

《数据库系统及应用》

课程实验指导书

西南交通大学交通运输与物流学院

2021 年 12 月

目 录

实验一 C#语言基础	1
一、实验目的	1
二、实验内容	1
三、实验仪器、设备及材料	1
四、实验原理	2
五、实验步骤	2
5.1 Visual Studio 的启动与退出	2
5.2 “Hello World!”程序与配置 C#开发环境	2
5.3 信息录入与输出的简单程序	4
5.4 值变量与引用变量的区别	5
5.5 String 类的简单应用	6
5.6 switch 语句与数组的简单应用	7
5.7 for 循环的简单示例	8
5.8 遍历数组中的元素	9
5.9 Array 类的用法	10
5.10 ArrayList 类的用法	11
5.11 静态字段与静态方法的应用	13
5.12 用“类”实现复数加减乘法	14
5.13 “类”的继承	16
实验二 C#窗体应用编程基础	19
一、实验目的	19
二、实验内容	19
三、实验仪器、设备及材料	19
四、实验原理	20
五、实验步骤	20
5.1 启动、新建、窗口调整	20
5.2 窗体的简单练习	21
5.3 复选框的应用	22
5.4 单选按钮的应用	24
5.5 文本框控件的应用	25
5.6 列表控件的应用	26
5.7 组合框控件的应用	27
5.8 TreeView 控件的应用	29
5.9 ListView 控件的应用	30
5.10 定时器控件的应用	32

5.11 打开和保存文件对话框的应用	33
5.12 下拉式菜单的应用	35
5.13 弹出式菜单的应用	38
5.14 通过静态字段传递数据	40
5.15 通过构造函数传递数据	42
5.16 多文档应用程序	43
实验三 ADO.NET 访问数据库基础.....	45
一、实验目的	45
二、实验内容	45
三、实验仪器、设备及材料.....	45
四、实验原理	46
五、实验步骤	46
预备工作.....	46
5.1 创建连接的方法.....	48
5.2 求 Product 表中指定分类的商品数.....	50
5.3 通过 MySqlCommand 对象执行更新操作	51
5.4 MySqlCommand 对象的 SQL 命令中参数的使用方法	52
5.5* (不做要求)存储过程的使用方法.....	53
5.6 MySqlDataReader 对象的使用方法	53
5.7DataSet 对象的使用方法	54
5.8 显示 Product 表的第一个记录.....	55
5.9Product 表记录浏览	57
5.10 求指定分类的商品销售总数量和总金额	59
5.11 通过 BindingNavigator 控件实现对 Product 表中所有记录的浏览、添加和删除操作	60
5.12 使用 DataView 对象在列表框中按单价升序、库存数量降序排序显示所有商品记录	65
5.13 用 DataGridView 控件显示所有商品记录.....	66
5.14Product 表的商品复杂查询.....	68
5.15Product 表的商品修改.....	72
实验四 数据库系统开发实例	75
一、实验目的	75
二、实验内容	75
三、实验仪器、设备及材料.....	75
四、实验原理	76
五、实验步骤	76
5.1 公共类设计	76
5.2 登陆窗体设计	79
5.3 主菜单窗体设计	80

5.4“添加新商品”功能设计	83
5.5“编辑商品信息”功能设计	85
5.6“增加老商品库存”功能设计	92
5.7“商品库存预警”功能设计	96
5.8“添加新顾客”功能设计	97
5.9“查看顾客购物信息”功能设计	99
5.10“顾客购物”功能设计	101
5.11“顾客退货”功能设计	104
5.12“按分类统计销售情况”功能设计	106
5.13“按商品统计销售情况”功能设计	108
5.14“用户管理”功能设计	109
5.15“设置商品类别”功能设计	114
5.16“系统初始化”功能设计	117
5.17 帮助功能设计	118

实验一 C#语言基础

一、实验目的

- (1) 本章实验的内容主要涵盖教材 1-6 章的内容，通过上机实验以加深对语言的理解；
- (2) 建立 C#的工作空间和应用，掌握 C#语言及程序设计，掌握多种数据类型的使用方法、常用控制语句的使用方法、数组的使用方法、类的使用。

二、实验内容

本实验主要包括如下部分的内容。

- 1 Visual Studio 的启动与退出
- 2 “Hello World!”程序与配置 C#开发环境
- 3 信息录入与输出的简单程序
- 4 值变量与引用变量的区别
- 5String 类的简单应用
- 6switch 语句与数组的简单应用
- 7for 循环的简单示例
- 8 遍历数组中的元素
- 9Array 类的用法
- 10ArrayList 类的用法
- 11 静态字段与静态方法的应用
- 12 用“类”实现复数加减乘法
- 13 “类”的继承

其中实验 1 和 2 是简单的引入部分。实验 3 基于第一章的内容。实验 4 和 5 基于第二章的内容。实验 6/7/8 基于第三章的内容。实验 9 和 10 基于第四章的内容。实验 11 和 12 基于第五章的内容。实验 13 基于第六章的内容。

三、实验仪器、设备及材料

硬件：计算机；《C#语言与数据库技术基础教程》

软件：Windows 操作系统，Visual Studio 2019（推荐使用最新版本）

四、实验原理

C#读作 C Sharp，是一种基于 .NET Framework 的、面向对象、类型安全的编程语言。C#语言和 .NET Framework 是相伴相生的，C#不是单独的一个的开发环境，而是 .NET Framework 的最好的一种开发语言。C#语言可以开发以 .NET Framework 为平台的应用程序；.NET Framework 为 C#开发的程序提供了运行环境。

C#的编程环境是 Visual Studio。Visual Studio 是一种开发基于 .NET Framework 应用程序的集成环境，由 Microsoft 开发，目前最新版本为 Visual Studio 2019。

五、实验步骤

5.1 Visual Studio 的启动与退出

启动步骤如下：

在 Windows 操作系统的“开始”菜单中，找到并单击 Visual Studio 2019，即可启动 Visual Studio 2019。

退出步骤如下：

在 Visual Studio 集成开发环境中单击右上角“关闭”按钮“×”；或选择顺次点击菜单栏“文件”->“退出”。退出时，Visual Studio 会自动判断用户是否对项目进行了修改，并会询问用户是否保存修改，这一点和 Microsoft office Word 等相似。

5.2 “Hello World!”程序与配置 C#开发环境

长期以来，编程界都认为刚接触一门新语言时，如果首先使用它来编写一个在屏幕上显示消息“Hello World!”的程序，将会带来好运。下面就新建 C#的“控制台应用”程序。

【例 1-1】创建一个控制台应用程序，输出“Hello World! ”。如图 1-1 所示。

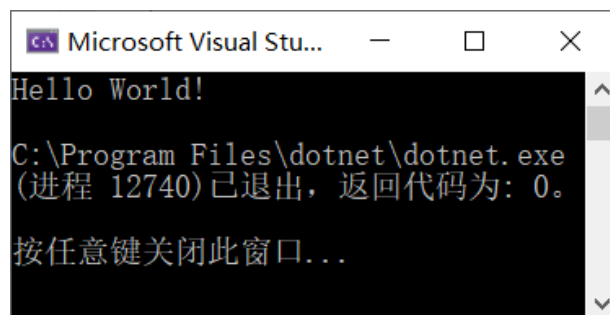


图 1-1 Hello World! 控制台应用程序外观图

本应用程序的制作过程如下。

首先在硬盘上创建一个目录，如“E:\Test_For_C#”，用以保存我们创建的项目。然后按照以下具体步骤依次操作。

1. 创建应用

创建新的控制台应用。在打开 Visual Studio 2019 之后，如图 1-2 所示。点击“创建新项目”。之后如图 1-2 所示。在上部输入“控制台应用”搜索，或使用“语言”与“平台”筛选选项进行筛选。输入项目名称为 Project_1，指定位置为上一步创建的目录，再点击确定。出现如图 1-4 的界面。



图 1-2 Visual Studio 2019 界面



图 1-3 创建控制台应用

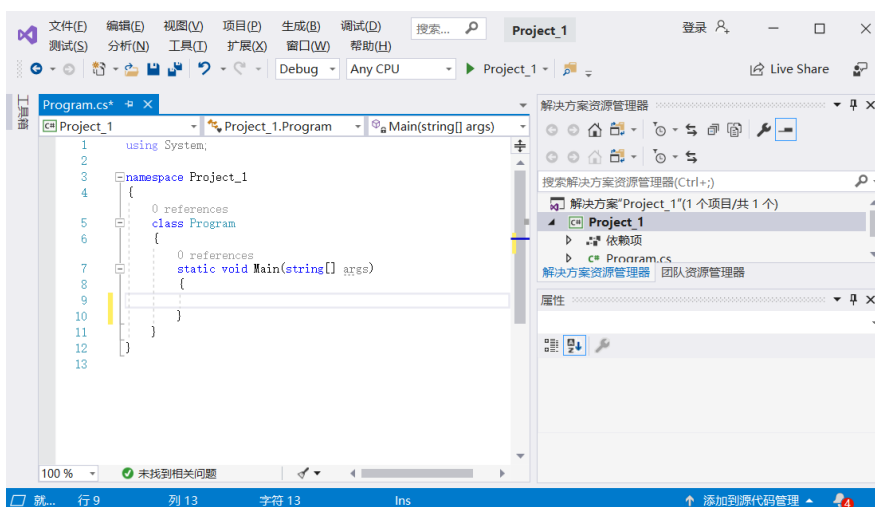


图 1-4 初始界面

2.配置 C#开发环境

在这里，就要确定自己所处的开发环境是 C#，依次点击菜单栏“工具”->“导入导出设置”，选择“重置所有设置”项->“否，仅重置设置，从而覆盖我的当前设置”，默认设置集合选择“Visual C#”，点击“完成”，就进行了环境配置。

3.编写程序

将光标移动到代码编辑窗口中 Main 函数处，输入 “Console.WriteLine(“Hello World!”);”，如图 1-5 所示。这里要注意语句语法的大小写。

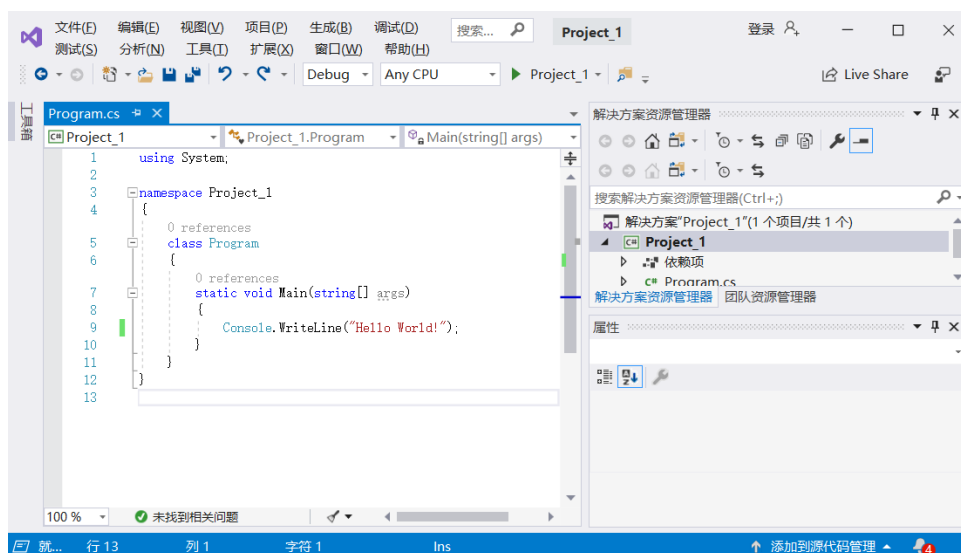


图 1-5 Project_1 项目

4.点击工具栏保存按钮（或按 Ctrl+S）保存项目。按 Ctrl+F5 直运行程序。结果如图 1-1 所示。

5.如果只想列出程序中引用的命名空间，右击鼠标，在快捷菜单选择“删除 using 和对其排序”即可。

5.3 信息录入与输出的简单程序

创建一个简单的程序，实现学生学号与姓名的录入，并在屏幕上再输出出来。

程序代码如下：

```
using System;
namespace basic_input_output
{
    class Student          //声明一个Student类
    {
        int student_number;    //类成员
        string name;          //类成员
        public void GetData()  //类成员
        {
            Console.WriteLine("输入一个学生信息:");
            Console.Write("学号:");
            student_number = int.Parse(Console.ReadLine());
        }
    }
}
```



```

        Console.WriteLine("姓名:");
        name = Console.ReadLine();
    }

    public void DispData() //类成员
    {
        Console.WriteLine("输出一个学生信息:");
        Console.WriteLine("学号:{0}, 姓名:{1}", student_number, name);
    }
}

class Program
{
    static void Main(string[] args)
    {
        Student student_1 = new Student(); //创建Student类的对象
        student_1.GetData(); //调用类方法
        student_1.DispData(); //调用类方法
    }
}

```

程序运行示例如图 1-6。

```

Microsoft Visual Studio 调试控制台
学号:2019
姓名:陆游
输出一个学生信息:
学号:2019, 姓名:陆游

C:\Program Files\dotnet\dotnet.exe (进程 5496) 已退出, 返回代码为: 0。
若要在调试停止时自动关闭控制台, 请启用“工具”->“选项”->“调试”->“
调试停止时自动关闭控制台”。
按任意键关闭此窗口...

```

图 1-6 信息录入与输出的简单程序

5.4 值变量与引用变量的区别

程序的内存空间分为栈空间和堆空间，值类型的数据在栈空间中分配，而引用类型数据（对象）在堆空间中分配。

```

using System;

namespace deferences_between_value_and_reference
{
    class MyClass
    {
        int n;
    }

    class Program
    {
        static void Main(string[] args)
        {
            int a = 1, b;

```

```

b = a;
Console.WriteLine("a、b的值是否相等:{0}", object.Equals(a, b));
Console.WriteLine("a、b的引用是否相等:{0}",
    object.ReferenceEquals(a, b));
MyClass obj1 = new MyClass();
MyClass obj2 = new MyClass();
Console.WriteLine("obj1、obj2的值是否相等:{0}",
    object.Equals(obj1, obj2));
Console.WriteLine("obj1、obj2的引用是否相等:{0}",
    object.ReferenceEquals(obj1, obj2));
obj2 = obj1;
Console.WriteLine("执行obj2=obj1");
Console.WriteLine("obj1、obj2的值是否相等:{0}",
    object.Equals(obj1, obj2));
Console.WriteLine("obj1、obj2的引用是否相等:{0}",
    object.ReferenceEquals(obj1, obj2));
    }
}
}

```

结果如图 1-7 所示。

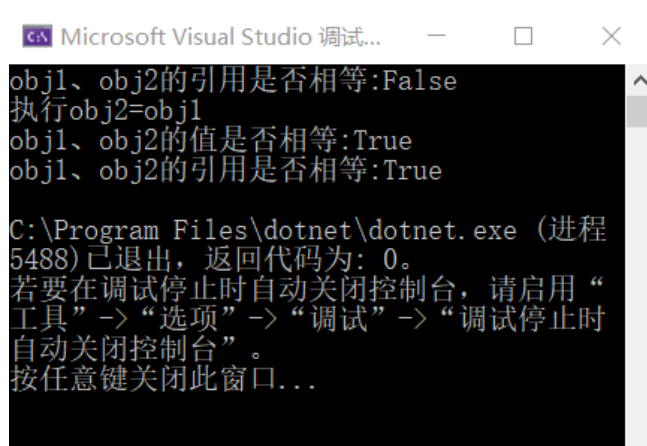


图 1-7 值变量与引用变量的区别

5.5 String 类的简单应用

String 类的变量是一串字符串。字符串的对象多种多样，可以是学号、姓名，也可以是住址、邮箱地址。了解如何使用 String 类，了解并使用 String 类的相关方法，是很有必要的。下面我们就来实现求出某一段字符串在一整个字符串中的位置。下面是获取输入邮箱的用户名（即去掉@和域名）的代码。

```

using System;
namespace interesting_usage_of_string
{
    class Program

```

```

{
    static void Main(string[] args)
    {
        string email;           //邮箱
        string name;             //用户名
        Console.Write("输入邮箱:");
        email = Console.ReadLine();
        int position = email.IndexOf("@"); //求@的位置。这里@也可以是字符串，比如".com", ".cn", "@@"等
        if (position > 0)
        {
            name = email.Substring(0, position);
            Console.WriteLine("邮箱:{0},用户名:{1}", email, name);
        }
        else Console.WriteLine("邮箱格式错误!");
    }
}

```

结果如图 1-8 所示

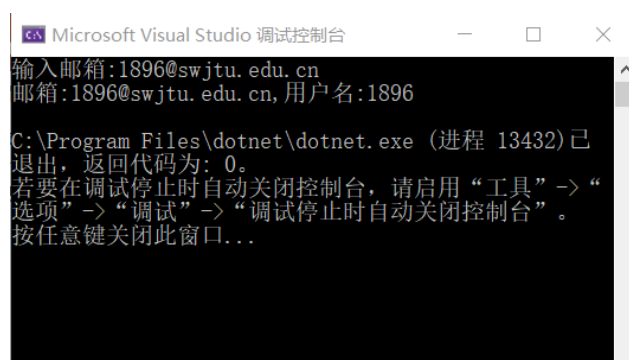


图 1-8 String 类的简单用法

5.6 switch 语句与数组的简单应用

switch 与 if 相关语句，都是选择控制语句。if 相关语句接触与使用更多，相信大家有更为清晰地认识，故不再练习。switch 语句作为不太常用的语句，在这里进行一下专门的练习。同时，字符串数组的使用也在这里进行了展示，作为对教材上两组示例的合并。

下面这个程序将实现获取今日的日期，并显示今日是周几、今天是否工作等信息。如“2019 年 7 月 9 日是星期二，我工作。”下面展示的示例代码用到的主要内容有字符串的合并、数组、DateTime、switch 语句。

```

using System;
namespace usage_of_switch
{
    class Program

```

```

{
    static void Main(string[] args)
    {
        DateTime d = DateTime.Now;
        int n = (int)d.DayOfWeek;
        string sentence;
        string[] weekdays = new string[] { "星期日", "星期一",
            "星期二", "星期三", "星期四", "星期五", "星期六" };
        sentence = d.Year + "年" + d.Month + "月" + d.Day + "日是"+weekdays[n]+"，"
        switch (n)
        {
            case 1:
            case 2:
            case 3:
            case 4:
            case 5:
                sentence = sentence + "我工作。";break;
            case 0:
            case 6:
                sentence = sentence + "我休息。";break;
        }
        Console.WriteLine(sentence);
    }
}

```

结果如图 1-9 所示。

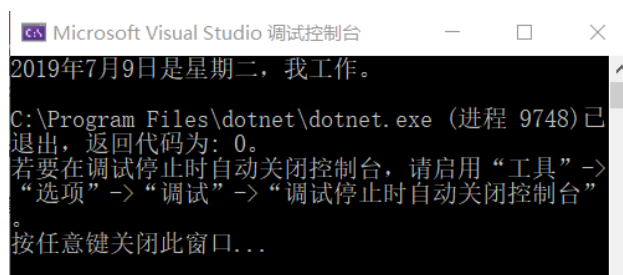


图 1-9switch 与数组的简单应用

5.7for 循环的简单示例

for 循环是形式简洁易懂的循环语句，在多种语言中都是常用的循环控制语句。掌握 for 语句的用法对于 C#和其他语言的使用都有很大的帮助。

下面的练习是使用 for 循环语句计算阶乘的累加值，即 $\sum_{i=1}^n i!$ ，其中的 n 由自己输入。

```

using System;
namespace simple_usage_of_for
{

```

```

class Program
{
    static void Main(string[] args)
    {
        int n;
        Console.Write("n:");
        n = int.Parse(Console.ReadLine());
        int total = 0, i, m = 1;
        for (i = 1; i <= n; i++)
        {
            m *= i;
            total += m;
        }
        Console.WriteLine("计算结果={0}", total);
    }
}

```

输入 n=5 的结果如图 1-10 所示。可以验证， $1!+2!+3!+4!+5!=1+2+6+24+120=153$ 。

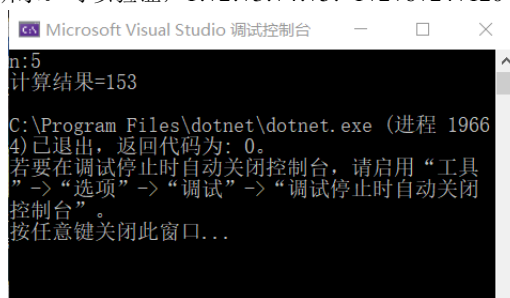


图 1-10 使用 for 循环计算阶乘的累加值

5.8 遍历数组中的元素

数组是所有编程语言中都需要用到的基础的数据组织方式，不同编程语言的数组特性与调用方法都有相似相同之处。下面的练习是使用 for 循环与 C# 的 foreach 语句来讲数组内的全部元素输出。

首先练习使用 for 循环的方式。

```

using System;
namespace print_array
{
    class Program
    {
        static void Main(string[] args)
        {
            int[,] b = { { 1, 2, 3 }, { 4, 5, 6 } };
            for (int i = 0; i < 2; i++)
            {
                for (int j = 0; j < 3; j++)

```

```

        Console.Write("{0} ", b[i, j]);
        Console.WriteLine();//这里只是将数组的两行分开显示
    }
}
}
}

```

结果如图 1-11 所示。

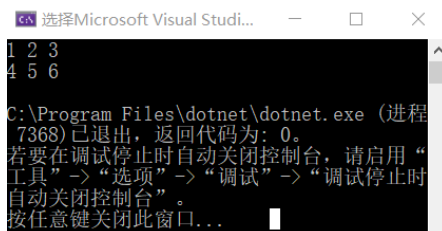


图 1-11 使用 for 循环历遍二维数组

```

using System;
namespace print_array
{
    class Program
    {
        static void Main(string[] args)
        {
            int[,] b = { { 1, 2, 3 }, { 4, 5, 6 } };
            foreach (int i in b) //也可以使用foreach (var i in e)
                Console.Write("{0} ", i);
            Console.WriteLine();
        }
    }
}

```

使用 foreach 语句的结果如图 1-12 所示。



图 1-12 使用 foreach 语句历遍

两种方式均可以实现历遍数组内的全部元素，foreach 更简洁，for 循环更灵活。

5.9 Array 类的用法

上文已经陈述数组应用的广泛性和重要性，因而掌握数组的常见属性和方法很有必要。比如，对一个未知的数组，可以使用代码语言来了解它的维度、每一个维度元素的个数等信息。

下文的例子中，我们首先自行创建了一个二维数组，之后使用 Rank 确定了维数，Length 确定数组元素的总个数，GetLowerBound 和 GetUpperBound 确定每一个维度的上下限。

```
using System;
namespace usage_of_Array
{
    class Program
    {
        static void Main(string[] args)
        {
            int i;
            int[,] A = new int[,] { { 2, 3, 4, 5 }, { 6, 7, 8, 9 }, { 1, 2, 3, 4 },
                                     { 5, 6, 7, 8 }, { 1, 2, 3, 4 }, { 5, 6, 7, 8 } };
            Console.WriteLine("A是{0}维数组", A.Rank);
            Console.WriteLine("A中元素个数: {0}", A.Length);
            for (i = 0; i < A.Rank; i++) //个人以为在这里i的初值设置为0更合适，因为索引都是从0开始的，在使用 A.GetLowerBound(i)的时候，不需要对i在进行额外处理
                Console.WriteLine("第{0}维, 下限为{1}, 上限为{2}", i+1,
A.GetLowerBound(i), A.GetUpperBound(i));
        }
    }
}
```

本例的结果如图 1-13 所示。

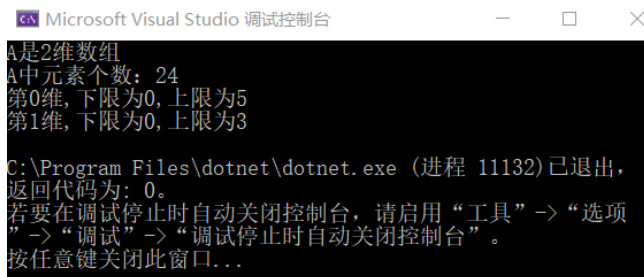


图 1-13 Array 类的用法简示

5. 10 ArrayList 类的用法

ArrayList 可以实现类似 Array 的功能，同时可以实现数组的动态化。下面的练习将从键盘读入一些数字，并将它们按照你的设计拆分成你预期输入的数字（因为电脑读入的是字符串），再对这些数字按照奇数与偶数进行分类与排序，并输出。

详细的要求是：定义两个 ArrayList 对象 odd_array 和 even_array 一次性输入一串整数，整数之间用空格分隔，然后将奇数存放在 odd_array 中，将偶数存放在 even_array 中，统计它们中的元素个数，并输出排序前、后的结果。

```
using System;
using System.Collections; //注意这一条，否则无法使用ArrayList
namespace usage_of_ArrayList
```

```

{
    class Program
    {
        static void Main(string[] args)
        {
            ArrayList odd_array = new ArrayList();
            ArrayList even_array = new ArrayList();
            Console.WriteLine("请输入若干整数，整数间以空空间隔开。");
            string array= Console.ReadLine();
            string[] split_array = array.Split(" "); //将整个字符串用“ ”（空格）分离
            开。计算机并不能“智能”地读取你大脑中的想法，你只能告诉它“我是用空格来将数字分离
            的！”

            foreach (string str_number in split_array)
            {
                int number = int.Parse(str_number);
                if (number % 2 == 1) // 算数符%为取余数，奇数余数为1，偶数为0。由于C#
                的判断值只能是bool类型，并不能像C/C++一样使用int类型进行判断。所以这里必须使用“==1”判
                断，得到bool类型的数字“0”或“1”。而在C/C++/Python中，这一次判断就可以省略。
                    odd_array.Add(number);
                else
                    even_array.Add(number);
            }
            Console.WriteLine("奇数个数: {0}; 偶数个数: {1}", odd_array.Count,
            even_array.Count);
            Console.Write("所有奇数: ");Print_array(odd_array);
            odd_array.Sort();
            Console.Write("排序后所有奇数: "); Print_array(odd_array);
            Console.Write("所有偶数: "); Print_array(even_array);
            odd_array.Sort();
            Console.Write("排序后所有偶数: "); Print_array(even_array);
        }

        static void Print_array(ArrayList array) //自定义了Print_array函数，因为主函数
        中多次使用了相关的功能
        {
            foreach (int number in array)
                Console.Write("{0} ", number); //这里有空格，使输出的数字可以间隔开
            Console.WriteLine();
        }
    }
}

```

当输入的字符串为“3 5 2 6 8 10 4 7 1 9”时，结果如图 1-14 所示。

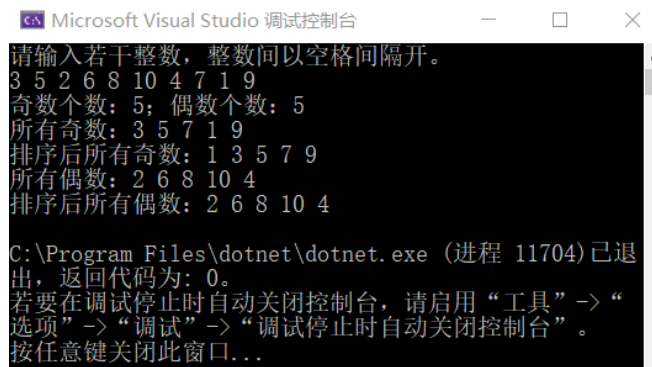


图 1-14 ArrayList 类的用法

5.11 静态字段与静态方法的应用

使用静态成员（静态字段与静态方法），实现对类对象自身的变量的操作，而不是操作类的对象的变量。

下面的例子要求统计任意名同学的语数外三科平均分，并顺次输出学号、姓名、语数外三科分数、三科平均分；并输出这些同学语文、数学、英语的三门课程的平均分。

```

using System;
namespace static_class
{
    class Score           //声明Score类
    {
        int no;           //学号
        string name;      //姓名
        int chinese;      //语文成绩
        int math;         //数学成绩
        int english;      //英语成绩
        static int chinese_sum = 0; //语文总分
        static int math_sum = 0;    //数学总分
        static int english_sum = 0; //英语总分
        static int number_of_students = 0; //总人数
        public Score(int n, string na, int d1, int d2, int d3) //构造函数
        {
            no = n; name = na;
            chinese = d1; math = d2; english = d3;
            chinese_sum += chinese;
            math_sum += math; english_sum += english;
            number_of_students++;
        }
        public void disp() //定义disp()方法
        {

```

```

        Console.WriteLine("学号:{0} 姓名:{1} 语文:{2} 数学:{3} 英语:{4}"+" 平均分:{5:f}", no, name, chinese, math, english, (double)(chinese + math + english) / 3);
    }

    public static double avg1() { return (double)chinese_sum /
number_of_students; }    //静态方法
    public static double avg2() { return (double)math_sum / number_of_students; }
//静态方法
    public static double avg3() { return (double)english_sum /
number_of_students; }    //静态方法
    }

    class Program
    {
        static void Main(string[] args)
        {
            Score s1 = new Score(1, "王华", 85, 89, 90);
            Score s2 = new Score(2, "李明", 78, 74, 65);
            Score s3 = new Score(3, "张兵", 82, 89, 82);
            Score s4 = new Score(4, "王超", 65, 98, 72);
            Console.WriteLine("输出平均分结果如下");
            s1.disp(); s2.disp(); s3.disp(); s4.disp();
            Console.WriteLine("语文平均分:{0} 数学平均分:{1} 英语平均
分:{2}", Score.avg1(), Score.avg2(), Score.avg3());
        }
    }
}

```

示例的运行结果如图1-15所示。

```

Microsoft Visual Studio 调试控制台
输出平均分结果如下
学号:1 姓名:王华 语文:85 数学:89 英语:90 平均分:88.00
学号:2 姓名:李明 语文:78 数学:74 英语:65 平均分:72.33
学号:3 姓名:张兵 语文:82 数学:89 英语:82 平均分:84.33
学号:4 姓名:王超 语文:65 数学:98 英语:72 平均分:78.33
语文平均分:77.5 数学平均分:87.5 英语平均分:77.25

C:\Program Files\dotnet\dotnet.exe (进程 9316) 已退出,
返回代码为: 0。
若要在调试停止时自动关闭控制台, 请启用“工具”->“选项”->“调试”->“调试停止时自动关闭控制台”。
按任意键关闭此窗口...

