《C＃程序设计》实验指导书

上机实验指导

网络应用开发的例子非常多，为了让读者对应用编程的各方面都有一个大概的认识，本上机实验指导以实际业务处理为素材，选取了有代表意义的部分内容，经过简化和变形处理，整理为一个个单独的实验。

在安排每个上机实验内容时，功能需求及实现步骤均与实际的业务处理有非常大的差别，而且有些步骤是多余的，或者实现的办法不是最简单的。但是完成实验的目的不仅仅是为了实现需要的功能，还要通过这些步骤领会涉及的多种技巧。另外，由于各个实验之间既有独立性，又相互关联，所以读者按照实验顺序完成一个个实验后， 即能对网络应用编程形成一个连贯的思路，又能快速掌握和巩固涉及的技术。

1、上机实验环境

开发工具：Miscrosoft visual studio 2017及Sql server 2017

2、实验报告要求

使用统一的实验报告模板，内容清晰，注意填写必要的信息：姓名、学号、班级和辅导教师。必须认真填写实验题目、实验目的等；实验步骤中要求列出当次实验中自己认为有意义的操作过程及各种必要的数据输入输出情况；写出主要的功能模块划分、设计界面及关键源代码，以及上机调试过程中遇到的问题和解决办法。

实验一　创建简单的.NET 应用程序

实验目的：

熟悉Miscrosoft Visual Studio开发环境，掌握如何在此开发环境下开发简单的.NET 应用程序，以及调试程序的基本操作技巧。

实验内容：

分别创建不同类型的.NET 应用程序项目，体会基本的设计与编程方法。

实验要求：

1) 通过实验掌握【工具箱】、【属性】窗口、【解决方案资源管理器】等的用法和基本操作技巧。
2) 通过实验观察各种应用程序的程序结构及特点；
3) 通过实验观察生成的可执行文件的存放位置，掌握项目备份与恢复的方法：
4) 通过实验掌握利用断点进行程序调试的方法。

实验步骤 1：创建一个简单的 Windows 应用程序项目

(1). 运行 Microsoft Visual Studio 2017，创建一个 Windows 应用程序项目，在【名称】文本框中将项目名改为 SimpleWindowsApplication，在【位置】文本框中输入保存的目录位置C:\CSharpExperiment，选择【创建解决方案的目录】复选框，然后单击【确定】按钮。

注意：千万不要将【位置】设置为网络映射的驱动器，一定要确保保存位置为本地硬盘的逻辑驱动器。原因有两点：一是用本地硬盘速度快；二是为了确保有足够的读写权限。特别是以后对数据库的操作以及网站设计，要求必须是本地硬盘，而网络映射的驱动器可能会因为没有相应的读写权限导致无法调试。另外还要注意，不要使用#之类的符号作为项目名称，最好用英文单词或者汉语拼音的组合给项目起名，并且使用有意义的名称。

(2). 在【解决方案资源管理器】中，单击 Form1.cs，修改 Form1.cs 为 FormMain.cs，然后分别修改窗体的【Text】属性、【Size】属性和【FormBoardStyle】属性，观察变化。

(3). 单击【BackgroundImage】属性右边的“…”，导入一个本地资源类型的图片。然后 选择 Form1.cs 的设计窗体，分别选择不同的【BackgroundImageLayout】属性值，观察变化， 然后删除该背景图片。

(4). 从【工具箱】中向设计窗体拖放一个 Label 控件，选中该 Label对象，按住<Ctrl>键，用鼠标左键将该对象按纵向排列复制 3 个，然后分别修改其【Text】属性为“编号”、“ 姓名”、“年龄”和“出生日期”。

(5). 选中 label1 对象，修改【AutoSize】属性为“False”，拖动该控件右下角改变大小， 观察变化，然后设置其【Font】属性为“隶书，2号，斜体”，选择【ForeColor】属性为红色，再分别选择不同的【TextAlign】属性值，观察变化。

(6). 选择【工具箱】，双击 TextBox 控件 4 次，观察窗体中添加的情况，然后分别拖动窗体中的 4 个TextBox 对象，放在对应的Label对象右边，并分别修改其【Name】属性为“textBoxId”、“textBoxName”、“textBoxAge”和“textBoxBirthDay”。

(7). 设置 textBoxName 的【Size】属性为“200,21”，然后按住鼠标左键，同时选中 4 个 TextBox 对象，在快捷工具栏中，分别选择“使宽度相同”、“使垂直间距相等”，观察变化。再选择“顶部对齐”，观察 4 个对象是不是叠在了一起，然后按<Ctrl>+<Z>组合键取消这次操作。

(8). 按<F5>键编译并运行，依次按<Tab>键，观察光标焦点顺序是否和想象的一致。然后结束程序运行。

(9). 选择【视图】→【Tab 键顺序】命令，分别按照自己希望的顺序依次单击各个TextBox对象，完成后，按<Esc>键结束<Tab>键顺序设置。注意：如果不希望某个控件用<Tab> 键获取焦点，可以设置其【TabStop】属性为“False”。

(10). 从【工具箱】中向设计窗体拖放一个 Button 控件，然后双击该对象，观察自动生成的代码，并在 button1_Click 事件中添加一行代码：this.Close();

(11). 运行该应用程序，然后单击【button1】按钮结束程序运行。

(12). 在 FormMain.cs 的代码编辑窗体中，利用鼠标左键选中下面的代码：

```
private void button1_Click(object sender, EventArgs e)
{
  this.Close();
}
```

然后单击快捷工具栏的“注释选中行”符号，将这几行代码作为注释。

(13). 运行应用程序，观察出错提示。在提示的对话框中单击【否】按钮。注意：调试时只要出现编译错误，不要单击【是】，因为继续运行没有意义。

(14). 切换到 FormMain.cs 的代码编辑窗体，同时选中刚才变为注释的行，单击快捷工具栏的【取消对选中行的注释】符号，再次运行程序，然后单击【button1】按钮结束应用程序运行。

(15). 在 FormMain.cs 的设计窗体中，修改 button1 的【Name】属性为“buttonOK”，【Text】属性为“确定”， 然后切换到FormMain.cs的代码编辑窗体， 修改 button1_Click 为 buttonOK_Click，运行观察编译出错提示，然后单击【否】按钮放弃运行。

(16). 切换到 FormMain.cs 的设计窗体，单击【确定】按钮，然后单击【属性】窗口中的雷电符号，在“Click”事件右边的下拉列表中选择“buttonOk_Click”，再次运行程序。

(17). 切换到 FormMain.cs 的代码编辑窗体，在 buttonOK_Click 上方单击鼠标右键，在弹出的快捷菜单中选择【重构】→【重命名】命令，将 buttonOK_Click 改为 buttonExit_Click， 重新运行程序，观察是否有编译错误，然后结束程序运行，体会重构的作用。

(18). 在【解决方案资源管理器】中，双击 FormMain.Designer.cs，观察自动生成的代码，体会设计窗体与自动生成的代码之间的关系。注意：不要修改自动生成的任何代码。

(19). 在【解决方案资源管理器】中，双击 Program.cs，观察 Mian 方法内的代码。

(20). 选择【文件】→【退出】命令，结束项目编辑。

(21). 找到 C:\CSharpExperiment，用鼠标右键单击该目录，从快捷菜单中选择【复制】命令，将该目录下的所有内容全部备份到分配给自己的网络映射驱动器空间中，或者直接备份到 U 盘中。注意：不要只备份某一个文件，应该将整个目录全部备份。

另外还要注意：以后再次打开这个项目前，仍需要先将该目录复制到本地硬盘中，不要在 U 盘或者网络映射驱动器中直接打开。

(22). 依次察看 C:\CSharpExperiment 目录下的各个子目录以及子目录下的文件，体会哪些是源代码文件，哪些是自动生成的文件。

实验步骤 2：创建一个简单的控制台应用程序项目

(1). 重新运行 VS2017，新建一个名为 SimpleConsoleApplication的控制台应用程序。

(2). 编写程序完成下列功能：

1) 从键盘接收一个字符串，例如“my friend”。

2) 输出相应的欢迎信息，例如“Welcome: my friend”。

提示：输入字母时，智能帮助会自动提示相应的信息，可以直接按回车键或者<Tab>键接受提示的信息。例如：输入到 conso 时智能帮助已经提示到 Console，直接按回车键即可。输入 f 时智能帮助已经提示到 for，直接按回车键，然后按<Tab>键，for 语句的整个结构就自动出来了，再依次按<Tab>键修改各参数的内容。不需要修改了，可以直接按回车键， 此时光标会直接转到循环体内部。

还有，如果将最后一个大括号删除，然后再重新添加上，系统会自动对整个代码按照嵌套层次进行统一的缩进处理，轻松解决了手工调整的麻烦。

其他编辑技巧请实验者自己总结。实现这一步功能的代码如下：

```
static void Main(string[] args)
    {
    Console.Write("请输入一个字符串：");
    string welcomeString = Console.ReadLine(); Console.WriteLine("Welcome:
{0}", welcomeString); Console.ReadLine();
    }
```

