行垂直翻转。提示：使用 Bitmap 类的 RotateFlip 方法实现。  
**(20 分)** (实例位置：光盘\mr\16\zmhh\02\_zmhh)

### 16.3.3 基本功训练 3——生成图片缩略图

视频讲解：光盘\mr\16\lx\生成图片缩略图.exe

实例位置：光盘\mr\16\zmhh\03

新建一个 Windows 窗体应用程序，在窗体中添加一个 ToolStrip 控件，用来作为窗体的工具栏；添加一个 FolderBrowserDialog 控件，用来选择图片文件夹；添加一个 ImageList 组件，用来存储生成的缩略图；添加一个 Panel 控件，用来显示生成的缩略图。生成图片缩略图的功能主要是通过自定义方法 GetReducedImage 实现的，其实现代码如下。

```
public bool GetReducedImage(double Percent, string targetFilePath)
{
    try
    {
        Bitmap bt = new Bitmap(120,120); //创建 Bitmap 实例
        Graphics g = Graphics.FromImage(bt); //创建 Graphics 实例
        g.Clear(Color.White); //设置画布背景颜色为白色
        Image ReducedImage; //缩略图
        Image.GetThumbnailImageAbort callb = new Image.GetThumbnailImageAbort(ThumbnailCallback);
        ImageWidth = Convert.ToInt32(ResourceImage.Width * Percent); //设置宽度
        ImageHeight = Convert.ToInt32(ResourceImage.Height * Percent); //设置高度
        //获取所谓图
        ReducedImage = ResourceImage.GetThumbnailImage(ImageWidth, ImageHeight, callb, IntPtr.Zero);
        if (ImageWidth > ImageHeight) //如果原图宽度大于高度
        {
            //缩放图片
            g.DrawImage(ReducedImage,0,(int)(120-ImageHeight)/2,ImageWidth,ImageHeight);
        }
        else
        {
            g.DrawImage(ReducedImage, (int)(120 - ImageWidth)/2, 0, ImageWidth, ImageHeight);
        }
        g.DrawRectangle(new Pen(Color.Gray), 0, 0, 119, 119); //绘制缩略图的边框
        bt.Save(@targetFilePath, ImageFormat.Jpeg); //保存缩略图
        bt.Dispose(); //释放对象
        ReducedImage.Dispose(); //释放对象
        return true;
    }
    catch (Exception e)
    {
        ErrMessage = e.Message;
        return false;
    }
}
```

程序运行结果如图 16.14 所示。



图 16.14 生成图片缩略图

**照猫画虎：**完善以上程序，为其增加一个状态栏，并在状态栏中显示图片的总数和选中图片的路径。(20分)(实例位置：光盘\mr\16\zmhh\03\_zmhh)

#### 16.3.4 基本功训练 4——以任意角度旋转图像

视频讲解：光盘\mr\16\lx\以任意角度旋转图像.exe

实例位置：光盘\mr\16\zmhh\04

新建一个 Windows 窗体应用程序，在窗体中添加两个 Button 控件，分别用来执行打开图像和以任意角度旋转图像操作；添加一个 PictureBox 控件，用来显示打开的图像；添加一个 Panel 控件，用来显示以任意角度旋转的图像。以任意角度旋转图像的功能是在 button1 控件的 Click 事件中实现的，其实现代码如下。

```
private void button1_Click(object sender, EventArgs e)
{
    Graphics g = this.panel1.CreateGraphics(); //实例化绘图对象
    float MyAngle = 0; //旋转的角度
    while (MyAngle < 360)
    {
        TextureBrush MyBrush = new TextureBrush(MyBitmap); //实例化 TextureBrush 类
        this.panel1.Refresh(); //使工作区无效
        MyBrush.RotateTransform(MyAngle); //以指定角度旋转图像
        g.FillRectangle(MyBrush, 0, 0, this.ClientRectangle.Width, this.ClientRectangle.Height); //绘制旋转后的图像
        MyAngle += 0.5f; //增加旋转的角度
        System.Threading.Thread.Sleep(50); //使线程休眠 50 毫秒
    }
}
```

程序运行结果如图 16.15 所示。

**照猫画虎：**制作一个程序，用来按顺时针方向以 90° 为单位旋转 JPG 图像。提示：使用 Image 类的 RotateFlip 方法实现。(20 分)(实例位置：光盘\mr\16\zmhh\04\_zmhh)



图 16.15 以任意角度旋转图像

### 16.3.5 基本功训练 5——浮雕效果显示图像

**视频讲解：**光盘\mr\16\lx\浮雕效果显示图像.exe

**实例位置：**光盘\mr\16\zmhh\05

新建一个 Windows 窗体应用程序，在窗体中添加一个 OpenFileDialog 控件，用来显示“打开文件”对话框；添加两个 Button 控件，分别用来执行打开图像和以浮雕效果显示图像的功能。以浮雕效果显示图像的功能是在 button2 控件的 Click 事件中实现的，其实现代码如下。

```
private void button2_Click(object sender, EventArgs e)
{
    try
    {
        myBitmap = new Bitmap(myImage);
        //实例化 Bitmap 类
        for (int i = 0; i < myBitmap.Width - 1; i++)
        {
            //遍历图片中的所有像素
            for (int j = 0; j < myBitmap.Height - 1; j++)
            {
                Color Color1 = myBitmap.GetPixel(i, j);
                //获取当前像素的颜色值
                Color Color2 = myBitmap.GetPixel(i + 1, j + 1);
                //获取斜点下像素的颜色值
                int red = Math.Abs(Color1.R - Color2.R + 128);
                //设置 R 色值
                int green = Math.Abs(Color1.G - Color2.G + 128);
                //设置 G 色值
                int blue = Math.Abs(Color1.B - Color2.B + 128);
                //设置 B 色值
                //颜色处理
                if (red > 255) red = 255;
                //如果 R 色值大于 255，则将 R 色值设为 255
                if (red < 0) red = 0;
                //如果 R 色值小于 0，则将 R 色值设为 0
                if (green > 255) green = 255;
                //如果 G 色值大于 255，则将 G 色值设为 255
                if (green < 0) green = 0;
                //如果 G 色值小于 0，则将 G 色值设为 0
                if (blue > 255) blue = 255;
                //如果 B 色值大于 255，则将 B 色值设为 255
                if (blue < 0) blue = 0;
                //如果 B 色值小于 0，则将 B 色值设为 0
                //为图像的像素点重新着色
                myBitmap.SetPixel(i, j, Color.FromArgb(red, green, blue));
            }
        }
        this.BackgroundImage = myBitmap;
        //显示处理后的图片
    }
}
```

```
catch {}  
}
```

程序运行结果如图 16.16 所示。



图 16.16 浮雕效果显示图像

**照猫画虎：**根据以上程序的实现原理，制作一个以底片效果显示图像的程序。(20 分)(实例位置：光盘\mr\16\zmhh\05\_zmhh )

照猫画虎栏目分数统计：

照猫画虎题目	1	2	3	4	5	总分数	
分数							

## 16.4 情景应用——拓展与实践

### 16.4.1 情景应用 1——绘制中文验证码

视频讲解：光盘\mr\16\lx\绘制中文验证码.exe

实例位置：光盘\mr\16\qjyy\01

上网浏览网页时，有些网站提供会员注册或发布帖子的功能。在注册或发布帖子时，很多都是只有输入正确验证码后，才能提交数据。验证码主要用于防止恶意提交数据，验证码大体可分为数字验证码、数字与字母混合、字母验证码及中文验证码。本实例可以随机生成中文验证码，并且可以验证输入的验证码是否正确。运行效果如图 16.17 所示。



图 16.17 绘制中文验证码

实现过程如下。

- (1) 创建一个 Windows 窗体应用程序，并将其命名为 ChineseCode。
- (2) 更改默认窗体 Form1 的 Name 属性为 Frm\_Main，在该窗体中添加一个 PictureBox 控件，用来显示生成的中文验证码；添加一个 TextBox 控件，用来输入验证码；添加两个 Button 控件，分别用来生成验证码和判断输入的验证码是否正确。
- (3) 绘制中文验证码的功能主要是通过自定义方法 CreateImage 实现的，该方法代码如下。

```
private void CreateImage()  
{
```

```

Encoding gb = Encoding.GetEncoding("gb2312");
object[] bytes = CreateCode(4);
//根据汉字编码的字节数组解码出中文汉字
string str1 = gb.GetString((byte[]))Convert.ChangeType(bytes[0], typeof(byte[]));
string str2 = gb.GetString((byte[]))Convert.ChangeType(bytes[1], typeof(byte[]));
string str3 = gb.GetString((byte[]))Convert.ChangeType(bytes[2], typeof(byte[]));
string str4 = gb.GetString((byte[]))Convert.ChangeType(bytes[3], typeof(byte[]));
txt = str1 + str2 + str3 + str4; //获取随机生成的 4 个汉字
if (txt == null || txt == String.Empty) //如果没有汉字
{
    return; //返回
}
Bitmap image = new Bitmap((int)Math.Ceiling((txt.Length * 21.5)), 22); //创建 Bitmap 实例用于绘制验证码
Graphics g = Graphics.FromImage(image); //创建 Graphics 对象
try
{
    Random random = new Random(); //生成随机生成器
    g.Clear(Color.White); //清空图片背景色
    for (int i = 0; i < 2; i++) //画图片的背景噪音线
    {
        Point tem_Point_1 = new Point(random.Next(image.Width), random.Next(image.Height));
        Point tem_Point_2 = new Point(random.Next(image.Width), random.Next(image.Height));
        g.DrawLine(new Pen(Color.Black), tem_Point_1, tem_Point_2);
    }
    Font font = new Font("宋体", 12, (FontStyle.Bold)); //画图片的前景噪音点
    LinearGradientBrush brush = new LinearGradientBrush(new Rectangle(0, 0, image.Width, image.Height),
    Color.Blue, Color.DarkRed, 1.2f, true);
    g.DrawString(txt, font, brush, 2, 2);
    for (int i = 0; i < 100; i++) //画图片的前景噪音点
    {
        Point tem_point = new Point(random.Next(image.Width), random.Next(image.Height));
        image.SetPixel(tem_point.X, tem_point.Y, Color.FromArgb(random.Next()));
    }
    //画图片的边框线
    g.DrawRectangle(new Pen(Color.Silver), 0, 0, image.Width - 1, image.Height - 1);
    pictureBox1.Image = image; //显示生成的中文验证码
}
catch {}
}

```

**DIY：**根据以上程序的实现原理制作一个程序，用来生成数字和字母混合的图形验证码。(20 分)(实例位置：光盘\mr\16\qjyy\01\_diy)

## 16.4.2 情景应用 2——批量图像格式转换

视频讲解：光盘\mr\16\lx\批量图像格式转换.exe

实例位置：光盘\mr\16\qjyy\02

计算机支持的图像格式有很多种，但是如果想将某种格式转换成另一种格式就需要用到图像格式转换工具。由于需要转换的图片可能会有很多，所以为了提高转换效率，开发出批量图像格式转换工具，通过

本实例可以将选择的多个图片转换成指定的图像格式。运行效果如图 16.18 所示。

实现过程如下。

(1) 创建一个 Windows 窗体应用程序，并将其命名为 PictureBatchConversion。

(2) 更改默认窗体 Form1 的 Name 属性为 Frm\_Main。在窗体中添加一个 ToolStrip 控件，用来作为窗体的工具栏；添加一个 ListView 控件，用来显示待转换的图片信息及转换的状态；添加一个 OpenFileDialog 控件，用来选择待转换的图片；添加一个 FolderBrowserDialog 控件，用来选择保存路径；添加一个 StatusStrip 控件，用来显示状态信息。

(3) 在 Frm\_Main 窗体的后台代码中创建一个 ConvertImage 方法用于进行图像格式转换。在该方法中使用 switch 语句判断要转换的类型，如果类型 Imgtype1 为 0，则将图像转换为 BMP 格式，代码如下。

```
case 0:
    for (int i = 0; i < path1.Length; i++)
    {
        string ImgName = path1[i].Substring(path1[i].LastIndexOf("\\") + 1,
path1[i].Length - path1[i].LastIndexOf("\\") - 1);
        ImgName = ImgName.Remove(ImgName.LastIndexOf("."));
        OlePath = path1[i].ToString();
        bt = new Bitmap(OlePath);
        path = path2 + "\\" + ImgName + ".bmp";
        bt.Save(path, System.Drawing.Imaging.ImageFormat.Bmp);
        listView1.Items[flags - 1].SubItems[6].Text = "已转换";
        tsslPlan.Text = "正在转换" + flags * 100 / path1.Length + "%";
        if (flags == path1.Length)
        {
            toolStrip1.Enabled = true;
            tsslPlan.Text = "图片转换全部完成";
        }
        flags++;
    }
    break;
```

图 16.18 批量图像格式转换

//如果选择第一项，则转换为 BMP 格式  
//遍历图片集合

//获取图片名称（带扩展名）  
//获取图片名称（不带扩展名）  
//获取图片所在路径  
//创建 Bitmap 对象  
//设置保存路径  
//保存  
//显示转换的状态  
//显示转换的进度  
//如果转换的数量等于图片总数量

(4) 如果类型 Imgtype1 为 1，则将图像转换为 JPEG 格式，代码如下。

```
case 1:
    for (int i = 0; i < path1.Length; i++)
    {
        string ImgName = path1[i].Substring(path1[i].LastIndexOf("\\") + 1,
path1[i].Length - path1[i].LastIndexOf("\\") - 1);
        ImgName = ImgName.Remove(ImgName.LastIndexOf("."));
        OlePath = path1[i].ToString();
        bt = new Bitmap(OlePath);
        path = path2 + "\\" + ImgName + ".jpeg";
        bt.Save(path, System.Drawing.Imaging.ImageFormat.Jpeg);
        listView1.Items[flags - 1].SubItems[6].Text = "已转换";
        tsslPlan.Text = "正在转换" + flags * 100 / path1.Length + "%";
        if (flags == path1.Length)
```

//如果选择第二项，则转换为 JPEG 格式  
//遍历图片集合

//获取图片名称（带扩展名）  
//获取图片名称（不带扩展名）  
//获取图片所在路径  
//创建 Bitmap 对象  
//设置保存路径  
//保存  
//显示转换的状态  
//显示转换的进度  
//如果转换的数量等于图片总数量



```

    {
        toolStrip1.Enabled = true;
        tsslPlan.Text = "图片转换全部完成"; //提示转换完成
    }
    flags++;
}
break;
(5) 如果类型 Imgtype1 为 2，则将图像转换为 PNG 格式，代码如下。
case 2: //如果选择第三项，则转换成 PNG 格式
for (int i = 0; i < path1.Length; i++)
{
    string ImgName = path1[i].Substring(path1[i].LastIndexOf("\\")) + 1,
path1[i].Length - path1[i].LastIndexOf("\\")) - 1); //获取图片名称（带扩展名）
ImgName = ImgName.Remove(ImgName.LastIndexOf(".")); //获取图片名称（不带扩展名）
OlePath = path1[i].ToString(); //获取图片所在路径
bt = new Bitmap(OlePath); //创建 Bitmap 对象
path = path2 + "\\" + ImgName + ".png"; //设置保存路径
bt.Save(path, System.Drawing.Imaging.ImageFormat.Png); //保存
listView1.Items[flags - 1].SubItems[6].Text = "已转换"; //显示转换的状态
tsslPlan.Text = "正在转换" + flags * 100 / path1.Length + "%"; //显示转换的进度
if (flags == path1.Length) //如果转换的数量等于图片总数量
{
    toolStrip1.Enabled = true;
    tsslPlan.Text = "图片转换全部完成"; //提示转换完成
}
flags++;
}
break;
(6) 如果类型 Imgtype1 为 3，则将图像转换为 GIF 格式，代码如下。
case 3: //如果选择第四项，则转换成 GIF 格式
for (int i = 0; i < path1.Length; i++) //遍历图片集合
{
    string ImgName = path1[i].Substring(path1[i].LastIndexOf("\\")) + 1,
path1[i].Length - path1[i].LastIndexOf("\\")) - 1); //获取图片名称（带扩展名）
ImgName = ImgName.Remove(ImgName.LastIndexOf(".")); //获取图片名称（不带扩展名）
OlePath = path1[i].ToString(); //获取图片所在路径
bt = new Bitmap(OlePath); //创建 Bitmap 对象
path = path2 + "\\" + ImgName + ".gif"; //设置保存路径
bt.Save(path, System.Drawing.Imaging.ImageFormat.Gif); //保存
listView1.Items[flags - 1].SubItems[6].Text = "已转换"; //显示转换的状态
tsslPlan.Text = "正在转换" + flags * 100 / path1.Length + "%"; //显示转换的进度
if (flags == path1.Length)
{
    toolStrip1.Enabled = true;
    tsslPlan.Text = "图片转换全部完成"; //提示转换完成
}
flags++;
}
break;
}

```

**书说明：**在进行图像格式转换时，如果原图是 GIF 动画，则转换成其他格式的图片后只能保留一帧。而如果将其他格式的图片转换成 GIF 格式也只能是静态的图片。

**DIY：**根据以上程序的实现原理制作一个程序，主要实现 GIF 动画和 JPG 图片之间的相互转换。(20 分)(实例位置：光盘\mr\16\qjyy\02\_diy)

### 16.4.3 情景应用 3——抓取网站整页面

**视频讲解：**光盘\mr\16\lx\抓取网站整页面.exe

**实例位置：**光盘\mr\16\qjyy\03

当浏览网站时，如果想保存网页，通常选择浏览器中“文件”/“另存为”命令将网页保存到本地磁盘中。但有时，网页不允许用这样的方法保存，所以就需要将整个网页内容抓取成图片保存起来，从而实现保存网站内容的功能，这里使用 C#实现了抓取网站整页面的功能。运行效果如图 16.19 所示。

实现过程如下。

(1) 创建一个 Windows 窗体应用程序，并将其命名为 WebSnap。

(2) 更改默认窗体 Form1 的 Name 属性为 Frm\_Main。在窗体中添加一个 ToolStrip 控件，用来作为窗体的工具栏；添加一个 SaveFileDialog 控件，用来选择保存位置；添加一个 WebBrowser 控件，用来显示加载的网页；添加一个 StatusStrip 控件，用来作为窗体的状态栏。

(3) 抓取网站整页面的功能是在自定义方法 Snapweb 中实现的，其代码如下。

```
private void Snapweb()
{
    this.TopMost = true;
    SelectObject(DestDC, Bhandle);
    int linewidth=0;
    int lineheight=0;
    int i=0;
    int j=0;
    winpoint = webBrowser1.Handle;
    GetWindowRect(winpoint,ref Rectangles);
    while (lineheight < webBrowser1.Document.Body.ScrollRectangle.Height - 199)
    {
        if (webBrowser1.Document.Body.ScrollTop == 0)
        {
            inputHide();
            //获取可见区域场景
            BitBlt(DestDC, 0, 0, webBrowser1.Document.Body.ClientRectangle.Width, webBrowser1.Document.
Body.ClientRectangle.Height, SourceDC, Rectangles.Left, Rectangles.Top, 13369376);
            linewidth = webBrowser1.Document.Body.ClientRectangle.Width+5; //赋值宽
            //窗体处于所有窗体的最前段
            //选择场景对象
            //宽
            //高
            //纵向计数器
            //横向计数器
            // webBrowser1 句柄
            //获得整个窗口的范围矩形
            //屏蔽输入法
        }
    }
}
```

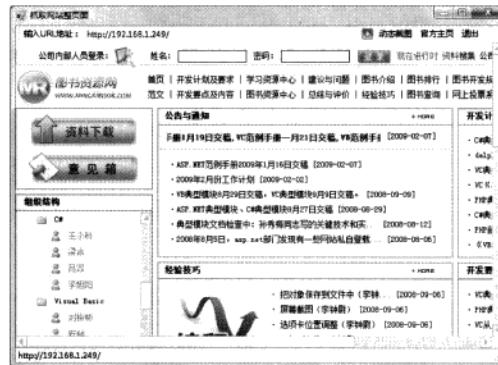


图 16.19 抓取网站整页面

```

if (linewidth < webBrowser1.Document.Body.ScrollRectangle.Width) //当前宽小于网页宽度时
{
    //获取整个宽度场景
    while (linewidth < webBrowser1.Document.Body.ScrollRectangle.Width - 199)
    {
        inputHide();
        webBrowser1.Document.Body.ScrollLeft += 199;
        BitBlt(DestDC, webBrowser1.Document.Body.ClientRectangle.Width + 199 * j, 0, 199,
webBrowser1.Document.Body.ClientRectangle.Height, SourceDC, Rectangles.Left + webBrowser1.Document.
Body.ClientRectangle.Width - 199, Rectangles.Top, 13369376);
        linewidth = linewidth + 199;
        j = j + 1;
    }
    if (linewidth >= webBrowser1.Document.Body.ScrollRectangle.Width - 199)
    {
        inputHide();
        webBrowser1.Document.Body.ScrollLeft += 199;
        BitBlt(DestDC, webBrowser1.Document.Body.ClientRectangle.Width + 199 * j, 0, webBrowser1.
Document.Body.ScrollRectangle.Width - linewidth, webBrowser1.Document.Body.ClientRectangle.Height, SourceDC,
Rectangles.Left + webBrowser1.Document.Body.ClientRectangle.Width - (webBrowser1.Document.Body.
ScrollRectangle.Width - linewidth), Rectangles.Top, 13369376);
        webBrowser1.Document.Body.ScrollLeft= 0;
        j = 0;
    }
}
lineheight = lineheight + webBrowser1.Document.Body.ClientRectangle.Height; //累计当前高度
}
else
{
    inputHide();
    BitBlt(DestDC, 0, webBrowser1.Document.Body.ClientRectangle.Height + 199 * i, webBrowser1.
Document.Body.ClientRectangle.Width, 199, SourceDC, Rectangles.Left, Rectangles.Top + webBrowser1.
Document.Body.ClientRectangle.Height - 199, 13369376); //获取可见区域场景
    linewidth = webBrowser1.Document.Body.ClientRectangle.Width; //当前宽度
    if (linewidth < webBrowser1.Document.Body.ScrollRectangle.Width)
    {
        while (linewidth < webBrowser1.Document.Body.ScrollRectangle.Width-199)
        {
            inputHide();
            webBrowser1.Document.Body.ScrollLeft += 199;
            BitBlt(DestDC, webBrowser1.Document.Body.ClientRectangle.Width + 199 * j, webBrowser1.
Document.Body.ClientRectangle.Height + 199 * i, webBrowser1.Document.Body.ClientRectangle.Width, 199,
SourceDC, Rectangles.Left + webBrowser1.Document.Body.ClientRectangle.Width - 199, Rectangles.Top +
webBrowser1.Document.Body.ClientRectangle.Height - 199, 13369376);
            linewidth = linewidth + 199;
            j = j + 1;
        }
        if(linewidth >=webBrowser1.Document.Body.ScrollRectangle.Width-199)
        {
            inputHide();
            webBrowser1.Document.Body.ScrollLeft+=199;
        }
    }
}

```

```

        BitBlt(DestDC, webBrowser1.Document.Body.ClientRectangle.Width + 199 * j, webBrowser1.
Document.Body.ClientRectangle.Height + 199 * i, webBrowser1.Document.Body.ScrollRectangle.Width - linewidth,
199, SourceDC, Rectangles.Left + webBrowser1.Document.Body.ClientRectangle.Width - (webBrowser1.Document.
Body.ScrollRectangle.Width - linewidth), Rectangles.Top + webBrowser1.Document.Body.ClientRectangle.Height -
199, 13369376);
        webBrowser1.Document.Body.ScrollLeft=0;
        j = 0;
    }
}
i = i + 1;
lineheight = lineheight + 199;
}
webBrowser1.Document.Body.ScrollTop+=199; //调整纵向滚动条位置
}
if (lineheight >= webBrowser1.Document.Body.ScrollRectangle.Height - 199)
{
    inputHide();
    BitBlt(DestDC, 0, webBrowser1.Document.Body.ClientRectangle.Height + 199 * i, webBrowser1.Document.
Body.ClientRectangle.Width, (webBrowser1.Document.Body.ScrollRectangle.Height - lineheight), SourceDC,
Rectangles.Left, Rectangles.Top + webBrowser1.Document.Body.ClientRectangle.Height - (webBrowser1.Document.
Body.ScrollRectangle.Height - lineheight), 13369376);
    linewidth = webBrowser1.Document.Body.ClientRectangle.Width; //当前宽度
    if(linewidth < webBrowser1.Document.Body.ScrollRectangle.Width) //当前宽度小于网页宽度时
    {
        while(linewidth<webBrowser1.Document.Body.ScrollRectangle.Width-199) //获取整个宽度场景
        {
            inputHide();
            webBrowser1.Document.Body.ScrollLeft+=199;
            BitBlt(DestDC, webBrowser1.Document.Body.ClientRectangle.Width + 199 * j, webBrowser1.
Document.Body.ClientRectangle.Height + 199 * i, webBrowser1.Document.Body.ClientRectangle.Width,
(webBrowser1.Document.Body.ScrollRectangle.Height - lineheight), SourceDC, Rectangles.Left + webBrowser1.
Document.Body.ClientRectangle.Width - 199, Rectangles.Top + webBrowser1.Document.Body.ClientRectangle.
Height - (webBrowser1.Document.Body.ScrollRectangle.Height - lineheight), 13369376);
            linewidth = linewidth + 199; //累计当前高度
            j = j + 1; //纵向计数器累计
        }
        if(linewidth >=webBrowser1.Document.Body.ScrollRectangle.Width-199)
        {
            inputHide();
            webBrowser1.Document.Body.ScrollLeft+=199; //调整纵向滚动条位置
            BitBlt(DestDC, webBrowser1.Document.Body.ClientRectangle.Width + 199 * i, webBrowser1.Document.
Body.ClientRectangle.Height + 199 * i, webBrowser1.Document.Body.ScrollRectangle.Width - linewidth,
(webBrowser1.Document.Body.ScrollRectangle.Height - lineheight), SourceDC, Rectangles.Left + webBrowser1.
Document.Body.ClientRectangle.Width - (webBrowser1.Document.Body.ScrollRectangle.Width - linewidth),
Rectangles.Top + webBrowser1.Document.Body.ClientRectangle.Height - (webBrowser1.Document.Body.
ScrollRectangle.Height - lineheight), 13369376);
            webBrowser1.Document.Body.ScrollLeft=0;
        }
    }
    i=0;
}

```

```
OpenClipboard(this.Handle);
EmptyClipboard();
SetClipboardData(2,Bhandle);
CloseClipboard();
Image ig = (Image)Clipboard.GetImage();
if (saveFileDialog1.ShowDialog() == DialogResult.OK)
{
    string path = saveFileDialog1.FileName;
    ig.Save(path);
    MessageBox.Show("保存成功","提示",MessageBoxButtons.OK,MessageBoxIcon.Exclamation);
    webBrowser1.Document.Body.ScrollTop = 0;
    ig.Dispose();
}
}
```

**DIY:** 制作一个程序，主要用来实现屏幕抓图功能，在抓图时，可以设置是否抓取鼠标，并且需要提供快捷键，方便使用者抓取屏幕。提示：主要使用系统 API 函数 GetSystemMetrics、CopyIcon 和 Graphics 类的 CopyFromScreen 方法实现。（20 分）（实例位置：光盘\mr\16\ai\03\div）

#### 16.4.4 情景应用 4——批量添加图片水印

 视频讲解: 光盘\mr\16\x\批量添加图片水印.exe

 实例位置：光盘\mr\16\qjvv\04

如今网络已经非常普及，很多人会选择网上购物。在网上开店的店主们会精心地处理自己的商品图片，并且还会在图片上加上水印，突出自己的店铺风格，实现广告效应。如果想在多个图片上添加同一个水印，那么就会变得非常复杂，所以本实例实现批量添加水印的功能，通过本实例可以为多个图片添加文字或图片水印。运行效果如图 16.20 所示。

实现过程如下。

(1) 创建一个 Windows 窗体应用程序，并将其命名为 IMGwatermark。

(2) 更改默认窗体 Form1 的 Name 属性为 FrmMain。在该窗体中添加一个 ListBox 控件，用来显示图片列表；添加两个 OpenFileDialog 控件，分别用来

选择所有图片和水印图片；添加一个 `ColorDialog` 控件，用来选择字体颜色；添加一个 `FontDialog` 控件，用来设置字体；添加一个 `FolderBrowserDialog` 控件，用来选择保存位置；添加一个 `PictureBox` 控件，用来显示添加水印的效果；添加一个 `TrackBar` 控件，用来设置水印图片透明度；添加一个 `ComboBox` 控件，用来选择添加水印的位置；添加 7 个 `Button` 控件，分别用来执行选择字体、选择水印图片、选择保存位置、加载图片、预览水印效果、添加水印效果和退出程序的功能。

(3) 为图片添加水印的功能是在自定义方法 AddFontWatermark 中实现的，其代码如下。

```
private void AddFontWatermark(string txt, string Iname, int i)
```



图 16.20 批量添加图片水印

```

b = new SolidBrush(fontColor); //创建 SolidBrush 对象
bt = new Bitmap(368,75); //创建 Bitmap 对象，设置水印图片大小
BigBt = new Bitmap(Image.FromFile(imgDirectoryPath + "\\\" + lName)); //通过图片路径创建 Bitmap 对象
Graphics g = Graphics.FromImage(bt); //创建 Graphics 对象
Graphics g1 = Graphics.FromImage(BigBt); //创建 Graphics 对象
g.Clear(Color.Gainsboro); //设置画布背景颜色
pbImgPreview.Image = bt; //设置 Image 属性
if (FontF == null) //判断是否设置字体
{
    f = new Font(txt,fontSize); //创建字体对象
    SizeF XMaxSize = g.MeasureString(txt, f); //获取字体的大小
    Fwidth = (int)XMaxSize.Width; //字体宽度
    Fheight = (int)XMaxSize.Height; //字体高度
    g.DrawString(txt,f,b,(int)(368 - Fwidth) / 2,(int)(75 - Fheight) / 2); //绘制文字水印
    if (cbbPosition.SelectedIndex==0) //正中
    {
        g1.DrawString(txt,f,b,(int)(BigBt.Width - Fwidth) / 2,(int)(BigBt.Height - Fheight) / 2);
    }
    if (cbbPosition.SelectedIndex == 1) //左上
    {
        g1.DrawString(txt,f,b,30,30);
    }
    if (cbbPosition.SelectedIndex == 2) //左下
    {
        g1.DrawString(txt,f,b,30,(int)(BigBt.Height - Fheight)-30);
    }
    if (cbbPosition.SelectedIndex == 3) //右上
    {
        g1.DrawString(txt,f,b,(int)(BigBt.Width - Fwidth), 30);
    }
    if (cbbPosition.SelectedIndex == 4) //右下
    {
        g1.DrawString(txt,f,b,(int)(BigBt.Width - Fwidth),(int)(BigBt.Height - Fheight)-30);
    }
}
else //如果设置了字体
{
    f = new Font(FontF,fontSize,fontStyle); //创建字体对象
    SizeF XMaxSize = g.MeasureString(txt, f); //获取字体的大小
    Fwidth = (int)XMaxSize.Width; //字体宽度
    Fheight = (int)XMaxSize.Height; //字体高度
    //绘制文字水印
    g.DrawString(txt,new Font(FontF,fontSize,fontStyle),b,(int)(368 - Fwidth) / 2,(int)(75 - Fheight) / 2);
    if (cbbPosition.SelectedIndex == 0) //正中
    {
        g1.DrawString(txt,new Font(FontF,fontSize,fontStyle),b,(int)(BigBt.Width - Fwidth) / 2,
        (int)(BigBt.Height - Fheight) / 2);
    }
    if (cbbPosition.SelectedIndex == 1) //左上
    {
        g1.DrawString(txt,new Font(FontF,fontSize,fontStyle),b,30,30);
    }
}

```

```

        }
        if (cbbPosition.SelectedIndex == 2) //左下
        {
            g1.DrawString(txt, new Font(FontF, fontSize, fontStyle), b, 30, (int)(BigBt.Height - Fheight)-30);
        }
        if (cbbPosition.SelectedIndex == 3) //右上
        {
            g1.DrawString(txt, new Font(FontF, fontSize, fontStyle), b, (int)(BigBt.Width - Fwidth), Fheight);
        }
        if (cbbPosition.SelectedIndex == 4) //右下
        {
            g1.DrawString(txt, new Font(FontF, fontSize, fontStyle), b,
                (int)(BigBt.Width - Fwidth), (int)(BigBt.Height - Fheight)-30);
        }
    }
    if (i == 1) //当 i 为 1 时，保存绘制了文字谁赢的图片
    {
        string ipath; //加水印后图片路径
        if (NewFolderPath.Length == 3) //判断是否为系统磁盘
            //如果是系统磁盘，则删除 “:” 后边的字符
            ipath = NewFolderPath.Remove(NewFolderPath.LastIndexOf(":") + 1);
        else //否则
            ipath = NewFolderPath; //获取路径
        //获取图片类型
        string imgstype = Iname.Substring(Iname.LastIndexOf(".") + 1, Iname.Length - 1 - Iname.LastIndexOf("."));
        if (imgstype.ToLower() == "jpeg" || imgstype.ToLower() == "jpg") //如果原图是 jpeg 格式
        {
            BigBt.Save(ipath + "\\_" + Iname, ImageFormat.Jpeg); //加水印后依然保存成 jpeg 格式
        }
        if (imgstype.ToLower() == "png") //如果原图是 png 格式
        {
            BigBt.Save(ipath + "\\_" + Iname, ImageFormat.Png); //加水印后依然保存成 png 格式
        }
        if (imgstype.ToLower() == "bmp") //如果原图是 bmp 格式
        {
            BigBt.Save(ipath + "\\_" + Iname, ImageFormat.Bmp); //加水印后依然保存成 bmp 格式
        }
        if (imgstype.ToLower() == "gif") //如果原图是 gif 格式
        {
            BigBt.Save(ipath + "\\_" + Iname, ImageFormat.Gif); //加水印后依然保存成 gif 格式
        }
        g1.Dispose();
        BigBt.Dispose();
    }
    if (i == 2) //加完图片水印后预览
    {
        if (cbbPosition.SelectedIndex == 0) //正中
        {
            g1.DrawImage(effect, (int)(BigBt.Width - effect.Width) / 2, (int)(BigBt.Height - effect.Height) / 2);
        }
    }
}

```

```

        }
        if (cbbPosition.SelectedIndex == 1) //左上
        {
            g1.DrawImage(effect, 30, 30);
        }
        if (cbbPosition.SelectedIndex == 2) //左下
        {
            g1.DrawImage(effect, 30, (int)(BigBt.Height - effect.Height) - 30);
        }
        if (cbbPosition.SelectedIndex == 3) //右上
        {
            g1.DrawImage(effect, (int)(BigBt.Width - effect.Width) - 30, 30);
        }
        if (cbbPosition.SelectedIndex == 4) //右下
        {
            g1.DrawImage(effect, (int)(BigBt.Width - effect.Width) - 30, (int)(BigBt.Height - effect.Height) - 30);
        }
    }
    if (i == 3) //加完图片水印后保存
    {
        if (cbbPosition.SelectedIndex == 0) //正中
        {
            g1.DrawImage(effect, (int)(BigBt.Width - effect.Width) / 2, (int)(BigBt.Height - effect.Height) / 2);
        }
        if (cbbPosition.SelectedIndex == 1) //左上
        {
            g1.DrawImage(effect, 30, 30);
        }
        if (cbbPosition.SelectedIndex == 2) //左下
        {
            g1.DrawImage(effect, 30, (int)(BigBt.Height - effect.Height) - 30);
        }
        if (cbbPosition.SelectedIndex == 3) //右上
        {
            g1.DrawImage(effect, (int)(BigBt.Width - effect.Width), 30);
        }
        if (cbbPosition.SelectedIndex == 4) //右下
        {
            g1.DrawImage(effect, (int)(BigBt.Width - effect.Width), (int)(BigBt.Height - effect.Height) - 30);
        }
        string ipath;
        if (NewFolderPath.Length == 3)
            ipath = NewFolderPath.Remove(NewFolderPath.LastIndexOf(":") + 1);
        else
            ipath = NewFolderPath;
        string imgstype = Iname.Substring(Iname.LastIndexOf(".") + 1, Iname.Length - 1 - Iname.LastIndexOf("."));
        if (imgstype.ToLower() == "jpeg" || imgstype.ToLower() == "jpg") //如果原图是 jpeg 格式
        {
            BigBt.Save(ipath + "\\_" + Iname, ImageFormat.Jpeg); //加水印后依然保存成 jpeg 格式
        }
    }
}

```

```
if (imgstype.ToLower() == "png") //如果原图是 png 格式
{
    BigBt.Save(ipath + "\\_" + Iname, ImageFormat.Png);
}
if (imgstype.ToLower() == "bmp") //如果原图是 bmp 格式
{
    BigBt.Save(ipath + "\\_" + Iname, ImageFormat.Bmp);
}
if (imgstype.ToLower() == "gif") //如果原图是 gif 格式
{
    BigBt.Save(ipath + "\\_" + Iname, ImageFormat.Gif);
}
```

**DIY:** 制作一个程序，主要实现为数码照片添加日期的功能。提示：首先使用 DrawString 方法在数码照片上绘制日期，然后使用 Bitmap 类的 Save 方法保存数码照片。(20 分)(实例位置：光盘\mr\16\qjyy\04\_diy)

#### 16.4.5 情景应用 5——打造自己的开心农场

视频讲解：光盘\mr\16\lx\打造自己的开心农场.exe

 实例位置：光盘\mr\16\qjyy\05

在一个完善的客户管理系统中，图像数据的存取是必不可少的，如用户头像等信息，这里使用 C#实现使用二进制存取用户头像的功能。运行效果如图 16.21 所示。

实现过程如下。

(1) 创建一个 Windows 窗体应用程序，并将其命名为 ChuffedFarm。

(2) 更改默认窗体 Form1 的 Name 属性为 Frm\_Main，在该窗体中添加 5 个 PictureBox 控件，分别用来实现播种、生长、开花、结果和收获功能；添加一个 Label 控件，用来显示仓库中果实的信息；添加一个 ToolTip 控件，用来作为图片控件的信息提示助手。

(3) 在 Frm\_Main 窗体中单击“播种”图片控件，将触发该图片控件的 Click 事件。在该事件中，程序首先创建一个 CPictureBox 图片控件，用来显示种子图片，然后设置该图片控件的相关属性。“播种”图片控件的 Click 事件代码如下。

```
private void pbxInseminate_Click(object sender, EventArgs e)
{
    if (PlantState != PlantState.Nothing && PlantState != PlantState.Harvest && PlantState !=
        PlantState.Inseminate) //若农场中存在未收获的农作物
    {
        MessageBox.Show("还未收获，无法播种!");
        return;
    }
}
```



图 16.21 打造自己的开心农场

```

}
this.PlantState = PlantState.Inseminate; //设置农作物处于播种状态
this.cpbxSeed = new CPictureBox(); //创建 CPictureBox 图片控件
//设置图片控件的背景颜色为透明色
this.cpbxSeed.BackColor = System.Drawing.Color.Transparent;
//设置图片控件背景图像的布局, 这里设置图像将沿图片控件的矩形工作区拉伸
this.cpbxSeed.BackgroundImageLayout = System.Windows.Forms.ImageLayout.Stretch;
//设置 Image 属性, 让图片控件显示种子图片
this.cpbxSeed.Image = ChuffedFarm.Properties.Resources.seed2;
this.cpbxSeed.Size = new System.Drawing.Size(ChuffedFarm.Properties.Resources.seed2.Width,
    ChuffedFarm.Properties.Resources.seed2.Height); //设置图片控件大小
//设置图片控件位置
this.cpbxSeed.Location = new
    System.Drawing.Point(this.pbxInseminate.Location.X-50,this.pbxInseminate.Location.Y - 80);
this.cpbxSeed.TabStop = true; //设置图片控件接收 Tab 键的焦点
this.cpbxSeed.IsInseminate = false; //表示种子还未种下
//把图片控件的 Click 事件绑定到 cpbxSeed_Click 方法
this.cpbxSeed.Click += new System.EventHandler(this.cpbxSeed_Click);
this.Controls.Add(this.cpbxSeed); //把图片控件添加到窗体中
tipSeed.SetToolTip(this.cpbxSeed, "这是种子"); //为图片控件设置提示助手
}

```

 技巧：当把一个图片控件放置到其他控件的上面时，若想使该图片控件不遮住它下面控件的背景颜色，可以设置该图片控件的 BackColor 属性值为 Transparent（透明色）。

(4) 单击“播种”图片控件之后，在窗体中移动鼠标指针，会发现生成的图片控件随着鼠标指针移动。鼠标指针在窗体中移动时，会触发窗体的 MouseMove 事件，在该事件中设置了图片控件的 Location 属性值为鼠标指针当前的位置。窗体的 MouseMove 事件代码如下。

```

private void Form1_MouseMove(object sender, MouseEventArgs e)
{
    if (this.cpbxSeed != null) //若已生成 CPictureBox 控件实例
    {
        if (this.Bounds.Contains(Cursor.Position)) //若光标在窗体内
        {
            if (this.cpbxSeed.IsInseminate == false) //若种子还未种下
            {
                if (this.PlantState == PlantState.Inseminate) //若农作物处于种植状态
                {
                    //设置图片控件的 Location 属性值为鼠标指针当前位置
                    this.cpbxSeed.Location = new Point(e.X-20 , e.Y-40 );
                }
            }
        }
    }
}

```

(5) 在种子种下之后，可按照农作物的生长顺序依次单击各个图片按钮，农场中的农作物会呈现不同的外观状态。当农作物结果之后，单击“收获”图片控件，程序将清空农场中的所有农作物，并记录下收获的果实数量。单击“收获”图片控件，触发该控件的 Click 事件，代码如下。

```

private void pbxHarvest_Click(object sender, EventArgs e)
{
    if (PlantState == PlantState.Nothing) //若还未播种
    {
        MessageBox.Show("还未播种，能收获吗？", "信息提示！");
        return;
    }
    if (PlantState == PlantState.Inseminate) //若刚播完种子
    {
        MessageBox.Show("刚刚播完种，还未生长，能收获吗？", "信息提示！");
        return;
    }
    if (PlantState == PlantState.Vegetate) //若农作物还处在生长期
    {
        MessageBox.Show("还处在生长期，并未开花，能收获吗？", "信息提示！");
        return;
    }
    if (PlantState == PlantState.BlossomOut) //若农作物还处在开花期
    {
        MessageBox.Show("正在花期，并未结果，能收获吗？", "信息提示！");
        return;
    }
    if (PlantState == PlantState.Harvest) //若农作物已被收获完毕
    {
        MessageBox.Show("都已经收过了？", "信息提示！");
        return;
    }
    IEnumerable<Control> cons = this.GetCPictureBoxes(); //获取所有的 CPictureBox 控件实例
    foreach (CPictureBox cpbx in cons)
    {
        intAmount++; //记录收获果实的数量
        cpbx.Dispose(); //销毁当前的 CPictureBox 控件实例
    }
    if (cons.Count<Control>() > 0) //若图片控件集合中还有元素
    {
        pbxHarvest_Click(sender, e); //递归调用 pbxHarvest_Click 方法
    }
    this.PlantState = PlantState.Harvest; //设置农作物处于收获完毕状态
    lbAmount.Text = "你的仓库里有" + intAmount.ToString() + "个果实！";
}

```

**DIY：**完善以上程序，使农场中的农作物能够根据播种时间的差异而呈现出多种生长状态，如有的作物在生长、有的在开花、有的已经结果。提示：使用多线程技术实现。（20 分）（实例位置：光盘\mr\16\qjyy\05\_diy）

情景应用 DIY 栏目分数统计：

DIY 题目	1	2	3	4	5	总分数	
分数							

## 16.5 自我测试

### 一、选择题（每题 10 分，5 道题）

1. 下面对于画笔的描述中，不正确的是（ ）。  
 A. 画笔用于绘制图形的框线  
 B. 画笔不但可以绘制实线，而且还可以绘制点划线和虚线等  
 C. StartCap 和 EndCap 属性可以设置线段的端点样式  
 D. 通过设置 DashCap 属性，可以自定义线段的端点样式
2. 下面有关字体的描述中，不正确的是（ ）。  
 A. 在 GDI+ 中，无法获取当前系统中安装了哪些字体  
 B. 字体的大小是有单位的，单位可以是点、英寸或毫米  
 C. 字体的样式取值于 FontStyle 枚举  
 D. 可以通过指定字体的名称来建立字体
3. 下面有关画刷的描述中，不正确的是（ ）。  
 A. SolidBrush 画刷是一种纯色实心画刷  
 B. PathGradientBrush 画刷可以绘制两个颜色之间的渐变  
 C. TextureBrush 画刷使用纹理填充图形内部  
 D. HatchBrush 画刷可以使用简单、重复性的图案填充图形内部
4. 以下（ ）方法不能绘制封闭图形。  
 A. DrawPloygon      B. DrawPie      C. DrawRectangle      D. DrawLine
5. 使用 g.DrawImage(image,0,0,50,100); 可以显示图像，则以下（ ）语句可以使该图像以水平镜像方式显示。  
 A. g.DrawImage(image,0,0,50,100);      B. g.DrawImage(image,0,0,50,-100);  
 C. g.DrawImage(image,0,0,-50,100);      D. g.DrawImage(image,0,0,-50,-100);

### 二、填空题（每题 10 分，5 道题）

1. 如果要设置整个绘图区的背景色，可以使用 Graphics 类的（ ）方法。
2. 可以使用 Color 结构的（ ）方法，把 R、G、B 值转换为颜色值。
3. 在图片框 pictureBox1 中显示 test.bmp 图片的语句是（ ）。
4. 使用 Graphics 类的 DrawArc 方法可以绘制一段圆弧，该圆弧的起始角度是指从圆弧的起始点沿（ ）方向转过的角度数。
5. 已知 Image image=Image.FromFile("test.jpg");，则使图像旋转 90° 的语句为（ ）。

测试分数统计：

类别	第 1 题	第 2 题	第 3 题	第 4 题	第 5 题
选择题分数					
填空题分数					
总分数					

## 16.6 行动指南

开始日期：\_\_\_\_年\_\_\_\_月\_\_\_\_日

结束日期：\_\_\_\_年\_\_\_\_月\_\_\_\_日

序号	内 容	行动指南	
1	照猫画虎栏目 分数( )	分数>75 分	优秀，基本功掌握得不错，加油！
		75 分>分数>50 分	及格，知识掌握得不牢，重新做一遍照猫画虎。
		分数<50 分	请使用光盘中提供的“实战能力测试系统”进行提高训练。
	情景应用栏目 分数( )	分数>75 分	优秀，综合应用能力很强。
		75 分>分数>50 分	及格，综合应用能力需提高，再练一遍情景应用。
	自我测试栏目 分数( )	分数<50 分	请使用光盘中提供的“实战能力测试系统”进行提高训练。
2	综合评价	分数>75 分	优秀，有成为编程高手的潜质。
	编程能力培养：本栏目提供了一些实践题目，对于培养编程能力很有效，要做好。	分数<75 分	请使用光盘中提供的“实战能力测试系统”进行提高训练。
		综合评价	反复训练后，以上各项分数都在 75 分以上，方可进入下一堂课学习。
		1. 制作一个简单的屏幕颜色拾取器，用来实时获取鼠标所在位置的颜色值。 2. 制作一个图片自动播放器，使指定文件夹中的图片能够以幻灯片的形式在窗体中进行播放。 3. 开发一个程序，主要实现使用柱形图分析商品销售走势的功能。	
3	编程习惯培养：本堂课培养读者在学习开发中记笔记的习惯，把开发中遇到的问题、总结的经验记录下来。		
4	创新能力培养：看看身边有没有可以用编程解决的问题，记录到右边的表格中。根据学习进度，尝试编写解决这些问题的小程序。		

## 16.7 成功可以复制——“盖茨第二”马克·扎克伯格

1984 年 5 月 14 日，马克·扎克伯格出生在纽约，他很小就展现出了计算机天赋，六年级就开始编程。对于学习编程的经历，马克·扎克伯格说：“我买了一些书，基本是自己在胡乱摸索，也向很多人请教过。我对计算机硬件不感兴趣，我只是对计算机程序很感兴趣。”高中时，他与好友达杰罗为 MP3 播放器编写了一个智能插件软件，并放在互联网上供免费下载。很快，包括美国在线（AOL）和微软在内的各大公司纷纷向他抛来了橄榄枝。但是扎克伯格却拒绝了年薪 95 万美元的工作机会，而选择去哈佛大学上学。

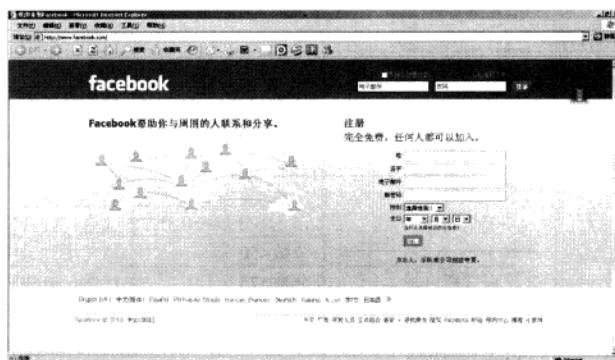
身为常春藤名校的学生，马克·扎克伯格曾闹出了一起著名的黑客事件。在新生第一年即将结束时，马克发现哈佛大学没有制作学生年鉴的惯例，于是向学校理事会提议制作一个在线版本，但是学校总是用资料难以收集等借口来推托。在正式跨入二年级的前一天，他侵入了哈佛的学生信息登记系统，获得了所有在校本科生的资料，并迅速架设了名为 Facemash 的网站。这个网站随机展示两个学生的照片，让来访者

判定哪一个更“漂亮”。在短短 4 个小时内，访问量达到 450 人次，22000 张照片被点击。校方发现该网站后当即掐断了马克宿舍的网络连接。在遭受了校方和哈佛校报的轮番训斥后，马克向全校学生进行了公开道歉。但他并不觉得自己做错了什么，“我始终认为，信息就应该得以利用。”

2004 年 2 月 4 日，社交网站 Facebook 在大学二年级生马克·扎克伯格的手中诞生了。当时它还只是一个仅面向哈佛在校生的小型网络，随后在同学达斯汀·莫斯可维茨的协助下，将 Facebook 推广至美国的其他大学。2004 年年底，Facebook 的注册人数已突破 100 万，马克·扎克伯格干脆从哈佛退学，全职营运网站。

在社交型网络发展的早期，Facebook 依靠数笔投资得以顺利运营。2005 年，雅虎曾正式开出了 10 亿美元的天价并购 Facebook，但被当时年仅 22 岁的马克一口回绝“我们既不打算卖掉公司，也不打算近期内 IPO（初次公开募股）”。马克始终坚持公司的目标是寻求长期发展，不必分心兼顾其他事情。

2007 年年底，仅仅用了不到 4 年的时间，Facebook 已成为美国最火爆的大学生社交网站，网站全球排名第 8 位。目前注册用户超过了 4 亿，同时在线人数超过了 1 亿。



Facebook 导航页

## ✓ 经典语录

最大的风险就是不去承担任何风险。在这个瞬息万变的世界里，不变即意味着失败。改变对于用户尤其是网络服务用户而言总是颠覆性的。

## ✓ 深度评价

机会就是别人不知道他知道了，别人不明白他明白了，别人犹豫或不做而他做了。当别人知道了，明白了，想要做时，他已经成功了！机会就是这样，总是偏爱少数人，因为大部分人都有一种惰性，喜欢跟风，人云亦云。

# 第 17 堂课

## 水晶报表与打印

( 视频讲解：137分钟)

水晶报表是内置于 Visual Studio 开发环境中的一种报表设计工具，它能帮助程序员在.NET 平台上创建高复杂度且专业级的互动式报表。另外，在 Windows 应用程序中还提供了一组打印组件，包括 PageSetupDialog、PrintDialog、PrintDocument、PrintPreviewControl 和 PrintPreviewDialog 组件等，开发程序时，开发人员可以直接使用这些组件控制打印的文本和数据格式。本堂课将对水晶报表及打印组件的使用进行详细讲解。

学习摘要：

- » 了解什么是水晶报表
- » 熟悉水晶报表的工作区
- » 了解水晶报表常见的数据源及数据访问模式
- » 掌握如何创建水晶报表
- » 掌握如何对水晶报表中的数据进行分组和排序
- » 掌握如何筛选水晶报表中的数据
- » 掌握如何在水晶报表中使用图表
- » 掌握如何创建子报表
- » 掌握 Windows 打印组件的使用

## 17.1 认识水晶报表

水晶报表协同数据库一起工作，可以帮助用户分析和解释重要的信息。使用水晶报表可以创建简单的报表，也可以创建复杂的、专业的报表，它可以从任何数据源生成所需要的报表。在设计好报表后，水晶报表可以将报表通过多种形式发布，如 Word、Excel 或 Web 网页等。高级的 Web 水晶报表还允许工作组中的其他成员在其 Web 浏览器中查看或共享报表。水晶报表可以通过控件的形式整合到软件开发人员的数据应用环境中。本节将对水晶报表进行简单介绍。

### 17.1.1 水晶报表概述

Crystal Reports 水晶报表自 1993 年开始就已经成为 Visual Studio 的一部分，如今已经成为了 Visual Studio 2008 中的标准报表创建工具。每套 Visual Studio 2008 都附带了该工具，并且直接集成到程序开发环境中，因此，集成到 Visual Studio.NET 中的 Crystal Reports 被称为 Crystal Reports.NET。利用 Crystal Reports.NET 能够在 Windows 环境中创建达到演示质量的交互式内容，另外，使用 Crystal Reports.NET 可在基于 GUI 的程序中创建复杂而专业的报表，然后，可以将报表连接到几乎所有数据源及代理数据，如结果集。使用 GUI 设计器中附带的向导，可以方便地设置格式化、分组、图表制作和其他条件。通过使用 Crystal Reports.NET 的查看器控件之一，可以在 Web 或 Windows 应用程序中承载报表。Windows 客户端和 HTML 3.2 或 4 客户端中的报表显示均具有高度交互性的特点，并且提供图表、报表导航和文本搜索等功能。

### 17.1.2 水晶报表工作区介绍

水晶报表设计器的环境大体可以分为左右两部分，左侧为“字段资源管理器”窗口，该窗口中显示的树型图包含了可以添加到水晶报表中的数据库字段、特殊字段、公式字段、参数字段、组名字段、运行总计字段、SQL 表达式字段、特殊字段和未绑定字段；右侧为“报表设计区”窗口，该窗口中显示了水晶报表的不同部分，即水晶报表的节。将“字段资源管理器”中的不同字段直接拖曳到水晶报表节中，即可设置在水晶报表的不同部分所显示的数据内容，用户可以根据需要设计各种样式的水晶报表。水晶报表设计器环境如图 17.1 所示。



图 17.1 水晶报表设计器

对于一张新的水晶报表，报表工作区被分成 5 部分，可以选择创建其他区域，也可以隐藏某些区域。水晶报表工作区如下。

- 报表头：“报表头”区域中的信息和对象只在水晶报表的开头显示一次。
- 页眉：“页眉”区域中的信息和对象显示在每个新页的开始位置，该区域通常包含只出现在每页顶部的信息，例如它可以包含文本字段，也可以用来包含字段标题等。图表不能放置在该区域中。放置在该区域中的公式在每个新页的开始进行一次求值。
- 详细资料：“详细资料”区域中的信息和对象随每条新记录显示，该区域包含水晶报表正文数据，例如，批量报表数据通常出现在该区域。图表不能放置在该区域中。放置在该区域中的公式对每条记录进行一次求值。
- 报表尾：“报表尾”区域中的信息和对象只在水晶报表的结束位置显示一次，该区域可用来输出只在水晶报表的末尾出现一次的信息（如统计）。放置在该区域中的图表包含整个水晶报表的数据，而放置在该区域中的公式只在水晶报表的结束位置进行一次求值。
- 页脚：“页脚”区域中的信息和对象显示在每页的底部，该区域通常包含页码和任何其他希望出现在每页底部的信息。图表不能放置在该区域中。放置在该区域中的公式在每个新页的结束位置进行一次求值。

另外，水晶报表中还包含其他报表区域，例如，如果将组、摘要或小计信息添加到水晶报表中，则报表设计器会自动添加另外两个区域，即“组头”和“组尾”。“组头”区域出现在“详细资料”区域的正上方，“组尾”区域出现在“详细资料”区域的正下方。每个新添加的区域也可以包含一个或多个子区域。“组头”区域中的对象输出显示在每个新组的开始位置，该区域通常包含组名字段，也可以用来显示包括组特定数据的图表。“组头”区域内容在每组的开始位置输出显示一次。“组尾”区域中的对象输出显示在每组的结束位置，该区域通常保存汇总数据，同时也可以用来显示图表。

## 17.2 水晶报表数据源和数据访问模式

### 17.2.1 Visual Studio 2008 中水晶报表数据源列举

水晶报表通过数据库驱动程序与数据库进行连接，用户可以根据下列数据源中的数据进行报表设计。

- 使用 ODBC 驱动程序的任何数据库。
- 使用 OLEDB 提供程序的任何数据库。
- Microsoft Access 数据库。
- Microsoft Excel 工作簿。
- ADO.NET 记录集。
- ADO.NET 记录集。
- CDO 记录集。
- DAO.NET 记录集。
- RDO 记录集。

### 17.2.2 水晶报表的数据访问模式

水晶报表的数据访问模式可以分为“提取模式”和“推入模式”两种。所谓提取模式，就是指驱动程

序会自行链接至数据库并根据需要来提取数据，开发人员不需要另外编写代码。

**技巧：**如果运行阶段未编写特定的代码，则使用提取模式。

如果采用推入模式，则开发人员必须自行编写代码来链接至数据库，执行 SQL 命令来创建数据集或数据记录集，并将该对象传递给水晶报表。

**注意：**提取模式只能访问 ODBC、OLEDB 与 Access/Excel 数据源，而推入模式可以通过 ADO.NET、ADO、CDO、DAO 与 RDO 来访问各种类型的数据源。

## 17.3 水晶报表基本操作

### 17.3.1 创建水晶报表并连接数据源

下面通过实例演示如何在 Visual Studio 2008 开发环境中创建水晶报表及连接数据源。

**例 17.01** 创建一个水晶报表，并连接 SQL Server 数据源，具体步骤如下。（实例位置：光盘\mr\17\sl\17.01）

- (1) 新建一个 Windows 应用程序，命名为 CreateCReport，默认窗体为 Form1.cs。
- (2) 在“解决方案资源管理器”窗口中选中当前项目名称，单击鼠标右键，在弹出的快捷菜单中选择“添加”/“新建项”命令，在弹出的“添加新项”对话框中选择 Reporting/“Crystal 报表”选项，如图 17.2 所示。

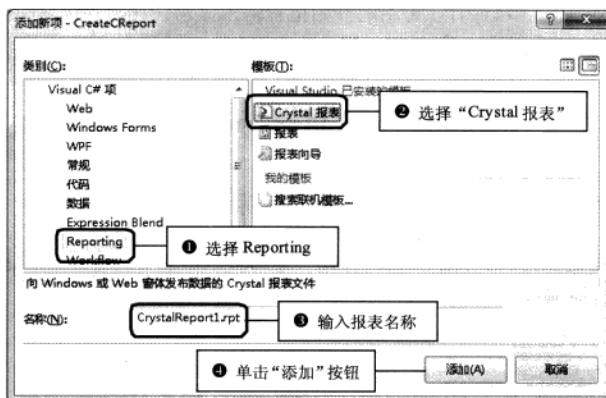


图 17.2 “添加新项”对话框

- (3) 单击“添加”按钮，弹出“Crystal Reports 库”对话框。在该对话框中设置报表的属性，选中“使用报表向导”单选按钮，并选择“选择专家”区域中的“标准”选项，如图 17.3 所示。
- (4) 单击“确定”按钮，打开“标准报表创建向导-数据”对话框，在此对话框中建立与数据源的连接，如图 17.4 所示。
- (5) 连接数据库之前，需要根据所要连接数据库的数据源来选择相应的“提供程序”。单击“创建新连接”项左侧的加号（+），然后双击其子项 OLE DB (ADO)，弹出“OLE DB (ADO)-提供程序”对话框，如图 17.5 所示。

(6) 由于本实例使用 SQL Server 2000 作为报表的数据源，所以在“提供程序”列表框中选择 Microsoft OLE DB Provider for SQL Server 选项，单击“下一步”按钮，弹出“OLE DB (ADO) -连接信息”对话框，如图 17.6 所示。



图 17.3 “Crystal Reports 库”对话框

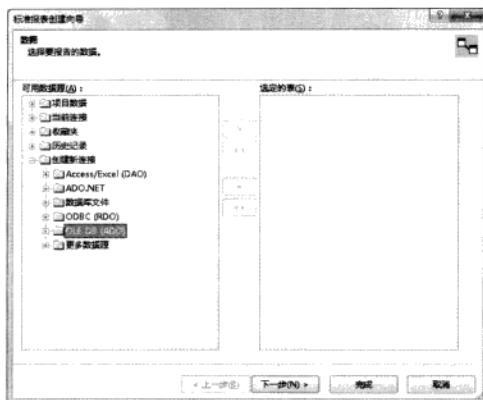


图 17.4 “标准报表创建向导-数据”对话框



图 17.5 “OLE DB (ADO) -提供程序”对话框



图 17.6 “OLE DB (ADO) -连接信息”对话框

(7) 在“OLE DB (ADO) -连接信息”对话框中依次进行设置，在“服务器”下拉列表框中选择所要连接的 SQL Server 服务器名称；在“数据库”下拉列表框中选择所要连接的 SQL Server 数据库，本实例数据库名为 db\_CSharp；根据不同的身份验证模式登录 SQL Server 来进行不同的设置，如果要采用“Windows 账户验证模式”登录，则选中“集成安全”复选框；如果要采用“SQL Server 账户验证模式”登录，分别在“用户 ID”与“密码”文本框中输入用来登录 SQL Server 的用户识别码与密码，并取消“集成安全”复选框的选中。完成上述设置后单击“完成”按钮，返回“标准报表创建向导-数据”对话框，在该对话框中选择要操作的数据表，本实例使用的是 tb\_stu 表，如图 17.7 所示。