```

图 1-15 类的静态字段和静态方法的应用

5.12 用“类”实现复数加减乘法

“类”在所有面向对象的编程语言中都有广泛的应用，使用者可以用“类”来模拟现实生活中的所有事情，一只布谷鸟，一支雪糕等等。使用“类”可以简洁便捷地实现相同类别事物的操作；同时使代码可读性更强，即使是其他人读到你的代码，也可以快速理解。

下面的示例创建了“复数类”与“复数运算类”。“复数类”实现实部与虚部的存储，虚数的输出显示；“复数运算类”分别实现了加减乘法。

```
using System;
namespace calculation_of_imaginary_numbers
{
    class Complex //复数类
    {
        public double e1 { set; get; }
        public double e2 { set; get; }
        public Complex(double e1, double e2)
        {
            this.e1 = e1;
            this.e2 = e2;
        }
        public void Dispcomp()
        {
            if (e2 >= 0)
                Console.WriteLine("{0} + {1}i", e1, e2);
            else
                Console.WriteLine("{0} - {1}i", e1, -e2);
        }
    }
    class ComOp //复数运算类
    {
        public Complex add(Complex c1, Complex c2) //加法addition
        {
            double e1 = c1.e1 + c2.e1;
            double e2 = c1.e2 + c2.e2;
            return new Complex(e1, e2);
        }

        public Complex sub(Complex c1, Complex c2) //减法subtraction
        {
            double e1 = c1.e1 - c2.e1;
            double e2 = c1.e2 - c2.e2;
            return new Complex(e1, e2);
        }

        public Complex mul(Complex c1, Complex c2) //乘法multiplication
        {
            double e1 = c1.e1 * c2.e1 - c1.e2 * c2.e2;
            double e2 = c1.e1 * c2.e2 + c1.e2 * c2.e1;
            return new Complex(e1, e2);
        }
    }
    class Program
    {
        static void Main(string[] args)
```

```

{
    Complex c1 = new Complex(2, 3); //设定两个虚数c1、c2的值
    Complex c2 = new Complex(-2, 5);
    Console.WriteLine("c1:"); c1.Dispcomp();
    Console.WriteLine("c2:"); c2.Dispcomp();
    Complex c3, c4, c5;
    ComOp op = new ComOp();
    c3 = op.add(c1, c2);
    Console.WriteLine("c3=c1 + c2");
    Console.WriteLine("c3:"); c3.Dispcomp();
    c4 = op.sub(c1, c2);
    Console.WriteLine("c4 =c1 - c2");
    Console.WriteLine("c4:"); c4.Dispcomp();
    c5 = op.mul(c1, c2);
    Console.WriteLine("c5=c1 * c2");
    Console.WriteLine("c5:"); c5.Dispcomp();
}
}
}
}

```

示例的运行结果如图 1-16 所示。

```

Microsoft Visual Studio 调试控制台
c1:2 + 3i
c2:-2 + 5i
c3=c1 + c2
c3:0 + 8i
c4 =c1 - c2
c4:4 - 2i
c5=c1 * c2
c5:-19 + 4i

C:\Program Files\dotnet\dotnet.exe (进程 16040) 已退出, 返回代码为: 0。
若要在调试停止时自动关闭控制台, 请启用“工具”->“选项”->“调试”->“调试停止时自动关闭控制台”。
按任意键关闭此窗口...

```

图 1-16 用“类”实现复数加减乘法

5.13 “类”的继承

“类”的继承功能使功能相近的类的关系更加清晰, 代码更简洁, 同时类的功能的修补更加容易 (子类的功能可以随父类的功能的修改而变化)。

本例设计一个员工类 **Employee**, 包括员工姓名和 **Gohome** 方法 (显示回家使用的交通工具)。交通工具包括地铁、自行车、汽车, 这些交通工具均从同一个抽象类中派生出来。最后显示若干员工回家使用交通工具的情况。

本例所输出的结果虽然也可以不使用“类”而实现，但为了更好地理解“类”的功能，仍希望大家可以使用“类”来完成。结果如何从示例代码中“游走”的，也希望大家可以有自己的理解

```
using System;
using System.Collections.Generic;
namespace inheritance_of_class
{
    abstract class Traff //交通方式类
    {
        abstract public void Run(); //抽象方法声明
    }
    class Tube : Traff //地铁类
    {
        public override void Run()
        {
            Console.Write("乘坐地铁");
        }
    }
    class Cars : Traff //汽车类
    {
        public override void Run()
        {
            Console.Write("驾驶汽车");
        }
    }
    class Bicycle : Traff //自行车类
    {
        public override void Run()
        {
            Console.Write("骑自行车");
        }
    }
    class Employee //员工类
    {
        string name; //姓名
        public Employee(string name)
        {
            this.name = name;
        }
        public void Gohome(Traff tool)
        {
            Console.Write("员工" + name);
            tool.Run();
            Console.WriteLine("回家");
        }
    }
}
```

```
}  
class Program  
{  
    static void Main(string[] args)  
    {  
        List<Employee> emplist = new List<Employee>();  
        Employee emp = new Employee("王华");  
        emplist.Add(emp);  
        emp = new Employee("李明");  
        emplist.Add(emp);  
        emp = new Employee("程军");  
        emplist.Add(emp);  
        emplist[0].Gohome(new Cars());  
        emplist[1].Gohome(new Bicycle());  
        emplist[2].Gohome(new Tube());  
    }  
}
```

示例的运行结果如图 1-17 所示。

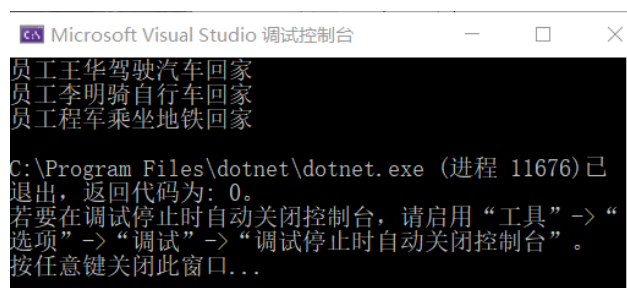


图 1-17 “类”的继承

实验二 C#窗体应用编程基础

一、实验目的

(1) 本章实验的内容主要为教材第 7 章的内容，通过上机实验以加深对本章节的理解。同时，窗口设计是一项要求实践操作的知识（或者说技术），上机实验是必不可少的。

(2) 了解 Windows 窗体应用程序的概况，掌握常见的控件设计、通用对话框设计、菜单设计、工具栏控件和状态栏控件的设计，多窗体调用和数据传递，多文档窗体的设计。

二、实验内容

本实验主要包括如下部分的内容。

- 1 启动、新建、窗口调整
- 2 窗体的简单练习
- 3 复选框的应用
- 4 单选按钮的应用
- 5 文本框控件的应用
- 6 列表控件的应用
- 7 组合框控件的应用
- 8TreeView 控件的应用
- 9ListView 控件的应用
- 10 定时器控件的应用
- 11 打开和保存文件对话框的应用
- 12 下拉式菜单的应用
- 13 弹出式菜单的应用
- 14 通过静态字段传递数据
- 15 通过构造函数传递数据
- 16 多文档应用程序

其中实验 1 和 2 是简单的引入部分。实验 3 基于第一章的内容。实验 4 和 5 基于第二章的内容。实验 6/7/8 基于第三章的内容。实验 9 和 10 基于第四章的内容。实验 11 和 12 基于第五章的内容。实验 13 基于第六章的内容。请大家务必携带教材《C#语言与数据库技术基础教程》！

三、实验仪器、设备及材料

硬件：计算机；《C#语言与数据库技术基础教程》

软件：Windows 操作系统，Visual Studio 2019（推荐使用最新版本）

四、实验原理

Windows 窗体控制程序（建成 WinForm 程序）是在用户计算机上运行的客户端应用程序，和控制台应用程序相比，界面具有更好的交互性。

五、实验步骤

5.1 启动、新建、窗口调整

- (1) 启动。启动 Visual Studio 2019
- (2) 新建。选择“Windows 窗体应用程序”模板，指定创建的位置（如 E:\Test_For_C#），自定义命名（如 WindowsFormsTest_1），点击“确定”按钮，即完成了新建。
- (3) 图 2-1 是理想中的界面，但实际操作中可能遇到多种问题，如工具箱条框缺失等。如果是工具箱条框缺失，可以依据图 2-2 的指示打开工具箱条框。其他界面问题，相信你可以通过“百度一下”自行解决。

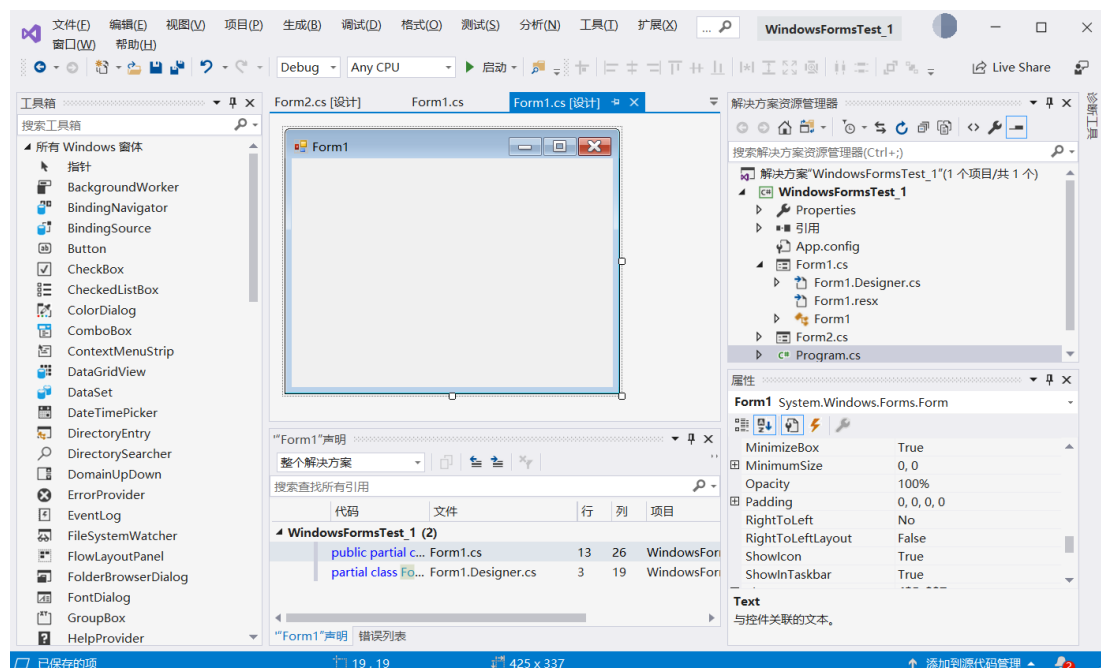


图 2-1 “Windows 窗体应用程序”的操作界面

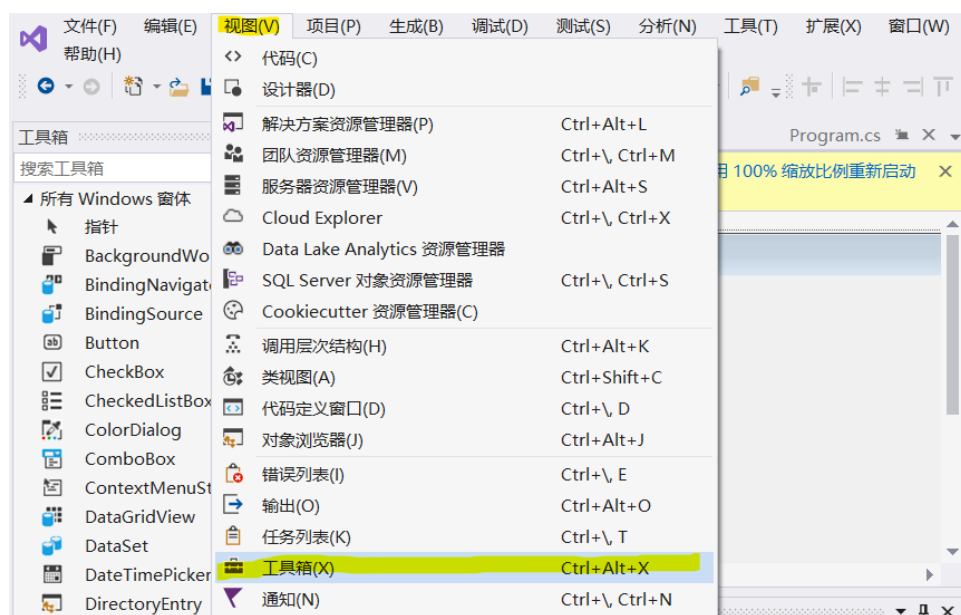


图 2-2 工具箱条框的打开

5.2 窗体的简单练习

在前面已经创建的项目的 Windows 窗体应用程序中添加一个 Form2 的窗体，其功能是显示 button1_Click 的两个参数的相关信息。

其步骤如下：

- (1) 在前面已经创建的项目（如 WindowsFormsTest_1）中，执行“项目添加 Windows 窗体”菜单命令，选择“Windows 窗体”模板，保持默认窗体名 Form2，单击“添加”按钮，在项目中添加一个 Form2 的空窗体。
- (2) 在本窗体中拖放两个标签（label1 和 label2）、两个文本框（textBox1 和 textBox2）和一个命令按钮 button1。Form2 窗体的设计界面如图 2-3(a)所示。
- (3) 设计如下事件处理方法：

```
private void Button1_Click(object sender, EventArgs e)
{
    Button bt = (Button)sender;
    textBox1.Text = bt.Name;
    textBox2.Text = e.ToString();
}
```

- (4) 修改 label1 和 label2 的属性，将 Text 框修改为对应的文字。
- (5) 修改 Program.cs 文件，将 Form2 设置为启动窗体。双击右上方“解决方案管理器”中的 Program.cs，即打开了 Program.cs；再找到代码 Application.Run(new Form1()); 将上一行代码中的 Form1 修改为 Form2，即本行修改后的结果如下一行所示。结果如图 2-4 所示。 Application.Run(new Form2());

(6) 单击工具栏中的“启动”按钮执行 Form2 窗体，单击 button1 命令按钮，其结果如图 2-3(b)所示，从中看到，button1_Click 的 sender 参数值为 button1，e 参数表示是鼠标事件参数。

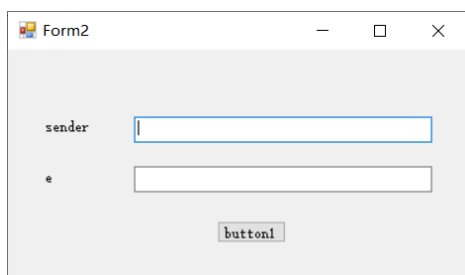


图 2-3(a)

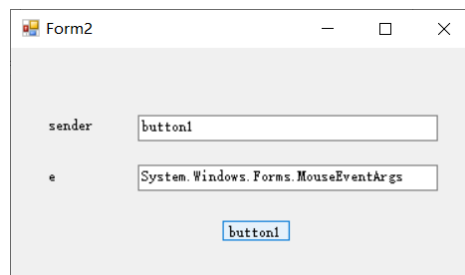


图 2-3(b)

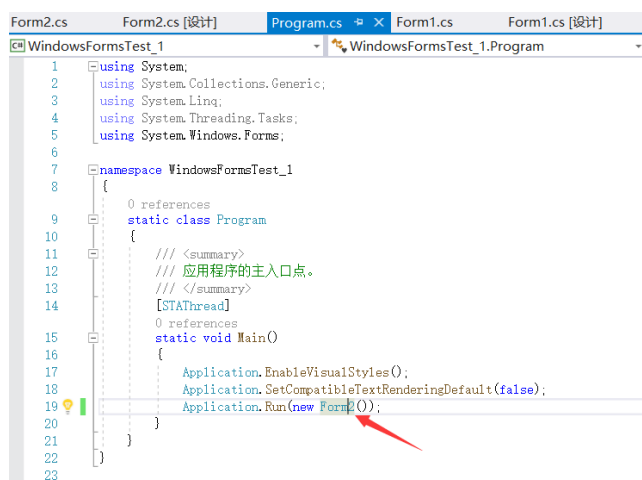


图 2-4 修改 Program.cs 中的 Form2

5.3 复选框的应用

本例是教材第 7 章 P170 的内容，主要要求与流程请参照教材，这里只提醒与指导需要注意的事项。

(1) textBox1 的字体可以在属性中选择。即鼠标右击 textBox1，选择“属性”（当然，如果左击选中 textBox1 后，Visual Studio 2019 右下角直接显示了“属性”选项卡，那是最好的。），就可以在 Visual Studio 2019 右下角，找到 Font 选项，更改为自己需要的字体。

(2) textBox1 添加后，在右上角会显示一个黑色的小三角形框，点击小三角形，就出现了“多行输入”的选项，根据自己的需要勾选或不勾选。如图 2-5 所示。



图 2-5 勾选 MultiLine

(3) checkbox1/2/3 显示的修改可以在其“属性”->“Text”中修改。逐个点击 checkbox1/2/3，就可以打开各自对应的代码编辑块，输入相应的控制语句。控制语句在如下的示例代码中。

```
private void CheckBox1_CheckedChanged(object sender, EventArgs e)
{
    {
        if (checkBox1.Checked)
        {
            Font newFont;
            newFont = new Font(textBox1.Font, textBox1.Font.Style |
FontStyle.Bold);
            textBox1.Font = newFont;
        }
        else
        {
            Font newFont;
            newFont = new Font(textBox1.Font, textBox1.Font.Style ^
FontStyle.Bold);
            textBox1.Font = newFont;
        }
    }
}
```

```
private void CheckBox2_CheckedChanged(object sender, EventArgs e)
{
    {
        if (checkBox2.Checked)
        { Font newFont;
            newFont = new Font(textBox1.Font, textBox1.Font.Style
|FontStyle.Underline);
            textBox1.Font = newFont;
        }
        else
        {
            Font newFont;
            newFont = new Font(textBox1.Font, textBox1.Font.Style ^
FontStyle.Underline);
            textBox1.Font = newFont;
        }
    }
}
```

```
private void CheckBox3_CheckedChanged(object sender, EventArgs e)
{
    if (checkBox3.Checked)
```

```

        { Font newFont;
          newFont = new Font(textBox1.Font,
textBox1.Font.Style|FontStyle.Italic);
          textBox1.Font = newFont;
        }
        else
        {
          Font newFont;
          newFont = new Font(textBox1.Font, textBox1.Font.Style ^
FontStyle.Italic);
          textBox1.Font = newFont;
        }
      }
    }
  }
}

```

(4) 在 Program.cs 中, new Form 所在行改为需要运行的窗体的名称。

(5) 输出结果如图 2-6 所示。当然, 你可以进行更美观的排版。

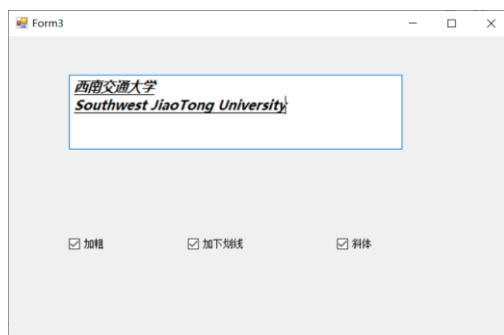


图 2-6 复选框的应用

5.4 单选按钮的应用

本例是教材第 7 章 P171 的内容, 主要要求与流程请参照教材, 这里只提醒与指导需要注意的事项。

(1) radioButton 的功能块, 均需要在 Form4.cs[设计]界面双击相应的 radioButton 打开, 并在大括号内输入相应语句。

(2) 修改 Program.cs 中的 new Form 后的数字, 本例应修改为:

```

static void Main()
{
    Application.EnableVisualStyles();
    Application.SetCompatibleTextRenderingDefault(false);
    Application.Run(new Form4());
}

```

(3) 示例代码如下。

```

private void RadioButton1_CheckedChanged(object sender, EventArgs e)
{
    if (textBox1.Text != "" && textBox2.Text != "")
    {

```

```

        int a = int.Parse(textBox1.Text);
        int b = int.Parse(textBox2.Text);
        textBox3.Text = (a + b).ToString();
    }
}

private void RadioButton2_CheckedChanged(object sender, EventArgs e)
{
    if (textBox1.Text != "" && textBox2.Text != "")
    {
        int a = int.Parse(textBox1.Text);
        int b = int.Parse(textBox2.Text);
        textBox3.Text = (a - b).ToString();
    }
}

private void RadioButton3_CheckedChanged(object sender, EventArgs e)
{
    if (textBox1.Text != "" && textBox2.Text != "")
    {
        int a = int.Parse(textBox1.Text);
        int b = int.Parse(textBox2.Text);
        textBox3.Text = (a * b).ToString();
    }
}
}

```

结果如图 2-7 所示。



图 2-7 单选按钮的应用

5.5 文本框控件的应用

本例是教材第 7 章 P174 的内容，主要要求与流程请参照教材，这里只提醒与指导需要注意的事项。

(1) Form5_Load 的功能块，需要在 Form5.cs[设计]界面双击 Form 标题框打开，并在大括号内输入相应语句。button1_Click 的功能块需要在 Form5.cs[设计]界面双击 button1 打开，并在大括号内输入相应语句。

(2) 修改 Program.cs 中的 new Form 后的数字，本例应修改为 Application.Run(new Form5());

```
static void Main()
{
    Application.EnableVisualStyles();
    Application.SetCompatibleTextRenderingDefault(false);
    Application.Run(new Form5());
}
```

(3) 示例代码如下。

```
private void Button1_Click(object sender, EventArgs e)
{
    if (textBox1.SelectedText != "")
    {
        textBox2.Text = "选择文字长度:" +
textBox1.SelectedText.Length.ToString();
        textBox2.Text += "\r\n" + "选择的文字:" + textBox1.SelectedText;
    }
}

private void Form5_Load(object sender, EventArgs e)
{
    textBox1.Text = "C#是一种安全的稳定的、简单的、优雅的," +
        "由C和C++衍生出来的面向对象的编程语言"; //为避免字符串过长，而在适当位置
    textBox2.Text = "";
}
```

打断

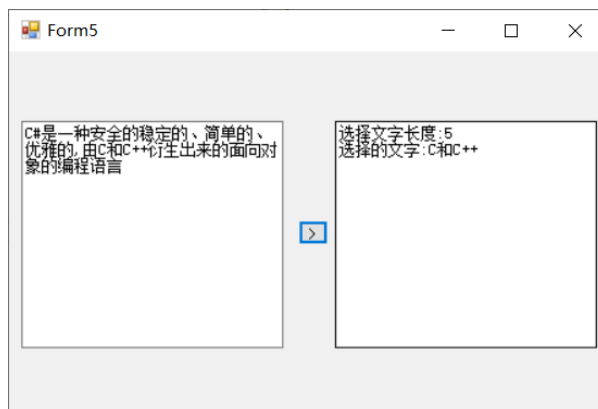


图 2-8 文本框控件的应用

5.6 列表控件的应用

本例是教材第 7 章 P178 的内容，主要要求与流程请参照教材，这里只提醒与指导需要注意的事项。

(1) Form6_Load 的功能块, 需要在 Form6.cs[设计]界面双击 Form 标题框打开, 并在大括号内输入相应语句。button1_Click 的功能块需要在 Form6.cs[设计]界面双击 button1 打开, 并在大括号内输入相应语句。

(2) 修改 Program.cs 中的 new Form 后的数字, 本例应修改为 Application.Run(new Form6());

```
static void Main()
{
    Application.EnableVisualStyles();
    Application.SetCompatibleTextRenderingDefault(false);
    Application.Run(new Form6());
}
```

(3) 示例代码如下。

```
private void Form6_Load(object sender, EventArgs e)
{
    checkedListBox1.Items.Add("中国"); checkedListBox1.Items.Add("美国");
    checkedListBox1.Items.Add("俄罗斯"); checkedListBox1.Items.Add("英国");
    checkedListBox1.Items.Add("法国"); checkedListBox1.CheckOnClick = true;
}

private void Button1_Click(object sender, EventArgs e)
{
    foreach (object item in checkedListBox1.CheckedItems)
        listBox1.Items.Add(item);
}
```

(4) 结果如图 2-9 所示。

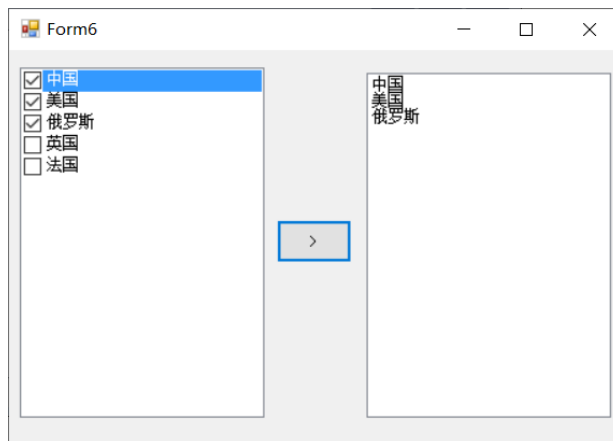


图 2-9 列表控件的应用

5.7 组合框控件的应用

本例是教材第 7 章 P180 的内容, 主要要求与流程请参照教材, 这里只提醒与指导需要注意的事项。

(1) Form7_Load 的功能块, 需要在 Form7.cs[设计]界面双击 Form 标题框打开, 并在大括号内输入相应语句。button1_Click 的功能块需要在 Form7.cs[设计]界面双击 button1 打开, 并在大括号内输入相应语句。

(2) 修改 Program.cs 中的 new Form 后的数字, 本例应修改为 Application.Run(new Form7());

```
static void Main()
{
    Application.EnableVisualStyles();
    Application.SetCompatibleTextRenderingDefault(false);
    Application.Run(new Form7());
}
```

(3) 示例代码如下。

```
private void Form7_Load(object sender, EventArgs e)
{
    comboBox1.Items.Add(5); comboBox1.Items.Add(8);
    comboBox1.Items.Add(12); comboBox1.Items.Add(21);
    comboBox2.Items.Add(3); comboBox2.Items.Add(6);
    comboBox2.Items.Add(9); comboBox2.Items.Add(12);
    label2.Text = "";
}

private void Button1_Click(object sender, EventArgs e)
{
    if ((comboBox1.SelectedIndex >= 0) && (comboBox2.SelectedIndex >= 0))
    {
        int n = int.Parse(comboBox1.SelectedItem.ToString());
        int m = int.Parse(comboBox2.SelectedItem.ToString());
        label2.Text = "计算结果:" + (n + m).ToString();
    }
    else
        label2.Text = "提示:没有选择合适的数字";
}
```

