实践教程部分

第1章 概述

1.1 实验目的及要求

(1) 熟悉 Visual Studio 2010(VS)开发环境。
(2) 掌握 Console 类项目的新建与运行。
(3) 掌握 WinForm 类项目的新建与运行。
(4) 控制台下的简单输出与输入。
(5) WinForm 下的 MessageBox、Label、Button 等的简单使用。

1.2 实验内容

1. 熟悉 Visual Studio 2010(VS)开发环境

下面以 WinForm 项目来讲解,主要分为标题栏、菜单栏、工具栏、工具箱、工作区、解决方案管理器、属性窗口等,如图 1-1 所示。

图 1-1 VS 开发环境

请熟悉上述各个菜单、解决方案管理器等的功能。Console 下的开发环境请读者自行对照这里来熟悉。

2. 掌握 Console 类项目的新建与运行

要创建 Console 类项目，首先需要选择新建项目菜单命令，如图 1-2 所示。

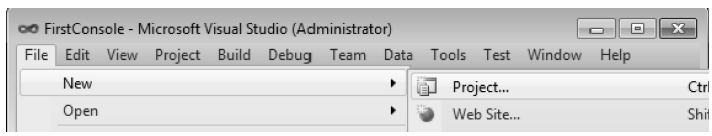

图 1-2　新建项目

其次在新建项目的对话框中选择相应的类型即可，如图 1-3 所示。

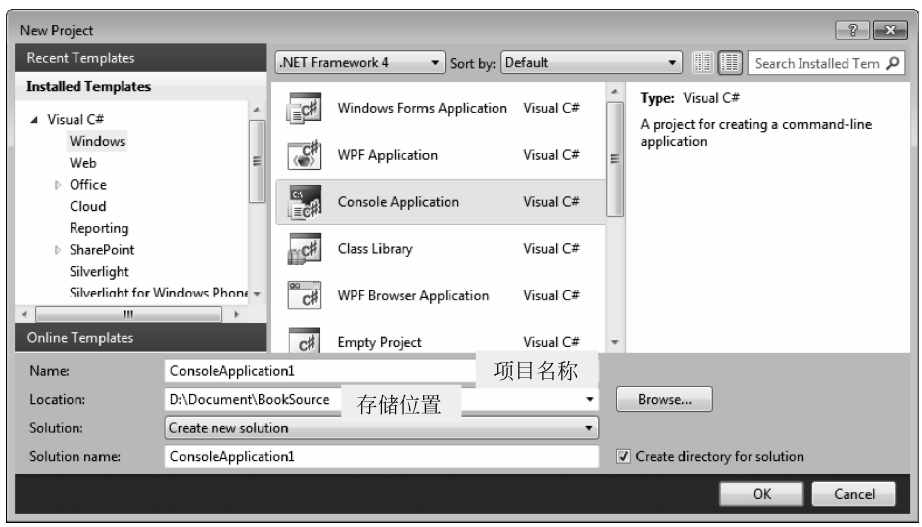

图 1-3　选择 Console Application

为了使项目方便维护，一般还应该在图 1-3 中设置项目的名称和存储位置(已在上图中标注)。

3. 掌握 WinForm 类项目的新建与运行

WinForm 类项目的创建与 Console 类项目的新建方式一样，只需要选择 Windows Forms Application 即可，其他与 Console 类项目完全一样，如图 1-4 所示。

4. 控制台下的简单输出与输入

在控制台下，输出主要是通过 Console. WriteLine()来实现的。而输入则主要是通过 Console. ReadLine()来实现的。

新建 Console 项目，在 Program. cs 的 Main 函数中输入如下语句。

```
class Program
{
    static void Main(string[] args)
    {
        //如下是手动输入的代码
```

```
            Console.WriteLine("请输入您的大名：");
            string sName = Console.ReadLine();   //获取用户的输入
            Console.WriteLine(sName + ",你好");
            Console.ReadLine();
        }
    }
```