(8) 单击“下一步”按钮，打开“标准报表创建向导-字段”对话框，通过此对话框可以设置在报表中显示的字段，如图 17.8 所示。

(9) 选择要显示的字段后单击“完成”按钮，完成水晶报表的创建，如图 17.9 所示。

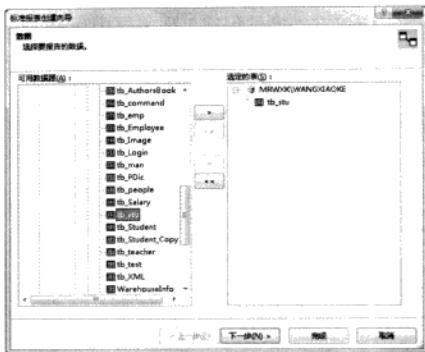


图 17.7 选择要操作的数据表

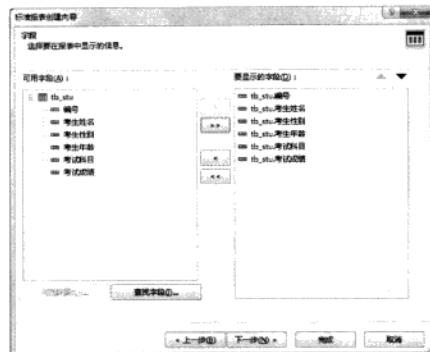


图 17.8 设置显示的字段

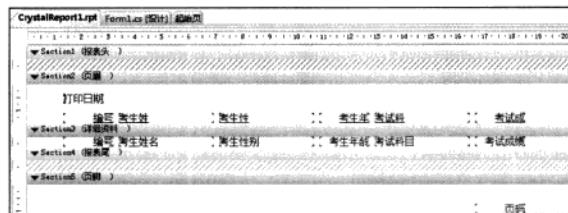


图 17.9 水晶报表创建完毕

(10) 为使水晶报表中的数据能够显示在窗体中，需要使用 CrystalReportViewer 控件。从“工具箱/报表”选项卡中向 Form1 窗体中拖放一个 CrystalReportViewer 控件，然后将其 ReportSource 属性设置为要显示的水晶报表，这里设置为 CrystalReport11 [CreateCReport.CrystalReport1]。实例运行效果如图 17.10 所示。

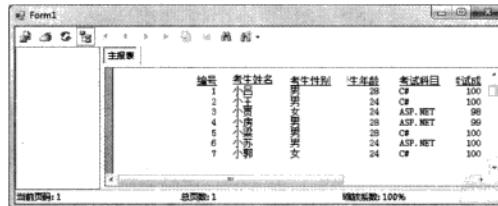


图 17.10 运行水晶报表

**技巧：**设计水晶报表时，可以使用“Crystal Reports-主工具栏”中的各个功能按钮设置数据的显示样式。

### 17.3.2 水晶报表中数据的分组与排序

#### 1. 分组

对水晶报表中的数据进行分组，可使报表中的数据阅读起来更加清晰明了。下面通过一个实例演示如何对水晶报表中的数据进行分组。

**例 17.02** 创建一个水晶报表，按性别对学生信息进行分组，具体步骤如下。（实例位置：光盘\mr\17\sl\17.02）

(1) 创建一个 Windows 应用程序，命名为 CReportGroup，默认窗体为 Form1.cs。

(2) 创建一个名为 CrystalReport1.rpt 的水晶报表，并连接 SQL Server 数据源。创建步骤可参见 17.3.1 节的相关内容。

(3) 在报表设计区域的任意空白处单击鼠标右键，在弹出的快捷菜单中选择“报表”/“组专家”命令，弹出“组专家”对话框。在该对话框的“可用字段”列表框的“报表字段”选项中选择“考生性别”字段，然后单击  按钮，将选中字段添加至“分组依据”列表框中，如图 17.11 所示。

(4) 分组字段选择完成后单击“选项”按钮，弹出“更改组选项”对话框。在该对话框中可以设置或更改分组字段，如图 17.12 所示。

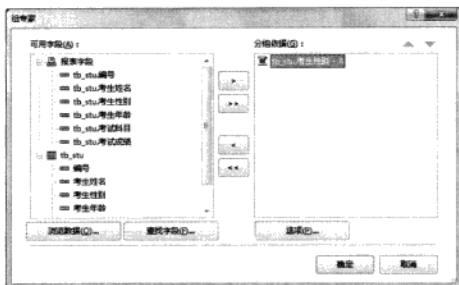


图 17.11 “组专家”对话框



图 17.12 “更改组选项”对话框

(5) 单击“确定”按钮，完成分组设置，返回“报表设计器”，如图 17.13 所示。

(6) 从“工具箱/报表”选项卡中向 Form1 窗体中拖放一个 CrystalReportViewer 控件，然后将其 ReportSource 属性设置为要显示的水晶报表。实例运行效果如图 17.14 所示。

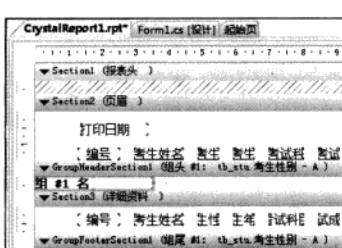


图 17.13 分组后的报表设计器

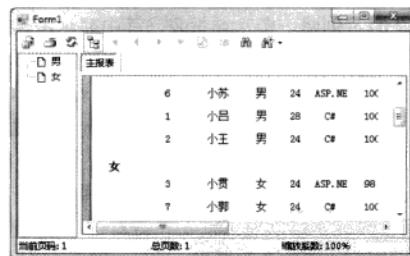


图 17.14 报表分组

## 2. 排序

如果要按照特定顺序重新排列报表的数据，那么需要对报表进行排序。排序对于需要按照指定顺序显示信息的简单报表特别有用。下面通过实例演示如何对报表数据进行排序。

**例 17.03** 创建一个水晶报表，按考生成绩降序排列显示学生信息，具体步骤如下。**(实例位置：光盘\mr\17\sl\17.03)**

(1) 创建一个 Windows 应用程序，命名为 CReportSort，默认窗体为 Form1.cs。

(2) 创建一个名为 CrystalReport1.rpt 的水晶报表，并连接 SQL Server 数据源。创建步骤可参见 17.3.1 节的相关内容。

(3) 在报表设计区域的任意空白处单击鼠标右键，在弹出的快捷菜单中选择“报表”/“记录排序专家”

选项，弹出“记录排序专家”对话框。在该对话框的“可用字段”列表框中选中“考试成绩”字段，然后单击 $\triangleright$ 按钮，添加到“排序字段”列表框中。在“排序方向”区域中选中“降序”单选按钮，如图 17.15 所示。

(4) 单击“确定”按钮返回报表设计器，从“工具箱/报表”选项卡中向 Form1 窗体中拖放一个 CrystalReportViewer 控件，然后将其 ReportSource 属性设置为要显示的水晶报表，实例运行效果如图 17.16 所示。

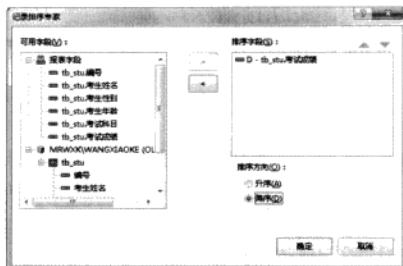


图 17.15 “记录排序专家”对话框

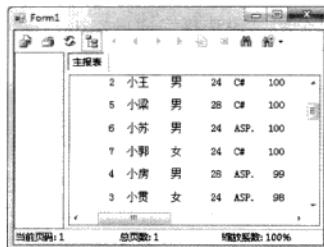


图 17.16 按照考试成绩降序排列

### 17.3.3 水晶报表中数据的筛选

在默认情况下，报表中的每条记录都将被显示出来，但在实际的应用中，可能只需要显示其中符合条件的记录内容。一般可以用两种方式来选择符合条件的记录：一种是利用“选择专家”筛选记录，另一种是开发人员自定义公式筛选记录。下面分别进行介绍。

#### 1. 利用“选择专家”筛选记录

**例 17.04** 创建一个水晶报表，其中显示考试成绩大于 99 的学生信息，具体步骤如下。（实例位置：光盘\mr\17\sh\17.04）

(1) 创建一个 Windows 应用程序，命名为 CReportFilter，默认窗体为 Form1.cs。

(2) 创建一个名为 CrystalReport1.rpt 的水晶报表，并连接 SQL Server 数据源。创建步骤参见 17.3.1 节的相关内容。

(3) 在报表设计区域的任意空白处单击鼠标右键，在弹出的快捷菜单中选择“报表”/“选择专家”命令，弹出“选择字段”对话框。在该对话框中选择要设置条件限制的表字段，如图 17.17 所示。

(4) 单击“确定”按钮，弹出“选择专家”对话框。在该对话框的选项卡的下拉列表框中有多种筛选条件可供选择，这里筛选考试成绩大于 99 的学生信息，如图 17.18 所示。

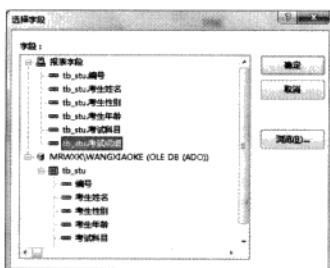


图 17.17 “选择字段”对话框

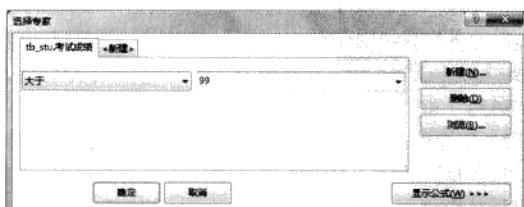


图 17.18 “选择专家”对话框

(5) 单击“确定”按钮返回报表设计器，从“工具箱/报表”选项卡中向 Form1 窗体中拖放一个 CrystalReportViewer 控件，然后将其 ReportSource 属性设置为要显示的水晶报表。实例运行效果如图 17.19 所示。

## 2. 开发人员自定义公式筛选记录

如果利用“选择专家”无法满足所需的筛选，那么开发人员就可以使用自定义公式筛选记录。在报表设计区域的空白处单击鼠标右键，在弹出的快捷菜单中选择“报表”/“选定公式”/“记录”命令，弹出“公式工作室-记录选定公式编辑器”对话框，如图 17.20 所示。



图 17.19 筛选考试成绩大于 99 的学生信息



图 17.20 “公式工作室-记录选定公式编辑器”对话框

在图 17.20 所示的对话框中输入具有限制条件的公式，并单击工具栏中的“检查”按钮，检查公式是否存在错误，在确认无误后单击“保存并关闭”按钮，完成记录的筛选操作。下面介绍常用的自定义公式模板。

### (1) 使用数字选择记录

- {文件.字段}>10：选择{文件.字段}值大于 10 的记录。
- {文件.字段}<10：选择{文件.字段}值小于 10 的记录。
- {文件.字段}>10 and {文件.字段}<100：选择{文件.字段}值大于 10，并且小于 100 的记录（不包括 10 和 100）。
- {文件.字段}>=10 and {文件.字段}<=100：选择{文件.字段}值大于等于 10，并且小于等于 100 的记录。

**例 17.05** 筛选考试成绩大于 99 的信息，可以在“公式工作室-记录选定公式编辑器”对话框中输入“{tb\_stu.考试成绩}>99”，如图 17.21 所示。

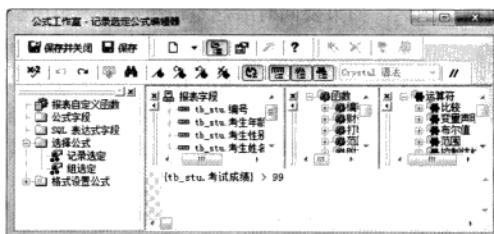


图 17.21 筛选考试成绩大于等于 99 的学生信息

### (2) 用字符串选择记录

- {文件.字段} startswith“a”：选择{文件.字段}值以字符“a”开始的记录。
- not ({文件.字段}) startswith“a”：选择{文件.字段}值不以字符“a”开始的记录。
- “528” in {文件.字段}[3 to 5]：选择{文件.字段}值中的第 3 位到第 5 位数字为“528”的记录。

- “cp” in {文件.字段}: 选择{文件.字段}值包含字符串“cp”的记录。
- (3) 使用日期选择记录
- Year ({文件.日期}) <2009: 选择{文件.日期}字段中的年份小于 2009 年的记录。
- Year ({文件.日期}) >2000 and Year ({文件.日期}) <2009: 选择{文件.日期}字段中的年份在 2000 年和 2009 年之间的记录(不包括 2000 年和 2009 年)。
- Year ({文件.日期}) >=2000 and Year ({文件.日期}) <=2009: 选择{文件.日期}字段中的年份在 2000 年和 2006 年之间的记录(包括 2000 年和 2009 年)。
- Month ({文件.日期}) in 3 to5: 选择{文件.日期}字段中月份为一年中 3 到 5 月份的记录(包括 3 月、4 月和 5 月)。
- Month ({文件.日期}) in [3,5]: 选择{文件.日期}字段中月份为一年中的第 3 个月和第 5 个月的记录(不包括 4 月)。
- (4) 使用预置数据范围选择记录
- {文件.日期} in LastFullMonth: 选择{文件.日期}字段中的日期在上个月整月范围内的记录(如果本月是 5 月，则选择 4 月的所有记录)。
- not {文件.日期} in LastFullMonth: 选择{文件.日期}字段中日期在上个月整月范围之外的记录(如果本月是 5 月，则选择除了 4 月以外日期的所有记录)。
- {文件.日期}<CurrentDate: 选择{文件.日期}字段中日期在今日之前的所有记录。
- (5) 使用日期/数字/字符组合选择记录
- “a” in {文件.字段}[1] and Month ({文件.日期}) in [3,5]: 选择{文件.日期}字段值以“a”开头并且月份是 3 月和 5 月的记录。
- “TS” in {文件.编号}[3 to 4] and {文件.数量}>100: 选择{文件.编号}字段中的第 3 和第 4 个字符分别为“T”和“S”，并且{文件.数量}字段值大于 100 的记录。

### 17.3.4 在水晶报表中使用图表

在水晶报表中使用图表，会让各项统计数据所蕴含的趋势、走向与彼此间的对比及差异等关系更加一目了然。同时可将图表摆放在报表头、报表尾、组头与组尾节中，不过随着摆放的节的不同，图表绘制的数据对象也将有所不同。位于报表头或报表尾节中的图表，则绘制整份报表的数据；位于组头或组尾节中的图表，则只绘制该组中的数据。当然，如果要同时绘制整份报表以及组中的数据，则可以在报表头或报表尾以及组头或组尾节中各建立一个图表。图表的类型很多，且每种类型都有一个适合自己的应用，大致可以分为条形图、线性图、面积图、饼图、圆环图、三维梯形图、三维曲面图、XY 散点图、雷达图、气泡图、股票图、数轴、量度、甘特图和漏斗形等。下面通过实例演示如何在水晶报表中使用图表。

**例 17.06** 创建一个水晶报表，在水晶报表中插入图表，用于显示考生考试成绩的差距，具体步骤如下。

(实例位置：光盘\mr\17\sl\17.06)

- (1) 创建一个 Windows 应用程序，并将其命名为 CReportImage，默认窗体为 Form1.cs。
- (2) 创建一个名为 CrystalReport1.rpt 的水晶报表，并连接 SQL Server 数据源。创建步骤可参见 17.3.1 节的相关内容。
- (3) 在“报表头”节中的空白处单击鼠标右键，在弹出的快捷菜单中选择“插入”/“图表”命令，弹出“图表专家”对话框，如图 17.22 所示。
- (4) 在“图表专家”对话框的“类型”选项卡的“图表类型”列表框中选择“条形图”，“自动设置图表选项”复选框默认为选中状态，表示图表的坐标轴、颜色、数据点、数据标记、图例与条形图大小等

设置都采用默认值，如果希望自定义这些设置，可以取消选中，此时将会显示“坐标轴”与“选项”两个选项卡，以便自定义相关设置，如图 17.23 所示。

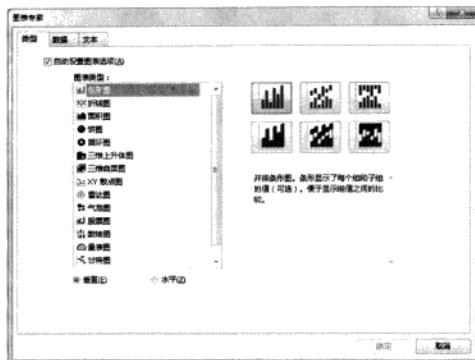


图 17.22 “图表专家”对话框

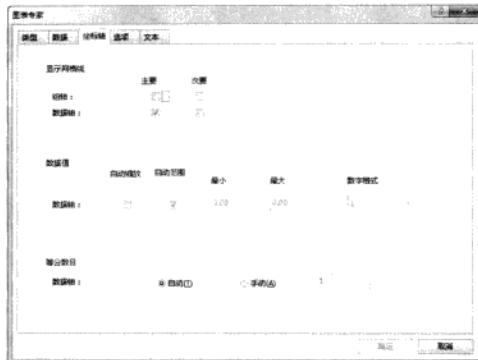


图 17.23 自定义设置图表选项

(5) 在“图表专家”对话框的“数据”选项卡中，从“位置”区域中的“放置图表”下拉列表框中选择“每个报表一次”选项，用于分析整份报表的数据。将“可用字段”列表框中的“tb\_stu.考生姓名”字段添加到“变更主体”列表中；将“可用字段”列表框中的“tb\_stu.考试成绩”字段添加到“显示值”列表框中。设置结果如图 17.24 所示。

(6) 单击“确定”按钮返回报表设计器，从“工具箱/报表”选项卡中向 Form1 窗体中拖放一个 CrystalReportViewer 控件，然后将其 ReportSource 属性设置为要显示的水晶报表。实例运行效果如图 17.25 所示。



图 17.24 “图表专家”对话框的“数据”选项卡

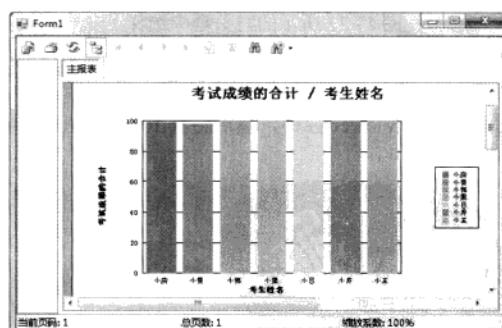


图 17.25 在水晶报表中使用图表

### 17.3.5 在水晶报表中创建子报表

子报表是内含于报表中的报表，在这类报表中，主报表与子报表相互联系，而子报表仅显示与主报表有关的记录。下面通过实例演示如何在水晶报表中创建子报表。

**例 17.07** 创建一个水晶报表，在水晶报表中插入图表，用于显示考生考试成绩的差距，具体步骤如下。  
(实例位置：光盘\mr\17\sl\17.07)

- (1) 创建一个 Windows 应用程序，并将其命名为 CReportChild，默认窗体为 Form1.cs。
- (2) 创建一个名为 CrystalReport1.rpt 的水晶报表，并连接 SQL Server 数据源。创建步骤可参见 17.3.1 节的相关内容。

(3) 在水晶报表的“详细资料”区域单击鼠标右键，在弹出的快捷菜单中选择“插入”/“子报表”命令，弹出“插入子报表”对话框，如图 17.26 所示。在该对话框中既可以选择现有报表作为子报表，也可以新建一个报表作为子报表，这里选中“使用报表向导创建子报表”单选按钮，输入创建的子报表名称，单击“报表向导”按钮，然后按照 17.3.1 节的步骤创建一个子报表。

(4) 单击“确定”按钮返回报表设计器，在“详细资料”区域拖动子报表的放置位置，然后选择子报表，单击鼠标右键，在弹出的快捷菜单中选择“更改子报表链接”命令，弹出“子报表链接”对话框，如图 17.27 所示。



图 17.26 “插入子报表”对话框



图 17.27 “子报表链接”对话框

(5) 在图 17.27 所示的对话框中选择“tb\_stu.编号”作为链接添加到“字段链接到”列表框中，如图 17.28 所示。

(6) 单击“确定”按钮返回报表设计器，从“工具箱/报表”选项卡中向 Form1 窗体中拖放一个 CrystalReportViewer 控件，然后将其 ReportSource 属性设置为要显示的水晶报表。实例运行效果如图 17.29 所示。

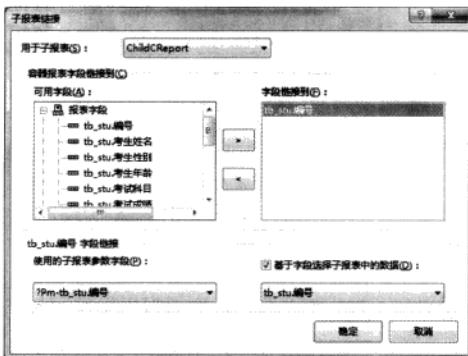


图 17.28 选择“tb\_stu.编号”作为链接

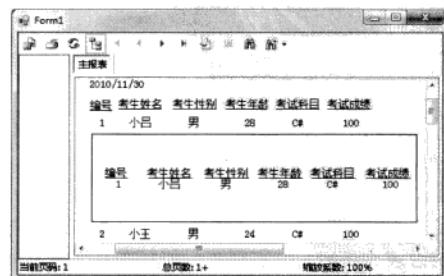


图 17.29 在水晶报表中显示子报表

## 17.4 Windows 打印组件的使用

通过 Windows 打印组件，可以方便、快捷地对文档进行预览、设置和打印。C#中的 Windows 打印组件如图 17.30 所示。下面对 Windows 打印组件进行详细介绍，并举例说明如何使用这些打印组件。

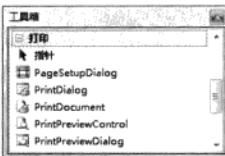


图 17.30 Windows 打印组件

### 17.4.1 使用 PageSetupDialog 组件设置打印文档信息

PageSetupDialog 组件用于设置页面详细信息以便打印，它允许用户设置边框和边距调整量、页眉和页脚以及纵向或横向打印。PageSetupDialog 组件的常用属性及说明如表 17.1 所示。

表 17.1 PageSetupDialog 组件的常用属性及说明

属性	说明
Document	获取页面设置的 PrintDocument 对象
AllowMargins	是否启用对话框的边距部分
AllowOrientation	是否启用对话框的方向部分（横向对纵向）
AllowPaper	是否启用对话框的纸张部分（纸张大小和纸张来源）
AllowPrinter	是否启用“打印机”按钮

下面对这几种常用的属性进行详细介绍。

#### (1) Document 属性

用于获取页面设置的 PrintDocument。

语法： public PrintDocument Document { get; set; }

说明：它的属性值表示从中获得页面设置的 PrintDocument。

#### (2) AllowMargins 属性

用于设置是否启用对话框的边距部分。

语法： public bool AllowMargins { get; set; }

说明：它的属性值表示，如果启用对话框的边距部分，则为 true；否则为 false。默认为 true。

#### (3) AllowOrientation 属性

用于设置是否启用对话框的方向部分（横向对纵向）。

语法： public bool AllowOrientation { get; set; }

说明：它的属性值表示，如果启用对话框的方向部分，则为 true；否则为 false。默认为 true。

#### (4) AllowPaper 属性

用于设置是否启用对话框的纸张部分（纸张大小和纸张来源）。

语法： public bool AllowPaper { get; set; }

说明：它的属性值表示，如果启用对话框的纸张部分，则为 true；否则为 false。默认为 true。

#### (5) AllowPrinter 属性

用于设置是否启用“打印机”按钮。

语法： public bool AllowPrinter { get; set; }

说明：它的属性值表示，如果启用“打印机”按钮，则为 true；否则为 false。默认为 true。

**例 17.08** 下面代码用来设置“页面设置”对话框中的打印文档，并启用页边距、方向、纸张和“打印机”按钮，代码如下。

```
pageSetupDialog1.Document = printDocument1;//设置 pageSetupDialog 组件的 Document 属性，并设置操作文档
this.pageSetupDialog1.AllowMargins = true;           //启用页边距
this.pageSetupDialog1.AllowOrientation = true;        //启用对话框的方向部分
this.pageSetupDialog1.AllowPaper = true;             //启用对话框的纸张部分
this.pageSetupDialog1.AllowPrinter = true;            //启用“打印机”按钮
this.pageSetupDialog1.ShowDialog();                  //显示“页面设置”对话框
```

### 17.4.2 使用 PrintDialog 组件显示打印对话框

PrintDialog 组件用于选择打印机、要打印的页以及确定其他与打印相关的设置，通过它可以选择全部打印、打印选定的页范围或打印选定内容等。PrintDialog 组件的常用属性及说明如表 17.2 所示。

表 17.2 PrintDialog 组件的常用属性及说明

属性	说明
Document	获取 PrinterSettings 类的 PrintDocument 对象
AllowCurrentPage	是否显示“当前页”选项按钮
AllowPrintToFile	是否启用“打印到文件”复选框
AllowSelection	是否启用“选择”选项按钮
AllowSomePages	是否启用“页”选项按钮

下面对这几种常用的属性进行详细介绍。

#### (1) Document 属性

用于获取 PrinterSettings 的 PrintDocument。

语法： public PrintDocument Document { get; set; }

说明：它的属性值表示 PrinterSettings 的 PrintDocument。

#### (2) AllowCurrentPage 属性

用于设置是否显示“当前页”按钮。

语法： public bool AllowCurrentPage { get; set; }

说明：它的属性值表示，如果显示“当前页”按钮，为 true；否则为 false。默认为 false。

#### (3) AllowPrintToFile 属性

用于设置是否启用“打印到文件”复选框。

语法： public bool AllowPrintToFile { get; set; }

说明：它的属性值表示，如果启用“打印到文件”复选框，为 true；否则为 false。默认为 true。

#### (4) AllowSelection 属性

用于设置是否启用“选择”按钮。

语法： public bool AllowSelection { get; set; }

说明：它的属性值表示，如果启用“选择”按钮，为 true；否则为 false。默认为 false。

#### (5) AllowSomePages 属性

用于设置是否启用“页”按钮。

语法：public bool AllowSomePages { get; set; }

说明：它的属性值表示，如果启用“页”按钮，为 true；否则为 false。默认为 false。

**例 17.09** 下面代码用来设置“打印”对话框中的打印文档，并启用“打印到文件”复选框、“当前页”按钮、“选择”按钮和“页”按钮，代码如下。

```
printDialog1.Document = printDocument1; //设置 printDialog 组件的 Document 属性，设置操作文档
printDialog1.AllowPrintToFile = true; //启用“打印到文件”复选框
printDialog1.AllowCurrentPage = true; //显示“当前页”按钮
printDialog1.AllowSelection = true; //启用“选择”按钮
printDialog1.AllowSomePages = true; //启用“页”按钮
printDialog1.ShowDialog(); //显示“打印”对话框
```

### 17.4.3 使用 PrintDocument 组件设置打印文档

PrintDocument 组件用于设置打印的文档，程序中常用到的是该组件的 PrintPage 事件和 Print 方法。PrintPage 事件在需要为当前页打印输出时发生；而 Print 方法则用于开始文档的打印进程。下面通过实例演示如何使用 PrintDocument 组件。

**例 17.10** 创建一个 Windows 应用程序，向窗体中添加一个 Button 控件、一个 PrintDocument 组件和一个 PrintPreviewDialog 组件。在 PrintDocument 组件的 PrintPage 事件中绘制打印的内容，然后在 Button 按钮的 Click 事件下设置 PrintPreviewDialog 的属性预览打印文档，并调用 PrintDocument 组件的 Print 方法开始文档的打印进程，代码如下。（实例位置：光盘\mr\17\sl\17.10）

```
private void printDocument1_PrintPage(object sender, System.Drawing.Printing.PrintPageEventArgs e)
{
    //通过 GDI+绘制打印文档
    e.Graphics.DrawString("蝶恋花", new Font("宋体", 15), Brushes.Black, 350, 80);
    e.Graphics.DrawLine(new Pen(Color.Black, (float)3.00), 100, 185, 720, 185);
    e.Graphics.DrawString("伫倚危楼风细细，望极春愁，黯黯生天际。", new Font("宋体", 12), Brushes.Black, 110,
195);
    e.Graphics.DrawString("草色烟光残照里，无言谁会凭阑意。", new Font("宋体", 12), Brushes.Black, 110, 220);
    e.Graphics.DrawString("拟把疏狂图一醉，对酒当歌，强乐还无味。", new Font("宋体", 12), Brushes.Black, 110,
245);
    e.Graphics.DrawString("衣带渐宽终不悔。为伊消得人憔悴。", new Font("宋体", 12), Brushes.Black, 110, 270);
    e.Graphics.DrawLine(new Pen(Color.Black, (float)3.00), 100, 300, 720, 300);
}
private void button1_Click(object sender, EventArgs e)
{
    if (MessageBox.Show("是否要预览打印文档", "打印预览", MessageBoxButtons.YesNo) == DialogResult.Yes)
    {
        this.printPreviewDialog1.UseAntiAlias = true; //开启操作系统的防锯齿功能
        this.printPreviewDialog1.Document = this.printDocument1; //设置要预览的文档
        printPreviewDialog1.ShowDialog(); //打开预览窗口
    }
}
```

```

    this.printDocument1.Print();
}
}

```

//调用 Print 方法直接打印文档

运行程序，单击窗体中的“打印”按钮，弹出“打印预览”窗体，如图 17.31 所示。

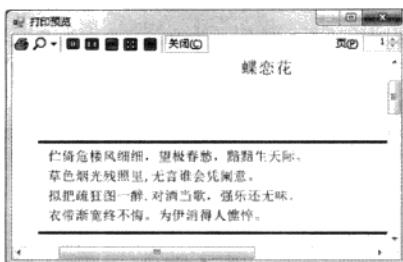


图 17.31 打印预览

#### 17.4.4 使用 PrintPreviewControl 组件设置打印预览文档

PrintPreviewControl 组件用于按文档打印时的外观显示文档，该组件只为用户提供一个预览打印文档功能，因此通常只有在希望编写自己的打印预览用户界面时才使用它。PrintPreviewControl 组件中比较重要的是 Document 属性，该属性用于设置要预览的文档。

语法：public PrintDocument Document { get; set; }

说明：它的属性值为 PrintDocument 对象，表示要预览的文档。

下面通过实例演示如何使用 PrintPreviewControl 组件。

**例 17.11** 创建一个 Windows 应用程序，向窗体中添加一个 PrintPreviewControl 组件和一个 PrintDocument 组件。在 PrintDocument 组件的 PrintPage 事件中绘制图像，然后在窗体的 Load 事件中设置 PrintPreviewControl 组件的 Document 属性，代码如下。（实例位置：光盘\mr\17\sl\17.11）

```

private void Form1_Load(object sender, EventArgs e)
{
    //设置 printPreviewControl1 组件的 Document 属性，设置要预览的文档
    printPreviewControl1.Document = printDocument1;
}

private void printDocument1_PrintPage(object sender, System.Drawing.Printing.PrintPageEventArgs e)
{
    string str = "明日科技.bmp"; //声明一个 string 类型变量，用于存储图片位置
    e.Graphics.DrawImage(Image.FromFile(str), 10, 10, 607, 452); //使用 DrawImage 方法绘制图像
}

```

程序运行结果如图 17.32 所示。



图 17.32 PrintPreviewControl 组件的应用

### 17.4.5 使用 PrintPreviewDialog 组件显示打印预览

PrintPreviewDialog 组件用于显示文档打印后的外观，该组件包含打印、放大、显示一页或多页和关闭此对话框的按钮。PrintPreviewDialog 组件的常见属性和方法有 Document 属性、UseAntiAlias 属性和 ShowDialog 方法，下面分别进行介绍。

#### (1) Document 属性

该属性用于设置要预览的文档。

语法：public PrintDocument Document { get; set; }

说明：它的属性值为 PrintDocument 对象，表示要预览的文档。

例 17.12 设置 PrintPreviewDialog 组件的 Document 属性为 printDocument1，代码如下。

```
printPreviewDialog1.Document = this.printDocument1; //设置预览文档
```

#### (2) UseAntiAlias 属性

该属性用于设置打印是否使用操作系统的防锯齿功能。

语法：public bool UseAntiAlias { get; set; }

说明：它的属性值表示，如果使用防锯齿功能，则为 true；否则为 false。

例 17.13 设置 UseAntiAlias 属性为 true，开启防锯齿功能，代码如下。

```
printPreviewDialog1.UseAntiAlias = true; //设置 UseAntiAlias 属性为 true，开启防锯齿功能
```

#### (3) ShowDialog 方法

该方法用于显示打印预览对话框。

语法：public DialogResult ShowDialog()

说明：它的返回值为 DialogResult 枚举值之一。

例 17.14 调用 PrintPreviewDialog 组件的 ShowDialog 方法，显示打印预览窗口，代码如下。

```
printPreviewDialog1.ShowDialog(); //使用 ShowDialog 方法显示预览窗口
```

## 17.5 照猫画虎——基本功训练

### 17.5.1 基本功训练 1——自定义横向打印

视频讲解：光盘\mr\17\lx\自定义横向打印.exe

实例位置：光盘\mr\17\zmhh\01

新建一个 Windows 窗体应用程序，向窗体中添加一个 DataGridView 控件，用来显示要打印的数据；添加一个 Panel 控件，用于预览打印方向；添加一个 ComboBox 控件，用于选择打印纸张；添加一个 Button 控件，用来执行打印预览操作。

单击“打印预览”按钮，调用公共类 PrintClass 中的构造函数对打印信息进行设置，然后调用该类中的 print 方法打印数据。“打印预览”按钮的 Click 事件代码如下。

```
private void button_Preview_Click(object sender, EventArgs e)
{
    //对打印信息进行设置
    PrintClass dgp = new PrintClass(this.dataGridView1, comboBox.PageSize.SelectedIndex, checkBox_Aspect.Checked);
    MSetUp(dgp); //记录窗体中打印信息的相关设置
```

```

string[] header = new string[dataGridView1.ColumnCount];           //创建一个与数据列相等的字符串数组
for (int p = 0; p < dataGridView1.ColumnCount; p++)             //记录所有列标题的名列
{
    header[p] = dataGridView1.Columns[p].HeaderCell.Value.ToString();
}
dgp.print();                                                       //显示打印预览窗体
}

```

在上面的代码中用到了公共类 PrintClass 中的构造函数和 print 方法，下面对它们进行详细讲解。PrintClass 类的构造函数主要用来对打印信息进行初始化，它有 3 个参数，分别用来表示打印数据、纸张大小和是否横向打印。PrintClass 类的构造函数实现代码如下。

```

public PrintClass(DataGridView datagrid, int PageS, bool landscape)
{
    this.datagrid = datagrid;                                     //获取打印数据
    this.PageSheet = PageS;                                       //纸张大小
    printdocument = new PrintDocument();                          //创建 PrintDocument 类
    pagesetupdialog = new PageSetupDialog();                      //创建 PageSetupDialog 类
    pagesetupdialog.Document = printdocument;                   //获取当前页的设置
    printpreviewdialog = new PrintPreviewDialog();                //创建 PrintPreviewDialog 类
    printpreviewdialog.Document = printdocument;                 //获取预览文档的信息
    printpreviewdialog.FormBorderStyle = FormBorderStyle.Fixed3D; //设置窗体的边框样式
    //横向打印的设置
    if (PageSheet >= 0)
    {
        if (landscape == true)
        {
            printdocument.DefaultPageSettings.Landscape = landscape; //横向打印
        }
    }
    pagesetupdialog.Document = printdocument;                   //设置打印文档
    printdocument.PrintPage += new PrintPageEventHandler(this.printdocument_Printpage); //事件的重载
}

```

print 方法为自定义的无返回值类型方法，该方法主要用来根据要打印的数据显示打印预览窗体。print 方法实现代码如下。

```

public void print()
{
    rowcount = 0;                                                 //记录数据的行数
    string paperName = Page_Size(PageSheet);                    //获取当前纸张的大小
    PageSettings storePageSetting = new PageSettings();          //创建一个 PageSettings 类的对象
    foreach (PaperSize ps in printdocument.PrinterSettings.PaperSizes)
        if (paperName == ps.PaperName)                           //查找当前设置纸张
            if (paperName == ps.PaperName)                         //如果找到当前纸张的名称
            {
                storePageSetting.PaperSize = ps;                  //获取当前纸张的信息
            }
    if (datagrid.DataSource.GetType().ToString() == "System.Data.DataTable") //判断数据类型
    {
        rowcount = ((DataTable)datagrid.DataSource).Rows.Count; //获取数据的行数
    }
    else if (datagrid.DataSource.GetType().ToString() == "System.Collections.ArrayList") //判断数据类型
    {

```

```

        rowCount = ((ArrayList)datagrid.DataSource).Count; //获取数据的行数
    }
    try
    {
        printdocument.DefaultPageSettings.Landscape = PageScape; //设置横向打印
        pagesetupdialog.Document = printdocument; //设置打印文档
        printpreviewdialog.ShowDialog(); //显示打印预览窗体
    }
    catch (Exception e)
    {
        throw new Exception("printer error." + e.Message);
    }
}

```

程序运行结果如图 17.33 所示。

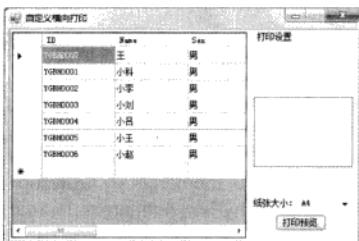


图 17.33 自定义横向打印

**熊猫画虎：**制作一个程序，主要实现自定义纵向打印的功能。提示：将 Printdocument 对象的 DefaultPageSettings.Landscape 属性设置为 false。(20 分)(实例位置：光盘\mr\17\zmhh\01\_zmhh)

### 17.5.2 基本功训练 2——自定义打印页码范围

■**视频讲解：**光盘\mr\17\lx\自定义打印页码范围.exe

■**实例位置：**光盘\mr\17\zmhh\02

新建一个 Windows 窗体应用程序，在窗体中添加一个 DataGridView 控件，用来显示要打印的数据；添加一个 TextBox 控件，用来输入打印的页码范围；添加两个 RadioButton 控件，分别用来设置全部打印和打印指定的页码；添加一个 PrintPreviewDialog 组件，用来显示打印预览对话框；添加一个 PrintDocument 组件，用来设置打印文档；添加一个 Button 控件，用来执行打印操作。

在打印文档之前，首先需要对要打印的文档进行设置，这时需要用到 PrintDocument 控件的 PrintPage 事件。在该事件中，首先判断是否有要打印的数据，如果有，则根据选中的单选按钮对数据进行打印。PrintDocument 控件的 PrintPage 事件代码如下。

```

private void printDocument1_PrintPage(object sender, System.Drawing.Printing.PrintPageEventArgs e)
{
    if (dataGridView1.Rows.Count > 0)
    {
        PrintPageWidth = e.PageBounds.Width; //获取打印纸张的宽度
        PrintPageHeight = e.PageBounds.Height; //获取打印纸张的高度
        #region 绘制边框线
        e.Graphics.DrawLine(myPen, leftmargin, topmargin, PrintPageWidth - leftmargin - rightmargin, topmargin);
    }
}

```

```

        e.Graphics.DrawLine(myPen, leftmargin, topmargin, leftmargin, PrintPageHeight - topmargin - buttonmargin);
        e.Graphics.DrawLine(myPen, leftmargin, PrintPageHeight - topmargin - buttonmargin, PrintPageWidth - leftmargin - rightmargin, PrintPageHeight - topmargin - buttonmargin);
        e.Graphics.DrawLine(myPen, PrintPageWidth - leftmargin - rightmargin, topmargin, PrintPageWidth - leftmargin - rightmargin, PrintPageHeight - topmargin - buttonmargin);
    #endregion
    #region 打印全部
    if (radioButton1.Checked)
    {
        int intPrintRows = currentpageindex * intRows;
        int j = 0;
        for (int i = 0 + (intPrintRows - 30); i < intPrintRows; i++)
        {
            if (i <= dataGridView1.Rows.Count - 2)
            {
                e.Graphics.DrawString(dataGridView1.Rows[i].Cells[0].Value.ToString(), myFont, myBrush,
leftmargin + 5, topmargin + j * rowgap + 5);
                e.Graphics.DrawString(dataGridView1.Rows[i].Cells[1].Value.ToString(), myFont, myBrush,
leftmargin + columnWidth1 + 5, topmargin + j * rowgap + 5);
                e.Graphics.DrawString(dataGridView1.Rows[i].Cells[2].Value.ToString(), myFont, myBrush,
leftmargin + columnWidth1 + columnWidth2 + 5, topmargin + j * rowgap + 5);
                e.Graphics.DrawLine(myPen, leftmargin, topmargin + j * rowgap + 1, PrintPageWidth -
leftmargin - rightmargin, topmargin + j * rowgap + 1);
                e.Graphics.DrawLine(myPen, leftmargin + columnWidth1, topmargin + j * rowgap, leftmargin +
columnWidth1, PrintPageHeight - topmargin - buttonmargin);
                e.Graphics.DrawLine(myPen, leftmargin + columnWidth1 + columnWidth2, topmargin + j *
rowgap, leftmargin + columnWidth1 + columnWidth2, PrintPageHeight - topmargin - buttonmargin);
                e.Graphics.DrawString("共 " + intPage + " 页 第 " + currentpageindex + " 页", myFont,
myBrush, PrintPageWidth - 200, (int)(PrintPageHeight - buttonmargin / 2));
                j++;
            }
        }
        currentpageindex++;
        if (currentpageindex <= intPage)
        {
            e.HasMorePages = true;
        }
        else
        {
            e.HasMorePages = false;
            currentpageindex = 1;
        }
    }
    #endregion
    else
    {
        #region 打印指定的一页
        if (page != 0)
        {
            if (page <= intPage)

```

//下一页的页码  
//如果当前页不是最后一页  
//打印副页  
//不打印副页  
//当前打印的页编号设为 1

```

    {
        int intPrintRows = page * intRows;
        int j = 0;
        for (int i = 0 + (intPrintRows - 30); i < intPrintRows; i++)
        {
            if (i <= dataGridView1.Rows.Count - 2)
            {
                e.Graphics.DrawString(dataGridView1.Rows[i].Cells[0].Value.ToString(), myFont,
myBrush, leftmargin + 5, topmargin + j * rowgap + 5);
                e.Graphics.DrawString(dataGridView1.Rows[i].Cells[1].Value.ToString(), myFont,
myBrush, leftmargin + columnWidth1 + 5, topmargin + j * rowgap + 5);
                e.Graphics.DrawString(dataGridView1.Rows[i].Cells[2].Value.ToString(), myFont,
myBrush, leftmargin + columnWidth1 + columnWidth2 + 5, topmargin + j * rowgap + 5);
                e.Graphics.DrawLine(myPen, leftmargin, topmargin + j * rowgap + 1,
PrintPageWidth - leftmargin - rightmargin, topmargin + j * rowgap + 1);
                e.Graphics.DrawLine(myPen, leftmargin + columnWidth1, topmargin + j *
rowgap, leftmargin + columnWidth1, PrintPageHeight - topmargin - bottomMargin);
                e.Graphics.DrawLine(myPen, leftmargin + columnWidth1 + columnWidth2, PrintPageHeight -
topmargin - bottomMargin, topmargin + j * rowgap, leftmargin + columnWidth1 + columnWidth2,
PrintPageHeight - topmargin - bottomMargin);
                e.Graphics.DrawString("共 " + intPage + " 页 第 " + page + " 页", myFont,
myBrush, PrintPageWidth - 500, (int)(PrintPageHeight - bottomMargin / 2));
                j++;
            }
        }
        page = 0;
    }
}

#region 打印指定的多页
else
{
    if (m < list.Count)
    {
        int startPage = Convert.ToInt32(list[m].ToString());
        int intPrintRows = startPage * intRows;
        int j = 0;
        for (int i = 0 + (intPrintRows - 30); i < intPrintRows; i++)
        {
            if (i <= dataGridView1.Rows.Count - 2)
            {
                e.Graphics.DrawString(dataGridView1.Rows[i].Cells[0].Value.ToString(), myFont,
myBrush, leftmargin + 5, topmargin + j * rowgap + 5);
                e.Graphics.DrawString(dataGridView1.Rows[i].Cells[1].Value.ToString(), myFont,
myBrush, leftmargin + columnWidth1 + 5, topmargin + j * rowgap + 5);
                e.Graphics.DrawString(dataGridView1.Rows[i].Cells[2].Value.ToString(), myFont,
myBrush, leftmargin + columnWidth1 + columnWidth2 + 5, topmargin + j * rowgap + 5);
                e.Graphics.DrawLine(myPen, leftmargin, topmargin + j * rowgap + 1,
PrintPageWidth - leftmargin - rightmargin, topmargin + j * rowgap + 1);
                e.Graphics.DrawLine(myPen, leftmargin + columnWidth1, topmargin + j *
rowgap, leftmargin + columnWidth1, PrintPageHeight - topmargin - bottomMargin);
                e.Graphics.DrawLine(myPen, leftmargin + columnWidth1 + columnWidth2, PrintPageHeight -
topmargin - bottomMargin, topmargin + j * rowgap, leftmargin + columnWidth1 + columnWidth2,
PrintPageHeight - topmargin - bottomMargin);
            }
        }
    }
}

```

单击“打印”按钮，首先判断是否指定页码范围，如果未指定，则直接进行全部打印；否则，将根据输入的页码范围打印指定页。“打印”按钮的 Click 事件代码如下。

```
private void button2_Click(object sender, EventArgs e)
{
    try
    {
        if (radioButton2.Checked)
        {
            if (textBox2.Text == "")
            {
                MessageBox.Show("请指定要打印的页码！", "警告", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                return;
            }
            else if (textBox2.Text.IndexOf(",") == -1)
            {
                if (textBox2.Text.IndexOf("-") == -1)
                {
                    page = Convert.ToInt32(textBox2.Text);
                    if (page > intPage)
                    {
                        MessageBox.Show("要打印的页码超过了文档的最大页，请重新设置！", "警告",
MessageButtons.OK, MessageBoxIcon.Warning);
                        textBox2.Text = "";
                        textBox2.Focus();
                        return;
                    }
                }
            }
        }
    }
}
```

```

    }
else
{
    string[] strSubPages = textBox2.Text.Split('-');
    int intStart = Convert.ToInt32(strSubPages[0].ToString());
    int intEnd = Convert.ToInt32(strSubPages[1].ToString());
    for (int j = intStart; j <= intEnd; j++)
        list.Add(j);                                //将要打印的页添加到集合中
    list.Sort();                                    //对 List 集合中的元素进行排序
}
}
else
{
    string[] strPages = textBox2.Text.Split(',');
    for (int i = 0; i < strPages.Length; i++)
    {
        if (strPages[i].IndexOf("-") == -1)
            list.Add(Convert.ToInt32(strPages[i]));
        else
        {
            string[] strSubPages = strPages[i].Split('-');
            int intStart = Convert.ToInt32(strSubPages[0].ToString());
            int intEnd = Convert.ToInt32(strSubPages[1].ToString());
            for (int j = intStart; j <= intEnd; j++)
                list.Add(j);
        }
    }
    list.Sort();      //对 list 集合中的元素排序
}
}
printPreviewDialog1.ShowDialog();
}
catch {}
}

```

程序运行结果如图 17.34 所示。

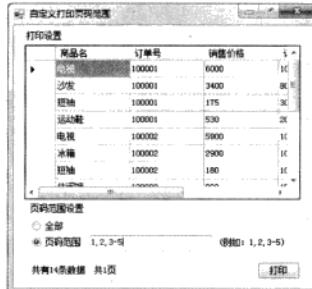


图 17.34 自定义打印页码范围

照猫画虎：根据以上程序的实现原理制作一个程序，用来实现只打印文档前 5 页的功能。(20 分)(实例位置：光盘\mr\17\zmhh\02\_zmhh)

### 17.5.3 基本功训练 3——打印商品入库单据

■ 视频讲解：光盘\mr\17\lx\打印商品入库单据.exe

■ 实例位置：光盘\mr\17\zmhh\03

新建一个 Windows 窗体应用程序，在窗体中添加一个 DataGridView 控件，用来显示要打印的商品入库单信息；添加 PrintDocument 和 PrintPreviewDialog 组件，分别用来设置打印文档和显示“打印预览”对话框；添加一个 Button 控件，用来执行打印商品入库单据操作。将商品入库单信息绘制到打印纸上的功能是在 PrintDocument 组件的 PrintPage 事件中实现的，代码如下。

```
private void printDocument1_PrintPage(object sender, System.Drawing.Printing.PrintPageEventArgs e)
{
    int printWidth = e.PageBounds.Width; // 定义打印纸宽度
    int printHeight = e.PageBounds.Height; // 定义打印纸高度
    int left = printWidth / 2 - 305; // 左边距
    int right = printWidth / 2 + 305; // 右边距
    int top = printHeight / 2 - 200; // 顶边距
    Brush myBrush = new SolidBrush(Color.Black); // 定义画刷
    Pen myPen = new Pen(Color.Black); // 定义画笔
    Font myFont = new Font("宋体", 12); // 定义字体
    // 绘制入库单标题
    e.Graphics.DrawString("商品入库单", new Font("宋体", 20, FontStyle.Bold), myBrush, new Point(printWidth / 2 - 100, top));
    e.Graphics.DrawLine(new Pen(Color.Black, 2), 300, top + 30, 480, top + 30);
    e.Graphics.DrawLine(new Pen(Color.Black, 2), 300, top + 34, 480, top + 34);
    e.Graphics.DrawString("吉林省明日科技有限公司", new Font("宋体", 9), myBrush, new Point(left + 2, top + 25));
    e.Graphics.DrawString("日期：" + DateTime.Now.ToString("yyyy-MM-dd"), new Font("宋体", 12), myBrush, new Point(right - 190, top + 25));
    e.Graphics.DrawRectangle(myPen, left, top + 42, 610, 230); // 绘制矩形框
    e.Graphics.DrawLine(myPen, left, top + 72, left + 610, top + 72); // 第一行
    e.Graphics.DrawLine(myPen, left, top + 102, left + 610, top + 102); // 第二行
    e.Graphics.DrawLine(myPen, left, top + 132, left + 610, top + 132); // 第三行
    e.Graphics.DrawLine(myPen, left, top + 162, left + 610, top + 162); // 第四行
    e.Graphics.DrawLine(myPen, left + 80, top + 42, left + 80, top + 272); // 第一列
    e.Graphics.DrawLine(myPen, left + 220, top + 42, left + 220, top + 72); // 第二列
    e.Graphics.DrawLine(myPen, left + 280, top + 42, left + 280, top + 72); // 第三列
    e.Graphics.DrawLine(myPen, left + 410, top + 42, left + 410, top + 132); // 第四列
    e.Graphics.DrawLine(myPen, left + 470, top + 42, left + 470, top + 162); // 第五列
    e.Graphics.DrawLine(myPen, left + 170, top + 102, left + 170, top + 162); // 第三行第二列
    e.Graphics.DrawLine(myPen, left + 220, top + 102, left + 220, top + 162); // 第三行第三列
    e.Graphics.DrawLine(myPen, left + 300, top + 132, left + 300, top + 162); // 第四行第四列
    e.Graphics.DrawLine(myPen, left + 360, top + 132, left + 360, top + 162); // 第四行第五列
    e.Graphics.DrawLine(myPen, left + 520, top + 132, left + 520, top + 162); // 第四行第七列
    // 第一行数据
    e.Graphics.DrawString("入库日期", myFont, myBrush, new Point(left + 2, top + 50));
    e.Graphics.DrawString(strInDate, myFont, myBrush, new Point(left + 82, top + 50));
    e.Graphics.DrawString("单据号", myFont, myBrush, new Point(left + 222, top + 50));
    e.Graphics.DrawString(strID, myFont, myBrush, new Point(left + 282, top + 50));
    e.Graphics.DrawString("入库人", myFont, myBrush, new Point(left + 412, top + 50));
    e.Graphics.DrawString(strInPeople, myFont, myBrush, new Point(left + 472, top + 50));
    // 第二行数据
}
```

```

e.Graphics.DrawString("供货商", myFont, myBrush, new Point(left + 2, top + 80));
e.Graphics.DrawString(strInProvider, myFont, myBrush, new Point(left + 82, top + 80));
e.Graphics.DrawString("产地", myFont, myBrush, new Point(left + 412, top + 80));
e.Graphics.DrawString(strPlace, myFont, myBrush, new Point(left + 472, top + 80));
//第三行数据
e.Graphics.DrawString("商品编号", myFont, myBrush, new Point(left + 2, top + 110));
e.Graphics.DrawString(strGID, myFont, myBrush, new Point(left + 82, top + 110));
e.Graphics.DrawString("名称", myFont, myBrush, new Point(left + 172, top + 110));
e.Graphics.DrawString(strGName, myFont, myBrush, new Point(left + 222, top + 110));
e.Graphics.DrawString("规格", myFont, myBrush, new Point(left + 412, top + 110));
e.Graphics.DrawString(strGSpec, myFont, myBrush, new Point(left + 472, top + 110));
//第四行数据
e.Graphics.DrawString("单位", myFont, myBrush, new Point(left + 2, top + 140));
e.Graphics.DrawString(strGUnit, myFont, myBrush, new Point(left + 82, top + 140));
e.Graphics.DrawString("单价", myFont, myBrush, new Point(left + 172, top + 140));
e.Graphics.DrawString(strGMoney, myFont, myBrush, new Point(left + 222, top + 140));
e.Graphics.DrawString("数量", myFont, myBrush, new Point(left + 302, top + 140));
e.Graphics.DrawString(strGNum, myFont, myBrush, new Point(left + 362, top + 140));
e.Graphics.DrawString("金额", myFont, myBrush, new Point(left + 472, top + 140));
e.Graphics.DrawString(strSMoney, myFont, myBrush, new Point(left + 522, top + 140));
//第五行数据
e.Graphics.DrawString("备注", myFont, myBrush, new Point(left + 2, top + 170));
e.Graphics.DrawString(strRemark, myFont, myBrush, new Point(left + 82, top + 170));
}

```

程序运行结果如图 17.35 所示。

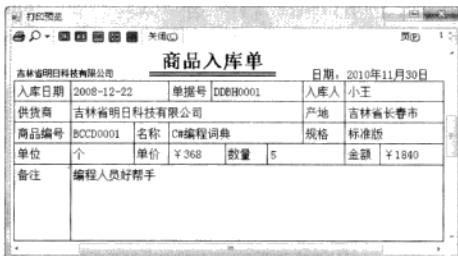


图 17.35 打印商品入库单据

**照猫画虎：**制作一个程序，用来实现打印商品出库单据的功能。提示：主要用到 tb\_OutGoods 数据表。  
**(20 分)** (实例位置：光盘\mr\17\zmhh\03\_zmhh)

#### 17.5.4 基本功训练 4——使图片成为整个报表的背景

■**视频讲解：**光盘\mr\17\lx\使图片成为整个报表的背景.exe

■**实例位置：**光盘\mr\17\zmhh\04

本实例主要实现将图片设置为整个报表背景的功能，其实现过程如下。

(1) 创建一个 Windows 窗体应用程序。

(2) 在当前项目中创建一个名为 CrystalReport1.rpt 的水晶报表，并连接 SQL Server 数据源。创建步骤可参见 17.3.1 节的相关内容。

(3) 使用鼠标右键单击水晶报表“页眉”节的空白处，在弹出的快捷菜单中选择“插入”/“图片”命令，在“页眉”处插入图片。

(4) 使用鼠标右键单击“页眉”节的标题轴，从弹出的快捷菜单中选择“节专家”命令，如图 17.36 所示。

(5) 弹出“节专家”对话框，如图 17.37 所示。在该对话框中选取“页眉”节，并选中“延伸到后续节”复选框，该操作会让页眉节的后续节往上平移，按照顺序打印在页眉节之上，只要添加在页眉节中的图片足够大，就会成为整张报表的背景图片。



图 17.36 选择“节专家”命令



图 17.37 “节专家”对话框

(6) 在窗体中添加一个 CrystalReportViewer 控件，并设置其 ReportSource 属性为要显示的水晶报表。实例运行效果如图 17.38 所示。



图 17.38 使图片成为整个报表的背景

**照猫画虎：**制作一个程序，主要实现设置水晶报表中节背景图片的功能。提示：通过在水晶报表的节中插入图片实现。(20 分)(实例位置：光盘\mr\17\zmhh\04\_zmhh)

## 17.5.5 基本功训练 5——设置水晶报表的打印日期与时间

■**视频讲解：**光盘\mr\17\lx\设置水晶报表的打印日期与时间.exe

■**实例位置：**光盘\mr\17\zmhh\05

本实例主要实现设置水晶报表的打印日期与时间的功能，其实现过程如下。

(1) 创建一个 Windows 窗体应用程序。

(2) 在当前项目中创建一个名为 MyCrystalReport.rpt 的水晶报表，并连接 SQL Server 数据源。创建步骤可参见 17.3.1 节的相关内容。

(3) 使用鼠标右键单击任意报表的空白处，并在弹出的快捷菜单中选择“报表”/“设置打印日期和时间”命令，如图 17.39 所示。

(4) 弹出“设置打印日期和时间”对话框，如图 17.40 所示。在此对话框中即可设置打印日期和时间。



图 17.39 选择“设置打印日期和时间”命令



图 17.40 “设置打印日期和时间”对话框

(5) 在设置打印日期和时间后，可以在字段资源管理器中将“打印日期”与“打印时间”字段拖放到水晶报表中，如图 17.41 所示。

(6) 在窗体中添加一个 CrystalReportViewer 控件，并设置其 ReportSource 属性为要显示的水晶报表。实例运行效果如图 17.42 所示。



图 17.41 字段资源管理器

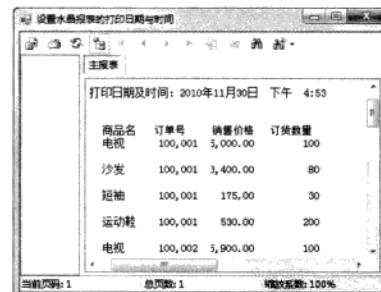


图 17.42 设置水晶报表的打印日期与时间

**照猫画虎：**制作一个程序，用来在水晶报表打印时显示最新的修改日期及时间。提示：通过在“字段资源管理器”中拖放“修改日期”和“修改时间”实现。(20 分)(实例位置：光盘\mr\17\zmhh\05\_zmhh)

**照猫画虎栏目分数统计：**

照猫画虎题目	1	2	3	4	5	总分数
分数						

## 17.6 情景应用——拓展与实践

### 17.6.1 情景应用 1——打印学生个人简历

视频讲解：光盘\mr\17\lx\打印学生个人简历.exe

实例位置：光盘\mr\17\qjyy\01

毕业生毕业之后的第一步就是找工作，而在找工作之前的一个重要任务就是制作一份精美的个人简历，本实例使用 Windows 打印组件实现打印学生个人简历的功能。运行效果如图 17.43 所示。

实现过程如下。

(1) 创建一个 Windows 窗体应用程序，并将其命名为 PrintStuResume。

(2) 在 PrintStuResume 项目中添加一个新窗体，并将其命名为 frmPrint，主要用来实现打印学生个人简历的功能。在 frmPrint 窗体中添加 23 个 Label 控件，分别用来显示学生的基本信息；添加一个 PictureBox 控件，用来显示学生的头像信息；添加一个 PrintPreviewDialog 打印组件，用来显示打印预览对话框；添加一个 PrintDocument 组件，用来设置打印文档；添加一个 Button 控件，用来执行打印操作。

(3) 当 frmPrint 窗体加载时，首先将要打印的学生的详细信息显示在相应的 Label 控件中，同时将要打印的学生的头像显示在 PictureBox 控件中，代码如下。

```
private void frmPrint_Load(object sender, EventArgs e)
{
    labName.Text = frmMain.strName; //显示学生姓名
    labSex.Text = frmMain.strSex; //显示学生性别
    labNPlace.Text = frmMain.strNPlace; //显示学生籍贯
    labBirthday.Text = Convert.ToDateTime(frmMain.strBirthday).ToShortDateString(); //显示学生出生年月
    labNation.Text = frmMain.strNation; //显示学生民族
    labSGao.Text = frmMain.strSGao; //显示学生身高
    labTZhong.Text = frmMain.strTZhong; //显示学生体重
    labHunYin.Text = frmMain.strHunYin; //显示学生婚否
    labXUELI.Text = frmMain.strXUELI; //显示学生学历
    labAddress.Text = frmMain.strAddress; //显示学生家庭地址
    labZY.Text = frmMain.strZHUANYE; //显示学生专业
    labYX.Text = frmMain.strBYYX; //显示学生毕业院校
    labBYSJ.Text = Convert.ToDateTime(frmMain.strBYSJ).ToShortDateString(); //显示学生毕业时间
    labWY.Text = frmMain.strWAIYU; //显示学生外语水平
    labTel.Text = frmMain.strTel; //显示学生联系电话
    labEmail.Text = frmMain.strEmail; //显示学生 E-mail
    labZW.Text = frmMain.strYPZW; //显示学生应聘职位
}
```

个人简历					
姓名	小王	性别	男	籍贯	山西省
出生年月	1985/6/8	民族	汉族	身高	178cm
体重	60kg	婚否	否	学历	大专
家庭地址 吉林省长春市东盛大街					
专业	可视化编程技术	毕业院校	长春***学院		
毕业时间	2006/6/1	外语水平	英语		
联系电话	13100000000	Email	xiaowang@163.com		
应聘职位	C# 工程师	求职类型	全职		
期望工作地点	长春	期望薪水	2000	工作年限	2 年
工作经历	吉林省明日科技有限公司				
特长					
C# WinForm 应用程序、网站开发					

图 17.43 打印学生个人简历

```

labQZLX.Text = frmMain.strQZLX; //显示学生求职类型
labPlace.Text = frmMain.strGZDQ; //显示学生期望工作地点
labSalary.Text = frmMain.strSalary; //显示学生期望薪水
labYear.Text = frmMain.strGZNX; //显示学生工作年限
labGZJL.Text = frmMain.strGZJL; //显示学生工作经历
labTC.Text = frmMain.strTECHANG; //显示学生特长
pictureBox1.Image = frmMain.imgPhoto; //显示学生头像
}
}

```

(4) 在打印学生个人简历之前,首先需要对打印纸张进行设置,这时需要用到 PrintDocument 控件的 PrintPage 事件。在该事件中,将要打印的学生个人简历的位图绘制到打印纸张上。PrintDocument 控件的 PrintPage 事件代码如下。

```

private void printDocument1_PrintPage(object sender, System.Drawing.Printing.PrintPageEventArgs e)
{
    e.Graphics.DrawImage(memoryImage, 0, 0); //打印学生简历
}

```

(5) 单击“打印”按钮,首先在“打印预览”对话框中显示要打印的学生个人简历,以方便对其进行打印。“打印”按钮的 Click 事件代码如下。

```

private void button1_Click(object sender, EventArgs e)
{
    button1.Visible = false; //将“打印”按钮设置为不可用
    CaptureScreen(); //获取简历图片
    printPreviewDialog1.ShowDialog(); //显示“打印预览”对话框
}