(4) 结果如图 2-10 所示。



图 2-10 组合框控件的应用

5.8TreeView 控件的应用

本例是教材第 7 章 P184 的内容，主要要求与流程请参照教材，这里只提醒与指导需要注意的事项。

(1) Form8_Load 的功能块，需要在 Form8.cs[设计]界面双击 Form 标题框打开，并在大括号内输入相应语句。TreeView1_AfterSelect 的功能块需要在 Form8.cs[设计]界面双击 TreeView1 打开，并在大括号内输入相应语句。

(2) 修改 Program.cs 中的 new Form 后的数字，本例应修改为 Application.Run(new Form8());

```
static void Main()
{
    Application.EnableVisualStyles();
    Application.SetCompatibleTextRenderingDefault(false);
    Application.Run(new Form8());
}
```

(3) 示例代码如下。

```
private void Form8_Load(object sender, EventArgs e)
{
    treeView1.Indent = 50;
    treeView1.Nodes.Add("计算机系");
    treeView1.Nodes[0].Nodes.Add("-一年级");
    treeView1.Nodes[0].Nodes[0].Nodes.Add("1501班");
    treeView1.Nodes[0].Nodes[0].Nodes.Add("1502班");
    treeView1.Nodes[0].Nodes.Add("-二年级");
    treeView1.Nodes[0].Nodes[1].Nodes.Add("1401班");
    treeView1.Nodes[0].Nodes[1].Nodes.Add("1402班");
    treeView1.Nodes.Add("电子工程系");
    treeView1.Nodes[1].Nodes.Add("-一年级");
    treeView1.Nodes[1].Nodes[0].Nodes.Add("1503班");
    treeView1.Nodes[1].Nodes[0].Nodes.Add("1504班");
    treeView1.Nodes[1].Nodes.Add("-二年级");
    treeView1.Nodes[1].Nodes[1].Nodes.Add("1403班");
    treeView1.Nodes[1].Nodes[1].Nodes.Add("1404班");
    treeView1.ExpandAll(); //展开所有节点
}

private void TreeView1_AfterSelect(object sender, TreeViewEventArgs e)
{
    TreeNode node = treeView1.SelectedNode;
    if (node.Level == 0)
        label1.Text = "你的选择:" + treeView1.SelectedNode.Text;
    else if (node.Level == 1)
    {
        TreeNode pnode = node.Parent;
```

```

        label1.Text = "你的选择:" + pnode.Text + "\\\" + node.Text;
    }
    else if (node.Level == 2)
    {
        TreeNode pnode = node.Parent;
        TreeNode ppnode = pnode.Parent;
        label1.Text="你的选择:" + pnode.Text + "\\\" + pnode.Text+"\" + node.
Text;
    }
}
}

```

(4) 结果如图 2-11 所示。

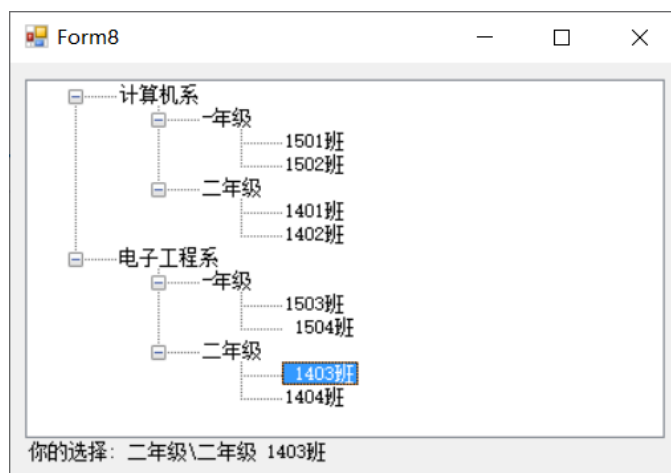


图 2-11 TreeView 控件的应用

5.9 ListView 控件的应用

本例是教材第 7 章 P187 的内容，主要要求与流程请参照教材，这里只提醒与指导需要注意的事项。

(1) 教材 P186 最下面倒数第 2 行，“和一个列表控件 treeView1。”有误，应修改为“和一个列表控件 ListView1。”，即添加的是 ListView 而不是 treeView。

(2) Form9_Load 的功能块，需要在 Form9.cs[设计]界面双击 Form 标题框打开，并在大括号内输入相应语句。button1_Click 的功能块需要在 Form9.cs[设计]界面双击 button1 打开，并在大括号内输入相应语句。

(3) 修改 Program.cs 中的 new Form 后的数字，本例应修改为 Application.Run(new Form9());

```

static void Main()
{
    Application.EnableVisualStyles();
    Application.SetCompatibleTextRenderingDefault(false);
    Application.Run(new Form9());
}

```

(4) 示例代码如下。

```
private void Form9_Load(object sender, EventArgs e)
{
    ListViewItem itemx = new ListViewItem();
    listView1.CheckBoxes = true;
    listView1.View = View.Details;
    listView1.Columns.Add("课程名", 140, HorizontalAlignment.Center); //添加3个
列
    listView1.Columns.Add("学分", 80, HorizontalAlignment.Center);
    listView1.Columns.Add("上课学期", 80, HorizontalAlignment.Center);
    itemx = listView1.Items.Add("计算机导论"); //添加第1个ListItem对象
    itemx.SubItems.Add("3"); //添加一个子项
    itemx.SubItems.Add("1学期"); //添加一个子项
    itemx = listView1.Items.Add("C语言"); //添加第2个ListItem对象
    itemx.SubItems.Add("3"); //添加一个子项
    itemx.SubItems.Add("2学期"); //添加一个子项
    itemx = listView1.Items.Add("数据结构"); //添加第3个ListItem对象
    itemx.SubItems.Add("4"); //添加一个子项
    itemx.SubItems.Add("3学期"); //添加一个子项
    itemx = listView1.Items.Add("数据库"); //添加第4个ListItem对象
    itemx.SubItems.Add("3"); //添加一个子项
    itemx.SubItems.Add("4学期"); //添加一个子项
    itemx = listView1.Items.Add("操作系统"); //添加第5个ListItem对象
    itemx.SubItems.Add("3"); //添加一个子项
    itemx.SubItems.Add("5学期"); //添加一个子项
    itemx = listView1.Items.Add("软件工程"); //添加第6个ListItem对象
    itemx.SubItems.Add("3"); //添加一个子项
    itemx.SubItems.Add("5学期"); //添加一个子项
    label1.Text = "";
}

private void Button1_Click(object sender, EventArgs e)
{
    string str = "你的选择:";
    foreach (ListViewItem item in listView1.Items)
        if (item.Checked)
            str += item.Text + " ";
    label1.Text = str;
}
```

(5) 结果如图 2-12 所示。

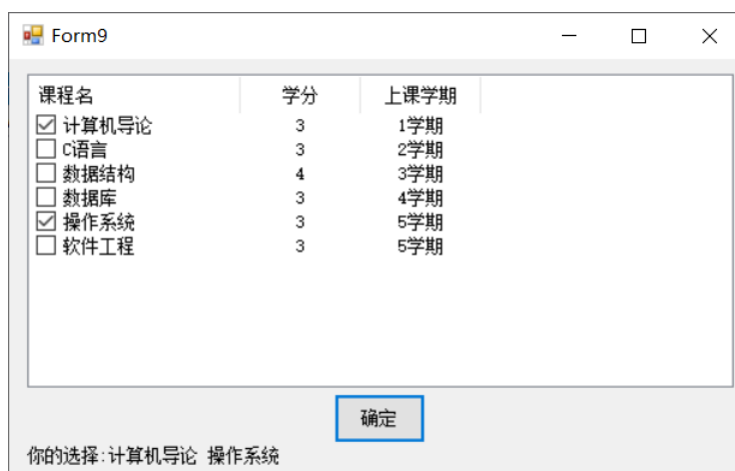


图 2-12 ListView 控件的应用

5.10 定时器控件的应用

本例是教材第 7 章 P189 的内容，主要要求与流程请参照教材，这里只提醒与指导需要注意的事项。

(1) Form10_Load 的功能块，需要在 Form10.cs[设计]界面双击 Form 标题框打开，并在大括号内输入相应语句。Timer1_Tick 的功能块需要在 Form10.cs[设计]界面双击 Timer1 打开，并在大括号内输入相应语句。

(2) 修改 Program.cs 中的 new Form 后的数字，本例应修改为 Application.Run(new Form10());

```
static void Main()
{
    Application.EnableVisualStyles();
    Application.SetCompatibleTextRenderingDefault(false);
    Application.Run(new Form10());
}
```

(3) 示例代码如下。

```
private void Timer1_Tick(object sender, EventArgs e)
{
    textBox1.Text = DateTime.Now.ToString("h:mm:ss");
}

private void Form10_Load(object sender, EventArgs e)
{
    textBox1.Text = DateTime.Now.ToString("h:mm:ss");
    timer1.Enabled = true;
    timer1.Interval = 1000;
}
```

(4) 结果如图 2-13 所示。

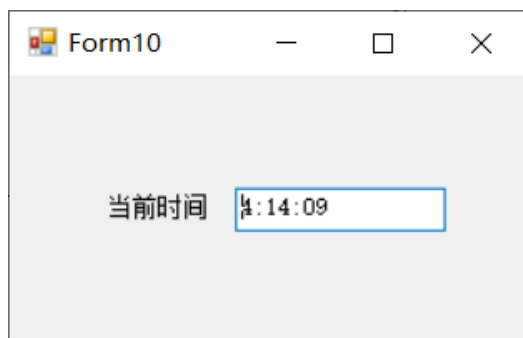


图 2-13 定时器控件的应用

5.11 打开和保存文件对话框的应用

本例是教材第 7 章 7.3 通用对话框 P192 的内容，主要要求与流程请参照教材，这里只提醒与指导需要注意的事项。

(1) `button.Enabled` 语句块是在教材上首先出现的语句块。在 Visual Studio 2019 默认打开 `Form11.cs` 中并无 `private void Form11_Load` 这一个语句块。要想调用出这一语句块、并让其正常发挥作用的方式是，在 `Form11.cs` [设计] 面板，双击 `Form11` 的标题框，界面就会自动跳转到 `Form11.cs`，并自动添加 `private void Form11_Load` 的语句块。我们需要做的，就是在语句块的大括号内添加如下所示两条语句。

```
private void Form11_Load(object sender, EventArgs e)
{
    button1.Enabled = true;
    button2.Enabled = false;
}
```

在 Visual Studio 2019 中，如不添加 `button1.Enabled` 的参数，默认即为 `true`，可以做一下尝试。在这两条语句添加之后，初始界面“打开”按钮即为活动的，“另存为”按钮即为灰色的，且点击无反应。如图 2-14 所示。

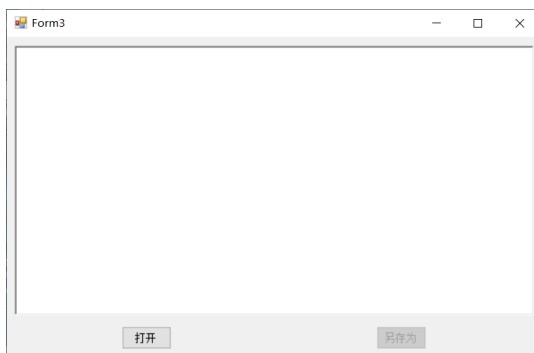


图 2-14 活动的“打开”按钮与未激活的“另存为”按钮

(2) `private void Button1_Click(object sender, EventArgs e)` 语句框的添加同样是双击 `button1`，手动输入是无效的。示例代码如下所示。为实现所需功能，要做的就是将大括号内的全部代码，“复制”到 Visual Studio 2019 对应的大括号内。

```
private void Button1_Click(object sender, EventArgs e)
{
    openFileDialog1.FileName = "";
    openFileDialog1.Filter = "RTF File(*.rtf) | *.rtf|TXT FILE(*.txt) | *.txt";
    openFileDialog1.ShowDialog();
    if (openFileDialog1.FileName != "")
        switch (openFileDialog1.FilterIndex)
        {
            case 1: //选择的是.rtf类型
                richTextBox1.LoadFile(openFileDialog1.FileName,
RichTextBoxStreamType.RichText);
                break;
            case 2: //选择的是.txt类型
                richTextBox1.LoadFile(openFileDialog1.FileName,
RichTextBoxStreamType.UnicodePlainText);
                break;
        }
        button2.Enabled = true;
    }
}
```

值得注意的是，在语句块最后有语句 `button2.Enabled = true`；将 `button2` 激活。可以自行测试，如果将这一语句删除（最快的方法就是在语句之前加上 `//` 变为注释，方便反复修改），`button2`（即另存为）按钮将永远不会产生作用。

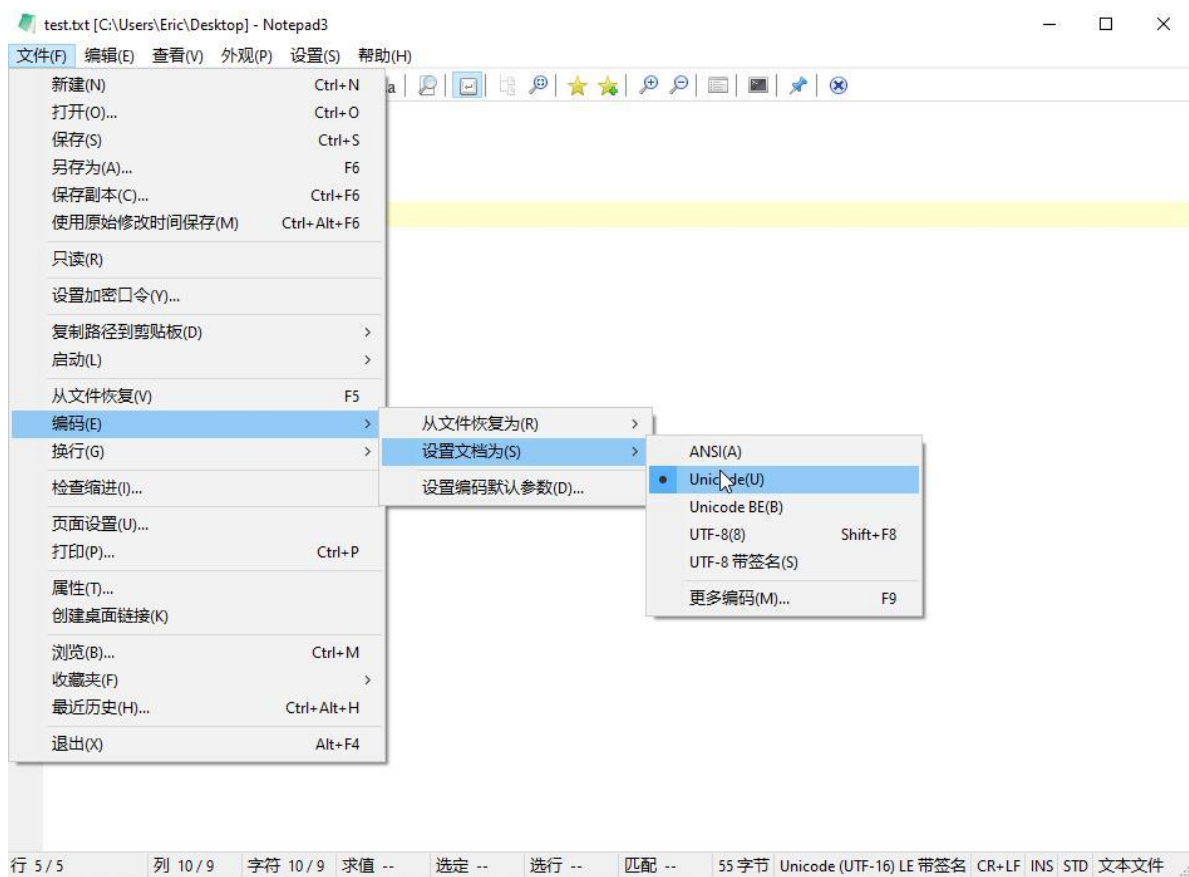
`private void Button2_Click(object sender, EventArgs e)` 所代表的 `button2` 的功能也和 `button1` 相同，都是需要在设计界面双击按钮，由 Visual Studio 2019 自动创建代码块，否则代码无效（即，点击按钮，并不会产生效果。可以自行尝试）。

```
private void Button2_Click (object sender, EventArgs e)
{
    saveFileDialog1.Filter = "RTF File(*.rtf) | * .rtf|TXT FILE(*.txt) |*.txt";
    if (saveFileDialog1.ShowDialog() == System.Windows.Forms.DialogResult.OK)
        switch (openFileDialog1.FilterIndex)
        {
            case 1://选择的是.rtf类型
                richTextBox1.SaveFile(saveFileDialog1.FileName,
RichTextBoxStreamType.RichText);
                break;
            case 2://选择的是.txt类型
                richTextBox1.SaveFile(saveFileDialog1.FileName,
RichTextBoxStreamType.UnicodePlainText);
                break;
        }
    }
```

```
}
}
```

(3) 关于.rtf 文件。.rtf 可以在 word 中创建, 并另存为.rtf 即可。也可以将.doc/.docx 文件的后缀修改为.rtf, 但并不推荐这样做, 因为我们的程序并不支持这样的格式(虽然 word 支持)。

(4) 关于.txt 文件。由于 RichTextBox 与.txt 的默认编码不同, 打开包含中文字符的.txt 文件时, 常会出现乱码。我们创建.txt 后, 请注意需将编码改为 Unicode 再输入中文。修改.txt 文件的编码, 可通过 Notepad3 等软件实现。如下图



(5) 以上就是关于本小节练习需要注意的内容。其他未言及之处都可以在教材中得到指点。

5.12 下拉式菜单的应用

本例是教材第 7 章 P196 的内容, 主要要求与流程请参照教材, 这里只提醒与指导需要注意的事项。这里并不沿用教材上的 Proj2 Form1, 而继续在同一个项目中新建 From12, 下面同理。

(1) 本例的操作要严格操作流程, 否则会导致错误的操作, 使结果无法输出。

(2) 在拖放三个标签(label)、三个文本框(textBox)、一个 MenuStrip 控件之后, 进行下列操作。

(i) 修改 label 的“属性”->“Text”为对应的名称。

(ii) 在 MenuStrip 控件顺次添加“运算”、“加法”、“减法”、“乘法”、“除法”。鼠标右击“除法”, 在弹出的菜单栏中选择“插入”, 并单击“Separator”, 即可在“除法”栏之上添加分隔线。如图 2-15 所示。(注: 这里 i 和 ii 的顺序可以颠倒。)

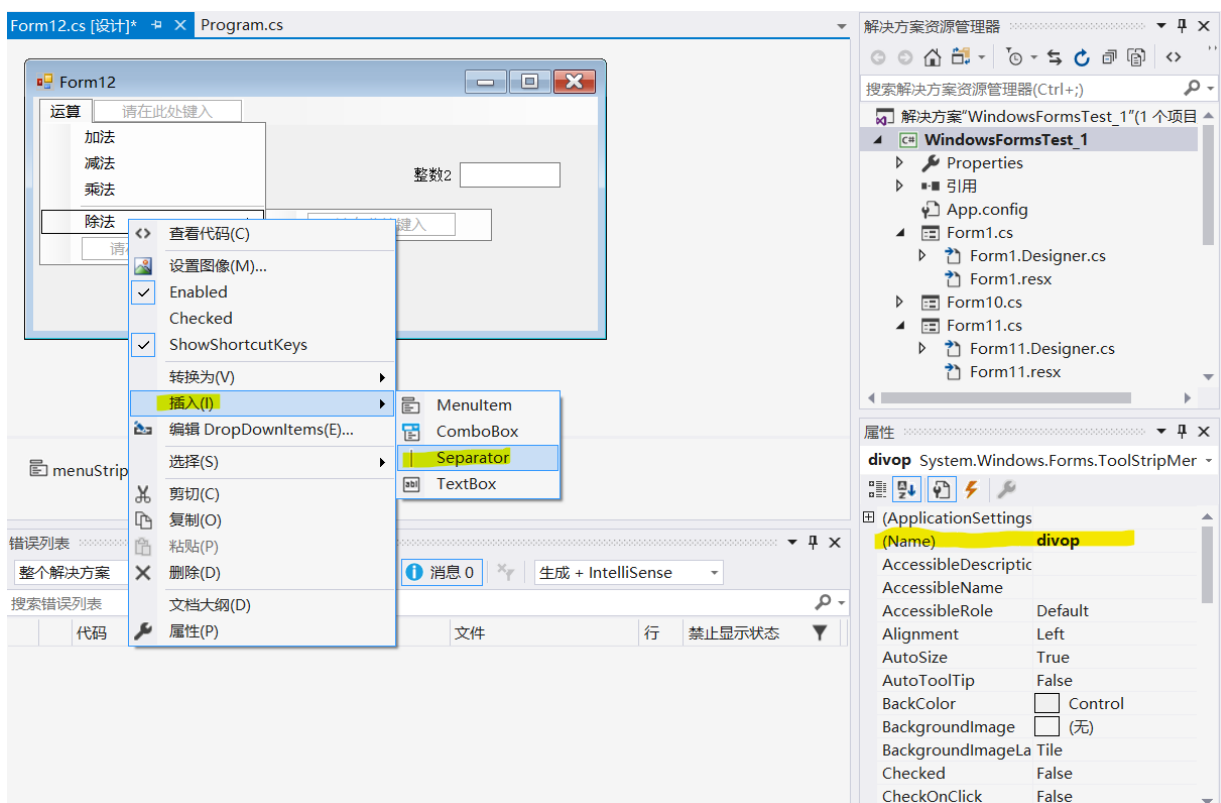


图 2-15 MenuStrip 的设置、分隔线的插入、Name 的修改

(iii) 选择“运算”、“加法”、“减法”、“乘法”、“除法”的属性，分别将其(Name)栏修改为 op, addop, subop, mulop, divop。这一项操作非常重要，只有在完成本部操作之后，才可以进行（3）的操作！

(3) 双击“运算”、“加法”、“减法”、“乘法”、“除法”5个菜单项，分别打开 Op_Click, Addop_Click, Subop_Click, Mulop_Click, Divop_Click 的功能块，在相应控制块内填入相应的语句。

示例代码如下。

```
private void Op_Click(object sender, EventArgs e)
{
    if (textBox2.Text == "" || Convert.ToInt16(textBox2.Text) == 0)
        divop.Enabled = false;
    else
        divop.Enabled = true;
}

private void Addop_Click(object sender, EventArgs e)
{
    int n;
    n = Convert.ToInt16(textBox1.Text) + Convert.ToInt16(textBox2.Text);
    textBox3.Text = n.ToString();
}
```



```
private void Subop_Click(object sender, EventArgs e)
{
    int n;
    n = Convert.ToInt16(textBox1.Text) - Convert.ToInt16(textBox2.Text);
    textBox3.Text = n.ToString();
}

private void Mulop_Click(object sender, EventArgs e)
{
    int n;
    n = Convert.ToInt16(textBox1.Text) * Convert.ToInt16(textBox2.Text);
    textBox3.Text = n.ToString();
}

private void Divop_Click(object sender, EventArgs e)
{
    int n;
    n = Convert.ToInt16(textBox1.Text) / Convert.ToInt16(textBox2.Text);
    textBox3.Text = n.ToString();
}
}
```

(4) 修改 Program.cs 中的 new Form 后的数字，本例应修改为 Application.Run(new Form12());

```
static void Main()
{
    Application.EnableVisualStyles();
    Application.SetCompatibleTextRenderingDefault(false);
    Application.Run(new Form12());
}
```

(5) 示例结果如图 2-16 所示。

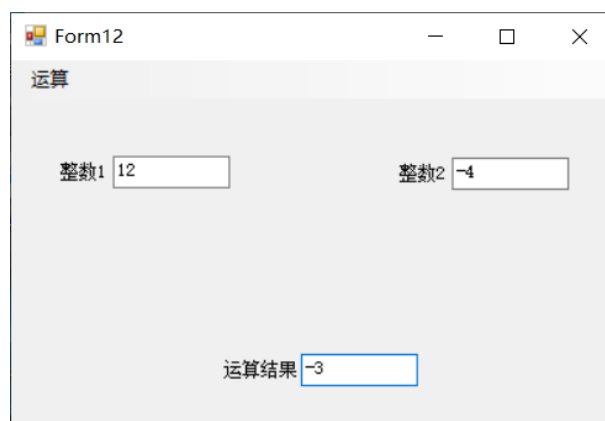


图 2-16 下拉式菜单的应用

5.13 弹出式菜单的应用

本例是教材第7章 P198 的内容，主要要求与流程请参照教材，这里只提醒与指导需要注意的事项。这里继续在同一个项目中新建 Form13，下面同理。

(1) 本例的操作要严格操作流程，否则会导致错误的操作，使结果无法输出。

(2) 在拖放三个标签 (label)、三个文本框 (textBox)、一个 ContextMenuStrip 控件之后，进行下列操作。

(i) 修改 label 的“属性”->“Text”为对应的名称。

(ii) 单击在 Form13 下方的 ContextMenuStrip1 控件，在“属性”->“(Name)”中修改为“op”。之后，在菜单栏顺次添加“加法”、“减法”、“乘法”、“除法”。鼠标右击“除法”，在弹出的菜单栏中选择“插入”，并单击“Separator”，即可在“除法”栏之上添加分隔线。如图 2-16 所示。(注：这里 i 和 ii 的顺序可以颠倒。)

(iii) 选择“加法”、“减法”、“乘法”、“除法”的属性，分别将其 (Name) 栏修改为 addop, subop, mulop, divop。这一项操作非常重要，只有在完成本部操作之后，才可以进行 (3) 的操作！

(3) 双击“加法”、“减法”、“乘法”、“除法”、“op”5 个菜单项，打开 Addop_Click, Subop_Click, Mulop_Click, Divop_Click, Op_Opening (要注意的是，在 Visual Studio 2019 中，双击 op 生成的是 Op_Opening 块，而不是 op_Opened 块) 的功能块，在相应控制块内填入相应的语句。