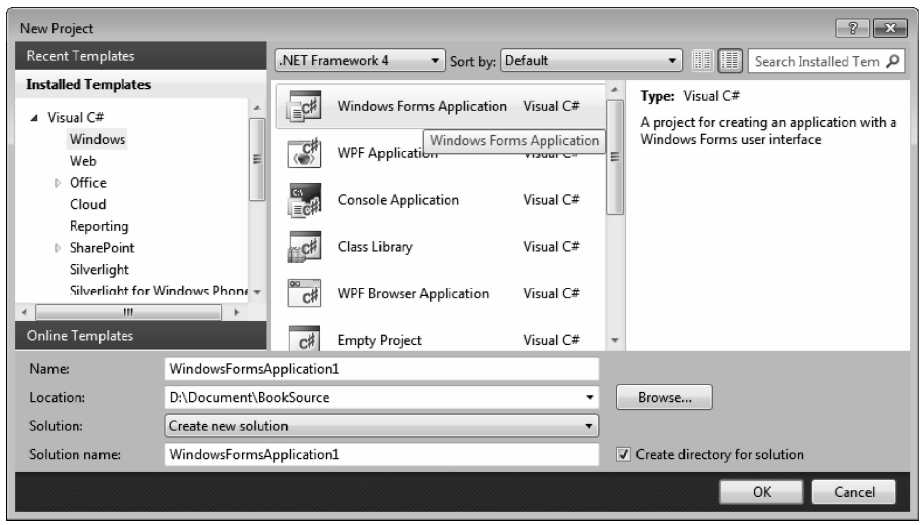

图 1-4 选择 Windows Forms Application

按 F5 键运行程序，效果如图 1-5 所示。

图 1-5 Console 程序

5. WinForm 下的 MessageBox、Label、TextBox、Button 等的简单使用

首先新建 WinForm 项目，往窗体上拖放一个 Label 控件，然后右击该 Label 控件，在右键快捷菜单中选择“属性”，在“属性窗口”中将 label1 的 Text 属性设置为“请输入姓名：”。

然后往 label1 的后面拖入一个 TextBox 控件，接着往 TextBox 控件后拖入一个 Button 控件，依上述方法，将 Button 的 Text 属性修改为“提交”。界面如图 1-6 所示。

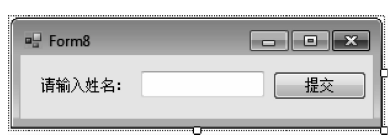

图 1-6 WinForm 界面

双击“提交”按钮，输入如下代码：

```
private void button1_Click(object sender, EventArgs e)
{
    MessageBox.Show("你好," + textBox1.Text + "!");
}
```

按 F5 键运行结果如图 1-7 所示。

图 1-7　WinForm 程序

➢ 请读者总结实验心得。

第2章

数据类型与运算符

2.1 实验目的及要求

(1) 熟悉常见数据类型。
(2) 熟悉数据类型的转换。
(3) 掌握常见的运算符。

2.2 实验内容

计算矩形面积和周长，要求用户输入2个整型数值(即矩形边长)，然后计算并输出结果。同时计算该矩形外接圆的面积和周长。

该题考查了数据类型及其转换，且考查了运算符，同时使用了Math.Sqrt()完成开方运算。

由于矩形边长强制使用整型值，所以边长变量使用int定义即可。另外由于使用Console.ReadLine()接收的输入为字符串，所以需要完成向整型值的转换，这可以使用int.Parse()或者Convert的相关方法完成。

由于矩形的边长为整数，所以其周长和面积也为整数，不过圆的周长和面积为浮点数，所以为了不再单独为外接圆定义周长和面积变量，故将矩形的周长和面积变量都定义为double类型。

矩形的周长计算公式为：2×(长＋宽)。

矩形的面积计算公式为：长×宽。

外接圆的面积计算公式为：PI×r^2(其中r^2＝(长×长＋宽×宽)/4)。

外接圆的周长计算公式为：2×PI×r。

另外，由于计算圆的周长和面积时，需要用到圆周率的值，该值适合定义为常量。

操作步骤如下。

首先新建Console项目，在Program.cs中输入如下代码：

```
class Program
{
```

```
    static void Main(string[] args)
    {
        Console.Write("请输入第一个数值并回车");
        string s1 = Console.ReadLine();
        Console.Write("请输入第二个数值并回车");
        string s2 = Console.ReadLine();
        //将用户的输入转换为数值
        int d1 = int.Parse(s1);
        int d2 = int.Parse(s2);

        //计算矩形的周长和面积
        double c = 2 * (d1 + d2);
        double a = d1 * d2;

        Console.WriteLine("矩形的周长为: " + c.ToString() + "; 面积为:" + a.ToString());

        //计算外接圆半径的平方值
        double r = d1 * d1/4 + d2 * d2/4;
        const double PI = 3.14;
        //计算外接圆的面积
        a = PI * r;
        c = 2 * PI * Math.Sqrt(r);

        Console.WriteLine("矩形外接圆的周长为: " + c.ToString() + "; 面积为:" + a.ToString());

        Console.ReadKey();
    }
}
```

