实验教程部分

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) 掌握控制台下的简单输入输出。
(5) 掌握 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 项目为例来讲解。Visual Studio 2010(VS)开发环境界面主要分为标题栏、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

请读者熟悉上述菜单、解决方案管理器等功能。

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 File→New→Project,如图 1-2 所示。

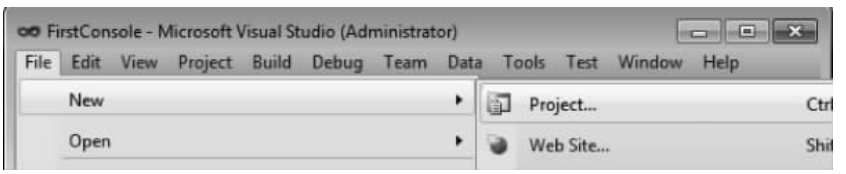

图 1-2　新建项目

其次在 New Project 对话框中选择相应的类型即可,如图 1-3 所示。

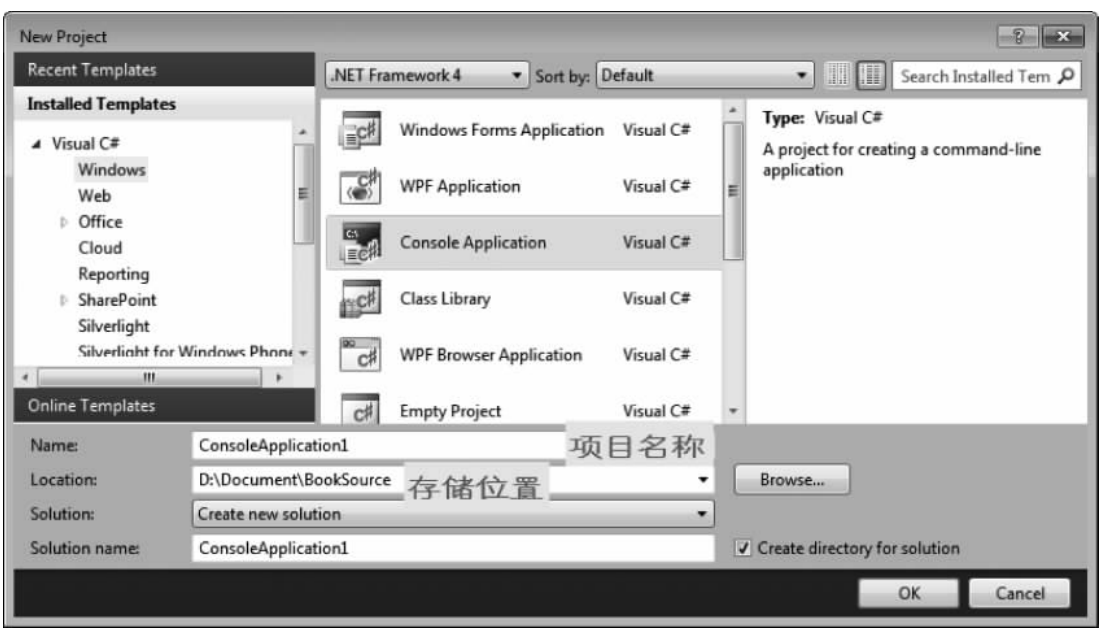

图 1-3　选择 ConsoleApplication

为了便于维护项目,一般还应该在图 1-3 中设置项目名称和存储位置。

3. 掌握 WinForm 类项目的新建与运行

WinForm 类项目的创建只是需要选择 Windows Forms Application 即可,其他与 Console 类项目完全一样,如图 1-4 所示。

4. 掌握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。

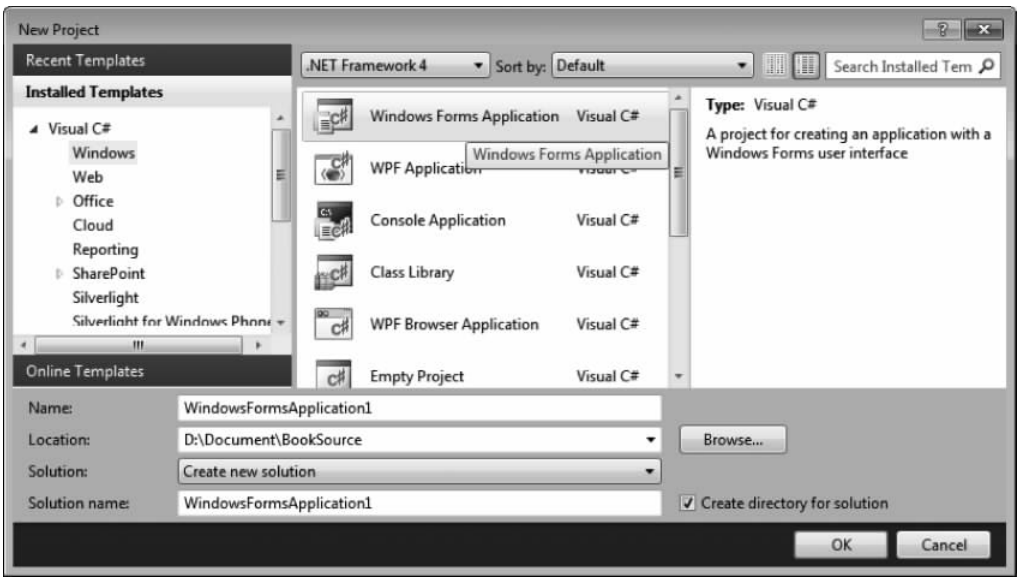

图 1-4　选择 Windows Forms Application

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)
    {
        //以下是手动输入的代码
        Console.WriteLine("请输入您的大名: ");
        string sName = Console.ReadLine();    //获取用户输入
        Console.WriteLine(sName + ",你好");
        Console.ReadLine();
    }
}
```

按 F5 键运行程序，结果如图 1-5 所示。

5. 掌握 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，然后选中并右击该 Label 控件，在弹出的快捷菜单中选择“属性”，在“属性窗口”中将 label1 的 Text 属性设置为“请输入姓名：”。

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

图 1-5　Console 程序界面