3) 在此基础上再增加一些语句，然后设置一些断点，体会利用断点调试程序的方法。

4) 结束项目调试，将该项目备份到分配给自己的网络空间或者 U 盘中。

实验步骤 3：创建一个简单的网站

(1). 重新运行 VS2017，新建一个网站，注意在弹出的窗体中，【位置】下拉列表中的选项应该选择为“文件系统”，【位置】的右边是要求将网站保存到的具体目录，在其中输入 C:\CSharpExperiment\SimpleWebSite，单击【确定】按钮。

(2). 修改<body>为<body style="text-align:center;">。

(3). 修改 为 。

注意：键入“width:760px;”后，按空格键，然后再键入“height”，键入过程中智能帮助就会提示相应的选项。

(4). 切换到设计视图，在div区域内输入一句话，例如“我的第一个网站”，然后在其下面添加一个Button控件，设置其【Width】属性为“70%”，【ID】属性为“buttonOK”，【Text】属性为“确定”。

(5). 用鼠标右键单击该标签，从快捷菜单中选择【样式】命令，将字体大小设为“20pt”，选择颜色为自定义的红色。

(6). 用鼠标拖动调整div区域的宽度，观察【确定】按钮的变化。

(7). 单击鼠标右键，从快捷菜单中选择【在浏览器中查看】命令，观察结果。

注意：第一次浏览时系统需要建立缓存，因此感觉速度比较慢，缓存以后就快了。

(8). 双击【确定】按钮，在代码隐藏类的 buttonOK_Click 事件中输入：

```
Response.Write("<script>alert('哈哈哈');</script>");
```

(9). 按<F5>键调试运行，观察弹出的警告对话框提示信息，然后直接单击【确定】按钮，

(10). 观察运行结果。

实验报告中要求回答的问题：

在 Windows 应用程序项目的基本操作中，你总结出了哪些操作技巧？还发现了哪些你认为有价值的操作技巧？

从 FormMain.Designer.cs 中的代码和 FormMain.cs 的设计窗体的对照比较中，你能得到什么结论？得到哪些启发？

实验二　C#基本编程方法

实验目的：

1) 练习 C#中变量声明和赋值的方法。
2) 练习类型转换的方法。
3) 练习分支语句的基本用法。
4) 练习循环语句的基本用法。

实验内容：

为银行个人存款客户提供一个“超级存款计算器”，以简单直观的操作界面为客户提供一个银行存款本息到期金额结算查询程序，以便客户决定选择那种存款方式。要求初始界面如右图所示。

图4.1 利率计算器

用户输入存款金额及相应信息后，单击【计算】按钮，程序能自动在【到期结算总额】中显示到期应得的本金和利息合计总金额。具体要求如下。

1) 存款金额不能低于 100 元，否则不进行计算并弹出对话框提示相应信息。
2) 计算方式提供按月算息、按季度算息和按年算息 3 种形式。
3) 按年算息是指每年计算一次应得的利息，并将应得的利息作为新存款添加到用户存款金额中。例如，存款人第一次存入金额 100 元，年利率为 2%，则：
   第一年的利息 x1：100×0.02 元，第一年结算余额 y1：100+x1
   第二年的利息 x2：y1×0.02 元，第二年结算余额 y2：y1+x2
   第三年的利息 x3：y2×0.02 元，第三年结算余额 y3：y2+x3
   ……
4) 按季度算息是指每季度计算一次应得的利息，并将应得的利息作为新存款添加到用户存款金额中。例如，存款人第一次存入金额 1000元，年利率为 2%，则第一个季度的利息为 1000×(0.02 ÷4) 元，第二个季度的利息为（1000+第一个季度的利息）×(0.02 ÷4) 元，依次类推。
5) 按月算息是指每月计算一次应得的利息，并将应得的利息作为附加存款添加到用户现有存款金额中。例如，存款人第一次存入金额 1000 元，年利率为 2%，则第一个月的利息为 1000×(0.02 ÷12) 元，第二个月的利息为（1000+第一个月的利息）×(0.02 ÷12) 元，依次类推。
6) 到期结算总金额要求输出结果四舍五入到小数点后两位。

实验要求：

1) 要求用 startAmount 表示初始存款金额。
2) 要求用 yearRate 表示年利率。

3) 要求用 years 表示年数。
4) 要求用 calculateFrequency 保存用户选择的计算方式，即“按月算息”、“按季度算息”和“按年算息”，当用户在【计算方式】中选择某个计算方式后，程序会根据选择结果 对 calculateFrequency 赋以相应的字符串值，如赋值为“按月算息”。
5) 要求用 rate 表示按选择的计算方式使用的利率。
6) 要求将计算出的结算总金额赋给 total 变量，并在只读的 textBoxTotal 中显示结果。
7) 要求程序中提供下列方法。

① Caculate 方 法

功能：计算到期结算金额。方法原型：private double Caculate(double startAmount, double rate, int count)，其中，startAmount 表示存款金额，rate 表示利率，count 表示叠加次数。返回值为到期金额。

② Convert StringToNumber方法

功能：将大于零的字符串转换为 32 位整数或者 64 位浮点数。并指明转换是否成功。方法原型（提供两种重载形式）：

private bool ConvertStringToNumber(string str, bool mustGreatThanZero, out int result)

private bool ConvertStringToNumber(string str, bool mustGreatThanZero, out double result)，其中，str 为被转换的字符串，must Great ThanZero 为是否有必须大于零的要求，result 为转换后的 32 位整数或者 64 位浮点数。

实验步骤提示：

(1). 创建一个名为 SuperCalculator 的 Windows 应用程序，修改 Form1.cs 为 FormMain.cs，然后完成图 4-1 的设计界面。

(2). 在 comboBoxCalculateFrequency 的【Items】属性中输入按月计算、按季度计算和按年计算三个选项。

(3). 通过窗体的 Shown 事件，让窗体界面显示时光标默认在存款金额文本框中闪烁。

```
private void FormMain_Shown(object sender, EventArgs e)
{
    textBoxStartAmount.Focus();
}
```

(4). 想办法用一个事件，保证修改输入信息中任何一个内容时，到期结算金额中都不能显示值，而只有单击了【计算】按钮才显示结算结果。

```
private void groupBox1_Enter(object sender, EventArgs e)
{
    //保证修改任一输入值时，不显示计算结果
    textBoxTotal.Clear();
}
```

(5). 在【计算】按钮的 Click 事件中，先判断输入信息是否符合要求，然后根据利息计算方式计算到期结算金额。实现代码中可以利用【SelectedItem】属性

判断选择的值，利用【SelectedIndex】属性判断是否选择了提供的选项。例如：

```
if (comboBoxCalculateFrequency.SelectedIndex == -1)
{
    MessageBox.Show("请选择提供的利息计算方式");
    return;
}
```

即如果没有选择任一个选项，SelectedIndex 属性返回-1。

(6). 如果希望通过代码设置程序开始运行时窗体的起始位置在屏幕中间，可以在 FormMain 的构造函数中添加代码。

```
public FormMain()
{
    InitializeComponent();
    this.StartPosition = FormStartPosition.CenterScreen;
}
```

这里有一个输入技巧：输入完 this.StartPosition 后面的“=”后，直接按空格键，系统会自动出现 FormStartPosition，然后直接键入“.”，再选择希望的枚举值后按回车键。凡是以枚举类型出现的都可以采用这种办法提高键入代码的速度。再举一个例子，输入：

```
MessageBox.Show("aa","bb",MessageBoxButtons.YesNo,MessageBoxIcon.Asterisk);
```

输入技巧为：输入"bb"后面的“,”后，按空格键，出现 MessageBoxButtons，选择 YesNo，回车，然后键入逗号，再按空格键，出现 MessageBoxIcon，选择 Asterisk，回车。最后输入其他内容。如果按空格后没有出现希望的选项，可以先用上下箭头键选择合适的重载方法，然后再按空格键。另外，在已有代码的前面增加内容时，如果希望输入时就有智能提示，当输入一个字母后还没有提示，可以先输入一个空格，然后在空格的前面添加内容就有提示了。

使用已经声明过的变量、对象名，或者输入类名时，或者键入“.”后，如果应该有对应的智能提示但是却没有看到智能提示时，说明前面的输入肯定有问题，这时不要再继续输入内容了，因为再输入的内容肯定也是错的，而应该思考一下问题在哪，这样才能有效地提高编写和调试程序的效率。

(7). 将重复使用的功能，用单独的方法实现，并使用 XML 注释方式给方法功能以及提供的参数添加对应的注释。使用 XML 注释方式时，注意一定要先定义好方法及参数，然后再在其上面按“///”添加注释。