程序运行结果如图 2-1 所示。

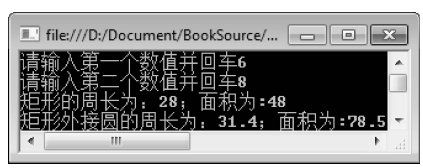

图 2-1　矩形及外接圆的面积和周长计算

上述程序也可以进行适当的简化，例如 int d1 = int.Parse(Console.ReadLine());。

第3章 程序控制

3.1 实验目的及要求

（1）掌握三种类型的程序控制语句：选择、循环、跳转。

（2）掌握 continue、break、return 的特性及适用场合。

3.2 实验内容

（1）从键盘上输入两个整数，由用户回答它们的和、差、积、商和取余的运算结果，并统计出正确答案的个数。

该题主要考查算术运算符及 if 语句。代码如下：

```
class Program
{
    //从键盘上输入两个整数,由用户回答它们的和、差、积、商和取余的运算结果,并统计出正确答
//案的个数
    static void Main(string[] args)
    {
        Console.WriteLine("请输入两个整数,每输入一个按回车键结束输入");
        int i = int.Parse(Console.ReadLine());
        int j = int.Parse(Console.ReadLine());

        int iCount = 0;  //记录答案正确的个数

        Console.Write("请回答和: ");
        if (i + j == int.Parse(Console.ReadLine()))
            iCount++;

        Console.Write("请回答差: ");
        if (i - j == int.Parse(Console.ReadLine()))
            iCount++;

        Console.Write("请回答积: ");
        if (i * j == int.Parse(Console.ReadLine()))
```

```
            iCount++;

        Console.Write("请回答商: ");
        if ((double)i / j == double.Parse(Console.ReadLine()))
            iCount++;

        Console.Write("请回答余数: ");
        if (i % j == int.Parse(Console.ReadLine()))
            iCount++;

        Console.WriteLine("\n 你回答正确了" + iCount.ToString() + "个");
        Console.ReadKey();
    }
}
```

程序运行结果如图 3-1 所示。

图 3-1　和、差、积、商和取余运算

上述求商时，注意写法。对上例，i/j 的结果是整数，故需要(double)i/j。

(2) 完成一个猜数字游戏，要求实现如下要求。

- 根据用户输入的两个正整数求乘积 M(应保证乘积＞100)。
- 再根据用户输入的另外一个值(N)来确定一个猜数字游戏的范围(介于 $M-N$ 和 $M+N$ 之间，且 $0<N<M$)。
- 然后在($M-N$)和($M+N$)之间生成一个数字 T，此数字即需要用户猜测的答案。
- 当用户输入的答案小于 T 时，提示他输入小了；如果用户猜测的数字大于 T，则提示他输入大了，此过程一直进行到用户猜测正确为止。

为帮助读者理解题意，举例说明如下。

- 让用户输入两个数，假设输入的数分别为 10、15，则 $M=150$。
- 再让用户输入 N，假设 N 为 50，则 $N=50$；$M-N=100$，$M+N=200$；也就是说告诉用户猜数字时在 100～200 之间猜测即可。
- 根据上步得到的范围 100～200，则此时可以随机产生一个介于 100～200 之间的数字，此数字即为用户所要猜测的数。
- 假如随机产生的数字为 180，此后不停地让用户猜测答案，用户的输入比答案大时，提示太大；用户的输入比答案小时，提示太小，一直持续到用户猜对为止。

提示：随机数的产生。

```
Random i = new Random();
int iResult = i.Next(min, max);
```

则 iResult 即是介于 min 和 max 之间的一个随机整数，不能取到 max。

该题稍微复杂些，需要仔细理解题意。

考查了 if 选择和 while 循环的使用，并复习了算术运算符的使用。其关键点有两个：

第一,指定范围内随机数的产生;第二,使用 while 循环实现多次猜测直到猜测正确为止。

```
class Program
{
    static void Main(string[] args)
    {
        int a, b, m, n;   //a、b 表示用户输入第一次的两个数,m、n 是用来计算取值范围的两个数
        int p, q, s;      //p、q 是表示产生的随机数,s 表示用户猜测的随机数

        Console.WriteLine("请输入数 a 和 b!");
        a = Convert.ToInt32(Console.ReadLine());
        b = Convert.ToInt32(Console.ReadLine());
        m = a * b;
        try
        {
            if (m < 100)
            {
                throw new Exception("输入的 a 和 b 的值太小!");
            }
        }
        catch (System.Exception e)
        {
            Console.WriteLine(e.Message);
            Console.WriteLine("please input any key to continue!");
            Console.ReadKey();
            Environment.Exit(0);
        }

        Console.WriteLine("请输入大于 0,小于" + m + "的数 n!");
        n = Convert.ToInt32(Console.ReadLine());
        p = m + n;
        if (m > n)
            q = m - n;
        else
            q = n - m;

        Console.WriteLine("数值在" + q + "和" + p + "之间,不能取到" + p);
        Random i = new Random();
        int iResult = i.Next(q, p);
        Console.WriteLine("请输入你猜测的数的值!");

        bool c = true;
        while (c)
        {
            s = Convert.ToInt32(Console.ReadLine());
            if (s > iResult)
            {
                Console.WriteLine("你输入的数字过大!");
                Console.WriteLine("请重新输入你猜测的数的值!");
            }
            if (s < iResult)
```

```
            {
                Console.WriteLine("你输入的数字太小!");
                Console.WriteLine("请重新输入你猜测的数的值!");
            }
            if (s == iResult)
            {
                Console.WriteLine("恭喜你,回答正确!");
                c = false;
            }
        }
        Console.ReadKey();
    }
}
```