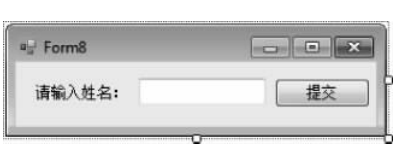

图 1-6　WinForm 程序界面 1

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 键运行程序,结果如图 1-7 所示。

图 1-7　WinForm 程序界面 2

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,要求用户输入两个整型数值(即矩形边长),然后计算并输出计算结果,同时计算该矩形外接圆的面积和周长。

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

由于矩形边长强制使用整型值,因此边长变量使用 int 定义即可。另外由于使用 Console. ReadLine()接收的输入为字符串,因此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,因此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,而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,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按 Enter 键");
        string s1 = Console.ReadLine();
```

```
            Console.Write("请输入第二个数值并按 Enter 键");
            string s2 = Console.ReadLine();
            //将用户输入转换为数值
            int d1 = int.Parse(s1);
            int d2 = int.Parse(s2);

            //计算矩形周长和面积
            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。例如将上述代码中

```
string s1 = Console.ReadLine();
int d1 = int.Parse(s1);
```

简化合并为如下语句：

```
int d1 = int.Parse(Console.ReadLine());
```

则上述部分代码可精简如下：

```
Console.Write("请输入第一个数值并按 Enter 键");
    int d1 = int.Parse(Console.ReadLine());
Console.Write("请输入第二个数值并按 Enter 键");
    int d2 = 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、商和取余运算结果，并统计出正确答案的个数。