实验报告中要求回答的问题

(1). 画出包含输入和输出数据的程序运行界面示例。

(2). 运行设计的程序，在实验报告中列出计算后的表 4-1 的内容，并说明程序计算结果和手工计算结果是否符合。

表 4-1 部分存款计算结果

初始金额（元）	利率（%）	年数	计算方式	到期结算总额（元）
1000	2	5	按月计息	
			按季度计息	

			按年计息	
3500	3.3	7	按月计息	
			按季度计息	
			按年计息	
5000	6．25	10	按月计息	
			按季度计息	
			按年计息	

(3). 写出实验中遇到的问题及解决方法。

(4). 所有功能实现后，与参考解答对照，写出自己设计的程序与参考解答相比有什么优缺点。

实验三 面向对象的编程基础

实验目的;

(1). 练习如何创建类和对象。

(2). 练习如何为定义的类编写相应的方法。

(3). 练习如何通过属性访问对象中的数据。

(4). 练习如何创建基类及其派生类。

实验内容:

在个人银行存款业务中，不同银行规定有不同的帐户类型，例如整存整取、零存整取、存本取息、通知存款、定额定期、定活两便和活期储蓄等。本实验不处理这么复杂的内容， 而是假定只提供两种帐户，一种是活期存款帐户，另一种是定期存款帐户。

实验要求为个人银行存款帐户定义两个类，一个是活期存款帐户 CheckingCustom 类，另一个是定期存款帐户 FixedCustom 类。要求用户操作界面如图 4-2 所示。

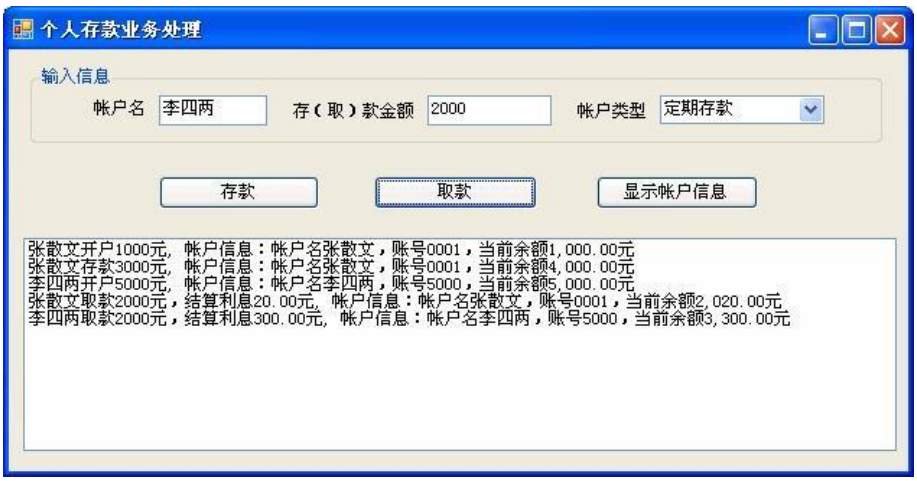


图4-2 存款业务处理界面示例

为了简化处理过程。假定实验中的“活期存款”和“定期存款”业务规定及功能实现要求如下。

1) 不论是活期存款帐户还是定期存款帐户，都可以随时存款和取款，而且规定一个人最多只能有一个活期帐户和一个定期帐户。创建活期帐户时，必须提供帐户名、帐户号和开户金额，业务处理均以帐户名为关键字。

2) 活期存款帐户号的范围必须为 0001～4999（包括 0001 和 4999），取款时，不论存款时间有多长，一律按 0.5%计算利息。

3) 定期存款帐户号的范围必须为 5000～9999（包括 5000 和 9999），取款时，不论存款时间有多长，一律按下列方法计算利息：当存款余额大于 500 时利息为 6%，否则利息为3%。

4) 每次取款之前，都要先根据当前余额结算一次利息，并将利息附加到余额中，然后再从中取出指定的款数。向现有帐户追加存款时，不进行结算。

5) 要允许用户随时查询所有帐户的信息。

6) 设计的程序要易于扩充，即需要增加存款业务类型时要能够利用已经实现的功能， 通过尽量少的代码快速实现，不要全部从头开始设计。

实验步骤提示

由于要求程序易于扩充，因此需要找出各种存款业务类型共有的功能，然后用一个基类实现，其他类在此基础上完成专有的功能即可。由于所有类中都应该包含存款、取款等操作， 而且所有类都必须有帐户名、帐户号和帐户余额三项，因此可以先定义一个名为 Custom 的基类，在基类中保存帐户名、帐户号和帐户余额，并完成所有类都有的存款、取款功能。然后让活期存款业务和定期存款业务继承自这个基类，再分别在各自的类中完成专有的功能。

具体编写步骤可以按照下面的提示进行。

(1). 创建一个名为 BankCustoms 的 Windows 应用程序项目，重命名 Form1.cs 为FormMain.cs，然后在此窗体上完成个人存款业务处理的设计界面。添加一个类文件 Custom.cs，处理活期存款和定期存款共有的业务。在 Custom 类中完成下列功能。

① 声明三个私有的成员变量保存帐户对应的信息，分别如下。

accountName：帐户名。

accountNumber：帐户号。

accountBalance：帐户余额。

然后给这三个私有成员定义相应的属性，分别命名为 AccountName、AccountNumber 和AccountBalance，并处理对应的访问权限。

② 编写一个公共的 Deposit 方法，向帐户中添加存款。

③ 编写一个公共的 Withdraw 方法，从帐户中取款。

(2). 向项目中添加一个名为 CheckingCustom.cs 类文件，处理活期存款业务，让其继承自 Custom 类，在 CheckingCustom 类中完成下列功能。

① 定义一个静态变量 newAccountNumber，提供准备产生的活期存款帐号，初值为 0001。注意每使用一次该帐号，其值都要自动加 1。

② 分别提供可以让外部访问的属性，包括帐户名、帐户号、余额和利率。

③ 提供一个带参数的构造函数，在构造函数中接收指定的帐户名和开户金额，并利用newAccountNumber 产生一个在活期存款规定帐户范围内的合法的帐户号，然后设置对应的属性。

④ 重写基类的 Withdraw 方法，使之符合活期存款业务的要求。

(3). 添加一个名为 FixedCustom.cs 的类文件，处理定期存款业务，让其继承自 Custom类，在 FixedCustom 类中完成除利息计算方式不同外、其他情况与活期存款业务相似的功能。

(4). 在存款业务处理界面的代码文件中完成下列功能。

① 分别定义保存活期存款和定期存款的 SortedList 泛型列表对象，使用 SortedList 的目的是为了利用键/值对进行处理。实现代码可以参看源程序。

② 分别实现存款、取款和显示帐户信息功能。

实验报告中要求回答的问题

(1). 画出有代表性的包含存款、取款和显示帐户信息的程序运行界面，运行界面可以使用多个图表示。

(2). 写出自己设计的代码与参考解答相比有哪些不同点。

(3). 写出实验中遇到的问题及解决方法。如果将帐户信息保存到数据库中，简要回答（不需要写实现代码，只要求提供设计思路）应该怎样实现存取款业务？

实验四 界面设计与文件存取

实验目的：

(1). 练习界面功能的设计方法。

(2). 练习文本文件的存取方法。

(3). 练习对话框的使用方法。

(4). 练习菜单的设计方法。

(5). 练习工具条和状态条的设计方法。

(6). 练习一个窗体调用另外一个窗体以及窗体间参数传递的方法。

实验内容：

重新设计和处理与个人存取款相关的业务，假定处理业务仅有活期存款一种，而且业务处理规定与实验三的活期存款业务相同。要求功能界面中包括菜单条、工具条和状态条，如图 4-3 所示，并要求在文本文件中保存帐户的相关信息。

图4-3 存款业务处理主界面

具体功能规定如下：

1) 在主菜单的【文件】子菜单下，有【打开】、【保存】命令。

【打开】命令负责从文件 bankdata.txt 中将所有存款人的信息读到内存中。要求程序开始运行后，选择【打开】命令后才将 bankdata.txt 的内

容读到内存。

【保存】命令负责将所有存款人的信息保存到文件 bankdata.txt 中。

在主菜单的【操作】子菜单下，有【存款】、【取款】和【显示所有帐户信息】命令。

【存款】命令负责处理存款人开户或者在其已有帐户上添加新的存款。

【取款】命令负责处理存款人在其已有帐户上取款。

【显示所有帐户信息】命令负责将内存中所有存款人的信息显示出来。

2) 工具条（或者叫快捷工具栏）内有三项：【打开文件】、【保存文件】和【显示所有帐户信息】命令。
3) 状态条（最下面一行）只要求显示【打开】、【保存】【存款】、【取款】和【显示所有帐户信息】5 项命令对应的功能解释，当操作某一个功能时，在状态条内显示该功能对应的详细解释信息。
4) 用户执行【显示所有帐户信息】命令时，要求弹出另外一个 Windows 窗体，在新窗体内将所有帐户信息显示出来。
5) 用户执行【保存】命令时，要求弹出一个对话框，询问是否将所有信息保存到文件中，如果用户选择“是”，则保存，否则不保存。