程序运行结果如图 3-2 所示。

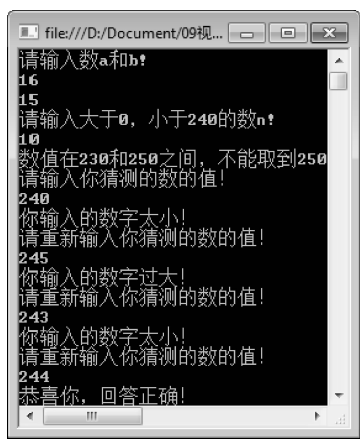

图 3-2　猜数字游戏

如上的 try…catch…块可以改造：当用户输入的值太小时，让用户继续输入，直到输入的 a 和 b 满足条件才执行后面的猜数字游戏。

上面通过变量 c 的值来控制何时退出循环，当用户猜测正确时即退出游戏。可以将其改造为：在猜对后，是继续下次游戏还是退出游戏。

第 4 章

面向对象基础

4.1 实验目的及要求

(1) 掌握面向对象的特性：封装、继承、多态。

(2) 掌握类成员：字段、属性、方法、构造函数、委托等。

(3) 理解 public、private、static 等关键字。

(4) 理解接口、抽象方法、虚方法、重载、重写等术语的含义。

4.2 实验内容

(1) 实现一个简单的数学运算类。要求如下。

- 定义一个类 SimpleMath；
- 为类编写静态方法，分别完成加、减、乘、除、开平方、幂运算；
- 定义三个字段，分别代表两个操作数和一个操作符；
- 定义几个普通方法，来完成加、减、乘、除等运算；
- 在需要的情况下可以自行选择构造函数或者重载。

该题比较简单，主要练习 static 的使用，另外练习了字段、方法及构造函数等的使用。此外还复习了 switch 选择语句的使用。

参考代码如下：

```
class SimpleMath
{
    double x, y;
    string op;

    public static void Add(double x, double y)
    {
        Console.WriteLine(x + "+" + y + "=" + (x + y));
    }

    public static void Sub(double x, double y)
```

```
    {
        Console.WriteLine(x + "-" + y + "=" + (x - y));
    }

    public static void Mul(double x, double y)
    {
        Console.WriteLine(x + "×" + y + "=" + (x * y));
    }

    public static void Div(double x, double y)
    {
        Console.WriteLine(x + "÷" + y + "=" + (x / y));
    }

    public static void Chengfang(double x, double y)
    {
        Console.WriteLine(x + "^" + y + "=" + Math.Pow(x, 1 / y));
    }

    public static void Mi(double x, double y)
    {
        Console.WriteLine(x + "$" + y + "=" + Math.Pow(x, y));
    }

    static void Main(string[] args)
    {
        do
        {
            SimpleMath simplemath = new SimpleMath();
            Console.WriteLine("\n输入两个数:");
            simplemath.x = Convert.ToDouble(Console.ReadLine());
            simplemath.y = Convert.ToDouble(Console.ReadLine());
            Console.WriteLine();
            Console.WriteLine("运算方式:\t加法:+\t减法:-\t乘法:*\t除法:/\t开方:^\
t幂:$");
            simplemath.op = Console.ReadLine();
            Console.WriteLine("选择:" + simplemath.op);

            switch (simplemath.op)
            {
                case "+":
                    Add(simplemath.x, simplemath.y);
                    break;
                case "-":
                    Sub(simplemath.x, simplemath.y);
                    break;
                case "*":
                    Mul(simplemath.x, simplemath.y);
                    break;
                case "/":
                    Div(simplemath.x, simplemath.y);
```

```
                    break;
                case "^":
                    Chengfang(simplemath.x, simplemath.y);
                    break;
                case "$":
                    Mi(simplemath.x, simplemath.y);
                    break;
            }
            Console.Write("继续(y/n)?");
        } while (Console.ReadLine() == "y");
    }
}
```