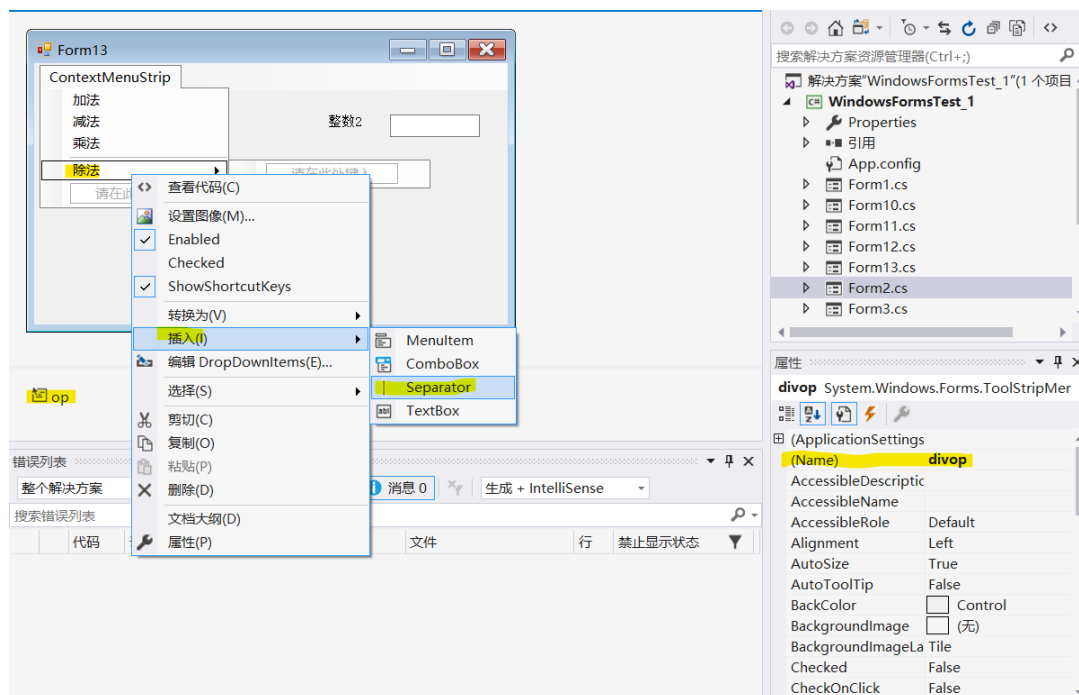


图 2-16 弹出式菜单的应用

(4) 示例代码如下。

```
private void Addop_Click(object sender, EventArgs e)
{
    int n;
    n = Convert.ToInt16(textBox1.Text) + Convert.ToInt16(textBox2.Text);
    textBox3.Text = n.ToString();
}

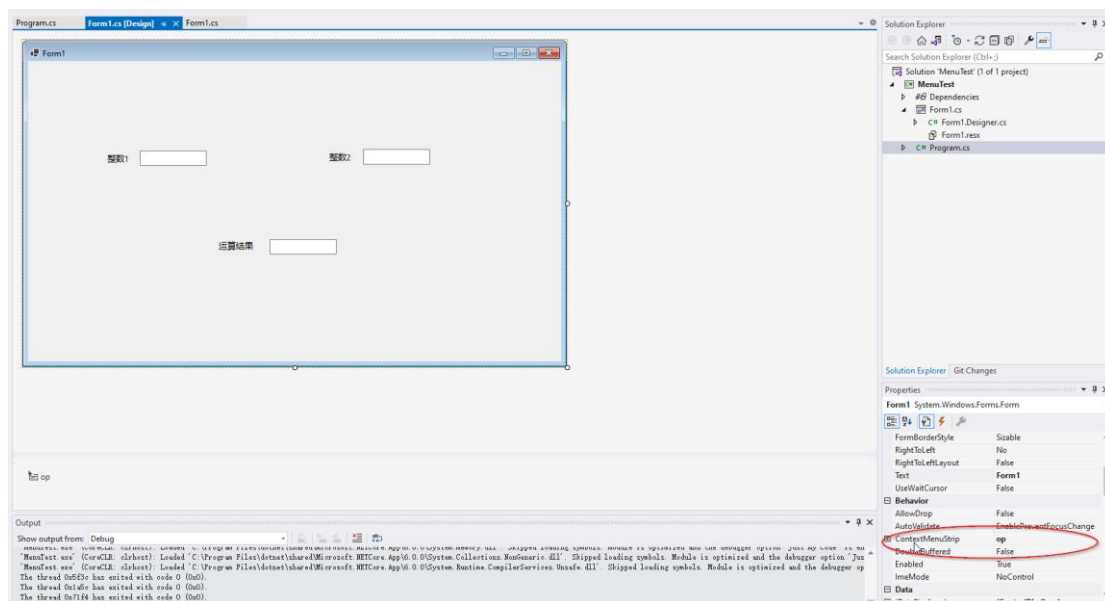
private void Subop_Click(object sender, EventArgs e)
{
    int n;
    n = Convert.ToInt16(textBox1.Text) - Convert.ToInt16(textBox2.Text);
    textBox3.Text = n.ToString();
}

private void Mulop_Click(object sender, EventArgs e)
{
    int n;
    n = Convert.ToInt16(textBox1.Text) * Convert.ToInt16(textBox2.Text);
    textBox3.Text = n.ToString();
}

private void Divop_Click(object sender, EventArgs e)
{
    int n;
    n = Convert.ToInt16(textBox1.Text) / Convert.ToInt16(textBox2.Text);
    textBox3.Text = n.ToString();
}

private void Op_Opening(object sender, CancelEventArgs e)
{
    if (textBox2.Text == "" || Convert.ToInt16(textBox2.Text) == 0)
        divop.Enabled = false;
    else
        divop.Enabled = true;
}
```

(5) 在窗体的属性处，将弹出式菜单与窗体绑定



结果如图 2-17 所示。



图 2-17 弹出式菜单的应用

5.14 通过静态字段传递数据

本例是教材第 7 章 P204 的内容，主要要求与流程请参照教材，这里只提醒与指导需要注意的事项。

- (1) 按照教材的指引，新建 class.cs，并输入相应代码。
- (2) 这里继续在同一个项目中新建 Windows 窗口项目，并在名称栏将默认的 Form14.cs 修改为 MForm.cs。下文 SForm.cs 的创建同理。

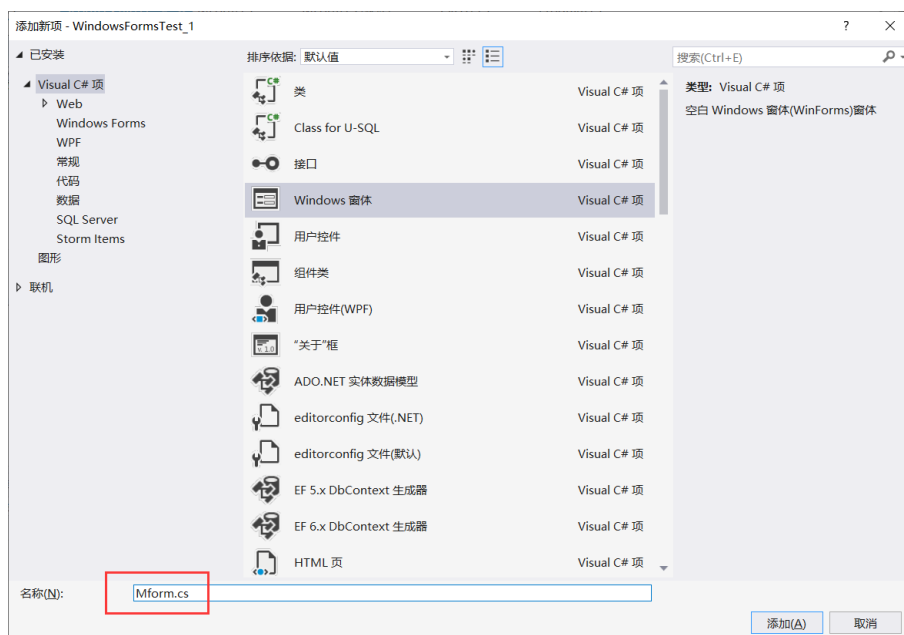


图 2-18 MForm 的创建

(3) 新建 MForm 后，创建 label 两个，textBox 两个，button 一个。并修改 label 和 button 的 Text 属性，让其显示指定的文字。

(4) MForm 中 Button1_Click 代码块要双击 MForm 中的 button1 打开。

MForm 示例代码如下。

```
private void Button1_Click(object sender, EventArgs e)
{
    TempData.mynum = int.Parse(textBox1.Text);
    //将文本框textBox1的值转换为整数后保存在静态字段mynum中
    TempData.mystr = textBox2.Text;
    //将文本框textBox2的值保存在静态字段mystr中
    Form myform = new SForm();
    myform.ShowDialog();
}
```

(5) 新建 SForm 窗体。SForm 中 SForm_Load、Button1_Click 代码块分别要双击 SForm 的标题栏、双击 SForm 中的 button1 打开。

SForm 示例代码如下。

```
private void SForm_Load(object sender, EventArgs e)
{
    textBox1.Text = TempData.mynum.ToString();
    //读出静态字段mynum中的数据
    textBox2.Text = TempData.mystr;
    //读出静态字段mystr中的数据
}

private void Button1_Click(object sender, EventArgs e)
```

```

{
    this.Close();
}

```

(6) 修改 Program.cs, 修改为 Application.Run(new MForm ());。示例运行结果如下。

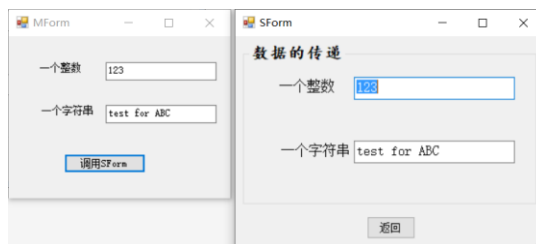


图 2-19 通过静态字段传递数据

5.15 通过构造函数传递数据

本例是教材第 7 章 P205 的内容, 主要要求与流程请参照教材, 这里只提醒与指导需要注意的事项。

(1) class.cs 代码即可以沿用 5.14 中的代码, 不做修改。

(2) 与 5.14 相同, 这里继续在同一个项目中新建 Windows 窗口项目, 并在名称栏将默认的 Form14.cs 修改为 MForm1.cs。下文 SForm1.cs 的创建同理。

(3) 新建 MForm1 后, 创建 label 两个, textBox 两个, button 一个。并修改 label 和 button 的 Text 属性, 让其显示指定的文字。

(4) MForm1 中 MForm1_Load、Button1_Click 代码块分别要双击 MForm1 的标题栏、双击 MForm1 中的 button1 打开。

MForm1 示例代码如下。

```

private void Button1_Click(object sender, EventArgs e)
{
    int num = int.Parse(textBox1.Text);
    string str = textBox2.Text;
    Form myform = new SForm1(num, str);
    myform.ShowDialog();
}

```

(5) 新建 SForm1 窗体。SForm1 中 SForm1_Load、Button1_Click 代码块分别要双击 SForm 的标题栏、双击 SForm 中的 button1 打开。

public partial class SForm1 : Form 和 public SForm1 可以在 SForm1.cs 中修改 (需要添加修饰的字段)。

SForm 示例代码如下。

```

public partial class SForm1 : Form
{
    private int mynum;
    private string mystr;
    public SForm1(int num, string str)

```

```

{
    InitializeComponent();
    mynum = num;
    mystr = str;
}

private void SForm1_Load(object sender, EventArgs e)
{
    textBox1.Text = mynum.ToString();
    textBox2.Text = mystr;
}

private void Button1_Click(object sender, EventArgs e)
{
    this.Close();
}
}

```

(6) 修改 Program.cs, 修改为 Application.Run(new MForm1());。示例运行结果如下。

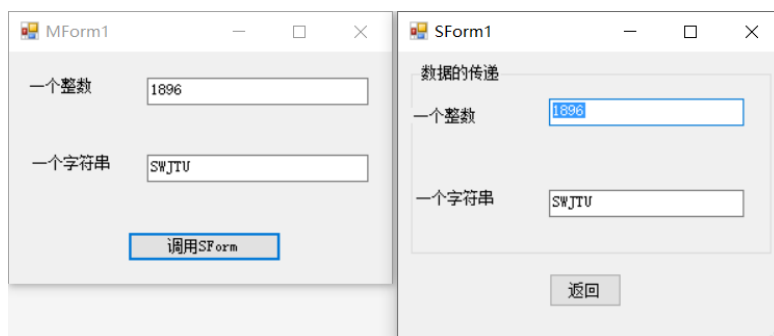


图 2-20 通过构造函数传递数据

5.16 多文档应用程序

本例是教材第 7 章 P207 的内容, 主要要求与流程请教材虽没有详细介绍, 但经过前面的练习相信你可以完成有关操作, 这里只提醒与指导需要注意的事项。

(1) 与 5.14 相同, 这里继续在同一个项目中新建 Windows 窗口项目, 并在名称栏将默认的 Form14.cs 修改为 Menu.cs。下文 Fun11.cs 的创建同理。

(2) 新建 Menu 窗体后, MenuStrip 的操作请直接参考 5.12 下拉菜单的操作。

(3) 将 Menu 窗体的 IsMdiContainer 设置为 True。如图 2-21 所示。要注意的是, 这一项是 Menu 窗体的属性, 鼠标左键单击 Menu 窗体的标题栏后, 右下角即可以显示。这一步操作也非常重要, 如果不执行这一步, 程序会报错, 不妨一试。

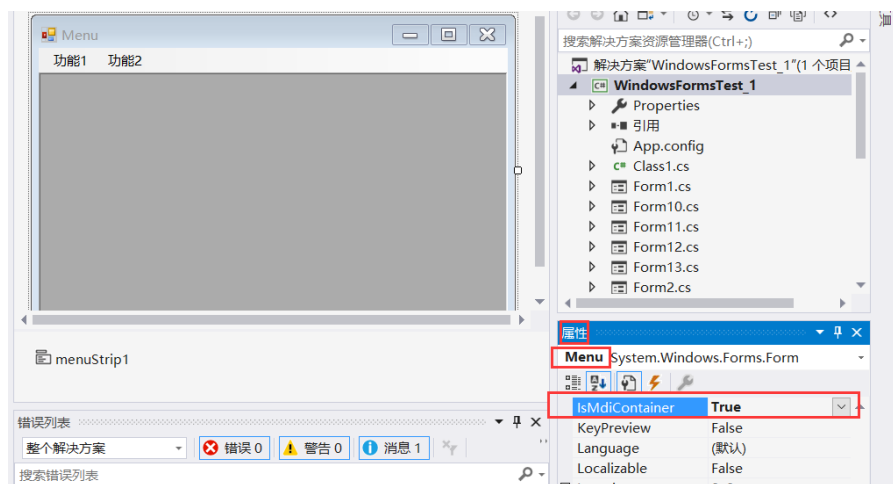


图 2-21IsMdiContainer 设置为 True

(4) 为了简便且不影响练习效果，遵循教材的操作，只为 fun11 设置子窗口即可。Menu 窗体应用的代码如下：

```
private void Fun11_Click(object sender, EventArgs e)
{
    Fun11 child = new Fun11(); //这里子窗口的名字直接使用了fun11。又因为Visual
    Studio 2019推荐大写，故写作了Fun11
    child.MdiParent = this;
    child.Show();
}
```

(5) 新建 Fun11 窗体，直接命名为 Fun11.cs。Fun11 窗体的设计非常简单。修改 Fun11 的“属性”->“StartPosition”为“CenterScreen”；拖入一个 label 即可，做出注释备注以后要完成的功能。

(6) 运行结果如图 2-22 所示。

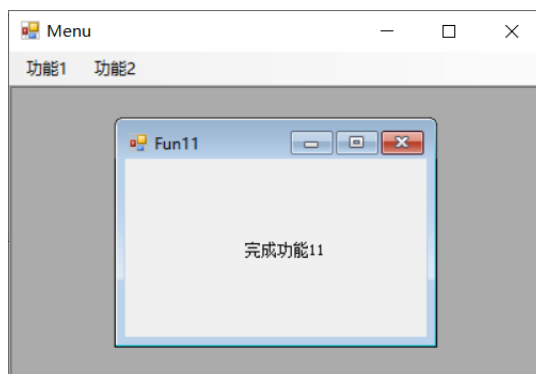


图 2-22 多文档应用程序

实验三 ADO.NET 访问数据库基础

一、实验目的

(1) 本章实验的内容主要涵盖教材 13 和 14 章的内容，通过上机实验以加深对语言和数据库的理解；

(2) 了解 ADO.NET 模型、ADO.NET 的数据访问对象、DataSet 对象的概况，了解数据控件。数据绑定、DataView 对象、DataGridview 控件。

二、实验内容

本实验主要包括如下部分的内容。

- 1 创建连接的方法
- 2 求 Product 表中指定分类的商品数
- 3 通过 SqlCommand 对象执行更新操作
- 4 SqlCommand 对象的 SQL 命令中参数的使用方法
- 5 存储过程的使用方法
- 6 SqlDataReader 对象的使用方法
- 7 DataSet 对象的使用方法
- 8 显示 Product 表的第一个记录
- 9 Product 表记录浏览
- 10 求指定分类的商品销售总数量和总金额
- 11 通过 BindingNavigator 控件实现对 Product 表中所有记录的浏览、添加和删除操作
- 12 使用 DataView 对象在列表框中按单价升序、库存数量降序排序显示所有商品记录
- 13 用 DataGridview 控件显示所有商品记录
- 14 Product 表的商品复杂查询
- 15 Product 表的商品修改

其中实验 1 和 2 是简单的引入部分。实验 3 基于第一章的内容。实验 4 和 5 基于第二章的内容。实验 6/7/8 基于第三章的内容。实验 9 和 10 基于第四章的内容。实验 11 和 12 基于第五章的内容。实验 13 基于第六章的内容。

请大家务必携带教材《C#语言与数据库技术基础教程》！

三、实验仪器、设备及材料

硬件：计算机；《C#语言与数据库技术基础教程》

软件：Windows 操作系统；Visual Studio 2019（推荐使用最新版本）；Microsoft SQL Server 2017；

MySQL；.NET Framework v4.7.2

四、实验原理

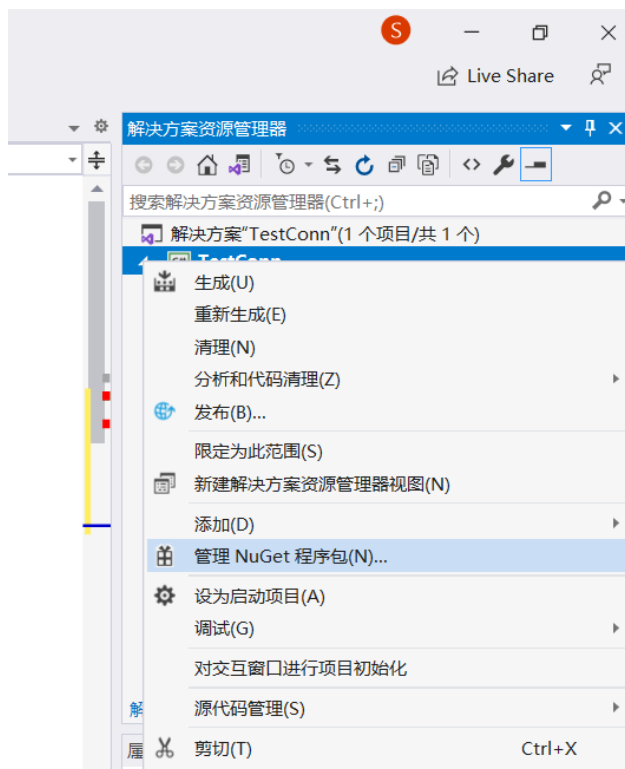
ADO.NET 是在 .NET Framework 上访问数据库的一组类库，它利用 .NET Data Provider（数据提供程序）以进行数据库的连接与访问，通过 ADO.NET，数据库程序设计人员能够很轻易地使用各种对象来访问符合自己需求的数据库内容。换句话说，ADO.NET 定义了一个数据库访问的标准接口，让提供数据库管理系统的各个厂商可以根据此标准开发对应的 .NET Data Provider，这样编写数据库应用程序的人员不必了解各类数据库底层运作的细节，只要学会 ADO.NET 所提供对象的模型，便可轻易地访问所有支持 .NET Data Provider 的数据库。所以 ADO.NET 是应用程序和数据源之间沟通的桥梁。

五、实验步骤

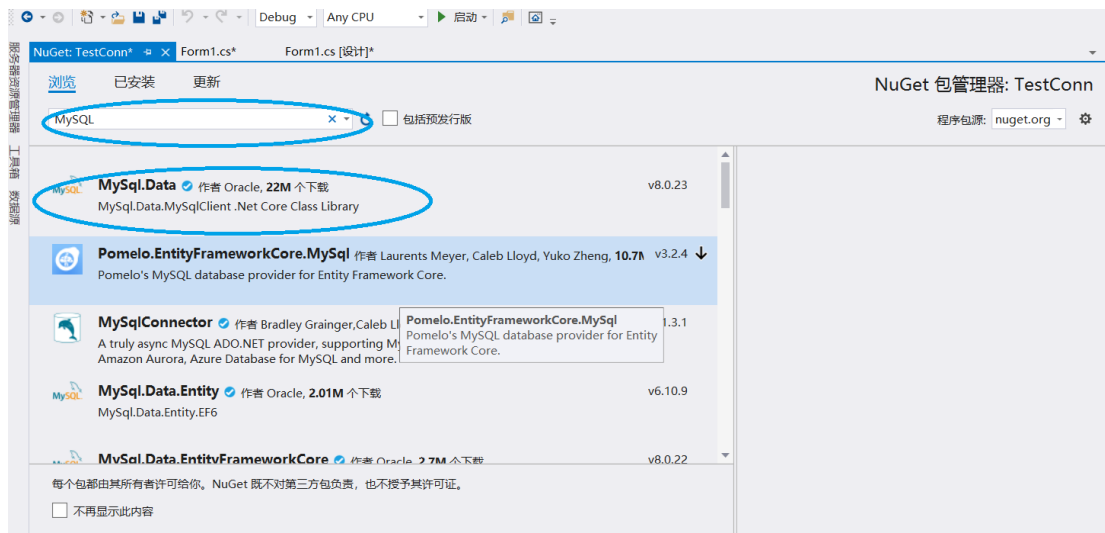
预备工作

Visual Studio 新建项目时默认没有加载 MySQL 的类库，需要先通过 NuGet 下载并加载类库，才能进行后续的连接和操作。安装 MySQL 类库（或其他类库）操作如下：

- 1) 在“解决方案资源管理器”窗口的项目处右键，打开“管理 NuGet 程序包”



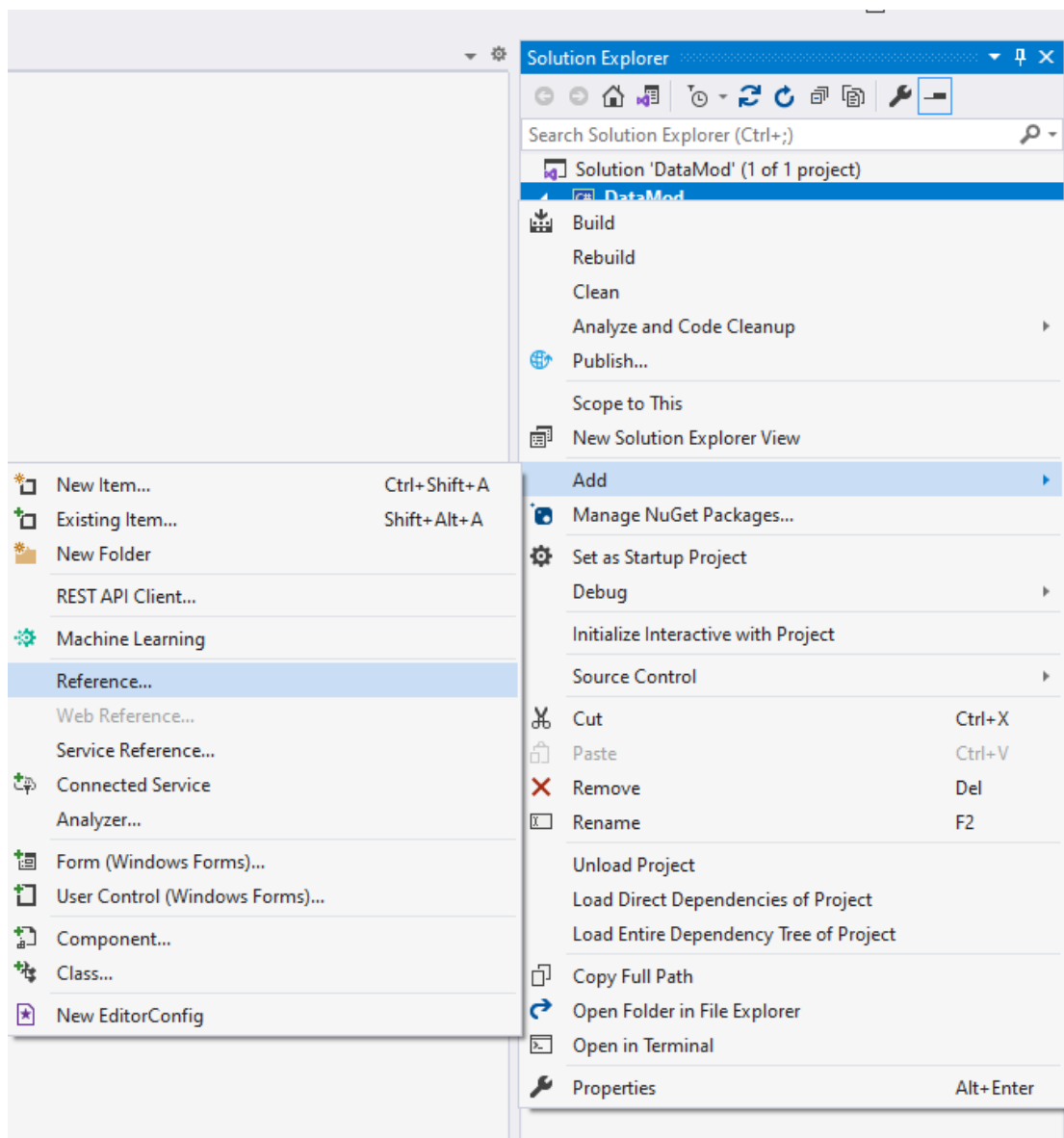
- 2) 寻找 MySQL.Data，并安装



使用旧版本 Visual Studio（如 VS10）的同学，可按照以下步骤操作：

（1）下载 Connector/NET 6.9.9，并安装。链接为 <https://dev.mysql.com/downloads/connector/net/>，点击 Archives，在 Product Version 处寻找 6.9.9，下载安装。如果电脑中有更高版本，可通过 MySQL Installer 卸载；如果无法卸载，请点击电脑左下角 ，输入“Run”，打开“运行”应用。输入“regedit”，打开注册表编辑器；使用快捷键 Ctrl+F，以“MySQL Connector Net”为关键词寻找所有高级版本使用的注册表并删除；然后在尝试安装 6.9.9 版本；

(2) 添加对应的引用；如截图所示。



(3) 通过“浏览”添加 MySQL 相关的类库，位置为“C:\Program Files (x86)\MySQL\mysql-connector-net-6.9.9\v4.0\Assemblies”

5.1 创建连接的方法

本例是教材第 13 章 P336【练一练】的内容，主要要求与流程请参照教材，这里只提醒与指导需要注意的事项。选择“Windows 窗体应用程序”模板，指定创建的位置（如 E:\Test_For_DataBase），自定义命名（如 WindowsFormsTest_ADO.NET），点击“确定”按钮，即完成了新建。

需要注意的一点是，本章的所有示例都需要在安装了 MySQL 的情况下完成。其他未尽事宜可以参考教材第九章。

(1) 在新建的 Form1 中，按照教材指示，添加 button 和 label。

(2) Button1_Click 功能块同理，是双击 button1 进入。

(3) 在程序命名空间处加载本程序所需的类库；

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using MySql.Data;
using MySql.Data.MySqlClient;
```

(4) 需要引起注意的是字符串 mystr 的设置与打开登入 MySQL 时选择的参数有关。示例代码沿用了 MySQL 默认给出的 Server 的名称。在实验中，同学们可以选择自己想要取用的名称。

示例代码如下：

```
private void Button1_Click(object sender, EventArgs e)
{
    string mystr;
    MySqlConnection myconn = new MySqlConnection();
    mystr = "server=localhost;user=root;database=smk;port=3306;password=****";
    //其中server=localhost说明连接的是本机默认的数据库；user=root说明连接的用户
    身份为root；port为服务器对应的网络端口，在安装MySQL时配置，默认是3306；password为预设的
    用户密码；对于部分电脑，除了需要界定server/user/database/port/password外，可能需要添
    加'Integrated Security=yes'选项。

    myconn.ConnectionString = mystr;
    myconn.Open();
    if (myconn.State == System.Data.ConnectionState.Open)
        label1.Text = "成功连接到SQL Server数据库";
    else
        label1.Text = "不能连接到SQL Server数据库";
    myconn.Close();
}
```

(5) 修改 Program.cs 文件中的启动项为当前窗体。

```
static void Main()
{
    Application.EnableVisualStyles();
    Application.SetCompatibleTextRenderingDefault(false);
    Application.Run(new Form1());
}
```

运行结果如图 3-1 所示。

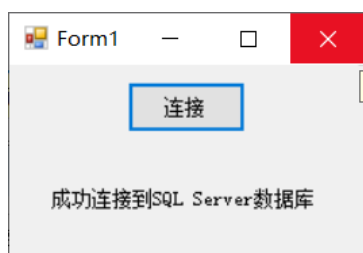


图 3-1 创建连接的方法

5.2 求 Product 表中指定分类的商品数

本例是教材第 13 章 P339 【练一练】的内容，主要要求与流程请参照教材，这里只提醒与指导需要注意的事项。

(1) 在新建的 Form2 中，按照教材指示，添加 button 和 label。

(2) Button1_Click 功能块同理，是双击 button1 进入。

(3) 需要引起注意的是字符串 mystr 的设置与打开登入 MySQL 时选择的参数有关。示例代码沿用了 MySQL 默认给出的服务器的名称。在实验中，同学们可以选择自己想要取用的名称。

示例代码如下：