实验步骤提示

本实验可以利用实验三已经完成的功能，复制其中的部分代码，以加快设计速度。具体设计可以按照下面的提示进行。

(1). 创建一个名为 MenuBankCustoms 的 Windows 应用程序项目。重命名 Form1.cs 为FormMain.cs，然后在此窗体上设计个人存款业务处理的功能界面。

(2). 向项目中添加一个类文件 Account.cs，提供以下属性。AccountName：存款人姓名，AccountNumber：存款人帐号， AccountBalance：帐户余额。同时在该类中提供下面的方法。Deposit 方法：负责实现存款功能。Withdraw 方法：负责实现取款功能。

(3). 向项目中添加一个类文件 Customers.cs，用于处理所有帐户的信息。类中提供下面的方法。

Load 方法：负责从文件 bankdata.txt 中将所有存款人的信息读入到 Account 类型的数组中，并返回该数组。

Save 方法：负责将所有存款人的信息保存到 bankdata.txt 文件中。该方法返回一个布尔型的值，表明保存是否成功。

CreateAccount 方法：负责创建新帐户，该方法返回新建的帐户。

(4). 向项目中添加一个 Windows 窗体文件 FormShowAccountMessage.cs，在该窗体中显示所有帐户信息。

实验报告中要求回答的问题

(1). 写出你认为有必要解释的关键步骤或代码。

(2). 写出实验中遇到的问题及解决方法。

(3). 如果通过 Internet 连网，即可以在任一地点存款和取款，简要回答需要考虑哪些问题？

实验五　ADO.NET 与数据操作

实验目的：

(1).　练习 SQL Server 数据库的创建与使用方法。
(2).　练习存储过程的创建与调用方法。
(3).　练习简单的统计查询方法。
(4).　练习绑定DataGridView 到数据源的方法。
(5).　练习绑定其他控件到数据源的方法。
(6).　练习数据库中图片的显示和存取方法。

实验内容：

设计一个 Windows 应用程序项目，实现简单的银行工作人员基本信息管理，以及利用存储过程实现统计和查询功能。具体要求如下。

1) 程序开始运行出现图 4-4 所示的主功能以及登录界面，系统功能仅有两种，一种是基本信息管理，另一种是统计男女人数。已知所有操作员和对应的密码已经保存在数据库中的某个表中，只有操作员密码正确时才允许使用系统功能。
2) 当操作员选中【基本信息管理】功能时，弹出如图 4-5 所示的窗体，可以对基本信息进行记录导航、添加、删除、修改、保存，以及对照片的导入和移除处理。

图4-4　主功能及登录设计界面

图4-5　基本信息管理子界面

3) 当操作员选中【统计男女人数】功能时，直接弹出一个对话框显示统计结果。

实验要求：

1) 要求使用数据绑定将【操作员】下拉列表框和数据库表的对应字段绑定，使用存储过程查询比较密码是否正确。
2) 弹出基本信息管理界面时，不能显示登录界面，退出基本信息管理后，登录界面重新显示出来。
3) 在基本信息管理操作中，无论操作员怎样操作，都不允许出现系统默认的错误，即应该捕获所有可能的异常。
4) 单击窗体下方 DataGridView 控件的任一单元格，上方的各文本框以及照片都要能根据单击的行及时变动显示的内容。
5) 使用另外一个存储过程统计男女人数。

实验步骤提示

(1). 创建一个名为 BankEmployee 的 Windows 应用程序项目。重命名 Form1.cs 为FormMain.cs，然后在此窗体上设计如图 4-4 所示的设计界面。

(2). 在项目中创建一个名为 Employee.mdf 的数据库，新建两个表和 3 个存储过程。

BasicTable：该表用于保存职员基本信息。Operator：该表用于保存操作员名称及密码。

SelectAllUserNameFromOperator：该存储过程用于填充下拉列表框。

SelectOperator：该存储过程用于判断选择的操作员及密码。

SimpleQuery：该存储过程用于统计男女人数。

(3). 完成 FormMain.cs 窗体的【进入】、【退出】以及【统计男女人数】功能。

(4). 向项目中添加一个新窗体 FormEmployee.cs，通过简单的拖放操作设计图 4-5 所示的设计界面，同时将各控件与数据库的对应字段绑定，然后在导航条的下面，添加一个居中显示的 Label 控件，用于提示异常以及操作提示等信息，最后修改 Program.cs 让其直接运行该窗体，以便快速调试该窗体提供的功能。

(5). 将 FormEmployee.cs 窗体导航条中的【第一个记录】、【上一个记录】、【下一个记录】、【最后一个记录】、【新添】和【删除】按钮的对应属性全部改为“（无）”，例 如【第一个记录】对应【MoveFirstItem】属性，【下一个记录】对应【MoveNextItem】属性等，以便自己用代码实现对应功能。

注意：这一步的目的主要是不让系统自动实现对应的功能，因为自动实现的功能无法捕获出现的异常；另一个目的是为了让实验者清楚导航条上的按钮功能到底是如何实现的。

(6). 实现导航条上对应按钮的功能。

(7). 实现照片的【导入】和【移除】功能。

(8). 完成从 FormMain.cs 调用 FormEmployee.cs 的功能，并修改 Program.cs 中的代码让其首先调用 FormMain 窗体。

(9).

实验报告中要求回答的问题

(1). 写出你认为有必要解释的关键步骤和代码。

(2). 写出实验中遇到的问题及解决方法。

(3). 总结数据库相关编程中应该注意哪些方面。

实验六　ASP.NET 网页设计

实验目的：

(1).　练习网页的基本布局及设计方法。

(2).　练习表格的基本用法。

(3).　练习 div 的基本用法。

(4).　练习层叠式样式表的基本用法。

实验内容：

设计一个简单的网页，浏览效果如图 4-6 所示。要求使用样式表控制网页中字体以及链接的颜色。

图4-6 网页浏览效果

实验要求：

假设网页中使用的图片已经设计好并保存在 picture 文件夹下，要求实验者先按照自己的思路考虑如何实现如图 4-6 所示的界面，然后按照实验步骤完成图 4-6 所示的网页效果，在完成过程中认真体会每一步的意图，以及相关的设计技巧。

实验步骤

(1). 新建一个位置为 C:\MyWeb\MyBankSite 的网站，在设计窗体下选择快捷工具栏中的目标架构下拉列表框，将 Default.aspx 网页的目标架构改为“Internet Explorer 6.0”，目的是为了在网页中使用 marquee。

(2). 在已有的 div 中添加 id 及样式，使其变为下面的内容。

。

(3). 在 div0 的下面添加一个 id 为“div1”的 div，将其变为下面的内容。

。

然后切换到设计视图，调整 div1 为适当大小。

(4). 在【解决方案资源管理器】中，用鼠标右键单击项目名，从快捷菜单中选择【新建文件夹】命令，创建一个名为“image”的文件夹，用于保存网站所用的所有图片。创建后， 用鼠标右键单击该文件夹，将 picture 文件夹中的所有图片一次性全部添加到该文件夹下。

(5). 在 Default.aspx 的【设计】视图下，将 tu1.jpg 从 image 文件夹中拖放到 div1 内。然后在该图的上面单击鼠标右键，从快捷菜单中选择【样式】→【位置】命令，设置【宽度 】为 100%，【高度】为 130px。

(6). 选中 div1，拖动其右下角再次调整 div1 的大小，观察图片宽度的变化，体会使用百分比和使用 px 为单位定义宽度的区别。

(7). 切换到【源】视图，在 tu1.jpg 图片的下方添加下面的代码。

<marquee style="color: #ff0033;">哈哈，祝贺我的网站设计成功</marquee>

(8). 单击鼠标右键，从快捷菜单中选择【在浏览器中查看】命令，观察效果，然后关闭浏览界面。

(9). 切换到【设计】视图，选择【布局】→【插入表】命令，插入一个 3 行 2 列、宽度为 100%的表。

(10). 同时选择表中第 1 行的两列，单击鼠标右键，从快捷菜单中选择【合并单元格】命令，然后选择合并后的单元格，单击鼠标右键，从快捷菜单中选择【样式】命令，在弹出的窗口中选择【背景】选项卡，将【背景色】改为“#ffcc33”，再选择【文本】选项卡，将 水平对齐方式设置为居中，单击【确定】按钮。

用同样的办法将第 3 行合并为一个单元格、居中，并设置【背景色】为“#dcdcdc”。

(11). 在【解决方案资源管理器】中，新建 4 个网页：MyWeb1.aspx、MyWeb2.aspx、MyWeb3.aspx 和 MyWeb4.aspx。

(12). 切换到 Default.aspx 的【设计】视图，在表中第 1 行输入：“政策法规 汇率计算 调查统计 网上银行”，然后选中“政策法规”，单击快捷工具栏中的【转换为超链接】图标，将其链接到 MyWeb1.aspx。按照同样的办法，将另外 3 项分别链接到 MyWeb2.aspx、MyWeb3.aspx 和 MyWeb4.aspx。

切换到【源】视图，观察生成的 HTML 代码。

(13). 重新切换到【设计】视图，选中表格第 2 行第 1 列单元格，单击鼠标右键，从快捷菜单中选择【样式】命令，修改【文本】选项卡中的【水平】属性为“右对齐”，【宽度 】属性为“50%”。