程序运行效果如图 4-1 所示。

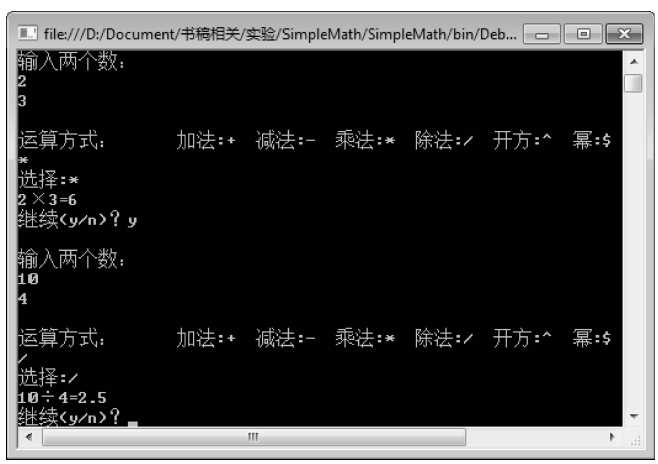

图 4-1 简单的数学运算类

(2) 实现一个矩形类,并创建其实例并使用。要求如下。

- 编写一个矩形类 rectangle。
- 字段成员为:length、width;且可供实例访问。
- 三个构造函数,其中第一个无参构造函数将长宽字段成员赋 0,第二个有参构造函数根据用户实例化传入的参数给长、宽赋值;第三个构造函数传入面积值。
- 3 个方法分别为:①判断该矩形是否为正方形;②计算周长;③计算面积。
- 用户采用第三个构造函数时,根据传入的面积参数,分解出一组长、宽最接近的值。
- 假设上述值都为整数。

该题目也比较简单,涉及的知识点有字段、构造函数、运算符、整数分解、循环语句等,当然读者可以在上述基础上增加属性等。参考代码如下:

```
class Rectangle
{
    public int Length;
    public int Width;

    //方法 1  判断矩形是否为正方形
```

```
        public bool IsSquare()
        {
            if (Length == Width)
                return true;
            else
                return false;
        }
        //方法 2  计算周长
        public int Perimeter()
        {
            int perimeter = 2 * (Length + Width);
            return perimeter;
        }
        //方法 3  计算面积
        public int Area()
        {
            int area = Length * Width;
            return area;
        }

        //构造函数
        //将长、宽字段赋值为 0
        public Rectangle()
        {
            Length = 0;
            Width = 0;
        }
        //有参构造函数,根据用户实例化传入的参数给长、宽赋值
        public Rectangle(int length, int width)
        {
            this.Length = length;
            this.Width = width;
        }
        //接收传入面积以及分解面积
        public Rectangle(int area)
        {
            int i, min;
            min = area;
            //分解面积,得到长和宽
            for (i = Convert.ToInt32( Math.Sqrt(area)); i >= 1; i-- )
            {
                if (area % i == 0)
                {
                    Width = i;
                    Length = area / Width;
                    break;
                }
            }
        }
    }
```

演示调用代码如下：

```
class Program
{
    static void Main(string[] args)
    {

        int length = 0,width = 0,perimeter,area;
        int length2, width2, mianji2 = 0;

        Console.WriteLine("请输入矩形的长：");

        try
        {
            length = Convert.ToInt32(Console.ReadLine());
        }
        catch (Exception e)
        {

            Console.WriteLine(e.Message);
            Environment.Exit(0);     //退出
        }

        Console.WriteLine("请输入矩形的宽：");
        try
        {
         width = Convert.ToInt32(Console.ReadLine());
        }
        catch (Exception e)
        {
            Console.WriteLine(e.Message);
            Environment.Exit(0);     //退出
        }

        Rectangle a = new Rectangle(length,width);
        if (a.IsSquare() == true)
        {
            Console.WriteLine("是正方形");
        }

        perimeter = a.Perimeter();
        Console.WriteLine("矩形的周长为：" + perimeter);
        area = a.Area();
        Console.WriteLine("矩形的面积为：" + area);

        Console.WriteLine("请输入指定矩形的面积：");
        try
        {
            mianji2 = Convert.ToInt32(Console.ReadLine());
        }
        catch (Exception e)
```

```
        {
            Console.WriteLine(e.Message);
            Environment.Exit(0);     //退出
        }

        Rectangle b = new Rectangle(mianji2);
        length2 = b.Length;
        width2 = b.Width;
        Console.WriteLine("分解后得到的长和宽分别为:" + length2 + "和" + width2);

        Console.ReadLine();
    }
}
```