```

(6) 上面的代码中用到了 CaptureScreen 方法,该方法为自定义的无返回值类型方法,它主要用来将要打印的学生个人简历绘制成 Image 图像。CaptureScreen 方法的实现代码如下。

```

private Bitmap memoryImage; //声明 Bitmap 对象
private void CaptureScreen()
{
    Graphics mygraphics = panel1.CreateGraphics(); //创建 panel 绘图对象
    Size s = panel1.Size; //定义打印的图片大小
    memoryImage = new Bitmap(s.Width, s.Height, mygraphics); //创建 Bitmap 对象
    Graphics memoryGraphics = Graphics.FromImage(memoryImage); //创建窗体绘图对象
    IntPtr dc1 = mygraphics.GetHdc(); //获取 panel 句柄
    IntPtr dc2 = memoryGraphics.GetHdc(); //获取绘图对象句柄
    //绘制 panel 控件中的内容
    BitBlt(dc2, 0, 0, panel1.ClientRectangle.Width, panel1.ClientRectangle.Height, dc1, 0, 0, 13369376); //释放 panel 句柄资源
    mygraphics.ReleaseHdc(dc1); //释放绘图对象句柄资源
    memoryGraphics.ReleaseHdc(dc2);
}

```

**DIY:** 根据以上程序的实现原理制作一个程序,用来打印窗体上的文字。(20 分)(实例位置: 光盘\mr\17\qjyy\01\_diy)

## 17.6.2 情景应用 2——批量打印学生证书

视频讲解: 光盘\mr\17\lx\批量打印学生证书.exe

实例位置: 光盘\mr\17\qjyy\02

学校在打印学生证书时,通常都需要批量进行打印,本节使用 C#制作了一个批量打印学生证书的实例。

运行本实例，在主窗体的 DataGridView 控件中选中多行，单击“打印”按钮，则对选中的多个学生证信息进行批量打印。运行效果如图 17.44 所示。

实现过程如下。

(1) 创建一个 Windows 窗体应用程序，并将其命名为 PrintStuCertificate。

(2) 在默认窗体中添加一个 DataGridView 控件，用来显示学生信息；添加一个 PrintPreviewDialog 打印组件，用来显示打印预览对话框；添加一个 PrintDocument 组件，用来设置打印文档；添加一个 PageSetupDialog 控件，用来设置打印页码；添加一个 Button 控件，用来执行批量打印学生证书操作。

(3) 在打印学生证书之前，首先需要对打印纸张进行设置，这时需要用到 PrintDocument 控件的 PrintPage 事件。在该事件中，首先判断是否有要打印的学生信息，如果有，则给全局变量赋值，以便打印指定学生的学生证；如果没有，则打印空学生证。PrintDocument 控件的 PrintPage 事件代码如下。

```
private void printDocument1_PrintPage(object sender, System.Drawing.Printing.PrintPageEventArgs e)
{
    #region 记录的打印的学生信息
    if (lists.Count > 0)
    {
        DataSet myds = BindInfo(" 编 号 ", dataGridView1.Rows[Convert.ToInt32(lists[currentPage - 1])].Cells[0].Value.ToString());
        strID = dataGridView1.Rows[Convert.ToInt32(lists[currentPage - 1])].Cells[0].Value.ToString();
        strName = myds.Tables[0].Rows[0][1].ToString();
        strSex = myds.Tables[0].Rows[0][2].ToString();
        DateTime dt = Convert.ToDateTime(myds.Tables[0].Rows[0][3].ToString());
        strBirthday = dt.Year + "年" + dt.Month + "月";
        strNPlace = myds.Tables[0].Rows[0][4].ToString();
        strRXSJ = Convert.ToDateTime(myds.Tables[0].Rows[0][5].ToString()).ToString("yyyy-MM-dd");
        strZY = myds.Tables[0].Rows[0][6].ToString();
        strFZRQ = DateTime.Now.ToString("yyyy-MM-dd");
        MemoryStream memoryImage = new MemoryStream((byte[])myds.Tables[0].Rows[0][7]);
        imgPhoto = Image.FromStream(memoryImage);
    }
    #endregion
    int printWidth = e.PageBounds.Width;
    int printHeight = e.PageBounds.Height;
    //绘制矩形边框
    e.Graphics.DrawRectangle(mypen, 344, 236, 480, 355);
    e.Graphics.DrawRectangle(mypen, 374, 266, 193, 295);
    e.Graphics.DrawRectangle(mypen, 601, 266, 193, 295);
    //填充左侧内容
    e.Graphics.DrawLine(mypen, 404, 301, 404, 561); //第一列
    e.Graphics.DrawLine(mypen, 476, 301, 476, 396); //第二列
    e.Graphics.DrawLine(mypen, 374, 301, 567, 301); //第一行
    e.Graphics.DrawLine(mypen, 374, 333, 476, 333); //第二行
}
```

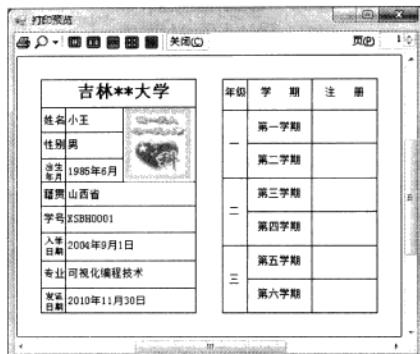


图 17.44 批量打印学生证书

```

e.Graphics.DrawLine(mypen, 374, 366, 476, 366);           //第三行
e.Graphics.DrawLine(mypen, 374, 396, 567, 396);           //第四行
e.Graphics.DrawLine(mypen, 374, 426, 567, 426);           //第五行
e.Graphics.DrawLine(mypen, 374, 460, 567, 460);           //第六行
e.Graphics.DrawLine(mypen, 374, 495, 567, 495);           //第七行
e.Graphics.DrawLine(mypen, 374, 530, 567, 530);           //第八行
e.Graphics.DrawString("吉林**大学", new Font("宋体", 16, FontStyle.Bold), myBrush, 415, 270);
e.Graphics.DrawString("姓名", myFont, myBrush, 375, 310);
e.Graphics.DrawString(strName, myFont, myBrush, 405, 310);
e.Graphics.DrawString("性别", myFont, myBrush, 375, 342);
e.Graphics.DrawString(strSex, myFont, myBrush, 405, 342);
e.Graphics.DrawString("出生", new Font("宋体", 8), myBrush, 377, 371);
e.Graphics.DrawString("年月", new Font("宋体", 8), myBrush, 377, 384);
e.Graphics.DrawString(strBirthday, myFont, myBrush, 405, 375);
e.Graphics.DrawString("籍贯", myFont, myBrush, 375, 405);
e.Graphics.DrawString(strNPlace, myFont, myBrush, 405, 405);
e.Graphics.DrawString("学号", myFont, myBrush, 375, 435);
e.Graphics.DrawString(strID, myFont, myBrush, 405, 435);
e.Graphics.DrawString("入学", new Font("宋体", 8), myBrush, 377, 465);
e.Graphics.DrawString("日期", new Font("宋体", 8), myBrush, 377, 478);
e.Graphics.DrawString(strRXSJ, myFont, myBrush, 405, 469);
e.Graphics.DrawString("专业", myFont, myBrush, 375, 504);
e.Graphics.DrawString(strZY, myFont, myBrush, 405, 504);
e.Graphics.DrawString("发证", new Font("宋体", 8), myBrush, 377, 535);
e.Graphics.DrawString("日期", new Font("宋体", 8), myBrush, 377, 548);
e.Graphics.DrawString(strFZRQ, myFont, myBrush, 405, 539);
if (imgPhoto != null)
    e.Graphics.DrawImage(imgPhoto, 479, 303, 86, 93);
//填充右侧内容
e.Graphics.DrawLine(mypen, 632, 266, 632, 561);           //第一列
e.Graphics.DrawLine(mypen, 713, 266, 713, 561);           //第二列
e.Graphics.DrawLine(mypen, 601, 306, 794, 306);           //第一行
e.Graphics.DrawLine(mypen, 601, 391, 794, 391);           //第二行
e.Graphics.DrawLine(mypen, 601, 476, 794, 476);           //第三行
e.Graphics.DrawString("年级", myFont, myBrush, 602, 276);
e.Graphics.DrawString("学期", myFont, myBrush, 646, 276);
e.Graphics.DrawString("注册", myFont, myBrush, 727, 276);
e.Graphics.DrawString("一", myFont, myBrush, 607, 341);
e.Graphics.DrawString("二", myFont, myBrush, 607, 426);
e.Graphics.DrawString("三", myFont, myBrush, 607, 511);
e.Graphics.DrawLine(mypen, 632, 348, 794, 348);           //分割第一学期
e.Graphics.DrawLine(mypen, 632, 433, 794, 433);           //分割第二学期
e.Graphics.DrawLine(mypen, 632, 518, 794, 518);           //分割第三学期
e.Graphics.DrawString("第一学期", myFont, myBrush, 642, 320);
e.Graphics.DrawString("第二学期", myFont, myBrush, 642, 362);
e.Graphics.DrawString("第三学期", myFont, myBrush, 642, 404);
e.Graphics.DrawString("第四学期", myFont, myBrush, 642, 446);
e.Graphics.DrawString("第五学期", myFont, myBrush, 642, 488);
e.Graphics.DrawString("第六学期", myFont, myBrush, 642, 530);
currentPage++;                                         //下一页的页码
if (currentPage <= lists.Count)                         //如果当前页不是最后一页

```

```

    {
        e.HasMorePages = true; //打印副页
    }
    else
    {
        e.HasMorePages = false; //不打印副页
        currentPage = 1; //当前打印的页编号设为 1
    }
}

```

(4) 单击“打印”按钮，判断是否选择了学生信息，如果已经选择，则将学生信息添加到一个 ArrayList 集合中，以便批量打印，同时设置打印方向为横向打印，最后显示“打印预览”对话框。“打印”按钮的 Click 事件代码如下。

```

private void button1_Click(object sender, EventArgs e)
{
    foreach (DataGridViewRow row in dataGridView1.Rows)
    {
        if (row.Selected)
            lists.Add(row.Index); //循环遍历 DataGridView 控件中的行
    }
    lists.Sort(); //判断行是否被选中
    pageSetupDialog1.PageSettings.Landscape = true; //将选中的行索引添加到集合中
    printPreviewDialog1.ShowDialog(); //对集合进行排序
} //设置横向打印
//显示“打印预览”对话框

```

**DIY：**制作一个程序，要求每次只能打印一个空学生证书。(20 分)(实例位置：光盘\mr\17\qjyy\02\_diy)

### 17.6.3 情景应用 3——订货总金额超过 10 万元显示“恭喜获奖”文字

**■ 视频讲解：**光盘\mr\17\lx\订货总金额超过 10 万元显示“恭喜获奖”文字.exe

**■ 实例位置：**光盘\mr\17\qjyy\03

商场如战场，为了增加公司效益，商家们想方设法争取一切能够利用的手段来吸引广大客户，其中给予某类客户一定奖励是商户吸引客户的一种手段。在对客户进行统计时，为了更方便地查看中奖用户，可以通过在统计报表中设置特殊颜色字段来进行标明，这样使客户查看数据时一目了然。本实例主要在水晶报表中实现订货总金额超过 10 万元时显示“恭喜获奖”文字的功能，运行效果如图 17.45 所示。

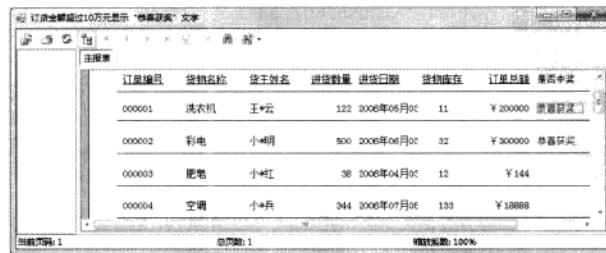


图 17.45 订货总金额超过 10 万元显示“恭喜获奖”文字

实现过程如下。

(1) 创建一个 Windows 窗体应用程序，并将其命名为 DisplayGetPrize。

(2) 创建一个名为 CrystalReport1.rpt 的水晶报表，并连接 SQL Server 数据源。创建步骤可参见 17.3.1 节的相关内容。

(3) 在水晶报表的“详细资料”栏中插入一个新文本对象，编辑文本为“恭喜获奖”。当文本编辑完成后，在文本对象上单击鼠标右键，在弹出的快捷菜单中选择“设置对象格式”命令，如图 17.46 所示。在打开的“格式编辑器”对话框的“字体”选项卡中设置文本颜色为红色，如图 17.47 所示。

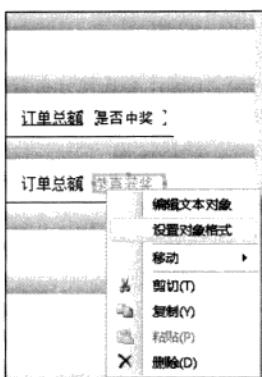


图 17.46 选择“设置对象格式”命令

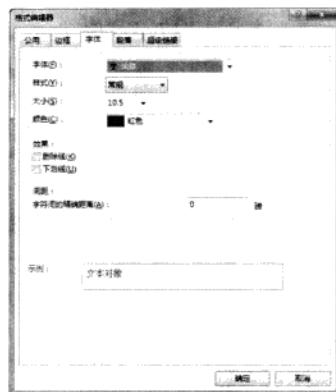


图 17.47 “格式编辑器”对话框

(4) 选择“格式编辑器”对话框中的“公用”选项卡，选中“抑制显示”复选框。单击该复选框右侧的图标按钮，打开如图 17.48 所示的对话框。在该对话框的工具栏处的下拉列表框中选择“Basic 语法”选项，并在对话框的右下空白部分编写如下代码。

```
formula = True
If {tb_OrderForm.订单总额} > 100000 Then
    formula = False
End If
```

(5) 单击“格式公式编辑器”工具栏中的“保存”或“保存并关闭”按钮，若没有任何提示，则表示该公式编辑成功；若出现如图 17.49 所示的错误提示框，则重新编辑该公式。

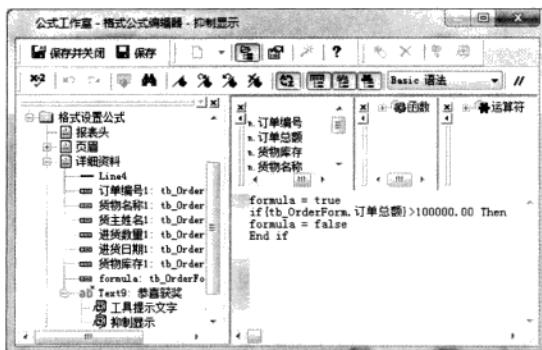


图 17.48 “格式公式编辑器”对话框



图 17.49 公式编辑错误提示框

(6) 单击“格式编辑器”对话框中的“确定”按钮，完成对该字段的编辑。

**DIY：**修改以上程序，使订货总金额超过 10 万元的客户以蓝色字体显示“恭喜获奖”文字。提示：在“格式编辑器”对话框中修改字体颜色即可。(20 分)(实例位置：光盘\mr\17\qjyy\03\_diy)

### 17.6.4 情景应用 4——部门销售量占公司总销售量的业绩百分比

■**视频讲解：**光盘\mr\17\lx\部门销售量占公司总销售量的业绩百分比.exe

■**实例位置：**光盘\mr\17\qjyy\04

在对大量数据进行分析时，最直观形象的表现形式莫过于使用比例显示，例如，统计商场的月销售量占全年销售量的百分比，统计全国男女比例情况等，诸如此类的数据对比统计形式使分析数据更加清晰。本实例主要统计部门销售量占公司总销售量的业绩百分比，运行效果如图 17.50 所示。

The screenshot shows a Windows application window with a title bar '部门销售量占公司总销售量的业绩百分比'. Inside, there's a report grid with columns: '商品编码', '商品名称', '所属部门', '销售数量', '货物价值', and '部门销售量'. Below the grid, a summary row displays '部门销售量占公司总销售量的业绩百分比: 75.32%' and '部门销售量占公司总销售量的业绩百分比: 20.68%'. At the bottom, status bars show '显示页数: 1', '总页数: 1', and '缩放级别: 100%'. On the left, a sidebar lists categories: '家用电器', '厨具', '日用品', '学习用品', and '家用厨具'.

图 17.50 部门销售量占公司总销售量的业绩百分比

实现过程如下。

- (1) 创建一个 Windows 窗体应用程序，并将其命名为 GetPercent。
- (2) 创建一个名为 MyCrystalReport.rpt 的水晶报表，并连接 SQL Server 数据源。创建步骤参见 17.3.1 节的相关内容。
- (3) 使用鼠标右键单击水晶报表中的空白位置，在弹出的快捷菜单中选择“插入”/“组”命令，在打开的“插入组”对话框中选择“所属部门”进行分组，如图 17.51 所示，单击“确定”按钮完成分组。
- (4) 使用鼠标右键单击水晶报表中的空白位置，在弹出的快捷菜单中选择“插入”/“汇总”命令，打开“插入汇总”对话框。在该对话框中进行如图 17.52 所示的设置，单击“确定”按钮，完成百分比统计。



图 17.51 “插入组”对话框



图 17.52 “插入汇总”对话框

**说明：**为了能够标明该数据的用途，可以在数据前插入文本对象对其进行说明。

**DIY：**制作一个程序，主要用来按部门统计各个部门的销售业绩。提示：将“插入汇总”对话框中的“以百分比显示”复选框设置为未选中。(20 分)(实例位置：光盘\mr\17\qjyy\04\_diy)

### 17.6.5 情景应用 5——按类别分组统计图书库存

■**视频讲解：**光盘\mr\17\lx\按类别分组统计图书库存.exe

■**实例位置：**光盘\mr\17\qjyy\05

在一些大型商场或书店中，由于商品种类繁多，在进行统计时需要进行分类，以方便工作人员更容易处理各类商品，从而提高工作人员的工作效率。本实例通过分组统计图书库存量来分析图书，运行效果如图 17.53 所示。



图 17.53 按类别分组统计图书库存

实现过程如下。

- (1) 创建一个 Windows 窗体应用程序，并将其命名为 GroupByClass。
- (2) 创建一个名为 MyCrystalReport.rpt 的水晶报表，并连接 SQL Server 数据源。创建步骤可参见 17.3.1 节的相关内容。
- (3) 在水晶报表设计器中的任意空白位置单击鼠标右键，在弹出的快捷菜单中选择“报表”/“组专家”命令，打开“组专家”对话框，如图 17.54 所示。从该对话框左侧的“报表字段”或者在表 tb\_CrystalReportBook 中选择合适的字段作为分组依据，这里以“书籍类别”作为分组依据。
- (4) 单击“组专家”对话框中的“选项”按钮，打开如图 17.55 所示的“更改组选项”对话框，在该对话框中设置或更改分组字段。



图 17.54 “组专家”对话框

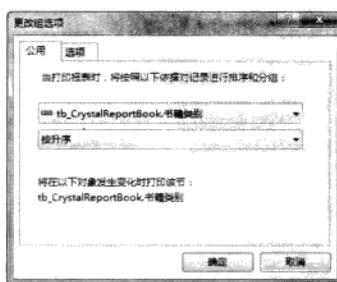


图 17.55 “更改组选项”对话框

- (5) 设置完成后，单击“组专家”对话框中的“确定”按钮，完成分组。

**DIY：**修改以上程序，使程序按照书籍编号分组统计图书库存。提示：在“组专家”对话框中修改分组字段即可。(20 分)(实例位置：光盘\mr\17\qjyy\05\_diy)

情景应用 DIY 栏目分数统计：

DIY 题目	1	2	3	4	5	总分数	
分数							

## 17.7 自我测试

### 一、选择题(每题 10 分, 5 道题)

1. 关于水晶报表的优点，以下描述错误的是（ ）。
 

A. 简单易用的界面	B. 更容易创建复杂的报表
C. 静态设置数据源	D. 降低了编码量
2. CrystalReports 水晶报表可以用各种形式发布，不包括以下哪一项（ ）。
 

A. Word 文档形式	B. Excel 形式	C. 电子邮件	D. Web Service 服务
--------------	-------------	---------	-------------------
3. Crystal 报表设计区域不含下列哪一项（ ）。
 

A. 报表页眉	B. 子报表	C. 详细资料	D. 报表页脚
---------	--------	---------	---------
4. Crystal 水晶报表通过数据库驱动程序与数据库进行连接，用户可根据相应的数据源中的数据进行报表设计，下列选项中不能作为水晶报表数据源的是（ ）。
 

A. 使用 ODBC 驱动程序的任何数据库 (RDO)	B. 使用 OLEDB 驱动程序的任何数据库 (ADO)
C. Microsoft Access 数据库或 Excel 工作簿 (DAO)	D. ADO 记录集
5. 在对数据进行排序过程中，一般遵循以下规则，其中描述错误的是（ ）。
 

A. “布尔值”升序排序	B. “日期/时间”升序排序
C. “数字/货币”升序排序	D. “字母”升序排序

### 二、填空题(每题 10 分, 5 道题)

1. 水晶报表的数据访问模式可以分为提取模式与（ ）两种。
2. 一般情况下，水晶报表可以通过如下两种方式来筛选记录，即利用（ ）筛选记录和开发人员自定义公式筛选记录。
3. 所谓的子报表也就是内含于报表中的报表，而内含子报表的报表则称为（ ）。
4. 子报表的应用大体上可分为两种，分别为（ ）和按需要显示子报表。
5. （ ）组件用来设置打印文档。

测试分数统计：

类别	第 1 题	第 2 题	第 3 题	第 4 题	第 5 题
选择题分数					
填空题分数					
总分数					

## 17.8 行 动 指 南

开始日期: 年 月 日

结束日期: 年 月 日

序号	内 容	行 动 指 南	
1	照猫画虎栏目 分数( )	分数>75 分	优秀, 基本功掌握得不错, 加油!
		75 分>分数>50 分	及格, 知识掌握得不牢, 重新做一遍照猫画虎。
		分数<50 分	请使用光盘中提供的“实战能力测试系统”进行提高训练。
	情景应用栏目 分数( )	分数>75 分	优秀, 综合应用能力很强。
		75 分>分数>50 分	及格, 综合应用能力需提高, 再练一遍情景应用。
		分数<50 分	请使用光盘中提供的“实战能力测试系统”进行提高训练。
2	自我测试栏目 分数( )	分数>75 分	优秀, 有成为编程高手的潜质。
		分数<75 分	请使用光盘中提供的“实战能力测试系统”进行提高训练。
综合评价		反复训练后, 以上各项分数都在 75 分以上, 方可进入下一堂课学习。	
3	编程能力培养: 本栏目提供了一些实践题目, 对于培养编程能力很有效, 要做好。	1. 开发一个分页打印的程序, 要求每页只能打印 20 条记录。	
		2. 开发一个程序, 主要实现动态绑定水晶报表的功能。	
		3. 开发一个程序, 用来通过水晶报表体现商品销售总量占全年总销售量的百分比。	
4	编程习惯培养: 本堂课培养读者在学习开发中记笔记的习惯, 把开发中遇到的问题、总结的经验记录下来。		
4	创新能力培养: 看看身边有没有可以用编程解决的问题, 记录到右边的表格中。根据学习进度, 尝试编写解决这些问题的小程序。		

## 17.9 成功可以复制——3D 王国的开创者约翰·沃克

约翰·沃克曾经是一位程序员, 在生产 UNIVAC (世界上第一台大型商用计算机) 的公司中担任系统程序设计师。那时还没有 PC 和互联网, 开发和使用价格昂贵的大型计算机只是少数人的专利。在工作之余, 沃克改良了当时非常受欢迎的 Animal 游戏, 改良后的游戏立刻引起热烈回响, 很多人都向他索取游戏磁盘, 这促使他想出了“游戏自我复制”的游戏传播方式。1975 年 1 月, 沃克编写了一个自我复制的程序 Pervade, 它可以在游戏启动时自动检查目录, 然后自我复制到任何一个尚未有拷贝的目录中, 或者覆盖已有的旧版游戏, 可以说, Pervade 程序算是病毒的前身。

20 世纪 70 年代, 随着微型计算机的出现, 有一小部分人意识到这将会成为计算机的主流, 那么怎样才能抓住这个机遇呢? 1977 年, 约翰·沃克和朋友成立了 Marinchip 公司, 试图开发微型计算机硬件系统, 但

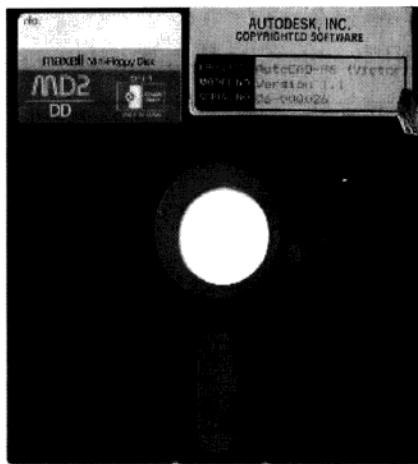
因 IBM 率先推出 PC 机（与其类似的微型计算机硬件），再加上运营资金不足，约翰·沃克放弃了走硬件发展的道路。

面对变化的市场，沃克决定调整战略，转而为市面上流行的PC机操作系统开发应用软件。在尝试了Basic编辑器、C编辑器及数据库软件后，沃克把目光锁定在绘图程序上。当时，市场上只有适用于大中型计算机的CAD绘图软件，且价格昂贵，而微型机上还缺少成熟的CAD软件。由于中小企业买不起大型计算机，只能依靠手工绘制工程图样，劳动强度大，工作效率低。

1982年1月，约翰·沃克和朋友合伙成立了Autodesk公司，购买了别人开发的Interact软件，重新改写并丰富其功能。12月，推出用于S-100和Z-80计算机的AutoCAD 1.0版。从此，AutoCAD一发而不可收，陆续推出了适用于各种操作系统的版本。Autodesk公司资产也在短短的7年内从最初的不足6万美元增长到6亿美元！这一切全依赖于约翰·沃克敏锐的洞察力。如今AutoCAD已成为世界排名第一的通用工程绘图软件。

越来越重的管理任务让喜欢平静生活的约翰·沃克有些痛苦，1989年，约翰·沃克交出管理权，之后他离开美国定居瑞士。不过，他现在还保留着编程的习惯，与硅谷最早的黑客群体保持着联系。

在风起云涌的 PC 时代，约翰·沃克成就了自己的传奇。



AutoCAD 1.1 软件磁盘

经典语录

企业的成功是建立在技术卓越的产品上的，如果我们不能成为技术和性能的领导者，我们将无法在市场上生存。

### 深度评价

Autodesk 公司资产在短短的 7 年内从最初的不足 6 万美元增长到 6 亿美元！这一切全依赖于约翰·沃克敏锐的洞察力，从而把握住了市场的先机。善于识别与把握先机是极为重要的。互联网时代要想谋求发展，必须要有极强的发现新兴事物、发现现有事物发展方向的个人能力，否则只能跟在别人之后，很难有大的发展。

# 第 18 堂课

## 网络编程

( 视频讲解：152分钟)

计算机网络实现了多个计算机互联系统，相互连接的计算机之间彼此能够进行数据交流。网络应用程序就是在已连接的不同的计算机上运行的程序，这些程序相互之间可以交换数据。而编写网络应用程序，首先必须明确其要使用的网络协议，TCP/IP 协议是网络应用程序的首选。C#作为一种编程语言，它提供了对网络编程的全面支持，例如，开发人员可以通过 C#语言制作一个简单的局域网聊天室等。本堂课将详细讲解网络编程方面的相关知识。

学习摘要：

- ▶ 了解局域网与因特网的概念
- ▶ 了解常见的几种网络协议
- ▶ 理解端口及套接字
- ▶ 掌握 System.NET 命名空间下主要类的使用
- ▶ 掌握 System.NET.Sockets 命名空间下主要类的使用
- ▶ 掌握 System.NET.Mail 命名空间下主要类的使用

## 18.1 计算机网络基础

在学习网络编程技术之前，首先要对网络有个基本的了解，本节首先从介绍局域网和因特网开始，然后再介绍 3 种常用的网络协议、端口及套接字等概念。

### 18.1.1 局域网与因特网介绍

为了实现两台计算机的通信，必须要有一个网络线路连接这两台计算机，如图 18.1 所示。



图 18.1 服务器、客户机和网络

服务器是指提供信息的计算机或程序，客户机是指请求信息的计算机或程序，网络主要是用来连接服务器与客户机以实现两者相互通信的。但有时，在某个网络中很难将服务器与客户机区分开。通常所说的“局域网”（Local Area Network，LAN），就是一群通过一定形式连接起来的计算机，它可以由两台计算机组成，也可以由同一区域中的上千台计算机组成。由 LAN 延伸到更大的范围，这样的网络称为“广域网”（Wide Area Network，WAN）。而大家熟悉的因特网（Internet），就是由无数的 LAN 和 WAN 组成的。

### 18.1.2 网络协议介绍

网络协议规定了计算机之间连接的物理、机械（网线与网卡的连接规定）、电气（有效的电平范围）等特征以及计算机之间的相互寻址规则、数据发送冲突的解决、长数据如何分段传送与接收等。就如同不同的国家有不同的法律一样，目前网络协议也有多种，下面介绍几种常用的网络协议。

#### 1. IP 协议

“IP”其实是 Internet Protocol 的简称，由此明显可知它是一种“网络协议”。Internet 网络采用的协议是 TCP/IP 协议，其全称是 Transmission Control Protocol/Internet Protocol。Internet 依靠 TCP/IP 协议，在全球范围内实现不同硬件结构、不同操作系统、不同网络系统的互联。在 Internet 网上存在数以亿计的主机，每一个主机在网络上通过为其分配的 Internet 地址表示自己，这个地址就是 IP 地址。到目前为止，IP 地址用 4 个字节，也就是 32 位的二进制数来表示，称为 Ipv4。为了便于使用，通常取用每个字节的十进制数，并且每个字节之间用圆点隔开来表示 IP 地址，如 192.168.1.1。如今人们正在尝试使用 16 个字节来表示 IP 地址，这就是 Ipv6，但 Ipv6 目前还没有投入使用。

TCP/IP 模式是一种层次结构，共分为 4 层，分别为应用层、传输层、互联网层和主机到网络层，各层实现特定的功能，提供特定的服务和访问接口，并具有相对的独立性，如图 18.2 所示。

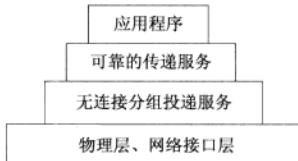


图 18.2 TCP/IP 层次结构

## 2. TCP 与 UDP 协议

在网络协议栈中，有两个高级协议是网络应用程序编写者应该了解的，分别是“传输控制协议”（Transmission Control Protocol，TCP）与“用户数据报协议”（User Datagram Protocol，UDP）。

TCP 协议是一种以固接连线为基础的协议，它提供两台计算机间可靠的数据传送，其可以保证从一端数据传送至连接的另一端时，数据能够确实送达，而且送达的数据的排列顺序和送出时的顺序相同，因此该协议适合可靠性要求较高的场合。就如同拨打电话一样，必须先拨号给对方，等两端确定连接后，才能相互听到对方说话，知道对方回应的是什么。

HTTP、FTP 和 Telnet 等都需要使用可靠的通信频道，例如 HTTP 从某个 URL 读取数据时，如果收到的数据顺序与发送时不相同，这样可能会出现一个混乱的 HTML 文件或一些无效的信息。

UDP 协议是无连接通信协议，其不保证可靠数据的传输，但能够向若干个目标发送数据，接收发自若干个源的数据，该协议以独立发送数据包的方式进行，这种方式就如同邮递员送信给收信人，可以寄出很多信给同一个人，而每一封信都是相对独立的，每封信送达的顺序并不重要，而收信人接收信件的顺序也不能保证与寄出信件的顺序相同。

UDP 协议适合于一些对数据准确性要求不高的场合，如网络聊天室、在线影片等；而由于 TCP 协议在认证上的额外耗费，因此有可能使传输速度减慢。UDP 协议可能会更适合这些对传输速度和时效要求非常高的网站，即使有一小部分数据包的遗失或传送顺序有所不同，但并不会严重危害该项通信。

**技巧：**一些防火墙和路由器会设置成不允许 UDP 数据包传输，因此若遇到 UDP 连接方面的问题，应先确定是否允许 UDP 协议。

## 3. POP3 协议

POP（Post Office Protocol 邮局协议）协议用于电子邮件的接收，目前常用第 3 版，所以称 POP3。通过 POP 协议，当客户机登录到服务器后，可以对自己的邮件进行删除，或是下载到本地。如表 18.1 所示为 POP3 协议的常用命令及描述。

表 18.1 POP3 协议的常用命令及描述

命 令	描 述
USER	此命令与下面的 pass 命令若都发送成功，将使状态转换
PASS	用户名所对应的密码
APOP	MD5 消息摘要
STAT	请求服务器发回关于邮箱统计资料（邮件总数和总字节数）
UIDL	回送邮件唯一标识符
LIST	回送邮件数量和每个邮件的大小
RETR	回送由参数标识的邮件的全部文本
DELE	服务器将由参数标识的邮件标记为删除，由 QUIT 命令执行
RSET	服务器将重置所有标记为删除的邮件，用于撤销 DELE 命令
TOP	服务器将回送由参数标识的邮件前 n 行内容，n 是正整数
NOOP	服务器返回一个肯定的响应，不做任何操作
QUIT	退出

**注意：**SMTP 服务器使用的端口号一般为 25，POP 服务器使用的端口号一般为 110。

在使用 POP3 协议实现电子邮件的接收功能时，首先需要配置 POP3 服务，步骤如下。

(1) 首先在“Windows 组件向导”对话框中选中“电子邮件服务”前的复选框，依次单击“下一步”

按钮，完成“电子邮件服务”的安装操作，如图 18.3 所示。

(2) 当添加完“电子邮件服务”后，打开“管理工具”窗口，这时将会出现一项新的功能“POP3 服务”，如图 18.4 所示。

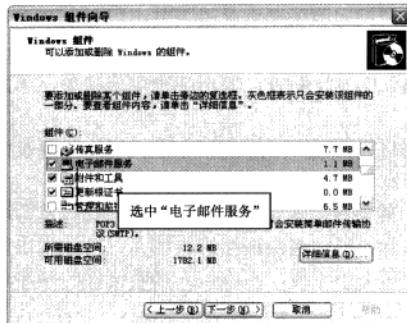


图 18.3 “Windows 组件向导”对话框



图 18.4 “管理工具”窗口

(3) 双击打开“POP3 服务”，在弹出的“添加域”对话框中添加一个新域（如 163.com），单击“确定”按钮，如图 18.5 所示。

(4) 添加完“域”后，选中相应的域，即可在弹出的“添加邮箱”对话框中添加邮箱名和密码，如图 18.6 所示。然后单击“确定”按钮，即可添加一个邮箱。

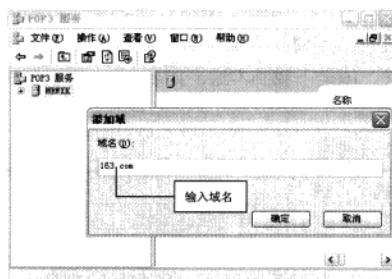


图 18.5 “添加域”对话框

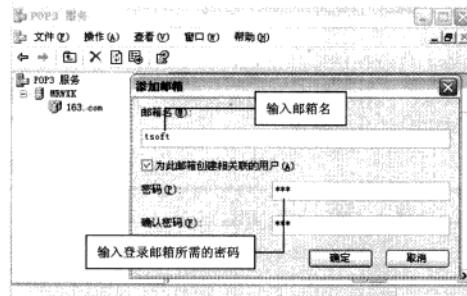


图 18.6 “添加邮箱”对话框

- ◆ 注意：① 所有域都在本机上分出一定的空间来存放信息，默认位置为“C:\Inetpub\mailroot\Mailbox”。  
 ② POP3 服务需要在 Windows Server 2003 系统以及更高版本中进行安装，Windows XP 或 Windows 7 系统中没有该服务。

### 18.1.3 端口及套接字介绍

一般而言，一台计算机只有单一的连到网络的“物理连接”(physical connection)，所有的数据都通过此连接，对内、对外送达特定的计算机，这就是端口。网络程序设计中的端口(port)并非真实的物理存在，而是一个假想的连接装置。端口被规定为一个在 0~65535 之间的整数。HTTP 服务一般使用 80 端口，FTP 服务使用 21 端口。假如一台计算机提供了 HTTP、FTP 等多种服务，那么客户机就会通过不同的端口来确定连接到服务器的哪项服务上，如图 18.7 所示。

技巧：0~1023 之间的端口号是用于一些知名的网络服务和应用，用户的普通网络应用程序应使用 1024 以上的端口号，以避免端口号与另一个应用或系统服务所用。

网络程序中的套接字（Socket）用于将应用程序与端口连接起来。套接字是一个假想的连接装置，就像插插头的设备“插座”，用于连接电器与电线，如图 18.8 所示。C# 将套接字抽象化为类，程序设计者只需创建 Socket 对象，即可使用套接字。

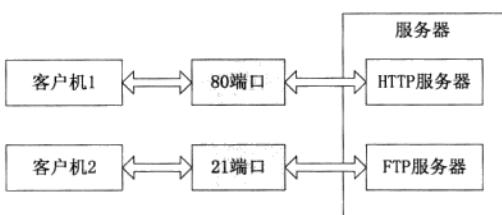


图 18.7 端口

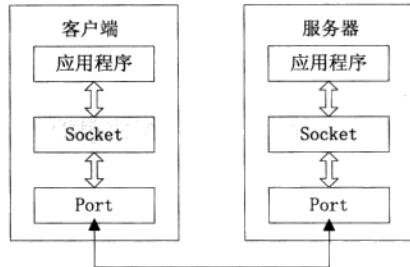


图 18.8 套接字

## 18.2 网络编程基础

使用 C# 进行网络编程时，通常都需要用到 System.NET 命名空间、System.NET.Sockets 命名空间和 System.NET.Mail 命名空间，下面对这 3 个命名空间及它们包含的主要类进行详细讲解。

### 18.2.1 System.NET 命名空间及相关类的使用

System.NET 命名空间为当前网络上使用的多种协议提供了简单的编程接口，而它所包含的 WebRequest 类和WebResponse 类形成了所谓的可插接式协议的基础。可插接式协议是网络服务的一种实现，它使用户能够开发出使用 Internet 资源的应用程序，而不必考虑各种不同协议的具体细节。下面对 System.NET 命名空间中的主要类进行详细讲解。

#### 1. Dns 类

Dns 类是一个静态类，它从 Internet 域名系统（DNS）检索关于特定主机的信息。在 IPHostEntry 类的实例中返回来自 DNS 查询的主机信息。如果指定的主机在 DNS 数据库中有多个入口，则 IPHostEntry 包含多个 IP 地址和别名。Dns 类的常用方法及说明如表 18.2 所示。

表 18.2 Dns 类的常用方法及说明

方 法	说 明
GetHostAddresses	返回指定主机的 Internet 协议（IP）地址
GetHostByAddress	获取 IP 地址的 DNS 主机信息
GetHostByName	获取指定 DNS 主机名的 DNS 信息
GetHostEntry	将主机名或 IP 地址解析为 IPHostEntry 实例
GetHostName	获取本地计算机的主机名

**例 18.01** 下面演示 Dns 类的使用方法，程序开发步骤如下。（实例位置：光盘\mr\18\sl\18.01）

- (1) 新建一个 Windows 应用程序，并将其命名为 UseDns，默认窗体为 Form1.cs。
- (2) 在 Form1 窗体中添加 4 个 TextBox 控件和一个 Button 控件，其中，TextBox 控件分别用来输入主机地址和显示主机 IP 地址、本地主机名、DNS 主机名；Button 控件用来调用 Dns 类中的各个方法获得主机 IP 地址、本地主机名和 DNS 主机名，并显示在相应的文本框中。

(3) 程序主要代码如下。

```
private void button1_Click(object sender, EventArgs e)
{
    if (textBox1.Text == string.Empty) //判断是否输入了主机地址
    {
        MessageBox.Show("请输入主机地址!");
    }
    else
    {
        textBox2.Text = string.Empty; //获取指定主机的 IP 地址
        IPAddress[] ips = Dns.GetHostAddresses(textBox1.Text); //循环访问获得的 IP 地址
        foreach(IPAddress ip in ips)
        {
            textBox2.Text += ip.ToString(); //将得到的 IP 地址显示在文本框中
        }
        textBox3.Text = Dns.GetHostName(); //获取本机名
        textBox4.Text = Dns.GetHostByName(Dns.GetHostName()).HostName; //根据指定的主机名获取 DNS 信息
    }
}
```

程序运行结果如图 18.9 所示。

## 2. IPAddress 类

IPAddress 类包含计算机在 IP 网络上的地址，它主要用来提供网际协议（IP）地址。IPAddress 类的常用字段、属性、方法及说明如表 18.3 所示。



图 18.9 Dns 类的使用

表 18.3 IPAddress 类的常用字段、属性、方法及说明

字段、属性及方法	说 明
Any 字段	提供一个 IP 地址，指示服务器应侦听所有网络接口上的客户端活动。此字段为只读
Broadcast 字段	提供 IP 广播地址。此字段为只读
Address 属性	网际协议（IP）地址
AddressFamily 属性	获取 IP 地址的地址族
IsIPv6LinkLocal 属性	获取地址是否为 IPv6 链接本地地址
IsIPv6SiteLocal 属性	获取地址是否为 IPv6 站点本地地址
GetAddressBytes 方法	以字节数组形式提供 IPAddress 的副本
Parse 方法	将 IP 地址字符串转换为 IPAddress 实例
TryParse 方法	确定字符串是否为有效的 IP 地址

**例 18.02** 下面演示 IPAddress 类的使用方法，程序开发步骤如下。（实例位置：光盘\mr\18\sl\18.02）

- (1) 新建一个 Windows 应用程序，并将其命名为 UseIPAddress，默认窗体为 Form1.cs。
- (2) 在 Form1 窗体中添加一个 TextBox 控件、一个 Button 控件和一个 Label 控件，其中，TextBox 控

件用来输入主机的网络地址或 IP 地址；Button 控件用来调用 IPAddress 类中的各个属性获取指定主机的 IP 地址信息；Label 控件用来显示获得的 IP 地址信息。

(3) 程序主要代码如下。

```
private void button1_Click(object sender, EventArgs e)
{
    label2.Text = string.Empty; // 初始化 Label 标签
    IPAddress[] ips = Dns.GetHostAddresses(textBox1.Text); // 获得指定主机的 IP 地址族
    foreach (IPAddress ip in ips) // 循环遍历得到的 IP 地址
    {
        // 在 Label 标签中显示得到的 IP 地址信息
        label2.Text += "网际协议地址：" + ip.Address + "\nIP 地址的地址族：" +
            ip.AddressFamily.ToString() + "\n 是否 IPv6 链接本地地址：" + ip.IsIPv6LinkLocal;
    }
}
```

程序运行结果如图 18.10 所示。

### 3. IPEndPoint 类

IPEndPoint 类包含应用程序连接到主机上的服务所需的主机和本地或远程端口信息。通过组合服务的主机 IP 地址和端口号，IPEndPoint 类形成到服务的连接点，它主要用来将网络端点表示为 IP 地址和端口号。IPEndPoint 类的常用字段、属性及说明如表 18.4 所示。

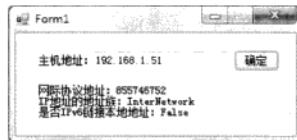


图 18.10 IPAddress 类的使用

表 18.4 IPEndPoint 类的常用字段、属性及说明

字段及属性	说 明
MaxPort 字段	指定可以分配给 Port 属性的最大值。MaxPort 值设置为 0x0000FFFF。此字段为只读
MinPort 字段	指定可以分配给 Port 属性的最小值。此字段为只读
Address 属性	获取或设置终结点的 IP 地址
AddressFamily 属性	获取网际协议 (IP) 地址族
Port 属性	获取或设置终结点的端口号

**例 18.03** 下面演示 IPEndPoint 类的使用方法，程序开发步骤如下。（实例位置：光盘\mr\18\sl\18.03）

(1) 新建一个 Windows 应用程序，并将其命名为 UseIPEndPoint，默认窗体为 Form1.cs。

(2) 在 Form1 窗体中添加一个 TextBox 控件、一个 Button 控件和一个 Label 控件，其中，TextBox 控件用来输入 IP 地址；Button 控件用来调用 IPEndPoint 类中的各个属性获取终结点的 IP 地址和端口号；Label 控件用来显示获得的 IP 地址和端口号。

(3) 程序主要代码如下。

```
private void button1_Click(object sender, EventArgs e)
{
    // 创建 IPEndPoint 对象
    IPEndPoint IPEPoint = new IPEndPoint(IPAddress.Parse(textBox1.Text), 80);
    // 使用 IPEndPoint 对象获取终结点的 IP 地址和端口号
    label2.Text = "IP 地址：" + IPEPoint.Address.ToString() + "\n 端口号：" + IPEPoint.Port;
}
```

程序运行结果如图 18.11 所示。

#### 4. WebClient 类

WebClient 类提供向 URI 标识的任何本地、Intranet 或 Internet 资源发送数据以及从这些资源接收数据的公共方法。WebClient 类的常用属性、方法及说明如表 18.5 所示。

表 18.5 WebClient 类的常用属性、方法及说明

属性及方法	说 明	属性及方法	说 明
QueryString 属性	获取或设置与请求关联的查询名称/值对集合	OpenRead 方法	为从具有指定 URI 的资源下载的数据打开一个可读的流
ResponseHeaders 属性	获取与响应关联的标头名称/值对集合	OpenWrite 方法	打开一个流以将数据写入具有指定 URI 的资源
DownloadFile 方法	将具有指定 URI 的资源下载到本地文件	UploadData 方法	将数据缓冲区上载到具有指定 URI 的资源
DownloadString 方法	以 String 或 URI 形式下载指定的资源	UploadFile 方法	将本地文件上载到具有指定 URI 的资源

**例 18.04** 下面演示 WebClient 类的使用方法，程序开发步骤如下。（实例位置：光盘\mr\18\sl\18.04）

(1) 新建一个 Windows 应用程序，并将其命名为 UseWebClient，默认窗体为 Form1.cs。

(2) 在 Form1 窗体中添加一个 TextBox 控件、一个 Button 控件和一个 RichTextBox 控件，其中，TextBox 控件用来输入标准网络地址；Button 控件用来获取指定网址中的网页内容，并将内容保存到一个文本文件中；RichTextBox 控件用来显示从指定网址中获取的网页内容。

(3) 程序主要代码如下。

```
private void button1_Click(object sender, EventArgs e)
{
    richTextBox1.Text = string.Empty;
    WebClient wclient = new WebClient(); //创建 WebClient 对象
    wclient.BaseAddress = textBox1.Text; //设置 WebClient 的基 URI
    wclient.Encoding = Encoding.UTF8; //指定下载字符串的编码方式
    //为 WebClient 对象添加标头
    wclient.Headers.Add("Content-Type", "application/x-www-form-urlencoded");
    Stream stream = wclient.OpenRead(textBox1.Text); //使用 OpenRead 方法获取指定网站的数据，并保存到 Stream 流中
    StreamReader sreader = new StreamReader(stream); //使用 Stream 流声明一个流读取变量 sreader
    string str = string.Empty; //声明一个变量，用来保存一行从 WebClient 下载的数据
    while ((str = sreader.ReadLine()) != null) //循环读取从指定网站获得的数据
    {
        richTextBox1.Text += str + "\n";
    }
    //调用 WebClient 对象的 DownloadFile 方法将指定网站的内容保存到文件中
    wclient.DownloadFile(textBox1.Text, DateTime.Now.ToString("yyyy-MM-dd") + ".txt");
    MessageBox.Show("保存到文件成功");
}
```

程序运行结果如图 18.12 所示。

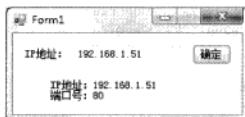


图 18.11 IPEndPoint 类的使用

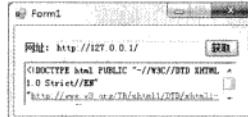


图 18.12 WebClient 类的使用

## 5. WebRequest 类和WebResponse 类

WebRequest 类是.NET Framework 的请求/响应模型的抽象基类，用于访问 Internet 数据。使用该请求/响应模型的应用程序可以用协议不可知的方式从 Internet 请求数据，在这种方式下，应用程序处理WebRequest 类的实例，而协议特定的子类则执行请求的具体细节。

WebResponse 类也是抽象基类，应用程序可以使用WebResponse 类的实例以协议不可知的方式参与请求和响应事务，而从WebResponse 类派生的协议类携带请求的详细信息。另外需要注意的是，客户端应用程序不直接创建WebResponse 对象，而是通过对WebRequest 实例调用 GetResponse 方法来进行创建。

WebRequest 类的常用属性、方法及说明如表 18.6 所示。

表 18.6 WebRequest 类的常用属性、方法及说明

属性及方法	说 明
ContentLength 属性	当在子类中被重写时，获取或设置所发送的请求数据的内容长度
Headers 属性	当在子类中被重写时，获取或设置与请求关联的标头名称/值对的集合
Method 属性	当在子类中被重写时，获取或设置要在此请求中使用的协议方法
Create 方法	初始化新的WebRequest
EndGetResponse 方法	当在子类中重写时，返回WebResponse
GetRequestStream 方法	当在子类中重写时，返回用于将数据写入 Internet 资源的 Stream
GetResponse 方法	当在子类中被重写时，返回对 Internet 请求的响应

WebResponse 类中的常用属性、方法及说明如表 18.7 所示。

表 18.7 WebResponse 类的常用属性、方法及说明

属性及方法	说 明
ContentLength 属性	当在子类中重写时，获取或设置接收的数据的内容长度
ContentType 属性	当在派生类中重写时，获取或设置接收的数据的内容类型
Headers 属性	当在派生类中重写时，获取与此请求关联的标头名称/值对的集合
ResponseUri 属性	当在派生类中重写时，获取实际响应此请求的 Internet 资源的 URI
Close 方法	当由子类重写时，关闭响应流
GetResponseStream 方法	当在子类中重写时，从 Internet 资源返回数据流

**例 18.05** 下面演示WebRequest 类和WebResponse 类的使用方法，程序开发步骤如下。（实例位置：光盘\mr\18\sl\18.05）

(1) 新建一个 Windows 应用程序，并将其命名为 UseWebResponseAndQuest，默认窗体为 Form1.cs。

(2) 在 Form1 窗体中添加一个 TextBox 控件、一个 Button 控件和一个 RichTextBox 控件，其中，TextBox 控件用来输入标准网络地址；Button 控件用来调用WebRequest 和WebResponse 类中的属性、方法获取指定网站的网页请求信息和网页内容；RichTextBox 控件用来显示根据指定网址获取的网页请求信息及网页内容。

(3) 程序主要代码如下。

```
private void button1_Click(object sender, EventArgs e)
{
    richTextBox1.Text = string.Empty;
    // 创建一个WebRequest 对象
    WebRequest webrequest = WebRequest.Create(textBox1.Text);
    // 设置用于对 Internet 资源请求进行身份验证的网络凭据
    webrequest.Credentials = CredentialCache.DefaultCredentials;
```

```

//调用 WebRequest 对象的各种属性获取 WebRequest 请求的相关信息
richTextBox1.Text = "请求数据的内容长度: " + webrequest.ContentLength;
richTextBox1.Text += "\n 该请求的协议方法: " + webrequest.Method;
richTextBox1.Text += "\n 访问 Internet 的网络代理: " + webrequest.Proxy;
richTextBox1.Text += "\n 与该请求关联的 Internet URI: " + webrequest.RequestUri;
richTextBox1.Text += "\n 超时时间: " + webrequest.Timeout;
//调用 WebRequest 对象的 GetResponse 方法创建一个 WebResponse 对象
WebResponse webresponse = webrequest.GetResponse();
//获取 WebResponse 响应的 Internet 资源的 URI
richTextBox1.Text += "\n 响应该请求的 Internet URI: " + webresponse.ResponseUri;
//调用 WebResponse 对象的 GetResponseStream 方法返回数据流
Stream stream = webresponse.GetResponseStream();
//使用创建的 Stream 对象创建一个 StreamReader 流读取对象
StreamReader sreader = new StreamReader(stream);
//读取流中的内容，并显示在 RichTextBox 控件中
richTextBox1.Text += "\n" + sreader.ReadToEnd();
sreader.Close();
stream.Close();
webresponse.Close();
}

```

程序运行结果如图 18.13 所示。

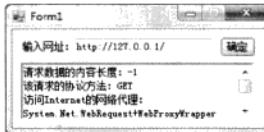


图 18.13 WebRequest 类和 WebResponse 类的使用

### 18.2.2 System.NET.Sockets 命名空间及相关类的使用

System.NET.Sockets 命名空间主要提供制作 Sockets 网络应用程序的相关类，其中 Socket 类、TcpClient 类、TcpListener 类和 UdpClinet 类较为常用，下面对它们进行详细介绍。

#### 1. Socket 类

Socket 类为网络通信提供了一套丰富的方法和属性，它主要用于管理连接，实现 Berkeley 通信端套接字接口，同时它还定义了绑定、连接网络端点及传输数据所需的各种方法，提供处理端点连接传输等细节所需要的功能。WebRequest、TcpClient 和 UdpClinet 等类在内部使用该类。Socket 类的常用属性及说明如表 18.8 所示。

表 18.8 Socket 类的常用属性及说明

属性	说 明
AddressFamily	获取 Socket 的地址族
Available	获取已经从网络接收且可供读取的数据量
Connected	获取一个值，该值指示 Socket 是在上次 Send 还是 Receive 操作时连接到远程主机
Handle	获取 Socket 的操作系统句柄
LocalEndPoint	获取本地终结点
RemoteEndPoint	获取远程终结点

Socket 类的常用方法及说明如表 18.9 所示。

表 18.9 Socket 类的常用方法及说明

方 法	说 明	方 法	说 明
Accept	为新建连接创建新的 Socket	Listen	将 Socket 置于侦听状态
Close	关闭 Socket 连接并释放所有关联的资源	Receive	接收来自绑定的 Socket 的数据
Connect	建立与远程主机的连接	Send	将数据发送到连接的 Socket
Disconnect	关闭套接字连接并允许重用套接字	SendTo	将数据发送到特定终结点

**例 18.06** 下面演示 Socket 类的使用方法，程序开发步骤如下。（实例位置：光盘\mr\18\sl\18.06）

(1) 新建一个 Windows 应用程序，并将其命名为 UseSocket，默认窗体为 Form1.cs。

(2) 在 Form1 窗体中添加两个 TextBox 控件和一个 Button 控件，其中，TextBox 控件分别用来输入要连接的主机及端口号；Button 控件用来连接远程主机，并获得其上的主页面内容。

(3) 程序主要代码如下。

```
private static Socket ConnectSocket(string server, int port)
{
    Socket socket = null; //创建 Socket 对象，并初始化为空
    IPHostEntry iphostentry = null; //创建 IPHostEntry 对象，并初始化为空
    iphostentry = Dns.GetHostEntry(server); //获得主机信息
    foreach (IPAddress address in iphostentry.AddressList) //循环遍历得到的 IP 地址列表
    {
        IPEndPoint IPEPoint = new IPEndPoint(address, port); //使用指定的 IP 地址和端口号创建 IPEndPoint
        //使用 Socket 的构造函数创建一个 Socket 对象，以便用来连接远程主机
        Socket newSocket = new Socket(IPEPoint.AddressFamily, SocketType.Stream, ProtocolType.Tcp);
        newSocket.Connect(IPEPoint); //调用 Connect 方法连接远程主机
        if (newSocket.Connected) //判断远程连接是否连接
        {
            socket = newSocket;
            break;
        }
        else
        {
            continue;
        }
    }
    return socket;
}

private static string SocketSendReceive(string server, int port) //获取指定服务器的主页面内容
{
    string request = "GET/HTTP/1.1\n 主机：" + server + "\n 连接：关闭\n";
    Byte[] btSend = Encoding.ASCII.GetBytes(request);
    Byte[] btReceived = new Byte[256];
    //调用自定义方法 ConnectSocket，使用指定的服务器名和端口号创建一个 Socket 对象
    Socket socket = ConnectSocket(server, port);
    if (socket == null)
        return ("连接失败！");
    socket.Send(btSend, btSend.Length, 0); //将请求发送到连接的服务器
    int intContent = 0;
    string strContent = server + "上的默认页面内容：" + "\n";
    while (intContent < 256)
    {
        intContent = socket.Receive(btReceived);
        strContent += Encoding.ASCII.GetString(btReceived, 0, intContent);
    }
    return strContent;
}
```

```

do
{
    intContent = socket.Receive(btReceived, btReceived.Length, 0); //从绑定的 Socket 接收数据
    strContent += Encoding.ASCII.GetString(btReceived, 0, intContent); //将接收到的数据转换为字符串类型
}
while (intContent > 0);
return strContent;
}

private void button1_Click(object sender, EventArgs e)
{
    string server = textBox1.Text; //指定主机名
    int port = Convert.ToInt32(textBox2.Text); //指定端口号
    //调用自定义方法 SocketSendReceive 获取指定主机的主页面内容
    string strContent = SocketSendReceive(server, port);
    MessageBox.Show(strContent);
}

```

程序运行结果如图 18.14 和图 18.15 所示。

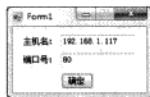


图 18.14 Socket 类的使用

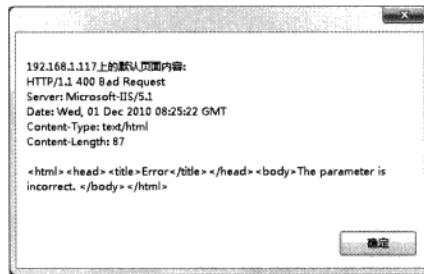


图 18.15 详细信息

## 2. TcpClient 类和 TcpListener 类

TcpClient 类用于在同步阻止模式下通过网络来连接、发送和接收流数据。为使 TcpClient 连接并交换数据，使用 TCP ProtocolType 类创建的 TcpListener 实例或 Socket 实例必须侦听是否有传入的连接请求。可以使用下面两种方法之一连接到该侦听器。

- 创建一个 TcpClient，并调用 3 个可用的 Connect 方法之一。
- 使用远程主机的主机名和端口号创建 TcpClient，此构造函数将自动尝试一个连接。

TcpListener 类用于在阻止同步模式下侦听和接受传入的连接请求。可使用 TcpClient 类或 Socket 类来连接 TcpListener，并且可以使用 IPEndPoint、本地 IP 地址及端口号或者仅使用端口号来创建 TcpListener 实例对象。

TcpClient 类的常用属性、方法及说明如表 18.10 所示。

表 18.10 TcpClient 类的常用属性、方法及说明

属性及方法	说 明
Available 属性	获取已经从网络接收且可供读取的数据量
Connected 属性	获取一个值，该值指示 TcpClient 的基础 Socket 是否已连接到远程主机
ReceiveBufferSize 属性	获取或设置接收缓冲区的大小
SendBufferSize 属性	获取或设置发送缓冲区的大小
Close 方法	释放此 TcpClient 实例，而不关闭基础连接

续表

属性及方法	说 明
Connect 方法	使用指定的主机名和端口号将客户端连接到 TCP 主机
GetStream 方法	返回用于发送和接收数据的 NetworkStream

TcpListener 类的常用属性、方法及说明如表 18.11 所示。

表 18.11 TcpListener 类的常用属性、方法及说明

属性及方法	说 明
LocalEndpoint 属性	获取当前 TcpListener 的基础 EndPoint
Server 属性	获取基础网络 Socket
AcceptSocket/AcceptTcpClient 方法	接受挂起的连接请求
Start 方法	开始侦听传入的连接请求
Stop 方法	关闭侦听器

**例 18.07** 下面演示 TcpClient 类和 TcpListener 类的使用方法，程序开发步骤如下。（实例位置：光盘\mr\18\sl\18.07）

(1) 新建一个 Windows 应用程序，并将其命名为 UseTCP， 默认窗体为 Form1.cs。

(2) 在 Form1 窗体中添加两个 TextBox 控件、一个 Button 控件和一个 RichTextBox 控件，其中，TextBox 控件分别用来输入要连接的主机及端口号；Button 控件用来执行连接远程主机操作；RichTextBox 控件用来显示远程主机的连接状态。

(3) 程序主要代码如下。

```
private void button1_Click(object sender, EventArgs e)
{
    TcpListener tcplistener = null; //创建一个 TcpListener 对象，并初始化为空
    IPAddress ipaddress = IPAddress.Parse(textBox1.Text); //创建一个 IPAddress 对象，用来表示网络 IP 地址
    int port = Convert.ToInt32(textBox2.Text); //定义一个 int 类型变量，用来存储端口号
    tcplistener = new TcpListener(ipaddress, port); //初始化 TcpListener 对象
    tcplistener.Start(); //开始 TcpListener 侦听
    richTextBox1.Text = "等待连接...\n";
    TcpClient tcpclient = null; //创建一个 TcpClient 对象，并赋值为空
    if (tcplistener.Pending()) //判断是否有挂起的连接请求
        tcpclient = tcplistener.AcceptTcpClient(); //使用 AcceptTcpClient 初始化 TcpClient 对象
    else
        tcpclient = new TcpClient(textBox1.Text, port); //使用 TcpClient 的构造函数初始化 TcpClient 对象
    richTextBox1.Text += "连接成功！\n";
    tcpclient.Close(); //关闭 TcpClient 连接
    tcplistener.Stop(); //停止 TcpListener 侦听
}
```

程序运行结果如图 18.16 所示。

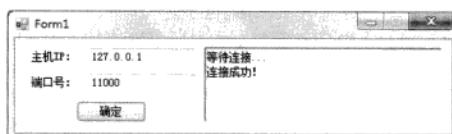


图 18.16 TcpClient 类的使用

### 3. UdpClient 类

UdpClient 类用于在阻止同步模式下发送和接收无连接 UDP 数据报。因为 UDP 是无连接传输协议，所以不需要在发送和接收数据前建立远程主机连接，但可以选择使用下面两种方法之一来建立默认远程主机。

使用远程主机名和端口号作为参数创建 UdpClient 类的实例。

创建 UdpClient 类的实例，然后调用 Connect 方法。

UdpClient 类的常用属性、方法及说明如表 18.12 所示。

表 18.12 UdpClient 类的常用属性、方法及说明

属性及方法	说 明
Available 属性	获取从网络接收的可读取的数据量
Client 属性	获取或设置基础网络 Socket
Close 方法	关闭 UDP 连接
Connect 方法	建立默认远程主机
Receive 方法	返回已由远程主机发送的 UDP 数据报
Send 方法	将 UDP 数据报发送到远程主机