```
private void Button1_Click(object sender, EventArgs e)
{
    string mystr, mysql;
    int num;
    MySqlConnection myconn = new MySqlConnection();
    MySqlCommand mycmd = new MySqlCommand();
    mystr = "server=localhost;user=root;database=smk;port=3306;password=****";
    myconn.ConnectionString = mystr;
    myconn.Open();
    mysql = "SELECT COUNT(*) FROM Product WHERE 分类='" + textBox1.Text.Trim()
+ "'";

    mycmd.CommandText = mysql;
    mycmd.Connection = myconn;
    num = int.Parse(mycmd.ExecuteScalar().ToString());
    if (num == 0)
        label2.Text = "你输入的分类不存在";
    else
        label2.Text = textBox1.Text.Trim() + "分类的商品数为" + num.ToString();
    myconn.Close();
}
```

(4) 在程序命名空间处加载本程序所需的类库，修改 Program.cs 中的启动项。运行结果如图 3-2 所示。

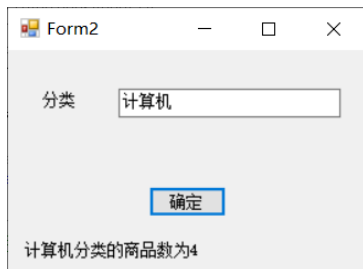


图 3-2 求 Product 表中制定分类的商品数

5.3 通过 MySqlCommand 对象执行更新操作

本例是教材第 13 章 P340【练一练】的内容，主要要求与流程请参照教材，这里只提醒与指导需要注意的事项。

(1) 在新建的 Form2 中，按照教材指示，添加 button 和 label。

(2) Button1_Click 功能块同理，是双击 button1 进入。

(3) 需要引起注意的是字符串 mystr 的设置与打开登录 MySQL 时选择的参数有关。示例代码沿用了 MySQL 默认给出的 Server 的名称。在实验中，同学们可以选择自己想要取用的名称。

示例代码如下：

```
string mystr, mysql;
MySqlConnection myconn = new MySqlConnection();
MySqlCommand mycmd = new MySqlCommand();
mystr = "server=localhost;user=root;database=smk;port=3306;password=****";
myconn.ConnectionString = mystr;
myconn.Open();
mysql = "INSERT INTO Product VALUES(' 3001',' 中信3001',' 通信',' 手机
',1800,100)";
mycmd.CommandText = mysql;
mycmd.Connection = myconn;
mycmd.ExecuteNonQuery();
label1.Text = "(1)插入编号为3001的商品记录";
mysql = "DELETE Product WHERE 商品编号=' 3001'";
mycmd.CommandText = mysql;
mycmd.Connection = myconn;
mycmd.ExecuteNonQuery();
label2.Text = "(2)删除编号为3001的商品记录";
myconn.Close();
```

(4) 添加 using MySql.Data.MySqlClient; using System.Configuration; 并修改 Program.cs 中的启动项。运行结果如图 3-3 所示。

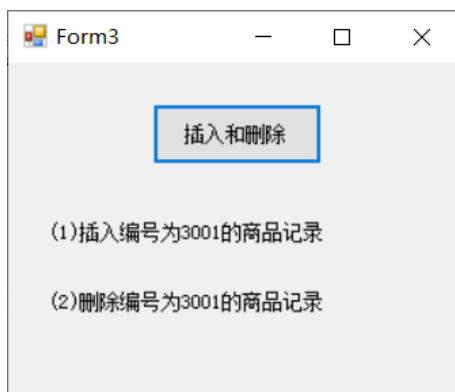


图 3-3 通过 SqlCommand 对象执行更新操作

5.4 MySqlCommand 对象的 SQL 命令中参数的使用方法

本例是教材第 13 章 P342【练一练】的内容，主要要求与流程请参照教材，这里只提醒与指导需要注意的事项。

(1) 在新建的 Form4 中，按照教材指示，添加 button 和 label。

(2) Button1_Click 功能块同理，是双击 button1 进入。

(3) 需要引起注意的是字符串 mystr 的设置与打开登录 MySQL 时选择的参数有关。示例代码沿用了 MySQL 默认给出的 Server 的名称。在实验中，同学们可以选择自己想要取用的名称。

示例代码如下：

```
private void Button1_Click(object sender, EventArgs e)
{
    string mystr, mysql;
    MySqlConnection myconn = new MySqlConnection();
    MySqlCommand mycmd = new MySqlCommand();
    mystr = "server=localhost;user=root;database=smk;port=3306;password=****";
    myconn.ConnectionString = mystr;
    myconn.Open();
    mysql = "SELECT AVG(单价) FROM Product WHERE 分类=@dj";
    mycmd.CommandText = mysql;
    mycmd.Connection = myconn;
    mycmd.Parameters.Add("@dj", MySqlDbType.VarChar, 10).Value
        = textBox1.Text;
    try
    {
        float avg = float.Parse(mycmd.ExecuteScalar().ToString());
        label2.Text = textBox1.Text.Trim() + "分类商品的平均单价为" +
avg.ToString();
    }
    catch (Exception ex)
    {
        label2.Text = "提示：输入的分类不存在";
    }
    myconn.Close();
}
```

(4) 修改 Program.cs 中的启动项为当前窗体。运行结果如图 3-4 所示。

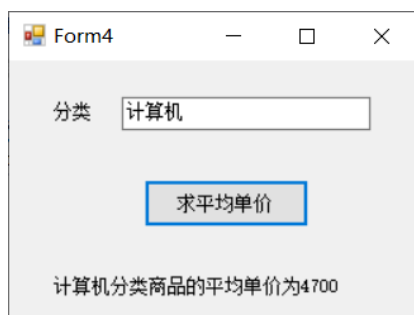


图 3-4SqlCommand 对象的 SQL 命令中参数的使用方法

5.5* (不做要求)存储过程的使用方法

传统行业，如金融/政府/企业使用 SQL Server/Oracle 较多，存储过程发挥的空间较大。然而，互联网行业用的比较少（如阿里巴巴的开发手册建议禁止使用存储过程）。

由于 MySQL 中存储过程的大部分功能可以被更灵活的脚本代替，比较鸡肋，此处不做要求。

5.6 MySqlConnection 对象的使用方法

本例是教材第 13 章 P336【练一练】的内容，主要要求与流程请参照教材，这里只提醒与指导需要注意的事项。

(1) 在新建的 Form6 中，按照教材指示，添加 button 和 label。

(2) Button1_Click 功能块同理，是双击 button1 进入。

(3) 需要引起注意的是字符串 mystr 的设置与打开登入 MySQL 时选择的参数有关。示例代码沿用了 MySQL 默认给出的 Server 的名称。在实验中，同学们可以选择自己想要取用的名称。

示例代码如下：

```
private void Button1_Click(object sender, EventArgs e)
{
    string mystr, mysql;
    MySqlConnection myconn = new MySqlConnection();
    MySqlCommand mycmd = new MySqlCommand();
    mystr = "server=localhost;user=root;database=smk;port=3306;password=***";
    myconn.ConnectionString = mystr;
    myconn.Open();
    mysql = "SELECT Sale.日期, Customer.姓名, Sale.商品编号, Sale.数量, Sale.金额";
    mysql += " FROM Customer, Sale";
    mysql += " WHERE Customer.顾客编号=Sale.顾客编号 ";
    mysql += "AND Customer.顾客编号=' " + textBox1.Text.Trim() + "' ";
    mycmd.CommandText = mysql;
    mycmd.Connection = myconn;
    MySqlDataReader myreader = mycmd.ExecuteReader();
    ListViewItem itemx = new ListViewItem();
    listView1.View = View.Details;
    //添加5个列
    listView1.Columns.Add("日期", 90, HorizontalAlignment.Center);
    listView1.Columns.Add("姓名", 70, HorizontalAlignment.Center);
    listView1.Columns.Add("商品编号", 80, HorizontalAlignment.Center);
    listView1.Columns.Add("数量", 50, HorizontalAlignment.Center);
    listView1.Columns.Add("金额", 50, HorizontalAlignment.Center);
}
```

```

while (myreader.Read())           //循环读取信息
{
    itemx = listView1.Items.Add(myreader["日期"].ToString()); //添加1个
    ListViewItem对象
    itemx.SubItems.Add(myreader[1].ToString()); //添加4个子项
    itemx.SubItems.Add(myreader[2].ToString());
    itemx.SubItems.Add(myreader.GetInt32(3).ToString());
    itemx.SubItems.Add(myreader[4].ToString());
}
}

```

(4) 在命名空间添加 `using MySql.Data;` `using MySql.Data.MySqlClient;` 修改 `Program.cs` 中的启动项。运行结果如图3-6所示。



图 3-5 SqlDataReader 对象的使用方法

5.7 DataSet 对象的使用方法

本例是教材第 13 章 P354 【练一练】的内容，主要要求与流程请参照教材，这里只提醒与指导需要注意的事项。

- (1) 在新建的 `Form7` 中，按照教材指示，添加 `button` 和 `label`。
 - (2) `Button1_Click` 功能块同理，是双击 `button1` 进入。
 - (3) 需要引起注意的是字符串 `mystr` 的设置与打开登入 `MySQL` 时选择的参数有关。示例代码沿用了 `MySQL` 默认给出的 `Server` 的名称。在实验中，同学们可以选择自己想要取用的名称。
- 示例代码如下：

```

private void Button1_Click(object sender, EventArgs e)
{
    string mystr, mysql;
    MySqlConnection myconn = new MySqlConnection();
    mystr = "server=localhost;user=root;database=smk;port=3306;password=****";
    myconn.ConnectionString = mystr;
    myconn.Open();
    mysql = "SELECT Sale.日期, Customer.姓名, Sale.商品编号, Sale.数量, Sale.金额";
    mysql += " FROM Customer, Sale";
}

```

```

mysql += " WHERE Customer.顾客编号=Sale.顾客编号 ";
mysql += " AND Customer.顾客编号='" + textBox1.Text.Trim() + "' ";
MySqlDataAdapter myda = new MySqlDataAdapter(mysql, myconn);
myconn.Close();
DataSet mydataset = new DataSet();
myda.Fill(mydataset, "mydata");
ListViewItem itemx = new ListViewItem();
listView1.View = View.Details;
//添加5个列
listView1.Columns.Add("日期", 90, HorizontalAlignment.Center);
listView1.Columns.Add("姓名", 70, HorizontalAlignment.Center);
listView1.Columns.Add("商品编号", 80, HorizontalAlignment.Center);
listView1.Columns.Add("数量", 50, HorizontalAlignment.Center);
listView1.Columns.Add("金额", 50, HorizontalAlignment.Center);
foreach (DataRow dr in mydataset.Tables[0].Rows)
{
    itemx = listView1.Items.Add(dr[0].ToString()); //添加1个ListItem对象
    itemx.SubItems.Add(dr[1].ToString()); //添加4个子项
    itemx.SubItems.Add(dr[2].ToString());
    itemx.SubItems.Add(dr[3].ToString());
    itemx.SubItems.Add(dr[4].ToString());
}
}

```

(4) 在命名空间添加 `using MySql.Data.MySqlClient;` 修改 `Program.cs` 中的启动项。运行结果如图 3-7 所示。



图 3-7 DataSet 对象的使用方法

5.8 显示 Product 表的第一个记录

本例是教材第 14 章 P361【练一练】的内容，主要要求与流程请参照教材，这里只提醒与指导需要注意的事项。

(1) 第 13 章的练习并没有修改 `App.config` 文件，本章将做此练习。在 Visual Studio 2019 右上角“解决方案管理器”打开 `App.config` 文件。可以看到 `App.config` 为 XML 格式的文档。XML 文档可以看做具有

树的结构，使用符号<node>和</node>（node 表示节点的名字）来表示节点的开始和结束，即相同名称的开始符号和结束符号中间的内容就是该节点的内容。同时，要注意到节点的树状结构。当然，XML 不是本章的重点，有兴趣的同学可以自行百度。

另外，还需要在引用中包含 System.Configuration 选项，可以参考教材 P337。其他不详尽之处也可以参考教材 P337，或在 StackOverflow 查询。

Data Source 修改为了惯常使用的名称。App.config 的编辑代码如下：

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <startup>
    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.7.2" />
  </startup>
  <connectionStrings>
    <add name="myDatabaseConnection"
connectionString="server=localhost;user=root;database=smk;port=3306;password=****;" />
  </connectionStrings>
</configuration>
```

(2) 在新建的 Form8 中，按照教材指示，添加控件。

(3) Form8_Load 功能块同理，是双击 Form8 的标题栏进入。

(4) 示例代码如下：

```
private void Form1_Load(object sender, EventArgs e)
{
    string mystr, mysql;
    MySqlConnection myconn = new MySqlConnection();
    mystr = System.Configuration.ConfigurationManager.
        ConnectionStrings["myDatabaseConnection"].ToString();
    myconn.ConnectionString = mystr;
    myconn.Open();
    mysql = "SELECT * FROM Product";
    MySqlDataAdapter myda = new MySqlDataAdapter(mysql, myconn);
    DataSet mydataset = new DataSet();
    myda.Fill(mydataset, "Product");
    myconn.Close();
    Binding mybinding = new Binding("Text", mydataset, "Product.商品编号");
    textBox1.DataBindings.Add(mybinding);
    //上两个语句等同：textBox1.DataBindings.Add("Text", mydataset, "Product.商
品编号");
    textBox2.DataBindings.Add("Text", mydataset, "Product.商品名称");
    textBox3.DataBindings.Add("Text", mydataset, "Product.分类");
    textBox4.DataBindings.Add("Text", mydataset, "Product.子类");
    textBox5.DataBindings.Add("Text", mydataset, "Product.单价");
    textBox6.DataBindings.Add("Text", mydataset, "Product.库存数量");
}
```

(5) 添加 `using MySql.Data.MySqlClient;` `using System.Configuration;` 并修改 `Program.cs` 中的启动项。运行结果如图 3-8 所示。

图 3-8 显示 Product 表的第一个记录

5.9 Product 表记录浏览

本例是教材第 14 章 P364 【练一练】的内容，主要要求与流程请参照教材，这里只提醒与指导需要注意的事项。

(1) 在新建的 `Form9` 中，按照教材指示，添加控件。

(2) `Form9_Load`、`button` 功能块同理，是双击 `Form9` 的标题栏、`button` 进入。需要注意的是，这里使用了类字段。

(3) 示例代码如下：

```
public partial class Form9 : Form
{
    BindingSource mybs;    //类字段
    public Form9()
    {
        InitializeComponent();
    }

    private void Form9_Load(object sender, EventArgs e)
    {
        string mystr, mysql;
        MySqlConnection myconn = new MySqlConnection();
        mystr = System.Configuration.ConfigurationManager.
            ConnectionStrings["myDatabaseConnection"].ToString();
        myconn.ConnectionString = mystr;
        myconn.Open();
        mysql = "SELECT * FROM Product";
        MySqlDataAdapter myda = new MySqlDataAdapter(mysql, myconn);
        DataSet mydataset = new DataSet();
        myda.Fill(mydataset, "Product");
    }
}
```

```

myconn.Close();
mybs = new BindingSource(mydataset, "Product");
//用数据源mydataset和表Product创建新实例mybs
textBox1.DataBindings.Add("Text", mybs, "商品编号");
textBox2.DataBindings.Add("Text", mybs, "商品名称");
textBox3.DataBindings.Add("Text", mybs, "分类");
textBox4.DataBindings.Add("Text", mybs, "子类");
textBox5.DataBindings.Add("Text", mybs, "单价");
textBox6.DataBindings.Add("Text", mybs, "库存数量");
}

private void Button1_Click(object sender, EventArgs e)
{
    if (mybs.Position != 0)
        mybs.MoveFirst();           //移到第一个记录
}

private void Button2_Click(object sender, EventArgs e)
{
    if (mybs.Position != 0)
        mybs.MovePrevious();        //移到上一个记录
}

private void Button3_Click(object sender, EventArgs e)
{
    if (mybs.Position != mybs.Count - 1)
        mybs.MoveNext();           //移到下一个记录
}

private void Button4_Click(object sender, EventArgs e)
{
    if (mybs.Position != mybs.Count - 1)
        mybs.MoveLast();           //移到最后一个记录
}
}

```

(4) 添加 `using MySql.Data.MySqlClient;` `using System.Configuration;` 并修改 `Program.cs` 中的启动项。运行结果如图 3-9 所示。

图 3-9 Product 表记录浏览

5.10 求指定分类的商品销售总数量和总金额

本例是教材第 14 章 P366【练一练】的内容，主要要求与流程请参照教材，这里只提醒与指导需要注意的事项。

- (1) 在新建的 Form10 中，按照教材指示，添加控件。
- (2) Form10_Load、button 功能块同理，是双击 Form10 的标题栏、button 进入。
- (3) 示例代码如下：

```
private void Form10_Load(object sender, EventArgs e)
{
    string mystr, mysql;
    MySqlConnection myconn = new MySqlConnection();
    mystr = System.Configuration.ConfigurationManager.
        ConnectionStrings["myDatabaseConnection"].ToString();
    myconn.ConnectionString = mystr;
    myconn.Open();
    mysql = "SELECT DISTINCT 分类 FROM Product";
    MySqlDataAdapter myda = new MySqlDataAdapter(mysql, myconn);
    DataSet mydataset = new DataSet();
    myda.Fill(mydataset, "Product");
    myconn.Close();
    comboBox1.DataSource = mydataset;
    comboBox1.DisplayMember = "Product. 分类";
}

private void Button1_Click(object sender, EventArgs e)
{
    if (comboBox1.Text != "")
    {
        string mystr, mysql;
        MySqlConnection myconn = new MySqlConnection();
        mystr = System.Configuration.ConfigurationManager.
            ConnectionStrings["myDatabaseConnection"].ToString();
        myconn.ConnectionString = mystr;
        myconn.Open();
        mysql = "SELECT SUM(Sale. 数量) AS 数量, SUM(Sale. 金额) AS 金额";
        mysql += " FROM Product, Sale";
    }
}
```

```

mysql += " WHERE Product. 商品编号=Sale. 商品编号";
mysql += " AND 分类='" + comboBox1.Text + "'";
MySqlDataAdapter myda = new MySqlDataAdapter(mysql, myconn);
DataSet mydataset = new DataSet();
myda.Fill(mydataset, "Product");
myconn.Close();
label2.Text = comboBox1.Text.Trim() + "分类商品\n 销售总数量为" +
    mydataset.Tables[0].Rows[0][0].ToString();
label2.Text += "\n 销售总金额为" +
mydataset.Tables[0].Rows[0][1].ToString();
    }
}

```

(4) 添加 `using MySql.Data.MySqlClient;` `using System.Configuration;` 并修改 Program.cs 中的启动项。运行结果如图 3-10 所示。

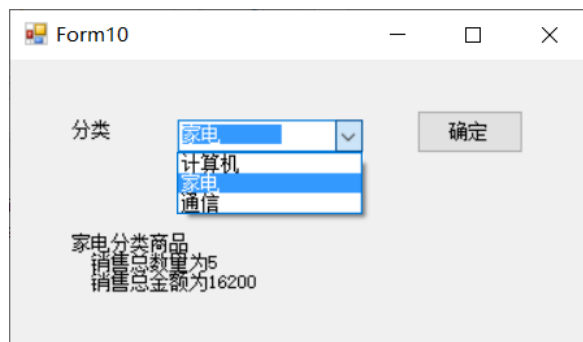


图 3-10 求指定分类的商品销售总数量和总金额

5.11 通过 BindingNavigator 控件实现对 Product 表中所有记录的浏览、添加和删除操作

本例是教材第 14 章 P369【练一练】的内容，主要要求与流程请参照教材，这里只提醒与指导需要注意的事项。

(1) 在新建的 Form11 中，按照教材指示，添加控件，如图 3-11 所示。



图 3-11 添加的控件示例

(2) ToolStripButton1 和 ToolStripButton2 的设置请参考教材 P368 BindingNavigator 控件的应用，如图 3-12, 3-13 所示。

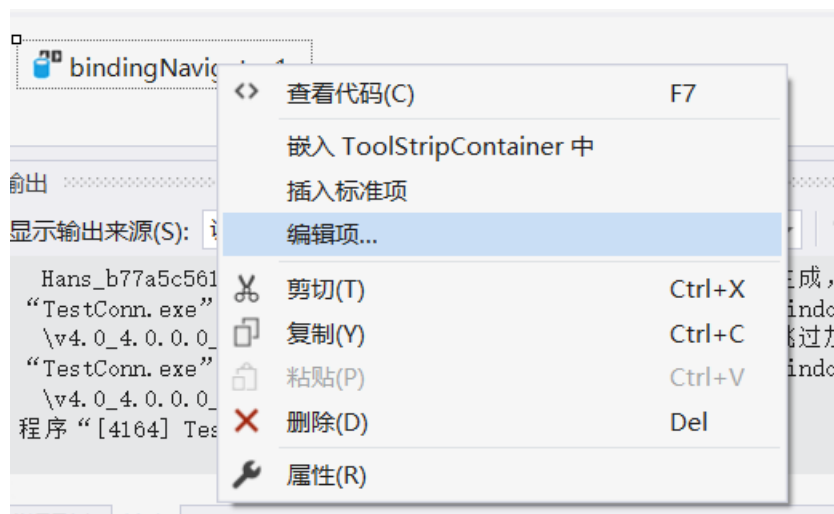


图 3-12 添加 ToolStripButton 与 ToolStripSeparator (1)

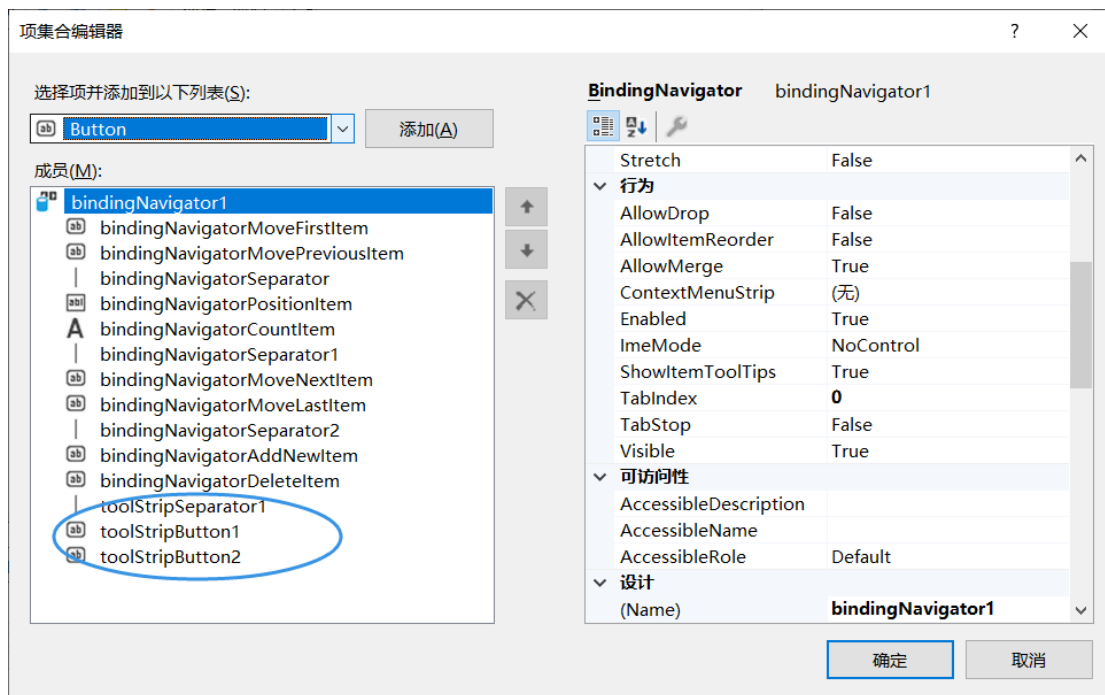


图 3-13 添加 ToolStripButton 与 ToolStripSeparator (2)

(3) Form11_Load、ToolStripButton_Click 功能块同理，是双击 Form11 的标题栏、ToolStripButton_Click 进入。

(4) BindingNavigatorDeleteItem_MouseDown 功能块的设置需要额外关注。如图 3-14(a)所示，点击导航区的删除按钮，在删除按钮的属性页点击闪电形状的按钮以查看所有的属性。之后找到“MouseDown”选项，双击，即可进入 Form.cs 对应的 BindingNavigatorDeleteItem_MouseDown 的功能块。

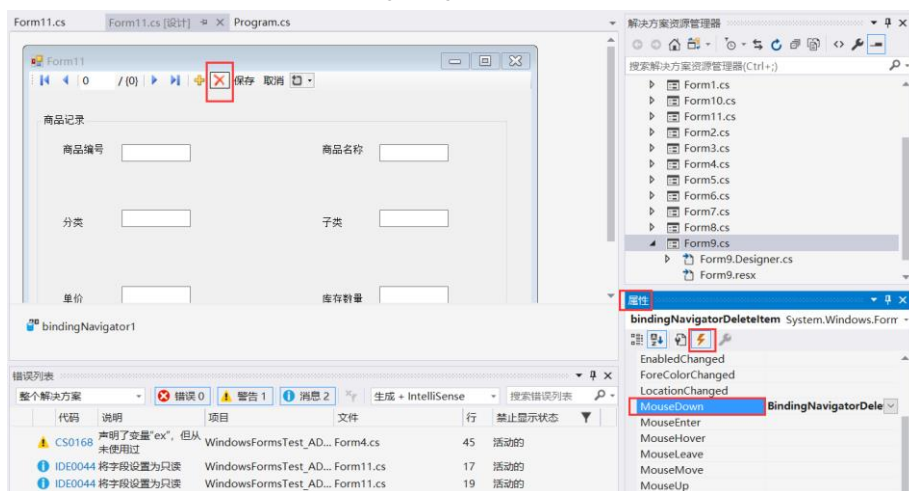


图 3-14(a) BindingNavigatorDeleteItem_MouseDown 功能块的设置

(5) 示例代码如下：

```
public partial class Form11 : Form
{
    MySqlDataAdapter myda; //5个窗体级字段
    DataSet mydataset = new DataSet();
    BindingSource mybs;
    MySqlConnection myconn = new MySqlConnection();
```

```

string spno;           //商品编号
public Form11()
{
    InitializeComponent();
}

private void Form11_Load(object sender, EventArgs e)
{
    string mystr, mysql;
    mystr = System.Configuration.ConfigurationManager.
        ConnectionStrings["myDatabaseConnection"].ToString();
    myconn.ConnectionString = mystr;
    myconn.Open();
    mysql = "SELECT * FROM Product";
    myda = new MySqlDataAdapter(mysql, myconn);
    myda.Fill(mydataset, "Product");
    myconn.Close();
    mybs = new BindingSource(mydataset, "Product");
    //用数据源mydataset和表Product创建新实例mybs
    textBox1.DataBindings.Add("Text", mybs, "商品编号");
    textBox2.DataBindings.Add("Text", mybs, "商品名称");
    textBox3.DataBindings.Add("Text", mybs, "分类");
    textBox4.DataBindings.Add("Text", mybs, "子类");
    textBox5.DataBindings.Add("Text", mybs, "单价");
    textBox6.DataBindings.Add("Text", mybs, "库存数量");
    bindingNavigator1.Dock = DockStyle.Bottom;
    bindingNavigator1.BindingSource = mybs;
}

private void ToolStripButton1_Click(object sender, EventArgs e) //保存
{
    mybs.EndEdit();
    if (this.mydataset.HasChanges() == false) //数据无变化, 返回
        return;
    //添加记录用的SQL语句
    string insertsql = String.Format("INSERT INTO Product(商品编号,"
        + "商品名称,分类,子类,单价,库存数量)
VALUES(' {0}', ' {1}', ' {2}', ' {3}', {4}, {5})",
        textBox1.Text.Trim(), textBox2.Text.Trim(),
        textBox3.Text.Trim(), textBox4.Text.Trim(),
        textBox5.Text.Trim(), textBox6.Text.Trim());
    SqlCommand insertcmd = new SqlCommand(insertsql, myconn);
    myda.InsertCommand = insertcmd;
    string deletesql = String.Format("DELETE Product WHERE 商品编号=' {0}' ",