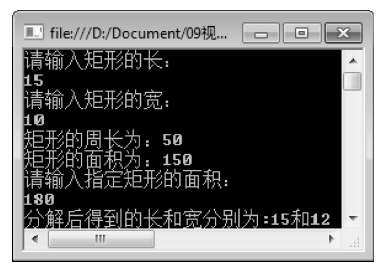

图 4-2　矩形类演示

程序运行结果如图 4-2 所示。

(3) 编写一个类 Cal,要求如下。

- 此类具有 2 个属性 OpNum1、OpNum2,一个无参虚方法 CalExpression,用于计算两数的结果。
- 一个接口 CalIt,其中包含一个方法 CalResult,用于计算两数的结果。
- 编写四个类,均继承于 Cal,分别完成两个操作数的四则运算。
- 编写一个类 CalOprs,包含一个方法 GetOpr,方法返回类型为 Cal,参数为加、减、乘、除之一。
- 编写 Main 中的测试代码部分。

此题目涉及的知识点有虚方法、接口、继承、属性、多态等。由于虚方法和接口的功能都是实现两个操作数的相应的数值运算,故此处仅以接口为例来实现,使用虚方法的方式请读者自行实现。如下分块介绍各个类或者接口的代码。

接口定义代码如下:

```
//ICalIt 接口
interface ICalIt
{
    double CalResult();
}
```

Cal 类代码如下:

```
//Cal 类, 该类实现了接口
class Cal:ICalIt
{
    private double opNum1;
    private double opNum2;

    public double OpNum1
    {
        get { return opNum1; }
        set { opNum1 = value; }
    }
```

```
    public double OpNum2
    {
        get { return opNum2; }
        set { opNum2 = value; }
    }

    public Cal()
    {
        this.opNum1 = 0;
        this.opNum2 = 0;
    }
    public Cal(double opNum1, double opNum2)
    {
        this.opNum1 = opNum1;
        this.opNum2 = opNum2;
    }
    //该方法定义为虚方法,将在子类中重写来实现各种运算
    public virtual double CalResult()
    {
        return 0;
    }
}
```

下面是实现四则运算的 4 个类的代码。

```
//实现加法的类
class CalAdd : Cal
{
    public CalAdd()
        : base()
    {
    }
    public CalAdd(double opNum1, double opNum2)
        : base(opNum1, opNum2)
    {
    }
    //重写基类方法
    public override double CalResult()
    {
        return this.OpNum1 + this.OpNum2;
    }
}

//实现减法的类
class CalSub : Cal
{
    public CalSub()
        : base()
    {
    }
```

```
    public CalSub(double opNum1, double opNum2)
        : base(opNum1, opNum2) { }
    //重写基类方法
    public override double CalResult()
    {
        return this.OpNum1 - this.OpNum2;
    }
}

//实现乘法的类
class CalMul : Cal
{
    public CalMul()
        : base()
    {
    }
    public CalMul(double opNum1, double opNum2)
        : base(opNum1, opNum2) { }
    //重写基类方法
    public override double CalResult()
    {
        return this.OpNum1 * this.OpNum2;
    }
}

//实现除法的类
class CalDiv : Cal
{
    public CalDiv()
        : base()
    {
    }
    public CalDiv(double opNum1, double opNum2)
        : base(opNum1, opNum2) { }
    //重写基类方法
    public override double CalResult()
    {
        if (this.OpNum2 == 0)
        {
            throw new Exception("除数不能为零.");
        }
        return this.OpNum1 / this.OpNum2;
    }
}
```

如下是 CalOprs 类，该类根据传入的运算符，实例化相应的运算类。代码如下：

```
class CalOprs
{
    public static Cal GetOpr(string oprsType)
    {
```

```
        if (oprsType == "+")
            return new CalAdd();
        else if (oprsType == "-")
            return new CalSub();
        else if (oprsType == "*")
            return new CalMul();
        else if (oprsType == "/")
            return new CalDiv();
        else
            return null;
    }
}
```

最后是供演示调用的类。代码如下：

```
class Program
{
    static void Main(string[] args)
    {
        Console.WriteLine("运算方式,\t 加法:+ 、\t 减法:- 、\t 乘法:* 、\t 除法:/");
        string op = Console.ReadLine();

        Cal cal = null;
        switch (op)
        {
            case "+":
                cal = CalOprs.GetOpr("+");
                break;
            case "-":
                cal = CalOprs.GetOpr("-");
                break;
            case "*":
                cal = CalOprs.GetOpr("*");
                break;
            case "/":
                cal = CalOprs.GetOpr("/");
                break;
            default:
                break;
        }

        Console.WriteLine("请输入两个操作数,每次以回车键结束输入");
        cal.OpNum1 = double.Parse(Console.ReadLine());
        cal.OpNum2 = double.Parse(Console.ReadLine());
        Console.WriteLine("{0}{1}{2}={3}", cal.OpNum1, op,cal.OpNum2, cal.CalResult());

        Console.ReadKey();
    }
}
```