该题主要考察算术运算符及 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 Enter 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，应注意写法。对于上例，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)来确定一个猜数游戏范围(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，提示输入小了；当用户输入的数字大于 T 时，则提示输入大了，此过程一直进行到用户猜正确为止。

为帮助读者理解题意，举例说明如下。

- 让用户输入两个数，假设输入的分别为 10、15，则 M＝150。
- 再让用户输入 N，假设输入 50，则 N＝50，M－N＝100，M＋N＝200。也就是说告诉用户猜数时在 100～200 范围内猜测即可。
- 根据上步得到的范围 100～200，此时可以随机产生一个 100～200 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，用户输入的数字大于 180 时，提示太大；用户输入的数字小于 180 时，提示太小，一直持续到用户猜正确为止。

提示：随机数的产生如下。

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

则 iResult 是一个介于 min 和 max 之间的随机整数，不能取到 max。

该题稍微复杂些，需要仔细理解题意。

该题考查 if 选择和 while 循环的使用，并复习算术运算符的使用。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。代码如下：

```
class Program
{
    static void Main(string[] args)
    {
        int a, b, m, n;  //ab 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("您输入的数字过大!");
                    Console.WriteLine("请重新输入你猜测的数的值!");
                }
                if (s < iResult)
                {
                    Console.WriteLine("您输入的数字过小!");
                    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。

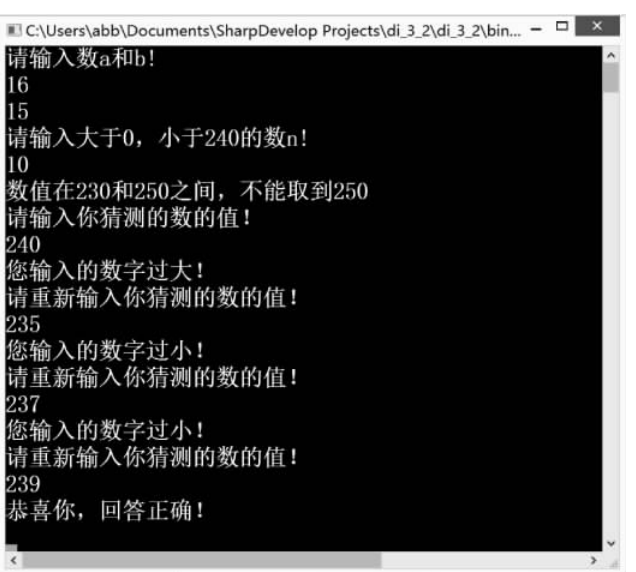

图 3-2　猜数字游戏

上述 try…catch 块可以改造：当用户输入的值过小时，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。

上述通过变量 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，由用户选择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。
- 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，另外还练习字段、方法及构造函数等的使用。此外还复习 switch 选择语句的使用。

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");
    }
}
```

程序运行结果如图 4-1 所示。

图 4-1　简单的数学运算类

(2) 实现一个矩形类,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 和 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。
- 三个方法的功能分别为:判断该矩形是否为正方形;计算周长;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。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 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;
        }
        //接收传入的面积值并分解面积
        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();
        }
    }
```

程序运行结果如图 4-2 所示。

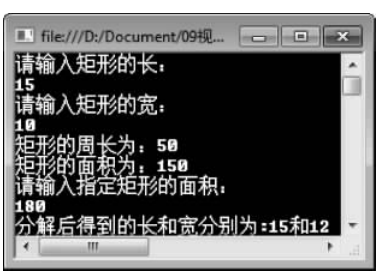

图 4-2　矩形类演示

(3) 编写一个类 Cal,要求如下。

- 此类具有两个属性 OpNum1 和 OpNum2;有一个无参虚方法 CalExpression(),用于计算两数的结果。
- 编写一个接口 ICalIt,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。

此题涉及到的知识点有虚方法、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。下面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;
    }
}
```

下面是分别实现四则运算的四个类的代码。

```
//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()
        {
        }
```