**例 18.08** 下面演示如何使用 UdpClient 类中的属性及方法，程序开发步骤如下。（实例位置：光盘\mr\18\sl\18.08）

(1) 新建一个 Windows 应用程序，并将其命名为 UseUDP，默认窗体为 Form1.cs。

(2) 在 Form1 窗体中添加 3 个 TextBox 控件、一个 Button 控件和一个 RichTextBox 控件，其中，TextBox 控件分别用来输入远程主机名、端口号及要发送的信息；Button 控件用来向指定的主机发送信息；RichTextBox 控件用来显示接收到的信息。

(3) 程序主要代码如下。

```
private void button1_Click(object sender, EventArgs e)
{
    richTextBox1.Text = string.Empty; //清空 RichTextBox
    UdpClient udpclient = new UdpClient(Convert.ToInt32(textBox2.Text)); //创建 UdpClient 对象
    //调用 UdpClient 对象的 Connect 建立默认远程主机
    udpclient.Connect(textBox1.Text, Convert.ToInt32(textBox2.Text));
    //定义一个字节数组，用来存放发送到远程主机的信息
    Byte[] sendBytes = Encoding.Default.GetBytes(textBox3.Text);
    //调用 UdpClient 对象的 Send 方法将 Udp 数据报发送到远程主机
    udpclient.Send(sendBytes, sendBytes.Length);
    //创建 IPEndPoint 对象，用来显示响应主机的标识
    IPEndPoint ipendpoint = new IPEndPoint(IPAddress.Any, 0);
    //调用 UdpClient 对象的 Receive 方法获得从远程主机返回的 Udp 数据报
    Byte[] receiveBytes = udpclient.Receive(ref ipendpoint);
    //将获得的 Udp 数据报转换为字符串形式
    string returnData = Encoding.Default.GetString(receiveBytes);
    richTextBox1.Text = "接收到的信息：" + returnData.ToString();
    //使用 IPEndPoint 对象的 Address 和 Port 属性获得响应主机的 IP 地址和端口号
    richTextBox1.Text += "\n这条信息来自主机" + ipendpoint.Address.ToString()
        + "上的" + ipendpoint.Port.ToString() + "端口";
    udpclient.Close(); //关闭 UdpClient 连接
}
```

程序运行结果如图 18.17 所示。

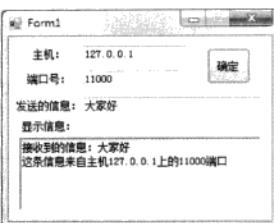


图 18.17 UdpClient 类的使用

### 18.2.3 System.NET.Mail 命名空间及相关类的使用

System.NET.Mail 命名空间包含用于将电子邮件发送到简单邮件传输协议（SMTP）服务器进行传送的类，其中 MailMessage 类用来表示邮件的内容，Attachment 类用来创建邮件附件，SmtpClient 类用来将电子邮件传输到指定用于邮件传送的 SMTP 主机上。下面对这 3 个类进行详细讲解。

#### 1. MailMessage 类

MailMessage 类表示可以使用 SmtpClient 类发送的电子邮件，它主要用于指定邮件的发送地址、收件人地址、邮件正文及附件等。MailMessage 类的常用属性及说明如表 18.13 所示。

表 18.13 MailMessage 类的常用属性及说明

属性	说明
Attachments	获取用于存储附加到此电子邮件的数据的附件集合
Body	获取或设置邮件正文
BodyEncoding	获取或设置用于邮件正文的编码
From	获取或设置此电子邮件的发信人地址
Headers	获取与此电子邮件一起传输的电子邮件标头
Sender	获取或设置此电子邮件的发件人地址
Subject	获取或设置此电子邮件的主题行
To	获取包含此电子邮件的收件人的地址集合

**例 18.09** 下面创建一个 MailMessage 邮件发送对象，并通过其属性设置邮件的发送人、接收人、主题和内容。代码如下。

```
MailAddress from = new MailAddress("tsoft@163.com"); //设置邮件发送人
MailAddress to = new MailAddress("tsoft@163.com"); //设置邮件接收人
MailMessage message = new MailMessage(from,to); //创建一个 MailMessage 对象
message.Subject = "邮件测试"; //设置发送邮件的主题
message.Body = "邮件正文"; //设置发送邮件的内容
```

#### 2. Attachment 类

Attachment 类表示电子邮件的附件，它需要与 MailMessage 类一起使用。创建完电子邮件的附件之后，若要将附件添加到邮件中，需要将附件添加到 MailMessage.Attachments 集合中。Attachment 类的常用属性、方法及说明如表 18.14 所示。

表 18.14 Attachment 类的常用属性、方法及说明

属性及方法	说 明
ContentDisposition 属性	获取附件的 MIME 内容标头信息
ContentId 属性	获取或设置附件的 MIME 内容 ID
ContentStream 属性	获取附件的内容流
ContentType 属性	获取附件的内容类型
Name 属性	获取或设置与附件关联的内容类型中的 MIME 内容类型名称值
NameEncoding 属性	指定用于 AttachmentName 的编码
TransferEncoding 属性	获取或设置附件的编码
CreateAttachmentFromString 方法	用字符串创建附件

**例 18.10** 下面创建一个 MailMessage 邮件发送对象，同时通过 Attachment 类创建一个附件，并设置该附件的时间信息，然后调用 MailMessage 邮件发送对象的 Attachments 属性的 Add 方法将创建的附件添加到要发送的邮件中。代码如下。

```

string file = "C:\邮件测试.txt"; //设置要发送的附件
MailAddress from = new MailAddress("soft@163.com"); //设置邮件发送人
MailAddress to = new MailAddress("soft@163.com"); //设置邮件接收人
MailMessage message = new MailMessage(from,to); //创建 MailMessage 类的对象
Attachment myAttachment = new Attachment(file, System.Net.Mime.MediaTypeNames.Application.Octet); //为要发送的邮件创建附件
System.Net.Mime.ContentDisposition disposition = myAttachment.ContentDisposition; //为附件添加时间信息
disposition.CreationDate = System.IO.File.GetCreationTime(file);
disposition.ModificationDate = System.IO.File.GetLastWriteTime(file);
disposition.ReadDate = System.IO.File.GetLastAccessTime(file);
message.Attachments.Add(myAttachment); //将创建的附件添加到邮件中

```

### 3. SmtpClient 类

SmtpClient 类用于将电子邮件发送到 SMTP 服务器以便传递。使用 SmtpClient 类实现发送电子邮件功能时必须指定以下信息。

- 用来发送电子邮件的 SMTP 主机服务器。
- 身份验证凭据（如果 SMTP 服务器要求）。
- 发件人的电子邮件地址。
- 收件人的电子邮件地址。
- 邮件内容。

SmtpClient 类的常用属性、方法及说明如表 18.15 所示。

表 18.15 SmtpClient 类的常用属性、方法及说明

属性及方法	说 明
Credentials 属性	获取或设置用于验证发件人身份的凭据
Host 属性	获取或设置用于 SMTP 事务的主机的名称或 IP 地址
Port 属性	获取或设置用于 SMTP 事务的端口
ServicePoint 属性	获取用于传输电子邮件的网络连接
Send 方法	将电子邮件发送到 SMTP 服务器以便传递，该方法在传输邮件的过程中将阻止其他操作
SendAsync 方法	发送电子邮件，该方法不会阻止调用线程

**例 18.11** 下面首先创建一个 MailMessage 邮件发送类的对象，然后通过 Attachment 类创建一个附件，并设置该附件的时间信息，接着通过设置 SmtpClient 类的 Credentials 属性来验证发件人身份，最后调用 SmtpClient 类的 Send 方法发送电子邮件。代码如下。

```

string file = "C:\邮件测试.txt";           //设置发送的附件
MailAddress from = new MailAddress("soft@163.com"); //设置邮件发送人
MailAddress to = new MailAddress("soft@163.com"); //设置邮件接收人
MailMessage message = new MailMessage(from,to); //创建 MailMessage 对象
message.Subject = "邮件测试";                //设置发送邮件的主题
message.Body = "邮件正文";                   //设置发送邮件的内容
Attachment myAttachment = new Attachment(file, System.Net.Mime.MediaTypeNames.Application.Octet); //创建附件信息
System.Net.Mime.ContentDisposition disposition = myAttachment.ContentDisposition; //为附件添加时间信息
disposition.CreationDate = System.IO.File.GetCreationTime(file);           //创建日期
disposition.ModificationDate = System.IO.File.GetLastWriteTime(file);        //更新日期
disposition.ReadDate = System.IO.File.GetLastAccessTime(file);              //读取日期
message.Attachments.Add(myAttachment); //将附件添加到邮件中
SmtpClient client = new SmtpClient("192.168.1.97", 25); //创建 SmtpClient 对象
client.Credentials = new System.Net.NetworkCredential("soft", "111"); //验证发件人身份
client.Send(message); //发送邮件

```

## 18.3 照猫画虎——基本功训练

### 18.3.1 基本功训练 1——通过 IP 地址获取主机名称

■ 视频讲解：光盘\mr\18\lx\通过 IP 地址获取主机名称.exe

■ 实例位置：光盘\mr\18\zmhh\01

创建一个 Windows 应用程序，在 Form1 窗体中主要添加一个 Button 控件和两个 TextBox 控件，分别用来执行获取主机名的方法和显示 IP 与机器名，主要代码如下。

```

private void button1_Click(object sender, EventArgs e)
{
    IPHostEntry hostInfo; //创建一个 IPHostEntry 实例, 存储主机信息
    try
    {
        hostInfo= Dns.Resolve(this.textBox1.Text); //根据输入的 IP 地址创建 IPHostEntry
    }
    catch (Exception ey) //如果有异常
    {
        MessageBox.Show(ey.Message); //显示异常信息
        this.textBox1.Focus(); //使输入 IP 地址的文本框获得焦点
        this.textBox1.SelectAll(); //选中其中的内容
        return;
    }
    this.textBox2.Text=hostInfo.HostName; //获取主机名称
}

```

程序运行结果如图 18.18 所示。



图 18.18 通过 IP 地址获得主机名称

**照猫画虎：**通过主机名称获取 IP 地址。提示：可以通过使用 Dns 类的 GetHostAddresses 方法来获取。  
(20 分)(实例位置：光盘\mr\18\zmhh\01\_zmhh)

### 18.3.2 基本功训练 2——得到本机 MAC 地址

**视频讲解：**光盘\mr\18\lx\得到本机 MAC 地址.exe

**实例位置：**光盘\mr\18\zmhh\02

MAC 地址是网卡的物理地址，它能够标识网络中一台唯一的计算机。创建一个 Windows 应用程序，在 Form1 窗体中添加两个 Label 控件，用来显示信息，如 MAC 地址，主要程序如下。

```
private void Form1_Load(object sender, EventArgs e)
{
    //创建一个 ManagementObjectSearcher 类的对象
    ManagementObjectSearcher nisc = new ManagementObjectSearcher("select * from Win32_NetworkAdapterConfiguration");
    foreach(ManagementObject nic in nisc.Get()) //遍历返回的结果集合
    {
        if(Convert.ToBoolean(nic["ipEnabled"]) == true) //如果 nic["ipEnabled"]值为 true
        {
            this.label2.Text=Convert.ToString(nic["MACAddress"]); //获取 MAC 地址
        }
    }
}
```

程序运行结果如图 18.19 所示。



图 18.19 得到本机 MAC 地址

**说明：**ManagementObjectSearcher 可用于枚举系统中的所有磁盘驱动器、网络适配器、进程及更多管理对象，或者用于查询所有处于活动状态的网络连接及暂停的服务等。

**照猫画虎：**获取显卡的序列号。提示：通过加载“SELECT \* FROM Win32\_VideoController”字符串创建 ManagementObjectSearcher 类的对象。(20 分)(实例位置：光盘\mr\18\zmhh\02\_zmhh)

### 18.3.3 基本功训练 3——获取网络中所有工作组名称

**视频讲解：**光盘\mr\18\lx\获取网络中所有工作组名称.exe

**实例位置：**光盘\mr\18\zmhh\03

创建一个 Windows 应用程序，在 Form1 窗体中添加一个 Button 控件和一个 listBox1 控件，分别用来退

出应用程序和显示工作组名称，主要代码如下。

```

private void Form1_Load(object sender, EventArgs e)
{
    DirectoryEntry MainGroup = new DirectoryEntry("WinNT:");
    foreach (DirectoryEntry domain in MainGroup.Children)
    {
        listBox1.Text = "";
        listBox1.Items.Add(domain.Name);
    }
}

```

程序运行结果如图 18.20 所示。



图 18.20 获取网络中所有工作组名称

**照猫画虎：**列出指定工作组中的所有计算机名。提示：可使用 `DirectoryEntry` 类的 `Name` 属性获取计算机名称。(20 分)(实例位置：光盘\mr\18\zmhh\03\_zmhh)

#### 18.3.4 基本功训练 4——获取网络中某台计算机的磁盘信息

■ 视频讲解：光盘\mr\18\lx\获取网络中某台计算机的磁盘信息.exe

■ 实例位置：光盘\mr\18\zmhh\04

创建一个 Windows 应用程序，在 `Form1` 窗体中添加一个 `FolderBrowserDialog` 控件，用来选择“磁盘”；添加 3 个 `TextBox` 控件，分别用来显示磁盘总容量、显示磁盘剩余空间和显示盘符。主要代码如下。

```

[DllImport("kernel32.dll", EntryPoint = "GetDiskFreeSpaceEx")]           //引入 kernel32.dll 程序集
public static extern int GetDiskFreeSpaceEx(string lpDirectoryName, out long lpFreeBytesAvailable, out long
lpTotalNumberOfBytes,
out long lpTotalNumberOfFreeBytes);                                     //引入指定磁盘的空间的 API 函数
private void button1_Click(object sender, EventArgs e)                      //获取磁盘信息
{
    if (this.folderBrowserDialog1.ShowDialog() == DialogResult.OK)          //判断是否选择磁盘或目录
    {
        long fb, ftb, tfb;
        string str = this.folderBrowserDialog1.SelectedPath;               //定义 long 型变量，存储磁盘信息
        this.textBox3.Text = str;                                            //获取选择的磁盘或目录的路径
        if (GetDiskFreeSpaceEx(str, out fb, out ftb, out tfb) != 0)         //显示磁盘符
        {
            string strfb = Convert.ToString(ftb / 1024 / 1024)+"M";      //如果返回值不等于 0，说明读取成功
            string strftb = Convert.ToString(tfb / 1024 / 1024)+"M";
            this.textBox1.Text = strfb;                                       //磁盘总容量
            this.textBox2.Text = strftb;                                      //磁盘剩余空间
        }
    }
}

```

```
        else
        {
            MessageBox.Show("NO");
        }
    }
```

程序运行结果如图 18.21 和图 18.22 所示。

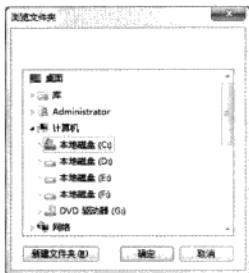


图 18.21 “浏览文件夹”对话框

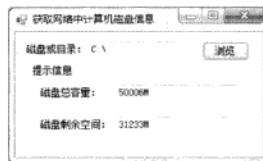


图 18.22 获取网络中某台计算机的磁盘信息

 说明：API 函数 GetDiskFreeSpaceEx 能够获取指定磁盘的空间。

**照猫画虎：**获取本地计算机所有磁盘的信息，包括本机磁盘总容量和本机磁盘总剩余空间。提示：可用 `DriveInfo` 类的 `GetDrives` 方法获取本机所有驱动器名称。(20 分) (实例位置：光盘\mr\18\zmhh\04\_zmhh)

### 18.3.5 基本功训练 5——编程实现 Ping 操作

 视频讲解：光盘\mr\18\lx\编程实现 Ping 操作.exe

 实例位置：光盘\mr\18\zmhh\05

创建一个 Windows 应用程序，在窗体上添加一个 Button 控件，用来实现 Ping 操作；添加 5 个 TextBox 控件，分别用来输入要 Ping 的 IP 地址、显示耗费时间、显示路由节点数、显示数据分段和显示缓冲区大小。代码如下。

```
private void button1_Click(object sender, EventArgs e)
{
    try
    {
        Ping PingInfo = new Ping(); //创建 Ping 类
        PingOptions PingOpt = new PingOptions(); //创建 PingOptions
        PingOpt.DontFragment = true; //是否设置数据分段
        string myInfo = "hyworkhyworkhyworkhyworkhywork"; //定义测试数据
        byte[] bufferInfo = Encoding.ASCII.GetBytes(myInfo); //获取测试数据的字节
        int TimeOut = 120; //超时时间
        PingReply reply = PingInfo.Send(this.textBox1.Text, TimeOut, bufferInfo, PingOpt); //发送数据包
        if (reply.Status == IPStatus.Success) //如果成功
        {
            this.textBox2.Text = reply.RoundtripTime.ToString(); //耗费时间
            this.textBox3.Text = reply.Options.Ttl.ToString(); //路由节点数
            this.textBox4.Text = (reply.Options.DontFragment ? "发生分段" : "没有发生分段"); //数据分段
            this.textBox5.Text = reply.Buffer.Length.ToString(); //缓冲区大小
        }
    }
}
```

```

        }
    else
    {
        MessageBox.Show("无法 Ping 通");
    }
}
catch (Exception ey)
{
    MessageBox.Show(ey.Message);
}
}

```

//如果当前状态不成功  
//弹出提示信息  
//如果发生异常  
//显示异常信息

程序运行结果如图 18.23 所示。

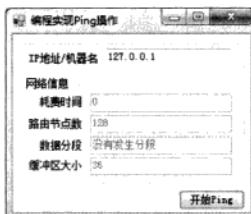


图 18.23 编程实现 Ping 操作

**说明：**使用 Ping 类的 Send 方法检测远程计算机。

**照猫画虎：**使用 Process 类实现 Ping 操作。提示：使用 Process 类的 StartInfo 属性来设置启动 Ping 命令进程的参数。（20 分）（实例位置：光盘\mr\18\zmhh\05\_zmhh）

照猫画虎栏目分数统计：

照猫画虎题目	1	2	3	4	5	总分数	
分数							

## 18.4 情景应用——拓展与实践

### 18.4.1 情景应用 1——获取网络信息及流量

**视频讲解：**光盘\mr\18\lx\获取网络信息及流量.exe

**实例位置：**光盘\mr\18\qjyy\01

本实例开发一个网络信息流量实时监控程序，运行本实例，可以在桌面右下角看到当前日期、时间及本地的网络信息流量。实例运行效果如图 18.24 所示。

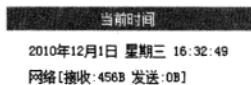


图 18.24 获取网络信息及流量

实现过程如下。

(1) 新建一个 Windows 应用程序，在 Form1 窗体中主要添加两个 Label 控件，分别用来显示当前时间和网络流量；添加一个 Timer 控件，用来时刻更新当前时间和网络流量。

(2) 主要程序代码。

Form1 窗体加载时，设置窗体在桌面右下角显示，同时显示网络的初始信息流量。实现代码如下：

```
private void Form1_Load(object sender, EventArgs e)
{
    this.TopMost = true;
    int Swidth = Screen.PrimaryScreen.WorkingArea.Width; //获取屏幕宽度
    int SHeight = Screen.PrimaryScreen.WorkingArea.Height; //获取屏幕高度
    this.DesktopLocation = new Point(Swidth - this.Width, SHeight - this.Height); //设置窗体加载时位置
    myNetInfo = new NetInfo();
    myNetStruct = myNetInfo.myNetStructs;
    myNetInfo.GetInfo(myNetStruct[0]);
}
```

上面的代码中用到了 GetInfo 方法，该方法用来初始化网络信息流量，并且启动计时器。GetInfo 方法实现代码如下：

```
public void GetInfo(NetStruct myNetStruct) //该方法用来初始化网络信息流量
{
    if (!listnets.Contains(myNetStruct))
    {
        listnets.Add(myNetStruct);
        myNetStruct.BeInfo(); //初始化信息流量
    }
    timer.Enabled = true; //启动计时器
}
```

上面代码中，BeInfo 方法用来初始化网络信息流量，timer 计时器对象用来实时更新网络信息流量，实现代码如下：

```
internal void BeInfo() //初始化流量
{
    receiveOldValue = receiveCounter.NextSample().RawValue; //接受信息
    sendOldValue = sendCounter.NextSample().RawValue; //发送信息
}
private Timer timer; //定义计时器
public NetInfo()
{
    timer = new Timer(1000); //创建一个计时器
    timer.Elapsed += new ElapsedEventHandler(timer_Elapsed); //定义 Elapsed 事件
}
private void timer_Elapsed(object sender, ElapsedEventArgs e) //Elapsed 事件的处理方法
{
    foreach (NetStruct myNetStruct in listnets)
        myNetStruct.ReInfo(); //刷新网络流量
}
```

timer 对象的 Elapsed 自定义事件中用到了 ReInfo 方法，该方法主要用来刷新网络信息流量，其实现代码如下：

```
internal void ReInfo() //刷新网络流量
{
```

```

receiveValue = receiveCounter.NextSample().RawValue; //接收的网络流量
sendValue = sendCounter.NextSample().RawValue; //发送的网络流量
receive = receiveValue - receiveOldValue; //新收到的网络流量
send = sendValue - sendOldValue; //新发送的网络流量
receiveOldValue = receiveValue; //记录前一次的接收流量
sendOldValue = sendValue; //记录前一次的发送流量
}

```

当 Form1 窗体中计时器启动时, 将当前日期时间和网络信息流量显示在相应的 Label 控件中, 实现代码如下:

```

private void timer1_Tick(object sender, EventArgs e)
{
    label2.Text = DateTime.Now.ToString("yyyy-MM-dd HH:mm:ss") + " " + getWeek() + " " + DateTime.Now.ToString("yyyy-MM-dd HH:mm:ss");
    //显示时间
    NetStruct NStruct = myNetStruct[0];
    label3.Text = "网络[接收: " + NStruct.Receive + "B 发送: " + NStruct.Send + "B]";
    //获取流量
    //显示流量
}

```

**DIY:** 监测当前网络连接状态。提示: 可考虑使用 API 函数 InternetGetConnectedState, 该函数用来判断当前计算机的网络状态。(20 分)(实例位置: 光盘\mr\18\qjyy\01\_diy)

#### 18.4.2 情景应用 2——远程关闭与重启计算机

视频讲解: 光盘\mr\18\lx\远程关闭与重启计算机.exe

实例位置: 光盘\mr\18\qjyy\02

本实例实现远程关闭或重启计算机的功能, 运行本实例, 确认远程计算机信息填写完整后, 单击“关闭”按钮, 关闭远程计算机; 单击“重启”按钮, 重启远程计算机。实例运行效果如图 18.25 所示。

实现过程如下。

(1) 创建一个 Windows 窗体应用程序, 在窗体中添加 4 个 TextBox 控件, 分别用来输入远程计算机的姓名、主机地址、登录用户和密码; 添加两个 Button 控件, 分别用来执行远程关闭和重启计算机操作。

(2) 程序主要代码如下。

该实例中主要用到了 CloseComputer 自定义方法, 该方法用来实现远程关闭或重启计算机功能, 它有一个 string 类型的参数, 用来表示要执行的操作命令(关闭或重启命令), 实现代码如下:

```

private void CloseComputer(string doinfo) //关闭或重启远程计算机
{
    ConnectionOptions op = new ConnectionOptions(); //创建 ConnectionOptions 对象
    op.Username = textBox4.Text; //设置远程机器用户名
    op.Password = textBox3.Text; //设置远程机器登录密码
    ManagementScope scope = new ManagementScope("\\\" + textBox2.Text + "\\root\\cimv2:Win32_Service", op);
    try
    {
        scope.Connect(); //连接远程对象
        ObjectQuery oq = new ObjectQuery("SELECT * FROM Win32_OperatingSystem"); //创建 ObjectQuery 对象
        ManagementObjectSearcher query1 = new ManagementObjectSearcher(scope, oq);
        ManagementObjectCollection queryCollection1 = query1.Get(); //得到 WMI 控制
    }
}

```



图 18.25 远程关闭与重启计算机

```

foreach (ManagementObject mobj in queryCollection1)           //遍历 WMI 控制集合
{
    string[] str = { "" };
    mobj.InvokeMethod(doinfo, str);                            //创建字符串数组
    }                                                       //获取控制信息
}
}
catch (Exception ey)
{
    MessageBox.Show(ey.Message);
}
}

```

**DIY:** 定时关闭计算机。提示：主要用到 API 函数 ExitWindowsEx，该函数用于退出、重启或注销系统。(20 分)(实例位置：光盘\mr\18\qjyy\02\_diy)

### 18.4.3 情景应用 3——创建 Web 页面浏览器

■ 视频讲解：光盘\mr\18\lx\创建 Web 页面浏览器.exe

■ 实例位置：光盘\mr\18\qjyy\03

Web 页面浏览器是一种可以显示网页服务器或文件系统的 HTML 文件内容，并让用户与这些文件交互的软件。本实例通过使用 C#制作一个简单的 Web 页面浏览器，程序运行效果如图 18.26 所示。

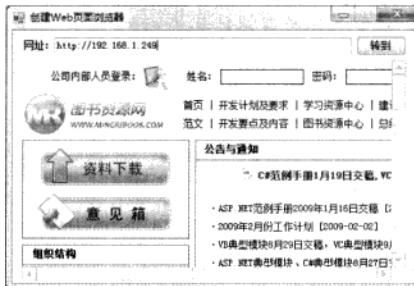


图 18.26 创建 Web 页面浏览器

实现步骤如下。

(1) 新建一个 Windows 应用程序，在窗体中添加一个 TextBox 控件、一个 Button 控件和一个 WebBrowser 控件，分别用来输入要浏览的网页地址、执行浏览网页操作和显示要浏览的网页。

(2) 程序主要代码如下。

```

private void button1_Click(object sender, EventArgs e)
{
    if (textBox1.Text != "")
    {
        Uri address = new Uri(textBox1.Text);          //创建一个 Uri 类型的变量，用来存储要浏览的网页地址
        webBrowser1.Url = address;                    //在 WebBrowser 控件中显示指定的网页
    }
}
private void textBox1_KeyPress(object sender, KeyPressEventArgs e) //在地址栏中按下 Enter 键
{
}

```

```

if (e.KeyChar == 13)
{
    if (textBox1.Text != "")
    {
        button1_Click(sender, e);
    }
}
}

```

**DIY：**实例中创建的浏览器只具有“转到”功能，这里要求创建一个具有“前进”、“后退”和“刷新”功能的完善浏览器。提示：可以调用 WebBrowser 控件的 GoForward、GoBack 和 Refresh 等方法来实现。

(20 分)(实例位置：光盘\mr\18\qjyy\03\_diy)

#### 18.4.4 情景应用 4——设计点对点聊天程序

■**视频讲解：**光盘\mr\18\lx\设计点对点聊天程序.exe

■**实例位置：**光盘\mr\18\qjyy\04

本实例使用 C# 开发了一个点对点聊天程序，该程序把本机作为服务器，可以直接将信息发送给对方。实例运行效果如图 18.27 所示。

实现过程如下。

(1) 新建一个 Windows 应用程序，在窗体上主要添加两个 RichTextBox 控件，分别用来输入聊天信息和显示聊天信息；添加两个 TextBox 控件，分别用来输入对方的主机和昵称；添加 3 个 Button 控件，分别用来清空聊天记录、发送信息和退出程序；添加一个 Timer 控件，用来时刻更新接收到的信息。

(2) 程序主要代码如下。

```

private void button2_Click(object sender, EventArgs e)
{
    try
    {
        IPAddress[] ip = Dns.GetHostAddresses(Dns.GetHostName()); //获取主机名
        string strmsg = "+" + txtName.Text + "(" + ip[0].ToString() + ")" + DateTime.Now.ToString("yyyy-MM-dd HH:mm:ss") + "+"
        +
        this.rtbSend.Text + "\n"; //定义消息格式
        TcpClient client = new TcpClient(txtIP.Text, 888); //创建 TcpClient 对象
        NetworkStream netstream = client.GetStream(); //创建 NetworkStream 网络流对象
        StreamWriter wstream = new StreamWriter(netstream, Encoding.Default); //创建数据写入对象
        wstream.WriteLine(strmsg); //将消息写入网络流
        wstream.Flush(); //释放网络流对象
        wstream.Close(); //关闭网络流对象
        client.Close(); //关闭 TcpClient
        rtbContent.AppendText(strmsg); //将发送的消息添加到文本框
        rtbContent.ScrollToCaret(); //自动滚动文本框的滚动条
        rtbSend.Clear(); //清空发送消息文本框
    }
    catch (Exception ex)

```

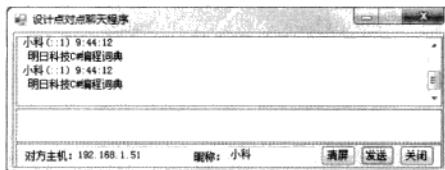


图 18.27 设计点对点聊天程序

```
{  
    MessageBox.Show(ex.Message);  
}  
}
```

**DIY:** 使用 UDP 协议设计聊天室。提示：在发送端使用 Socket 类的 SendTo 方法发送消息，在接收端使用 UdpClient 类的 Receive 方法接收消息。（20 分）（实例位置：光盘\mr\18\qjyy\04\diy）

#### 18.4.5 情景应用 5——电子邮件的发送与接收

 视频讲解：光盘\mr\18\lx\电子邮件的发送与接收.exe

 实例位置：光盘\mr\18\qjyy\05

本实例讲解如何使用 C# 实现电子邮件的发送与接收。运行本实例，在主窗体中单击“邮件发送”菜单项，打开“邮件发送”窗体，如图 18.28 所示；单击“邮件接收”菜单项，打开“邮件接收”窗体，如图 18.29 所示。



图 18.28 电子邮件的发送



图 18.29 电子邮件的接收

实现过程如下。

(1) 新建一个 Windows 应用程序，在 Form1 窗体中添加一个 `MenuStrip` 控件，用来作为菜单栏。

(2) 再新添加两个 Windows 窗体，分别命名为 frmSend.cs 和 frmReceive.cs，其中，frmSend.cs 用来作为“邮件发送”窗体，frmReceive.cs 用来作为“邮件接收”窗体。

(3) 主要代码如下。

```
private void btnSend_Click(object sender, EventArgs e) //单击“邮件发送”窗体中的“发送”按钮
{
    try
    {
        MailAddress from = new MailAddress(txtSend.Text); //设置邮件发送人
        MailAddress to = new MailAddress(txtTo.Text); //设置邮件接收人
        MailMessage message = new MailMessage(from, to); //创建一个 MailMessage 对象
        message.Subject = Base64Encode(txtSubject.Text); //设置发送邮件的主题
        message.Body = Base64Encode(txtContent.Text); //设置发送邮件的内容
        SmtpClient client = new SmtpClient(txtServer.Text, Convert.ToInt32(txtPort.Text)); //创建 SmtpClient 邮件发送对象
        client.Credentials = new System.Net.NetworkCredential(txtName.Text, txtPwd.Text); //设置用于验证发件人身份的凭据
        client.Send(message); //发送邮件
        MessageBox.Show("发送成功");
    }
}
```

```

        }
    catch
    {
        MessageBox.Show("发送失败!");
    }
}

private void btnReceive_Click(object sender, EventArgs e) //单击“邮件接收”窗体中的“接收”按钮
{
    try
    {
        if (txtNum.Text != "") //若要读取的邮件的序号不为空
        {
            if (Convert.ToInt32(txtNum.Text) > k || Convert.ToInt32(txtNum.Text) <= 0) //如输入邮件序号不合法
            {
                MessageBox.Show("输入的索引错误!");
            }
            else //若输入的邮件序号合法
            {
                richTextBox1.Clear(); //清空 RichTextBox 控件
                string[] arrRets;
                arrRets = PopMail(tcpclient, Convert.ToInt32(txtNum.Text)); //获得远程主机上指定邮件的信息
                richTextBox1.AppendText("当前是第" + txtNum.Text + "封信" + "\n"); //获取邮件是第几封
                richTextBox1.AppendText("邮件日期：" + arrRets[1] + "\n"); //获取邮件日期
                richTextBox1.AppendText("发信人：" + arrRets[2] + "\n"); //获取发信人
                richTextBox1.AppendText("收信人：" + arrRets[3] + "\n"); //获取收信人
                richTextBox1.AppendText("邮件主题：" + Base64Decode(arrRets[4]) + "\n"); //获取邮件主题
                richTextBox1.AppendText("邮件内容：" + Base64Decode(arrRets[5])); //获取邮件内容
            }
        }
        else
        {
            MessageBox.Show("邮件索引错误!");
        }
    }
    catch
    {
    }
}
}

```

**说明：**因为本实例用到 POP3 服务器，所以本实例只能在 Windows Server 2003 或更高级的服务器版操作系统中运行。

**DIY：**调用 OutLook 发送邮件。提示：调用 OutLook 发送邮件时主要用到 API 函数 ShellExecute，该函数主要用来查找与指定文件关联在一起的程序文件名。(20分)(实例位置：光盘\mr\18\qjyy\05\_diy)

情景应用 DIY 栏目分数统计：

DIY 题目	1	2	3	4	5	总分数	
分数							

## 18.5 自我测试

### 一、选择题（每题 10 分，5 道题）

1. 在下面的选项中，（ ）是电子邮件传输协议。  
A. HTTP      B. SMTP      C. POP      D. HTML
2. 文件传输是基于（ ）的。  
A. TCP 协议      B. UDP 协议      C. TCP 和 UDP 协议      D. 都不是
3. 下面关于 UDP 协议的描述中，错误的是（ ）。  
A. 数据通过 UDP 传输存在丢包的可能，安全性不如 TCP  
B. UDP 传输速度一定比 TCP 快  
C. UDP 的数据传输是面向无连接的，而 TCP 的数据传输是面向有连接的  
D. 视频、聊天等数据的传输都可以使用 UDP
4. 在 TCP/IP 协议中，如果出现阻塞情况，下面（ ）情况最有可能发生。  
A. 连接错误      B. 释放缓存      C. 丢包      D. 包错误
5. 下面关于网络端口的有效范围，（ ）个是正确的。  
A. 0~255 之间      B. 0~1023 之间      C. 0~65535 之间      D. 无限制

### 二、填空题（每题 10 分，5 道题）

1. 服务器是指提供信息的计算机或程序，客户机是指请求信息的计算机或程序，网络主要是用来连接（ ）与（ ）实现两者相互通信的。
2. IPAddress 类包含计算机在 IP 网络上的地址，它主要用来提供（ ）。
3. IPEndPoint 类包含应用程序连接到主机上的服务所需要的端口信息，它主要用来将网络端点表示为（ ）和（ ）。
4. TcpClient 类用于在同步阻止模式下通过网络来（ ）、（ ）和（ ）。
5. TcpListener 类用于在阻止同步模式下（ ）和（ ）。

测试分数统计：

类别	第 1 题	第 2 题	第 3 题	第 4 题	第 5 题
选择题分数					
填空题分数					
总分数					

## 18.6 行动指南

开始日期： 年 月 日

结束日期： 年 月 日

序号	内 容	行动指南	
		分数>75 分	优秀，基本功掌握得不错，加油！
1	照猫画虎栏目	75 分>分数>50 分	及格，知识掌握得不牢，重新做一遍照猫画虎。
	分数（ ）	分数<50 分	请使用光盘中提供的“实战能力测试系统”进行提高训练。

续表

序号	内 容	行 动 指 南	
1	情景应用栏目 分数( )	分数>75 分	优秀，综合应用能力很强。
		75 分>分数>50 分	及格，综合应用能力需提高，再练一遍情景应用。
		分数<50 分	请使用光盘中提供的“实战能力测试系统”进行提高训练。
	自我测试栏目 分数( )	分数>75 分	优秀，有成为编程高手的潜质。
		分数<75 分	请使用光盘中提供的“实战能力测试系统”进行提高训练。
	综合评价	反复训练后，以上各项分数都在 75 分以上，方可进入下一堂课学习。	
2	编程能力培养：本栏目提供了一些实践题目，对于培养编程能力很有效，要做好。	1. 尝试制作一个网络播放器。提示：在制作网络播放器时，主要用到 Windows 系统中的 Window Media Player 组件，使用 Windows Media Player 组件播放网络上的歌曲时，只需将其 URL 属性设置为网络歌曲对应的 URL 地址即可。 2. 获取指定地区天气预报。提示：获取天气预报时主要用到 <code>HttpWebRequest</code> 类的 <code>GetResponse</code> 方法和 <code>HttpWebResponse</code> 类的 <code>GetResponseStream</code> 方法。其中 <code>GetResponse</code> 方法用来返回来自 Internet 资源的响应； <code>GetResponseStream</code> 方法用来获取流，该流用于读取来自服务器的响应体。	
3	编程习惯培养：本堂课培养读者在学习开发中记笔记的习惯，把开发中遇到的问题、总结的经验记录下来。		
4	创新能力培养：看看身边有没有可以用编程解决的问题，记录到右边的表格中。根据学习进度，尝试编写解决这些问题的小程序。		

## 18.7 成功可以复制——80后新贵、泡泡网 CEO 李想

1981 年，李想出生在河北石家庄。小时候，李想不太喜欢课堂，他总喜欢在实践中快学快用。需要什么才学什么，学了什么就马上用起来。上初中时，老师的一句激励：“学习不好不要紧，但一定要做个优秀的人”成为他在课堂上所学到的最有价值的知识。高三时，李想迷上了个人网站，他把所有的时间都用在计算机上，像许多电脑迷一样，他也建了一个个人网站，“一开始是自己做着玩，但我这个人喜欢争强好胜，别人做得好，我就要比别人做得更好。”他把自己喜欢的计算机硬件产品都放在网上，有很多人上网和他交流，慢慢地就有了访问量，几个月后访问量达到 1 万人次/天。这时候，广告商就找上门来。当时李想的网站每个月有六七千元的广告收入，这对一个学生来说，简直太奢侈了。

但好景不长。1999 年下半年互联网泡沫破灭，李想的广告一个都没了。虽然遭遇挫折，但李想并不气馁，因为做网站是一个可以让他全力以赴的事情，也是他特别喜欢的事情。高中毕业后，李想没有选择继续读书，而是选择了自己创业。2000 年，李想和一个朋友创办了 PCPOP（电脑泡泡）网站，初始投资就是做网站淘到的第一桶金，将近 10 万元。新网站很快就有了访问量，但却见不着效益，因为在石家庄没有收入机会，李想决定移师北京。

2001 年年底，李想到了北京，开始租了一间民房，半年后，网站访问量每天有 3~5 万人，广告商又找上门来，而且开出了更高的广告价格。2002 年，他们搬进了写字楼，2003 年他们的收入达到 200 万元，2004 年超过了 1000 万元。李想是如何实现爆发式增长的呢？一方面，PCPOP 网站的内容基本上都是原创的，公司身居中关村，每天都能关注市场的最新动态，另外厂商也会把最新的产品拿来给他们测评，他们的专业性特征明显，而且更新速度非常快。另一方面，这些年市场消费行为也在发生变化。原先顾客要买一件商品，可能到电脑市场转上好几天，而现在更多的用户可以先在网上找到想要的东西，谁的网站专业性强，更新速度快，自然就会吸引更多的眼球。

因为兴趣和爱好，李想在 24 岁就成就了自己的互联网传奇。2010 年 4 月，李想成为搜狐网的首席策划师，我们期待着他的更大成功！

### 经典语录

做事要坚定，我当初如果不放弃考大学，也许今天我只是一名普通的打工者；做事要认真，如果做一件事情比别人多付出 5% 的努力，就有可能拿到比别人多 200% 的回报。

### 深度评价

在专业 IT 网站前 5 名中，除了李想的 PCPOP 网站是白手起家做起来的，其他都是靠几千万美金的投资“砸”出来的。李想的成功，得益于他做事认真和全力以赴的态度；得益于他“比别人多付出 5% 的努力”的精神；得益于他“学习不好不要紧，但一定要做个优秀的人”的人生信条。



泡泡网首页

# 第 19 堂课

## 线程的使用

( 视频讲解：142分钟)

如果一次只完成一件事情，那是一个不错的想法，但事实上很多事情都是同时进行的，所以在 C# 中为了模拟这种状态，引入了线程机制，简单地说，当程序同时完成多件事情时，就是所谓的多线程程序。多线程运用广泛，开发人员可以使用多线程对要执行的操作分段执行，这样可以大大提高程序的运行速度和性能。本堂课将由浅入深地介绍多线程，除了介绍多线程的概念之外，还配备了实例让读者了解如何使程序具有多线程功能。

学习摘要：

- » 了解线程的基础知识
- » 掌握线程类 Thread 的使用
- » 掌握如何创建线程
- » 掌握如何挂起和恢复线程的执行
- » 掌握如何休眠线程
- » 掌握如何终止线程的执行
- » 掌握如何设置线程执行的优先级
- » 了解线程同步机制
- » 掌握常用的几种实现线程同步的方法

## 19.1 线程概述

每个正在操作系统上运行的应用程序都是一个进程，一个进程可以包括一个或多个线程。线程是操作系统分配处理器时间的基本单元，在进程中可以有多个线程同时执行代码。每个线程都维护异常处理程序、调度优先级和一组系统用于在调度该线程前保存线程上下文的结构。线程上下文包括为使线程在线程的宿主进程地址空间中无缝地继续执行所需的所有信息，包括线程的 CPU 寄存器组和堆栈。本节将对线程进行详细讲解。

### 19.1.1 线程的定义与分类

世间万物都会同时完成很多工作，如人体同时进行呼吸、血液循环、思考问题等活动，用户既可以用计算机听歌，又可以使用它阅读文件，而这些活动完全可以同时进行，这种思想放在 C# 中被称为并发，而并发完成的每一件事情则被称为线程。

在人们的生活中，并发机制非常重要，但并不是所有的程序语言都支持线程，在以往的程序中，多以一个任务完成后再进行下一个项目的模式进行开发，这样下一个任务的开始必须等待前一个任务的结束。C# 语言提供并发机制，程序员可以在程序中执行多个线程，每一个线程完成一个功能，并与其他线程并发执行，这种机制被称为多线程。

多线程是非常复杂的机制，如果此时读者不能体会这句话的含义，可以尝试同时阅读 3 本书，首先阅读第一本书第 1 章，然后再阅读第二本书的第 1 章，再阅读第 3 本书的第 1 章，回头再阅读第一本书的第 2 章，依次类推，不用很长时间读者就可以体会多线程的复杂性。

既然多线程这么复杂，那么它在操作系统上是如何工作的呢？其实 C# 中的多线程在每个操作系统中的运行方式也存在差异。笔者着重说明多线程在 Windows 操作系统的运行模式，Windows 操作系统是多任务操作系统，它以进程为单位，一个进程是一个包含有自身地址的程序，每个独立执行的程序称为进程，也就是正在执行的程序，在系统中可以分配给每个进程一段有限的使用 CPU 的时间（也可以称为 CPU 时间片），CPU 在片段时间中执行某个进程，然后下一个时间片又跳至另一个进程中去执行。由于 CPU 转换较快，所以使得每个进程好像是同时执行一样。

图 19.1 表明了 Windows 操作系统的执行模式。

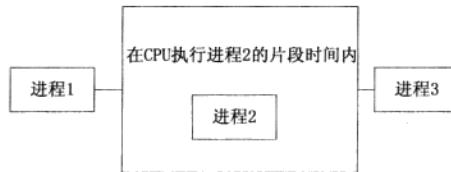


图 19.1 Windows 操作系统的执行模式

线程是进程中的执行流程，一个进程中可以同时包括多个线程，每个线程也可以得到一小段程序的执行时间，这样一个进程就可以具有多个并发执行的线程。在单线程中，程序代码按调用顺序依次往下执行，如果需要一个进程同时完成多段代码的操作，就需要产生多线程。

**例 19.01** 新建一个 Windows 应用程序，程序会在 Program.cs 文件中自动生成一个 Main 方法，该方法就是主线程的启动入口点。Main 方法代码如下。

```
[STAThread]
static void Main()
{
    Application.EnableVisualStyles(); //启用应用程序的可视样式
    //将某些控件上定义的 UseCompatibleTextRendering 属性设置为应用程序范围内的默认值
    Application.SetCompatibleTextRenderingDefault(false);
    Application.Run(new Form1()); //运行窗体 Form1
}
```

### 19.1.2 多线程的使用

一般情况下，需要用户交互的软件都必须尽可能快地对用户的活动作出反应，以便提供丰富多彩的用户体验，但同时它又必须执行必要的计算以便尽可能快地将数据呈现给用户，这时可以使用多线程来实现。

#### 1. 多线程的优点

要提高对用户的响应速度并且处理所需数据以便几乎同时完成工作，使用多线程是一种最为强大的技术，在具有一个处理器的计算机上，多线程可以通过利用用户事件之间很小的时间段在后台处理数据来达到这种效果。例如，通过使用多线程，在另一个线程正在重新计算同一应用程序中的电子表格的其他部分时，用户可以编辑该电子表格。

单个应用程序域可以使用多线程来完成以下任务。

- 通过网络与 Web 服务器和数据库进行通信。
- 执行占用大量时间的操作。
- 区分具有不同优先级的任务。
- 使用用户界面可以在将时间分配给后台任务时仍能快速作出响应。

#### 2. 多线程的缺点

使用多线程有好处，同时也有坏处，建议一般不要在程序中使用太多的线程，这样可以最大限度地减少操作系统资源的使用，并提高性能。

如果在程序中使用了多线程，可能会产生如下问题。

- 系统将为进程、AppDomain 对象和线程所需的上下文信息使用内存。因此，可以创建的进程、AppDomain 对象和线程的数目会受到可用内存的限制。
- 跟踪大量的线程将占用大量的处理器时间。如果线程过多，则其中大多数线程都不会产生明显的进度。如果大多数当前线程处于一个进程中，则其他进程中的线程的调度频率就会很低。
- 使用许多线程控制代码执行非常复杂，并可能产生许多 bug。
- 销毁线程需要了解可能发生的问题并对那些问题进行处理。

### 19.1.3 线程的生命周期

线程具有生命周期，它包含 3 个状态，分别为出生状态、就绪状态和运行状态。出生状态就是用户在创建线程时处于的状态，在用户使用该线程实例调用 Start 方法之前，线程都处于出生状态；当用户调用 Start 方法后，线程处于就绪状态（又被称为可执行状态）；当线程得到系统资源后就进入运行状态。

一旦线程进入可执行状态，它会在就绪与执行状态下辗转，同时也有可能进入等待、休眠、阻塞或死亡状态。当处于运行状态下的线程调用 Thread 类中 Suspend 方法时，该线程处于等待状态，进入等待状态

的线程必须调用 Thread 类中的 Resume 方法才能被唤醒；当线程调用一个 Thread 类中的 Sleep 方法时，线程就进入休眠状态；如果一个线程在运行状态下发出输入/输出请求，该线程将进入阻塞状态，在其等待输入/输出结束时，线程进入就绪状态，对于阻塞的线程来说，即使系统资源空闲，线程依然不能回到执行状态；当线程执行完毕时，线程进入死亡状态。

**说明：**关于使线程处于不同状态下的方法笔者会在 19.3 节中进行详细讲解，在这里读者只需对线程的多个状态进行了解即可。

图 19.2 描述了线程的生命周期的各个状态。

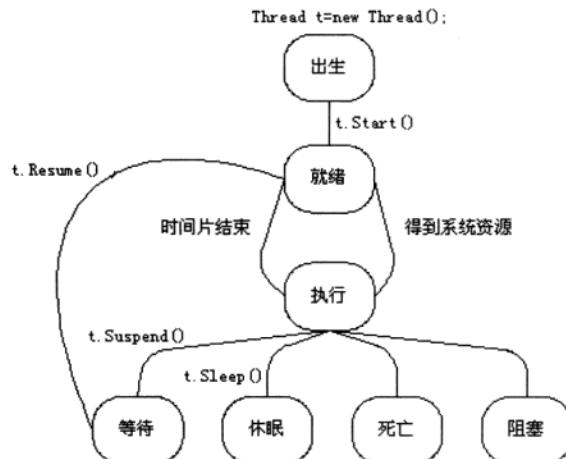


图 19.2 线程的生命周期状态图

虽然多线程看起来像同时执行，但事实上在同一时间点上只有一个线程被执行，只是线程之间切换较快，所以才会使人产生线程是同时进行的假象。在 Windows 操作系统中，系统会为每个线程分配一小段 CPU 时间片，当 CPU 时间结束时，将当前线程转换到下一个线程，即使该线程没有结束。

## 19.2 C# 中的线程类 Thread

C# 中对线程进行操作时，主要用到了 Thread 类，通过使用 Thread 类，可以对线程进行创建、挂起、恢复、休眠、终止及设置优先级等操作，本节将对 Thread 类进行详细讲解。

Thread 类位于 System.Threading 命名空间下，System.Threading 命名空间提供一些使得可以进行多线程编程的类和接口。除同步线程活动和访问数据的类（Mutex、Monitor、Interlocked 和 AutoResetEvent 等）外，该命名空间还包含一个 ThreadPool 类（它允许用户使用系统提供的线程池）和一个 Timer 类（它在线程池线程上执行回调方法）。

Thread 类主要用于创建并控制线程、设置线程优先级并获取其状态。一个进程可以创建一个或多个线程以执行与该进程关联的部分程序代码，线程执行的程序代码由 ThreadStart 委托或 ParameterizedThreadStart 委托指定。

在线程运行期间，不同的时刻会表现为不同的状态，但它总是处于由 ThreadState 定义的一个或多个状

态中。用户可以通过使用 ThreadPriority 枚举为线程定义优先级，但不能保证操作系统会接受该优先级。

Thread 类的常用属性及说明如表 19.1 所示。

表 19.1 Thread 类的常用属性及说明

属性	说明
CurrentThread	获取当前正在运行的线程
IsAlive	获取一个值，该值指示当前线程的执行状态
Name	获取或设置线程的名称
Priority	获取或设置一个值，该值指示线程的调度优先级
ThreadState	获取一个值，该值包含当前线程的状态

Thread 类的常用方法及说明如表 19.2 所示。

表 19.2 Thread 类的常用方法及说明

方法	说明
Abort	在调用此方法的线程上引发 ThreadAbortException，以开始终止此线程的过程。调用此方法通常会终止线程
Join	阻止调用线程，直到某个线程终止时为止
Resume	继续已挂起的线程
Sleep	将当前线程阻止指定的毫秒数
Start	使线程被安排进行执行
Suspect	挂起线程，或者如果线程已挂起，则不起作用

**例 19.02** 下面演示使用 Thread 类的相关方法和属性，开始运行一个线程，并获得该线程的相关信息，程序开发步骤如下。（实例位置：光盘\mr\19\s\19.02）

(1) 新建一个 Windows 应用程序，并将其命名为 UseThread，默认窗体为 Form1.cs。

(2) 在 Form1 窗体中添加一个 RichTextBox 控件，用来显示获得的线程相关信息。

(3) 程序主要代码如下。

```
private void Form1_Load(object sender, EventArgs e)
{
    string strInfo = string.Empty; // 定义一个字符串，用来记录线程相关信息
    Thread myThread = new Thread(new ThreadStart(threadOut)); // 创建 Thread 线程对象
    myThread.Start(); // 启动主线程
    // 获取线程相关信息
    strInfo = "线程唯一标识符：" + myThread.ManagedThreadId;
    strInfo += "\n线程名称：" + myThread.Name;
    strInfo += "\n线程状态：" + myThread.ThreadState.ToString();
    strInfo += "\n线程优先级：" + myThread.Priority.ToString();
    strInfo += "\n是否为后台线程：" + myThread.IsBackground;
    Thread.Sleep(1000); // 使主线程休眠 1 秒钟
    myThread.Abort("退出"); // 通过主线程阻止新开线程
    myThread.Join(); // 等待新开的线程结束
    MessageBox.Show("线程运行结束");
    richTextBox1.Text = strInfo;
}

public void threadOut()
{
```

```
    MessageBox.Show("主线程开始运行");
}
```

**◆ 注意：**在程序中使用线程时，需要在命名空间区域添加 System.Threading 命名空间，下面遇到时将不再提示。

运行程序，先后弹出如图 19.3 和图 19.4 所示的对话框，然后显示如图 19.5 所示的主窗体，并在主窗体中显示获得的线程相关信息。



图 19.3 线程开始运行



图 19.4 线程运行结束

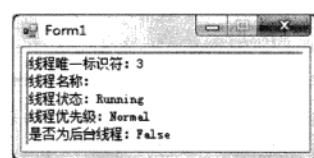


图 19.5 主窗体

## 19.3 线程调度

线程调度即对线程的操作，主要包括线程的创建、挂起、恢复、休眠、终止及设置线程的优先级，本节将对其进行详细讲解。

### 19.3.1 创建线程

创建一个线程非常简单，只需将其声明并为其提供线程起始点处的方法委托即可。在创建新的线程时需要使用 Thread 类，Thread 类具有接受一个 ThreadStart 委托或 ParameterizedThreadStart 委托的构造函数，该委托包装了调用 Start 方法时由新线程调用的方法。创建了 Thread 类的对象之后，线程对象已存在并已配置，但并未创建实际的线程，这时，只有在调用 Start 方法后才会创建实际的线程。

Start 方法用来使线程被安排进行执行，它有两种重载形式，下面分别介绍。

(1) 导致操作系统将当前实例的状态更改为 ThreadState.Running。

语法：public void Start()

说明：该方法无参数、无返回值。

(2) 使操作系统将当前实例的状态更改为 ThreadState.Running，并选择线程执行所需要的方法。语法如下。

语法：public void Start(Object parameter)

说明：参数 parameter 表示一个对象，包含线程执行的方法要使用的数据。

**◆ 注意：**如果线程已经终止，就无法通过再次调用 Start 方法来重新启动。

**例 19.03** 创建一个控制台应用程序，其中自定义一个静态的 void 类型方法 createThread，然后在 Main 方法中通过创建 Thread 对象创建一个新的线程，并调用 Start 方法启动该线程。代码如下。（实例位置：光盘\mr\19\s\19.03）

```
static void Main(string[] args)
{
    Thread myThread; //声明线程
```

```

myThread = new Thread(new ThreadStart(createThread)); //用线程起始点的 ThreadStart 委托创建该线程的实例
myThread.Start(); //启动线程
Console.ReadLine();
}
public static void createThread()
{
    Console.WriteLine("创建线程");
}

```

程序运行结果如图 19.6 所示。

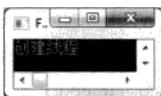


图 19.6 运行结果

注意：线程的入口（本例中为 createThread）不带任何参数。

### 19.3.2 线程的挂起与恢复

创建完一个线程并启动之后，还可以挂起、恢复、休眠或终止它，本节主要对线程的挂起与恢复进行讲解。

线程的挂起与恢复分别可以通过调用 Thread 类中的 Suspend 方法和 Resume 方法实现，下面对这两个方法进行详细介绍。

#### 1. Suspend 方法

该方法用来挂起线程，如果线程已挂起，则不起作用。

语法：public void Suspend()

说明：调用 Suspend 方法挂起线程时，.NET 允许要挂起的线程再执行几个指令，目的是为了到达.NET 认为线程可以安全挂起的状态。

#### 2. Resume 方法

该方法用来继续已挂起的线程。

语法：public void Resume()

说明：通过 Resume 方法来恢复被暂停的线程时，无论调用了多少次 Suspend 方法，调用 Resume 方法均会使另外的线程脱离挂起状态，并导致当前线程继续执行。

**例 19.04** 创建一个控制台应用程序，其中通过创建 Thread 对象创建一个新的线程，然后调用 Start 方法启动该线程，之后先后调用 Suspend 方法和 Resume 方法挂起和恢复创建的线程。代码如下。（实例位置：光盘\mr\19\s\19.04）

```

static void Main(string[] args)
{
    Thread myThread; //声明线程
    myThread = new Thread(new ThreadStart(createThread)); //用线程起始点的 ThreadStart 委托创建该线程的实例
    myThread.Start(); //启动线程
    myThread.Suspend(); //挂起线程
    myThread.Resume(); //恢复挂起的线程
    Console.ReadLine();
}

```

```

    }
    public static void createThread()
    {
        Console.WriteLine("创建线程");
    }
}

```

### 19.3.3 线程休眠

线程休眠主要通过 Thread 类的 Sleep 方法实现，该方法用来将当前线程阻止指定的时间，它有两种重载形式，下面分别进行介绍。

- (1) 将当前线程挂起指定的时间。

语法：public static void Sleep(int millisecondsTimeout)

说明：参数 millisecondsTimeout 表示线程被阻止的毫秒数，指定零以指示应挂起此线程以使其他等待线程能够执行；指定 Infinite 以无限期阻止线程。

- (2) 将当前线程阻止指定的时间，语法如下。

语法：public static void Sleep(TimeSpan timeout)

说明：参数 timeout 表示线程被阻止的时间量的 TimeSpan，指定零以指示应挂起此线程以使其他等待线程能够执行；指定 Infinite 以无限期阻止线程。

**例 19.05** 下面代码用来使当前线程休眠 1 秒钟，代码如下。

```
Thread.Sleep(1000); //使线程休眠 1 秒钟
```

### 19.3.4 终止线程

终止线程可以分别使用 Thread 类的 Abort 方法和 Join 方法实现，下面对这两个方法进行详细介绍。

#### 1. Abort 方法

Abort 方法用来终止线程，它有两种重载形式，下面分别介绍。

- (1) 终止线程，在调用此方法的线程上引发 ThreadAbortException 异常，以开始终止此线程的过程。

语法：public void Abort()

(2) 终止线程，在调用此方法的线程上引发 ThreadAbortException 异常，以开始终止此线程并提供有关线程终止的异常信息的过程。

语法：public void Abort(Object stateInfo)

说明：参数 stateInfo 是一个对象，它包含应用程序特定的信息（如状态），该信息可供正被终止的线程使用。

**例 19.06** 创建一个控制台应用程序，在其中开始了一个线程，然后调用 Thread 类的 Abort 方法终止了已开启的线程。代码如下。（实例位置：光盘\mr\19\s1\19.06）

```

static void Main(string[] args)
{
    Thread myThread; //声明线程
    myThread = new Thread(new ThreadStart(createThread)); //用线程起始点的 ThreadStart 委托创建该线程的实例
    myThread.Start(); //启动线程
    myThread.Abort(); //终止线程
    Console.ReadLine();
}

```

```
public static void createThread()
{
    Console.WriteLine("线程实例");
}
```

**◆ 注意：**线程的 Abort 方法用于永久地停止托管线程。在调用 Abort 方法时，公共语言运行库在目标线程中引发 ThreadAbortException 异常，目标线程可捕捉此异常。一旦线程被中止，它将无法重新启动。

## 2. Join 方法

Join 方法用来阻止调用线程，直到某个线程终止时为止，它有 3 种重载形式，下面分别介绍。

(1) 在继续执行标准的 COM 和 SendMessage 消息处理期间，阻止调用线程，直到某个线程终止为止。

语法：public void Join()

(2) 在继续执行标准的 COM 和 SendMessage 消息处理期间，阻止调用线程，直到某个线程终止或经过指定时间为止。

语法：public bool Join(int millisecondsTimeout)

说明：参数 millisecondsTimeout 表示等待线程终止的毫秒数。如果线程已终止，则返回值为 true；如果线程在经过了 millisecondsTimeout 参数指定的时间后未终止，则返回值为 false。

(3) 在继续执行标准的 COM 和 SendMessage 消息处理期间，阻止调用线程，直到某个线程终止或经过指定时间为止。

语法：public bool Join(TimeSpan timeout)

说明：参数 timeout 表示等待线程终止的时间量的 TimeSpan。如果线程已终止，则返回值为 true；如果线程在经过了 timeout 参数指定的时间量后未终止，则返回值为 false。

**例 19.07** 创建一个控制台应用程序，其中调用了 Thread 类的 Join 方法等待线程终止，代码如下。（实例位置：光盘\mr\19\s\19.07）

```
static void Main(string[] args)
{
    Thread myThread; //声明线程
    myThread = new Thread(new ThreadStart(createThread)); //用线程起始点的 ThreadStart 委托创建该线程的实例
    myThread.Start(); //启动线程
    myThread.Join(); //阻止调用该线程，直到该线程终止为止
    Console.ReadLine();
}

public static void createThread()
{
    Console.WriteLine("线程实例");
}
```

**◆ 注意：**如果在应用程序中使用了多线程，辅助线程还没有执行完毕，在关闭窗体时必须要关闭辅助线程，否则会引发异常。

## 19.3.5 线程的优先级

线程优先级指定一个线程相对于另一个线程的相对优先级，每个线程都有一个分配的优先级。在公共语言运行库内创建的线程最初被分配为 Normal 优先级，而在公共语言运行库外创建的线程，在进入公共语言运行库时将保留其先前的优先级。

线程是根据其优先级而调度执行的，用于确定线程执行顺序的调度算法随操作系统的不同而不同。在某些操作系统下，具有最高优先级（相对于可执行线程而言）的线程经过调度后总是首先运行。如果具有相同优先级的多个线程都可用，则程序将遍历处于该优先级的线程，并为每个线程提供一个固定的时间片来执行。只要具有较高优先级的线程可以运行，具有较低优先级的线程就不会执行。如果在给定的优先级上不再有可运行的线程，则程序将移到下一个较低的优先级并在该优先级上调度线程以执行。如果具有较高优先级的线程可以运行，则具有较低优先级的线程将被抢先，并允许具有较高优先级的线程再次执行。除此之外，当应用程序的用户界面在前台和后台之间移动时，操作系统还可以动态调整线程优先级。

例如，在多任务操作系统中，每个线程都会得到一小段 CPU 时间片进行执行，在时间结束时，将轮换另一个线程进入执行状态，这时系统会选择与当前线程优先级相同的线程进行执行。系统始终选择就绪状态下优先级较高的线程进入执行状态。图 19.7 表明了处于各个优先级状态下的线程的运行顺序。

在图 19.7 中，优先级为 5 的线程 A 首先得到 CPU 时间片，当该时间结束后，轮换到与线程 A 相同优先级的线程 B，当线程 B 的运行时间结束后，会继续轮换到线程 A，直到线程 A 与线程 B 都执行完毕，才会轮换到线程 C，当线程 C 结束后，最后才会轮到线程 D。

**说明：**一个线程的优先级不影响该线程的状态，该线程的状态在操作系统可以调度该线程之前必须为 Running。

线程的优先级值及说明如表 19.3 所示。

表 19.3 线程的优先级值及说明

优 先 级 值	说 明
AboveNormal	可以将 Thread 安排在具有 Highest 优先级的线程之后，在具有 Normal 优先级的线程之前
BelowNormal	可以将 Thread 安排在具有 Normal 优先级的线程之后，在具有 Lowest 优先级的线程之前
Highest	可以将 Thread 安排在具有任何其他优先级的线程之前
Lowest	可以将 Thread 安排在具有任何其他优先级的线程之后
Normal	可以将 Thread 安排在具有 AboveNormal 优先级的线程之后，在具有 BelowNormal 优先级的线程之前。默认情况下，线程具有 Normal 优先级