(14). 从【工具箱】中选择 HTML 的 Div，将其拖放到第 2 行第 1 列单元格内，设置其【Id】属性为 div3，样式为：背景图使用 image/bg1.gif，不平铺；文本水平对齐方式为左对齐；行间距为 30pt；上面和下面均留出 15px 的空白。

(15). 从【解决方案资源管理器】中向 div3 内拖放一个 tu2.jpg 图片，然后从【工具箱】内向图片的下面拖放 3 个 HyperLink 控件，分别设置其【Text】属性和【Navigateurl】属性。由于我们在这个实验中关注的不是实际链接的网页，所以链接时随便选择哪个网页均可。

(16). 按同样的办法设置表格第 2 行右边单元格的内容。只不过单元格的文本对齐方式选择左对齐。

(17). 在表格第 3 行内输入一些内容。然后在浏览器中查看网页效果。

(18). 在【解决方案资源管理器】中，新建一个名为 BodyStyle.css 的样式表文件。在样式表文件的 Body 后面的大括号内单击鼠标右键，从快捷菜单中选择【生成样式】命令，将字体颜色设置为 Web 调色板中的某种颜色。按<Ctrl>+<S>组合键保存该样式表。

(19). 在 Default.aspx 的【源】视图中，先看看 head 块内有什么代码，然后按<F4>键弹出【属性】窗口，在其最上面的下拉列表框中选中“Document”选项，设置其【StyleSheet】属性为“BodyStyle.css”，再看看 head 块内自动添加了什么代码。

切换到【设计】视图，观察表格最下面一行内容是否为样式表中设置的颜色。

(20). 在样式表文件内的 Body 块外面，单击鼠标右键，从快捷菜单中选择【添加样式规则】命令，选择【元素】下拉列表框中的“A”，单击【确定】按钮。然后在 A 块内单击鼠 标右键，从快捷菜单中选择【生成样式】命令，选择 Web 调色板中的某种颜色。设置颜色后， 按<Ctrl>+<S>组合键保存样式表。再切换到 Default.aspx 的【设计】视图，观察所有链接颜色是否发生了变化。

按照同样的方法，在样式表文件中将所有链接样式设置为不带下划线。

(21). 在样式表文件内的最下面，单击鼠标右键，从快捷菜单中选择【添加样式规则】命令，在弹出的窗体中选择【类名】，在其文本框中输入“mydiv”，然后单击其右边的【>】 将其添加到样式规则层次结构中；再选择【元素】下拉列表框中的“A”，单击【>】将其添 加到样式规则层次结构中，然后单击【确定】按钮。

(22). 在刚生成的.mydiv A 块内，单击鼠标右键，从快捷菜单中选择【生成样式】命令，

(23). 在该块内设置一种颜色，保存样式表。

(24). 切换到 Default.aspx 的【设计】视图，选中 div3，设置其【Class】属性为“mydiv”，观察 div3 内的链接颜色是否为样式表 mydiv 中设置的颜色，并观察 div3 以外的其他链接颜色是否有变化。

用同样的办法，在样式表文件中将 mydiv 的链接样式设置为带下划线。

(25). 在浏览器中查看网页设计效果。

实验报告中要求回答的问题

(1). 写出你认为有必要解释的关键步骤和代码。
(2). 写出实验中遇到的问题及解决方法。
(3). 你从这个实验中得到了哪些启发？

实验七 ASP.NET 网站开发

实验目的：

(1). 练习母版页的使用方法。
(2). 练习在代码中引用母版页的控件的方法。
(3). 练习在 Web 窗体应用程序中访问 SQL Server 数据库的方法。
(4). 练习GridView 中格式化显示的设置方法。
(5). 练习处理 Web 窗体事件的方法。
(6). 练习网页间数据传递的方法。
(7). 练习一个窗体引用另一窗体中的属性和控件的方法。

实验内容：

开发一个针对存款人的活期存款查询网站，操作者可以通过 Internet 浏览器随时查看所有存款人的活期存款帐户信息。

实验要求：

1) 实验中要求有一个母版页，三个引用母版页的网页，具体名称及功能如下。

MyBank.master：母版页，设计界面如图 4-7 所示。

CheckingQuery.aspx：显示所有帐户的活期存款信息，运行效果如图 4-8 所示。QueryResult1.aspx：单击每行的【帐户号】超链接后显示的信息，运行效果如图 4-9 所示。QueryResult2.aspx：单击每行的【选择】链接或者【存款人】链接后显示的结果，运行效果如图 4-10 所示。

图4-7 母版页的设计界面

图4-8 CheckingQuery.aspx的运行界面

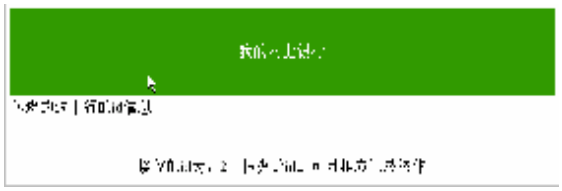


图4-9 QueryResult1.aspx的运行界面

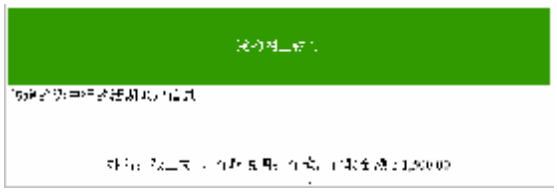


图4-10 QueryResult2.aspx的运行界面

2) 活期存款信息保存在 BankDatabase.mdf 的 CheckingAccount 表中，表中字

段有【帐户名】、【帐户号】、【存取说明】、【存取金额】、【余额】和【id】，其中【id】字使用自动编号。

3) 显示如图 4-8 所示的每一笔存取款信息后，要求用 3 种方法分别实现每行的【选择】、【帐户名】和【存款人】的目标链接，单击 3 种链接的任何一个链接都能转到另一个网页，并在另一个网页中将单击的行的相关信息显示出来。

4) 每一笔存取金额和余额均使用图 4-8 所示的货币形式输出。

实验步骤

(1) 创建一个位置为 C:\MyWeb\MyNewBankSite 的网站，然后删除自动生成的 Default.aspx。

(2) 在【解决方案资源管理器】中，用鼠标右键单击项目名，从快捷菜单中选择【添加新项】→【母版页】命令，输入名称为 MyBank.master，单击【添加】按钮。

(3) 在 MyBank.master 的【源】视图中，将光标定位在 form 下一行的 内，按<F4> 键弹出【属性】窗口，单击【Style】属性右边的“…”，在弹出的样式生成器窗口中选择【文　本】选项卡，设置【水平】属性为“居中”，单击【确定】按钮。

(4) 切换到 MyBank.master 的【设计】视图，在 ContentPlaceHolder1 的左边单击鼠标，然后选择【布局】→【插入表】命令，插入一个 3 行 1 列、宽度为 720px 的表。

(5) 将 ContentPlaceHolder1 拖放到表的第 3 行第 1 列内。

(6) 单击表的第 1 行第 1 列，确认选中的是<td>，然后单击鼠标右键，从快捷菜单中选择【样式】命令，将该单元格的前景色设置为白色，背景色设置为绿色，水平、垂直对齐方式均设为居中，【位置】中的宽度设为“100%”，单击【确定】按钮。

(7) 在表的第 1 行第 1 列输入“我的网上银行”，然后拖动表中第 1 行下面的线，将第1 行的高度拖至 100px。

(8) 单击表的第 2 行第 1 列，确认选中的是<td>，然后单击鼠标右键，选择【样式】命令，将第 2 行第 1 列的宽度设为 100%，水平对齐方式为左对齐，单击【确定】按钮，然后从【工具箱】中拖放一个 Label 控件到第 2 行内，将其【ID】属性和【Text】属性均设置为“labelAccount”，【Width】属性设置为“80%”。

(9) 切换到【源】视图，观察生成的代码中表的第 3 行第 1 列宽度值，再切换为【设计】视图，观察 ContentPlaceHolder1 的宽度有什么变化。

(10) 将表的第 3 行第 1 列宽度设置为“100%”。完成母版页的设计。

(11) 用鼠标右键单击项目名，从快捷菜单中选择【添加新项】，然后选择【SQL 数据库】模板，在 App_Data 文件夹下添加一个数据库文件 BankDatabase.mdf。双击该数据库名，在【服务器资源管理器】中添加一个名为 CheckingAccount 的表，在表结构中添加相关字段，并填入数据。

注意：定义表结构时不要将 id 放在最上面，应该放在最下面，而且修改表数据时不要将光标移到 id 列内，否则退出修改数据前将无法继续输入汉字（这是 50727 版本的 bug）。另外还要注意不要忘记设置主键。