程序运行效果如图 4-3 所示。

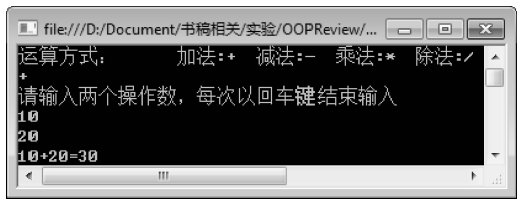

图 4-3　多态、继承等综合演示

此例也使用了一种知名的设计模式，有兴趣的读者自行钻研，此处不再展开。

(4) 编写一个定时触发的事件及所有相关演示程序。详细要求如下。

- 编写一个自定义类 DataEventArgs，继承于 EventArgs，用于在事件处理中传递数据，所需传递数据包含当前时间、当前第几次触发这两项信息。
- 编写一个事件触发类，其中含一个方法 Fire(int i)，根据传入的 i 的次数决定触发多少次事件。
- 编写一个事件接收类，该类中要求完成对多播的演示(即要求有多个处理过程绑定到上述事件)，在事件处理过程中输出通过自定义参数传递过来的数据信息。
- 定时触发的过程可以借助 Thread.Sleep 及循环来实现。

该题主要考查的是事件。下面分块给出代码并给予适当的注释。

首先看 DataEventArgs 类的代码，如下：

```
public class DataEventArgs:EventArgs
{
    private DateTime dt;
    private int count;
    public DateTime TrigTime
    {
        get
        {
            return dt;
        }
    }
    public int Count
    {
        get
        {
            return count;
        }
    }

    public DataEventArgs()
    {

    }
    public DataEventArgs(DateTime dt, int count)
    {
        this.dt = dt;
```

```
        this.count = count;
    }
}
```

事件触发类代码如下：

```
//触发事件类
public class EventTrigger
{
    public event MyDelegate MyDelegateEvent;
    public void Fire(int max)
    {
        //按照指定的此处来触发时间,每次触发的时间间隔为 1000ms
        for (int i = 0; i <= max; i++)
        {
            if (MyDelegateEvent != null)
            {
                MyDelegateEvent(this, new DataEventArgs(DateTime.Now, i));
                Thread.Sleep(1000);
            }
        }
    }
}
```

事件订阅类代码如下：

```
//事件订阅类
public class EventSubscription
{
    public void SubscriptionEvent()
    {
        EventTrigger et = new EventTrigger();
        //订阅
        et.MyDelegateEvent += new MyDelegate(eventTrig_MyDelegateEvent);
        et.Fire(10);   //触发 10 次
    }
    //从 DataEventArgs 的参数 e 中获取第几次触发和各次触发的时间
    void eventTrig_MyDelegateEvent(object sender, DataEventArgs e)
    {
         Console.WriteLine("事件于第{0}次触发,触发时间为:{1}", e.Count, e.TrigTime.
ToString());
    }
}
```

演示调用代码如下：

```
class Program
{
    static void Main(string[] args)
    {
        EventSubscription es = new EventSubscription();
        es.SubscriptionEvent();
```

```
            Console.ReadKey();
        }
    }
```

程序执行效果如图 4-4 所示。

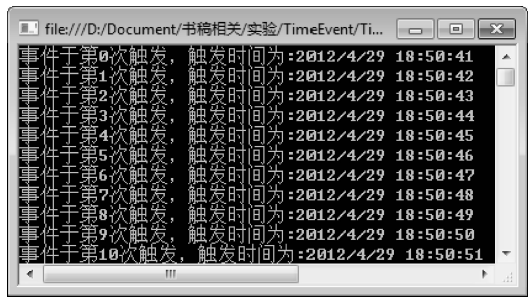

图 4-4　定时触发的事件

第5章

数组

5.1 实验目的及要求

(1) 掌握数组的定义。
(2) 掌握数组的遍历。
(3) 掌握常用排序算法。
(4) 了解 params 关键字。

5.2 实验内容

实现一个简单的数组处理类。要求如下。

- 实现整型数组元素的排序输出；
- 通过重载实现字符数组的排序输出；
- 对整型数组进行求和；
- 对整型数组求最大值；
- 实现字符反转；
- 对整型数组求最小值(使用 params 方式)；
- 若无特别说明,传入参数时,要求传入数组名称。