开发人员可以通过访问线程的 Priority 属性来获取和设置其优先级。Priority 属性用来获取或设置一个值，该值指示线程的调度优先级，其语法如下。

语法：public ThreadPriority Priority { get; set; }

说明：属性值是 ThreadPriority 枚举值之一，默认值为 Normal。

**例 19.08** 创建一个控制台应用程序，其中创建了两个 Thread 线程对象，并设置第一个 Thread 对象的优先级为最低，然后调用 Start 方法开启这两个线程，代码如下。（实例位置：光盘\mr\19\sl\19.08）

```
static void Main(string[] args)
{
    Thread thread1=new Thread(new ThreadStart(Thread1)); //使用自定义方法 Thread1 声明线程
    thread1.Priority = ThreadPriority.Lowest; //设置线程的调度优先级
```

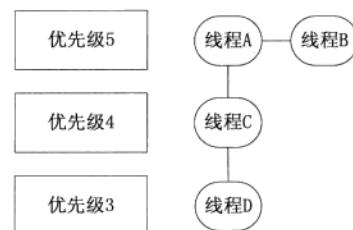


图 19.7 处于各个优先级状态下的线程的运行顺序

```

Thread thread2 = new Thread(new ThreadStart(Thread2));
thread1.Start(); //使用自定义方法 Thread2 声明线程
thread2.Start(); //开启线程一
Console.ReadLine(); //开启线程二
}
static void Thread1()
{
    Console.WriteLine("线程一");
}
static void Thread2()
{
    Console.WriteLine("线程二");
}

```

程序运行结果如图 19.8 所示。



图 19.8 设置线程的优先级

## 19.4 线程同步

在单线程程序中，每次只能做一件事情，后面的事情需要等待前面的事情完成后才可以进行，但是如果使用多线程程序，就会发生两个线程抢占资源的问题，如两个人同时说话、两个人同时过同一个独木桥等，所以在多线程编程中，需要防止这些资源访问的冲突。C#提供线程同步机制来防止资源访问的冲突，其中主要用到 lock 关键字、Monitor 类和 Mutex 类。本节将对线程同步机制及其实现进行详细讲解。

### 19.4.1 线程同步机制

应用程序中使用多线程的一个好处是每个线程都可以异步执行。对于 Windows 应用程序，耗时的任务可以在后台执行，而使应用程序窗口和控件保持响应。对于服务器应用程序，多线程处理提供了用不同线程处理每个传入请求的能力，否则，在完全满足前一个请求之前，将无法处理每个新请求。然而，线程的异步性意味着必须协调对资源（如文件句柄、网络连接和内存）的访问，否则，两个或更多的线程可能在同一时间访问相同的资源，而每个线程都不知道其他线程的操作，结果将产生不可预知的数据损坏。

线程同步是指并发线程高效、有序地访问共享资源所采用的技术。所谓同步，是指某一时刻只有一个线程可以访问资源，只有当资源所有者主动放弃了代码或资源的所有权时，其他线程才可以使用这些资源。

### 19.4.2 使用 lock 关键字实现线程同步

lock 关键字可以用来确保代码块完成运行，而不会被其他线程中断，它是通过在代码块运行期间为给定对象获取互斥锁来实现的。

lock 语句以关键字 lock 开头，它有一个作为参数的对象，在该参数的后面还有一个一次只能有一个线程执行的代码块。lock 语句语法格式如下。

```
Object thisLock = new Object();
lock (thisLock)
{
    //要运行的代码块
}
```

提供给 lock 语句的参数必须为基于引用类型的对象，该对象用来定义锁的范围。严格来说，提供给 lock 语句的参数只是用来唯一标识由多个线程共享的资源，所以它可以是任意类实例，实际上，此参数通常表示需要进行线程同步的资源。例如，如果一个容器对象将被多个线程使用，则可以将该容器传递给 lock 语句，而 lock 语句中的代码块将访问该容器。只要其他线程在访问该容器前先锁定该容器，则对该对象的访问将是安全同步的。

通常，最好避免锁定 public 类型或不受应用程序控制的对象实例。例如，如果该实例可以被公开访问，则 lock (this) 可能会有问题，因为不受控制的代码也可能会锁定该对象，这将可能导致死锁，即两个或更多个线程等待释放同一对象。出于同样的原因，锁定公共数据类型（相比于对象）也可能导致问题，锁定字符串尤其危险，因为字符串被公共语言运行库（CLR）“暂留”，这意味着在整个程序中任何给定字符串都只有一个实例，因此，只要在应用程序进程中的任何具有相同内容的字符串上放置了锁，就将锁定应用程序中该字符串的所有实例，因此，最好锁定不会被暂留的私有或受保护成员。

 **说明：**事实上 lock 语句是用 Monitor 类来实现的，它等效于 try/finally 语句块，使用 lock 关键字通常比直接使用 Monitor 类更可取，一方面是因为 lock 更简洁，另一方面是因为 lock 确保了即使受保护的代码引发异常，也可以释放基础监视器，这是通过 finally 关键字来实现的，无论是否引发异常，它都执行关联的代码块。

**例 19.09** 使用 lock 关键字锁定当前线程，在当前线程释放锁定之前，其他线程将无法访问该锁定标记。代码如下。

```
static void Main(string[] args)
{
    lock (new Program()) //锁定当前线程
    {
        Console.WriteLine("锁定线程");
        Console.ReadLine();
    }
}
```

 **说明：**如果在静态方法中使用 lock 关键字，则不能使用 this。

### 19.4.3 使用 Monitor 驱动对象实现线程同步

Monitor 类提供了同步对对象的访问机制，它通过向单个线程授予对象锁来控制对对象的访问，对象锁提供限制访问代码块（通常称为临界区）的能力。当一个线程拥有对象锁时，其他任何线程都不能获取该锁。

Monitor 类的主要功能如下。

- 它根据需要与某个对象相关联。
- 它是未绑定的，也就是说可以直接从任何上下文调用它。
- 不能创建 Monitor 类的实例。

Monitor 类的常用方法及说明如表 19.4 所示。

表 19.4 Monitor 类的常用方法及说明

方 法	说 明
Enter	在指定对象上获取排他锁
Exit	释放指定对象上的排他锁
Pulse	通知等待队列中的线程锁定对象状态的更改
PulseAll	通知所有的等待线程对象状态的更改
TryEnter	试图获取指定对象的排他锁
Wait	释放对象上的锁并阻止当前线程，直到它重新获取该锁

注意：使用 Monitor 类锁定的是对象（即引用类型）而不是值类型。

**例 19.10** 创建一个控制台应用程序，其中自定义了一个 LockThread 方法。在该方法中首先使用 Monitor 类的 Enter 方法锁定当前线程，然后再调用 Monitor 类的 Exit 方法释放当前线程，最后在 Main 方法中通过 Program 的对象调用 LockThread 自定义方法。代码如下。（实例位置：光盘\mr\19\sl\19.10）

```
static void Main(string[] args)
{
    Program myProgram = new Program();           //创建对象
    myProgram.LockThread();                      //调用锁定线程方法
    Console.ReadLine();
}

void LockThread()
{
    Monitor.Enter(this);                        //锁定当前线程
    Console.WriteLine("锁定线程以实现线程同步");
    Monitor.Exit(this);                         //释放当前线程
}
```

程序运行结果如图 19.9 所示。



图 19.9 运行结果

说明：Monitor 类有很好的控制能力，例如，它可以使使用 Wait 方法指示活动的线程等待一段时间，当线程完成操作时，还可以使用 Pulse 方法或 PulseAll 方法通知等待中的线程。

#### 19.4.4 使用 Mutex 类实现线程同步

当两个或更多线程需要同时访问一个共享资源时，系统需要使用同步机制来确保一次只有一个线程使用该资源。Mutex 类是同步基元，它只向一个线程授予对共享资源的独占访问权。如果一个线程获取了互斥体，则要获取该互斥体的第二个线程将被挂起，直到第一个线程释放该互斥体。Mutex 类与监视器类似，它防止多个线程在同一时间同时执行某个代码块，然而与监视器不同的是，Mutex 类可以用来使跨进程的线程同步。

可以使用 WaitHandle.WaitOne 方法请求互斥体的所属权，拥有互斥体的线程可以在对 WaitOne 方法的重复调用中请求相同的互斥体而不会阻止其执行，但线程必须调用同样多次数的 ReleaseMutex 方法以释放互斥体的所属权。Mutex 类强制线程标识，因此互斥体只能由获得它的线程释放。

当用于进程间同步时，Mutex 称为“命名 Mutex”，因为它将用于另一个应用程序，因此它不能通过全局变量或静态变量共享，而必须为它指定一个名称，才能使两个应用程序访问同一个 Mutex 对象。

Mutex 类的常用方法及说明如表 19.5 所示。

表 19.5 Mutex 类的常用方法及说明

方 法	说 明
Close	在派生类中被重写时，释放由当前 WaitHandle 持有的所有资源
OpenExisting	打开现有的已命名互斥体
ReleaseMutex	释放 Mutex 一次
SignalAndWait	原子操作的形式，向一个 WaitHandle 发出信号并等待另一个
WaitAll	等待指定数组中的所有元素都收到信号
WaitAny	等待指定数组中的任一元素收到信号
WaitOne	当在派生类中重写时，阻止当前线程，直到当前的 WaitHandle 收到信号

使用 Mutex 类实现线程同步很简单，首先创建一个 Mutex 对象，它的构造函数中比较常用的有 public Mutex(bool initiallyOwned)，其中，参数 initiallyOwned 指定了创建该对象的线程是否希望立即获得其所有权，当在一个资源得到保护的类中创建 Mutex 对象时，常将该参数设置为 false；然后在需要单线程访问的地方调用其等待方法，等待方法请求 Mutex 对象的所有权，这时，如果该所有权被另一个线程所拥有，则阻塞请求线程，并将其放入等待队列中，请求线程将保持阻塞，直到 Mutex 对象收到其所有者线程发出将其释放的信号为止。所有者线程在终止时释放 Mutex 对象，或者调用 ReleaseMutex 方法来释放 Mutex 对象。

 **说明：**尽管 Mutex 类可用于进程内的线程同步，但是使用 Monitor 类通常更为可取，因为 Monitor 监视器是专门为.NET Framework 而设计的，因而它可以更好地利用资源。相比之下，Mutex 类是 Win32 构造的包装。尽管 Mutex 类比监视器更为强大，但是相对于 Monitor 类，它所需要的互操作转换更消耗计算资源。

**例 19.11** 创建一个控制台应用程序，其中自定义了一个 LockThread 方法。在该方法中首先使用 Mutex 对象的 WaitOne 方法阻止当前线程，然后再调用 Mutex 对象的 ReleaseMutex 方法释放 Mutex 对象，即释放当前线程，最后在 Main 方法中通过 Program 的对象调用 LockThread 自定义方法。代码如下。（实例位置：光盘\mr\19\s\19.11）

```
static void Main(string[] args)
{
    Program myProgram = new Program(); //创建对象
    myProgram.LockThread(); //调用锁定线程方法
    Console.ReadLine();
}

void LockThread()
{
    Mutex myMutex=new Mutex(false); //创建 Mutex 对象
    myMutex.WaitOne(); //阻止当前线程
    Console.WriteLine("锁定线程以实现线程同步");
    myMutex.ReleaseMutex(); //释放 Mutex 对象
}
```

程序运行结果如图 19.10 所示。



图 19.10 运行结果

## 19.5 照猫画虎——基本功训练

### 19.5.1 基本功训练 1——判断线程的运行状态

视频讲解：光盘\mr\19\lx\判断线程的运行状态.exe

实例位置：光盘\mr\19\zmhh\01

在软件开发中常常需要获得线程的状态，用线程对象的 `IsAlive` 属性可以判断线程的状态。创建一个 Windows 应用程序，在窗体中主要添加如下代码。

```
private void Form1_Load(object sender, EventArgs e)           //窗体 Load 事件的处理方法
{
    Thread th = new System.Threading.Thread(Function);      //创建一个线程对象
    th.Start();                                              //启动线程，调用线程执行方法
    if (th.IsAlive)                                         //如果此线程已启动
    {
        label1.Text = "线程已启动";                          //输出线程状态
    }
    else
    {
        label1.Text = "线程未启动";                          //输出线程状态
    }
}
public void Function()                                     //线程执行方法
{
}
```

程序运行结果如图 19.11 所示。

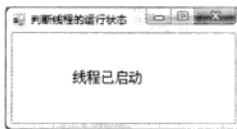


图 19.11 判断线程的运行状态

照猫画虎：修改上面的程序，终止线程，然后输出线程状态。提示：可调用 `Abort` 方法终止线程。（20 分）（实例位置：光盘\mr\19\zmhh\01\_zmhh）

### 19.5.2 基本功训练 2——使用线程遍历文件夹

视频讲解：光盘\mr\19\lx\使用线程遍历文件夹.exe

实例位置：光盘\mr\19\zmhh\02

创建一个 Windows 应用程序，在窗体上主要添加一个 `TextBox` 控件、一个 `Button` 控件、一个 `TreeView`

控件和一个 FolderBrowserDialog 控件，分别用来显示指定文件夹、打开“浏览文件夹”对话框、显示指定文件夹下的目录和选择指定文件夹，主要代码如下。

```

public delegate void AddFile();
public void SetAddFile()
{
    this.Invoke(new AddFile(RunAddFile));
}
public void RunAddFile()
{
    TreeNode TNode = new TreeNode();
    Files_Copy(treeView1, tempstr, TNode, 0);
    Thread.Sleep(0);
    thdAddFile.Abort();
}
private void button1_Click(object sender, EventArgs e)
{
    if (folderBrowserDialog1.ShowDialog() == DialogResult.OK)//确定选择指定文件夹
    {
        textBox1.Text = folderBrowserDialog1.SelectedPath; //显示文件夹路径
        tempstr = folderBrowserDialog1.SelectedPath; //获取文件夹路径
        thdAddFile = new Thread(new ThreadStart(SetAddFile)); //创建一个线程
        thdAddFile.Start(); //执行当前线程
    }
}

```

程序运行结果如图 19.12 所示。



图 19.12 使用线程遍历文件夹

**说明：** 使用线程遍历文件夹，能够防止因子文件夹或文件过多而造成计算机假死现象的发生。

**熊猫画虎：** 参照上面的程序，使用线程删除指定文件夹中的所有文件。提示：删除文件可以使用 FileInfo 类的 Delete 方法。（20 分）（实例位置：光盘\mr\19\zmhh\02\_zmhh）

### 19.5.3 基本功训练 3——使用线程休眠控制图片以百叶窗效果显示

**视频讲解：** 光盘\mr\19\lx\使用线程休眠控制图片以百叶窗效果显示.exe

**实例位置：** 光盘\mr\19\zmhh\03

创建一个 Windows 应用程序，在窗体上添加一个 OpenFileDialog 控件、两个 Button 控件，分别用来选

择图片文件、打开“打开文件”对话框和实现以百叶窗效果显示图片的操作，主要代码如下。

```

private void button2_Click(object sender, EventArgs e)
{
    try
    {
        Bitmap myBitmap = (Bitmap)this.BackgroundImage.Clone(); //以窗体背景图像创建位图对象
        int intWidth = myBitmap.Width; //获取位图的宽度
        int intHeight = myBitmap.Height / 20; //获取位图的高度
        Graphics myGraphics = this.CreateGraphics(); //创建画布对象
        myGraphics.Clear(Color.WhiteSmoke); //清除整个画布并以指定背景色填充
        Point[] myPoint = new Point[30]; //创建一个 Point 类型的数组
        for (int i = 0; i < 30; i++) //遍历 myPoint 数组中的元素
        {
            myPoint[i].X = 0; //设置点的横坐标
            myPoint[i].Y = i * intHeight; //设置点的纵坐标
        }
        Bitmap bitmap = new Bitmap(myBitmap.Width, myBitmap.Height); //创建一个新的位图对象
        for (int m = 0; m < intHeight; m++) //按照 myBitmap 位图的高度（即像素点个数）进行循环
        {
            for (int n = 0; n < 20; n++) //循环 20 次
            {
                for (int j = 0; j < intWidth; j++) //按照 myBitmap 位图的宽度（即像素点个数）进行循环
                {
                    bitmap.SetPixel(myPoint[n].X + j, myPoint[n].Y + m, myBitmap.GetPixel(myPoint[n].X + j, myPoint[n].Y + m)); //设置此 bitmap 位图中指定像素的颜色
                }
            }
            this.Refresh(); //窗体刷新
            this.BackgroundImage = bitmap; //设置窗体背景图像
            System.Threading.Thread.Sleep(100); //线程休眠 100 毫秒
        }
    }
    catch(Exception ex)
    {
        MessageBox.Show(ex.Message); //显示异常信息对话框
    }
}

```

程序运行结果如图 19.13 所示。

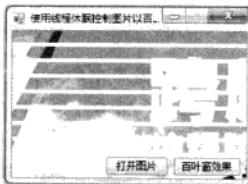


图 19.13 使用线程休眠控制图片以百叶窗效果显示

**照猫画虎：**使用线程休眠控制以不同分辨率显示图像。提示：设置图像分辨率，可使用 **Bitmap** 类的 **SetResolution** 方法。(20 分)(实例位置：光盘\mr\19\zmhh\03\_zmhh)

#### 19.5.4 基本功训练 4——使用线程读取数据库中的数据

 视频讲解：光盘\mr\19\lx\使用线程读取数据库中的数据.exe

 实例位置：光盘\mr\19\zmhh\04

创建一个 Windows 应用程序，在窗体上添加一个 DataGridView 控件和一个 Button 控件，分别用来显示数据记录和实现读取数据的操作，主要代码如下。

```
Thread myThread;
public delegate void AddFile();
public void SetAddFile()
{
    this.Invoke(new AddFile(RunAddFile)); //对指定的线程进行托管
}
public void RunAddFile() //设置线程
{
    AccessDataComponent(Application.StartupPath + "\\db_21.mdb", "tb_EmployeeInfo");
    //DataGridView 控件绑定数据源
    Thread.Sleep(0); //挂起主线程
    myThread.Abort(); //执行线程
}
private void button1_Click(object sender, EventArgs e) //单击“读取”按钮
{
    myThread = new Thread(new ThreadStart(SetAddFile)); //创建线程对象
    myThread.Start(); //开始运行扫描 IP 的线程
}
```

程序运行结果如图 19.14 所示。



图 19.14 使用线程读取数据库中的数据

**技巧:** 在获取数据库的大量数据时,由于数据量过大,会出现计算机的假死现象,这时可以通过创建一个子线程,然后用子线程来获取数据库中的数据并进行显示,这样就可以在获取数据库中数据的同时进行其他的操作。

照猫画虎：参照上面的实例，使用线程删除数据库中的记录。提示：执行删除命令的 SQL 语句，可使用 OleDbCommand 类的 ExecuteNonQuery 方法。（20 分）（实例位置：光盘\mr\19\zmhh\04\_zmhh）

### 19.5.5 基本功训练 5——使用线程实现大容量数据的计算

 视频讲解：光盘\mr\19\lx\使用线程实现大容量数据的计算.exe

 实例位置：光盘\mr\19\zmhh\05

在计算机中执行多个计算时，如果某些计算过程很长，则计算机可能会产生假死现象。这时，可以开

启动一个子线程去执行较长的计算过程，而主线程则继续执行后面的计算。创建一个 Windows 应用程序，在窗体上主要添加 3 个 TextBox 控件和一个 Button 控件，分别用来显示 3 个计算结果和实现计算数据的操作。

```

Thread myThread; //声明线程引用
public delegate void AddFile(); //定义托管线程
public void SetAddFile() //设置托管线程
{
    this.Invoke(new AddFile(RunAddFile)); //对指定的线程进行托管
}
public void RunAddFile() //设置线程
{
    textBox2.Text = Power(7, 50).ToString(); //计算 7 的 50 次幂，Power 方法用于计算 n 的 m 次幂
    Thread.Sleep(0); //挂起主线程
    myThread.Abort(); //中止线程
}
private void button1_Click(object sender, EventArgs e) //单击“计算”按钮
{
    textBox1.Text = Power(2, 4).ToString(); //计算 2 的 4 次幂
    myThread = new Thread(new ThreadStart(SetAddFile)); //创建线程对象，绑定线程方法 SetAddFile
    myThread.Start(); //开始运行扫描 IP 的线程
    textBox3.Text = Power(2, 2).ToString(); //计算 2 的 2 次幂
}

```

程序运行结果如图 19.15 和图 19.16 所示。

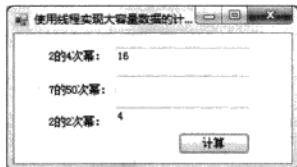


图 19.15 当计算量过多时计算后面的

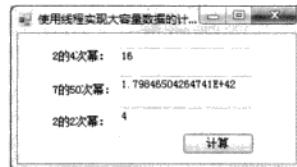


图 19.16 使用子线程完成数据的计算

**照猫画虎：**参照上面的例子，使用线程实现从 1 加到 10000000。提示：使用线程对象绑定实现大容量加法计算的方法，然后启动该线程。（20 分）（实例位置：光盘\mr\19\zmhh\05\_zmhh）

照猫画虎栏目分数统计：

照猫画虎题目	1	2	3	4	5	总分数	
分数							

## 19.6 情景应用——拓展与实践

### 19.6.1 情景应用 1——使用线程扫描局域网 IP 地址

视频讲解：光盘\mr\19\lx\使用线程扫描局域网 IP 地址.exe

实例位置：光盘\mr\19\qjyy\01

在局域网中扫描 IP 地址，为了使计算机不出现假死现象，可以利用线程来完成 IP 的扫描。首先应用

IPAddress 类将 IP 地址转换成网际协议的 IP 地址，然后使用 IPHostEntry 对象加载 IP 地址来获取其对应的主机名，如果有主机名，则表示当前 IP 已被使用，并将该 IP 地址显示在列表中。该过程可以通过执行子线程来完成，实例运行效果如图 19.17 所示。

实现过程如下。

(1) 新建一个 Windows 应用程序，在窗体上主要添加两个 TextBox 控件，分别用来输入开始地址和结束地址；添加一个 Button 控件，用来执行扫描局域网 IP 操作；添加一个 ListView 控件，用来显示搜索到的 IP 地址；添加一个 ProgressBar 控件，用来显示扫描进度；添加一个 Timer 控件，用来刷新 ListView 控件中的 IP 和 ProgressBar 控件的进度。

(2) 程序主要代码如下。

```

private Thread myThread;
int intStrat = 0;
int intEnd = 0;
private void button1_Click(object sender, EventArgs e)
{
    try
    {
        if (button1.Text == "开始")
        {
            listView1.Items.Clear();
            textBox1.Enabled = textBox2.Enabled = false;
            strIP = "";
            strflag = textBox1.Text;
            StartIPAddress = textBox1.Text;
            EndIPAddress = textBox2.Text;
            intStrat = Int32.Parse(StartIPAddress.Substring(StartIPAddress.LastIndexOf(".") + 1)); // 扫描的起始值
            intEnd = Int32.Parse(EndIPAddress.Substring(EndIPAddress.LastIndexOf(".") + 1)); // 扫描的终止值
            progressBar1.Minimum = intStrat;
            progressBar1.Maximum = intEnd;
            progressBar1.Value = progressBar1.Minimum;
            timer1.Start();
            button1.Text = "停止";
            myThread = new Thread(new ThreadStart(this.StartScan));
            myThread.Start();
        }
        else
        {
            textBox1.Enabled = textBox2.Enabled = true;
            button1.Text = "开始";
            timer1.Stop();
            progressBar1.Value = intEnd;
            if (myThread != null)
            {
                if (myThread.ThreadState == ThreadState.Running)
                {
                    // 声明线程引用
                    // 定义存储扫描起始值的变量
                    // 定义存储扫描终止值的变量
                    // 单击“开始”按钮
                    // 若还未开始搜索
                    // 清空 ListView 控件中的项
                    // 字符串变量赋值为空字符串
                    // 获得开始 IP 地址
                    // 获得终止 IP 地址
                    // 指定进度条最大值
                    // 指定进度条最小值
                    // 指定进度条初始值
                    // 开始运行计时器
                    // 设置按钮文本为停止
                    // 使用 StartScan 方法创建线程
                    // 开始运行扫描 IP 的线程
                    // 若已开始搜索
                    // 设置按钮文本为开始
                    // 停止运行计时器
                    // 设置进度条的值为最大值
                    // 判断线程对象是否为空
                    // 若扫描 IP 的线程正在运行
                }
            }
        }
    }
}

```



图 19.17 使用线程扫描局域网 IP 地址

```
    myThread.Abort(); //终止线程
}
}
}
}
catch {}
}
```

**DIY:** 使用线程批量复制与移动文件。提示：移动文件可调用 File.Move 方法，复制文件可调用 File.Copy 方法。(20 分)(实例位置: 光盘\mr\19\qjyy\01 diy)

## 19.6.2 情景应用 2——使用线程制作小游戏

 视频讲解：光盘\mr\19\lx\使用线程制作小游戏.exe

 实例位置：光盘\mr\19\qjyy\02

开心农场、QQ 农场可谓是近几年来最为广泛流行的虚拟现实小游戏，本实例实现了类似 QQ 农场的小游戏。运行程序后，单击窗体中的“播种”按钮，然后将种子拖到土地中种下，过了一夜时间后，农作物会显示出不同生长时期的图像。本实例运行效果如图 19.18 所示。

实现过程如下。

(1) 创建一个 Windows 窗体应用程序，然后在项目中添加一个自定义控件，并将其命名为 CPictureBox。该控件扩展自标准的 PictureBox 控件，用来显示农作物在不同时期的图像。然后在 Form1 窗体上主要添加两个 PictureBox 控件，分别加一个 Label 控件，用来显示仓库中果实的数量。

(2) 程序主要代码如下。

```

private CPictureBox cpbxSeed;
private void cpbxSeed_Click(object sender, EventArgs e)
{
    if (this.cpbxSeed != null && this.cpbxSeed.IsInseminate == false) //判断是否开始播种
    {
        this.cpbxSeed.IsInseminate = true; //表示种下——即确定位置
        Thread t = new Thread(GrowProcess); //创建线程实例，用来控制农作物生长过程
        t.IsBackground = true; //设置为后台线程
        t.Start(this.cpbxSeed); //启动线程
    }
}
private void GrowProcess(object obj) //该方法用来显示农作物在不同生长期的图像
{
    CPictureBox cpbx = obj as CPictureBox; //创建一个 CPictureBox 实例
    Thread.Sleep(5000); //生长耗时 5 秒钟
    cpbx.Image = ChuffedFarm.Properties.Resources.grow; //设置生长时的图片
    cpbx.PlantState = PlantState.Vegetate; //设置农作物处于生长状态
    Thread.Sleep(5000); //开花耗时 5 秒钟
    cpbx.Image = ChuffedFarm.Properties.Resources.bloom; //设置开花时的图片
    cpbx.PlantState = PlantState.BlossomOut; //设置农作物处于开花状态
}

```



图 19.18 使用线程制作小游戏

```

    Thread.Sleep(5000);
    cpbx.Image = ChuffedFarm.Properties.Resources.fruit;
    cpbx.PlantState = PlantState.MakeFruitage;
}

```

**说明：**上面代码中的 CPictureBox 控件是一个自定义控件，其用来显示农作物在不同时期的图像（包括种子图像、生长期图像、开花期图像、结果期图像），该控件的代码详见本书配套光盘中的源程序。

**DIY：**使用线程实现从左向右以拉伸的方式显示图像。提示：使用 Bitmap 类的 Clone 方法实现图像的内存拷贝。（20 分）（实例位置：光盘\mr\19\qjyy\02\_diy）

### 19.6.3 情景应用 3——有进度条的文件异步复制功能

**视频讲解：**光盘\mr\19\lx\有进度条的文件异步复制功能.exe

**实例位置：**光盘\mr\19\qjyy\03

在开发一些文档管理系统时，可能需要对文件进行复制、导入、导出的操作。如果文件太大，可能会在复制时长时间占用主线程，从而导致出现主线程假死现象。针对这种情况，可以使用线程及数据流对象自带的异步读取功能来完成大文件的复制操作。实例运行效果如图 19.19 所示。

实现过程如下。

(1) 新建一个“Windows 窗体控件库”应用程序，在当前项目名称上单击鼠标右键，在弹出的快捷菜单中选择“添加”/“Windows 窗体”命令，在打开的“添加新项”对话框中添加一个名称为 Frm\_Plan 的新窗体。

(2) 切换到主窗体的设计界面，在主窗体中添加两个 Label 控件，用于说明源文件和目的文件的路径，分别设置 Text 属性为“源文件”和“目的文件”；添加两个 TextBox 控件，用于显示源文件和目的文件的路径；添加 3 个 Button 控件，分别用于选择源文件和目的文件的路径以及按行复制操作。

(3) 程序主要代码如下。

```

//在窗体的“第一次显示事件”中执行线程，用于在线程中完成文件的复制操作
private void Frm_Plan_Shown(object sender, EventArgs e)
{
    thdAddFile = new Thread(new ThreadStart(SetAddFile));
    thdAddFile.Start();
}

public delegate void AddFile();
public void SetAddFile()
{
    this.Invoke(new AddFile(RunAddFile));
}

public void RunAddFile()
{
    CopyFile();
    thdAddFile.Abort();
}

public void CopyFile()

```

//创建一个线程  
 //执行当前线程  
  
 //定义托管线程  
 //完成线程和托管功能  
  
 //对指定的线程进行托管  
  
 //在线程中执行文件的复制操作  
  
 //复制文件  
 //执行线程  
  
 //通过源文件和目的文件的路径对文件进行异步读取

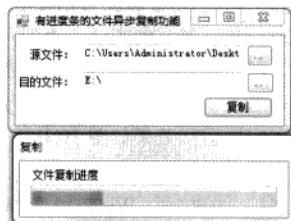


图 19.19 有进度条的文件异步复制功能

```

{
    if (filePath.Length == 0 || ToFile.Length == 0)           //如果没有设置源文件和目的文件的路径
        Close();                                            //关闭“复制”对话框
    fs = new FileStream(filePath, FileMode.Open, FileAccess.Read); //通过源文件路径以只读方式创建 FileStream 类
    totalSize = (int)fs.Length;                             //获取源文件的大小
    this.progressBar1.Maximum = totalSize;                  //设置进度条的最大值
    stream = fs;                                           //记录源文件的文件流
    FileInfo fileInfo = new FileInfo(filePath);            //创建 FileInfo 类
    ToFile += "\\\" + fileInfo.Name;                       //设置目的文件
    if (File.Exists(ToFile))                               //如果目的文件存在
        File.Delete(ToFile);                            //删除目的文件
    if (totalSize > BUFFER_SIZE)                         //如果源文件的大小大于流的传输大小
    {
        buffer = new byte[BUFFER_SIZE];                  //根据传输大小设置字节
        timer1.Start();                                //开启计时器
        //开始文件的异步读取
        stream.BeginRead(buffer, 0, BUFFER_SIZE, new AsyncCallback(AsyncCopyFile), null);
    }
    else                                                 //若是小文件
    {
        fileInfo.CopyTo(ToFile, true);                  //对文件直接进行复制
        position = totalSize;                          //设置进度条的进度
        fs.Close();                                    //关闭当前流
        timer1.Start();                                //启动计时器
    }
}
//异步读取完成后所调用的方法，该方法主要是通过文件流的传输大小对文件进行异步传输
private void AsyncCopyFile(IAsyncResult ar)
{
    int readedLength;                                //记录传输流的字节数
    lock (stream)                                     //在读取访问的同时防止其他进程更改
    {
        readedLength = stream.EndRead(ar);           //等待挂起的异步读取完成
    }
    //通过目的文件的路径以追加方式创建 FileStream 类
    FileStream fsWriter = new FileStream(ToFile, FileMode.Append, FileAccess.Write); //将源文件中读取的流写入到目的文件中
    fsWriter.Write(buffer, 0, buffer.Length);          //关闭写操作
    fsWriter.Close();                                //记录传递流的字节数
    if (position >= totalSize)                      //如果写入完成
    {
        stream.Close();                            //关闭当前流
        return;
    }
    lock (stream)                                     //在读取访问的同时防止其他进程更改
    {
        int leftSize = totalSize - position;         //获取没有复制的文件大小
        if (leftSize < BUFFER_SIZE)                  //如果小于流的传输大小
            buffer = new byte[leftSize];             //设置字节大小
        //开始文件的异步读取
    }
}

```

```

        stream.BeginRead(buffer, 0, buffer.Length, new AsyncCallback(AsyncCopyFile), null);
    }
}

```

**DIY：**异步加载并播放声音文件。提示：可调用 SoundPlayer 类的 LoadAsync 方法从流或 Web 资源中异步加载 WAV 文件。(20 分)(实例位置：光盘\mr\19\qjyy\03\_diy)

#### 19.6.4 情景应用 4——使用线程控制向窗体中拖放图片并显示

■ 视频讲解：光盘\mr\19\lx\使用线程控制向窗体中拖放图片并显示.exe

■ 实例位置：光盘\mr\19\qjyy\04

当用户显示多个不同目录中的图片时，为了减少图片打开或关闭操作，可以直接将图片拖放到窗体中进行显示。本实例通过使用线程控制向窗体中拖放图片并显示，实例运行效果如图 19.20 所示。

实现过程如下。

(1) 新建一个 Windows 应用程序，在 Form1 窗体中添加一个 treeView1 控件，设置 Dock 属性为 Fill，设置 PathSeparator 属性为“＼”，设置 AllowDrop 属性为 True。另外，设置 Form1 窗体的 AllowDrop 属性也为 True。

(2) 程序主要代码如下。

```

public void SetDragImageToFrm(TreeView TV, DragEventArgs e)//向 TreeView 控件添加被拖放的文件夹目录
{
    if (Var_Style == false)
    {
        e.Effect = DragDropEffects.Copy; //设置拖放操作中目标放置类型为复制
        String[] str_Drop = (String[])e.Data.GetData(DataFormats.FileDrop, true); //检索数据格式相关联的数据
        tempstr = str_Drop[0]; //获取拖放文件夹的目录
        thdAddFile = new Thread(new ThreadStart(SetAddFile)); //创建一个线程
        thdAddFile.Start(); //执行当前线程
    }
}

public delegate void AddFile(); //定义托管线程
public void SetAddFile() //实现线程的托管
{
    this.Invoke(new AddFile(RunAddFile)); //对指定的线程进行托管
}

public void RunAddFile() //将文件夹的遍历在线程中执行
{
    TreeNode TNode = new TreeNode(); //创建一个线程
    Files_Copy(treeView1, tempstr, TNode, 0); //遍历指定文件夹中所有子文件夹和文件
    Thread.Sleep(0); //休眠主线程
    thdAddFile.Abort(); //执行线程
}

```

**DIY：**使用线程获得系统当前的动态时间。提示：需要通过 Thread 类的 IsBackground 属性将当前线程设置为后台执行。(20 分)(实例位置：光盘\mr\19\qjyy\04\_diy)



图 19.20 使用线程控制向窗体中拖放图片并显示

### 19.6.5 情景应用 5——使用多线程制作端口扫描工具

视频讲解：光盘\mr\19\lx\使用多线程制作端口扫描工具.exe

实例位置：光盘\mr\19\qjyy\05

在扫描计算机端口时，为了防止程序出现假死现象，可使用多线程技术，也就是使用 Thread 类为端口扫描操作另外开启一个子线程。本实例实现使用多线程制作端口扫描工具的功能，实例运行效果如图 19.21 所示。

实现步骤如下。

(1) 新建一个 Windows 应用程序，在窗体 Form1 中主要添加一个 ComboBox 控件，用来选择工作组；添加一个 ListView 控件，用来显示已扫描到的端口号；添加两个 TextBox 控件，用来设置端口号的扫描范围；添加一个 Button 控件，用来执行局域网端口扫描操作。

(2) 程序主要代码如下。

```

private Thread myThread;
private void button1_Click(object sender, EventArgs e)
{
    listView1.Items.Clear();
    try
    {
        if (button1.Text == "扫描")
        {
            intport = 0;
            progressBar1.Minimum = Convert.ToInt32(textBox1.Text);
            progressBar1.Maximum = Convert.ToInt32(textBox2.Text);
            progressBar1.Value = progressBar1.Minimum;
            timer1.Start();
            button1.Text = "停止";
            intstart = Convert.ToInt32(textBox1.Text);
            intend = Convert.ToInt32(textBox2.Text);
            myThread = new Thread(new ThreadStart(this.StartScan));
            //使用自定义方法 StartScan 创建线程对象
            myThread.Start(); //开始运行扫描端口号的线程
        }
        else
        {
            button1.Text = "扫描";
            timer1.Stop();
            progressBar1.Value = Convert.ToInt32(textBox2.Text);
            if (myThread != null)
            {
                if (myThread.ThreadState == ThreadState.Running) //判断扫描端口号的线程是否正在运行
                {
                    myThread.Abort(); //终止线程
                }
            }
        }
    }
}
//声明线程引用
//单击“扫描”按钮
//清空 ListView 控件中的项
//若程序未进入扫描状态
//初始化已用端口号
//指定进度条最小值
//指定进度条最大值
//指定进度条初始值
//开始运行计时器
//设置按钮文本为停止
//为开始扫描的端口号赋值
//为结束扫描的端口号赋值
//若程序已进入扫描状态
//设置按钮文本为扫描
//停止运行计时器
//设置进度条的值为最大值
//判断线程对象是否为空
//判断扫描端口号的线程是否正在运行
//终止线程

```



图 19.21 使用多线程制作端口扫描工具

```

        }
    }
}
catch {}
}

```

**说明：**上面的代码在创建线程时用到了自定义方法 StartScan，该方法无参数且无返回值，主要用来根据用户输入的开始端口号和结束端口号实现端口扫描的功能。

**DIY：**使用线程制作动画效果的状态栏。提示：可使用线程的 Sleep 方法间隔性地更改 ToolStripStatusLabel 控件的 Image 属性。（20 分）（实例位置：光盘\mr\19\qjyy\05\_diy）

情景应用 DIY 栏目分数统计：

DIY 题目	1	2	3	4	5	总分数	
分数							

## 19.7 自我测试

### 一、选择题（每题 10 分，5 道题）

- 在 C# 中，通过调用 Thread 类的 Sleep(int x) 方法来实现禁止线程运行，其中 x 代表（ ）。  
 A. 禁止线程运行的微秒数      B. 禁止线程运行的毫秒数  
 C. 禁止线程运行的秒数      D. 禁止线程运行的 CPU 时间数
- 下面关于线程的描述中，错误的是（ ）。  
 A. 线程需要占用内存，线程越多，占用的内存也越多      B. 需要使用 CPU 时间跟踪线程  
 C. 线程对共享资源的访问会相互影响      D. 程序中使用的线程越多越好
- 在 C# 应用程序中，如果要终止一个线程，则应使用 Thread 类的（ ）。  
 A. Suspend 方法      B. Resume 方法      C. Sleep 方法      D. Abort 方法
- 下面关于 ThreadPriority 类型的枚举值中，表示线程优先级别最高的是（ ）。  
 A. Normal      B. Highest      C. AboveNormal      D. BelowNormal
- 下面哪个关键字或类型不能实现线程同步（ ）。  
 A. lock      B. Monitor      C. Mutex      D. TryEnter

### 二、填空题（每题 10 分，5 道题）

- 创建新的线程时需要使用 Thread 类，Thread 类具有接受一个（ ）或（ ）的构造函数。
- 线程具有生命周期，它包含 3 个状态，分别为（ ）、（ ）和（ ）。
- C# 中对线程进行操作时，主要用到了 Thread 类，通过使用 Thread 类，可以对线程进行（ ）、（ ）、（ ）、（ ）、（ ）等操作。
- 线程的挂起与恢复分别可以通过调用 Thread 类中的（ ）和（ ）来实现。
- lock 关键字可以用来确保代码块完成运行，而不会被其他线程中断，它是通过在代码块运行期间为给定对象获取（ ）来实现的。

## 测试分数统计：

类别	第 1 题	第 2 题	第 3 题	第 4 题	第 5 题
选择题分数					
填空题分数					
总分数					

## 19.8 行动指南

开始日期： 年 月 日

结束日期： 年 月 日

序号	内 容	行动指南	
1	照猫画虎栏目 分数( )	分数>75 分	优秀，基本功掌握得不错，加油！
		75 分>分数>50 分	及格，知识掌握得不牢，重新做一遍照猫画虎。
	情景应用栏目 分数( )	分数<50 分	请使用光盘中提供的“实战能力测试系统”进行提高训练。
		分数>75 分	优秀，综合应用能力很强。
	自我测试栏目 分数( )	75 分>分数>50 分	及格，综合应用能力需提高，再练一遍情景应用。
		分数<50 分	请使用光盘中提供的“实战能力测试系统”进行提高训练。
2	综合评价	请使用光盘中提供的“实战能力测试系统”进行提高训练。	
	编程能力培养：本栏目提供了一些实践题目，对于培养编程能力很有效，要做好。	反复训练后，以上各项分数都在 75 分以上，方可进入下一堂课学习。	
3	编程习惯培养：本堂课培养读者在学习开发中记笔记的习惯，把开发中遇到的问题、总结的经验记录下来。	仿照现今流行的下载软件，尝试开发一个使用多线程技术下载文件的小工具。提示：对每个下载任务都创建一个 Thread 实例，然后通过启动该线程实例来执行相应的下载方法。另外，还可以通过 Thread 类的 Priority 属性来设置当前线程的优先级。	
4	创新能力培养：看看身边有没有可以用编程解决的问题，记录到右边的表格中。根据学习进度，尝试编写解决这些问题的小程序。		

## 19.9 成功可以复制——IT “大王” 王志东

1967 年，王志东出生在东莞虎门镇的一个教师家庭，他的童年在半饥半饱的状态下度过，清苦的家况

成为他日后学习和创业的初始动力。在大学时王志东对计算机产生了浓厚的兴趣，大三时，在中关村，王志东给皮包公司做过推销，也做过汉化、二次开发、系统集成等技术工作，为以后的软件开发奠定了基础。1991 年 6 月，王志东独立完成了当时国内第一个实用化 Windows 3.0 中文环境——BDWin 3.0，成为北大方正 1991 年的七大成果之一。但是当时王志东所在的公司并没有推广这个项目，这使王志东很伤心，并且辞职。但是他没有放弃，于 1992 年 4 月成立了自己的公司“新天地电子信息技术研究所”。想要按照硅谷的方式进行发展，实现自己的硅谷梦。1992 年 5 月，王志东独立研制并推出全球第一套实用 Windows 3.1 中文平台“中文之星（Chinese Star 1.1）”。“中文之星”一经推出即在国内迅速普及，加速了我国的计算机应用进程，并创造了很好的社会效益和经济效益。但由于同合伙人意见不合，王志东黯然离开新天地，他的硅谷梦也化为泡影。王志东深感挫折，但是他的才能没有被埋没，1993 年 12 月，王志东出任新组建的“四通利方信息技术有限公司”总经理，时年 26 岁。主持开发的 RichWin 系列软件于 1995 年下半年推出，成为当时世界上水平最高的外挂式中文平台系统软件，迅速占领市场。1998 年 12 月，四通利方宣布成功并购位于“硅谷”的华渊生活资讯网，成立新浪网，王志东出任总裁，1 年后兼任 CEO。2001 年 6 月，由于意见分歧王志东被解除新浪网的首席执行官和董事职务。但是王志东平静地看待这一切，于当年年底，他创立了北京点击科技有限公司，重新走上创业之路。

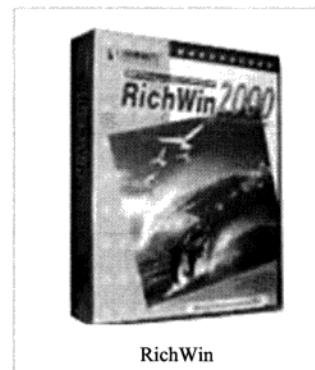
在王志东的亲自带领下，2002 年年底，点击科技推出了国内第一个协同应用平台——竞开协同应用平台（GENEKING），接着又推出了基于“竞开协同应用平台”的第一个产品——“竞开协同之星”，紧接着又推出一系列竞开系列软件。竞开（Geneking）系列协同软件被计世资讯、赛迪顾问等多家权威机构评为“中国协同软件市场领导型产品”和“中国协同软件市场第一品牌”，并被中国软件行业协会评为“优秀软件产品”。

### 经典语录

创业的过程就是实现梦想的过程，所以我是追着我自己的梦而走的。

### 深度评价

王志东的成功之路有过辉煌也有过低谷，他追逐着自己的梦想一路走来，创造了一段传奇人生。正如他所说“年轻人就应该有梦想有追求”，失败了没关系，因为“青春正如一张白纸”。



RichWin

# 第20堂课

## 异常处理与程序调试

( 视频讲解：30分钟)

开发程序时，不仅要注意程序代码的准确性与合理性，还要处理程序中可能出现的异常情况。.NET 框架提供了一套称为结构化异常处理的标准错误机制，在这种机制中，如果出现错误或任何预期之外的事件，都会引发异常。另外，在编写程序的过程中，不可避免地会出现错误，而有的错误不容易被发觉，从而导致程序运行错误，为了排除这些非常隐蔽的错误，对编写好的代码要进行程序调试，这样才能确保应用程序能够成功运行。本堂课详细介绍了如何在 C# 中进行异常处理与程序调试。

### 学习摘要：

- » 了解异常处理与程序调试的概念
- » 熟悉常用的公共异常类
- » 掌握 throw 语句的使用
- » 掌握 try…catch 语句的使用
- » 掌握 try…catch…finally 语句的使用
- » 掌握如何在代码中插入断点
- » 掌握如何启动、中断和停止调试
- » 掌握如何单步执行程序
- » 掌握如何设置程序运行到指定位置

## 20.1 异常处理与程序调试概述

在编写程序时，不仅要关心程序的正常操作，还应检查代码错误以及可能发生的各种不可预期的事件，在现代编程语言中，异常处理是解决这些问题的主要方法。异常处理是一种功能强大的机制，用于处理应用程序可能产生的错误或其他可以中断程序执行的异常情况。异常处理可以捕捉程序执行所发生的错误，通过它可以有效、快速地构建各种用来处理程序异常情况的程序代码。

在.NET 类库中提供了针对各种异常情形所设计的异常类，这些类包含了异常的相关信息。配合异常处理语句，应用程序能够轻易地避免程序执行时可能中断应用程序的各种错误。.NET 框架中的公共异常类如表 20.1 所示，这些异常类都是 System.Exception 的直接或间接子类。

表 20.1 公共异常类及说明

异常类	说 明
System.ArithemticException	在算术运算期间发生的异常
System.ArrayTypeMismatchException	当存储一个数组时，如果由于被存储的元素的实际类型与数组的实际类型不兼容而导致存储失败，就会引发此异常
System.DivideByZeroException	在试图用零除整数值时引发
System.IndexOutOfRangeException	在试图使用小于零或超出数组界限的下标索引数组时引发
System.InvalidCastException	当从基类型或接口到派生类型的显示转换在运行时失败，就会引发此异常
System.NullReferenceException	在需要使用引用对象的场合，如果使用 null 引用，就会引发此异常
System.OutOfMemoryException	在分配内存的尝试失败时引发
System.OverflowException	在选中的上下文中所进行的算术运算、类型转换或转换操作导致溢出时引发的异常
System.StackOverflowException	挂起的方法调用过多而导致执行堆栈溢出时引发的异常
System.TypeInitializationException	在静态构造函数引发异常并且没有可以捕捉到它的 catch 子句时引发

程序调试是在程序中查找错误的过程，在开发过程中，程序调试是检查代码并验证它能够正常运行的有效方法。另外，在开发程序时，如果发现程序不能正常工作，也可以通过程序调试找出有关问题。

 **说明：**在测试期间进行程序调试是很有用的，因为它对希望产生的代码结果提供了另外一级的验证。发布程序之后，程序调试提供了重新创建和检测程序错误的方法，同时可以帮助查找代码中的错误。

## 20.2 异常处理语句的使用

在 C# 程序中可以使用异常处理语句处理异常，常用的异常处理语句有 throw 语句、try…catch 语句和 try…catch…finally 语句，通过这 3 种异常处理语句，可以对可能产生异常的程序代码进行监控。下面将对这 3 种异常处理语句进行详细讲解。

### 20.2.1 使用 throw 语句抛出异常

throw 语句用于主动引发一个异常，使用该语句可以在特定的情形下自动抛出异常。throw 语句的基本格式如下。

```
throw ExObject
```

说明：ExObject 表示所要抛出的异常对象，这个异常对象是派生自 System.Exception 类的对象。

**例 20.01** 创建一个控制台应用程序，创建一个 int 类型的方法 MyInt，此方法有两个 string 类型的参数 a 和 b。在该方法中，使 a 做分子，b 做分母，如果分母的值是 0，则通过 throw 语句抛出 DivideByZeroException 异常。代码如下。（实例位置：光盘\mr\20\sl\20.01）

```

class test
{
    public int MyInt(string a, string b)          //创建一个 int 类型的方法，参数分别是 a 和 b
    {
        int int1;                                //声明一个 int 类型的变量 int1
        int int2;                                //声明一个 int 类型的变量 int2
        int num;                                 //声明一个 int 类型的变量 num
        int1 = int.Parse(a);                      //将参数 a 强制转换成 int 类型后赋给 int1
        int2 = int.Parse(b);                      //将参数 b 强制转换成 int 类型后赋给 int2
        if (int2 == 0)                            //判断 int2 是否等于 0，如果等于 0，抛出异常
        {
            throw new DivideByZeroException();   //抛出 DivideByZeroException 类的异常
            return 0;
        }
        else
        {
            num = int1 / int2;                  //计算 int1 除以 int2 的值
            return num;                        //返回计算结果
        }
    }
    static void Main(string[] args)
    {
        while (true)
        {
            Console.WriteLine("请输入分子：");      //提示输入分子
            string str1 = Console.ReadLine();       //获取键盘输入的值
            Console.WriteLine("请输入分母：");      //提示输入分母
            string str2 = Console.ReadLine();       //获取键盘输入的值
            test tt = new test();                  //创建 test 类
            //调用 test 类中的 MyInt 方法，获取键盘输入的分子与分母相除得到的值
            Console.WriteLine("分子除以分母的值：" + tt.MyInt(str1, str2));
        }
    }
}

```

运行以上程序，如果输入的分母为 0，则程序出错，错误信息如图 20.1 所示。

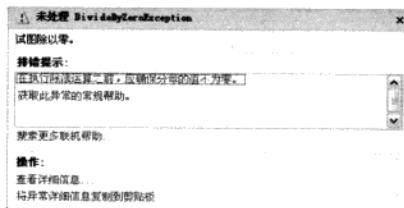


图 20.1 分母为 0 时的错误信息

## 20.2.2 使用 try...catch 语句捕捉异常

try...catch 语句允许在 try 后面的大括号 {} 中放置可能发生异常情况的程序代码，并对这些程序代码进行监控，而 catch 后面的大括号 {} 中则放置处理错误的程序代码，以处理程序发生的异常。try...catch 语句的基本格式如下。

```
try
{
    被监控的代码
}
catch(异常类名 异常变量名)
{
    异常处理
}
```

**说明：**在 catch 语句中，异常类名必须为 System.Exception 或从 System.Exception 派生的类型。当 catch 语句指定了异常类名和异常变量名后，就相当于声明了一个具有给定名称和类型的异常变量，此异常变量表示当前正在处理的异常。

**例 20.02** 创建一个控制台应用程序，声明一个 object 类型的变量 obj，其初始值为 null，然后将 obj 强制转换成 int 类型赋给 int 类型变量 N，使用 try...catch 语句捕获异常，代码如下。（实例位置：光盘\mr\20\sl\20.02）

```
static void Main(string[] args)
{
    try                                //使用 try...catch 语句
    {
        object obj = null;              //声明一个 object 变量，初始值为 null
        int N = (int)obj;               //将 object 类型强制转换成 int 类型
    }
    catch (Exception ex)               //捕获异常
    {
        Console.WriteLine("捕获异常：" + ex); //输出异常
    }
    Console.ReadLine();
}
```

程序运行结果如图 20.2 所示。

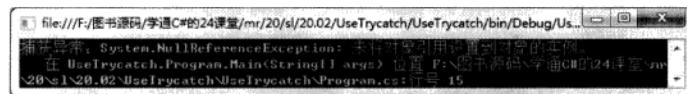


图 20.2 捕获异常

**说明：**① 上面的实例是直接使用 System.Exception 类捕获异常，使用其他异常类捕获异常的方法与其类似，这里不再赘述。

② 在 try...catch 语句中可以包含多个 catch 语句，但程序只执行第一个 catch 语句中的信息，其他的 catch 语句将被忽略。

### 20.2.3 使用 try…catch…finally 语句捕捉异常

将 finally 语句与 try…catch 语句结合，可以形成 try…catch…finally 语句。finally 语句同样以区块的方式存在，它被放在所有 try…catch 语句的最后面，程序执行完毕，最后都会跳到 finally 语句区块，执行其中的代码。其基本格式如下。

```
try
{
    被监控的代码
}
catch(异常类名 异常变量名)
{
    异常处理
}
...
finally
{
    程序代码
}
```

**说明：**无论程序是否产生异常，最后都会执行 finally 语句区块中的程序代码。

**技巧：**如果程序中有一些在任何情况下都必须执行的代码，那么可以将它们放在 finally 语句的区块中。

**例 20.03** 创建一个控制台应用程序，声明一个 string 类型变量 str，并初始化为“C#编程词典”；然后声明一个 object 变量 obj，将 str 赋给 obj；最后声明一个 int 类型的变量 i，将 obj 强制转换成 int 类型后赋给变量 i，这样必然会导致转换错误，抛出异常。在 finally 语句中输出“程序执行完毕...”，这样，无论程序是否抛出异常，都会执行 finally 语句中的代码，代码如下。（实例位置：光盘\mr\20\s\20.03）

```
static void Main(string[] args)
{
    string str = "C#编程词典";           //声明一个 string 类型的变量 str
    object obj = str;                     //声明一个 object 类型的变量 obj
    try                                //使用 try…catch 语句
    {
        int i = (int)obj;                //将 obj 强制转换成 int 类型
    }
    catch(Exception ex)                //获取异常
    {
        Console.WriteLine(ex.Message);   //输出异常信息
    }
    finally                            //finally 语句
    {
        Console.WriteLine("程序执行完毕..."); //输出“程序执行完毕...”
    }
    Console.ReadLine();
}
```

程序运行结果如图 20.3 所示。



图 20.3 finally 语句的使用

## 20.3 常用的程序调试操作

为了保证代码能够正常运行，需要对代码进行程序调试。常用的程序调试操作包括断点操作、开始、中断和停止程序的执行、单步执行程序以及使程序运行到指定的位置，下面将对这几种常用的程序调试操作进行详细介绍。

### 20.3.1 断点操作

断点是一个信号，它通知调试器在某个特定点上暂时将程序执行挂起。当执行在某个断点处挂起时，称程序处于中断模式。进入中断模式并不会终止或结束程序的执行，执行可以在任何时候继续。断点提供了一种强大的工具，能够在需要的时间和位置挂起执行，与逐句或逐条指令地检查代码不同的是，它可以让程序一直执行，直到遇到断点，然后开始调试，这大大加快了调试过程。下面介绍如何使用断点调试程序。

#### 1. 插入断点

插入断点的方法大体上可以分为 3 种，分别介绍如下。

- 在要设置断点的行旁边的灰色空白处单击，如图 20.4 所示。

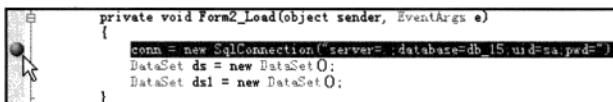


图 20.4 在代码行旁边的灰色空白处单击

- 选择某行代码，单击鼠标右键，在弹出的快捷菜单中选择“断点” / “插入断点”命令，如图 20.5 所示。



图 20.5 右键插入断点

- 选中要设置断点的代码行，选择菜单栏中的“调试” / “切换断点”命令，如图 20.6 所示。

#### 2. 删除断点

删除断点的方法大体上可以分为 4 种，分别介绍如下。

- 单击设置了断点的代码行左侧的红色圆点。
- 在设置了断点的代码行左侧的红色圆点上单击鼠标右键，选择“删除断点”选项。
- 在设置了断点的代码行上单击鼠标右键，在弹出的快捷菜单中选择“断点” / “删除断点”命令，

如图 20.7 所示。

- 选择设置了断点的代码行，然后选择菜单栏中的“调试” / “切换断点”命令。

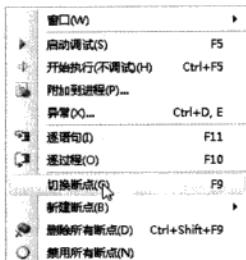


图 20.6 菜单栏插入断点

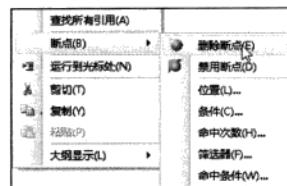


图 20.7 右键删除断点

### 20.3.2 开始、中断和停止程序的执行

当程序编写完毕后需要对程序代码进行调试，这时可以使用开始、中断和停止操作控制代码运行的状态，下面对这 3 种操作进行详细介绍。

#### 1. 开始执行

开始执行是最基本的调试功能之一，其操作为：从“调试”菜单（如图 20.8 所示）中选择“启动调试”命令或在源窗口中右击可执行代码中的某行，然后从弹出的快捷菜单中选择“运行到光标处”命令（如图 20.9 所示）。

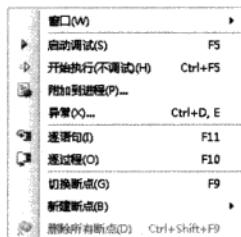


图 20.8 “调试”菜单

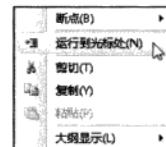


图 20.9 某行代码的右键菜单

除了使用上述方法开始执行外，还可以直接单击工具栏中的  按钮，启动调试，如图 20.10 所示。



图 20.10 工具栏中的启动调试按钮

如果选择“启动调试”命令，则应用程序将会启动并一直运行到断点。可以在任何时刻中断执行，以检查值、修改变量或检查程序状态，如图 20.11 所示。

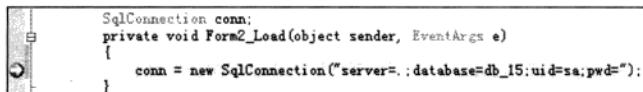


图 20.11 选择“启动调试”命令后的运行结果

如果选择“运行到光标处”命令，则应用程序将会启动并一直运行到断点或光标位置，具体要看是断点在前还是光标在前，可以在源窗口中设置光标位置。如果光标在断点的前面，则代码首先运行到光标处，如图 20.12 所示。

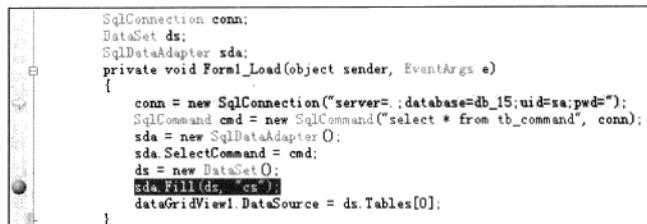


图 20.12 如果光标在断点前则只运行到光标处

## 2. 中断执行

当执行到达一个断点或发生异常时，调试器将中断程序的执行，但程序并不退出，可以随时恢复执行。但选择“调试”/“全部中断”命令后，调试器将停止所有在调试器下运行的程序的执行。“调试”菜单中的“全部中断”命令如图 20.13 所示。

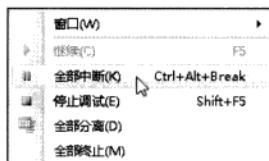


图 20.13 选择“调试”/“全部中断”命令

除了通过选择“调试”/“全部中断”命令中断执行外，也可以通过单击工具栏中的■按钮中断执行，如图 20.14 所示。



图 20.14 工具栏中的中断执行按钮

## 3. 停止执行

停止执行意味着终止正在调试的程序并结束调试会话，可以通过选择菜单栏中的“调试”/“停止调试”命令来结束运行和调试，也可以通过单击工具栏中的■按钮停止执行，如图 20.14 所示。

### 20.3.3 单步执行

通过单步执行，调试器每次只执行一行代码。单步执行主要是通过逐语句、逐过程和跳出这 3 种方法实现的。“逐语句”和“逐过程”的主要区别是当某一行包含函数调用时，“逐语句”仅执行调用本身，然后在函数内的第一个代码行处停止；而“逐过程”执行整个函数，然后在函数外的第一行处停止。如果代码位于函数调用的内部并想返回调用函数时，应使用“跳出”，“跳出”将一直执行代码，直到函数返回，然后在调用函数中的返回点处中断。

当启动调试后，可以单击工具栏中的■按钮执行“逐语句”操作，单击■按钮执行“逐过程”操作，

单击图 20.15 按钮执行“跳出”操作，如图 20.15 所示。

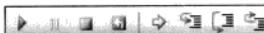


图 20.15 单步执行的 3 种命令

**◆ 注意：**除了在工具栏中单击这 3 个按钮外，还可以通过快捷键执行这 3 种操作，即在启动调试后，按 F11 键执行“逐语句”操作，按 F10 键执行“逐过程”操作，按 Shift+F10 键执行“跳出”操作。

#### 20.3.4 运行到指定位置

如果希望程序运行到指定的位置，可以通过在指定代码行上单击鼠标右键，在弹出的快捷菜单中选择“运行到光标处”命令，这样当程序运行到光标处时会自动暂停。另外，也可以在指定的位置插入断点，同样可以使程序运行到插入断点的代码行时自动暂停。

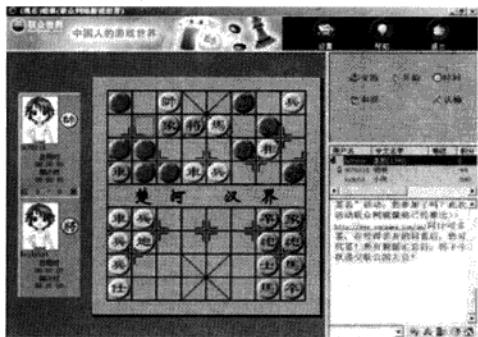
**◆ 说明：**关于如何插入断点和选择“运行到光标处”命令，可参见 20.3.1 节和 20.3.2 节的相关内容。