```

```

spno);

SqlCommand deletecmd = new SqlCommand(deletesql, myconn);
myda.DeleteCommand = deletecmd;
myda.Update(mydataset, "Product");
}

private void ToolStripButton2_Click(object sender, EventArgs e) //取消
{
    mybs.CancelEdit();
}

private void BindingNavigatorDeleteItem_MouseDown(object sender, MouseEventArgs
e)
{
    //删除之前，先获取当前商品编号
    spno = textBox1.Text.Trim();
}
}

```

(4) 添加 `using MySql.Data.MySqlClient;` `using System.Configuration;` 并修改 `Program.cs` 中的启动项。运行结果如图 3-14(b)所示，在数据库文件中的操作结果如图 3-14(c)所示。

(b)

商品编号	商品名称	分类	子类	单价	库存数量
1001	联想1001	计算机	笔记本	5600	48
1002	联想1002	计算机	台式机	4300	39
1003	美的1003	家电	电冰箱	2800	49
1004	海尔1004	家电	电冰箱	3800	49
1005	戴尔1005	计算机	台式机	4200	49
1006	宏基1006	计算机	台式机	4700	49
1007	长虹1007	家电	电视机	3500	47
1008	TCL1008	家电	电视机	2600	49
1009	小米1009	通信	手机	2500	50
1010	华为1010	通信	手机	2200	50
5001	test	test	test	42	1
NULL	NULL	NULL	NULL	NULL	NULL

(c)

图 3-14(b)(c)通过 BindingNavigator 控件实现对 Product 表中所有记录的浏览、添加和删除操作

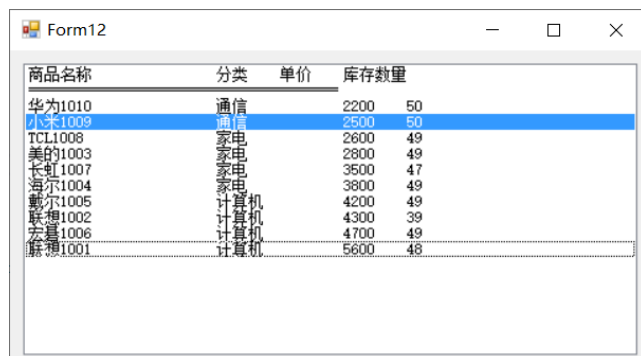
5.12 使用 DataView 对象在列表框中案单价升序、库存数量降序排序显示所有商品记录

本例是教材第 14 章 P373【练一练】的内容，主要要求与流程请参照教材，这里只提醒与指导需要注意的事项。

- (1) 在新建的 Form12 中，按照教材指示，添加控件。
- (2) Form12_Load 功能块同理，是双击 Form12 的标题栏进入。
- (3) 示例代码如下：

```
private void Form12_Load(object sender, EventArgs e)
{
    string mystr, mysql;
    MySqlConnection myconn = new MySqlConnection();
    mystr = System.Configuration.ConfigurationManager.
        ConnectionStrings["myDatabaseConnection"].ToString();
    myconn.ConnectionString = mystr;
    myconn.Open();
    mysql = "SELECT * FROM Product";
    MySqlDataAdapter myda = new MySqlDataAdapter(mysql, myconn);
    DataSet mydataset = new DataSet();
    myda.Fill(mydataset, "Product");
    myconn.Close();
    DataView mydv = new DataView(mydataset.Tables["Product"]);
    mydv.Sort = "单价 ASC, 库存数量 DESC";
    listBox1.Items.Add("商品名称\t\t分类\t\t单价\t\t库存数量");
    listBox1.Items.Add("=====");
    for (int i = 0; i < mydv.Count; i++)
    {
        listBox1.Items.Add(String.Format("{0}\t\t{1}\t\t{2}\t\t{3}",
            mydv[i]["商品名称"], mydv[i]["分类"], mydv[i]["单价"], mydv[i]["库存数量"]));
    }
}
```

- (4) 添加 `using MySql.Data.MySqlClient;` `using System.Configuration;` 并修改 Program.cs 中的启动项。运行结果如图 3-15 所示。



商品名称	分类	单价	库存数量
华为1010	通信	2200	50
海信1003	通信	2500	50
TCL1008	家电	2600	49
美的1003	家电	2800	49
长虹1007	家电	3500	47
海尔1004	家电	3800	49
戴尔1005	计算机	4200	49
联想1002	计算机	4300	39
宏碁1006	计算机	4700	49
联想1001	计算机	5600	48

图 3-15 使用 DataView 对象在列表框中案单价升序、库存数量降序排序显示所有商品记录

5.13 用 DataGridView 控件显示所有商品记录

本例是教材第 14 章 P382【练一练】的内容，主要要求与流程请参照教材，这里只提醒与指导需要注意的事项。

(1) 在新建的 Form12 中，按照教材指示，添加 DataGridView 和 label 控件。注意课本 P377 页创建了 SQL Server 的连接，而此处我们需要创建 MySQL 的连接，如图 3-16：

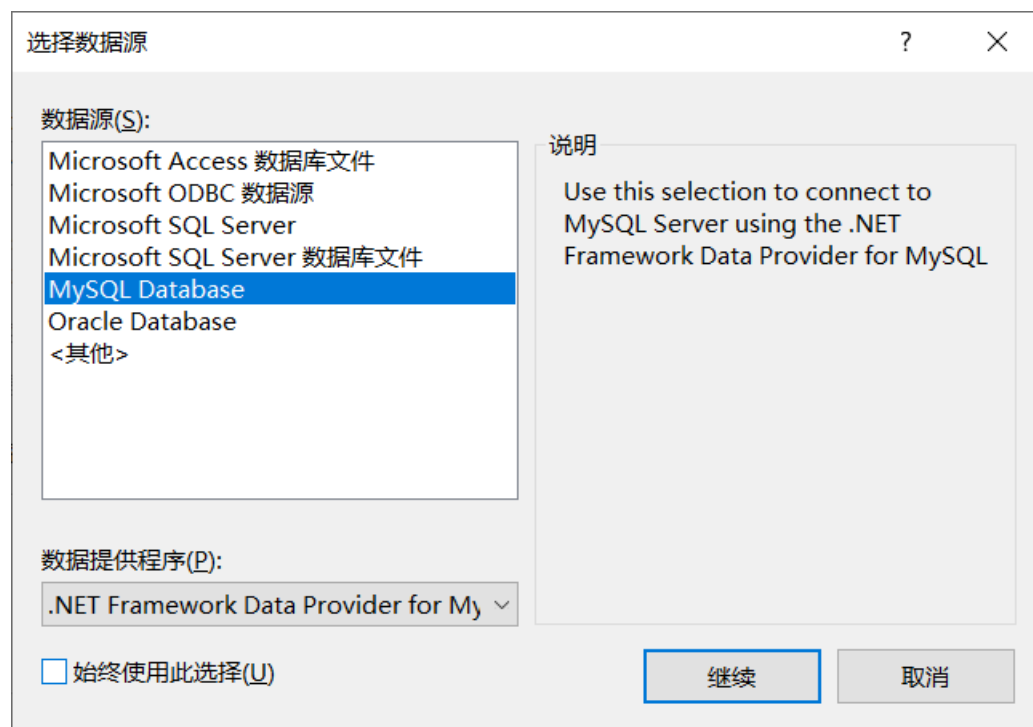


图 3-16 创建 DataGridView 与 MySQL 数据库的连接

(2) DataGridView 控件的连接相关操作需要参考教材 P375“创建 DataGridView 控件”。不同的操作存在与第(6)和第(9)——即服务器名的输入和“编辑列”对话框的打开。

(3) 服务器名，用户名和密码需要注意。如图 3-17(a)沿用的就是 localhost，root 和自设密码。在完成相关输入后，可以点击测试连接，确保 DataGridView 可以正确连接的我们的目标数据库。

(4) “编辑列”对话框的打开。因为编译器版本的差异，Visual Studio 2019 中，DataGridView 的“编辑列”对话框出现在其对应的属性页中，如图 3-17(b)所示。

(5) Form13_Load 功能块同理，是双击 Form13 的标题栏进入。dataGridView1_CellContentClick 功能块同理，是双击 DataGridView 之后进入的。

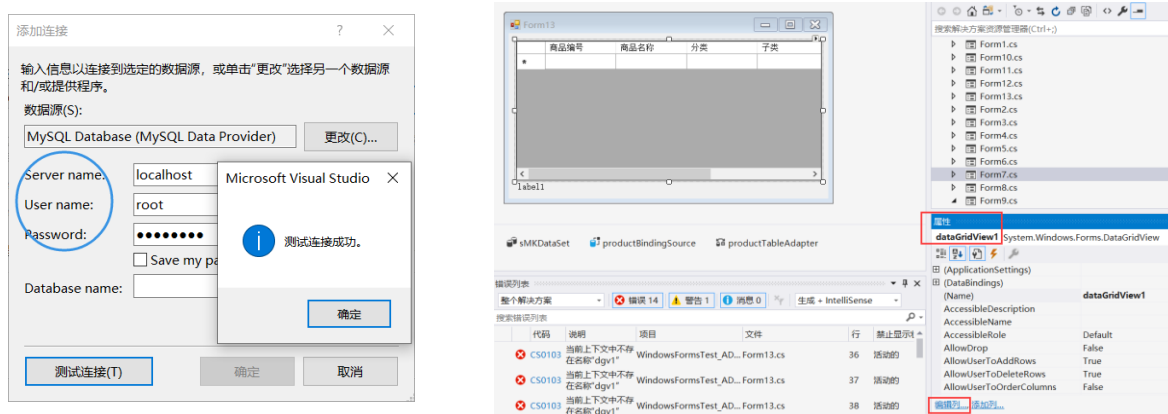


图 3-17(a)添加连接

(b) “编辑列”对话框的打开

(6) 示例代码如下：

```
private void Form13_Load(object sender, EventArgs e)
{
    // TODO: 这行代码将数据加载到表“smkDataSet.Product”中。您可以根据需要移动
    或删除它。

    this.productTableAdapter.Fill(this.smkDataSet.Product);
    string mystr, mysql;
    MySqlConnection myconn = new MySqlConnection();
    mystr = System.Configuration.ConfigurationManager.
        ConnectionStrings["myDatabaseConnection"].ToString();
    myconn.ConnectionString = mystr;
    myconn.Open();
    mysql = "SELECT * FROM Product";
    MySqlDataAdapter myda = new MySqlDataAdapter(mysql, myconn);
    DataSet mydataset = new DataSet();
    myda.Fill(mydataset, "Product");
    myconn.Close();
    dataGridView1.DataSource = mydataset.Tables["Product"];
    dataGridView1.GridColor = Color.RoyalBlue;
    dataGridView1.ScrollBars = ScrollBars.Vertical;
    dataGridView1.CellBorderStyle = DataGridViewCellBorderStyle.Single;
    dataGridView1.ReadOnly = true;           //设置为只读的
    dataGridView1.SelectionMode = DataGridViewSelectionMode.FullRowSelect;
    dataGridView1.AlternatingRowsDefaultCellStyle.ForeColor = Color.Blue;
    dataGridView1.AlternatingRowsDefaultCellStyle.BackColor = Color.Tomato;
    dataGridView1.Columns[0].Width = 90;
    dataGridView1.Columns[1].Width = 100;
    dataGridView1.Columns[2].Width = 80;
    dataGridView1.Columns[3].Width = 80;
    dataGridView1.Columns[4].Width = 70;
```

```

        dataGridView1.Columns[5].Width = 90;
        label1.Text = "";
    }

    private void dataGridView1_CellContentClick(object sender,
DataGridViewCellEventArgs e)
    {
        try
        {
            if (e.RowIndex < dataGridView1.RowCount - 1)
                label1.Text = "选择的商品编号为:" +
                    dataGridView1.Rows[e.RowIndex].Cells[0].Value;
        }
        catch (Exception ex)
        {
            MessageBox.Show("需选中一个商品记录", "信息提示");
        }
    }
}

```

(7) 添加 `using MySql.Data.MySqlClient;` `using System.Configuration;` 并修改 `Program.cs` 中的启动项。运行结果如图 3-17(c)所示。



商品编号	商品名称	分类	子类	单价	库存数量
1001	联想1001	计算机	笔记本	5600	48
1002	联想1002	计算机	台式机	4300	39
1003	美的1003	家电	电冰箱	2800	49
1004	海尔1004	家电	电冰箱	3600	49
1005	戴尔1005	计算机	台式机	4200	49
1006	宏碁1006	计算机	台式机	4700	49
1007	长虹1007	家电	电视机	3500	47
1008	TCL1008	家电	电视机	2600	49
1009	小米1009	通信	手机	2500	50

图 3-17(c)用 DataGridView 控件显示所有商品记录

5.14 Product 表的复杂查询

本例是教材第 14 章 P384【练一练】的内容，主要要求与流程请参照教材，这里只提醒与指导需要注意的事项。

- (1) 新建了 `Class.cs` 文件，在其中添加 `TempData` 类；
- (2) 在新建的 `Form14` 和 `Form14-1` 中，按照教材指示，添加控件。`Form14` 和 `Form14-1` 的 `ControlBox` 属性都设置为了 `False`。
- (3) 所有功能块同理，是双击控件进入。
- (4) `Form14` 示例代码如下：

```

private void Form14_Load(object sender, EventArgs e)
{
    comboBox1.Items.Add("");
}

```



```

        comboBox1.Items.Add("计算机");
        comboBox1.Items.Add("通信");
        comboBox1.Items.Add("家电");
        comboBox2.Items.Add("");
        comboBox2.Items.Add("电视机");
        comboBox2.Items.Add("电冰箱");
        comboBox2.Items.Add("手机");
    }

    private void Button1_Click(object sender, EventArgs e)
    {
        string condstr = "";
        if (textBox1.Text != "")
            condstr = "商品编号 Like '%" + textBox1.Text.Trim() + "%'";
        if (comboBox1.Text != "")
        {
            if (condstr == "")
                condstr = "分类='" + comboBox1.Text.Trim() + "'";
            else
                condstr += " AND 分类='" + comboBox1.Text.Trim() + "'";
        }
        if (comboBox2.Text != "")
        {
            if (condstr == "")
                condstr = "子类='" + comboBox2.Text.Trim() + "'";
            else
                condstr += " AND 子类='" + comboBox2.Text.Trim() + "'";
        }
        if (textBox2.Text != "")
        {
            if (textBox2.Text != "")
                condstr = "商品名称 Like '%" + textBox2.Text.Trim() + "%'";
            else
                condstr += " AND 商品名称 Like '%" + textBox2.Text.Trim() + "%'";
        }
        if (textBox3.Text != "" && textBox4.Text != "")
        {
            double dj1, dj2;
            try
            {
                dj1 = double.Parse(textBox3.Text.Trim());
                dj2 = double.Parse(textBox4.Text.Trim());
            }
            catch (Exception ex)

```

```

        {
            MessageBox.Show("单价区间输入有错误：" + ex.Message);
            return;
        }
        if (dj1 > dj2)
        {
            MessageBox.Show("单价区间输入有错误", "操作提示");
            return;
        }
        if (condstr == "")
            condstr = "单价>=" + dj1.ToString() + " AND 单价<=" +
dj2.ToString();
        else
            condstr += " AND (单价>=" + dj1.ToString() + " AND 单价<=" +
dj2.ToString() + ")";
    }
    else if (textBox3.Text != "" || textBox4.Text != "")
    {
        MessageBox.Show("单价区间上下界输入有错误", "操作提示");
        return;
    }
    TempData.condstr = condstr;
    Form myform = new Form14_1();
    myform.ShowDialog();
}

private void Button2_Click(object sender, EventArgs e)
{
    textBox1.Text = ""; textBox2.Text = "";
    textBox3.Text = ""; textBox4.Text = "";
    comboBox1.Text = ""; comboBox2.Text = "";
    textBox1.Focus();
}

private void Button3_Click(object sender, EventArgs e)
{
    this.Close();
}

```

Form14-1 示例代码如下:

```

namespace WindowsFormsTest_ADO.NET
{
    public partial class Form14_1 : Form
    {
        DataView mydv = new DataView();
    }
}

```

```
public Form14_1()
{
    InitializeComponent();
}

private void Form14_1_Load(object sender, EventArgs e)
{
    bind();
    dataGridView1.EnableHeadersVisualStyles = false;
    dataGridView1.MultiSelect = false;
    dataGridView1.Columns[0].ReadOnly = true;
    dataGridView1.Columns[1].ReadOnly = true;
    dataGridView1.Columns[2].ReadOnly = true;
    dataGridView1.Columns[3].ReadOnly = true;
    dataGridView1.Columns[4].ReadOnly = true;
    dataGridView1.Columns[5].ReadOnly = true;
    dataGridView1.Columns[0].Width = 100;
    dataGridView1.Columns[1].Width = 120;
    dataGridView1.Columns[2].Width = 90;
    dataGridView1.Columns[3].Width = 90;
    dataGridView1.Columns[4].Width = 90;
    dataGridView1.Columns[5].Width = 100;
}

private void bind()    //绑定数据
{
    string mystr, mysql;
    MySqlConnection myconn = new MySqlConnection();
    mystr = System.Configuration.ConfigurationManager.
        ConnectionStrings["myDatabaseConnection"].ToString();
    myconn.ConnectionString = mystr;
    myconn.Open();
    mysql = "SELECT * FROM Product";
    MySqlDataAdapter myda = new MySqlDataAdapter(mysql, myconn);
    DataSet mydataset = new DataSet();
    myda.Fill(mydataset, "Product");
    myconn.Close();
    mydv = mydataset.Tables["Product"].DefaultView; //获得DataView对象mydv
    mydv.RowFilter = TempData.condstr;                //过滤DataView中的记录
    dataGridView1.DataSource = mydv;
}

private void Button1_Click(object sender, EventArgs e)
{
    this.Close();
}
```

```
}  
}  
}
```

(5) 添加 `using MySql.Data.MySqlClient;` `using System.Configuration;` 并修改 `Program.cs` 中的启动项。运行结果如图 3-18 所示。

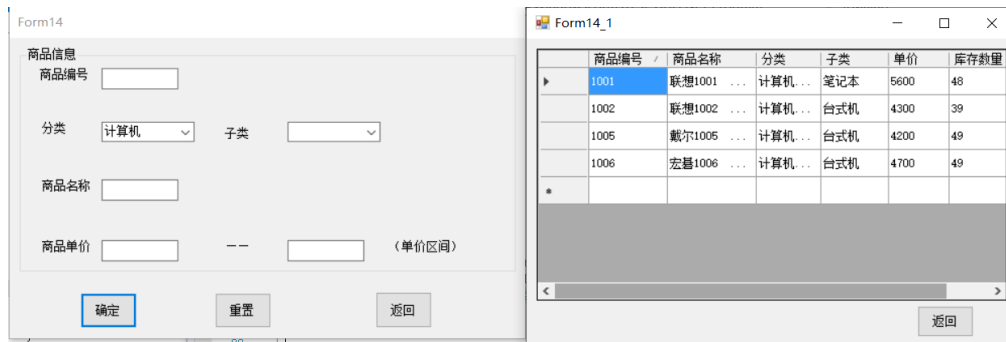


图 3-18 Product 表的商品复杂查询

5.15 Product 表的商品修改

本例是教材第 14 章 P388【练一练】的内容，主要要求与流程请参照教材，这里只提醒与指导需要注意的事项。

(1) 在新建的 `Form14` 中，按照教材指示，添加控件。参考 5.13 与 5.14 设置 `DataGridView` 控件的数据源：

(2) 所有控件功能块同理，是双击控件进入。

(3) 示例代码如下：

```
namespace WindowsFormsTest_ADO.NET  
{  
    public partial class Form15 : Form  
    {  
        MySqlDataAdapter myda;  
        DataSet mydataset = new DataSet();  
        public Form15()  
        {  
            InitializeComponent();  
        }  
  
        private void Button1_Click(object sender, EventArgs e)  
        {  
            MySqlCommandBuilder mycmdbuilder = new MySqlCommandBuilder(myda);  
            //获取对应的修改命令  
            if (mydataset.HasChanges()) //如果有数据改动  
            {  
                try  
                {  

```

```

        myda.Update(mydataset, "Product"); //更新数据源
    }
    catch (Exception ex)
    {
        MessageBox.Show("数据修改不正确，如商品编号重复等", "信息提示");
    }
}

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

private void Form15_Load(object sender, EventArgs e)
{
    string mystr, mysql;
    MySqlConnection myconn = new MySqlConnection();
    mystr = System.Configuration.ConfigurationManager.
        ConnectionStrings["myDatabaseConnection"].ToString();
    myconn.ConnectionString = mystr;
    myconn.Open();
    mysql = "SELECT * FROM Product";
    myda = new MySqlDataAdapter(mysql, myconn);
    myda.Fill(mydataset, "Product");
    myconn.Close();
    dgv1.DataSource = mydataset.Tables["Product"];
    dgv1.GridColor = Color.RoyalBlue;
    dgv1.ScrollBars = ScrollBars.Vertical;
    dgv1.CellBorderStyle = DataGridViewCellBorderStyle.Single;
    dgv1.SelectionMode = DataGridViewSelectionMode.FullRowSelect;
    dgv1.AlternatingRowsDefaultCellStyle.ForeColor = Color.Blue;
    dgv1.AlternatingRowsDefaultCellStyle.BackColor = Color.Tomato;
    dgv1.Columns[0].Width = 100;
    dgv1.Columns[1].Width = 100;
    dgv1.Columns[2].Width = 80;
    dgv1.Columns[3].Width = 80;
    dgv1.Columns[4].Width = 70;
    dgv1.Columns[5].Width = 100;
    dgv1.AllowUserToAddRows = true; //允许添加记录
    dgv1.AllowUserToDeleteRows = false; //不允许删除记录
}
}
}

```

(4) 添加 `using MySql.Data.MySqlClient;` `using System.Configuration;` 并修改 Program.cs 中的启动项。数据库实现的修改如图 3-19(b)所示。

Form15

	商品编号	商品名称	分类	子类	单价	库存数量
▶	1001	联想1001 ...	计算机	笔记本	5600	48
	1002	联想1002 ...	计算机	台式机	4300	39
	1003	美的1003 ...	家电	电冰箱	2800	49
	1004	海尔1004 ...	家电	电冰箱	3800	49
	1005	戴尔1005 ...	计算机	台式机	4200	49
	1006	宏碁1006 ...	计算机	台式机	4700	49
	1007	长虹1007 ...	家电	电视机	3500	47
	1008	TCL1008 ...	家电	电视机	2600	49
	1009	小米1009 ...	通信	手机	2500	50
	1010	华为1010 ...	通信	手机	2200	50
	3001	海尔3001 ...	家电	洗衣机	1800	60
*						

确定修改 返回

MSI\SQLEXPRESS....K - dbo.Product

	商品编号	商品名称	分类	子类	单价	库存数量
	1001	联想1001	计算机	笔记本	5600	48
	1002	联想1002	计算机	台式机	4300	39
	1003	美的1003	家电	电冰箱	2800	49
	1004	海尔1004	家电	电冰箱	3800	49
	1005	戴尔1005	计算机	台式机	4200	49
	1006	宏碁1006	计算机	台式机	4700	49
	1007	长虹1007	家电	电视机	3500	47
	1008	TCL1008	家电	电视机	2600	49
	1009	小米1009	通信	手机	2500	50
	1010	华为1010	通信	手机	2200	50
▶	3001	海尔3001	家电	洗衣机	1800	60
*	NULL	NULL	NULL	NULL	NULL	NULL

图 3- 19(a)Product 表的商品修改

(b)数据库文件中实现的同步

(5) 修改数据后，在 DataGrip 中关闭数据库再重新打开，观察数据库中数据的变化。

实验四 数据库系统开发实例

一、实验目的

- (1) 本章实验的内容主要涵盖教材 15 章的内容，通过上机实验以加深对语言和数据库开发的理解；
- (2) 掌握一个简单的超市系统的开发。

二、实验内容

本实验主要包括如下部分的内容。

- 1 公共类设计
- 2 登陆窗体设计
- 3 主菜单窗体设计
- 4 “添加新商品”功能设计
- 5 “编辑商品信息”功能设计
- 6 “增加老商品库存”功能设计
- 7 “商品库存预警”功能设计
- 8 “添加新顾客”功能设计
- 9 “查看顾客购物信息”功能设计
- 10 “顾客购物”功能设计
- 11 “顾客退货”功能设计
- 12 “按分类统计销售情况”功能设计
- 13 “按商品统计销售情况”功能设计
- 14 “用户管理”功能设计
- 15 “设置商品类别”功能设计
- 16 “系统初始化”功能设计
- 17 帮助功能设计

以上实验的操作流程请主要参考教材与之前的章节出现的内容，下文主要给出示例所使用的代码。

请大家务必携带教材《C#语言与数据库技术基础教程》！

三、实验仪器、设备及材料

硬件：计算机；《C#语言与数据库技术基础教程》

软件：Windows 操作系统；Visual Studio 2019（推荐使用最新版本）；MySQL 8.0.23；DataGrip;.NET Framework v4.7.2

四、实验原理

SM 系统是一个采用会员制的超市管理系统。先将顾客信息录入到系统中，每个顾客对应一个顾客编号，同样，所有商品信息也要录入到系统中，每个商品对应一个商品编号（实际应用中可以采用商品条形码，只需要增加条码机设备即可），然后可以使用 SM 系统完成购物和统计等功能。本系统的主要功能如下。

- (1) 实现商品信息的添加、编辑、增加库存和库存报警功能。
- (2) 实现顾客信息的添加、编辑和购物情况查询功能。
- (3) 实现顾客的购物和退货操作功能。
- (4) 实现各种商品销售统计功能。
- (5) 实现各种系统数据的设置功能。
- (6) 实现系统用户的管理和控制功能。

五、实验步骤

5.1 公共类设计

(1) App.config 文件的引用如下：

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <connectionStrings>
    <add name="myconnstring" connectionString="
server=localhost;user=root;database=smk;port=3306;password=****;"
    providerName=" MySql.Data.MySqlClient " />
    <add name="WindowsFormsTest_ADO.NET.Properties.Settings.SMKConnectionString"
    connectionString="
server=localhost;user=root;database=smk;port=3306;password=****;"
    providerName=" MySql.Data.MySqlClient " />
  </connectionStrings>
  <startup>
    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.7.2" />
  </startup>
</configuration>
```

(2) 新建类 CommDb.cs。示例代码如下：

```
namespace SM
{
    public class CommDB
    {
```



```

public CommDB()                //默认构造函数
{
}

//*****
//返回SELECT语句执行后记录集中的行数
//*****

public int Rownum(string sql)
{ //sql参数指出SQL语句
    int i = 0;
    string mystr = System.Configuration.ConfigurationManager.
        ConnectionStrings["myconnstring"].ToString();
    //从App.config文件获取连接字符串
    MySqlConnection myconn = new MySqlConnection();
    myconn.ConnectionString = mystr;
    myconn.Open();
    MySqlCommand mycmd = new MySqlCommand(sql, myconn);
    MySqlDataReader myreader = mycmd.ExecuteReader();
    while (myreader.Read())      //循环读取信息
    { i++; }
    myconn.Close();
    return i;                    //返回读取的行数
}

//*****
//返回SELECT语句执行后唯一行的唯一字段值
//*****

public string Returnafield(string sql)
{ //sql指出SQL语句
    string fn;
    string mystr = System.Configuration.ConfigurationManager.
        ConnectionStrings["myconnstring"].ToString();
    //从App.config文件获取连接字符串
    MySqlConnection myconn = new MySqlConnection();
    myconn.ConnectionString = mystr;
    myconn.Open();
    MySqlCommand mycmd = new MySqlCommand(sql, myconn);
    MySqlDataReader myreader = mycmd.ExecuteReader();
    myreader.Read();
    fn = myreader[0].ToString().Trim();
    myconn.Close();
    return fn;                  //返回读取的数据
}

//*****
//执行SQL语句，返回是否成功执行。SQL语句最好是如下：
//UPDATE 表名 SET 字段名=value, 字段名=value WHERE 字段名=value
//DELETE FROM 表名 WHERE 字段名=value

```