该题主要考查了数组相关的简单操作。其中求和、求最大值、求最小值都比较简单。

```
class ArrayDemo
{
    //对整型数组进行求和
    public int Sum(int[] a)
    {
        int s = 0;
        for (int i = 0; i < a.Length; i++)
        {
            s = s + a[i];
        }
        return s;
    }
```

```
//对整型数组求最大值
//从数组中逐个比较,挑选最大的值赋给变量 max
public int Max(int[] a)
{
    int max = a[0];
    for (int i = 0; i < a.Length; i++)
    {
        if (a[i] > max)
            max = a[i];
    }
    return max;
}


//对整型数组求最小值
public int Min(int[] a)
{
    int min = a[0];
    for (int i = 0; i < a.Length; i++)
    {
        if (a[i] < min)
            min = a[i];
    }
    return min;
}

//对整型数组进行排序
public void Sort(int[] a)
{
    int t;
    //冒泡法排序
    for (int i = 0; i < a.Length; i++)
    {
        for (int j = 0; j < a.Length - i - 1; j++)
        {
            if (a[j] > a[j + 1])
            {
                t = a[j];                    //交换 a[i]和 a[j]
                a[j] = a[j + 1];
                a[j + 1] = t;
            }
        }
    }
}

//对字符型数组进行排序
public void Sort(char[] b)
{
    char t;
    for (int i = 0; i < b.Length; i++)      //排序
    {
        for (int j = 0; j < b.Length - i - 1; j++)
        {
```

```
                if (b[j] > b[j + 1])
                {
                    t = b[j];                    //交换 a[i]和 a[j]
                    b[j] = b[j + 1];
                    b[j + 1] = t;
                }
            }
        }
    }

    //对字符型数组进行反转
    public void Reverse(char[ ] b)
    {
        char t;
        for (int i = 0, j = b.Length - 1; i < b.Length & j > i; i++, j-- )
        {
            t = b[i];
            b[i] = b[j];
            b[j] = t;
        }
    }
}
```

上述程序的执行结果如图 5-1 所示。

图 5-1 简单数组处理类

其中字符反转的操作,若借助于后文所学的字符串相关知识,将会更简单。

读者可以使用 Array 内置的功能来实现上述部分功能。

第6章

字符串

6.1 实验目的及要求

(1) 字符串相关静态函数。
(2) 字符串相关的实例函数。

6.2 实验内容

请实现对一个文件路径的解析,要求分析出如下几个方面的信息。

- 文件名(不带路径信息);
- 文件后缀;
- 文件所在的磁盘;
- 文件所在的文件夹。

该题主要考查了字符串的相关函数。参考代码如下:

```
class PathParse
{
    //属性
    public string FullFileName
    {
        get;
        set;
    }
    //构造函数
    public PathParse(string sFileName)
    {
        FullFileName = sFileName;
    }
    //获取带后缀的文件名
    public string GetFileName()
    {
        return GetFileName(true);
    }
```

```
    //重载版本,可以决定获取的文件名是否带后缀
    public string GetFileName(bool withExt)
    {
        int i = FullFileName.LastIndexOf('\\');
        if (withExt)
        {
            return FullFileName.Substring(i + 1);
        }
        else
        {
            string s = FullFileName.Substring(i + 1);
            return s.Substring(0, s.LastIndexOf('.'));
        }
    }
    //获取扩展名
    public string GetExtension()
    {
        int i = FullFileName.LastIndexOf('.');
        return FullFileName.Substring(i);
    }
    //获取所在的盘符
    public string GetDrive()
    {
        return FullFileName.Substring(0, 3);
    }
    //获取所在的目录
    public string GetDirectory()
    {
        int i = FullFileName.LastIndexOf('\\');
        return FullFileName.Substring(0,i + 1);
    }
}
```

演示调用程序为：

```
class Program
{
    static void Main(string[] args)
    {
        string sPath = @"D:\Document\教学\09 微软 Demo\09MSDemo\09MSDemo.sln";
        PathParse pp = new PathParse(sPath);
        Console.WriteLine("当前分析的文件名为: " + pp.FullFileName);
        Console.WriteLine("所在的盘符: " + pp.GetDrive());
        Console.WriteLine("所在的文件夹: " + pp.GetDirectory());
        Console.WriteLine("文件名: " + pp.GetFileName());
        Console.WriteLine("不带后缀的文件名: " + pp.GetFileName(false));
        Console.WriteLine("后缀为: " + pp.GetExtension());
        Console.ReadKey();
    }
}
```

代码运行结果如图 6-1 所示。

图 6-1　文件路径解析

上述带后缀的文件名、后缀的获取可以使用 split()函数来实现，请读者自行实现。

由于区分文件名的“\”和区分后缀的“.”应该从字符串的后面开始定位，故使用了 LastIndexOf 函数。