### 20.4 成功可以复制——IT 风云人物鲍岳桥

出生于 1967 年的鲍岳桥在杭州大学数学系读书时就开始非常迷恋计算机。那时学校里学生上机的机会很少，鲍岳桥想方设法和机房看门的教师搞好关系，终于弄到了一份机房管理员的差事，这样鲍岳桥基本上所有的时间都可以上机了，大四时，他几乎天天泡在机房，这一段时间的“恶补”为他将来做软件打下了基础。

1989 年，鲍岳桥大学毕业分到了杭州一个橡胶厂。就是在这个橡胶厂，鲍岳桥开发出了 FOXBASE 反编译软件、普通码中文输入系统和 PTDOS，开始在国内软件编程领域崭露头角。

1998 年初，鲍岳桥离开北京电脑公司，与简晶等人组建了北京联众电脑技术有限公司。北京西北郊外马连洼，鲍岳桥、简晶、王建华三人挤在只有两个房间的联众公司里，一边写程序，一边交流写程序的心得。当时只有王建华的 Windows 编程经验多一些，鲍岳桥和简晶的工作就是学习，边学习边工作。为了提高效率，简晶将家搬到了办公室附近，鲍岳桥和王建华一人弄一个木兰小摩托，跑来跑去，风尘仆仆。工作是从 1998 年的大年初二开始的，联众的框架设计用了将近两个月的时间，完全基于 NT 平台。鲍岳桥称，这个框架从一开始就考虑得很完善，之后的几次升级基本没有再做改动。接下来，王建华负责服务器端编程，鲍岳桥负责“游戏大厅”的开发，简晶负责具体游戏的设计。1998 年 6 月 4 日，联众游戏开通。刚开始没人知道这个网站，一个来玩的人都没有。于是这 3 个人就发挥自身力量四处找网友，拉他们过来看看。在种种努力下联众的人气越来越旺。联众成立 3 年，同时在线人数每年都以 9 倍的速度向前滚动：1999 年初 1000 人同时在线，2000 年初 9000 人同时在线，2001 年初达到了 8 万人。



联众游戏界面

注册用户方面，1999 年初是 3 万人同时在线，2000 年初是 70 万人，2001 年初是 700 万人。鲍岳桥称，棋牌类网站，联众全球第一。

 经典语录

我个人感觉我不是一个很聪明的人，但我是一个非常投入的人，我做一件事时整天头脑里就想着这一件事。我始终相信就算你笨一点，只要你投入的精力比别人多很多，就会比别人做得好。

 深度评价

不害怕吃苦、投入精力去做，这是鲍岳桥成功的原因，这也正是编程人员应该具备的品质。勤能补拙是良训，哪怕我们天资不如人，只要我们热爱编程，坚持去做，就会取得应有的硕果。

# 第21堂课

## Windows 应用程序打包部署

( 视频讲解：14分钟)

Windows 应用程序的开发是程序设计的重要组成部分，但在开发完成之后，就必然会面对程序的打包问题，如何将应用程序打包并制作成安装程序在客户机上部署成为每个 Windows 应用程序开发完成之后必须要解决的问题。本堂课通过使用 Visual Studio 2008 集成开发环境中的打包部署工具对 Windows 应用程序打包部署进行详细讲解。

学习摘要：

- ▶ 了解 Windows Installer
- ▶ 掌握如何创建 Windows 项目
- ▶ 掌握如何制作基本的 Windows 安装程序
- ▶ 掌握如何为 Windows 安装程序创建快捷方式
- ▶ 掌握如何为 Windows 安装程序添加注册表项
- ▶ 掌握如何为 Windows 安装程序添加程序组
- ▶ 掌握如何部署 Windows 应用程序
- ▶ 熟悉如何使用“发布”部署 Windows 应用程序

## 21.1 Windows Installer 介绍

Windows Installer 提供自定义安装的基础，它基于数据驱动模型，该模型在一个软件包中提供所有的安装数据和指令。相比而言，传统的脚本安装程序基于过程模型，为应用程序安装提供脚本指令。脚本安装程序强调“如何”安装，而 Windows Installer 则强调安装“什么”。

利用 Windows Installer 创建安装程序时，每台计算机都保留一个信息数据库，其中的信息与它所安装的应用程序有关，包括文件、注册表项和组件等内容。卸载应用程序时，系统将检查数据库以确保在卸载该应用程序前没有其他应用程序依赖于已经安装的文件、注册表项或组件等，这样可以防止在卸载一个应用程序后中断另一个应用程序。

Windows Installer 可以安装和管理公共语言运行库程序集，其开发人员可以将程序集安装到全局程序集缓存中，或者安装到为特定应用程序隔离的位置上。

Windows Installer 具有以下支持公共语言运行库程序集的功能。

- 安装、修复或移除全局程序集缓存中的程序集。
- 安装、修复或移除为特定应用程序指定的专用位置上的程序集。
- 回滚失败的程序集安装、修复或移除操作。
- 即需即装全局程序集缓存中具有强名称的程序集。
- 即需即装为特定应用程序指定的专用位置中的程序集。
- 修补程序集。
- 公布指向程序集的快捷方式。

Visual Studio 2008 中的部署工具建立在 Windows Installer 的基础之上，它可以为迅速部署和维护使用 Visual Studio 2008 生成的应用程序提供丰富的功能。

## 21.2 创建 Windows 安装项目

要对一个 Windows 应用程序进行打包部署，首先需要创建 Windows 安装项目。创建 Windows 安装项目的步骤如下。

(1) 在 Visual Studio 2008 集成开发环境中打开一个要部署的项目，在“解决方案”上单击鼠标右键，在弹出的快捷菜单中选择“添加”/“新建项目”命令，如图 21.1 所示。



图 21.1 在项目中创建部署项目

(2) 弹出“添加新项目”对话框，在“项目类型”列表框中选择“其他项目类型”/“安装和部署”节点，在右侧的“Visual Studio 已安装的模板”列表中选择“安装项目”，在“名称”文本框中输入安装项目名称，这里取其默认名 Setup1，在“位置”下拉列表框中选择存放安装项目文件的目标地址，如图 21.2 所示。

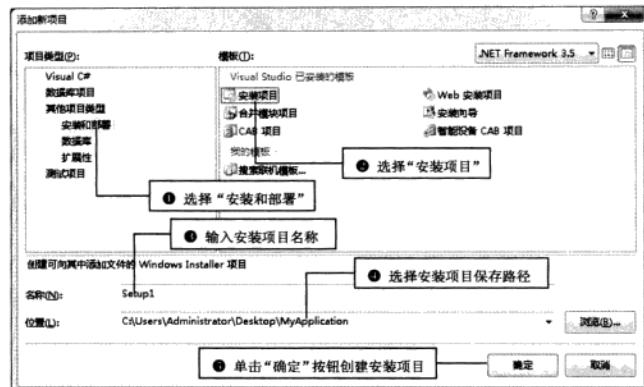


图 21.2 “添加新项目”对话框

(3) 单击“确定”按钮，即可创建一个 Windows 安装项目，如图 21.3 所示。



图 21.3 创建完成的 Windows 安装项目

## 21.3 制作 Windows 安装程序

创建完 Windows 安装项目之后，接下来讲解如何制作 Windows 安装程序。一个完整的 Windows 安装程序通常包括项目输出文件、内容文件、桌面快捷方式和注册表项等，下面讲解如何在创建 Windows 安装程序时添加这些内容。

### 21.3.1 添加项目输出

为 Windows 安装程序添加项目输出文件的步骤如下。

(1) 在“文件系统”的“目标计算机上的文件系统”节点下选中“应用程序文件夹”，单击鼠标右键，在弹出的快捷菜单中选择“添加”/“项目输出”命令，如图 21.4 所示。

(2) 弹出如图 21.5 所示的“添加项目输出组”对话框。在该对话框的“项目”下拉列表框中选择要部署的应用程序，然后选择要输出的类型，这里选择“主输出”，单击“确定”按钮，即可将项目输出文件添加到 Windows 安装程序中。



图 21.4 选择“添加”/“项目输出”命令



图 21.5 “添加项目输出组”对话框

### 21.3.2 添加内容文件

为 Windows 安装程序添加内容文件的步骤如下。

(1) 在 Visual Studio 2008 集成开发环境的中间位置单击鼠标右键，在弹出的快捷菜单中选择“添加”/“文件”命令，如图 21.6 所示。

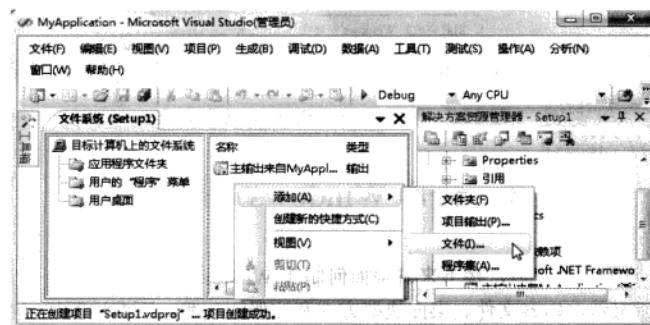


图 21.6 选择“添加”/“文件”命令

(2) 弹出如图 21.7 所示的“添加文件”对话框。在该对话框中选择要添加的内容文件，单击“打开”按钮，即可将选中的内容文件添加到 Windows 安装程序中。

(3) 添加完内容文件的 Windows 安装项目如图 21.8 所示。



图 21.7 “添加文件”对话框



图 21.8 添加完内容文件的 Windows 安装项目

### 21.3.3 创建桌面快捷方式

为 Windows 安装程序创建桌面快捷方式的步骤如下。

(1) 在 Visual Studio 2008 集成开发环境的中间位置选中“主输出来自 MyApplication（活动）”，单击鼠标右键，在弹出的快捷菜单中选择“创建 主输出来自 MyApplication（活动）的快捷方式”命令，如图 21.9 所示。

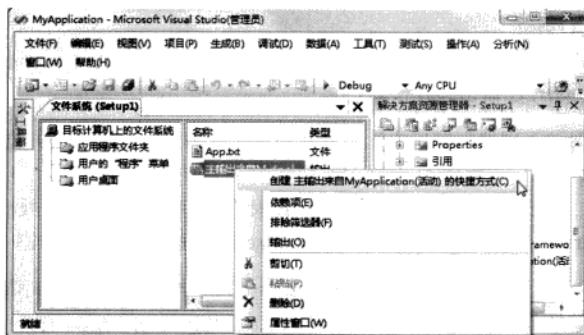


图 21.9 选择“创建 主输出来自 MyApplication（活动）的快捷方式”命令

(2) 此时添加了一个“主输出来自 MyApplication (活动) 的快捷方式”选项，将其重命名为“快捷方式”，如图 21.10 所示。

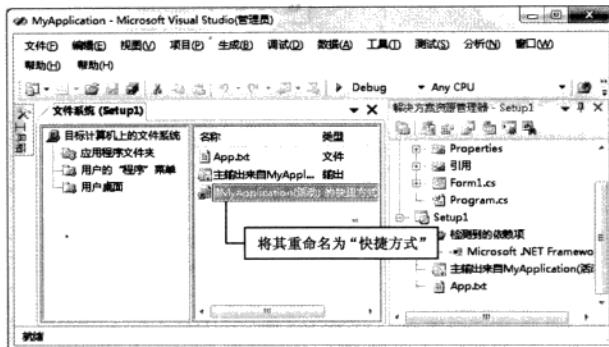


图 21.10 重命名快捷方式

(3) 选中创建的“快捷方式”，然后用鼠标将其拖放到左边“文件系统”下的“用户桌面”文件夹中，如图 21.11 所示，这样就为该 Windows 安装程序创建了一个桌面快捷方式。



图 21.11 将“快捷方式”拖放到“用户桌面”文件夹中

#### 21.3.4 添加注册表项

为 Windows 安装程序添加注册表项的步骤如下。

(1) 在“解决方案资源管理器”窗口中选中安装项目，单击鼠标右键，在弹出的快捷菜单中选择“视图” / “注册表”命令，如图 21.12 所示。

(2) 在 Windows 安装项目的左侧显示“注册表”选项卡，在该选项卡中依次展开 HKEY\_CURRENT\_USER/Software 节点，然后对注册表项[Manufacturer]进行重命名，如图 21.13 所示。

注意：[Manufacturer]注册表项用方括号括起来，表示它是一个属性，它将被替换为输入的部署项目的 Manufacturer 属性值。

(3) 选中注册表项，单击鼠标右键，在弹出的快捷菜单中选择“新建” / “字符串值”命令，如图 21.14

所示，这样即可为添加的注册表项初始化一个值。



图 21.12 选择“视图”/“注册表”命令

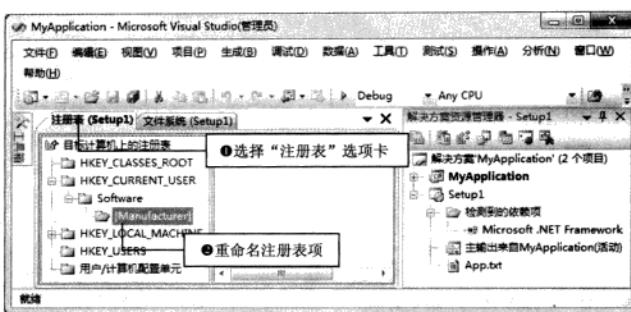


图 21.13 “注册表”选项卡

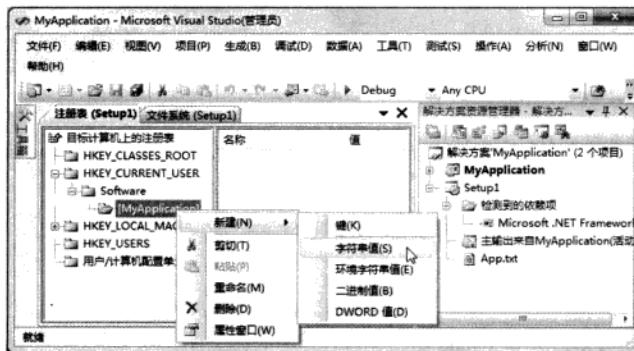


图 21.14 选择“新建”/“字符串值”命令

(4) 选中添加的注册表项值，单击鼠标右键，在弹出的快捷菜单中选择“属性窗口”命令，弹出“属性”窗口，如图 21.15 所示，这里可以对注册表项的值进行修改。

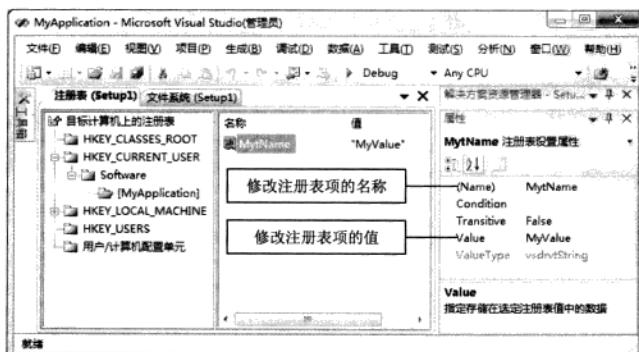


图 21.15 “属性”窗口

按照以上步骤，即可为 Windows 安装程序添加一个注册表项。

### 21.3.5 生成 Windows 安装程序

添加完 Windows 安装程序所需的项目输出文件、内容文件、桌面快捷方式和注册表项等内容后，在“解决方案资源管理器”窗口中选中 Windows 安装项目，单击鼠标右键，在弹出的快捷菜单中选择“生成”命令（见图 21.16），即可生成一个 Windows 安装程序，如图 21.17 所示。



图 21.16 选择“生成”命令

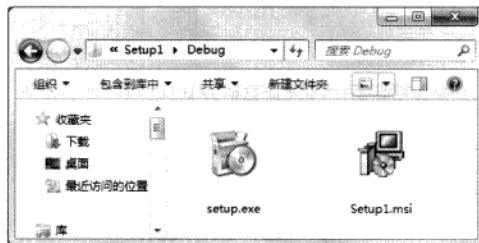


图 21.17 生成的 Windows 安装程序

## 21.4 部署 Windows 应用程序

制作完 Windows 安装程序之后，那么如何将 Windows 应用程序部署到客户机器上呢？本节紧接上面创建的 Windows 安装程序对如何部署 Windows 应用程序进行详细介绍。

- (1) 双击生成的 setup.exe 文件或 Setup1.msi 文件，弹出 Setup1 界面，如图 21.18 所示。
- (2) 单击“下一步”按钮，跳转到“Setup1-选择安装文件夹”界面，如图 21.19 所示，在该界面中可以单击“浏览”按钮选择要安装的路径。



图 21.18 Setup1 界面

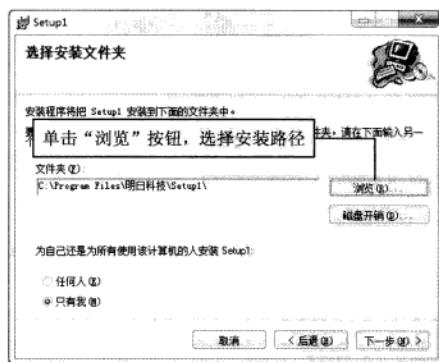


图 21.19 “Setup1-选择安装文件夹”界面

- (3) 设置完安装路径之后，单击“下一步”按钮，跳转到“Setup1-确认安装”界面，如图 21.20 所示。

(4) 单击“下一步”按钮，跳转到“Setup1-正在安装 Setup1”界面，如图 21.21 所示，在该界面中显示安装进度。



图 21.20 “Setup1-确认安装”界面



图 21.21 “Setup1-正在安装 Setup1”界面

(5) 安装完成后自动跳转到“Setup1-安装完成”界面，如图 21.22 所示，单击“关闭”按钮，完成 Windows 应用程序的安装。

(6) Windows 应用程序安装完成后，自动在桌面上创建一个快捷方式，如图 21.23 所示，双击该快捷方式即可运行程序。



图 21.22 “Setup1-安装完成”界面



图 21.23 Windows 应用程序的桌面快捷方式

## 21.5 成功可以复制——暴雪公司的领航者迈克·莫汉

暴雪公司 CEO 迈克·莫汉（Mike Morhaime）是一位相当具有传奇色彩的人物，他从只有 3 个办公桌的办公室起家，经过多年的打拼将暴雪公司发展成为一家全球知名的电脑游戏及电视游戏软件公司。他在暴雪公司的 20 年中，成功开发了魔兽争霸系列、星际争霸系列、黑破坏神系列以及《魔兽世界》游戏。魔兽争霸及星际争霸均被多项知名电子竞技比赛列为主要比赛项目。而单是《魔兽世界》游戏，每年就可以为暴雪公司带来高达将近 10 亿美元的收入。

迈克·莫汉在小学时就和哥哥妹妹们一起买了名为 Bally Professional Arcade 的游戏机，而他对于其上游戏的制作颇感兴趣。他在上学时就一直在研究如何让那些程序更好地运行。

作为美国加州大学电子工程系的学生，迈克·莫汉有更多的机会接触到游戏制作与开发，这也为他后

来创业奠定了基础。1991 年，大学毕业的迈克·莫汉和合作者共同创建了暴雪游戏开发公司。

一个大型游戏的开发，往往需要一年甚至更长的时间，投入与回报的周期比较长。初期，迈克·莫汉为了支付员工的薪水，经常需要从个人信用卡提取现金，依赖个人的借款来支撑公司。在那段时间里，他们的压力相当大，既要调动雇员的最大积极性发挥其才智，又要承受只出不入“耕耘期”的阵痛。当时，从资金流动的视角看，他们几乎一无所有。

1995 年对于迈克·莫汉来说是真正具有意义的一年，他们连续推出了《魔兽争霸 2》、《星际争霸》、《黑破坏神 2》等大作，受到了市场的追捧和热销。2007 年《魔兽世界：燃烧的远征》在发售后 24 小时内卖出了 240 万套，《魔兽世界》全球注册用户更是达到 1000 万。《魔兽世界》的成功使得迈克·莫汉成为享誉世界的游戏设计大师，也使他创办的暴雪公司一跃成为世界最顶级的游戏开发商，创造了游戏产业的一个神话。



《魔兽争霸》游戏画面

### 经典语录

优秀的游戏是我们的事业。

### 深度评价

借用 John Carmack 的话更容易了解迈克·莫汉的成功：游戏的程序就是一行行的代码，需要你的灵魂才能赋予它以生命，有了生命的游戏才能带给大家快乐，也才能给你所想要的，你的游戏是你的热情，是你的汗水，是你的喜怒哀乐，是你的梦想，你的游戏，其实就是你自己。我从未度过没有编程的一天，这就是我的全部。

# 第 4 部分

## 实战篇

- » 第 22 堂课 企业人事管理系统
- » 第 23 堂课 房屋中介管理系统
- » 第 24 堂课 进销存管理系统



# 第22堂课

## 企业人事管理系统

( 视频讲解：150分钟)

人事管理是现代企业管理工作不可缺少的一部分，是推动企业走向科学化、规范化的必要条件。员工是企业生存的主要元素，员工的增减、变动将直接影响到企业的整体运作。企业员工越多、分工越细、联系越密，所要做的统计工作就越多，人事管理的难度就越大。随着企业的不断壮大，自动化的企业人事管理系统就显得非常有必要，本堂课将通过使用 C# 3.5+SQL Server 2005 技术开发一个企业人事管理系统。

### 学习摘要：

- ▶ 掌握如何用自定义方法对不同的数据表进行添加和修改操作
- ▶ 掌握如何用自定义方法快速实现多条件的查询
- ▶ 掌握如何在数据库中读取或写入图片
- ▶ 掌握如何将数据信息以自定义表格的形式插入到 Word 中
- ▶ 掌握如何向 Word 中插入数据库中的图片

## 22.1 系统分析

### 22.1.1 需求分析

基于其他企业人事管理软件的不足，要求能够制作一个企业人事管理系统，可以方便、快捷地对职工信息进行添加、修改及删除操作，并且可以在数据库中存储相应职工的照片。为了能够更好地存储职工信息，可以将职工信息添加到 Word 文档中，这样不但便于保存，还可以通过 Word 文档进行打印。

### 22.1.2 可行性分析

根据《GB8567—88 计算机软件产品开发文件编制指南》中可行性分析的要求，制定可行性研究报告如下。

#### 1. 引言

##### (1) 编写目的

为了给软件开发企业的决策层提供是否进行项目实施的参考依据，现以文件的形式分析项目的风险、项目需要的投资与效益。

##### (2) 背景

XXX 科技有限公司是一家以计算机硬件销售为主的企业，为了更好地对公司内部的人员进行管理，现需要委托其他公司开发一个人事管理相关软件，项目名称为“企业人事管理系统”。

#### 2. 可行性研究的前提

##### (1) 要求

- 可以真正地实现对企业人事的管理。
- 系统的功能要符合本企业的实际情况。
- 系统的功能操作要方便、易懂，不要有多余或复杂的操作。
- 可以方便地对人事信息进行输出打印。

##### (2) 目标

方便地对企业内部的人事档案及岗位调动等进行管理。

##### (3) 评价尺度

项目需要在两个月内交付用户使用，系统分析人员需要 3 天内到位，用户需要 5 天时间确认需求分析文档，去除其中可能出现的问题，如用户可能临时有事，占用 7 天时间确认需求分析。那么程序开发人员需要在 50 天的时间内进行系统设计、程序编码、系统测试、程序调试和系统打包部署工作，其间还包括了员工每周的休息时间。

#### 3. 投资及效益分析

##### (1) 支出

根据预算，公司计划投入 8 个人，为此需要支付 9 万元的工资及各种福利待遇；项目的安装、调试以及用户培训、员工出差等费用支出需要 2.5 万元；在项目后期维护阶段预计需要投入 3 万元的资金，累计项目投入需要 14.5 万元资金。

### (2) 收益

客户提供项目开发资金 30 万元，对于项目后期进行的改动采取协商的原则，根据改动规模额外提供资金。因此，从投资与收益的效益比上，公司大致可以获得 15.5 万元的利润。

另外，当项目完成后，会给公司提供资源储备，包括技术、经验的积累。

## 4. 结论

根据上面的分析，在技术上不会存在问题，因此项目延期的可能性很小；在效益上，公司投入 8 个人、两个月的时间获利 15.5 万元，比较可观。另外，公司还可以储备项目开发的经验和资源。因此，认为该项目可以开发。

### 22.1.3 编写项目计划书

根据《GB8567—88 计算机软件产品开发文件编制指南》中的项目开发计划要求，结合单位实际情况，设计项目计划书如下。

#### 1. 引言

##### (1) 编写目的

为了能使项目按照合理的顺序开展，并保证按时、高质量的完成，现拟订项目计划书，将项目开发生命周期中的任务范围、团队组织结构、团队成员的工作任务、团队内外沟通协作方式、开发进度、检查项目工作等内容描述出来，作为项目相关人员之间的共识、约定以及项目生命周期内的所有项目活动的行动基础。

##### (2) 背景

企业人事管理系统是本公司与 XXX 科技有限公司签定的待开发项目，项目性质为人事管理类型，可以方便企业管理者对企业内部的人事变更、调动等的管理，项目周期为两个月。项目背景规划如表 22.1 所示。

表 22.1 项目背景规划

项目名称	签订项目单位	项目负责人	参与开发部门
企业人事管理系统	甲方：XXX 科技有限公司	甲方：王经理	设计部门
	乙方：TM 科技有限公司	乙方：高经理	开发部门 测试部门

#### 2. 概述

##### (1) 项目目标

项目应当符合 SMART 原则，把项目要完成的工作用清晰的语言描述出来。企业人事管理系统的主要目标就是为企业的管理者提供一套能够方便地对企业内部人员的变更及调动等进行管理的软件。

##### (2) 应交付成果

项目开发完成后，交付的内容如下。

- 以光盘的形式提供企业人事管理系统的源程序、系统数据库文件、系统打包文件和系统使用说明书。
- 系统发布后，进行无偿维护和服务 6 个月，超过 6 个月后将进行系统有偿维护与服务。

##### (3) 项目开发环境

开发本项目所用的操作系统可以是 Windows 2000 Server、Windows XP、Windows Server 2003 或 Windows 7，开发工具为 Visual Studio 2008，数据库采用 SQL Server 2005。

#### (4) 项目验收方式与依据

项目验收分为内部验收和外部验收两种方式。项目开发完成后，首先进行内部验收，由测试人员根据用户需求和项目目标进行验收。项目在通过内部验收后，交给客户进行外部验收，验收的主要依据为需求规格说明书。

### 3. 项目团队组织

#### (1) 组织结构

本公司针对该项目组建了一个由公司副经理、项目经理、系统分析员、软件工程师、界面设计师和测试人员构成的开发团队，团队结构如图 22.1 所示。

#### (2) 人员分工

为了明确项目团队中每个人的任务分工，现制定人员分工表，如表 22.2 所示。

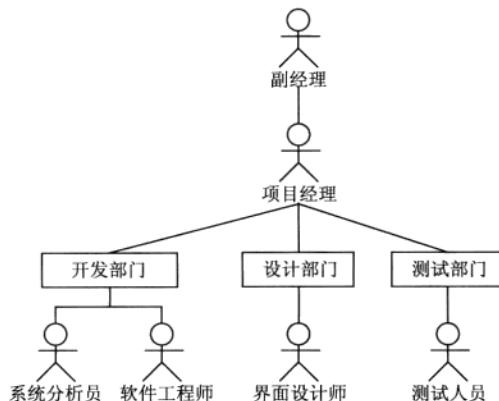


图 22.1 项目开发团队结构

表 22.2 人员分工表

姓 名	技术 水 平	所 属 部 门	角 色	工 作 描 述
秦某	MBA	经理部	副经理	负责项目的审批、决策的实施
汉某	MBA	项目开发部	项目经理	负责项目的前期分析、策划、项目开发进度的跟踪、项目质量的检查
魏某	中级系统分析员	项目开发部	系统分析员	负责系统功能分析、系统框架设计
唐某	中级软件工程师	项目开发部	软件工程师	负责软件设计与编码
宋某	中级软件工程师	项目开发部	软件工程师	负责软件设计与编码
元某	初级软件工程师	项目开发部	软件工程师	负责软件编码
明某	中级美工设计师	设计部	界面设计师	负责软件的界面设计
清某	中级系统测试工程师	测试部	测试人员	对软件进行测试以及编写软件测试文档

## 22.2 系统设计

### 22.2.1 系统目标

根据企业对人事管理的要求，制定企业人事管理系统目标如下。

- 操作简单方便、界面简洁美观。
- 在查看员工信息时，可以对当前员工的家庭情况、培训情况进行添加、修改、删除操作。
- 方便快捷的全方位数据查询。
- 按照指定的条件对员工进行统计。
- 可以将员工信息以表格的形式导出到 Word 文档中以便进行打印。
- 灵活的数据备份、还原及清空功能。
- 由于该系统的使用对象较多，要有较好的权限管理。

- 能够在当前运行的系统中重新进行登录。
- 系统运行稳定、安全可靠。

### 22.2.2 系统功能结构

企业人事管理系统的功能结构如图 22.2 所示。

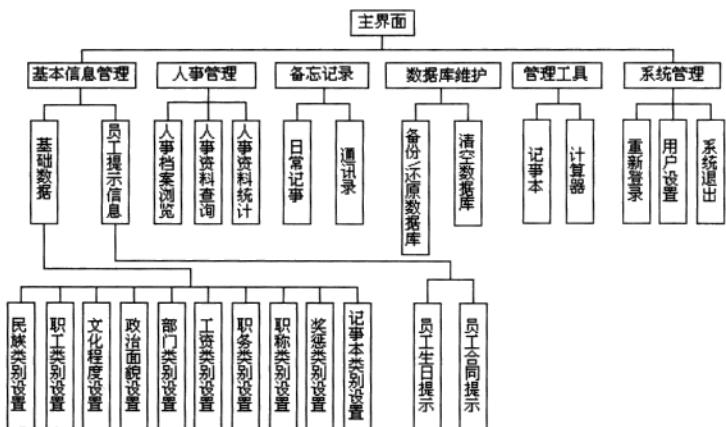


图 22.2 企业人事管理系统的功能结构图

### 22.2.3 系统业务流程图

企业人事管理系统的业务流程图如图 22.3 所示。

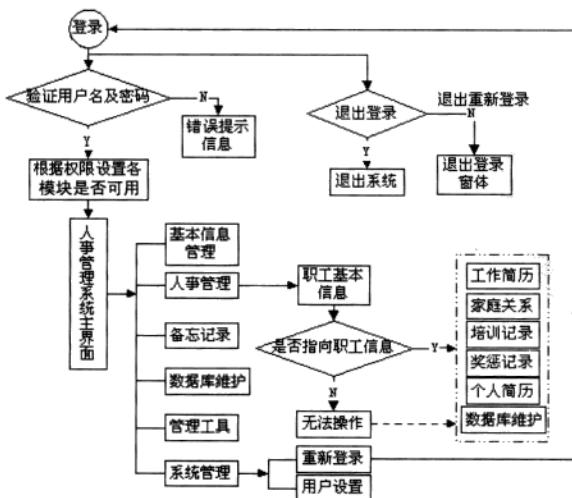


图 22.3 企业人事管理系统的业务流程图

## 22.2.4 系统编码规范

开发程序时，往往会有多人参与，为了使程序的结构与代码风格标准化，以便使每个参与开发的人员尽可能直观地查看和理解其他人编写的代码，需要在编码之前制定一套统一的编码规范，下面对本系统中比较特殊的编码规范进行说明。

### (1) 窗体命名规范

在创建一个窗体时，首先对窗体的 ID 进行命名，在本系统中统一命名为“F\_+窗体名称”，其中窗体名称最好是英文形式的窗体说明，便于开发者通过窗体 ID 就能知道该窗体的作用，如登录窗体的 ID 名为 F\_Login。

在窗体中调用其他窗体时，必须对调用窗体进行引用，其引用的变量名为“Frm+窗体名称”，如登录窗体的引用名为 FrmLogin。

### (2) 添加、修改操作中各控件的命名规范

在对数据进行编辑时，如果数据表中的字段过多，很难将窗体中对应的控件值组合成 SQL 语句，为了便于对数据库中的信息进行添加、修改操作，各字段所对应的控件命名为“表名（或部分表名）\_数字”，这里的数字是根据数据表中相应字段的顺序进行编号的。例如，将一个控件与 tb\_WordResume（工作简历表）数据表中的第 3 个字段建立关系，应将其 Name 属性设为 Word\_2。

### (3) 查询操作中各控件的命名规范

当使用多字段对数据表中的数据进行查询时，将窗体中相应的控件值组合成查询语句是非常麻烦的，为了能够快速组合查询条件，可以将设置查询条件的控件命名为“表名\_相应字段名”。当查询条件需要逻辑运算符时，将记录逻辑运算符的控件命名为“相应字段名\_Sign”。这样就可以通过字段名来组合查询条件。例如，查询年龄大于 30 岁的职工，年龄的字段名为 Age，条件控件名为 Find\_Age，逻辑控件名为 Age\_Sign，通过条件控件和逻辑控件便可以组合成查询条件。

## 22.3 系统运行环境

本系统的程序运行环境具体如下。

- 系统开发平台：Microsoft Visual Studio 2008。
- 系统开发语言：C#。
- 数据库管理软件：Microsoft SQL Server 2005。
- 运行平台：Windows XP (SP2) /Windows 2000 Server (SP4) /Windows Server 2003 (SP1) /Windows 7。
- 运行环境：Microsoft .NET Framework SDK v3.5。
- 分辨率：最佳效果 1024 像素×768 像素。

## 22.4 数据库与数据表设计

在开发项目时，对数据库的操作是必不可少的，数据库设计是根据程序的需求及其实现功能所制定的，其是否具有合理性将直接影响到程序的开发过程。

### 22.4.1 数据库分析

企业人事管理系统主要用来记录一个企业中所有员工的基本信息以及每个员工的工作简历、家庭成员、奖惩记录等，数据量是根据企业员工的多少来决定的。SQL Server 2005 数据库系统在安全性、准确性和运行速度方面有绝对的优势，并且处理数据量大、效率高，而且可与 SQL Server 2000 数据库无缝连接，所以本系统采用了 SQL Server 2005 数据库作为后台数据库，数据库命名为 db\_PWMS，其中包含了 23 张数据表，用于存储不同的信息，详细信息如图 22.4 所示。



图 22.4 企业人事管理系统中用到的数据表

### 22.4.2 创建数据库

在 SQL Server 2005 中创建数据库 db\_PWMS 的具体步骤如下。

(1) 选择“开始” / “所有程序” / Microsoft SQL Server 2005 CTP / SQL Server Management Studio 命令，如图 22.5 所示。



图 22.5 选择 SQL Server Management Studio 命令

**说明：**笔者机器安装的是 SQL Server 2005 简体中文版，所以在开始菜单中显示 Microsoft SQL Server 2005 CTP，如果用户安装的是 SQL Server 2005 专业版，则会在开始菜单中显示 Microsoft SQL Server 2005。

(2) 打开如图 22.6 所示的“连接到服务器”对话框。在该对话框中，选择登录的服务器名称和身份验

证方式，然后输入登录用户名和登录密码。

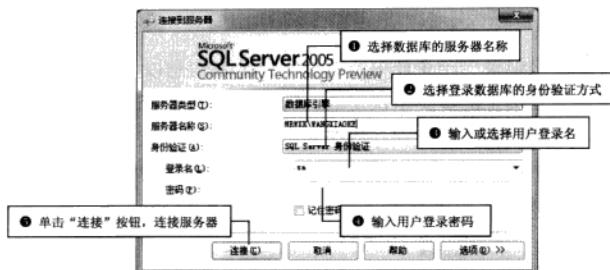


图 22.6 “连接到服务器”对话框

(3) 单击“连接”按钮，连接到指定的 SQL Server 2005 服务器，然后展开服务器节点，选中“数据库”节点，单击鼠标右键，在弹出的快捷菜单中选择“新建数据库”命令，如图 22.7 所示。



图 22.7 选择“新建数据库”命令

(4) 打开如图 22.8 所示的“新建数据库”对话框。在该对话框中输入新建的数据库的名称“db\_PWMS”，选择数据库所有者和存放路径，这里的数据库所有者一般为默认。

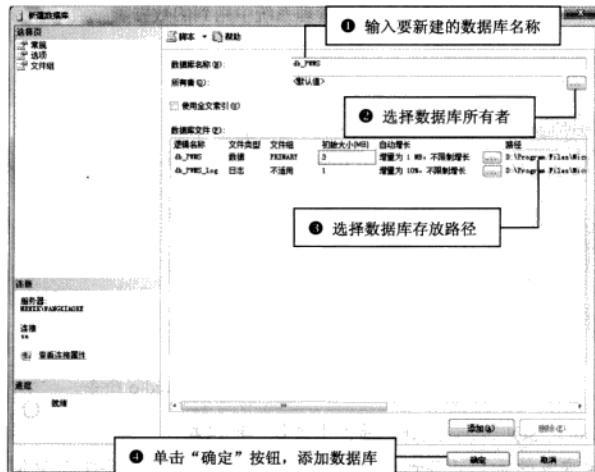


图 22.8 “新建数据库”对话框

(5) 单击“确定”按钮，即可新建一个 db\_PWMS 数据库，如图 22.9 所示。



图 22.9 新建的 db\_PWMS 数据库

### 22.4.3 创建数据表

在已经创建的数据库 db\_PWMS 中创建 23 个数据表，创建完成后的数据表及其记录数据如图 22.10 所示。



图 22.10 创建完成后的数据表及其记录数据

下面以 tb\_Login 表为例介绍创建数据表的过程。

(1) 展开新建的 db\_PWMS 数据库节点，选中“表”节点，单击鼠标右键，在弹出的快捷菜单中选择“新建表”命令，如图 22.11 所示。



图 22.11 选择“新建表”命令

(2) 在 SQL Server 2005 管理器的右边显示一个新表，这里输入要创建的表中所需要的字段，并设置主键，如图 22.12 所示。

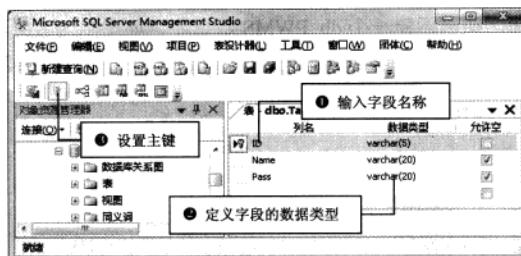


图 22.12 添加字段

(3) 单击“保存”按钮，弹出“选择名称”对话框，如图 22.13 所示。在该对话框中输入要新建的表名“tb\_Login”，单击“确定”按钮，即可在数据库中添加一个表。

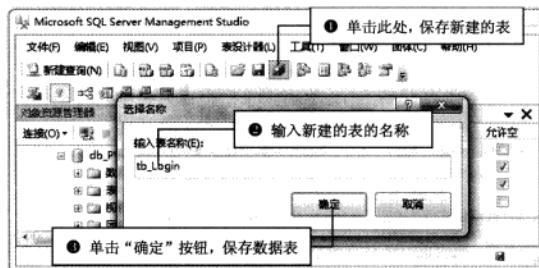


图 22.13 “选择名称”对话框

由于篇幅有限，其他数据表的创建过程就不介绍了，相信读者能够举一反三，结合下面给出的数据表结构，其他数据表的创建也不成问题。

tb\_UserPope (用户权限表)

tb\_UserPope 表用于保存每个操作员使用程序的相关权限，该表的结构如表 22.3 所示。

表 22.3 用户权限表

字 段 名	数 �据 类 型	主 键 否	描 述
AutoID	int	是	自动编号
ID	varchar(5)	否	操作员编号
PopeName	varchar(50)	否	权限名称
Pope	int	否	权限标识

tb\_Stuffbasic (职工基本信息表)

tb\_Stuffbasic 表用于保存职工的基本信息，该表的结构如表 22.4 所示。

表 22.4 职工基本信息表

字 段 名	数 据 类 型	主 键 否	描 述
ID	varchar(5)	是	职工编号
StuffName	varchar(20)	否	职工姓名
Folk	varchar(20)	否	民族
Birthday	datetime	否	出生日期

续表

字段名	数据类型	主键否	描述
Age	int	否	年龄
Kultur	varchar(14)	否	文化程度
Marriage	varchar(4)	否	婚姻
Sex	varchar(4)	否	性别
Visage	varchar(14)	否	政治面貌
IDCard	varchar(20)	否	身份证号
workdate	datetime	否	单位工作时间
WorkLength	int	否	工龄
Employee	varchar(20)	否	职工类型
Business	varchar(10)	否	职务类型
Laborage	varchar(10)	否	工资类别
Branch	varchar(14)	否	部门类别
Duthcall	varchar(14)	否	职称类别
Phone	varchar(14)	否	电话
Handset	varchar(11)	否	手机
School	varchar(24)	否	毕业学校
Speciality	varchar(20)	否	主修专业
GraduateDate	datetime	否	毕业时间
Address	varchar(50)	否	家庭地址
Photo	image	否	个人照片
BeAware	varchar(30)	否	省
City	varchar(30)	否	市
M_Pay	float	否	月工资
Bank	varchar(20)	否	银行账号
Pact_B	datetime	否	合同起始日期
Pact_E	datetime	否	合同结束日期
Pact_Y	float	否	合同期限

 tb\_Family (家庭关系表)

tb\_Family 表用于保存家庭关系的相关信息，该表的结构如表 22.5 所示。

表 22.5 家庭关系表

字段名	数据类型	主键否	描述
ID	varchar(5)	是	编号
Sut_ID	varchar(5)	否	职工编号
LeaguerName	varchar(20)	否	家庭成员名称
Nexus	varchar(10)	否	与本人的关系
BirthDate	datetime	否	出生日期
WordUnit	varchar(24)	否	工作单位
Business	varchar(10)	否	职务
Visage	varchar(10)	否	政治面貌

tb\_WordResume (工作简历表)

tb\_WordResume 表用于保存工作简历的相关信息，该表的结构如表 22.6 所示。

表 22.6 工作简历表

字段名	数据类型	主键否	描述
ID	varchar(5)	是	编号
Sut_ID	varchar(5)	否	职工编号
BeginDate	datetime	否	开始时间
EndDate	datetime	否	结束时间
WordUnit	varchar(24)	否	工作单位
Branch	varchar(14)	否	部门
Business	varchar(14)	否	职务

 tb\_RANDP (奖惩表)

tb\_RANDP 表用于保存职工奖惩记录的信息，该表的结构如表 22.7 所示。

表 22.7 奖惩表

字段名	数据类型	主键否	描述
ID	varchar(5)	是	编号
Sut_ID	varchar(5)	否	职工编号
RPKind	varchar(20)	否	奖惩种类
RPDate	datetime	否	奖惩时间
SealMan	varchar(10)	否	批准人
QuashDate	datetime	否	撤销时间
QuashWhys	varchar(50)	否	撤销原因

 tb\_TrainNote (培训记录表)

tb\_TrainNote 表用于保存职员培训记录的相关信息，该表的结构如表 22.8 所示。

表 22.8 培训记录表

字段名	数据类型	主键否	描述
ID	varchar(5)	是	编号
Sut_ID	varchar(5)	否	职工编号
TrainFashion	varchar(20)	否	培训方式
BeginDate	datetime	否	培训开始时间
EndDate	datetime	否	培训结束时间
Speciality	varchar(20)	否	培训专业
TrainUnit	varchar(30)	否	培训单位
KulturMemo	varchar(50)	否	培训内容
Charge	float	否	费用
Effect	varchar(20)	否	效果

 **说明：**由于篇幅有限，这里只列举了重要的数据表的结构，其他的数据表结构可参见本书配套光盘中的数据库文件。

### 22.4.4 数据表逻辑关系

为了使读者能够更好地了解职工基本信息表与其他各表之间的关系，在这里给出数据表关系图，如图 22.14 所示。通过图 22.14 可以看出，职工基本信息表的一些字段可以在相关联的表中获取指定的值，并通过职工基本信息表的 ID 值与家庭关系表、培训记录表、奖惩表等建立关系。

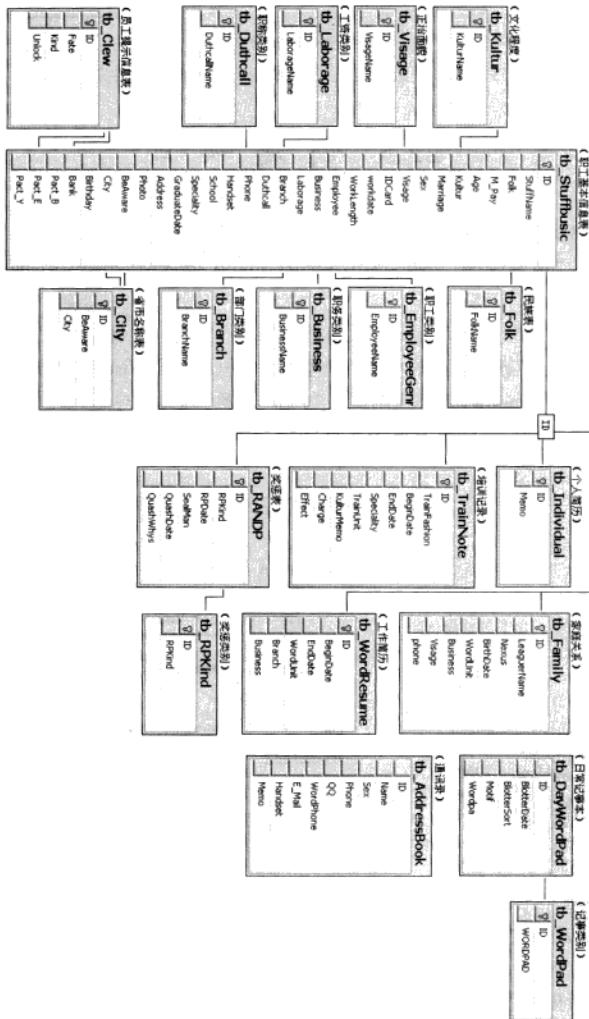


图 22.14 职工基本信息表与各表之间的关系

为了使读者能够更好地理解登录表与用户权限表、权限模块表之间的关系，下面给出其表间关系图，如图 22.15 所示。通过图 22.15 可以看出，在用户登录时，可以根据用户 ID 在用户权限表中调用相关的权

限。当添加用户时，可以通过权限模块表中的信息将权限名称自动添加到用户权限表中，以方便在前台中对用户进行添加操作。

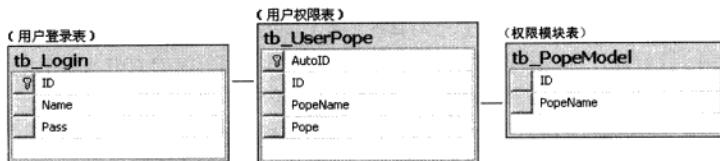


图 22.15 用户登录表与用户权限表、权限模块表之间的关系

## 22.5 创建项目

在 Visual Studio 2008 开发环境中创建 PWMS 项目的具体步骤如下。

(1) 选择“开始” / “所有程序” / Microsoft Visual Studio 2008/Microsoft Visual Studio 2008 命令，如图 22.16 所示。

(2) 打开 Visual Studio 2008 开发环境，在菜单栏中选择“文件” / “新建” / “项目”命令，打开如图 22.17 所示的“新建项目”对话框。在该对话框的“项目类型”列表框中选择 Visual C# 节点，在右侧的“Visual Studio 已安装的模板”列表中选择“Windows 窗体应用程序”，在“名称”文本框中输入项目名称，这里输入“PWMS”，在“位置”下拉列表框中选择存放项目文件的目标地址，单击“确定”按钮，即可创建一个空白的 PWMS 项目。



图 22.16 选择 Microsoft Visual Studio 2008 命令

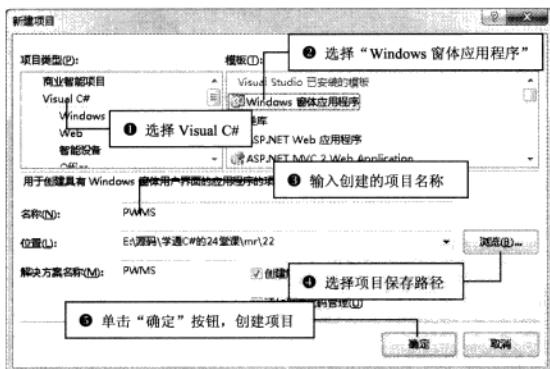


图 22.17 “新建项目”对话框

(3) 创建完 PWMS 项目之后，为了方便以后的开发工作和规范系统的整体架构，可以把系统中可能用到的文件夹先创建出来（例如，创建一个名为 Database 的文件夹，用于保存程序中用到的数据库文件），这样在开发时，只需将所创建的类文件或窗体文件保存到相应的文件夹中即可。在项目中创建文件夹非常简单，只需选中当前项目，单击鼠标右键，在弹出的快捷菜单中选择“添加” / “新建文件夹”命令即可，如图 22.18 所示。

(4) 按照以上步骤，依次创建企业人事管理系统中可能用到的文件夹并重命名。下面给出创建完成后的效果，如图 22.19 所示。

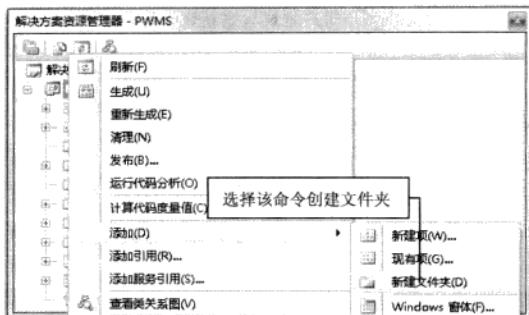


图 22.18 选择“添加”/“新建文件夹”命令



图 22.19 文件夹组织结构图

## 22.6 公共类设计

开发项目时，可以将数据库的相关操作以及对一些控件的设置、遍历等封装在自定义类中，以便在开发程序时调用，这样可以提高代码的重用性。本系统创建了 MyMeans 和 MyModule 两个公共类，分别存放在 DataClass 和 ModuleClass 文件夹中，下面对这两个公共类中比较重要的方法进行详细讲解。

### 22.6.1 MyMeans 公共类

MyMeans 公共类封装了本系统中所有与数据库连接的方法，可以通过该类的方法与数据库建立连接，并对数据信息进行添加、修改、删除及读取等操作。首先在命名空间区域引用 System.Data.SqlClient 命名空间，并定义全局变量及对象，主要代码如下。

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Data;
using System.Data.SqlClient;
using System.Windows.Forms;
namespace PWMS.DataClass
{
    class MyMeans
    {
        #region 全局变量
        public static string Login_ID = ""; // 定义全局变量，记录当前登录的用户编号
        public static string Login_Name = ""; // 定义全局变量，记录当前登录的用户名
        // 定义静态全局变量，记录“基础信息”各窗体中的表名、SQL 语句以及要添加和修改的字段名
        public static string Mean_SQL = "", Mean_Table = "", Mean_Field = "";
        // 定义一个 SqlConnection 类型的静态公共变量 My_con，用于判断数据库是否连接成功
        public static SqlConnection My_con;
        // 定义 SQL Server 2005 连接字符串，用户在使用时将 Data Source 改为自己的 SQL Server 2005 服务器名
        public static string M_str_sqlcon = "Data Source=mrwxk\wangxiao;Database=db_PWMS;User id=sa;PWD=";
```

```

public static int Login_n = 0; //用户登录与重新登录的标识
//存储职工基本信息表中的 SQL 语句
public static string AllSql = "Select * from tb_Stuffbasic";
#endregion
.....自定义方法，如 getcon()、con_close()、getcom()等方法
}
}

```

下面对 MyMeans 类中的自定义方法进行详细介绍。

### 1. getcon 方法

getcon 方法是用 static 定义的静态方法，其功能是建立与数据库的连接，然后通过 SqlConnection 对象的 Open 方法打开与数据库的连接，并返回 SqlConnection 对象的信息。getcon 方法的主要代码如下。

```

public static SqlConnection getcon()
{
    My_con = new SqlConnection(M_str_sqlcon); //用 SqlConnection 对象与指定的数据库相连接
    My_con.Open(); //打开数据库连接
    return My_con; //返回 SqlConnection 对象的信息
}

```

### 2. con\_close 方法

con\_close 方法的主要功能是对数据库操作后，通过该方法判断是否与数据库连接，如果连接，则关闭并释放数据库连接。con\_close 方法的主要代码如下。

```

public void con_close()
{
    if (My_con.State == ConnectionState.Open){ //判断是否打开与数据库的连接
        My_con.Close(); //关闭数据库的连接
        My_con.Dispose(); //释放 My_con 变量的所有空间
    }
}

```

### 3. getcom 方法

getcom 方法的主要功能是用 SqlDataReader 对象以只读的方式读取数据库中的信息，并以 SqlDataReader 对象进行返回，其中 SQLstr 参数表示传递的 SQL 语句。getcom 方法的主要代码如下。

```

public SqlDataReader getcom(string SQLstr)
{
    getcon(); //打开与数据库的连接
    //创建一个 SqlCommand 对象，用于执行 SQL 语句
    SqlCommand My_com = My_con.CreateCommand();
    My_com.CommandText = SQLstr; //获取指定的 SQL 语句
    SqlDataReader My_read = My_com.ExecuteReader(); //执行 SQL 语句，生成一个 SqlDataReader 对象
    return My_read;
}

```

### 4. getsqlcom 方法

getsqlcom 方法的主要功能是通过 SqlCommand 对象执行数据库中的添加、修改和删除操作，并在执行完后关闭与数据库的连接，其中 SQLstr 参数表示传递的 SQL 语句。getsqlcom 方法的主要代码如下。

```

public void getsqlcom(string SQLstr)
{
}

```

```

getcon(); //打开与数据库的连接
SqlCommand SQLcom = new SqlCommand(SQLstr, My_con);
//创建一个 SqlCommand 对象，用于执行 SQL 语句
SQLcom.ExecuteNonQuery(); //执行 SQL 语句
SQLcom.Dispose(); //释放所有空间
con_close(); //调用 con_close 方法，关闭数据库连接
}

```

### 5. getDataSet 方法

getDataSet 方法的主要功能是通过 SqlDataAdapter 对象执行数据库查询操作，并将查询结果填充到 DataSet 数据集中进行返回，其中 SQLstr 参数表示要查询的 SQL 语句，tableName 表示要填充 DataSet 数据集的数据表名。getDataSet 方法的主要代码如下。

```

public DataSet getDataSet(string SQLstr, string tableName)
{
    getcon(); //打开与数据库的连接
    SqlDataAdapter SQLda = new SqlDataAdapter(SQLstr, My_con); //实例化 SqlDataAdapter 对象
    DataSet My_DataSet = new DataSet(); //创建 DataSet 对象
    SQLda.Fill(My_DataSet, tableName); //填充 DataSet 数据集
    con_close(); //关闭数据库的连接
    return My_DataSet; //返回 DataSet 对象的信息
}

```

## 22.6.2 MyModule 公共类

MyModule 类将系统中所有窗体的动态调用以及动态生成添加、修改、删除和查询的 SQL 语句等全部封装到了指定的自定义方法中，以便在开发程序时进行重复调用，这样可以大大简化程序的开发过程。由于该类中应用了可视化组件的基类和对数据库进行操作的相关对象，所以在命名空间区域引用 System.Windows.Forms 和 System.Data.SqlClient 命名空间，并定义程序用到的全局变量及对象。主要代码如下。

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
//以下是添加的命名空间
using System.Windows.Forms;
using System.Data;
using System.Data.SqlClient;
namespace PWMS.ModuleClass
{
    class MyModule
    {
        #region 公共变量
        //声明 MyMeans 类的一个对象，以调用其方法
        DataClass.MyMeans MyDataClass = new PWMS.DataClass.MyMeans();
        public static string ADDs = ""; //用来存储添加或修改的 SQL 语句
        public static string FindValue = ""; //存储查询条件
        public static string Address_ID = ""; //存储通讯录添加修改时的 ID 编号
        public static string User_ID = ""; //存储用户的 ID 编号
        public static string User_Name = ""; //存储用户名
    }
}

```

```

    #endregion
    .....自定义方法，如 Show_Form()、TreeMenuF()、Part_SaveClass()等方法
}
}

```

因篇幅有限，下面只对几个比较重要的方法进行介绍。

### 1. Show\_Form 方法

Show\_Form 方法通过 FrmName 参数传递窗体名称以及调用相应的子窗体，由于本系统中存在公共窗体，也就是在同一个窗体模块中可以显示不同的窗体，所以用参数 n 来进行标识。调用公共窗体，实际上就是通过不同的 SQL 语句，在显示窗体时以不同的数据进行显示，以实现不同窗体的显示效果。Show\_Form 方法的主要代码如下。

```

public void Show_Form(string FrmName, int n)
{
    if (n == 1)
    {
        if (FrmName == "人事档案管理") //判断当前要打开的窗体
        {
            PerForm.F_ManFile FrmManFile = new PWMS.PerForm.F_ManFile();
            FrmManFile.Text = "人事档案管理"; //设置窗体名称
            FrmManFile.ShowDialog(); //显示窗体
            FrmManFile.Dispose();
        }
        if (FrmName == "人事资料查询")
        {
            PerForm.F_Find FrmFind = new PWMS.PerForm.F_Find();
            FrmFind.Text = "人事资料查询";
            FrmFind.ShowDialog();
            FrmFind.Dispose();
        }
        if (FrmName == "人事资料统计")
        {
            PerForm.F_Stat FrmStat = new PWMS.PerForm.F_Stat();
            FrmStat.Text = "人事资料统计";
            FrmStat.ShowDialog();
            FrmStat.Dispose();
        }
        if (FrmName == "员工生日提示")
        {
            InfoAddForm.F_ClewSet FrmClewSet = new PWMS.InfoAddForm.F_ClewSet();
            FrmClewSet.Text = "员工生日提示"; //设置窗体名称
            //设置窗体的 Tag 属性，用于在打开窗体时判断窗体的显示类型
            FrmClewSet.Tag = 1;
            FrmClewSet.ShowDialog(); //显示窗体
            FrmClewSet.Dispose();
        }
        if (FrmName == "员工合同提示")
        {
            InfoAddForm.F_ClewSet FrmClewSet = new PWMS.InfoAddForm.F_ClewSet();
            FrmClewSet.Text = "员工合同提示";
        }
    }
}

```

```

FrmClewSet.Tag = 2;
FrmClewSet.ShowDialog();
FrmClewSet.Dispose();
}
if (FrmName == "日常记事")
{
    PerForm.F_WordPad FrmWordPad = new PWMS.PerForm.F_WordPad();
    FrmWordPad.Text = "日常记事";
    FrmWordPad.ShowDialog();
    FrmWordPad.Dispose();
}
if (FrmName == "通讯录")
{
    PerForm.F_AddressList FrmAddressList = new PWMS.PerForm.F_AddressList();
    FrmAddressList.Text = "通讯录";
    FrmAddressList.ShowDialog();
    FrmAddressList.Dispose();
}
if (FrmName == "备份/还原数据库")
{
    PerForm.F_HaveBack FrmHaveBack = new PWMS.PerForm.F_HaveBack();
    FrmHaveBack.Text = "备份/还原数据库";
    FrmHaveBack.ShowDialog();
    FrmHaveBack.Dispose();
}
if (FrmName == "清空数据库")
{
    PerForm.F_ClearData FrmClearData = new PWMS.PerForm.F_ClearData();
    FrmClearData.Text = "清空数据库";
    FrmClearData.ShowDialog();
    FrmClearData.Dispose();
}
if (FrmName == "重新登录")
{
    F_Login FrmLogin = new F_Login();
    FrmLogin.Tag = 2;
    FrmLogin.ShowDialog();
    FrmLogin.Dispose();
}
if (FrmName == "用户设置")
{
    PerForm.F_User FrmUser = new PWMS.PerForm.F_User();
    FrmUser.Text = "用户设置";
    FrmUser.ShowDialog();
    FrmUser.Dispose();
}
if (FrmName == "计算器")
{
    System.Diagnostics.Process.Start("calc.exe");
}

```

```

if (FrmName == "记事本")
{
    System.Diagnostics.Process.Start("notepad.exe");
}
if (FrmName == "系统帮助")
{
    System.Diagnostics.Process.Start("readme.doc");
}
}
if (n == 2)
{
    String FrmStr = "";
    if (FrmName == "民族类别设置") //记录窗体名称
    {
        DataClass.MyMeans.Mean_SQL = "select * from tb_Folk"; //SQL 语句
        DataClass.MyMeans.Mean_Table = "tb_Folk"; //表名
        DataClass.MyMeans.Mean_Field = "FolkName"; //添加、修改数据的字段名
        FrmStr = FrmName;
    }
    if (FrmName == "职工类别设置")
    {
        DataClass.MyMeans.Mean_SQL = "select * from tb_EmployeeGenre";
        DataClass.MyMeans.Mean_Table = "tb_EmployeeGenre";
        DataClass.MyMeans.Mean_Field = "EmployeeName";
        FrmStr = FrmName;
    }
    if (FrmName == "文化程度设置")
    {
        DataClass.MyMeans.Mean_SQL = "select * from tb_Kultur";
        DataClass.MyMeans.Mean_Table = "tb_Kultur";
        DataClass.MyMeans.Mean_Field = "KulturName";
        FrmStr = FrmName;
    }
    if (FrmName == "政治面貌设置")
    {
        DataClass.MyMeans.Mean_SQL = "select * from tb_Visage";
        DataClass.MyMeans.Mean_Table = "tb_Visage";
        DataClass.MyMeans.Mean_Field = "VisageName";
        FrmStr = FrmName;
    }
    if (FrmName == "部门类别设置")
    {
        DataClass.MyMeans.Mean_SQL = "select * from tb_Branch";
        DataClass.MyMeans.Mean_Table = "tb_Branch";
        DataClass.MyMeans.Mean_Field = "BranchName";
        FrmStr = FrmName;
    }
    if (FrmName == "工资类别设置")
    {
        DataClass.MyMeans.Mean_SQL = "select * from tb_Laborage";
    }
}