```
//INSERT INTO 表名 (字段名, 字段名) values (value, value)
//*****
public void ExecuteNonQuery(string sql)
{
    string mystr = System.Configuration.ConfigurationManager.
        ConnectionStrings["myconnstring"].ToString();
    MySqlConnection myconn = new MySqlConnection();
    myconn.ConnectionString = mystr;
    myconn.Open();
    MySqlCommand mycmd = new MySqlCommand(sql, myconn);
    mycmd.ExecuteNonQuery();
    myconn.Close();
}
//*****
//执行SELECT语句, 返回DataSet对象
//*****
public DataSet ExecuteQuery(string sql, string tname)
{
    string mystr = System.Configuration.ConfigurationManager.
        ConnectionStrings["myconnstring"].ToString();
    MySqlConnection myconn = new MySqlConnection();
    myconn.ConnectionString = mystr;
    myconn.Open();
    MySqlDataAdapter myda = new MySqlDataAdapter(sql, myconn);
    DataSet myds = new DataSet();
    myda.Fill(myds, tname);
    myconn.Close();
    return myds;
}
//*****
//执行SELECT语句, 返回聚合函数结果
//*****
public string ExecuteAggregateQuery(string sql)
{
    string jg;
    string mystr = System.Configuration.ConfigurationManager.
        ConnectionStrings["myconnstring"].ToString();
    MySqlConnection myconn = new MySqlConnection();
    myconn.ConnectionString = mystr;
    myconn.Open();
    MySqlCommand mycmd = new MySqlCommand();
    mycmd.CommandText = sql;
    mycmd.Connection = myconn;
    jg = mycmd.ExecuteScalar().ToString();
}
```

```

        myconn.Close();
        return jg;
    }
}

public class TempData
{
    public static int tag;        //操作标志: 1: 添加, 2: 修改
    public static string sql;    //窗体之间传递的SQL语句
    public static string userlevel; //当前用户级别
}
}

```

5.2 登陆窗体设计

(1) 按照教材 P399 的指示, 设计窗体和所有控件。

(2) 示例代码如下:

```

private void button1_Click(object sender, EventArgs e)
{
    CommDB mydb = new CommDB();
    string mystr;
    mystr = "SELECT * FROM SUser WHERE "
        + "用户名=' " + textBox1.Text.Trim()
        + "' AND 密码=' " + textBox2.Text.Trim() + "' ";
    int i = mydb.Rownum(mystr);
    if (i == 0)
    {
        MessageBox.Show("本次登录失败!", "操作提示");
        return;
    }
    else
    {
        mystr = "SELECT 级别 FROM SUser WHERE "
            + "用户名=' " + textBox1.Text.Trim()
            + "' AND 密码=' " + textBox2.Text.Trim() + "' ";
        TempData.userlevel = mydb.Returnafield(mystr);
        Form myform = new Main();
        myform.ShowDialog();
        this.Close();
    }
}

private void button2_Click(object sender, EventArgs e)
{
    this.Close();
}

```

```
}
```

5.3 主菜单窗体设计

(1) 按照教材 P401 的指示，设计窗体和所有控件。

(2) 示例代码如下：

```
private void Main_Load(object sender, EventArgs e)
{
    this.Text += "-" + TempData.userlevel;
    if (TempData.userlevel == "管理员") //管理员限制
    {
        toolStrip1.Items["toolStripButton1"].Enabled = false;
        toolStrip1.Items["toolStripButton2"].Enabled = false;
        menuStrip1.Items["gwgl"].Enabled = false;
    }
    else //操作员功能限制
    {
        menuStrip1.Items["spxxgl"].Enabled = false;
        menuStrip1.Items["gkxxgl"].Enabled = false;
        menuStrip1.Items["cxgl"].Enabled = false;
        menuStrip1.Items["xtgl"].Enabled = false;
    }
}

private void 添加新商品menu_Click(object sender, EventArgs e)
{
    Form myform = new addsp();
    myform.MdiParent = this; //建立父子窗体关系
    myform.Show();
}

private void 编辑商品信息menu_Click(object sender, EventArgs e)
{
    Form myform = new editsp();
    myform.MdiParent = this;
    myform.Show();
}

private void 增加老商品库存menu_Click(object sender, EventArgs e)
{
    Form myform = new addspkc();
    myform.MdiParent = this;
    myform.Show();
}
```

```
}

private void 商品库存报警menu_Click(object sender, EventArgs e)
{
    Form myform = new spkcbj();
    myform.MdiParent = this;
    myform.Show();
}

private void 添加新顾客menu_Click(object sender, EventArgs e)
{
    Form myform = new addgk();
    myform.MdiParent = this;
    myform.Show();
}

private void 编辑顾客信息menu_Click(object sender, EventArgs e)
{
    Form myform = new editgk();
    myform.MdiParent = this;
    myform.Show();
}

private void 查看顾客购物信息menu_Click(object sender, EventArgs e)
{
    Form myform = new querygkgw();
    myform.MdiParent = this;
    myform.Show();
}

private void 顾客购物menu_Click(object sender, EventArgs e)
{
    Form myform = new gkgw();
    myform.MdiParent = this;
    myform.Show();
}

private void 顾客退货menu_Click(object sender, EventArgs e)
{
    Form myform = new gkth();
    myform.MdiParent = this;
    myform.Show();
}
```

```
private void 按分类统计销售情况menu_Click(object sender, EventArgs e)
{
    Form myform = new tjfl();
    myform.MdiParent = this;
    myform.Show();
}
```

```
private void 按子类统计销售情况menu_Click(object sender, EventArgs e)
{
    Form myform = new tjzl();
    myform.MdiParent = this;
    myform.Show();
}
```

```
private void 按商品统计销售情况menu_Click(object sender, EventArgs e)
{
    Form myform = new tjsp();
    myform.MdiParent = this;
    myform.Show();
}
```

```
private void 用户管理menu_Click(object sender, EventArgs e)
{
    Form myform = new setuser();
    myform.MdiParent = this;
    myform.Show();
}
```

```
private void 设置商品类别menu_Click(object sender, EventArgs e)
{
    Form myform = new setsplb();
    myform.MdiParent = this;
    myform.Show();
}
```

```
private void 设置地区信息menu_Click(object sender, EventArgs e)
{
    Form myform = new setarea();
    myform.MdiParent = this;
    myform.Show();
}
```

```
private void 系统初始化menu_Click(object sender, EventArgs e)
{

```

```
        Form myform = new init();
        myform.MdiParent = this;
        myform.Show();
    }

    private void 关于menu_Click(object sender, EventArgs e)
    {
        Form myform = new about();
        myform.MdiParent = this;
        myform.Show();
    }

    private void 联系信息menu_Click(object sender, EventArgs e)
    {
        Form myform = new corporation();
        myform.MdiParent = this;
        myform.Show();
    }

    private void toolStripButton1_Click(object sender, EventArgs e)
    {
        Form myform = new gkgw();
        myform.MdiParent = this;
        myform.Show();
    }

    private void toolStripButton2_Click(object sender, EventArgs e)
    {
        Form myform = new gkth();
        myform.MdiParent = this;
        myform.Show();
    }

    private void toolStripButton3_Click(object sender, EventArgs e)
    {
        this.Close();
    }
}
```

5.4 “添加新商品” 功能设计

- (1) 按照教材 P404 的指示，设计窗体和所有控件。
- (2) 示例代码如下：

```
public partial class addsp : Form
```

```

{
    CommDB mydb = new CommDB();
    public addsp()
    {
        InitializeComponent();
    }
    private void addsp_Load(object sender, EventArgs e)
    {
        DataSet mydataset;
        string mystr = "SELECT distinct 分类 FROM Prodsort";
        mydataset = mydb.ExecuteQuery(mystr, "Prodsort");
        comboBox1.DataSource = mydataset.Tables["Prodsort"];
        comboBox1.DisplayMember = "分类";
    }

    private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
    {
        DataSet mydataset;
        string mystr = "SELECT distinct 子类 FROM Prodsort WHERE 分类=' "
            + comboBox1.Text.Trim() + "'";
        mydataset = mydb.ExecuteQuery(mystr, "Prodsort");
        comboBox2.DataSource = mydataset.Tables["Prodsort"];
        comboBox2.DisplayMember = "子类";
    }

    private void button1_Click(object sender, EventArgs e)
    {
        if (textBox1.Text == "" || textBox2.Text == ""
            || textBox3.Text == "" || textBox4.Text == ""
            || comboBox1.Text == "" || comboBox2.Text == "")
        {
            MessageBox.Show("所有数据项必须输入", "操作提示");
            return;
        }
        string mystr = "SELECT * FROM Product WHERE 商品编号=' "
            + textBox1.Text.Trim() + "'";
        int i = mydb.Rownum(mystr);
        if (i != 0)
        {
            MessageBox.Show("输入的商品编号已存在", "操作提示");
            return;
        }
        try
        {

```



```

double dj = double.Parse(textBox3.Text.Trim());
int sl = int.Parse(textBox4.Text.Trim());
mystr = "INSERT INTO Product(商品编号, 分类, 子类, " +
        "商品名称, 单价, 库存数量) VALUES(' "
        + textBox1.Text.Trim() + ", ' "
        + comboBox1.Text.Trim() + ", ' "
        + comboBox2.Text.Trim() + ", ' "
        + textBox2.Text.Trim() + ", "
        + dj.ToString() + ", "
        + sl.ToString() + ")";
mydb.ExecuteNonQuery(mystr);
}
catch (Exception ex)
{
    MessageBox.Show("输入的商品数据有错误: " + ex.Message);
    return;
}
button2_Click(sender, e);
}

private void button2_Click(object sender, EventArgs e)
{
    textBox1.Text = ""; textBox2.Text = "";
    textBox3.Text = ""; textBox4.Text = "";
    comboBox1.Text = ""; comboBox2.Text = "";
    textBox1.Focus();
}

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

```

5.5 “编辑商品信息” 功能设计

(1) 按照教材 P407 的指示, 设计窗体和所有控件。其中需要设计三个窗体, 分别为 editsp、editsp1、editsp2。

(2) editsp 示例代码如下:

```

public partial class editsp : Form
{
    CommDB mydb = new CommDB();
    public editsp()

```

```

{
    InitializeComponent();
}

private void editsp_Load(object sender, EventArgs e)
{
    DataSet mydataset;
    string mystr = "SELECT distinct 分类 FROM Prodsort";
    mydataset = mydb.ExecuteQuery(mystr, "Prodsort");
    DataRow nrow = mydataset.Tables["Prodsort"].NewRow(); //插入一个空行
    nrow["分类"] = "";
    mydataset.Tables["Prodsort"].Rows.InsertAt(nrow, 0);
    comboBox1.DataSource = mydataset.Tables["Prodsort"];
    comboBox1.DisplayMember = "分类";
}

private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
{
    comboBox2.Text = "";
    DataSet mydataset;
    string mystr = "SELECT distinct 子类 FROM Prodsort WHERE 分类=' "
        + comboBox1.Text.Trim() + " " ";
    mydataset = mydb.ExecuteQuery(mystr, "Prodsort");
    DataRow nrow = mydataset.Tables["Prodsort"].NewRow(); //插入一个空行
    nrow["子类"] = "";
    mydataset.Tables["Prodsort"].Rows.InsertAt(nrow, 0);
    comboBox2.DataSource = mydataset.Tables["Prodsort"];
    comboBox2.DisplayMember = "子类";
}

private void button1_Click(object sender, EventArgs e)
{
    string condstr = "";
    if (textBox1.Text != "")
        condstr = "商品编号 Like '%" + textBox1.Text.Trim() + "%' ";
    if (comboBox1.Text != "")
    {
        if (condstr == "")
            condstr = "分类=' " + comboBox1.Text.Trim() + " " ";
        else
            condstr += " AND 分类=' " + comboBox1.Text.Trim() + " " ";
    }
    if (comboBox2.Text != "")
    {
        if (condstr == "")
            condstr = "子类=' " + comboBox2.Text.Trim() + " " ";
    }
}

```

```

        else
            condstr += " AND 子类='" + comboBox2.Text.Trim() + "'";
    }
    if (textBox2.Text != "")
    {
        if (textBox2.Text != "")
            condstr = "商品名称 Like '%" + textBox2.Text.Trim() + "%'";
        else
            condstr += " AND 商品名称 Like '%" + textBox2.Text.Trim() + "%'";
    }
    if (textBox3.Text != "" && textBox4.Text != "")
    {
        double dj1, dj2;
        try
        {
            dj1 = double.Parse(textBox3.Text.Trim());
            dj2 = double.Parse(textBox4.Text.Trim());
        }
        catch (Exception ex)
        {
            MessageBox.Show("单价区间输入有错误: " + ex.Message);
            return;
        }
        if (dj1 > dj2)
        {
            MessageBox.Show("单价区间输入有错误", "操作提示");
            return;
        }
        if (condstr == "")
            condstr = "单价>=" + dj1.ToString() + " AND 单价<=" +
dj2.ToString();
        else
            condstr += " AND (单价>=" + dj1.ToString() + " AND 单价<=" +
dj2.ToString() + ")";
    }
    else if (textBox3.Text != "" || textBox4.Text != "")
    {
        MessageBox.Show("单价区间上下界输入有错误", "操作提示");
        return;
    }
    if (condstr != "")
        TempData.sql = "SELECT * FROM Product WHERE " + condstr;
    else
        TempData.sql = "";

```

```

        editspl myform = new editspl();
        myform.ShowDialog();
    }

    private void button2_Click(object sender, EventArgs e)
    {
        textBox1.Text = ""; textBox2.Text = "";
        textBox3.Text = ""; textBox4.Text = "";
        comboBox1.Text = ""; comboBox2.Text = "";
        textBox1.Focus();
    }

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

```

editspl 示例代码如下:

```

public partial class editspl : Form
{
    CommDB mydb = new CommDB();
    public editspl()
    {
        InitializeComponent();
    }

    private void editspl_Load(object sender, EventArgs e)
    {
        bind();
        dataGridView1.EnableHeadersVisualStyles = false;
        dataGridView1.MultiSelect = false;
        dataGridView1.ReadOnly = true;
        dataGridView1.Columns[0].Width = 100;
        dataGridView1.Columns[1].Width = 120;
        dataGridView1.Columns[2].Width = 90;
        dataGridView1.Columns[3].Width = 90;
        dataGridView1.Columns[4].Width = 90;
        dataGridView1.Columns[5].Width = 100;
        if (dataGridView1.Rows.Count > 0)
            dataGridView1.Rows[0].Selected = true; //默认选择第1行
    }

    private void editspl_Activated(object sender, EventArgs e)

```

```

{
    bind();
    if (dataGridView1.Rows.Count > 0)
        dataGridView1.Rows[0].Selected = true; //默认选择第1行
}

private void bind() //绑定数据
{
    string mystr;
    if (TempData.sql == "")
        mystr = "SELECT * FROM Product";
    else
        mystr = TempData.sql;
    DataSet mydataset = mydb.ExecuteQuery(mystr, "Product");
    dataGridView1.DataSource = mydataset.Tables["Product"];
}

private void button1_Click(object sender, EventArgs e)
{
    //修改记录
    if (dataGridView1.SelectedRows.Count > 0)
    {
        string spno =
dataGridView1.SelectedRows[0].Cells[0].Value.ToString().Trim();
        string spmc =
dataGridView1.SelectedRows[0].Cells[1].Value.ToString().Trim();
        string spfl =
dataGridView1.SelectedRows[0].Cells[2].Value.ToString().Trim();
        string spzl =
dataGridView1.SelectedRows[0].Cells[3].Value.ToString().Trim();
        string spdj =
dataGridView1.SelectedRows[0].Cells[4].Value.ToString().Trim();
        string spsl =
dataGridView1.SelectedRows[0].Cells[5].Value.ToString().Trim();
        Form myform = new editsp2(spno, spmc, spfl, spzl, spdj, spsl); //带传递数
据调用editsp2的构造函数
        myform.ShowDialog();
    }
}

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

private void button2_Click(object sender, EventArgs e)

```

```

        { //删除记录
            if (dataGridView1.SelectedRows.Count > 0)
            {
                string spno =
dataGridView1.SelectedRows[0].Cells[0].Value.ToString().Trim();
                DialogResult result = MessageBox.Show("真的要删除" + spno
                    + "编号的商品吗?", "操作提示", MessageBoxButtons.YesNo);
                if (result == DialogResult.Yes)
                {
                    string mystr = "DELETE Product WHERE 商品编号='" + spno + "'";
                    mydb.ExecuteNonQuery(mystr);
                    bind();
                }
            }
        }

private void dataGridView1_CellClick(object sender, DataGridViewCellEventArgs
e)
{
    //单击任何单元时选择该单元所在的行
    if (e.RowIndex>=0 && e.RowIndex<dataGridView1.Rows.Count)
        dataGridView1.Rows[e.RowIndex].Selected = true;
}

private void 修改ToolStripMenuItem_Click(object sender, EventArgs e)
{
    //菜单命令: 修改
    button1_Click(sender, e);
}

private void 删除ToolStripMenuItem_Click(object sender, EventArgs e)
{
    //菜单命令: 删除
    button2_Click(sender, e);
}
}

```

editsp2 示例代码如下:

```

namespace SM
{
    public partial class editsp2 : Form
    {
        CommDB mydb = new CommDB();
        string spno, spname;
        string spfl, spzl;
        string spdj, spsl;
        public editsp2(string spno1, string spname1, string spfl1, string spzl1, string

```

```

spdjl, string spsll)
{
    InitializeComponent();
    spno = spnol; spfl = spfl1; spz1 = spzll;
    spname = spname1; spdj = spdjl; spsl = spsll;
}

private void editsp2_Load(object sender, EventArgs e)
{
    DataSet mydataset;
    string mystr = "SELECT distinct 分类 FROM Prodsort";
    mydataset = mydb.ExecuteQuery(mystr, "Prodsort");
    comboBox1.DataSource = mydataset.Tables["Prodsort"];
    comboBox1.DisplayMember = "分类";
    textBox1.Text = spno;
    textBox1.Enabled = false;
    comboBox1.Text = spfl;
    comboBox2.Text = spz1;
    textBox2.Text = spname;
    textBox3.Text = spdj;
    textBox4.Text = spsl;
}

private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
{
    DataSet mydataset;
    string mystr = "SELECT distinct 子类 FROM Prodsort WHERE 分类=' "
        + comboBox1.Text.Trim() + " ";
    mydataset = mydb.ExecuteQuery(mystr, "Prodsort");
    comboBox2.DataSource = mydataset.Tables["Prodsort"];
    comboBox2.DisplayMember = "子类";
}

private void button1_Click(object sender, EventArgs e)
{
    string mystr;
    if (textBox1.Text == "" || textBox2.Text == ""
        || textBox3.Text == "" || textBox4.Text == ""
        || comboBox1.Text == "" || comboBox2.Text == "")
    {
        MessageBox.Show("所有数据项必须输入");
        return;
    }
    try

```

```

    {
        double dj = double.Parse(textBox3.Text.Trim());
        int sl = int.Parse(textBox4.Text.Trim());
        mystr = "UPDATE Product SET 分类=' "
            + comboBox1.Text.Trim() + "',子类=' "
            + comboBox2.Text.Trim() + "',商品名称=' "
            + textBox2.Text.Trim() + "',单价=' "
            + dj.ToString() + "',库存数量=' "
            + sl.ToString() + " WHERE 商品编号=' "
            + spno + "' ";
        mydb.ExecuteNonQuery(mystr);
    }
    catch (Exception ex)
    {
        MessageBox.Show("修改的商品数据有错误: " + ex.Message);
        return;
    }
    this.Close();
}

private void button2_Click(object sender, EventArgs e)
{
    this.Close();
}
}
}

```

5.6 “增加老商品库存” 功能设计

(1) 按照教材 P414 的指示，设计窗体和所有控件。这里共有 addspkc、addspkc1 两个窗体。

(2) addspkc 示例代码如下：

```

namespace SM
{
    public partial class addspkc : Form
    {
        CommDB mydb=new CommDB();
        public addspkc()
        {
            InitializeComponent();
        }
        private void addspkc_Load(object sender, EventArgs e)
        {
            DataSet mydataset;

```



```

string mystr = "SELECT distinct 分类 FROM Prodsort";
mydataset = mydb.ExecuteQuery(mystr, "Prodsort");
DataRow nrow = mydataset.Tables["Prodsort"].NewRow(); //插入一个空行
nrow["分类"] = "";
mydataset.Tables["Prodsort"].Rows.InsertAt(nrow, 0);
comboBox1.DataSource = mydataset.Tables["Prodsort"];
comboBox1.DisplayMember = "分类";
}

private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
{
    comboBox2.Text = "";
    DataSet mydataset;
    string mystr = "SELECT distinct 子类 FROM Prodsort WHERE 分类=' "
        + comboBox1.Text.Trim() + " '";
    mydataset = mydb.ExecuteQuery(mystr, "Prodsort");
    DataRow nrow = mydataset.Tables["Prodsort"].NewRow(); //插入一个空行
    nrow["子类"] = "";
    mydataset.Tables["Prodsort"].Rows.InsertAt(nrow, 0);
    comboBox2.DataSource = mydataset.Tables["Prodsort"];
    comboBox2.DisplayMember = "子类";
}

private void button1_Click(object sender, EventArgs e)
{
    string condstr = "";
    if (textBox1.Text != "")
        condstr = "商品编号 Like '%" + textBox1.Text.Trim() + "%'";
    if (comboBox1.Text != "")
    {
        if (condstr == "")
            condstr = "分类=' " + comboBox1.Text.Trim() + " '";
        else
            condstr += " AND 分类=' " + comboBox1.Text.Trim() + " '";
    }
    if (comboBox2.Text != "")
    {
        if (condstr == "")
            condstr = "子类=' " + comboBox2.Text.Trim() + " '";
        else
            condstr += " AND 子类=' " + comboBox2.Text.Trim() + " '";
    }
    if (textBox2.Text != "")
    {
        if (textBox2.Text != "")

```

```

        condstr = "商品名称 Like '%" + textBox2.Text.Trim() + "%' ";
    else
        condstr += " AND 商品名称 Like '%" + textBox2.Text.Trim() + "%' ";
    }
    if (textBox3.Text != "" && textBox4.Text != "")
    {
        double dj1, dj2;
        try
        {
            dj1 = double.Parse(textBox3.Text.Trim());
            dj2 = double.Parse(textBox4.Text.Trim());
        }
        catch (Exception ex)
        {
            MessageBox.Show("单价区间输入有错误: " + ex.Message);
            return;
        }
        if (dj1 > dj2)
        {
            MessageBox.Show("单价区间输入有错误", "操作提示");
            return;
        }
        if (condstr == "")
            condstr = "单价>=" + dj1.ToString() + " AND 单价<=" +
dj2.ToString();
        else
            condstr += " AND (单价>=" + dj1.ToString() + " AND 单价<=" +
dj2.ToString() + ")";
    }
    else if (textBox3.Text != "" || textBox4.Text != "")
    {
        MessageBox.Show("单价区间上下界输入有错误", "操作提示");
        return;
    }
    if (condstr != "")
        TempData.sql = "SELECT * FROM Product WHERE " + condstr;
    else
        TempData.sql = "";
    addspkc1 myform = new addspkc1();
    myform.ShowDialog();
}

private void button2_Click(object sender, EventArgs e)
{

```

```

        textBox1.Text = ""; textBox2.Text = "";
        textBox3.Text = ""; textBox4.Text = "";
        comboBox1.Text = ""; comboBox2.Text = "";
        textBox1.Focus();
    }

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

addspck1 示例代码如下:
namespace SM
{
    public partial class addspck1 : Form
    {
        CommDB mydb = new CommDB();
        public addspck1()
        {
            InitializeComponent();
        }

        private void addspck1_Load(object sender, EventArgs e)
        {
            string mystr;
            if (TempData.sql == "")
                mystr = "SELECT * FROM Product";
            else
                mystr = TempData.sql;
            DataSet mydataset = mydb.ExecuteQuery(mystr, "Product");
            dataGridView1.DataSource=mydataset.Tables["Product"];
            dataGridView1.EnableHeadersVisualStyles = false;
            dataGridView1.Columns.Add("tjsl", "增加数量"); //增加一个列
            dataGridView1.Columns[0].ReadOnly = true;
            dataGridView1.Columns[1].ReadOnly = true;
            dataGridView1.Columns[2].ReadOnly = true;
            dataGridView1.Columns[3].ReadOnly = true;
            dataGridView1.Columns[4].ReadOnly = true;
            dataGridView1.Columns[5].ReadOnly = true;
            dataGridView1.Columns[0].Width = 100;
            dataGridView1.Columns[1].Width = 110;

```

```

        dataGridView1.Columns[2].Width = 80;
        dataGridView1.Columns[3].Width = 80;
        dataGridView1.Columns[4].Width = 80;
        dataGridView1.Columns[5].Width = 100;
        dataGridView1.Columns[6].Width = 100;
    }

    private void button1_Click(object sender, EventArgs e)
    {
        for (int i = 0; i < dataGridView1.Rows.Count; i++)
            if (dataGridView1.Rows[i].Cells[6].Value != null)
                Updateks(dataGridView1.Rows[i].Cells[0].Value.ToString().Trim(),
                    dataGridView1.Rows[i].Cells[6].Value.ToString().Trim());
        this.Close();
    }

    protected void Updateks(string spno, string addks)
    {
        string mystr = "UPDATE Product SET 库存数量=库存数量+"
            + addks + " WHERE 商品编号='" + spno + "'";
        mydb.ExecuteNonQuery(mystr);
    }

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

```

5.7 “商品库存预警” 功能设计

(1) 按照教材 P416 的指示，设计窗体和所有控件。

(2) 示例代码如下：

```

namespace SM
{
    public partial class spkcbj : Form
    {
        public spkcbj()
        {
            InitializeComponent();
        }

        private void spkcbj_Load(object sender, EventArgs e)
    }
}

```

```

{
    CommDB mydb = new CommDB();
    string mystr;
    mystr = "SELECT * FROM Product WHERE 库存数量<=40";
    DataSet mydataset = mydb.ExecuteQuery(mystr, "Product");
    dataGridView1.DataSource = mydataset.Tables["Product"];
    dataGridView1.EnableHeadersVisualStyles = false;
    dataGridView1.MultiSelect = false;
    dataGridView1.ReadOnly = true;
    dataGridView1.Columns[0].Width = 100;
    dataGridView1.Columns[1].Width = 120;
    dataGridView1.Columns[2].Width = 90;
    dataGridView1.Columns[3].Width = 90;
    dataGridView1.Columns[4].Width = 90;
    dataGridView1.Columns[5].Width = 100;
}

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

```

5.8 “添加新顾客”功能设计

(1) 按照教材 P418 的指示，设计窗体和所有控件。

(2) 示例代码如下：

```

namespace SM
{
    public partial class addgk : Form
    {
        CommDB mydb = new CommDB();
        public addgk()
        {
            InitializeComponent();
        }

        private void addgk_Load(object sender, EventArgs e)
        {
            DataSet mydataset;
            string mystr = "SELECT distinct 省份 FROM Area";
            mydataset = mydb.ExecuteQuery(mystr, "Area");
        }
    }
}

```

```

        comboBox1.DataSource = mydataset.Tables["Area"];
        comboBox1.DisplayMember = "省份";
    }

    private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
    {
        DataSet mydataset;
        string mystr = "SELECT distinct 城市 FROM Area WHERE 省份=' "
            + comboBox1.Text.Trim() + "' ";
        mydataset = mydb.ExecuteQuery(mystr, "Area");
        comboBox2.DataSource = mydataset.Tables["Area"];
        comboBox2.DisplayMember = "城市";
    }

    private void comboBox2_SelectedIndexChanged(object sender, EventArgs e)
    {
        DataSet mydataset;
        string mystr = "SELECT distinct 县 FROM Area WHERE 城市=' "
            + comboBox2.Text.Trim() + "' ";
        mydataset = mydb.ExecuteQuery(mystr, "Area");
        comboBox3.DataSource = mydataset.Tables["Area"];
        comboBox3.DisplayMember = "县";
    }

    private void button1_Click(object sender, EventArgs e)
    {
        if (textBox1.Text == "" || textBox2.Text == ""
            || comboBox1.Text == "" || comboBox2.Text == "" || comboBox3.Text == "")
        {
            MessageBox.Show("所有数据项必须输入", "操作提示");
            return;
        }

        string mystr = "SELECT * FROM Customer WHERE 顾客编号=' "
            + textBox1.Text.Trim() + "' ";
        int i = mydb.Rownum(mystr);
        if (i != 0)
        {
            MessageBox.Show("输入的顾客编号已存在", "操作提示");
            return;
        }

        string xb;
        if (radioButton1.Checked)
            xb = "男";
        else if (radioButton2.Checked)

```