(12) 向【解决方案资源管理器】中添加一个新的 Web 窗体

CheckingQuery.aspx，注意添加时要同时选中【选择母版页】和【将代码放在单独的文件中】复选框。

(13) 切换到【设计】视图，从【工具箱】中向 Content1 内拖放一个 SqlDataSource 控件和一个 GridView 控件，利用向导配置 SqlDataSource1，在生成查询语句时选中包括 id 在内的所有字段，然后设置GridView1 的数据源为 SqlDataSource1。

(14) 设置 GridView1 的【Width】属性为“100%”。

(15) 单击 GridView1 右上方的三角符号，单击【自动套用格式】按钮，然后选择一种显示格式。

(16) 再次单击 GridView1 右上方的三角符号，单击【编辑列】按钮，在弹出的窗口中 删除【选中的字段】内的【帐户名】和【id】，注意不要选择【自动生成字段】复选框，然后从【可用字段】选项内向【选定的字段】中添加一个【HyperLinkField】字段，并将其移到最上面，设置下列属性。

【DataNavigateUrlFields】属性：id

【DataNavigateUrlFormatString】属性：QueryResult1.aspx?id={0}

【DataTextField】属性：帐户名

【HeadText】属性：帐户名

【Target】属性：_blank

(17) 向【解决方案资源管理器】中添加一个新的 Web 窗体 QueryResult1.aspx，注意添加时要同时选中【选择母版页】和【将代码放在单独的文件中】复选框，然后从【工具箱】向 Content1 内拖放一个 Label 控件，设置其【ID】属性为“labelResult”，并在 Page_Load 事件中加入下面的代码。

```
protected void Page_Load(object sender, EventArgs e)
{
Label myLabel = (Label)this.Master.FindControl("labelAccount");
myLabel.Text = "传递的选中行的 id 信息";
labelResult.Text = string.Format("接收的 id 为：{0}，根据此 id 即可对相应记录操作", Request.QueryString[0]);
}
```

这一步演示了如何引用母版页中的控件，即利用 Master 的 FindControl 方法获取引用的对象，然后再对该对象进行操作。

同时，这一步也实现了使用 Request.QueryString 传递参数的方法。但是这种方法仅适用于比较简单的情况，当需要传递的参数较多以及传递的参数内容是汉字等情况时，一般不使用这种方法。

(18) 切换到 CheckingQuery.aspx 的【源】视图，按<F5>调试运行，分别单击不同行的帐户名链接，观察显示的结果。

(19) 切换到 CheckingQuery.aspx 的【设计】视图，单击 GridView1 右上方的三角符号， 选择【启用选定内容】复选框，使其自动在最左边添加一个【选择】列。

(20) 设置 GridView1 的【DataKeyNames】属性为“帐户名”，以便在代码中使用SelectedValue 属性获取单击的对应行的帐户名。

(21) 在 GridView1 的代码方式下，分别定义 UserName 属性、Explian 属性和 Money 属性，以便被另一个网页引用。

```
private string userName;
public string UserName
```

```
{
get
{
return userName;
}}
private string explain; public string Explian
{
get{
return explain;
}}
private string money;
public string Money
{
get
{
return money;
}
```

注意：这里定义的属性只有 get，虽然也可以使用 set，但是由于网页显示方式的特殊性，使用 set 没有意义。

(22) 在 Page_load 事件中添加下面的代码。

```
protected void Page_Load(object sender, EventArgs e)
{
Label myLabel = (Label)this.Master.FindControl("labelAccount"); if (myLabel != null)
{
myLabel.Text = "活期帐户查询";
}}
```

(23) 添加 GridView1 的 SelectedIndexChanged 事件，在其事件中添加下面的代码。

```
protected void GridView1_SelectedIndexChanged(object sender, EventArgs e)
{
userName = GridView1.SelectedValue.ToString(); // 帐户名 explain = GridView1.SelectedRow.Cells[3].Text; // 存取说明 money = GridView1.SelectedRow.Cells[4].Text; // 存取金额
Server.Transfer("QueryResult2.aspx");
}
```

注意，要让同一个应用程序的另一个网页引用本网页的属性，不能调用 Response.Redirect 方法转到另一个网页，应该调用 Server.Transfer 方法。

(24) 向【解决方案资源管理器】中添加一个新的 Web 窗体 QueryResult2.aspx，注意添加时要同时选中【选择母版页】和【将代码放在单独的文件中】复选框，然后在【设计】视图下向 Content1 内拖放一个 Label 控件，设置其【ID】属性为“labelResult”。

(25) 在 QueryResult2.aspx 的【源】视图中，在第 2 行添加下面的代码。

```
<%@ PreviousPageType VirtualPath="~/CheckingQuery.aspx" %>
```

使用这一行代码的目的是为了指明谁调用了本网页，以便在本网页的后台代码中

使用PreviousPage 属性获取前一个网页中的内容。

顺便说明一下，如果在母版页中定义了属性，当希望在此网页中获取母版页的属性时，可以添加下面的代码。

```
<%@ MasterType virtualPath="~/MyBank.master"%>
```

由于本实验没有在母版页中定义属性，所以【源】视图中是否有 MasterType 这一行对输出结果没有影响。

(26) 在 QueryResult2.aspx 的 Page_Load 事件中添加下面的代码。

```
protected void Page_Load(object sender, EventArgs e)
{
Label myLabel = (Label)this.Master.FindControl("labelAccount");
myLabel.Text = "传递的选中行的活期帐户信息";
if (this.IsPostBack == false)
{
labelResult.Text = string.Format("姓名：{0}，存取说明：{1}，存取金额{2}",
this.PreviousPage.UserName,
this.PreviousPage.Explian, this.PreviousPage.Money);
}}
```

也可以通过调用 PreviousPage.FindControl 方法获取前一个网页中的控件值，在这一步骤中，没有演示这个功能。

(27) 切换到 CheckingQuery.aspx 的【设计】视图，单击鼠标右键，从快捷菜单中选择【在浏览器中查看】命令，然后分别单击每一行的【选择】链接，观察数据传递的效果。

至此，实现了传递参数的第二种方法。

(28) 在 CheckingQuery.aspx 的【设计】视图下，单击 GridView1 右上角的三角符号，然后单击【编辑列】按钮，从【可用字段】列表中向【选定的字段】列表中添加一个【ButtonField】字段，然后设置下列属性。

【DataTextField】属性：帐户名

【ButtonType】属性：Link

【HeaderText】属性：存款人

【CommandName】属性：Select

注意：【CommandName】属性的“Select”有固定的含义，使用“Select”的目的是为了让其选择当前记录，并触发 GridView1 的 SelectedIndexChanged 事件，与【CommandName】属性相关的有固定含义的字符串还有：Cancel、Delete、Edit、Page、Sort 和 Update。按<F5>键编译并运行，观察参数传递的效果。至此，实现了传递参数的三种方法。

实验报告中要求回答的问题

(1) 写出你认为有必要解释的关键步骤和代码。

(2) 写出实验中遇到的问题及解决方法。

(3) 你从这个实验中得到了哪些启发？

实验八　网络呼叫应答提醒系统

实验目的：

（1）　练习 UDP 应用编程的方法。

（2）　练习动画窗体的设计方法。

（3）　练习自动显示和隐藏窗体的方法。

（4）　通过实验步骤学习系统托盘的设计方法。

实验内容：

1) 开发一个简单的基于 UDP 的网络呼叫应答及时提醒系统。程序运行时，系统自动弹出一个逐渐变大的动画窗体，同时在任务栏内显示一个托盘图标。窗体变化到最大程度后的界面效果如图 4-13 所示。

2) 用户关闭窗体后，窗体消失，但并不结束程序运行。关闭窗体后，如果用户双击任务栏内的托盘图标，系统仍然会自动弹出与启动程序时效果相同的逐渐变大的动画窗体。

3) 如果用户用鼠标右键单击任务栏内的托盘图标，系统会自动弹出一个快捷菜单，菜单中提供【呼叫对方】和【退出程序】两个选项，托盘和快捷菜单的运行效果如图 4-14 所示。

图 4-13 呼 叫 窗体

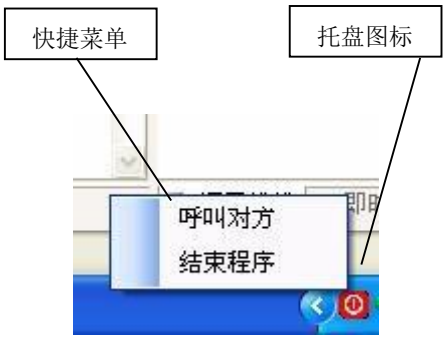


图4-14 屏幕右下方显示的托盘图标

4) 系统弹出快捷菜单后，如果用户选择【呼叫对方】命令，程序自动弹出动画窗体，并向接收方发出自己的 IP 地址和呼叫信息；如果用户选择【退出程序】命令，则结束整个系统的运行。