```

```

DataClass.MyMeans.Mean_Table = "tb_Laborage";
DataClass.MyMeans.Mean_Field = "LaborageName";
FrmStr = FrmName;
}
if (FrmName == "职务类别设置")
{
    DataClass.MyMeans.Mean_SQL = "select * from tb_Business";
    DataClass.MyMeans.Mean_Table = "tb_Business";
    DataClass.MyMeans.Mean_Field = "BusinessName";
    FrmStr = FrmName;
}
if (FrmName == "职称类别设置")
{
    DataClass.MyMeans.Mean_SQL = "select * from tb_Duthcall";
    DataClass.MyMeans.Mean_Table = "tb_Duthcall";
    DataClass.MyMeans.Mean_Field = "DuthcallName";
    FrmStr = FrmName;
}
if (FrmName == "奖惩类别设置")
{
    DataClass.MyMeans.Mean_SQL = "select * from tb_RPKind";
    DataClass.MyMeans.Mean_Table = "tb_RPKind";
    DataClass.MyMeans.Mean_Field = "RPKind";
    FrmStr = FrmName;
}
if (FrmName == "记事本类别设置")
{
    DataClass.MyMeans.Mean_SQL = "select * from tb_WordPad";
    DataClass.MyMeans.Mean_Table = "tb_WordPad";
    DataClass.MyMeans.Mean_Field = "WordPad";
    FrmStr = FrmName;
}
InfoAddForm.F_Basic FrmBasic = new PWMS.InfoAddForm.F_Basic();
FrmBasic.Text = FrmStr; //设置窗体名称
FrmBasic.ShowDialog(); //显示调用的窗体
FrmBasic.Dispose();
}
}

```

## 2. GetMenu 方法

GetMenu 方法的主要功能是将 ToolStrip 菜单中的菜单项按照级别动态地添加到 TreeView 控件的相应节点中，其中 treeV 参数表示要添加节点的 TreeView 控件，MenuS 参数表示要获取信息的 ToolStrip 菜单。GetMenu 方法的主要代码如下。

```

#region 将 ToolStrip 控件中的信息添加到 TreeView 控件中
public void GetMenu(TreeView treeV, ToolStrip MenuS)
{
    //遍历 ToolStrip 组件中的一级菜单项
    for (int i = 0; i < MenuS.Items.Count; i++)
    {
        //将一级菜单项的名称添加到 TreeView 组件的根节点中，并设置当前节点的子节点 newNode1
    }
}

```

```

TreeNode newNode1 = treeV.Nodes.Add(MenuS.Items[i].Text);
//将当前菜单项的所有相关信息存入到 ToolStripDropDownItem 对象中
ToolStripDropDownItem newmenu = (ToolStripDropDownItem)MenuS.Items[i];
//判断当前菜单项中是否有二级菜单项
if (newmenu.HasDropDownItems && newmenu.DropDownItems.Count > 0)
    for (int j = 0; j < newmenu.DropDownItems.Count; j++) //遍历二级菜单项
    {
        //将二级菜单名称添加到 TreeView 组件的子节点 newNode1 中，并设置当前节点的子节点
        TreeNode newNode2 = newNode1.Nodes.Add(newmenu.DropDownItems[j].Text);
        //将当前菜单项的所有相关信息存入到 ToolStripDropDownItem 对象中
        ToolStripDropDownItem newmenu2 = (ToolStripDropDownItem)newmenu.DropDownItems[j];
        //判断二级菜单项中是否有三级菜单项
        if (newmenu2.HasDropDownItems && newmenu2.DropDownItems.Count > 0)
            for (int p = 0; p < newmenu2.DropDownItems.Count; p++) //遍历三级菜单项
                //将三级菜单名称添加到 TreeView 组件的子节点 newNode2 中
                newNode2.Nodes.Add(newmenu2.DropDownItems[p].Text);
    }
}
#endregion

```

### 3. Clear\_Control 方法

Clear\_Control 方法的主要功能是清空可视化控件集中指定控件的文本信息及图片，它主要用于在添加数据信息时对相应文本框进行清空，其中 Con 参数表示可视化控件的控件集合。Clear\_Control 方法的主要代码如下。

```

#region 遍历清空指定的控件
public void Clear_Control(Control.ControlCollection Con)
{
    foreach (Control C in Con)
        //遍历可视化组件中的所有控件
        if (C.GetType().Name == "TextBox")
            //判断是否为 TextBox 控件
            if (((TextBox)C).Visible == true)
                //判断当前控件是否为显示状态
                ((TextBox)C).Clear(); //清空当前控件
        if (C.GetType().Name == "MaskedTextBox")
            //判断是否为 MaskedTextBox 控件
            if (((MaskedTextBox)C).Visible == true)
                //判断当前控件是否为显示状态
                ((MaskedTextBox)C).Clear(); //清空当前控件
        if (C.GetType().Name == "ComboBox")
            //判断是否为 ComboBox 控件
            if (((ComboBox)C).Visible == true)
                //判断当前控件是否为显示状态
                ((ComboBox)C).Text = ""; //清空当前控件的 Text 属性值
        if (C.GetType().Name == "PictureBox")
            //判断是否为 PictureBox 控件
            if (((PictureBox)C).Visible == true)
                //判断当前控件是否为显示状态
                ((PictureBox)C).Image = null; //清空当前控件的 Image 属性
    }
}
#endregion

```

### 4. Part\_SaveClass 方法

Part\_SaveClass 方法的主要功能是通过部分控件名 BoxName 与 i 值（数字）相结合，在可视化控件集中查找指定的控件，并根据 Sarr 参数中的字段名组合成添加或修改语句，将生成后的语句存储在公共变量 ADDs 中。Part\_SaveClass 方法的主要代码如下。

```

#region 保存添加或修改的信息
public void Part_SaveClass(string Sarr, string ID1, string ID2, Control.ControlCollection Contr, string BoxName,
string TableName, int n, int m)
{
    string tem_Field = "", tem_Value = "";
    int p = 2;
    if (m == 1){
        if (ID1 != "" && ID2 == ""){
            tem_Field = "ID";
            tem_Value = "" + ID1 + "";
            p = 1;
        }
        else{
            tem_Field = "Sut_id,ID";
            tem_Value = "" + ID1 + "," + ID2 + "";
        }
    }
    else{
        if (m == 2){
            if (ID1 != "" && ID2 == ""){
                tem_Value = "ID=" + ID1 + "";
                p = 1;
            }
            else
                tem_Value = "Sut_ID=" + ID1 + ",ID=" + ID2 + "";
        }
    }

    if (m > 0){ //生成部分添加、修改语句
        string[] Parr = Sarr.Split(Convert.ToChar(','));
        for (int i = p; i < n; i++)
        {
            //通过 BoxName 参数获取要进行操作的控件名称
            string sID = BoxName + i.ToString();
            foreach (Control C in Contr) //遍历控件集中的相关控件
                if (C.GetType().Name == "TextBox" | C.GetType().Name == "MaskedTextBox" | C.GetType().Name == "ComboBox")
                    if (C.Name == sID){ //如果在控件集中找到相应的组件
                        string Ctext = C.Text;
                        if (C.GetType().Name == "MaskedTextBox") //如果当前是 MaskedTextBox 控件
                            Ctext = Date_Format(C.Text); //对当前控件的值进行格式化
                        if (m == 1){ //组合 SQL 语句中 insert 的相关语句
                            tem_Field = tem_Field + "," + Parr[i];
                            if (Ctext == "")
                                tem_Value = tem_Value + "," + "NULL";
                            else
                                tem_Value = tem_Value + "," + "" + Ctext + "";
                        }
                        if (m == 2) //组合 SQL 语句中 update 的相关语句
                            if (Ctext=="")
                                tem_Value = tem_Value + "," + Parr[i] + "=NULL";
                        }
                    }
                }
            }
        }
    }
}

```

```

        else
            tem_Value = tem_Value + "," + Parr[i] + "=" + Ctext + "";
        }
    }
}
ADDs = "";
if (m == 1)                                //生成 SQL 的添加语句
    ADDs = "insert into " + TableName + "(" + tem_Field + ") values(" + tem_Value + ")";
if (m == 2)                                //生成 SQL 的修改语句
    if (ID2 == "")                           //根据 ID2 参数，判断修改语句的条件
        ADDs = "update " + TableName + " set " + tem_Value + " where ID=" + ID1 + "";
    else
        ADDs = "update " + TableName + " set " + tem_Value + " where ID=" + ID2 + "";
}
#endregion

```

Part\_SaveClass 方法中的参数说明如表 22.9 所示。

表 22.9 Part\_SaveClass 方法中的参数说明

参 数 值	描 述
Sarr	要添加或修改表的部分字段名称，字段名必须以“,”号分隔
ID1	数据表中的 ID 字段名，在修改表时可用于条件字段
ID2	数据表中的职工编号字段名，可以为空
Contr	可视化控件集，用于在该控件集中查找控件信息
BoxName	获取控件的部分名称，用于查找相关控件
TableName	要进行添加、修改的数据表名称
n	控件集中要获取控件信息的个数
m	标识，用于判断是生成添加语句还是修改语句

注意：在 Part\_SaveClass 方法中所查找的控件名必须以 BoxName+i 格式命名（如 Word\_1）。

## 5. Find\_Grids 方法

Find\_Grids 方法的主要功能是查找指定可视化控件集中控件名包含 TName 参数值的所有控件，并根据控件名称获取相应表的字段名。当查找的控件为 TextBox 时，根据当前控件的部分名称，查找相应的 ComboBox 控件（用来记录逻辑运算符），通过 ANDSign 参数将具有相关性的控件组合成查询条件，存入到公共变量 FindValue 中。Find\_Grids 方法的主要代码如下。

```

#region 组合查询条件
public void Find_Grids(Control.ControlCollection GBox, string TName, string ANDSign)
{
    string sID = "";                                //定义局部变量
    if (FindValue.Length > 0)
        FindValue = FindValue + ANDSign;
    foreach (Control C in GBox){                     //遍历控件集上的所有控件
        if (C.GetType().Name == "TextBox" | C.GetType().Name == "ComboBox"){ //判断是否为要遍历的控件
            if (C.GetType().Name == "ComboBox" && C.Text != ""){           //当指定控件不为空时
                sID = C.Name;

```

```

//当 TName 参数是当前控件名中的部分信息时
if (sID.IndexOf(TName) > -1){
    //用 “_” 符号分隔当前控件的名称，获取相应的字段名
    string[] Astr = sID.Split(Convert.ToChar('_'));
    //生成查询条件
    FindValue = FindValue + "(" + Astr[1] + " = " + C.Text + ")" + ANDSign;
}
}

if (C.GetType().Name == "TextBox" && C.Text != "")//如果当前为 TextBox 控件，并且控件不为空
{
    sID = C.Name;                                //获取当前控件的名称
    //判断 TName 参数值是否为当前控件名的子字符串
    if (sID.IndexOf(TName) > -1)
    {
        string[] Astr = sID.Split(Convert.ToChar('_'));//以“_”为分隔符，将控件名存入到一维数组中
        string m_Sgin = "";                         //用于记录逻辑运算符
        string mID = "";                           //用于记录字段名
        if (Astr.Length > 2)                      //当数组的元素个数大于 2 时
            mID = Astr[1] + "_" + Astr[2];          //将最后两个元素组成字段名
        else
            mID = Astr[1];                        //获取当前条件所对应的字段名称
        foreach (Control C1 in GBox)                //遍历控件集
        {
            if (C1.GetType().Name == "ComboBox")      //判断是否为 ComboBox 组件
                if ((C1.Name).IndexOf(mID) > -1)//判断当前组件名是否包含条件组件的部分文件名
                {
                    if (C1.Text == "")                  //当查询条件为空时
                        break;                      //退出本次循环
                    else
                    {
                        m_Sgin = C1.Text;           //将条件值存储到 m_Sgin 变量中
                        break;
                    }
                }
            if (m_Sgin != "")                     //当该条件不为空时
                //组合 SQL 语句的查询条件
                FindValue = FindValue + "(" + mID + m_Sgin + C.Text + ")" + ANDSign;
        }
    }
}

//当存储查询条件的变量不为空时，删除逻辑运算符 AND 和 OR
if (FindValue.Length > 0)
{
    if (FindValue.IndexOf("AND") > -1)           //判断是否用 AND 连接条件
        FindValue = FindValue.Substring(0, FindValue.Length - 4);
    if (FindValue.IndexOf("OR") > -1)             //判断是否用 OR 连接条件
        FindValue = FindValue.Substring(0, FindValue.Length - 3);
}
else

```

```

    FindValue = "";
}
#endregion

```

Find\_Grids 方法中的参数说明如表 22.10 所示。

表 22.10 Find\_Grids 方法中的参数说明

参 数 值	描 述
GBox	用于查找的控件集
TName	获取控件的部分名称，用于查找相关控件
ANDSign	逻辑运算符 AND 或 OR

注意：在 Find\_Grids 方法中所查找的条件控件 ComboBox 或 TextBox 必须以“TName+相应字段名”命名（例如查找民族类别的控件，其控件名为 Find\_Folk，Find\_ 是传递的 Tname 参数值，Folk 则是相应表的字段名）；存储逻辑运算符的 ComboBox 控件必须以“相应表的字段名+\_Sign”命名（例如当 TextBox 控件名为 Find\_Age 时，相应的 ComboBox 控件名为 Age\_Sign），这样便于根据控件名称进行组合。

## 6. GetAutocoding 方法

GetAutocoding 方法的主要功能是在添加数据时自动获取添加数据的编号。其实现过程是通过表名和 ID 字段在表中查找最大的 ID 值，并将 ID 值加 1 进行返回，当表中无记录时，返回“0001”。TableName 参数表示进行自动编号的表名，ID 参数表示数据表的编号字段。GetAutocoding 方法的主要代码如下。

```

#region 自动编号
public String GetAutocoding(string TableName, string ID)
{
    //查找指定表中 ID 号为最大的记录
    SqlDataReader MyDR = MyDataClass.getcom("select max(" + ID + ") NID from " + TableName);
    int Num = 0;
    if (MyDR.HasRows)
    {
        MyDR.Read();                                //当查找到记录时
        if (MyDR[0].ToString() == "")                //读取当前记录
            return "0001";
        Num = Convert.ToInt32(MyDR[0].ToString());   //将当前找到的最大编号转换成整数
        ++Num;                                       //最大编号加 1
        string s = string.Format("{0:0000}", Num);  //将整数值转换成指定格式的字符串
        return s;                                     //返回自动生成的编号
    }
    else
    {
        return "0001";                             //当数据表没有记录时，返回
    }
}
#endregion

```

## 7. TreeMenuF 方法

TreeMenuF 方法是在单击 TreeView 控件的节点时被调用，其主要功能是通过所选节点的文本名称，在MenuStrip 控件中进行遍历查找，如果找到，并且为可用状态，则通过 Show\_Form 方法动态调用相关的窗体。

TreeMenuF 方法的主要代码如下。

```
#region 用 TreeView 控件调用 StatusStrip 控件下各菜单的单击事件
public void TreeMenuF(MenuStrip MenuS, TreeNodeMouseClickEventArgs e)
{
    string Men = "";
    for (int i = 0; i < MenuS.Items.Count; i++) //遍历 MenuStrip 控件中主菜单项
    {
        Men = ((ToolStripDropDownItem)MenuS.Items[i]).Name; //获取主菜单项的名称
        //如果 MenuStrip 控件的菜单项没有子菜单
        if (Men.IndexOf("Menu") == -1)
        {
            //当节点名称与菜单项名称相等时
            if (((ToolStripDropDownItem)MenuS.Items[i]).Text == e.Node.Text)
                //判断当前菜单项是否可用
                if (((ToolStripDropDownItem)MenuS.Items[i]).Enabled == false)
                {
                    MessageBox.Show("当前用户无权限调用" + "\"" + e.Node.Text + "\"" + "窗体");
                    break;
                }
            else
                //调用相应的窗体
                Show_Form(((ToolStripDropDownItem)MenuS.Items[i]).Text.Trim(), 1);
        }
        ToolStripMenuItem newmenu = (ToolStripMenuItem)MenuS.Items[i];
        //遍历二级菜单项
        if (newmenu.HasDropDownItems && newmenu.DropDownItems.Count > 0)
            for (int j = 0; j < newmenu.DropDownItems.Count; j++)
            {
                Men = newmenu.DropDownItems[j].Name; //获取二级菜单项的名称
                if (Men.IndexOf("Menu") == -1)
                {
                    if ((newmenu.DropDownItems[j]).Text == e.Node.Text)
                        if ((newmenu.DropDownItems[j]).Enabled == false)
                        {
                            MessageBox.Show("当前用户无权限调用" + "\"" + e.Node.Text + "\"" + "窗体");
                            break;
                        }
                    else
                        Show_Form((newmenu.DropDownItems[j]).Text.Trim(), 1);
                }
                ToolStripMenuItem newmenu2 = (ToolStripMenuItem)newmenu.DropDownItems[j];
                //遍历三级菜单项
                if (newmenu2.HasDropDownItems && newmenu2.DropDownItems.Count > 0)
                    for (int p = 0; p < newmenu2.DropDownItems.Count; p++)
                    {
                        if ((newmenu2.DropDownItems[p]).Text == e.Node.Text)
                            if ((newmenu2.DropDownItems[p]).Enabled == false)
                            {
                                MessageBox.Show("当前用户无权限调用" + "\"" + e.Node.Text + "\"" + "窗体");
                                break;
                            }
                    }
                }
            }
        }
    }
}
```

```

        }
    else
        if ((newmenu2.DropDownItems[p].Text.Trim() == "员工生日提示" ||
(newmenu2.DropDownItems[p].Text.Trim() == "员工合同提示")
            Show_Form((newmenu2.DropDownItems[p].Text.Trim(), 1);
        else
            Show_Form((newmenu2.DropDownItems[p].Text.Trim(), 2);
        }
    }
}
#endifregion

```

## 8. MainPope 方法

MainPope 方法的主要功能是通过当前登录用户的名称，在权限用户表中查询当前用户的所有权限，并根据权限设置菜单栏中各菜单项的可用状态，其中，MenuS 参数为要设置的菜单栏控件，UName 参数为当前用户的名称。MainPope 方法的主要代码如下。

```

#region 根据用户权限设置主窗体菜单
public void MainPope(MenuStrip MenuS, String UName)
{
    string Str = "";
    string MenuName = "";
    //获取当前登录用户的信息
    DataSet DSet = MyDataClass.getDataSet("select ID from tb_Login where Name=" + UName + "", "tb_
Login");
    string UID = Convert.ToString(DSet.Tables[0].Rows[0][0]);                                //获取当前用户编号
    //获取当前用户的权限信息
    DSet = MyDataClass.getDataSet("select ID,PopeName,Pope from tb_UserPope where ID=" + UID + "", "tb_
UserPope");
    bool bo = false;
    for (int k = 0; k < DSet.Tables[0].Rows.Count; k++)                                         //遍历当前用户的权限名称
    {
        Str = Convert.ToString(DSet.Tables[0].Rows[k][1]);                                     //获取权限名称
        if (Convert.ToInt32(DSet.Tables[0].Rows[k][2]) == 1)                                 //判断权限是否可用
            bo = true;
        else
            bo = false;
        for (int i = 0; i < MenuS.Items.Count; i++)                                         //遍历菜单栏中的一级菜单项
        {
            //记录当前菜单项下的所有信息
            ToolStripMenuItem newmenu = (ToolStripMenuItem)MenuS.Items[i];
            //如果当前菜单项有子级菜单项
            if (newmenu.HasDropDownItems && newmenu.DropDownItems.Count > 0)
                for (int j = 0; j < newmenu.DropDownItems.Count; j++)                         //遍历二级菜单项
                {
                    MenuName = newmenu.DropDownItems[j].Name;                               //获取当前菜单项的名称
                    if (MenuName.IndexOf(Str) > -1)                                       //如果包含权限名称
                        newmenu.DropDownItems[j].Enabled = bo;                            //根据权限设置可用状态
                    //记录当前菜单项的所有信息
                }
        }
    }
}

```

```

        ToolStripDropDownItem newmenu2 = (ToolStripDropDownItem)newmenu.
DropDownItems[j];
        //如果当前菜单项有子级菜单项
        if (newmenu2.HasDropDownItems && newmenu2.DropDownItems.Count > 0)
        //遍历三级菜单项
        for (int p = 0; p < newmenu2.DropDownItems.Count; p++)
        {
            //获取当前菜单项的名称
            MenuName = newmenu2.DropDownItems[p].Name;
            if (MenuName.IndexOf(Str) > -1)           //如果包含权限名称
                newmenu2.DropDownItems[p].Enabled = bo; //根据权限设置可用状态
        }
    }
}
#endregion

```

### 9. Amend\_Pope 方法

Amend\_Pope 方法的主要功能是修改指定用户的权限，其中，GBox 参数为包含权限复选框的容器控件，TID 参数为当前用户的编号。Amend\_Pope 方法的主要代码如下。

```

#region 修改指定用户权限
public void Amend_Pope(Control.ControlCollection GBox, string TID)
{
    string CheckName = "";
    int tt = 0;                                //定义一个变量，用来表示是否拥有权限
    foreach (Control C in GBox)                 //循环查找 GroupBox 包含的控件
    {
        if (C.GetType().Name == "CheckBox")      //判断控件类型是否为 CheckBox
        {
            if (((CheckBox)C).Checked)          //判断复选框是否被选中
                tt = 1;
            else
                tt = 0;
            CheckName = C.Name;
            string[] Astr = CheckName.Split(Convert.ToChar('_')); //截取复选框的名称，并存放到一个数组中
            MyDataClass.getsqlcom("update tb_UserPope set Pope=" + tt + " where (ID=" + TID + ") and
(PopeName=" + Astr[1].Trim() + ")");
        }
    }
}
#endregion

```

## 22.7 登录模块设计

- 本模块使用的数据表：tb\_Login

登录模块主要是通过输入正确的用户名和密码进入主窗体，它可以提高程序的安全性，保护数据资料

不外泄。登录窗体的运行结果如图 22.20 所示。



图 22.20 登录窗体的运行结果

### 22.7.1 设计登录窗体

新建一个 Windows 窗体，并将其命名为 F\_Login.cs，其主要用于实现系统的登录功能，此时将窗体的 FormBorderStyle 属性设置为 None，以便去掉窗体的标题栏。登录窗体用到的主要控件如表 22.11 所示。

表 22.11 登录窗体用到的主要控件

控件类型	控件ID	主要属性设置	用途
TextBox	textName	无	输入登录用户名
	textPass	PasswordChar 属性设置为*	输入登录用户密码
Button	butLogin	Text 属性设置为“登录”	登录
	butClose	Text 属性设置为“取消”	取消
PictureBox	pictureBox1	SizeMode 属性设置为 StretchImage	显示登录窗体的背景图片

### 22.7.2 按回车键时移动鼠标焦点

当用户在“用户名”文本框中输入值并按下回车键时，将鼠标焦点移动到“密码”文本框中；当在“密码”文本框中输入值并按下回车键时，将鼠标焦点移动到“登录”按钮上。实现代码如下。

```
private void textName_KeyPress(object sender, KeyPressEventArgs e)
{
    if (e.KeyChar == '\r') //判断是否按下回车键
        textPass.Focus(); //将鼠标焦点移动到“密码”文本框中
}
private void textPass_KeyPress(object sender, KeyPressEventArgs e)
{
    if (e.KeyChar == '\r') //判断是否按下回车键
        butLogin.Focus(); //将鼠标焦点移动到“登录”按钮上
}
```

### 22.7.3 登录功能的实现

当用户输入用户名和密码后，单击“登录”按钮进行登录。在“登录”按钮的 Click 事件中，首先判断用户名和密码是否为空，如果为空，则弹出提示框，通知用户将登录信息填写完整，否则判断用户名和密码是否正确，如果正确，则进入本系统。详细代码如下。

```
private void butLogin_Click(object sender, EventArgs e)
{
```

```

if (textName.Text != "" & textPass.Text != "")
{
    //用自定义方法 getcom()在 tb_Login 数据表中查找是否有当前登录用户
    SqlDataReader temDR = MyClass.getcom("select * from tb_Login where Name='" + textName.Text.Trim() + "' and Pass='" + textPass.Text.Trim() + "'");
    bool ifcom = temDR.Read(); //必须用 Read 方法读取数据
    //当有记录时，表示用户名和密码正确
    if (ifcom)
    {
        DataClass.MyMeans.Login_Name = textName.Text.Trim(); //将用户名记录到公共变量中
        DataClass.MyMeans.Login_ID = temDR.GetString(0); //获取当前操作员编号
        DataClass.MyMeans.My_con.Close(); //关闭数据库连接
        DataClass.MyMeans.My_con.Dispose(); //释放所有资源
        DataClass.MyMeans.Login_n = (int)(this.Tag); //记录当前窗体的 Tag 属性值
        this.Close(); //关闭当前窗体
    }
    else
    {
        MessageBox.Show("用户名或密码错误！", "提示", MessageBoxButtons.OK, MessageBoxIcon.Information);
        textName.Text = "";
        textPass.Text = "";
    }
    MyClass.con_close(); //关闭数据库连接
}
else
    MessageBox.Show("请将登录信息填写完整！", "提示", MessageBoxButtons.OK, MessageBoxIcon.Information);
}

```

## 22.8 系统主窗体设计

### ■ 本模块使用的数据表：tb\_UserPop

主窗体是程序操作过程中必不可少的，它是人机交互中的重要环节。通过主窗体，用户可以调用系统相关的各子模块，快速掌握本系统中所实现的各个功能。在企业人事管理系统中，当登录窗体验证成功后，用户将进入主窗体，主窗体被分为 4 个部分：最上方是系统菜单栏，可以通过它调用系统中的所有子窗体；菜单栏下方是工具栏，它以按钮的形式使用户能够方便地调用最常用的子窗体；窗体的左侧是一个树型导航菜单，该导航菜单中的各节点是根据菜单栏中的项自动生成的；在窗体的最下方，用状态栏显示当前登录的用户名。企业人事管理系统主窗体的运行结果如图 22.21 所示。



图 22.21 企业人事管理系统主窗体的运行结果

### 22.8.1 设计菜单栏

菜单栏运行效果如图 22.22 所示。

本系统的菜单栏是通过 ToolStrip 控件实现的，设计菜单栏的具体步骤如下。

(1) 从工具箱中拖放一个 ToolStrip 控件置于企业人事管理系统的主窗体中，如图 22.23 所示。

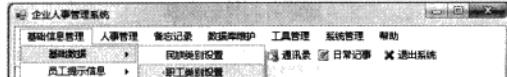


图 22.22 菜单栏运行效果

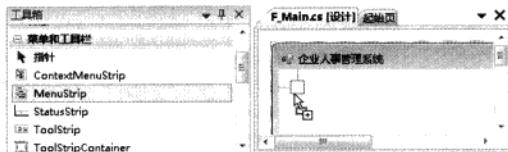


图 22.23 拖放 ToolStrip 控件

(2) 为菜单栏中的各个菜单项设置菜单名称，如图 22.24 所示。在输入菜单名称时，系统会自动产生输入下一个菜单名称的提示。

(3) 选中菜单项，单击其“属性”窗口中的 DropDownListItems 属性后面的按钮，弹出“项集合编辑器”对话框，如图 22.25 所示。在该对话框中可以为菜单项设置 Name 名称，也可以继续通过单击其 DropDownListItems 属性后面的按钮添加子项。

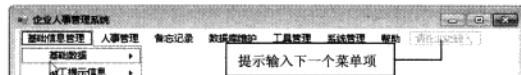


图 22.24 为菜单栏添加项



图 22.25 为菜单栏中的项命名并添加子项

当菜单栏设计完成之后，单击菜单栏中的各菜单项调用相应的子窗体，为了使程序的制作过程更加简便，将所有子窗体的调用封装到 MyModule 公共类的 Show\_Form 方法中，只需要获取当前调用窗体的名称及标识便可以调用相应的窗体。下面以单击“人事管理” / “人事档案管理”菜单项为例进行说明，代码如下。

```
private void Tool_Stuffbasic_Click(object sender, EventArgs e)
{
    //用 MyModule 公共类中的 Show_Form()方法调用各窗体
    MyMenu.Show_Form(sender.ToString().Trim(), 1);
}
```

说明：`sender.ToString().Trim()` 表示获取当前对象的 Text 属性值，即当前单击菜单项的文本，如果调用的是“基础信息管理” / “基础数据”下的子菜单项，则把 `Show_Form` 方法中的 1 改为 2，因为“基础数据”菜单下的所有子菜单项调用的是一个公共窗体。

## 22.8.2 设计工具栏

工具栏运行效果如图 22.26 所示。



图 22.26 工具栏运行效果

本系统的工具栏是通过 ToolStrip 控件实现的，设计工具栏的具体步骤如下。

- (1) 从工具箱中拖放一个 ToolStrip 控件置于企业人事管理系统的主窗体中，单击 ToolStrip 控件后面的下拉按钮，可以选择为工具栏添加哪种控件，如图 22.27 所示。



图 22.27 为工具栏添加控件

- (2) 为工具栏添加完控件之后选中添加的工具栏项，单击鼠标右键，在弹出的快捷菜单中选择“设置图像”命令，可以为工具栏项设置显示的图像，如图 22.28 所示。

- (3) 工具栏中的项默认只显示图像，如果需要同时显示文本和图像，则可以选中工具栏项，单击鼠标右键，在弹出的快捷菜单中选择DisplayStyle/ImageAndText 命令，如图 22.29 所示。

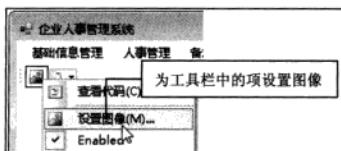


图 22.28 选择“设置图像”命令

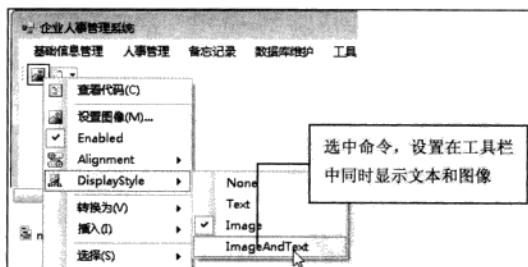


图 22.29 选择DisplayStyle/ImageAndText 命令

按照以上步骤依次添加工具栏项。

工具栏主要是为用户提供一种方便的操作系统常用功能的方式，它在实现时只要调用菜单栏中相应菜单项的 Click 事件即可。例如，“人事档案管理”工具栏项的 Click 事件代码如下。

```
private void Button_Stuffbasic_Click(object sender, EventArgs e)
{
    if (Tool_Stuffbasic.Enabled==true)
        Tool_Stuffbasic_Click(sender, e); //调用人事档案管理菜单项的单击事件
}
```

```

    else
        MessageBox.Show("当前用户无权限调用" + "\"" + ((ToolStripButton)sender).Text + "\"" + "窗体");
}

```

### 22.8.3 设计导航菜单

导航菜单运行效果如图 22.30 所示。

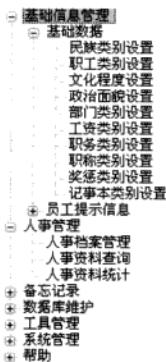


图 22.30 导航菜单运行效果

本系统的导航菜单是通过 TreeView 控件实现的，导航菜单中的项是根据菜单栏自动生成的，它主要调用了公共类 MyModule 下的 GetMenu 方法，代码如下。

```

//实例化公共类 MyModule 的一个对象
ModuleClass.MyModule MyMenu = new PWMS.ModuleClass.MyModule();
//使用菜单栏中的项填充导航菜单
MyMenu.GetMenu(treeView1, menuStrip1);

```

当使用树型导航菜单的下拉列表打开相应的子窗体时，可以在 TreeView 控件的节点单击事件（NodeMouseClick）中调用相应的子窗体。代码如下。

```

private void treeView1_NodeMouseClick(object sender, TreeNodeMouseClickEventArgs e)
{
    if (e.Node.Text.Trim() == "系统退出") //如果当前节点的文本为“系统退出”
    {
        Application.Exit(); //关闭应用程序
    }
    //用 MyModule 公共类中的 TreeMenuF()方法调用各窗体
    MyMenu.TreeMenuF(menuStrip1, e);
}

```

 **说明：**TreeMenuF 方法是在 MyModule 公共类中定义的，用来通过当前节点的文本信息在 menuStrip1 控件中进行遍历查找，如果找到，并且为可用状态，则调用相应窗体，否则将弹出“当前用户无权限调用 XXX 窗体”对话框。

### 22.8.4 设计状态栏

状态栏运行效果如图 22.31 所示。

||欢迎使用企业人事管理系统|| 当前登录用户： mr

图 22.31 状态栏运行效果

本系统的状态栏是通过 StatusStrip 控件实现的，设计状态栏的具体步骤如下。

(1) 从工具箱中拖放一个 StatusStrip 控件置于企业人事管理系统的主窗体中，单击该控件后面的下拉按钮，可以选择为状态栏添加哪种控件，如图 22.32 所示。

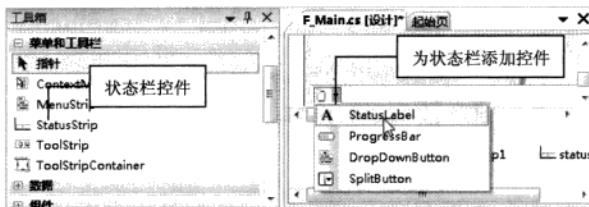


图 22.32 为状态栏添加控件

(2) 本系统中的状态栏主要显示欢迎信息和当前登录的用户，因此这里用到 3 个 StatusLabel 控件，其中前两个 StatusLabel 控件的 Text 属性分别设置为 “||欢迎使用企业人事管理系统||” 和 “当前登录用户：”，第 3 个 StatusLabel 控件用来显示当前登录的用户名。状态栏设计完成后的效果如图 22.33 所示。

||欢迎使用企业人事管理系统|| 当前登录用户：toolStripStatusLabel3

图 22.33 状态栏设计效果

在状态栏中显示当前登录用户名的实现代码如下。

```
statusStrip1.Items[2].Text = DataClass.MyMeans.Login_Name; //在状态栏显示当前登录的用户名
```

## 22.9 人事档案管理模块设计

■ 本模块使用的数据表：tb\_Folk、tb\_Kultur、tb\_Visage、tb\_EmployeeGenre、tb\_Business、tb\_Laborage、tb\_Branch、tb\_Duthcall、tb\_City、tb\_Stuffbusic、tb\_WordResume、tb\_Family、tb\_TrainNote、tb\_RANDP、tb\_Individual

人事档案管理窗体是用来对职工的基本信息、家庭情况、工作简历、培训记录等进行浏览，而且可以进行添加、修改及删除操作。在主窗体中，可以通过菜单栏中的“人事管理” / “人事档案管理”调用人事档案浏览窗体，也可以通过工具栏中的“人事档案管理”按钮或导航菜单中的下拉列表进行调用。人事档案管理窗体由 4 部分组成，分别为分类查询、浏览按钮、职工姓名表和信息操作，其中分类查询主要是通过职工的类别对职工进行简单查询；浏览按钮是通过按钮对职工姓名表进行浏览；职工姓名表用来显示当前所记录的所有职工姓名；信息操作是用来对职工的相关信息进行添加、修改、删除、浏览等操作，并可以将职工的基本信息在 Word 文档中以自定义表格的形式进行显示。人事档案管理窗体的运行结果如图 22.34 所示。

说明：虽然人事档案管理模块中有多个面板，但它们实现的功能大部分是相同的，因此下面仅以“职工基本信息”面板为例进行讲解。

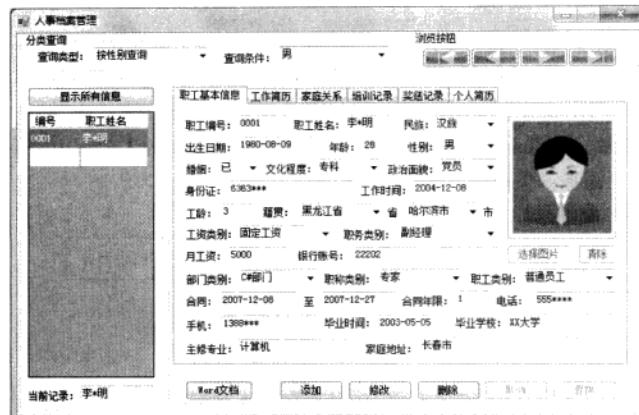


图 22.34 人事档案管理窗体的运行结果

### 22.9.1 设计人事档案管理窗体

新建一个 Windows 窗体，并将其命名为 F\_ManFile.cs，主要用于对企业的人事档案信息进行管理。F\_ManFile 窗体用到的主要控件如表 22.12 所示。

表 22.12 人事档案管理窗体用到的主要控件

控件类型	控件ID	主要属性设置	用途
<input checked="" type="checkbox"/> TextBox	S_0	将其 ReadOnly 属性设置为 True	自动生成职工编号
	S_1	无	输入职工姓名
	S_4	无	输入年龄
	S_9	无	输入身份证号
	S_11	无	输入工龄
	S_25	无	输入月工资
	S_26	无	输入银行账号
	S_29	无	输入合同期限
	S_17	无	输入电话号码
	S_18	无	输入手机号码
	S_19	无	输入毕业学校
	S_20	无	输入主修专业
	S_22	无	输入家庭地址
<input checked="" type="checkbox"/> MaskedTextBox	textBox1	无	显示当前查看的记录是第几条
	S_3	无	输入职工出生日期
	S_10	无	输入工作时间
	S_27	无	输入合同开始日期
	S_28	无	输入合同结束日期
	S_21	无	输入毕业时间

续表

控件类型	控件ID	主要属性设置	用途
ComboBox	comboBox1	其 Items 属性设置可参见图 22.35	选择查询类型
	comboBox2	无	选择查询条件
	S_2	无	选择民族
	S_7	在其 Items 属性中添加两项，分别为“男”和“女”	选择性别
	S_6	在其 Items 属性中添加两项，分别为“已”和“未”	选择婚姻状态
	S_5	无	选择文化程度
	S_8	无	选择政治面貌
	S_23	无	选择省份
	S_24	无	选择市
	S_14	无	选择工资类别
	S_13	无	选择职务类别
	S_15	无	选择部门类别
	S_16	无	选择职称类别
	S_12	无	选择职工类别
Button	button1	无	查看所有员工信息
	N_First	无	查看第一条记录
	N_Previous	无	查看上一条记录
	N_Next	无	查看下一条记录
	N_Cauda	无	查看最后一条记录
	Img_Save	将其 Enabled 属性设置为 False	选择职工头像
	Img_Clear	将其 Enabled 属性设置为 False	清除职工头像
	Sub_Table	无	将职工信息导出到 Word 文档中
	Sut_Add	无	清空各文本框及下拉列表，以执行添加操作
	Sut_Amend	无	将“保存”按钮设置为可用于执行修改操作
	Sut_Delete	无	删除选中的职工信息
	Sut_Cancel	将其 Enabled 属性设置为 False	将各按钮的状态恢复到初始化时的状态
	Sut_Save	将其 Enabled 属性设置为 False	执行职工添加或修改操作
OpenFileDialog	openFileDialog1	无	打开选择职工头像的对话框
PictureBox	S_Photo	将其SizeMode 属性设置为 StretchImage	显示选择的职工头像
DataGridView	dataGridView1	将其SelectionMode 属性设置为 FullRowSelect	显示职工编号和姓名信息
TabControl	tabControl1	添加 6 个面板，并分别将其 Text 属性设置为“职工基本信息”、“工作简历”、“家庭关系”、“培训记录”、“奖惩记录”和“个人简历”	显示人事档案管理窗体中的各个控制面板

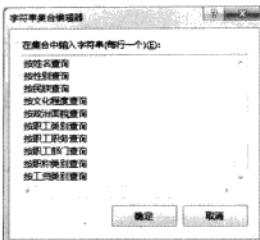


图 22.35 “查询类型”下拉列表的 Items 属性设置

## 22.9.2 添加/修改人事档案信息

单击“添加”按钮，首先调用 MyModule 公共类中的 Clear\_Control 方法，将指定控件集下的控件进行清空，然后根据表名和 ID 字段调用 MyModule 公共类中的 GetAutocoding 方法进行自动编号。代码如下。

```
private void Sut_Add_Click(object sender, EventArgs e)
{
    MyMC.Clear_Control(tabControl1.TabPages[0].Controls); //清空职工基本信息的相应文本框
    S_0.Text = MyMC.GetAutocoding("tb_Stuffbasic", "ID"); //自动添加编号
    hold_n = 1; //用于记录添加操作的标识
    MyMC.Ena_Button(Sut_Add, Sut_Amend, Sut_Cancel, Sut_Save, 0, 0, 1, 1);
    groupBox5.Text = "当前正在添加信息";
    Img_Clear.Enabled = true; //使图片选择按钮为可用状态
    Img_Save.Enabled = true;
}
```

单击“修改”按钮，该按钮的功能只是用 `hold_n` 标识记录当前为修改状态，并修改其他相关按钮的可用状态。代码如下。

```
private void Sut_Amend_Click(object sender, EventArgs e)
{
    hold_n = 2; //用于记录修改操作的标识
    MyMC.Ena_Button(Sut_Add, Sut_Amend, Sut_Cancel, Sut_Save, 0, 0, 1, 1);
    groupBox5.Text = "当前正在修改信息";
    Img_Clear.Enabled = true; //使图片选择按钮为可用状态
    Img_Save.Enabled = true;
}
```

单击“保存”按钮，根据 hold\_ 标识判断执行的是添加操作还是修改操作，并调用“取消”按钮的单击事件功能，将各按钮的状态恢复到初始状态。代码如下。

```
private void Sut_Save_Click(object sender, EventArgs e)
{
    if (tabControl1.SelectedTab.Name == "tabPage6") //如果当前是“个人简历”选项卡
    {
        //通过 MyMeans 公共类中的 getcom()方法查询当前职工是否添加了个人简历
        SqlDataReader Read_Memo = MyDataClass.getcom("Select * from tb_Individual where ID=" + tem_ID
+ "'''");
        if (Read_Memo.Read()) //如果有记录
            //将当前设置的个人简历进行修改
            MyDataClass.getsqlcom("update tb_Individual set Memo=" + Ind_Mome.Text + " where ID=" +
tem_ID + "'''");
    }
}
```

```

    else
        //如果没有记录，则进行添加操作
        MyDataClass.getsqlcom("insert into tb_Individual (ID,Memo) values('" + tem_ID + ":" + Ind_
Mome.Text + "')");
    }
    else
    {
        //如果当前是“职工基本信息”选项卡
        {
            //定义字符串变量，并存储“职工基本信息表”中的所有字段
            string All_Field = "ID,StuffName,Folk,Birthday,Age,Kultur,Marriage,Sex,Visage,IDCard,Workdate,
WorkLength,Employee,Business,Laborage,Branch,Duthcall,Phone,Handset,School,Speciality,GraduateDate,Ad
dress,BeAware,City,M_Pay,Bank,Pact_B,Pact_E,Pact_Y";
            if (hold_n == 1 || hold_n == 2)                                //判断当前是添加还是修改操作
            {
                ModuleClass.MyModule.ADDs = "";                         //清空 MyModule 公共类中的 ADDs 变量
                //用 MyModule 公共类中的 Part_SaveClass()方法组合添加或修改的 SQL 语句
                MyMC.Part_SaveClass(All_Field, S_0.Text.Trim(), "", tabControl1.TabPages[0].Controls, "S_",
"tb_Stuffbasic", 30, hold_n);
                //如果 ADDs 变量不为空，则通过 MyMeans 公共类中的 getsqlcom()方法执行添加、修改操作
                if (ModuleClass.MyModule.ADDs != "")
                    MyDataClass.getsqlcom(ModuleClass.MyModule.ADDs);
            }
            if (Ima_n > 0)                                         //如果图片标识大于 0
            {
                //通过 MyModule 公共类中的 SaveImage()方法将图片存入数据库中
                MyMC.SaveImage(S_0.Text.Trim(), imgBytesIn);
            }
            Sut_Cancel_Click(sender, e);                            //调用“取消”按钮的单击事件
        }
    }
}
}

```

在添加和修改人事档案信息时，当为职工选择头像后，需要将选中的头像转换成字节数组，然后再存放到数据库中。将头像转换成字节数组的实现代码如下。

```

#region 将图片转换成字节数组
public void Read_Image(OpenFileDialog openF, PictureBox MyImage)
{
    openF.Filter = "*.*.jpg|*.jpeg|*.bmp|*.gif";           //指定 OpenFileDialog 控件打开的文件格式
    if (openF.ShowDialog(this) == DialogResult.OK)             //如果打开了图片文件
    {
        try
        {
            //将图片文件存入到 PictureBox 控件中
            MyImage.Image = System.Drawing.Image.FromFile(openF.FileName);
            string strimg = openF.FileName.ToString();          //记录图片的所在路径
            //将图片以文件流的形式进行保存
            FileStream fs = new FileStream(strimg, FileMode.Open, FileAccess.Read);
            BinaryReader br = new BinaryReader(fs);
            imgBytesIn = br.ReadBytes((int)fs.Length);           //将流读入到字节数组中
        }
        catch
        {
            MessageBox.Show("您选择的图片不能被读取或文件类型不对！", "错误", MessageBoxButtons.OK,

```

```

    MessageBoxIcon.Warning);
    S_Photo.Image = null;
}
}
#endregion

```

### 22.9.3 删除人事档案信息

单击“删除”按钮，将职工基本信息表中的当前记录全部删除，同时根据当前记录的编号删除工作简历表、家庭关系表、培训记录表、奖惩记录表和个人简历表中的相关记录。代码如下。

```

private void Sut_Delete_Click(object sender, EventArgs e)
{
    if (dataGridView1.RowCount < 2) //判断 dataGridView1 控件中是否有记录
    {
        MessageBox.Show("数据表为空，不可以删除。");
        return;
    }
    //删除职工信息表中的当前记录及其他相关表中的信息
    MyDataClass.getsqlcom("Delete tb_Stuffbasic where ID=" + S_0.Text.Trim() + "");
    MyDataClass.getsqlcom("Delete tb_WordResume where Sut_ID=" + S_0.Text.Trim() + "");
    MyDataClass.getsqlcom("Delete tb_Family where Sut_ID=" + S_0.Text.Trim() + "");
    MyDataClass.getsqlcom("Delete tb_TrainNote where Sut_ID=" + S_0.Text.Trim() + "");
    MyDataClass.getsqlcom("Delete tb_RANDP where Sut_ID=" + S_0.Text.Trim() + "");
    MyDataClass.getsqlcom("Delete tb_WordResume where Sut_ID=" + S_0.Text.Trim() + "");
    MyDataClass.getsqlcom("Delete tb_Individual where ID=" + S_0.Text.Trim() + ");
    Sut_Cancel_Click(sender, e); //调用“取消”按钮的单击事件
}

```

### 22.9.4 单条件查询人事档案信息

单条件查询人事档案信息运行效果如图 22.36 所示。



图 22.36 单条件查询人事档案信息运行效果

当在“查询类型”下拉列表框中选择查询的类型时，“查询条件”下拉列表框中的值随之改变，然后在“查询条件”下拉列表框中选择要查询的内容，系统将根据选择的查询条件调用自定义方法 Condition\_Lookup 在数据库中查找相关记录，并显示在 DataGridView 控件中。单条件查询人事档案信息的实现代码如下。

```

private void comboBox1_TextChanged(object sender, EventArgs e)
{
    //向 comboBox2 控件中添加相应的查询条件
    switch (comboBox1.SelectedIndex)
    {
        case 0: //职工姓名
            {
                MyMC.CityInfo(comboBox2, "select distinct StuffName from tb_Stuffbasic", 0);
                tem_Field = "StuffName";
            }
    }
}

```

```

        break;
    }
    case 1: //性别
    {
        comboBox2.Items.Clear();
        comboBox2.Items.Add("男");
        comboBox2.Items.Add("女");
        tem_Field = "Sex";
        break;
    }
    case 2:
    {
        MyMC.CoPassData(comboBox2, "tb_Folk"); //民族类别
        tem_Field = "Folk";
        break;
    }
    case 3:
    {
        MyMC.CoPassData(comboBox2, "tb_Kultur"); //文化程度
        tem_Field = "Kultur";
        break;
    }
    case 4:
    {
        MyMC.CoPassData(comboBox2, "tb_Visage"); //政治面貌
        tem_Field = "Visage";
        break;
    }
    case 5:
    {
        MyMC.CoPassData(comboBox2, "tb_EmployeeGenre"); //职工类别
        tem_Field = "Employee";
        break;
    }
    case 6:
    {
        MyMC.CoPassData(comboBox2, "tb_Business"); //职务类别
        tem_Field = "Business";
        break;
    }
    case 7:
    {
        MyMC.CoPassData(comboBox2, "tb_Branch"); //部门类别
        tem_Field = "Branch";
        break;
    }
    case 8:
    {
        MyMC.CoPassData(comboBox2, "tb_Duthcall"); //职称类别
        tem_Field = "Duthcall";
    }
}

```

```

        break;
    }
    case 9:
    {
        MyMC.CoPassData(comboBox2, "tb_Laborage"); //工资类别
        tem_Field = "Laborage";
        break;
    }
}
private void comboBox2_TextChanged(object sender, EventArgs e)
{
    try
    {
        tem_Value = comboBox2.SelectedItem.ToString();
        Condition_Lookup(tem_Value);
    }
    catch
    {
        comboBox2.Text = "";
        MessageBox.Show("只能以选择方式查询。");
    }
}
}

```

实现单条件查询人事档案信息时，用到了自定义方法 Condition\_Lookup，该方法用来根据指定的条件查找职工信息，并显示在 DataGridView 控件中。Condition\_Lookup 方法实现代码如下。

```

#region 按条件显示“职工基本信息”表的内容
public void Condition_Lookup(string C_Value)
{
    MyDS_Grid = MyDataClass.getDataSet("Select * from tb_Stuffbasic where " + tem_Field + "=" + tem_Value
+ "", "tb_Stuffbasic");
    dataGridView1.DataSource = MyDS_Grid.Tables[0];
    textBox1.Text = Grid_Inof(dataGridView1); //显示职工信息表的当前记录
}
#endregion

```

## 22.9.5 逐条查看人事档案信息

“浏览按钮”区域中的 4 个按钮主要实现逐条查看人事档案信息功能，其运行效果如图 22.37 所示。



图 22.37 逐条查看人事档案信息运行效果

当单击这 4 个按钮时，程序根据按钮的 ID 值判断将要执行“第一条”、“前一条”、“下一条”和“最后一条”4 项操作中的一项操作。“浏览按钮”区域中的 4 个按钮的实现代码如下。

```

private void N_First_Click(object sender, EventArgs e) //第一条
{
    int ColInd = 0;
    //判断 DataGridView 控件的当前单元格的列索引
}

```

```

if (dataGridView1.CurrentCell.ColumnIndex == -1 || dataGridView1.CurrentCell.ColumnIndex > 1)
    ColInd = 0;
else
    ColInd = dataGridView1.CurrentCell.ColumnIndex;
if (((Button)sender).Name == "N_First") //判断当前单击的是否为“第一条”
{
    dataGridView1.CurrentCell = this.dataGridView1[ColInd, 0]; //将当前控件的索引设置为 0
    MyMC.Ena_Button(N_First, N_Previous, N_Next, N_Cauda, 0, 0, 1, 1);
}
if (((Button)sender).Name == "N_Previous") //判断当前单击的是否为“前一条”
{
    if (dataGridView1.CurrentCell.RowIndex == 0) //判断当前行的索引是否为 0
    {
        //调用公共类中的方法设置 4 个按钮的状态
        MyMC.Ena_Button(N_First, N_Previous, N_Next, N_Cauda, 0, 0, 1, 1);
    }
    else
    {
        dataGridView1.CurrentCell = this.dataGridView1[ColInd, dataGridView1.CurrentCell.RowIndex - 1]; //重新为当前单元格赋值
        MyMC.Ena_Button(N_First, N_Previous, N_Next, N_Cauda, 1, 1, 1, 1);
    }
}
if (((Button)sender).Name == "N_Next") //判断当前单击的是否为“下一条”
{
    //判断当前行索引是否为最后一行
    if (dataGridView1.CurrentCell.RowIndex == dataGridView1.RowCount - 2)
    {
        //调用公共类中的方法设置 4 个按钮的状态
        MyMC.Ena_Button(N_First, N_Previous, N_Next, N_Cauda, 1, 1, 0, 0);
    }
    else
    {
        //重新为当前单元格赋值
        dataGridView1.CurrentCell = this.dataGridView1[ColInd, dataGridView1.CurrentCell.RowIndex + 1];
        MyMC.Ena_Button(N_First, N_Previous, N_Next, N_Cauda, 1, 1, 1, 1);
    }
}
if (((Button)sender).Name == "N_Cauda") //判断当前单击的是否为“最后一条”
{
    //将当前单元格索引设置为最后一行
    dataGridView1.CurrentCell = this.dataGridView1[ColInd, dataGridView1.RowCount - 2];
    MyMC.Ena_Button(N_First, N_Previous, N_Next, N_Cauda, 1, 1, 0, 0);
}
}

private void N_Previous_Click(object sender, EventArgs e) //上一条
{
    N_First_Click(sender, e);
}

private void N_Next_Click(object sender, EventArgs e) //下一条
{
}

```

```

{
    N_First_Click(sender, e);
}
private void N_Cauda_Click(object sender, EventArgs e) //最后一条
{
    N_First_Click(sender, e);
}
}

```

## 22.9.6 将人事档案信息导出为 Word 文档

将人事档案信息导出为 Word 文档的运行效果如图 22.38 所示。



图 22.38 Word 文档运行效果

为了便于职工信息的存储及打印，可以单击“Word 文档”按钮，将职工信息以表格的形式存入到 Word 文档中。将人事档案信息导出为 Word 文档的实现代码如下。

```

private void but_Table_Click(object sender, EventArgs e)
{
    object Nothing = System.Reflection.Missing.Value;
    object missing = System.Reflection.Missing.Value;
    //创建 Word 文档
    Word.Application wordApp = new Word.ApplicationClass();
    Word.Document wordDoc = wordApp.Documents.Add(ref Nothing, ref Nothing, ref Nothing, ref Nothing);
    wordApp.Visible = true;
    //设置文档宽度
    wordApp.Selection.PageSetup.LeftMargin = wordApp.CentimetersToPoints(float.Parse("2"));
    wordApp.ActiveWindow.ActivePane.HorizontalPercentScrolled = 11;
    wordApp.Selection.PageSetup.RightMargin = wordApp.CentimetersToPoints(float.Parse("2"));
    Object start = Type.Missing;
}

```

```

Object end = Type.Missing;
PictureBox pp = new PictureBox(); //新建一个 PictureBox 控件
int p1 = 0;
for (int i = 0; i < MyDS_Grid.Tables[0].Rows.Count; i++)
{
    try
    {
        byte[] pic = (byte[])(MyDS_Grid.Tables[0].Rows[i][23]); //将数据库中的图片转换成二进制流
        MemoryStream ms = new MemoryStream(pic); //将字节数组存入到二进制流中
        pp.Image = Image.FromStream(ms); //在二进制流 Image 控件中显示
        pp.Image.Save(@"C:\22.bmp"); //将图片存入到指定的路径
    }
    catch
    {
        p1 = 1;
    }
}
object rng = Type.Missing;
string strInfo = "职工基本信息表" + "(" + MyDS_Grid.Tables[0].Rows[i][1].ToString() + ")";
start = 0;
end = 0;
wordDoc.Range(ref start, ref end).InsertBefore(strInfo); //插入文本
wordDoc.Range(ref start, ref end).Font.Name = "Verdana"; //设置字体
wordDoc.Range(ref start, ref end).Font.Size = 20; //设置字体大小
wordDoc.Range(ref start, ref end).ParagraphFormat.Alignment = Word.WdParagraphAlignment.wdAlignParagraphCenter; //设置字体居中
start = strInfo.Length;
end = strInfo.Length;
wordDoc.Range(ref start, ref end).InsertParagraphAfter(); //插入回车
object missingValue = Type.Missing;
//如果 location 超过已有字符的长度将会出错，其必须要比“明细表”串多一个字符
object location = strInfo.Length;
Word.Range rng2 = wordDoc.Range(ref location, ref location);
wordDoc.Tables.Add(rng2, 14, 6, ref missingValue, ref missingValue);
wordDoc.Tables.Item(1).Rows.HeightRule = Word.WdRowHeightRule.wdRowHeightAtLeast;
wordDoc.Tables.Item(1).Rows.Height = wordApp.CentimetersToPoints(float.Parse("0.8"));
wordDoc.Tables.Item(1).Range.Font.Size = 10;
wordDoc.Tables.Item(1).Range.Font.Name = "宋体";
//设置表格样式
wordDoc.Tables.Item(1).Borders.Item(Word.WdBorderType.wdBorderLeft).LineStyle = Word.WdLineStyle.wdLineStyleSingle;
wordDoc.Tables.Item(1).Borders.Item(Word.WdBorderType.wdBorderLeft).LineWidth = Word.WdLineWidth.wdLineWidth050pt;
wordDoc.Tables.Item(1).Borders.Item(Word.WdBorderType.wdBorderLeft).Color = Word.WdColor.wdColorAutomatic;
//设置右对齐
wordApp.Selection.ParagraphFormat.Alignment = Word.WdParagraphAlignment.wdAlignParagraphRight;
//第 1 行显示
wordDoc.Tables.Item(1).Cell(1, 5).Merge(wordDoc.Tables.Item(1).Cell(5, 6));
//第 1 行显示
wordDoc.Tables.Item(1).Cell(6, 5).Merge(wordDoc.Tables.Item(1).Cell(6, 6));

```

```

//第 1 行显示
wordDoc.Tables.Item(1).Cell(9, 4).Merge(wordDoc.Tables.Item(1).Cell(9, 6));
//第 1 行显示
wordDoc.Tables.Item(1).Cell(12, 2).Merge(wordDoc.Tables.Item(1).Cell(12, 6));
//第 1 行显示
wordDoc.Tables.Item(1).Cell(13, 2).Merge(wordDoc.Tables.Item(1).Cell(13, 6));
//第 1 行显示
wordDoc.Tables.Item(1).Cell(14, 2).Merge(wordDoc.Tables.Item(1).Cell(14, 6));
//第 1 行赋值
wordDoc.Tables.Item(1).Cell(1, 1).Range.Text = "职工编号：";
wordDoc.Tables.Item(1).Cell(1, 2).Range.Text = MyDS_Grid.Tables[0].Rows[i][0].ToString();
wordDoc.Tables.Item(1).Cell(1, 3).Range.Text = "职工姓名：";
wordDoc.Tables.Item(1).Cell(1, 4).Range.Text = MyDS_Grid.Tables[0].Rows[i][1].ToString();
//插入图片
if (p1 == 0)
{
    string FileName = @"C:\22.bmp";                                //图片所在路径
    object LinkToFile = false;
    object SaveWithDocument = true;
    object Anchor = wordDoc.Tables.Item(1).Cell(1, 5).Range; //指定图片插入的区域
    //将图片插入到单元格中
    wordDoc.Tables.Item(1).Cell(1, 5).Range.InlineShapes.AddPicture(FileName, ref LinkToFile, ref
SaveWithDocument, ref Anchor);
}
p1 = 0;
//第 2 行赋值
wordDoc.Tables.Item(1).Cell(2, 1).Range.Text = "民族类别：";
wordDoc.Tables.Item(1).Cell(2, 2).Range.Text = MyDS_Grid.Tables[0].Rows[i][2].ToString();
wordDoc.Tables.Item(1).Cell(2, 3).Range.Text = "出生日期：";
try
{
    wordDoc.Tables.Item(1).Cell(2, 4).Range.Text = Convert.ToString(Convert.ToDateTime(MyDS_
Grid.Tables[0].Rows[i][3]).ToShortDateString());
}
catch { wordDoc.Tables.Item(1).Cell(2, 4).Range.Text = ""; }
//第 3 行赋值
wordDoc.Tables.Item(1).Cell(3, 1).Range.Text = "年龄：";
wordDoc.Tables.Item(1).Cell(3, 2).Range.Text = Convert.ToString(MyDS_Grid.Tables[0].Rows[i][4]);
wordDoc.Tables.Item(1).Cell(3, 3).Range.Text = "文化程度：";
wordDoc.Tables.Item(1).Cell(3, 4).Range.Text = MyDS_Grid.Tables[0].Rows[i][5].ToString();
//第 4 行赋值
wordDoc.Tables.Item(1).Cell(4, 1).Range.Text = "婚姻：";
wordDoc.Tables.Item(1).Cell(4, 2).Range.Text = MyDS_Grid.Tables[0].Rows[i][6].ToString();
wordDoc.Tables.Item(1).Cell(4, 3).Range.Text = "性别：";
wordDoc.Tables.Item(1).Cell(4, 4).Range.Text = MyDS_Grid.Tables[0].Rows[i][7].ToString();
//第 5 行赋值
wordDoc.Tables.Item(1).Cell(5, 1).Range.Text = "政治面貌：";
wordDoc.Tables.Item(1).Cell(5, 2).Range.Text = MyDS_Grid.Tables[0].Rows[i][8].ToString();
wordDoc.Tables.Item(1).Cell(5, 3).Range.Text = "单位工作时间：";
try

```

```

{
    wordDoc.Tables.Item(1).Cell(5, 4).Range.Text = Convert.ToString(Convert.ToDateTime(MyDS_Grid.
Tables[0].Rows[0][10]).ToShortDateString());
}
catch { wordDoc.Tables.Item(1).Cell(5, 4).Range.Text = ""; }
//第 6 行赋值
wordDoc.Tables.Item(1).Cell(6, 1).Range.Text = "籍贯: ";
wordDoc.Tables.Item(1).Cell(6, 2).Range.Text = MyDS_Grid.Tables[0].Rows[i][24].ToString();
wordDoc.Tables.Item(1).Cell(6, 3).Range.Text = MyDS_Grid.Tables[0].Rows[i][25].ToString();
wordDoc.Tables.Item(1).Cell(6, 4).Range.Text = "身份证: ";
wordDoc.Tables.Item(1).Cell(6, 5).Range.Text = MyDS_Grid.Tables[0].Rows[i][9].ToString();
//第 7 行赋值
wordDoc.Tables.Item(1).Cell(7, 1).Range.Text = "工龄: ";
wordDoc.Tables.Item(1).Cell(7, 2).Range.Text = Convert.ToString(MyDS_Grid.Tables[0].Rows[i][11]);
wordDoc.Tables.Item(1).Cell(7, 3).Range.Text = "职工类别: ";
wordDoc.Tables.Item(1).Cell(7, 4).Range.Text = MyDS_Grid.Tables[0].Rows[i][12].ToString();
wordDoc.Tables.Item(1).Cell(7, 5).Range.Text = "职务类别: ";
wordDoc.Tables.Item(1).Cell(7, 6).Range.Text = MyDS_Grid.Tables[0].Rows[i][13].ToString();
//第 8 行赋值
wordDoc.Tables.Item(1).Cell(8, 1).Range.Text = "工资类别: ";
wordDoc.Tables.Item(1).Cell(8, 2).Range.Text = MyDS_Grid.Tables[0].Rows[i][14].ToString();
wordDoc.Tables.Item(1).Cell(8, 3).Range.Text = "部门类别: ";
wordDoc.Tables.Item(1).Cell(8, 4).Range.Text = MyDS_Grid.Tables[0].Rows[i][15].ToString();
wordDoc.Tables.Item(1).Cell(8, 5).Range.Text = "职称类别: ";
wordDoc.Tables.Item(1).Cell(8, 6).Range.Text = MyDS_Grid.Tables[0].Rows[i][16].ToString();
//第 9 行赋值
wordDoc.Tables.Item(1).Cell(9, 1).Range.Text = "月工资: ";
wordDoc.Tables.Item(1).Cell(9, 2).Range.Text = Convert.ToString(MyDS_Grid.Tables[0].Rows[i][26]);
wordDoc.Tables.Item(1).Cell(9, 3).Range.Text = "银行账号: ";
wordDoc.Tables.Item(1).Cell(9, 4).Range.Text = MyDS_Grid.Tables[0].Rows[i][27].ToString();
//第 10 行赋值
wordDoc.Tables.Item(1).Cell(10, 1).Range.Text = "合同起始日期: ";
try
{
    wordDoc.Tables.Item(1).Cell(10, 2).Range.Text = Convert.ToString(Convert.ToDateTime(MyDS_Grid.
Tables[0].Rows[i][28]).ToShortDateString());
}
catch { wordDoc.Tables.Item(1).Cell(10, 2).Range.Text = ""; }
wordDoc.Tables.Item(1).Cell(10, 3).Range.Text = "合同结束日期: ";
try
{
    wordDoc.Tables.Item(1).Cell(10, 4).Range.Text = Convert.ToString(Convert.ToDateTime(MyDS_Grid.
Tables[0].Rows[i][29]).ToShortDateString());
}
catch { wordDoc.Tables.Item(1).Cell(10, 4).Range.Text = ""; }
wordDoc.Tables.Item(1).Cell(10, 5).Range.Text = "合同期限: ";
wordDoc.Tables.Item(1).Cell(10, 6).Range.Text = Convert.ToString(MyDS_Grid.Tables[0].Rows[i][30]);
//第 11 行赋值
wordDoc.Tables.Item(1).Cell(11, 1).Range.Text = "电话: ";
wordDoc.Tables.Item(1).Cell(11, 2).Range.Text = MyDS_Grid.Tables[0].Rows[i][17].ToString();