```

        xb="女";
    else
        xb="";
    mystr = "INSERT INTO Customer(顾客编号, 姓名, 性别, "
        + "省份, 城市, 县) VALUES(' "
        + textBox1.Text.Trim() + "', ' "
        + textBox2.Text.Trim() + "', ' "
        + xb + "', ' "
        + comboBox1.Text.Trim() + "', ' "
        + comboBox2.Text.Trim() + "', ' "
        + comboBox3.Text.Trim() + "')";
    mydb.ExecuteNonQuery(mystr);
    button2_Click(sender, e);
}

private void button2_Click(object sender, EventArgs e)
{
    textBox1.Text = ""; textBox2.Text = "";
    radioButton1.Checked = true;
    radioButton2.Checked = false;
    comboBox1.Text = ""; comboBox2.Text = "";
    comboBox3.Text = "";
    textBox1.Focus();
}

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

```

5.9 “查看顾客购物信息” 功能设计

- (1) 按照教材 P420 的指示，设计窗体和所有控件。
- (2) 示例代码如下：

```

namespace SM
{
    public partial class querykgw : Form
    {
        public querykgw()
        {
            InitializeComponent();
        }
    }
}

```

```

    }
    private void querygkgw_Load(object sender, EventArgs e)
    {
        sumlabel.Text = "";
    }
    private void button1_Click(object sender, EventArgs e)
    {
        if (textBox1.Text != "")
        {
            CommDB mydb = new CommDB();
            string mystr;
            mystr = "SELECT * FROM Sale WHERE 顾客编号='"
                + textBox1.Text.Trim()
                + "' AND DATEDIFF(DAY,日期,'" + DateTime.Now + "') <=7";
            DataSet mydataset = mydb.ExecuteQuery(mystr, "Sale");
            dataGridView1.DataSource = mydataset.Tables["Sale"];
            dataGridView1.EnableHeadersVisualStyles = false;
            dataGridView1.MultiSelect = false;
            dataGridView1.ReadOnly = true;
            dataGridView1.Columns[0].Width = 100;
            dataGridView1.Columns[1].Width = 100;
            dataGridView1.Columns[2].Width = 100;
            dataGridView1.Columns[3].Width = 100;
            dataGridView1.Columns[4].Width = 70;
            dataGridView1.Columns[5].Width = 90;
            string sumsl, sumjr;
            if (dataGridView1.Rows.Count > 0)
            {
                mystr = "SELECT SUM(数量) FROM Sale WHERE 顾客编号='"
                    + textBox1.Text.Trim() + "' ";
                sumsl = mydb.ExecuteAggregateQuery(mystr);
                mystr = "SELECT SUM(金额) FROM Sale WHERE 顾客编号='"
                    + textBox1.Text.Trim() + "' ";
                sumjr = mydb.ExecuteAggregateQuery(mystr);
            }
            else
            {
                sumsl = "0"; sumjr = "0";
            }
            sumlabel.Text = "总件数:" + sumsl.Trim() + ", 总金额:" + sumjr.Trim();
        }
        else
        {
            MessageBox.Show("请输入正确的顾客编号", "操作提示");
        }
    }
}

```



```

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

```

5.10 “顾客购物” 功能设计

(1) 按照教材 P422 的指示，设计窗体和所有控件。

(2) 示例代码如下：

```

namespace SM
{
    public partial class gkgw : Form
    {
        CommDB mydb = new CommDB();
        int sumsl = 0;
        double sumjr = 0;
        public gkgw()
        {
            InitializeComponent();
        }

        private void gkgw_Load(object sender, EventArgs e)
        {
            dataGridView1.EnableHeadersVisualStyles = false;
            dataGridView1.Columns.Add("spno", "商品编号"); //增加一个列
            dataGridView1.Columns.Add("spdj", "单价"); //增加一个列
            dataGridView1.Columns.Add("spsl", "数量"); //增加一个列
            dataGridView1.Columns.Add("spxj", "小计"); //增加一个列
            //此时dataGridView1中自动添加一个空行
            dataGridView1.Columns[0].Width = 120;
            dataGridView1.Columns[1].Width = 80;
            dataGridView1.Columns[2].Width = 80;
            dataGridView1.Columns[3].Width = 100;
            sumlabel.Text = "";
        }

        private void button1_Click(object sender, EventArgs e)
        {
            string dj;
            double xj;
            if (textBox1.Text != "" && textBox2.Text != "" && textBox3.Text != "")

```

```

        {
            string mystr = "SELECT * FROM Customer WHERE 顾客编号='" +
textBox1.Text.Trim() + "'";
            int i = mydb.Rownum(mystr);
            if (i == 0)
            {
                MessageBox.Show("输入的客户不存在，重新输入顾客编号");
                return;
            }
            try
            {
                mystr = "SELECT 单价 FROM Product WHERE 商品编号='" +
                    + textBox2.Text.Trim() + "'";
                dj = mydb.Returnafield(mystr);
                xj = int.Parse(textBox3.Text.Trim()) * Convert.ToDouble(dj);
                dataGridView1.Rows.Add(textBox2.Text.Trim(), dj.ToString(),
textBox3.Text.Trim(), xj.ToString());
            }
            catch (Exception ex)
            {
                MessageBox.Show("输入的商品不存在或数量有错误，重新输入：" +
ex.Message);
                return;
            }
            sumsl += int.Parse(textBox3.Text.Trim());
            sumjr += xj;
            sumlabel.Text = "件数:" + sumsl.ToString().Trim() + ", 总金额:" +
sumjr.ToString().Trim() + "元";
            textBox2.Text = "";
            textBox3.Text = "1";
            textBox2.Focus();
        }
        else
        {
            MessageBox.Show("输入的购物信息有错误，重新输入", "操作提示");
            return;
        }
    }

private void 删除ToolStripMenuItem_Click(object sender, EventArgs e)
{
    if (dataGridView1.Rows.Count>1 && dataGridView1.SelectedRows.Count > 0)
    {
        try
    
```

```

        {
            int s1 =
int.Parse(dataGridView1.SelectedRows[0].Cells[2].Value.ToString());
            double xj =
double.Parse(dataGridView1.SelectedRows[0].Cells[3].Value.ToString());
            sumsl -= s1;
            sumjr -= xj;
            sumlabel.Text = "件数:" + sumsl.ToString().Trim() + ", 总金额:" +
sumjr.ToString().Trim() + "元";
            dataGridView1.Rows.Remove(dataGridView1.SelectedRows[0]);
        }
        catch (Exception ex)
        {
            MessageBox.Show("没有选择合适的删除行: " + ex.Message);
        }
    }
}

```

e) private void dataGridView1_CellClick(object sender, DataGridViewCellEventArgs

```

{ //单击任何单元时选择该单元所在的行
    if (e.RowIndex >= 0 && e.RowIndex < dataGridView1.Rows.Count)
        dataGridView1.Rows[e.RowIndex].Selected = true;
}

```

```

private void button2_Click(object sender, EventArgs e)
{

```

```

    int i;
    string mystr;
    string gkno = textBox1.Text.Trim();
    string spno, spsl, spxj;
    for (i = 0; i < dataGridView1.Rows.Count; i++)
    {
        if (dataGridView1.Rows[i].Cells[0].Value != null)
        {
            // (1) 将dataGridView1控件中每行插入到Sale表中
            spno = dataGridView1.Rows[i].Cells[0].Value.ToString();
            spsl = dataGridView1.Rows[i].Cells[2].Value.ToString();
            spxj = dataGridView1.Rows[i].Cells[3].Value.ToString();
            mystr = "INSERT INTO Sale(日期, 顾客编号, 商品编号, 数量, 金额)

```

VALUES(' "

```

            + DateTime.Now + "', ' "
            + gkno + "', ' "
            + spno + "', ' "

```

```

        + spsl + ","
        + spxj + ")";
mydb.ExecuteNonQuery(mystr);
// (2) 修改Product表中对应商品的库存数量
mystr = "UPDATE Product SET 库存数量=库存数量 - "
        + spsl + " WHERE 商品编号='" + spno + "'";
mydb.ExecuteNonQuery(mystr);
    }
}
this.Close();
}
private void closebutton_Click(object sender, EventArgs e)
{
    this.Close();
}
}
}

```

5.11 “顾客退货” 功能设计

- (1) 按照教材 P425 的指示，设计窗体和所有控件。
- (2) 示例代码如下：

```

namespace SM
{
    public partial class gkth : Form
    {
        CommDB mydb = new CommDB();
        public gkth()
        {
            InitializeComponent();
        }

        private void gkth_Load(object sender, EventArgs e)
        {
        }

        private void button1_Click(object sender, EventArgs e)
        {
            if (textBox1.Text != "")
                bind();
            else
                MessageBox.Show("请输入正确的顾客编号", "操作提示");
        }

        private void bind()
        {
        }
    }
}

```

```

{
    string mystr;
    mystr = "SELECT * FROM Sale WHERE 顾客编号=' "
        + textBox1.Text.Trim()
        + " " AND DATEDIFF(DAY, 日期, ' " + DateTime.Now + " ") <=7";
    DataSet mydataset = mydb.ExecuteQuery(mystr, "Sale");
    dataGridView1.DataSource = mydataset.Tables["Sale"];
    dataGridView1.EnableHeadersVisualStyles = false;
    dataGridView1.MultiSelect = false;
    dataGridView1.ReadOnly = true;
    dataGridView1.Columns[0].Width = 100;
    dataGridView1.Columns[1].Width = 100;
    dataGridView1.Columns[2].Width = 100;
    dataGridView1.Columns[3].Width = 100;
    dataGridView1.Columns[4].Width = 70;
    dataGridView1.Columns[5].Width = 90;
}

private void dataGridView1_CellClick(object sender, DataGridViewCellEventArgs
e)
{
    //单击任何单元时选择该单元所在的行
    if (e.RowIndex >= 0 && e.RowIndex < dataGridView1.Rows.Count)
        dataGridView1.Rows[e.RowIndex].Selected = true;
}

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

private void 退货ToolStripMenuItem_Click(object sender, EventArgs e)
{
    if (dataGridView1.SelectedRows.Count > 0)
    {
        string mystr;
        string xsno =
dataGridView1.SelectedRows[0].Cells[0].Value.ToString().Trim();
        string spno =
dataGridView1.SelectedRows[0].Cells[3].Value.ToString().Trim();
        string spsl =
dataGridView1.SelectedRows[0].Cells[4].Value.ToString().Trim();
        DialogResult result = MessageBox.Show("真的退货" + spno
            + "编号的商品" + spsl + "件吗?", "操作提示",

```

```

MessageBoxButtons.YesNo);
        if (result == DialogResult.Yes)
        {
            // (1) 将退货的购物记录从Sale表中删除
            mystr = "DELETE Sale WHERE 销售编号=" + xsno;
            mydb.ExecuteNonQuery(mystr);
            // (2) 修改Product表中对应商品的库存数量
            mystr = "UPDATE Product SET 库存数量=库存数量 + "
                + spsl + " WHERE 商品编号='" + spno + "'";
            mydb.ExecuteNonQuery(mystr);
            bind();
        }
    }
    else
        MessageBox.Show("没有选择合适的要退货的销售记录", "操作提示");
}
}
}

```

5.12 “按分类统计销售情况” 功能设计

(1) 按照教材 P429 的指示，设计窗体和所有控件。

(2) 示例代码如下：

```

namespace SM
{
    public partial class tjfl : Form
    {
        public tjfl()
        {
            InitializeComponent();
        }

        private void tjfl_Load(object sender, EventArgs e)
        {
            textBox1.Text = DateTime.Now.ToString("d");
            textBox2.Text = DateTime.Now.ToString("d");
            sumlabel.Text = "";
        }

        private void button1_Click(object sender, EventArgs e)
        {
            if (textBox1.Text != "" && textBox2.Text != "")
            {

```

```

string rq1 = textBox1.Text.ToString();
string rq2 = textBox2.Text.ToString();
CommDB mydb = new CommDB();
string mystr;
mystr = "SELECT Product. 分类, SUM(Sale. 数量) AS 数量, "
      + "SUM(Sale. 金额) AS 金额 FROM Sale, Product "
      + "WHERE Product. 商品编号=Sale. 商品编号 AND "
      + " Sale. 日期 >=' " + rq1 + "' AND Sale. 日期<=' " + rq2 + "' "
      + "GROUP BY Product. 分类";
DataSet mydataset = mydb.ExecuteQuery(mystr, "Sale");
dataGridView1.DataSource = mydataset.Tables["Sale"];
dataGridView1.EnableHeadersVisualStyles = false;
dataGridView1.MultiSelect = false;
dataGridView1.ReadOnly = true;
dataGridView1.Columns[0].Width = 150;
dataGridView1.Columns[1].Width = 150;
dataGridView1.Columns[2].Width = 150;
string sumsl, sumjr;
if (dataGridView1.Rows.Count > 0)
{
    mystr = "SELECT SUM(数量) FROM Sale WHERE Sale. 日期 >=' "
          + rq1 + "' AND Sale. 日期<=' " + rq2 + "' ";
    sumsl = mydb.ExecuteAggregateQuery(mystr);
    mystr = "SELECT SUM(金额) FROM Sale WHERE Sale. 日期 >=' "
          + rq1 + "' AND Sale. 日期<=' " + rq2 + "' ";
    sumjr = mydb.ExecuteAggregateQuery(mystr);
}
else
{
    sumsl = "0"; sumjr = "0";
}
sumlabel.Text = "总件数:" + sumsl.Trim() + ", 总金额:" + sumjr.Trim()+
"元";
}
else
    MessageBox.Show("请输入正确的日期", "操作提示");
}

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

```

5.13 “按商品统计销售情况” 功能设计

(1) 按照教材 P431 的指示，设计窗体和所有控件。

(2) 示例代码如下：

```
namespace SM
{
    public partial class tjsp : Form
    {
        public tjsp()
        {
            InitializeComponent();

        }

        private void tjsp_Load(object sender, EventArgs e)
        {
            sptextBox.Text = "";
            textBox1.Text = DateTime.Now.ToString("d");
            textBox2.Text = DateTime.Now.ToString("d");
            sumlabel.Text = "";
        }

        private void button1_Click(object sender, EventArgs e)
        {
            if (sptextBox.Text != "" && textBox1.Text != "" && textBox2.Text != "")
            {
                string rq1 = textBox1.Text.ToString();
                string rq2 = textBox2.Text.ToString();
                CommDB mydb = new CommDB();
                string mystr;
                mystr = "SELECT 日期,SUM(数量) AS 数量,SUM(金额) AS 金额"
                    + " FROM Sale "
                    + "WHERE 商品编号=' " + sptextBox.Text.Trim() + "' AND "
                    + " Sale.日期 >=' " + rq1 + "' AND Sale.日期<=' " + rq2 + "' "
                    + "GROUP BY 日期 ORDER BY 日期";
                DataSet mydataset = mydb.ExecuteQuery(mystr, "Sale");
                dataGridView1.DataSource = mydataset.Tables["Sale"];
                dataGridView1.EnableHeadersVisualStyles = false;
                dataGridView1.MultiSelect = false;
                dataGridView1.ReadOnly = true;
                dataGridView1.Columns[0].Width = 150;
                dataGridView1.Columns[1].Width = 150;
                dataGridView1.Columns[2].Width = 150;
                string sumsl, sumjr;
            }
        }
    }
}
```



```

        if (dataGridView1.Rows.Count > 0)
        {
            mystr = "SELECT SUM(数量) FROM Sale "
                + " WHERE 商品编号='" + sptextBox.Text.Trim()
                + "' AND 日期 >='" + rq1 + "' AND 日期<='" + rq2 + "' ";
            sumsl = mydb.ExecuteAggregateQuery(mystr);
            mystr = "SELECT SUM(金额) FROM Sale "
                + " WHERE 商品编号='" + sptextBox.Text.Trim()
                + "' AND 日期 >='" + rq1 + "' AND 日期<='" + rq2 + "' ";
            sumjr = mydb.ExecuteAggregateQuery(mystr);
        }
        else
        {
            sumsl = "0"; sumjr = "0";
        }
        sumlabel1.Text = "总件数:" + sumsl.Trim() + ", 总金额:" + sumjr.Trim() +
            "元";
    }
    else
        MessageBox.Show("请输入正确的商品编号和日期", "操作提示");
}

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

```

5.14 “用户管理” 功能设计

- (1) 按照教材 P433 的指示，设计窗体和所有控件。其中包括 setuser、setuser1 两个窗体
- (2) setuser 示例代码如下：

```

namespace SM
{
    public partial class setuser : Form
    {
        CommDB mydb = new CommDB();
        public setuser()
        {
            InitializeComponent();
        }
    }
}

```

```

private void setuser_Load(object sender, EventArgs e)
{
    bind();
    dataGridView1.EnableHeadersVisualStyles = false;
    dataGridView1.MultiSelect = false;
    dataGridView1.ReadOnly = true;
    dataGridView1.Columns[0].Width = 130;
    dataGridView1.Columns[1].Width = 130;
    dataGridView1.Columns[2].Width = 130;
}

private void bind()
{
    string mystr;
    mystr = "SELECT * FROM SUser";
    DataSet mydataset = mydb.ExecuteQuery(mystr, "SUser");
    dataGridView1.DataSource = mydataset.Tables["SUser"];
}

private void button1_Click(object sender, EventArgs e) //添加
{
    TempData.tag = 1;
    Form myform = new setuser1();
    myform.ShowDialog();
    bind();
}

private void button2_Click(object sender, EventArgs e) //修改
{
    //修改记录
    TempData.tag = 2;
    if (dataGridView1.SelectedRows.Count > 0)
    {
        string userno =
dataGridView1.SelectedRows[0].Cells[0].Value.ToString().Trim();
        string userpass =
dataGridView1.SelectedRows[0].Cells[1].Value.ToString().Trim();
        string userlevel =
dataGridView1.SelectedRows[0].Cells[2].Value.ToString().Trim();
        Form myform = new setuser1(userno, userpass, userlevel); //带传递数
据调用setuser1的构造函数
        myform.ShowDialog();
        bind();
    }
    else

```

```

        MessageBox.Show("必须选择一个要修改的记录", "操作提示");
    }

    private void button3_Click(object sender, EventArgs e) //删除
    {
        //删除记录
        if (dataGridView1.SelectedRows.Count > 0)
        {
            string userno =
dataGridView1.SelectedRows[0].Cells[0].Value.ToString().Trim();
            DialogResult result = MessageBox.Show("真的要删除" + userno
                + "用户吗?", "操作提示", MessageBoxButtons.YesNo);
            if (result == DialogResult.Yes)
            {
                string mystr = "DELETE SUser WHERE 用户名='" + userno + "'";
                mydb.ExecuteNonQuery(mystr);
                bind();
            }
        }
        else
            MessageBox.Show("必须选择一个要删除的记录", "操作提示");
    }

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

    private void dataGridView1_CellClick(object sender, DataGridViewCellEventArgs
e)
    {
        //单击任何单元时选择该单元所在的行
        if (e.RowIndex >= 0 && e.RowIndex < dataGridView1.Rows.Count)
            dataGridView1.Rows[e.RowIndex].Selected = true;
    }

    private void 添加ToolStripMenuItem_Click(object sender, EventArgs e)
    {
        button1_Click(sender, e);
    }

    private void 修改ToolStripMenuItem_Click(object sender, EventArgs e)
    {
        button2_Click(sender, e);
    }

```

```
    }

    private void 删除ToolStripMenuItem_Click(object sender, EventArgs e)
    {
        button3_Click(sender, e);
    }
}
}
```

setuser1 示例代码如下:

```
namespace SM
{
    public partial class setuser1 : Form
    {
        string usernum, userpass, userlevel;
        public setuser1()
        {
            InitializeComponent();
        }
        public setuser1(string no, string pass, string level)
        {
            InitializeComponent();
            usernum = no;
            userpass = pass;
            userlevel = level;
        }

        private void setuser1_Load(object sender, EventArgs e)
        {
            comboBox1.Items.Add("操作员");
            comboBox1.Items.Add("管理员");
            if (TempData.tag == 1) //添加
            {
                textBox1.Text = "";
                textBox2.Text = "";
                comboBox1.Text = "";
                textBox1.Focus();
            }
            else if (TempData.tag == 2) //修改
            {
                textBox1.Text = usernum;
                textBox1.Enabled = false;
                textBox2.Text = userpass;
                comboBox1.Text = userlevel;
                textBox1.Focus();
            }
        }
    }
}
```

```

    }
}
private void button1_Click(object sender, EventArgs e)
{
    string mystr;
    CommDB mydb = new CommDB();
    if (textBox1.Text != "" && textBox2.Text != "" && comboBox1.Text != "")
    {
        if (TempData.tag == 1)           //添加
        {
            mystr = "SELECT * FROM SUser WHERE 用户名='" + textBox1.Text.Trim()
+ "'";

            int i = mydb.Rownum(mystr);
            if (i == 1)
            {
                MessageBox.Show("输入的用户名重复, 重新输入", "操作提示");
                return;
            }
            mystr = "INSERT INTO SUser(用户名, 密码, 级别) VALUES(' "
+ textBox1.Text.Trim() + "', ' "
+ textBox2.Text.Trim() + "', ' "
+ comboBox1.Text.Trim() + "')";
            mydb.ExecuteNonQuery(mystr);
        }
        else if (TempData.tag == 2)       //修改
        {
            mystr = "UPDATE SUser SET 密码=' "
+ textBox2.Text.Trim() + "', 级别=' "
+ comboBox1.Text.Trim() + "' WHERE 用户名=' "
+ textBox1.Text.Trim() + "'";
            mydb.ExecuteNonQuery(mystr);
        }
        this.Close();
    }
    else
        MessageBox.Show("必须输入所有数据项", "操作提示");
}
private void button2_Click(object sender, EventArgs e)
{
    this.Close();
}
}
}

```

5.15 “设置商品类别” 功能设计

(1) 按照教材 P438 的指示，设计窗体和所有控件。

(2) 示例代码如下：

```
namespace SM
{
    public partial class setsplb : Form
    {
        CommDB mydb = new CommDB();
        int tag;           //1:添加, 2: 修改
        string recno;      //记录编号
        public setsplb()
        {
            InitializeComponent();
        }

        private void setsplb_Load(object sender, EventArgs e)
        {
            bind();
            tag = 1;       //表示添加操作
        }

        private void bind()
        {
            string mystr;
            mystr = "SELECT * FROM Prodsort";
            DataSet mydataset = mydb.ExecuteQuery(mystr, "Prodsort");
            dataGridView1.DataSource = mydataset.Tables["Prodsort"];
            dataGridView1.EnableHeadersVisualStyles = false;
            dataGridView1.MultiSelect = false;
            dataGridView1.ReadOnly = true;
            dataGridView1.Columns[0].Width = 140;
            dataGridView1.Columns[1].Width = 100;
            dataGridView1.Columns[2].Width = 100;
            taglabel.Text = "添加操作";
        }

        private void button1_Click(object sender, EventArgs e)
        {
            string mystr;
            if (tag == 1)    //当前为添加操作
            {
                if (textBox1.Text != "" && textBox2.Text != "")
                {

```

```

        mystr = "SELECT * FROM Prodsort WHERE 分类=' "
            + textBox1.Text.Trim() + "' AND 子类=' "
            + textBox2.Text.Trim() + "' ";
        int i = mydb.Rownum(mystr);
        if (i == 1)
        {
            MessageBox.Show("商品分类重复, 重新输入", "操作提示");
            return;
        }
        mystr = "INSERT INTO Prodsort(分类,子类) VALUES(' "
            + textBox1.Text.Trim() + "', ' "
            + textBox2.Text.Trim() + "')";
        mydb.ExecuteNonQuery(mystr);
        bind();
        textBox2.Text = "";
        textBox2.Focus();
    }
    else
        MessageBox.Show("必须输入所有数据项", "操作提示");
}
else if (tag == 2) //当前为修改操作
{
    mystr = "UPDATE Prodsort SET 分类=' " + textBox1.Text.Trim() + "' , "
        + "子类=' " + textBox2.Text.Trim() + " "
        + "' WHERE 编号=' " + recno + "' ";
    mydb.ExecuteNonQuery(mystr);
    bind();
    tag = 1; //修改后恢复为添加操作
    taglabel1.Text = "添加操作";
    textBox1.Text = "";
    textBox2.Text = "";
    textBox1.Focus();
}
}

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

private void dataGridView1_CellClick(object sender, DataGridViewCellEventArgs
e)
{
    //单击任何单元时选择该单元所在的行

```

```

        if (e.RowIndex >= 0 && e.RowIndex < dataGridView1.Rows.Count)
            dataGridView1.Rows[e.RowIndex].Selected = true;
    }

    private void 删除ToolStripMenuItem_Click(object sender, EventArgs e)
    {
        //删除记录
        if (dataGridView1.SelectedRows.Count > 0)
        {
            string flno =
dataGridView1.SelectedRows[0].Cells[0].Value.ToString().Trim();
            DialogResult result = MessageBox.Show("真的要删除" + flno
                + "编号的分类记录吗?", "操作提示", MessageBoxButtons.YesNo);
            if (result == DialogResult.Yes)
            {
                string mystr = "DELETE Prodsort WHERE 编号='" + flno + "'";
                mydb.ExecuteNonQuery(mystr);
                bind();
            }
        }
        else
            MessageBox.Show("必须选择要删除的记录", "操作提示");
    }

    private void 修改ToolStripMenuItem_Click(object sender, EventArgs e)
    {
        //修改记录
        if (dataGridView1.SelectedRows.Count > 0)
        {
            tag = 2;           //表示修改操作
            string no =
dataGridView1.SelectedRows[0].Cells[0].Value.ToString().Trim();
            string fl =
dataGridView1.SelectedRows[0].Cells[1].Value.ToString().Trim();
            string zl =
dataGridView1.SelectedRows[0].Cells[2].Value.ToString().Trim();
            taglabel1.Text = "修改编号" + no;
            recno = no;
            textBox1.Text = fl;
            textBox2.Text = zl;
        }
        else
            MessageBox.Show("必须选择要修改的记录", "操作提示");
    }

    private void 复制分类ToolStripMenuItem_Click(object sender, EventArgs e)

```



```

    {
        if (dataGridView1.SelectedRows.Count > 0)
        {
            string fl =
dataGridView1.SelectedRows[0].Cells[1].Value.ToString().Trim();
            textBox1.Text = fl;
        }
        else
            MessageBox.Show("必须选择要复制的记录", "操作提示");
    }
}
}

```

5.16 “系统初始化” 功能设计

(1) 按照教材 P441 的指示，设计窗体和所有控件。

(2) 示例代码如下：

```

namespace SM
{
    public partial class init : Form
    {
        CommDB mydb = new CommDB();
        public init()
        {
            InitializeComponent();
        }

        private void button2_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            DialogResult result = MessageBox.Show("真的要进行系统初始化吗?", "操作提示",
            MessageBoxButtons.YesNo);
            if (result == DialogResult.Yes)
            {
                deltable("Area");
                deltable("Prodsort");
                deltable("Customer");
                deltable("Product");
                deltable("Sale");
            }
        }
    }
}

```

```

        deltable("SUser");
        string mystr = "INSERT INTO SUser(用户名, 密码, 级别)"
            + " VALUES('system','manager','管理员)";
        mydb.ExecuteNonQuery(mystr);
        MessageBox.Show("系统初始化完毕, 下次以system/manager管理员进入本系统",
            "操作提示");
        this.Close();
    }
    else
        this.Close();
}

private void deltable(string tn)
{
    string mystr = "DELETE " + tn;
    mydb.ExecuteNonQuery(mystr);
}
}
}

```

5.17 帮助功能设计

(1) 按照教材 P443 的指示, 设计窗体和所有控件。其中包括“关于……”功能设计和“联系信息”功能设计两个窗体。

(2) “关于……”功能设计示例代码如下:

```

namespace SM
{
    public partial class about : Form
    {
        public about()
        {
            InitializeComponent();
        }

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

```

“联系信息”功能设计示例代码如下:

```

namespace SM

```

```
{  
    public partial class corporation : Form  
    {  
        public corporation()  
        {  
            InitializeComponent();  
        }  
  
        private void button1_Click(object sender, EventArgs e)  
        {  
            this.Close();  
        }  
    }  
}
```