5) 程序运行时，系统自动创建一个线程在端口 8001 监听网络呼叫信息，当接收到某人呼叫时，弹出一个对话框，提示呼叫方发送的信息。

实验要求：

先运行参考解答的程序，观察运行效果，然后按照实验步骤独立完成实验内容。注意一定不要采用直接复制参考解答中的源代码的办法完成实验，因为那样就失去了实验的意义。

实验步骤提示

（1）　创建一个名为 MessageAwake 的 Windows 应用程序项目，在【解决方案资源管理器】中，将 Form1.cs 换名为 FormMain.cs，然后设计图 4-13 所示的界面。

（2）　从【工具箱】中向设计窗体拖放一个 Timer 控件，以便控制窗体动画效果。

（3）　从【工具箱】中向设计窗体拖放一个 ContextMenuStrip 控件，设计如图

4-14 所示的快捷菜单。

（4） 在【解决方案资源管理器】中，用鼠标右键单击项目名，选择【添加】→【新建项】命令，在弹出的窗口中选择“图标文件”模板，输入文件名为“demo.ico”，然后单击【添加】按钮，在图标设计窗口中设计一个托盘图标。

（5） 在【解决方案资源管理器】中，用鼠标右键单击 demo.ico 文件，从快捷菜单中选择【属性】命令，在属性窗口中，将【复制到输出目录】属性设置为“始终复制”，【生成操作】属性设置为“无”。

（6） 分析下面的源代码，完成实验要求的功能。

```
using System;
using System.Collections.Generic;
using System.ComponentModel; using System.Data;
using System.Drawing; using System.Text;
using System.Windows.Forms;
//添加的命名空间引用using System.Net;
using System.Net.Sockets; using System.Threading; namespace MessageAwake
{
public partial class FormMain : Form
{
private System.Windows.Forms.NotifyIcon myNotifyIcon; private bool isExit
= false;
private int formHeight;
//使用的接收端口号private int port = 8001;
private UdpClient udpClient; public FormMain()
{
InitializeComponent(); formHeight = 0; this.Height = formHeight;
timer1.Enabled = true;
}
private void Form1_Load(object sender, EventArgs e)
{
//在当前窗体的容器中创建托盘图标NotifyIcon 的实例
myNotifyIcon = new NotifyIcon(this.components);
//指定托盘图标
myNotifyIcon.Icon = new Icon("demo.ico");
//鼠标悬停在托盘图标上方时显示的内容
myNotifyIcon.Text = "网络呼叫提醒\n";
//设置关联的上下文菜单
myNotifyIcon.ContextMenuStrip = this.contextMenuStrip1;
//显示托盘图标
myNotifyIcon.Visible = true;
//添加用户双击任务栏中的托盘图标时触发的事件
myNotifyIcon.DoubleClick += new EventHandler(myNotifyIcon_DoubleClick);
//获取本机第一个可用IP地址
IPAddress myIP =
(IPAddress)Dns.GetHostAddresses(Dns.GetHostName()).GetValue(0);
//为了在同一台机器调试，此IP也作为默认远程IP
textBoxRemoteIP.Text = myIP.ToString();
```

```
//创建一个线程接收远程主机发来的信息
Thread myThread = new Thread(new ThreadStart(ReceiveData));
myThread.Start();
textBoxSendMessage.Focus();
}
void myNotifyIcon_DoubleClick(object sender, EventArgs e)
{
ShowThisForm();
}
private void ShowThisForm()
{
this.Height = 0;
timer1.Enabled = true;
this.Show();
}
private void Form1_FormClosing(object sender, FormClosingEventArgs e)
{
if (isExit == false)
{
//不关闭窗体
e.Cancel = true;
//隐藏窗体
this.Hide();
}
}
private void 结束程序ToolStripMenuItem_Click(object sender, EventArgs e)
{
isExit = true;
udpClient.Close();
Application.Exit();
}
private void 呼叫对方ToolStripMenuItem_Click(object sender, EventArgs e)
{
ShowThisForm();
}
/// <summary>
/// 接收线程
/// </summary>
private void ReceiveData()
{
//在本机指定的端口接收
udpClient = new UdpClient(port);
IPEndPoint remote = null;
//接收从远程主机发送过来的信息;
while (true)
{
```

```
Try
{
//关闭udpClient 时此句会产生异常
byte[] bytes = udpClient.Receive(ref remote);
string str = Encoding.UTF8.GetString(bytes, 0, bytes.Length);
MessageBox.Show(str, string.Format("收到来自[{0}]的呼叫", remote));
}
catch
{
//退出循环, 结束线程
break;
}
}
}
/// <summary>
/// 发送数据到远程主机
/// </summary>
private void sendData()
{
UdpClient myUdpClient = new UdpClient();
IPAddress remoteIP;
if (IPAddress.TryParse(textBoxRemoteIP.Text, out remoteIP) == false)
{
MessageBox.Show("远程IP格式不正确");
return;
}
IPEndPoint iep = new IPEndPoint(remoteIP, port);
byte[] bytes = System.Text.Encoding.UTF8.GetBytes(textBoxSendMessage.Text);
try
{
myUdpClient.Send(bytes, bytes.Length, iep);
textBoxSendMessage.Clear();
myUdpClient.Close();
textBoxSendMessage.Focus();
}
catch (Exception err)
{
MessageBox.Show(err.Message, "发送失败");
}
finally
{
myUdpClient.Close();
}
}
private void buttonSend_Click(object sender, EventArgs e)
{
```

```
sendData();
}
private void timer1_Tick(object sender, EventArgs e)
{
if (formHeight < 235)
{
this.Height = formHeight;
formHeight += 5;
}
else
{
timer1.Enabled = false;
formHeight = 0;
}
}
}
}
```

实验报告中要求回答的问题

（1） 写出你认为有必要解释的关键步骤和代码。
（2） 写出实验中遇到的问题及解决方法。
（3） 你从这个实验中得到了哪些启发？

实验九 简单网络聊天系统

实验目的：
（1） 练习 TcpClient 和 TcpListener 的用法。
（2） 练习 NetworkStream 的用法。
（3） 练习 BinaryRead 和 BinaryWriter 的用法。
（4） 练习线程的创建和使用方法。
（5） 练习解决 TCP 协议消息边界问题的另一种方法。

实验内容：

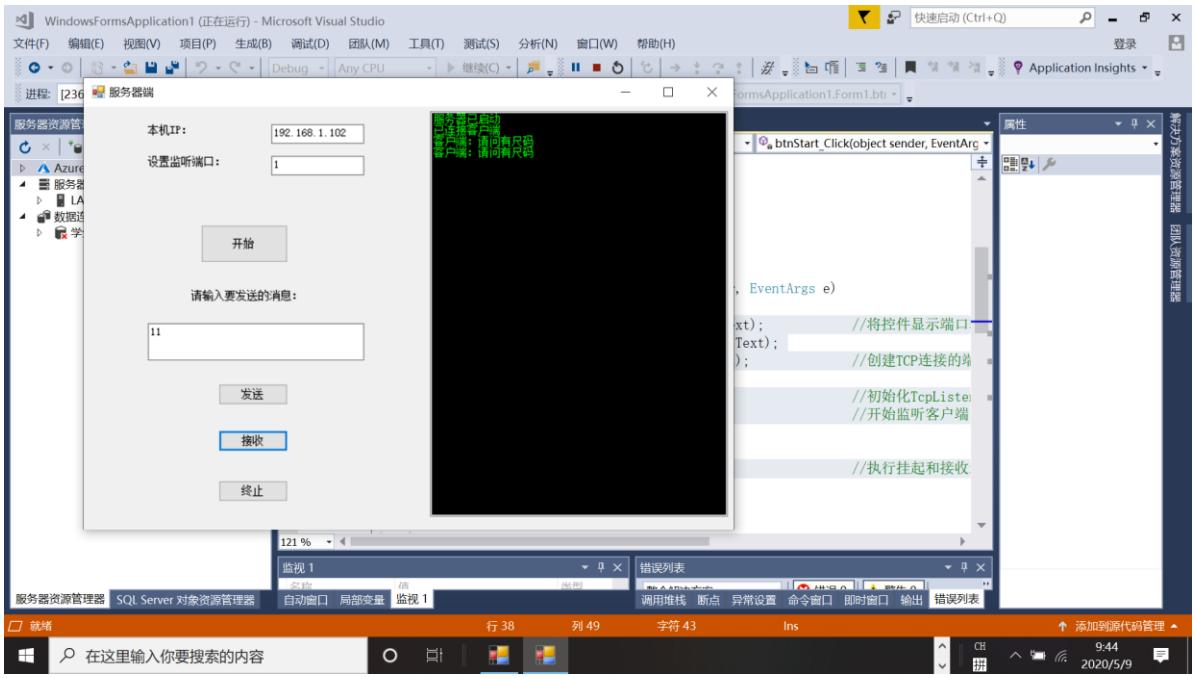
开发一个简单的基于 TCP 的网络聊天系统，服务器端和客户端设计界面分别如图 4-11和 4-12 所示。