```

```

wordDoc.Tables.Item(1).Cell(11, 3).Range.Text = "手机: ";
wordDoc.Tables.Item(1).Cell(11, 4).Range.Text = MyDS_Grid.Tables[0].Rows[i][18].ToString();
wordDoc.Tables.Item(1).Cell(11, 5).Range.Text = "毕业时间: ";
try
{
    wordDoc.Tables.Item(1).Cell(11, 6).Range.Text = Convert.ToString(Convert.ToDateTime(MyDS_Grid.
Tables[0].Rows[i][21]).ToShortDateString());
}
catch { wordDoc.Tables.Item(1).Cell(11, 6).Range.Text = ""; }
//第 12 行赋值
wordDoc.Tables.Item(1).Cell(12, 1).Range.Text = "毕业学校: ";
wordDoc.Tables.Item(1).Cell(12, 2).Range.Text = MyDS_Grid.Tables[0].Rows[i][19].ToString();
//第 13 行赋值
wordDoc.Tables.Item(1).Cell(13, 1).Range.Text = "主修专业: ";
wordDoc.Tables.Item(1).Cell(13, 2).Range.Text = MyDS_Grid.Tables[0].Rows[i][20].ToString();
//第 14 行赋值
wordDoc.Tables.Item(1).Cell(14, 1).Range.Text = "家庭地址: ";
wordDoc.Tables.Item(1).Cell(14, 2).Range.Text = MyDS_Grid.Tables[0].Rows[i][22].ToString();
wordDoc.Range(ref start, ref end).InsertParagraphAfter();           //插入回车
//设置字体居中
wordDoc.Range(ref start, ref end).ParagraphFormat.Alignment = Word.WdParagraphAlignment.
wdAlignParagraphCenter;
}

```

## 22.10 人事资料查询模块设计

■ 本模块使用的数据表: tb\_Stuffbasic

在人事资料查询窗体中，可以通过在“基本信息”和“个人信息”区域中设置查询条件，对职工基本信息进行查询。人事资料查询窗体的运行结果如图 22.39 所示。

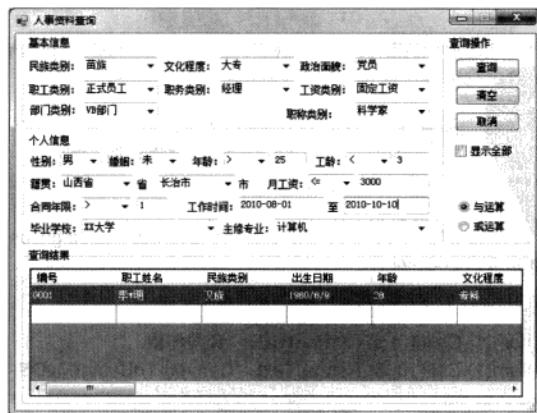
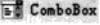
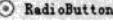


图 22.39 人事资料查询窗体的运行结果

### 22.10.1 设计人事资料查询窗体

新建一个 Windows 窗体，并将其命名为 F\_Find.cs，主要用于对企业的人事档案信息进行查询。F\_Find 窗体用到的主要控件如表 22.13 所示。

表 22.13 人事资料查询窗体用到的主要控件

控件类型	控件ID	主要属性设置	用途
 TextBox	Find_Age	无	输入年龄
	Find_WorkLength	无	输入工龄
	Find_M_Pay	无	输入月工资
	Find_Pact_Y	无	输入合同期限
	Find1_WorkDate	无	输入工作开始时间
	Find2_WorkDate	无	输入工作结束时间
	Find_Folk	无	选择民族类别
	Find_Kultur	无	选择文化程度
	Find_Visage	无	选择政治面貌
	Find_Employee	无	选择职工类别
	Find_Business	无	选择职务类别
	Find_Laborage	无	选择工资类别
	Find_Branch	无	选择部门类别
	Find_Duthcall	无	选择职称类别
	Find_Sex	无	选择性别
	Find_Marriage	无	选择婚姻状态
 ComboBox	Age_Sign	在其 Items 属性中添加 6 项，分别为“=”、“<”、“>”、“<=”、“>=”和“!=”	选择年龄条件
	WorkLength_Sign	在其 Items 属性中添加 6 项，分别为“=”、“<”、“>”、“<=”、“>=”和“!=”	选择工龄条件
	Find_BeAware	无	选择省份
	Find_City	无	选择市
	M_Pay_Sign	在其 Items 属性中添加 6 项，分别为“=”、“<”、“>”、“<=”、“>=”和“!=”	选择月工资条件
 CheckBox	Pact_Y_Sign	在其 Items 属性中添加 6 项，分别为“=”、“<”、“>”、“<=”、“>=”和“!=”	选择合同期限条件
	Find_School	无	选择毕业学校
	Find_Speciality	无	选择主修专业
	checkBox1	无	是否显示全部人事档案信息
	radioButton1	将 Checked 属性设置为 True	是否按与运算执行查询操作
 RadioButton	radioButton2	无	是否按或运算执行查询操作

续表

控件类型	控件ID	主要属性设置	用途
ab Button	button1	无	按指定条件执行查询操作
	button2	无	清空查询条件
	button3	无	关闭当前窗体
DataGridView	dataGridView1	将其SelectionMode 属性设置为 FullRowSelect	显示查询到的人事档案信息

## 22.10.2 多条件查询人事资料

在窗体上设置完查询条件后，单击“查询”按钮进行查询，该按钮通过调用 MyModule 公共类中的 Find\_Grids 方法将指定控件集上的控件组合成查询语句，然后调用 MyMeans 公共类中的 getDataSet 方法在数据表中根据组合的查询语句查询记录，并显示在 dataGridView1 控件上。代码如下。

```
ModuleClass.MyModule MyMC = new PWMS.ModuleClass.MyModule();
DataClass.MyMeans MyDataClass = new PWMS.DataClass.MyMeans();
private void button1_Click(object sender, EventArgs e)
{
    ModuleClass.MyModule.FindValue = "";
    string Find_SQL = Sut_SQL;
    MyMC.Find_Grids(groupBox1.Controls, "Find", ARsign); //清空存储查询语句的变量
    MyMC.Find_Grids(groupBox2.Controls, "Find", ARsign); //存储显示数据表中所有信息的 SQL 语句
    //将指定控件集下的控件组合成查询条件
    MyMC.Find_Grids(groupBox2.Controls, "Find", ARsign);
    //当合同的起始日期和结束日期不为空时
    if (MyMC.Date_Format(Find1_WorkDate.Text) != "" && MyMC.Date_Format(Find2_WorkDate.Text) != "")
    {
        if (ModuleClass.MyModule.FindValue != "") //如果 FindValue 字段不为空
            //用 ARsign 变量连接查询条件
            ModuleClass.MyModule.FindValue = ModuleClass.MyModule.FindValue + ARsign;
        //设置合同日期的查询条件
        ModuleClass.MyModule.FindValue = ModuleClass.MyModule.FindValue + " (" + "workdate>=" +
        Find1_WorkDate.Text + " AND workdate<=" + Find2_WorkDate.Text + ")";
    }
    if (ModuleClass.MyModule.FindValue != "") //如果 FindValue 字段不为空
        //将查询条件添加到 SQL 语句的尾部
        Find_SQL = Find_SQL + " where " + ModuleClass.MyModule.FindValue;
    //按照指定的条件进行查询
    MyDS_Grid = MyDataClass.getDataSet(Find_SQL, "tb_Stuffbasic");
    //在 dataGridView1 控件中显示查询的结果
    dataGridView1.DataSource = MyDS_Grid.Tables[0];
    dataGridView1.AutoGenerateColumns = true;
    checkBox1.Checked = false;
}
```

## 22.11 通讯录模块设计

■ 本模块使用的数据表：tb\_AddressBook

通讯录模块主要对企业人事管理系统中的通讯录信息进行管理，包括对通讯录信息的添加、修改、删

除和查询等操作。通讯录窗体的运行结果如图 22.40 所示。

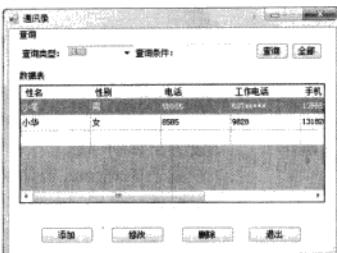


图 22.40 通讯录窗体的运行结果

### 22.11.1 设计通讯录窗体

新建一个 Windows 窗体，并将其命名为 F\_AddressList.cs，主要用于对企业的通讯录信息进行管理。F\_AddressList 窗体用到的主要控件如表 22.14 所示。

表 22.14 通讯录窗体用到的主要控件

控件类型	控件ID	主要属性设置	用途
ComboBox	comboBox1	在其 Items 属性中添加 3 项，分别为“姓名”、“性别”和“邮箱地址”	选择查询的类型
TextBox	textBox1	无	输入查询条件
Button	button5	无	按指定的条件查询通讯录信息
	button1	无	显示全部通讯录信息
	Address_Add	无	打开“添加通讯录”窗体
	Address_Amend	无	打开“修改通讯录”窗体
	Address_Delete	无	删除选中的通讯录信息
	Address_Quit	无	关闭当前窗体
DataGridView	dataGridView1	将其SelectionMode 属性设置为 FullRowSelect，ReadOnly 属性设置为 True	显示通讯录信息

### 22.11.2 添加/修改通讯录信息

添加通讯录和修改通讯录窗体的运行效果分别如图 22.41 和图 22.42 所示。

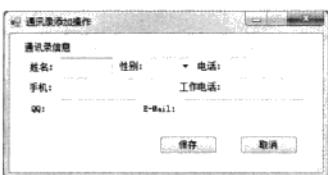


图 22.41 添加通讯录

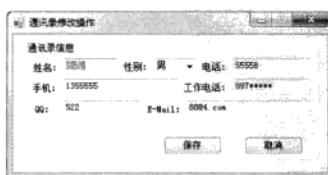


图 22.42 修改通讯录

在 F\_AddressList 窗体中单击“添加” / “修改”按钮，实例化 F\_Address 窗体的一个对象，并分别为该

对象的 Tag 属性赋值为 1 和 2，以标识在 F\_Address 窗体中将执行哪种操作。“添加” / “修改”按钮的实现代码如下。

```
private void Address_Add_Click(object sender, EventArgs e)
{
    //实例化 F_Address 窗体类对象
    InfoAddForm.F_Address FrmAddress = new PWMS.InfoAddForm.F_Address();
    FrmAddress.Text = "通讯录添加操作";           //设置窗体的标题
    FrmAddress.Tag = 1;                          //设置 F_Address 窗体的 Tag 属性为 1，以标识执行添加操作
    FrmAddress.ShowDialog(this);                 //以对话框形式显示窗体
    ShowAll();
}

private void Address_Amend_Click(object sender, EventArgs e)
{
    //实例化 F_Address 窗体类对象
    InfoAddForm.F_Address FrmAddress = new PWMS.InfoAddForm.F_Address();
    FrmAddress.Text = "通讯录修改操作";           //设置窗体的标题
    FrmAddress.Tag = 2;                          //设置 F_Address 窗体的 Tag 属性为 2，以标识执行修改操作
    FrmAddress.ShowDialog(this);                 //以对话框形式显示窗体
    ShowAll();
}
```

在“添加” / “修改”按钮的 Click 事件中用到了 ShowAll 方法，该方法为自定义的无返回值类型方法，它主要用来将通讯录信息显示在 DataGridView 控件中，并根据 DataGridView 控件中的行数确定“修改”按钮和“删除”按钮的可用状态。ShowAll 方法的实现代码如下。

```
public void ShowAll()
{
    ModuleClass.MyModule.Address_ID = "";
    //调用公共类中的方法获取所有通讯录信息，并存储到 DataSet 数据集中
    MyDS_Grid = MyDataClass.getDataSet(AllSql, "tb_AddressBook");
    dataGridView1.DataSource = MyDS_Grid.Tables[0];      //为 DataGridView 控件设置数据源
    dataGridView1.Columns[0].Visible = false;            //设置 DataGridView 控件的第一列不可见
    if (dataGridView1.RowCount > 1)                      //判断 DataGridView 控件中是否有行
    {
        Address_Amend.Enabled = true;                  //设置“修改”按钮可用
        Address_Delete.Enabled = true;                //设置“删除”按钮可用
    }
    else
    {
        Address_Amend.Enabled = false;                //设置“修改”按钮不可用
        Address_Delete.Enabled = false;              //设置“删除”按钮不可用
    }
}
```

F\_Address 窗体在加载时，首先判断其 Tag 属性值，如果为 1，则为“通讯录添加”窗体，这时调用 MyModule 公共类中的 GetAutocoding 方法自动生成一个通讯录编号；如果为 2，则为“通讯录修改”窗体，这时需要将指定通讯录编号的所有信息显示在相应的 TextBox 文本框中。F\_Address 窗体的 Load 事件代码如下。

```
private void F_Address_Load(object sender, EventArgs e)
{
    if ((int)(this.Tag) == 1)                         //判断窗体的 Tag 属性值是否为 1，以标识执行添加操作
```

```

{
    //自动生成通讯录编号
    Address_ID = MyMC.GetAutocoding("tb_AddressBook", "ID");
}
if ((int)this.Tag == 2) //判断窗体的 Tag 属性值是否为 2, 以标识执行添加操作
{
    //根据指定条件查找通讯录信息, 并将结果存储在 DataSet 数据集中
    MyDS_Grid = MyDataClass.getDataSet("select ID,Name,Sex,Phone,Handset,WordPhone,QQ,E_Mail
from tb_AddressBook where ID=" + ModuleClass.MyModule.Address_ID + "", "tb_AddressBook");
    Address_ID = MyDS_Grid.Tables[0].Rows[0][0].ToString(); //记录通讯录编号
    this.Address_1.Text = MyDS_Grid.Tables[0].Rows[0][1].ToString(); //显示姓名
    this.Address_2.Text = MyDS_Grid.Tables[0].Rows[0][2].ToString(); //显示性别
    this.Address_3.Text = MyDS_Grid.Tables[0].Rows[0][3].ToString(); //显示电话
    this.Address_4.Text = MyDS_Grid.Tables[0].Rows[0][4].ToString(); //显示手机号
    this.Address_5.Text = MyDS_Grid.Tables[0].Rows[0][5].ToString(); //显示工作电话
    this.Address_6.Text = MyDS_Grid.Tables[0].Rows[0][6].ToString(); //显示 QQ 号码
    this.Address_7.Text = MyDS_Grid.Tables[0].Rows[0][7].ToString(); //显示 E-Mail
}
}

在 F_Address 窗体中单击“保存”按钮, 判断“姓名”文本框是否为空, 如果不为空, 则执行通讯录添加或修改操作, 并在添加或修改成功后将该窗体关闭, 否则弹出“人员姓名不能为空”信息提示框。“保存”按钮的实现代码如下。
private void button1_Click(object sender, EventArgs e)
{
    if (this.Address_1.Text != "")
    {
        MyMC.Part_SaveClass("ID,Name,Sex,Phone,Handset,WordPhone,QQ,E_Mail", Address_ID, "", this.
groupBox1.Controls["Address_"], "tb_AddressBook", 8, (int)this.Tag); //调用公共类中的方法组合 SQL 语句
        MyDataClass.getsqlcom(ModuleClass.MyModule.ADDs); //执行 SQL 语句
        this.Close(); //关闭当前窗体
    }
    else
        MessageBox.Show("人员姓名不能为空。");
}
}

```

### 22.11.3 删除通讯录信息

在 F\_AddressList 窗体中单击“删除”按钮, 调用 MyMeans 公共类中的 getsqlcom 方法执行删除通讯录信息操作。“删除”按钮的实现代码如下。

```

private void Address_Delete_Click(object sender, EventArgs e)
{
    if (MessageBox.Show("确定要删除该条信息吗?", "提示", MessageBoxButtons.OKCancel, MessageBoxIcon.
Question) == DialogResult.OK) //判断是否在弹出的对话框中单击“确定”按钮
    {
        //调用公共类中的方法删除选中的通讯录信息
        MyDataClass.getsqlcom("Delete tb_AddressBook where ID=" + ModuleClass.MyModule.Address_ID + "");
        ShowAll();
    }
}

```

#### 22.11.4 查询通讯录信息

查询通讯录信息的运行效果如图 22.43 所示。



图 22.43 查询通讯录信息的运行效果

当在“查询类型”下拉列表框中选择查询的类型，并在“查询条件”文本框中输入查询关键字后，单击“查询”按钮，程序根据选择的查询类型和输入的查询关键字在数据表中查找相关记录，并显示在 DataGridView 控件中，同时根据 DataGridView 控件中的行数确定“修改”按钮和“删除”按钮的可用状态。查询通讯录信息的实现代码如下。

```
private void button5_Click(object sender, EventArgs e)
{
    if (textBox1.Text == "")
    {
        MessageBox.Show("请输入查询条件。");
        return;
    }
    ModuleClass.MyModule.Address_ID = "";
    //调用公共类中的方法获取所有通讯录信息，并存储到 DataSet 数据集中
    MyDS_Grid = MyDataClass.getDataSet(AllSql+" where "+Find_Field+" like '%"+textBox1.Text.Trim()+"%'",
"tb_AddressBook");
    dataGridView1.DataSource = MyDS_Grid.Tables[0];
    dataGridView1.Columns[0].Visible = false;
    if (dataGridView1.RowCount > 1)
    {
        Address_Amend.Enabled = true;           //为 DataGridView 控件设置数据源
        Address_Delete.Enabled = true;          //设置 DataGridView 控件的第一列不可见
                                                //判断 DataGridView 控件中是否有行
    }
    else
    {
        Address_Amend.Enabled = false;          //设置“修改”按钮不可用
        Address_Delete.Enabled = false;         //设置“删除”按钮不可用
    }
}
```

## 22.12 用户设置模块设计

- 本模块使用的数据表：tb\_Login、tb\_UserPope

用户设置模块主要对企业人事管理系统中的用户信息进行管理，包括对用户信息的添加、修改和删除等操作，而且还可以为指定的用户设置操作权限。另外，如果要对管理员信息进行修改、删除和设置操作权限操作，系统会提示不能对管理员进行操作。用户设置窗体的运行结果如图 22.44 所示。

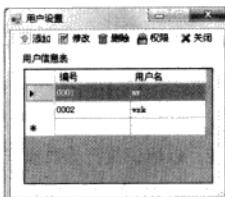


图 22.44 用户设置窗体的运行结果

### 22.12.1 设计用户设置窗体

新建一个 Windows 窗体，并将其命名为 F\_User.cs，主要用于对该系统的用户信息进行管理。F\_User 窗体用到的主要控件如表 22.15 所示。

表 22.15 用户设置窗体用到的主要控件

控件类型	控件ID	主要属性设置	用途
ToolStrip	toolStrip1	添加 5 个 ToolStripButton 按钮，并分别命名为 tool_UserAdd、tool_UserAmend、tool_UserDelete、tool_UserPopedom 和 tool_Close	作为该窗体中的工具栏
DataGridView	dataGridView1	将其 SelectionMode 属性设置为 FullRowSelect	显示用户信息

### 22.12.2 添加/修改用户信息

添加用户信息和修改用户信息窗体的运行效果分别如图 22.45 和图 22.46 所示。

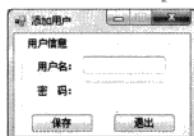


图 22.45 添加用户信息



图 22.46 修改用户信息

在 F\_User 窗体中，单击工具栏中的“添加”/“修改”按钮，实例化 F\_UserAdd 窗体的一个对象，并分别为该对象的 Tag 属性赋值为 1 和 2，以标识在 F\_UserAdd 窗体中将执行哪种操作。工具栏中“添加”/“修改”按钮的实现代码如下。

```

private void tool_UserAdd_Click(object sender, EventArgs e)
{
    //实例化 F_UserAdd 窗体类对象
    Perform.F_UserAdd FrmUserAdd = new F_UserAdd();
    //设置 F_UserAdd 窗体的 Tag 属性为 1，以标识执行添加操作
    FrmUserAdd.Tag = 1;
    FrmUserAdd.Text = tool_UserAdd.Text + "用户";
    FrmUserAdd.ShowDialog(this);
}

private void tool_UserAmend_Click(object sender, EventArgs e)
{
    if (ModuleClass.MyModule.User_ID.Trim() == "0001")          //判断选择的是否为超级用户
    {
}

```

```

    {
        MessageBox.Show("不能修改超级用户。");
        return;
    }
    PerForm.F_UserAdd FrmUserAdd = new F_UserAdd();           //实例化 F_UserAdd 窗体类对象
    //设置 F_UserAdd 窗体的 Tag 属性为 2，以标识执行修改操作
    FrmUserAdd.Tag = 2;
    FrmUserAdd.Text = tool_UserAmend.Text + "用户";
    FrmUserAdd.ShowDialog(this);                                //设置 F_UserAdd 窗体的标题
                                                               //以对话框形式显示窗体
}

```

在 F\_UserAdd 窗体中单击“保存”按钮，判断“用户名”文本框和“密码”文本框是否为空，如果为空，弹出提示信息；否则，根据该窗体的 Tag 属性值判断是执行用户添加操作还是执行用户修改操作。“保存”按钮的实现代码如下。

```

private void button1_Click(object sender, EventArgs e)
{
    if (text_Name.Text == "" && text_Pass.Text == "")          //判断用户名和密码是否为空
    {
        MessageBox.Show("请将用户名和密码添加完整。");
        return;
    }
    DSet = MyDataClass.getDataSet("select Name from tb_Login where Name=" + text_Name.Text + "", "tb_Login");
    //判断窗体的 Tag 属性是否为 2，以执行修改操作
    if ((int)this.Tag == 2 && text_Name.Text == ModuleClass.MyModule.User_Name)
    {
        MyDataClass.execSQL("update tb_Login set Name=" + text_Name.Text + ",Pass=" + text_Pass.Text
+ " where ID=" + ModuleClass.MyModule.User_ID + "");
        return;
    }
    if (DSet.Tables[0].Rows.Count > 0)                          //判断用户名是否已经存在
    {
        MessageBox.Show("当前用户名已存在，请重新输入。"); //弹出提示信息
        text_Name.Text = "";
        text_Pass.Text = "";
        return;
    }
    //判断窗体的 Tag 属性是否为 1，以执行添加操作
    if ((int)this.Tag == 1)
    {
        AutoID = MyMC.GetAutocoding("tb_Login", "ID");          //自动生成编号
        //调用公共类中的方法添加用户信息
        MyDataClass.execSQL("insert into tb_Login (ID,Name,Pass) values(" + AutoID + "," + "
text_Name.Text + "," + text_Pass.Text + ")");
        MyMC.ADD_Pope(AutoID, 0);                                //为新添加的用户设置权限
        MessageBox.Show("添加成功。");
    }
    else
    {
        //调用公共类中的方法修改用户信息
        MyDataClass.execSQL("update tb_Login set Name=" + text_Name.Text + ",Pass=" + text_Pass.Text

```

```

+ " where ID=" + ModuleClass.MyModule.User_ID + "'");
//判断新添加的用户编号是否与登录用户的编号相同
if (ModuleClass.MyModule.User_ID == DataClass.MyMeans.Login_ID)
    DataClass.MyMeans.Login_Name = text_Name.Text; //设置登录用户名为“用户名”文本框的值
    MessageBox.Show("修改成功。");
}
this.Close(); //关闭当前窗体
}

```

### 22.12.3 删除用户基本信息

在 F\_User 窗体中，单击工具栏中的“删除”按钮，判断要删除的用户是否为管理员，如果是，则弹出提示信息，提示不能修改管理员信息；否则，删除选中的用户信息，同时删除其权限信息。工具栏中“删除”按钮的实现代码如下。

```

private void tool_UserDelete_Click(object sender, EventArgs e)
{
    if (ModuleClass.MyModule.User_ID != "")
    {
        if (ModuleClass.MyModule.User_ID.Trim() == "0001") //判断要删除的用户是不是超级用户
        {
            MessageBox.Show("不能删除超级用户。");
            return;
        }
        //删除用户信息
        MyDataClass.getsqlcom("Delete tb_Login where ID=" + ModuleClass.MyModule.User_ID.Trim() + "");
        //删除用户权限信息
        MyDataClass.getsqlcom("Delete tb_UserPope where ID=" + ModuleClass.MyModule.User_ID.Trim() + "");
        //在数据库中查找所有用户信息，并将结果存储在 DataSet 数据集中
        MyDS_Grid = MyDataClass.getDataSet("select ID as 编号,Name as 用户名 from tb_Login", "tb_Login");
        dataGridView1.DataSource = MyDS_Grid.Tables[0]; //为 DataGridView 控件设置数据源
    }
    else
        MessageBox.Show("无法删除空数据表。");
}

```

### 22.12.4 设置用户操作权限

在 F\_User 窗体中，单击工具栏中的“权限”按钮，弹出 F\_UserPope 窗体，如图 22.47 所示。



图 22.47 设置用户操作权限运行效果

在用户权限设置窗体中可以设置用户的权限，在该窗体中选中要拥有权限的复选框，单击“保存”按钮，调用 MyModule 公共类中的 Amend\_Pope 方法为用户设置权限，同时将 MyMeans 公共类中的静态变量 Login\_n 设置为 2，以便在调用“重新登录”窗体时使用新设置的权限对其进行初始化。设置用户操作权限的实现代码如下。

```
private void User_Save_Click(object sender, EventArgs e)
{
    //调用公共类的 Amend_Pope 方法为指定的用户设置权限
    MyMC.Amend_Pope(groupBox2.Controls, ModuleClass.MyModule.User_ID);
    //判断登录用户的编号是否与修改的用户编号相同
    if (DataClass.MyMeans.Login_ID == ModuleClass.MyModule.User_ID)
        //将静态变量 Login_n 设置为 2，以便在调用“重新登录”窗体时使用新设置的权限对其进行初始化
        DataClass.MyMeans.Login_n = 2;
}
```

## 22.13 数据库维护模块设计

数据库维护模块主要对企业人事管理系统中的数据库信息进行备份和还原操作，其运行结果如图 22.48 所示。

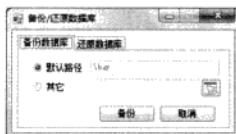


图 22.48 数据库维护窗体的运行结果

### 22.13.1 设计数据库维护窗体

新建一个 Windows 窗体，并将其命名为 F\_HaveBack.cs，主要用于对该系统的数据库进行备份和还原。F\_HaveBack 窗体用到的主要控件如表 22.16 所示。

表 22.16 数据库维护窗体用到的主要控件

控件类型	控件ID	主要属性设置	用途
TextBox	textBox1	将 Text 属性设置为 \bar	备份数据库的默认路径
	textBox2	无	指定备份数据库的路径
	textBox3	无	输入备份文件的路径名
Button	button1	无	执行数据库备份操作
	button2	无	选择要存放备份文件的路径
	button3	无	关闭当前窗体
	button4	无	选择备份文件的存放路径
	button5	无	执行数据库还原操作
	button6	无	关闭当前窗体
RadioButton	radioButton1	将 Checked 属性设置为 True	是否将备份文件放到默认路径中
	radioButton2	无	是否指定新的备份文件路径

续表

控件类型	控件ID	主要属性设置	用途
OpenFileDialog	openFileDialog1	无	选择备份文件对话框
FolderBrowserDialog	folderBrowserDialog1	无	选择存放备份文件路径的对话框
TabControl	tabControl1	添加两个面板，并分别将其Text属性设置为“备份数据库”和“还原数据库”	显示“备份数据库”和“还原数据库”两个面板

### 22.13.2 备份数据库

在“备份数据库”面板中单击“备份”按钮，程序首先判断是将备份文件存到默认路径下还是存放到用户选择的路径下，然后对数据库文件进行备份。备份数据库的实现代码如下。

```
private void button1_Click(object sender, EventArgs e)
{
    string Str_dar = "";
    if (radioButton1.Checked == true) //判断默认路径是否选中
        Str_dar = System.Environment.CurrentDirectory + "\\bar\\";
    if (radioButton2.Checked == true) //判断自定义路径是否选中
        Str_dar = textBox2.Text + "\\";
    if (textBox2.Text == "" & radioButton2.Checked == true)
    {
        MessageBox.Show("请选择备份数据库文件的路径。");
        return;
    }
    try
    {
        //定义数据库备份的 SQL 语句
        Str_dar = "backup database db_PWMS to disk=" + Str_dar + (System.DateTime.Now.ToString("yyyy-MM-dd HH:mm:ss")) +
        ToString() + MyMC.Time_Format(System.DateTime.Now.ToString()) + ".bak" + "";
        MyDataClass.getsqlcom(Str_dar); //调用公共类中的方法执行数据库备份操作
        MessageBox.Show("数据备份成功！", "提示", MessageBoxButtons.OK, MessageBoxIcon.Information);
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "提示", MessageBoxButtons.OK, MessageBoxIcon.Information);
    }
}
```

### 22.13.3 还原数据库

还原数据库窗体的运行效果如图 22.49 所示。

在“还原数据库”面板中单击“还原”按钮，程序首先调用 kill 命令将与 db\_PWMS 数据库有关的进程全部强行关闭，然后重新备份该数据库的日志文件，同时对该数据库进行还原操作。还原数据库的实现代码如下。

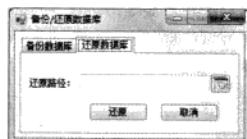


图 22.49 还原数据库窗体的运行效果

```

private void button5_Click(object sender, EventArgs e)
{
    if (textBox3.Text == "") //判断备份文件路径是否为空
    {
        MessageBox.Show("请选择备份数据库文件的路径。");
        return;
    }
    try
    {
        //判断数据库连接状态是否打开
        if (DataClass.MyMeans.My_con.State == ConnectionState.Open)
        {
            DataClass.MyMeans.My_con.Close(); //关闭数据库连接
        }
        string DateStr = "Data Source=mrwxk\wangxiao;Database=master;User id=sa;PWD=";
        SqlConnection conn = new SqlConnection(DateStr);
        conn.Open(); //打开数据库连接
        //-----杀掉所有连接 db_PWMS 数据库的进程-----
        string strSQL = "select spid from master..sysprocesses where dbid=db_id('db_PWMS')";
        SqlDataAdapter Da = new SqlDataAdapter(strSQL, conn); //实例化 SqlDataAdapter 类对象
        DataTable spidTable = new DataTable(); //实例化 DataTable 对象
        Da.Fill(spidTable); //填充 DataTable 数据表
        SqlCommand Cmd = new SqlCommand(); //实例化 SqlCommand 对象
        Cmd.CommandType = CommandType.Text; //设置 SqlCommand 命令的类型
        Cmd.Connection = conn; //设置 SqlCommand 命令的连接对象
        //循环访问 DataTable 数据表中的行
        for (int iRow = 0; iRow < spidTable.Rows.Count; iRow++)
        {
            Cmd.CommandText = "kill " + spidTable.Rows[iRow][0].ToString(); //强行关闭用户进程
            Cmd.ExecuteNonQuery(); //执行 SqlCommand 命令
        }
        conn.Close(); //关闭数据库连接
        conn.Dispose(); //释放数据库连接资源
        //重新连接数据库
        SqlConnection Tem_con = new SqlConnection(DataClass.MyMeans.M_str_sqlcon); //打开数据库连接
        Tem_con.Open();
        //使用数据库还原语句实例化 SqlCommand 对象
        SqlCommand SQLcom = new SqlCommand("backup log db_PWMS to disk=" +
            + textBox3.Text.Trim() + "use master restore database db_PWMS from disk=" +
            + textBox3.Text.Trim() + "", Tem_con);
        SQLcom.ExecuteNonQuery(); //执行数据库还原操作
        SQLcom.Dispose(); //释放 SqlCommand 对象
        Tem_con.Close(); //关闭数据库连接
        Tem_con.Dispose(); //释放数据库连接资源
        MessageBox.Show("数据还原成功！", "提示", MessageBoxButtons.OK, MessageBoxIcon.Information);
        MyDataClass.con_open(); //退出当前应用程序
        MyDataClass.con_close();
        MessageBox.Show("为了避免数据丢失，在数据库还原后将关闭整个系统。");
        Application.Exit();
    }
}

```

```

    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "提示", MessageBoxButtons.OK, MessageBoxIcon.Information);
    }
}

```

## 22.14 运行项目

在模块设计及代码编写完成之后，单击 Visual Studio 2008 开发环境工具栏中的 图标，或者在菜单中选择“调试”/“启动调试”或“调试”/“开始执行（不调试）”命令，运行该项目，弹出企业人事管理系统的“登录”对话框，如图 22.50 所示。



图 22.50 “登录”对话框

在“登录”对话框中输入用户名和密码，单击“登录”按钮，进入企业人事管理系统的主窗体，然后用户可以通过对主窗体中的菜单栏、工具栏和导航菜单进行操作，以便调用其各个子模块。例如，在主窗体中单击工具栏中的“人事档案管理”按钮，可以弹出“人事档案管理”窗体，如图 22.51 所示。在该窗体中，用户可以对人事档案信息进行添加、删除、修改、查询及导出为 Word 等操作。

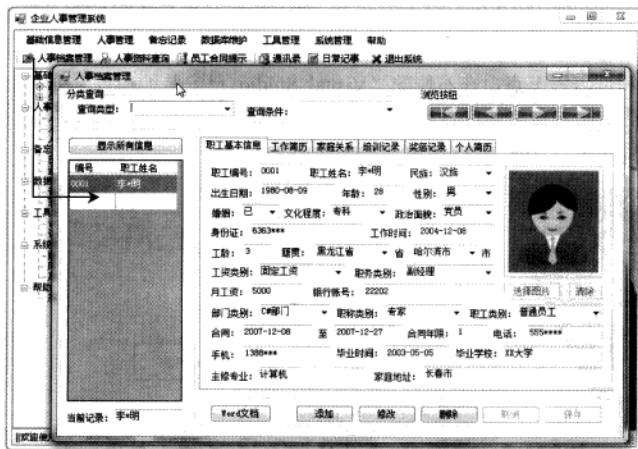


图 22.51 “人事档案管理”窗体

再例如，在主窗体中单击工具栏中的“通讯录”按钮，可以弹出“通讯录”窗体，在“通讯录”窗体中单击“添加”按钮，可以弹出“通讯录添加操作”对话框，如图 22.52 所示，在该对话框中，用户可以对通讯录信息进行添加操作。

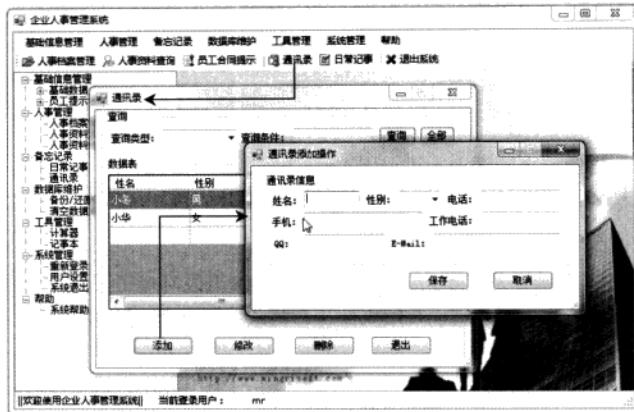


图 22.52 “通讯录添加操作”对话框

## 22.15 系统打包部署

系统开发完成后，如何将系统打包并制作成安装程序在客户机上安装运行是一个很重要的内容，因为只有在客户机上运行使用了，系统才会变得有意义。本节将详细介绍如何对企业人事管理系统进行打包部署。

**说明：**为了便于程序的打包，最好将项目中所用到的图片（如图标和窗体的背景图片）存入到资源文件中。

对企业人事管理系统进行打包部署的步骤如下。

(1) 选中创建 PWMS 项目时自动生成的 PWMS 解决方案，单击鼠标右键，在弹出的快捷菜单中选择“添加”/“新建项目”命令，弹出“新建项目”对话框。在该对话框中的“项目类型”列表框中选择“其他项目类型”/“安装和部署”节点，在右侧的“Visual Studio 已安装的模板”列表中选择“安装项目”选项，在“名称”文本框中输入安装项目名称，这里输入“PWMSSetup”，在“位置”下拉列表框中选择存放安装项目文件的目标地址，如图 22.53 所示。

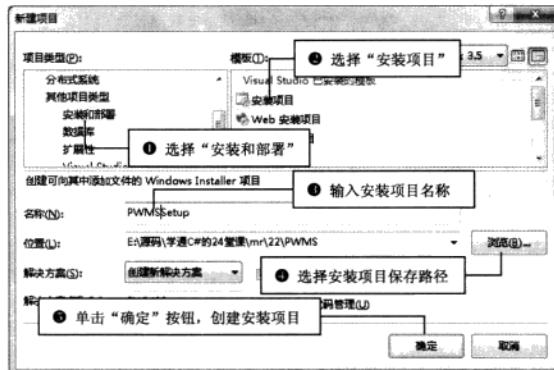


图 22.53 “新建项目”对话框

(2) 单击“确定”按钮，即可创建一个 PWMSSetup 安装项目，如图 22.54 所示。

(3) 在“文件系统”的“目标计算机上的文件系统”节点下选中“应用程序文件夹”，单击鼠标右键，在弹出的快捷菜单中选择“添加”/“项目输出”命令，弹出如图 22.55 所示的“添加项目输出组”对话框。在该对话框中的“项目”下拉列表框中选择要部署的应用程序，然后选择要输出的类型，这里选择“主输出”选项，单击“确定”按钮，即可将项目输出文件添加到 PWMSSetup 安装项目中。

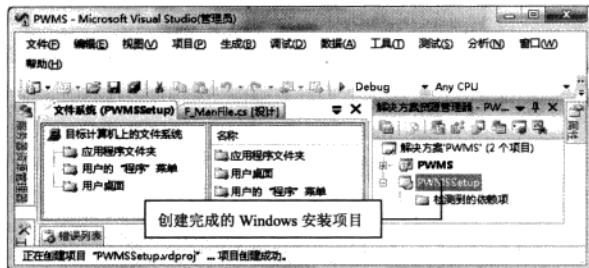


图 22.54 创建完成的 Windows 安装项目

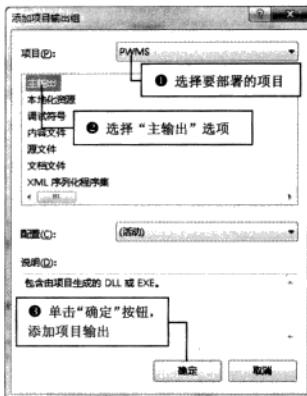


图 22.55 “添加项目输出组”对话框

(4) 在 Visual Studio 2008 集成开发环境的中间位置单击鼠标右键，在弹出的快捷菜单中选择“添加”/“文件”命令，弹出如图 22.56 所示的“添加文件”对话框。在该对话框中选择要添加的内容文件，单击“打开”按钮，即可将选中的内容文件添加到 PWMSSetup 安装项目中。



图 22.56 “添加文件”对话框

注意：在添加内容文件时需要添加数据库文件。

(5) 添加完内容文件之后，在 Visual Studio 2008 集成开发环境的中间位置选中“主输出来自 PWMS（活动）”选项，单击鼠标右键，在弹出的快捷菜单中选择“创建 主输出来自 PWMS（活动）的快捷方式”命令，如图 22.57 所示。

(6) 此时添加了一个“主输出来自 PWMS（活动）的快捷方式”选项，将其重命名为“企业人事管理系统”，如图 22.58 所示。



图 22.57 选择“创建 主输出来自 PWMS(活动)的快捷方式”命令

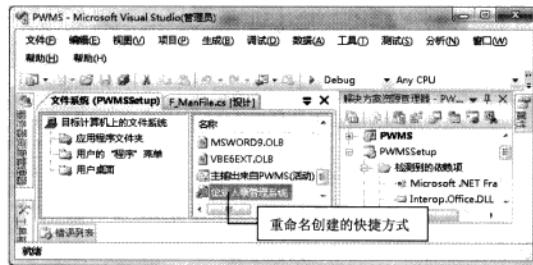


图 22.58 重命名快捷方式

(7) 选中创建的“快捷方式”，然后将其拖放到左边“文件系统”下的“用户桌面”文件夹中，如图 22.59 所示，这样就为该 PWMSSetup 安装项目创建了一个桌面快捷方式。



图 22.59 将“快捷方式”拖放到“用户桌面”文件夹中

(8) 在“解决方案资源管理器”窗口中选中安装项目，单击鼠标右键，在弹出的快捷菜单中选择“视图”/“注册表”命令，在 Windows 安装项目的左侧显示“注册表”选项卡，在该选项卡中依次展开 HKEY\_CURRENT\_USER/Software 节点，然后对注册表项[Manufacturer]进行重命名，如图 22.60 所示。

**注意：**[Manufacturer]注册表项用方括号括起来，表示它是一个属性，它将被替换为输入的部署项目的 Manufacturer 属性值。

(9) 选中添加的注册表项，单击鼠标右键，在弹出的快捷菜单中选择“新建”/“字符串值”命令，为添加的注册表项初始化一个值。选中添加的注册表项值，单击鼠标右键，在弹出的快捷菜单中选择“属性窗口”命令，弹出“属性”窗口，如图 22.61 所示，这里可以对注册表项的值进行修改。

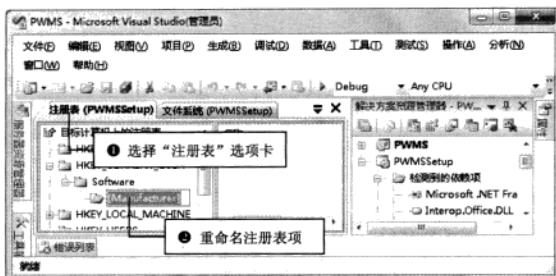


图 22.60 “注册表”选项卡



图 22.61 “属性”窗口

(10) 添加完企业人事管理系统所需的项目输出文件、内容文件、快捷方式和注册表项等内容后，在“解决方案资源管理器”窗口中选中添加的 PWMSSetup 安装项目，单击鼠标右键，在弹出的快捷菜单中选择“生成”命令，即可生成企业人事管理系统的安装程序。生成的企业人事管理系统的安装文件如图 22.62 所示。



图 22.62 生成的企业人事管理系统的安装文件

系统打包完成之后，双击 setup.exe 或 PWMSSetup.msi 文件，即可将企业人事管理系统的安装到自己的计算机上。

## 22.16 开发常见问题与解决

### 22.16.1 程序为什么会无法运行

问题描述：双击企业人事管理系统的可执行文件运行该程序，弹出如图 22.63 所示的信息提示，单击“确

定”按钮后直接退出应用程序。



图 22.63 数据库连接失败的信息提示

解决方法：该错误提示主要是由于无法登录指定的服务器所引起的，解决该问题时只需将程序中 MyMeans 公共类中数据库连接字符串中的 Data Source 属性修改为本机的 SQL Server 2005 服务器名，并重新生成解决方案即可。

## 22.16.2 为什么无法添加职工基本信息

问题描述：在企业人事管理系统的档案管理模块中添加职工基本信息时，当输入基本信息后，单击“保存”按钮，弹出如图 22.64 所示的错误提示框。

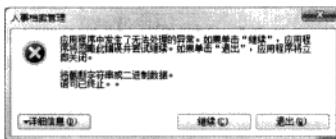


图 22.64 添加职工基本信息时出现的错误提示

解决方法：该问题主要是由于输入职工基本信息时输入了非法的数据而引起的，例如，正常的手机号是 11 位，如果用户输入的手机号位数超过了 11 位，就会出现上面的错误提示。解决该问题有两种方法，分别介绍如下。

- (1) 在输入职工基本信息时，严格按照实际情况输入正确的数据。
- (2) 为“保存”按钮实现添加职工基本信息的代码捕捉异常，如果出现问题，就会弹出提示框，实现代码如下。

```
//定义字符串变量，并存储“职工基本信息表”中的所有字段
string All_Field = "ID,StuffName,Folk,Birthday,Age,Kultur,Marriage,Sex,Visage,ICard,Workdate,WorkLength,
Employee,Business,Laborage,Branch,Duthcall,Phone,Handset,School,Speciality,GraduateDate,Address,BeAw
are,City,M_Pay,Bank,Pact_B,Pact_E,Pact_Y",
try
{
    if (hold_n == 1 || hold_n == 2) //判断当前是添加还是修改操作
    {
        ModuleClass.MyModule.ADDs = ""; //清空 MyModule 公共类中的 ADDs 变量
        //用 MyModule 公共类中的 Part_SaveClass()方法组合添加或修改的 SQL 语句
        MyMC.Part_SaveClass(All_Field, S_0.Text.Trim(), "", tabControl1.TabPages[0].Controls, "S_", "tb_
        Stuffbasic", 30, hold_n);
        //如果 ADDs 变量不为空，则通过 MyMeans 公共类中的 getsqlcom()方法执行添加、修改操作
        if (ModuleClass.MyModule.ADDs != "")
            MyDataClass.getsqlcom(ModuleClass.MyModule.ADDs);
    }
    if (Ima_n > 0) //如果图片标识大于 0
        //如果图片标识大于 0
    }
}
```

```

    {
        //通过 MyModule 公共类中的 SaveImage()方法将图片存入数据库中
        MyMC.SaveImage(S_0.Text.Trim(), imgBytesIn);
    }
    Sut_Cancel_Click(sender, e); //调用“取消”按钮的单击事件
}
catch
{
    MessageBox.Show("请输入正确的职工信息！"); //弹出信息提示框
}

```

### 22.16.3 选择职工头像时出现异常怎么办

问题描述：在企业人事管理系统的人事档案管理模块中添加职工基本信息时，当单击“选择图片”按钮为职工选择头像时，如果想取消该操作，就会出现异常。

解决方法：该问题主要是由于没有判断用户单击的是“打开”按钮还是“取消”按钮而引起的，解决该问题时只需添加一段判断用户选择的是“打开”按钮的代码即可，代码如下。

```
if(openF.ShowDialog(this) == DialogResult.OK) //如果打开了图片文件
```

### 22.16.4 数据库还原不成功应该如何解决

问题描述：在企业人事管理系统的数据库维护模块中，当选择完数据库的备份文件后，单击“还原”按钮，弹出如图 22.65 所示的信息提示。

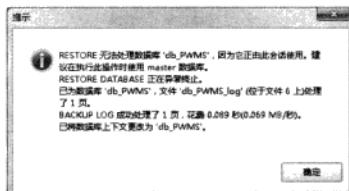


图 22.65 数据库还原时出现的信息提示

解决方法：该错误提示主要是由于 db\_PWMS 数据库正在使用而引起的，解决该问题时只需将 SQL Server 2005 的 SQL Server Management Studio 关闭，同时关闭 PWMS 项目即可。

### 22.16.5 出现 Word 引用问题怎么办

问题描述：运行企业人事管理系统时，弹出如图 22.66 所示的错误提示。

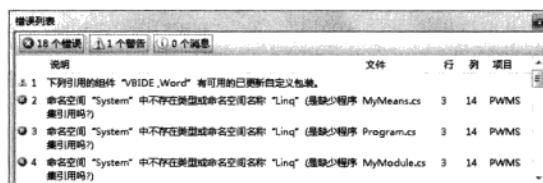


图 22.66 Word 引用问题的信息提示

解决方法：该错误提示主要是由于没有添加 Word 9.0 引用而引起的，解决该问题时需要选中当前项目，单击鼠标右键，在弹出的快捷菜单中选择“添加引用”命令，然后在弹出的“添加引用”对话框中选择 COM 选项卡，在其中找到 Microsoft Word 9.0 Object Library，选中添加即可。

### 22.16.6 COM 选项卡中没有 Word 9.0 引用怎么办

问题描述：向企业人事管理系统中添加 Word 9.0 引用时，发现 COM 选项卡中没有 Microsoft Word 9.0 Object Library，但是有 Microsoft Word 11.0 Object Library，将其添加之后，运行程序，也弹出如图 22.56 所示的错误提示。

解决方法：COM 选项卡中没有 Microsoft Word 9.0 Object Library，说明用户的机器上没有安装 Office 2000，而这个引用只有安装 Office 2000 后才会出现；COM 选项卡中有 Microsoft Word 11.0 Object Library，说明用户的机器上安装了 Office 2003，使用 Office 2003 中的 Word 引用时，需要使用全名引用，例如，对于代码中声明 Word 对象的地方都需要使用 Microsoft.Office.Interop.Word 进行替换。

### 22.16.7 为什么使用全名声明 Word 对象后还出现错误

问题描述：在企业人事管理系统中添加了 Word 11.0 引用，并使用全名声明了 Word 对象后，运行程序，弹出如图 22.67 所示的错误提示。

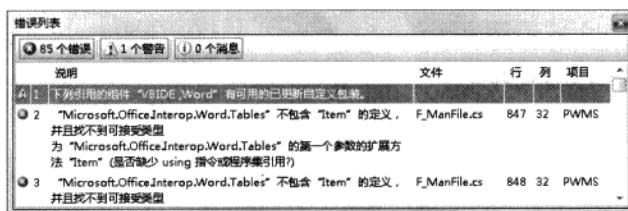


图 22.67 不包括 Item 的错误提示

解决方法：该错误提示主要是由于 Word 11.0 中的 Word 对象中不包括 Item 方法而引起的，解决该问题时需要将“人事档案管理”窗体中的“Word 文档”按钮的 Click 事件下的代码替换为如下代码。

```

object Nothing = System.Reflection.Missing.Value;
object missing = System.Reflection.Missing.Value;
//创建 Word 文档
Microsoft.Office.Interop.Word.Application wordApp = new Microsoft.Office.Interop.Word.ApplicationClass();
Microsoft.Office.Interop.Word.Document wordDoc = wordApp.Documents.Add(ref Nothing, ref Nothing, ref Nothing);
wordApp.Visible = true;
//设置文档宽度
wordApp.Selection.PageSetup.LeftMargin = wordApp.CentimetersToPoints(float.Parse("2"));
wordApp.ActiveWindow.ActivePane.HorizontalPercentScrolled = 11;
wordApp.Selection.PageSetup.RightMargin = wordApp.CentimetersToPoints(float.Parse("2"));
Object start = Type.Missing;
Object end = Type.Missing;
PictureBox pp = new PictureBox();
int p1 = 0;
for (int i = 0; i < MyDS_Grid.Tables[0].Rows.Count; i++)
    //新建一个 PictureBox 控件

```

```

{
    try
    {
        byte[] pic = (byte[])(MyDS_Grid.Tables[0].Rows[i][23]);           //将数据库中的图片转换成二进制流
        MemoryStream ms = new MemoryStream(pic);                          //将字节数组存入到二进制流中
        pp.Image = Image.FromStream(ms);                                //在二进制流 Image 控件中显示
        pp.Image.Save(@"C:\22.bmp");                                    //将图片存入到指定的路径
    }
    catch
    {
        p1 = 1;
    }
    object rng = Type.Missing;
    string strInfo = "职工基本信息表" + "(" + MyDS_Grid.Tables[0].Rows[i][1].ToString() + ")";
    start = 0;
    end = 0;
    wordDoc.Range(ref start, ref end).InsertBefore(strInfo);          //插入文本
    wordDoc.Range(ref start, ref end).Font.Name = "Verdana";          //设置字体
    wordDoc.Range(ref start, ref end).Font.Size = 20;                  //设置字体大小
    wordDoc.Range(ref start, ref end).ParagraphFormat.Alignment = Microsoft.Office.Interop.Word.WdParagraphAlignment.wdAlignParagraphCenter; //设置字体居中
    start = strInfo.Length;
    end = strInfo.Length;
    wordDoc.Range(ref start, ref end).InsertParagraphAfter();           //插入回车
    object missingValue = Type.Missing;
    //如果 location 超过已有字符的长度将会出错，其必须要比“明细表”串多一个字符
    object location = strInfo.Length;
    Microsoft.Office.Interop.Word.Range rng2 = wordDoc.Range(ref location, ref location);
    Microsoft.Office.Interop.Word.Table tab = wordDoc.Tables.Add(rng2, 14, 6, ref missingValue, ref missingValue);
    tab.Rows.HeightRule = Microsoft.Office.Interop.Word.WdRowHeightRule.wdRowHeightAtLeast;
    tab.Rows.Height = wordApp.CentimetersToPoints(float.Parse("0.8"));
    tab.Range.Font.Size = 10;
    tab.Range.Font.Name = "宋体";
    //设置表格样式
    tab.Borders.InsideLineStyle = Microsoft.Office.Interop.Word.WdLineStyle.wdLineStyleSingle;
    tab.Borders.InsideLineWidth = Microsoft.Office.Interop.Word.WdLineWidth.wdLineWidth050pt;
    tab.Borders.InsideColor = Microsoft.Office.Interop.Word.WdColor.wdColorAutomatic;
    wordApp.Selection.ParagraphFormat.Alignment = Microsoft.Office.Interop.Word.WdParagraphAlignment.wdAlignParagraphRight; //设置右对齐
    tab.Cell(1, 5).Merge(tab.Cell(5, 6));                                //第 5 行显示
    tab.Cell(6, 5).Merge(tab.Cell(6, 6));                                //第 6 行显示
    tab.Cell(9, 4).Merge(tab.Cell(9, 6));                                //第 9 行显示
    tab.Cell(12, 2).Merge(tab.Cell(12, 6));                              //第 12 行显示
    tab.Cell(13, 2).Merge(tab.Cell(13, 6));                              //第 13 行显示
    tab.Cell(14, 2).Merge(tab.Cell(14, 6));                              //第 14 行显示
    //第 1 行赋值
    tab.Cell(1, 1).Range.Text = "职工编号：";
    tab.Cell(1, 2).Range.Text = MyDS_Grid.Tables[0].Rows[i][0].ToString();
    tab.Cell(1, 3).Range.Text = "职工姓名：";
    tab.Cell(1, 4).Range.Text = MyDS_Grid.Tables[0].Rows[i][1].ToString();
}

```

```

//插入图片
if (p1 == 0)
{
    string FileName = @"C:\22.bmp"; //图片所在路径
    object LinkToFile = false;
    object SaveWithDocument = true;
    object Anchor = tab.Cell(1, 5).Range; //指定图片插入的区域
    //将图片插入到单元格中
    tab.Cell(1, 5).Range.InlineShapes.AddPicture(FileName, ref LinkToFile, ref SaveWithDocument, ref
Anchor);
}
p1 = 0;
//第 2 行赋值
tab.Cell(2, 1).Range.Text = "民族类别：";
tab.Cell(2, 2).Range.Text = MyDS_Grid.Tables[0].Rows[i][2].ToString();
tab.Cell(2, 3).Range.Text = "出生日期：";
try
{
    tab.Cell(2, 4).Range.Text = Convert.ToString(Convert.ToDateTime(MyDS_Grid.Tables[0].Rows[i][3]).ToShortDateString());
}
catch { tab.Cell(2, 4).Range.Text = ""; }
//第 3 行赋值
tab.Cell(3, 1).Range.Text = "年龄：";
tab.Cell(3, 2).Range.Text = Convert.ToString(MyDS_Grid.Tables[0].Rows[i][4]);
tab.Cell(3, 3).Range.Text = "文化程度：";
tab.Cell(3, 4).Range.Text = MyDS_Grid.Tables[0].Rows[i][5].ToString();
//第 4 行赋值
tab.Cell(4, 1).Range.Text = "婚姻：";
tab.Cell(4, 2).Range.Text = MyDS_Grid.Tables[0].Rows[i][6].ToString();
tab.Cell(4, 3).Range.Text = "性别：";
tab.Cell(4, 4).Range.Text = MyDS_Grid.Tables[0].Rows[i][7].ToString();
//第 5 行赋值
tab.Cell(5, 1).Range.Text = "政治面貌：";
tab.Cell(5, 2).Range.Text = MyDS_Grid.Tables[0].Rows[i][8].ToString();
tab.Cell(5, 3).Range.Text = "单位工作时间：";
try
{
    tab.Cell(5, 4).Range.Text = Convert.ToString(Convert.ToDateTime(MyDS_Grid.Tables[0].Rows[0][10]).ToShortDateString());
}
catch { tab.Cell(5, 4).Range.Text = ""; }
//第 6 行赋值
tab.Cell(6, 1).Range.Text = "籍贯：";
tab.Cell(6, 2).Range.Text = MyDS_Grid.Tables[0].Rows[i][24].ToString();
tab.Cell(6, 3).Range.Text = MyDS_Grid.Tables[0].Rows[i][25].ToString();
tab.Cell(6, 4).Range.Text = "身份证：";
tab.Cell(6, 5).Range.Text = MyDS_Grid.Tables[0].Rows[i][9].ToString();
//第 7 行赋值
tab.Cell(7, 1).Range.Text = "工龄：";

```

```

tab.Cell(7, 2).Range.Text = Convert.ToString(MyDS_Grid.Tables[0].Rows[i][11]);
tab.Cell(7, 3).Range.Text = "职工类别：";
tab.Cell(7, 4).Range.Text = MyDS_Grid.Tables[0].Rows[i][12].ToString();
tab.Cell(7, 5).Range.Text = "职务类别：";
tab.Cell(7, 6).Range.Text = MyDS_Grid.Tables[0].Rows[i][13].ToString();
//第 8 行赋值
tab.Cell(8, 1).Range.Text = "工资类别：";
tab.Cell(8, 2).Range.Text = MyDS_Grid.Tables[0].Rows[i][14].ToString();
tab.Cell(8, 3).Range.Text = "部门类别：";
tab.Cell(8, 4).Range.Text = MyDS_Grid.Tables[0].Rows[i][15].ToString();
tab.Cell(8, 5).Range.Text = "职称类别：";
tab.Cell(8, 6).Range.Text = MyDS_Grid.Tables[0].Rows[i][16].ToString();
//第 9 行赋值
tab.Cell(9, 1).Range.Text = "月工资：";
tab.Cell(9, 2).Range.Text = Convert.ToString(MyDS_Grid.Tables[0].Rows[i][26]);
tab.Cell(9, 3).Range.Text = "银行账号：";
tab.Cell(9, 4).Range.Text = MyDS_Grid.Tables[0].Rows[i][27].ToString();
//第 10 行赋值
tab.Cell(10, 1).Range.Text = "合同起始日期：";
try
{
    tab.Cell(10, 2).Range.Text = Convert.ToString(Convert.ToDateTime(MyDS_Grid.Tables[0].Rows[i][28]).ToShortDateString());
}
catch { tab.Cell(10, 2).Range.Text = ""; }
tab.Cell(10, 3).Range.Text = "合同结束日期：";
try
{
    tab.Cell(10, 4).Range.Text = Convert.ToString(Convert.ToDateTime(MyDS_Grid.Tables[0].Rows[i][29]).ToShortDateString());
}
catch { tab.Cell(10, 4).Range.Text = ""; }
tab.Cell(10, 5).Range.Text = "合同期限：";
tab.Cell(10, 6).Range.Text = Convert.ToString(MyDS_Grid.Tables[0].Rows[i][30]);
//第 11 行赋值
tab.Cell(11, 1).Range.Text = "电话：";
tab.Cell(11, 2).Range.Text = MyDS_Grid.Tables[0].Rows[i][17].ToString();
tab.Cell(11, 3).Range.Text = "手机：";
tab.Cell(11, 4).Range.Text = MyDS_Grid.Tables[0].Rows[i][18].ToString();
tab.Cell(11, 5).Range.Text = "毕业时间：";
try
{
    tab.Cell(11, 6).Range.Text = Convert.ToString(Convert.ToDateTime(MyDS_Grid.Tables[0].Rows[i][21]).ToShortDateString());
}
catch { tab.Cell(11, 6).Range.Text = ""; }
//第 12 行赋值
tab.Cell(12, 1).Range.Text = "毕业学校：";
tab.Cell(12, 2).Range.Text = MyDS_Grid.Tables[0].Rows[i][19].ToString();
//第 13 行赋值

```



```
tab.Cell(13, 1).Range.Text = "主修专业: ";
tab.Cell(13, 2).Range.Text = MyDS_Grid.Tables[0].Rows[i][20].ToString();
//第 14 行赋值
tab.Cell(14, 1).Range.Text = "家庭地址: ";
tab.Cell(14, 2).Range.Text = MyDS_Grid.Tables[0].Rows[i][22].ToString();
wordDoc.Range(ref start, ref end).InsertParagraphAfter();           //插入回车
wordDoc.Range(ref start, ref end).ParagraphFormat.Alignment = Microsoft.Office.Interop.Word.WdParagraphAlignment.wdAlignParagraphCenter;
//设置字体居中
}
```

## 22.17 小 结

本堂课根据软件的开发流程，对企业人事管理系统的开发过程进行了详细讲解。通过对本堂的学习，读者应能够掌握如何用自定义方法对多个不同的数据表进行添加、修改、删除以及多字段组合查询等操作，另外，还应掌握如何将数据库中的信息导出到 Word 文档中，以方便打印。