图4-11　服务器端设计界面　　图4-12　客户端设计界面

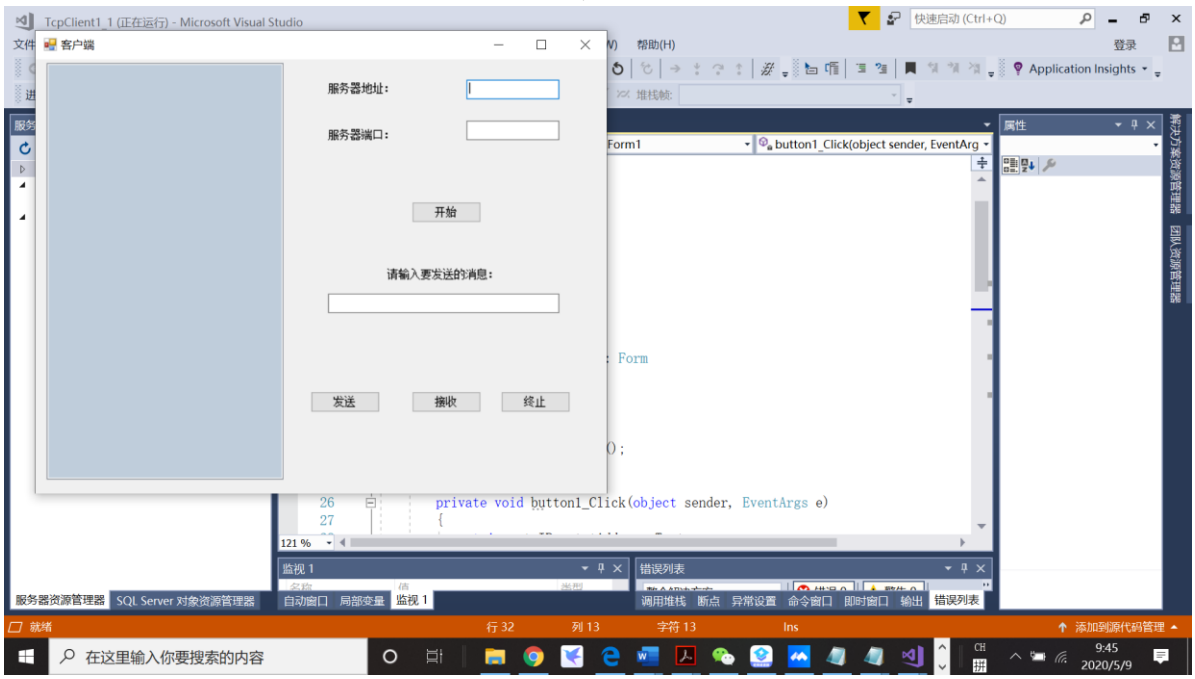
实验要求：

先看懂参考解答的源程序，然后尝试独立完成实验内容。注意一定不要采用直接复制参考解答中的源代码的办法完成实验，因为那样就失去了实验的意义。

参考代码：



服务器端



客户端

服务器端主要代码：

服务器开始监听

```
int port = Convert.ToInt32(txtPort.Text);          //将控件显示端口转化为int
            IPAddress ip = IPAddress.Parse(txtIP.Text);
            IPEndPoint p = new IPEndPoint(ip,port);          //创建TCP
```

连接的端点

```
            server = new TcpListener(p);                                //初始化
TcpListener的新实例
            server.Start();                                             //开始监
听客户端的请求
            listBox1.Items.Add("服务器已启动");

            client = server.AcceptTcpClient();                          //执行挂
起和接收连接请求，获得客户端对象
            listBox1.Items.Add("已连接客户端");
```

发送消息

```
byte[] msg = Encoding.UTF8.GetBytes(txtMessage.Text);
            stream = client.GetStream();                                //获取用于
读取和写入的流对象
            stream.Write(msg,0,msg.Length);
```

接收消息

```
byte[] bytes = new byte[512];
            stream = client.GetStream();                                //获取用于
读取和写入的流对象
            stream.Read(bytes, 0, bytes.Length);                        //读取来自
客户端的消息
            string data = Encoding.UTF8.GetString(bytes,0,bytes.Length);
            listBox1.Items.Add("客户端: "+data);
```

中断连接

```
if (stream != null) stream.Close();
            if (client != null) client.Close();
            if (server != null)  server.Stop();
```

客户端主要代码：

连接服务器

```
string strIP = txtAddress.Text;
            int port = Convert.ToInt32(txtPort.Text);

            client = new TcpClient();                       //创建TCP客户端
            client.Connect(strIP,port);
```

向服务器发送消息

```
byte[] msg = Encoding.UTF8.GetBytes(txtMessage.Text);
            stream = client.GetStream();                                //获取用于
读取和写入的流对象
            stream.Write(msg, 0, msg.Length);                           //向客户
端发送一个响应信息
```

接收消息

```
byte[] bytes = new byte[512];
            stream = client.GetStream();                          //获取用于读取和写入的流对象
            stream.Read(bytes, 0, bytes.Length);                  //读取来自客户端的消息
            string data = Encoding.UTF8.GetString(bytes, 0, bytes.Length);
            listBox1.Items.Add("服务器: " + data);
```

中断连接

```
if (stream != null) stream.Close();
            if (client != null) client.Close();
```

实验步骤提示

（1） 创建一个名为 ChatServer 的 Windows 应用程序项目，设计服务器端程序。
（2） 创建一个名为 ChatClient 的 Windows 应用程序项目，设计客户端程序。
（3） 分别运行服务器端程序和客户端程序，验证程序是否正确。

实验报告中要求回答的问题

（1） 写出你认为有必要解释的关键步骤和代码。
（2） 写出实验中遇到的问题及解决方法。
（3） 你从这个实验中得到了哪些启发？

实验十　文件数据加密与解密

实验目的：

（1） 练习数据加密与解密的实现方法。
（2） 练习将加密数据保存到文件中的方法。
（3） 练习从文件中读取加密后的数据的方法。

实验内容：

设计一个 Windows 应用程序，实现下面的功能。

1) 使用某种加密算法加密窗体上 TextBox 控件中显示的文字，然后将其保存到文本文件 MyData.txt 中。
2) 程序运行时，首先自动判断加密后的文件是否存在，如果文件存在，则根据加密密钥对加密内容进行解密，并将解密后的结果显示在窗体的 TextBox 控件中。程序运行效果如图 4-16 所示。

实验步骤提示

要顺利完成本实验，需要解决 4 个问题：一是采用哪种算法进行加密；二是如何将加密后的数据保存到文件中；三是如何保存加密密钥；四是如何将加密后的内容读取出来并进行解密处理。

由于只有知道加密密钥才能进行解密，而实验中要求程序开始运行时就判断是否有加密后的文件，并对文件进行解密。因此如何在加密的同时将加密密钥也保存下来是本完成本实验的关键。

这个实验的实现方法有多种，参考解答仅仅是其中比较简单的一种。具体实现方法可以按照下面的步骤进行。

(1) 创建一个名为 DataEncrypt 的 Windows 应用程序，在【解决方案资源管理器】中，将 Form1.cs 换名为 FormMain.cs。从【工具箱】中向设计窗体拖放一个 TextBox 控件、一个【保存】按钮和一个【打开】按钮。界面设计效果如图 4-15 所示。在【保存】按钮的 Click 事件代码中，创建 MyData.txt 文件，先将加密密钥写入文件中，然后再写入加密后的文本内容。

(2) 编写一个 DecryptFromFile 方法，先读出加密密钥，再读出加密后的文本，并根据读出的密钥进行解密，然后将解密结果显示出来。

(3) 采用这种办法，既可以实现保存密钥的目的，也起到了加密的作用。虽然该方法将加密密钥和实际数据保存在一起感觉不太安全，但是即使攻击者打开加密文件，也很难将加密密钥分离出来。当然，如果希望让加密效果更复杂些，也可以先写入几个字节的随机数，然后再保存加密密钥，读出时只需要将这几个字节先读出来，然后再读密钥即可。或者将加密密钥和被加密的数据交叉保存也会使破解的难度增大。对于一般的应用来说，这些方法已经够了。

(4) 在窗体的 Load 事件和【打开】按钮的 Click 事件中分别调用 DecryptFromFile 方法，完成解密的功能。

(5) 运行设计的程序，输入一些文字信息，单击【保存】按钮，然后打开 bin 目录下的MyData.txt 文件，观察加密后显示的结果。

图4-15 实验十的设计界面

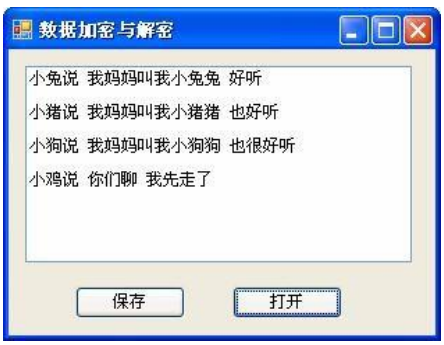


图4-16 实验十的运行效果

实验报告中要求回答的问题

(1) 写出你认为有必要解释的关键代码。

(2) 写出实验中遇到的问题及解决方法。

(3) 你从这个实验中得到了哪些启发？