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 2017 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 2017 开发环境。

VS 开发环境界面如图 1-1 所示。

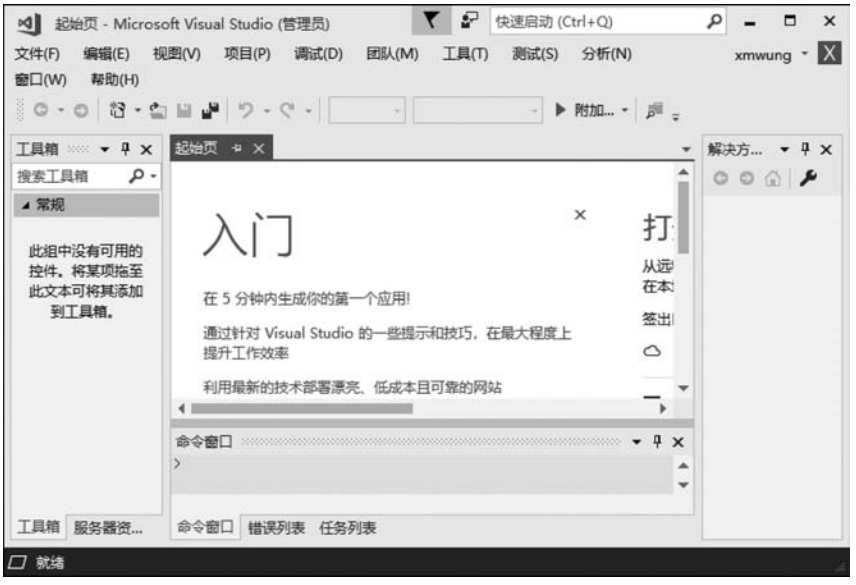

图 1-1　VS 开发环境

（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”对话框中选择相应的类型即可，如图 1-3 所示。

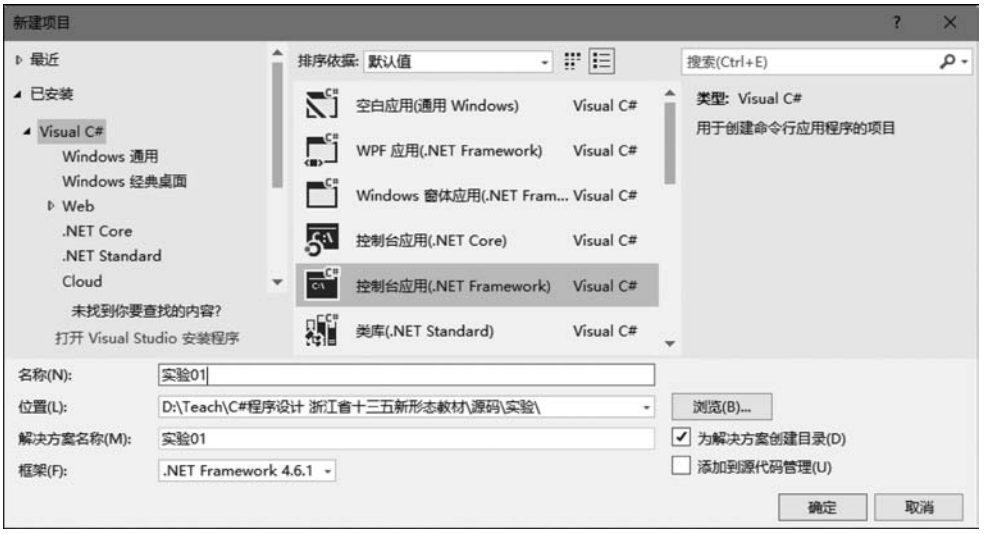

图 1-3　选择控制台应用

为了项目方便维护，一般还应该在图 1-3 中设置项目的名称和存储位置，某些特殊情况下，还应选择合适的框架版本。

（3）控制台下的简单输入输出。

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();          //获取用户输入
        Console.WriteLine(sName + ",你好");
        Console.ReadLine();
    }
}
```

按 F5 键运行程序，效果如图 1-4 所示。

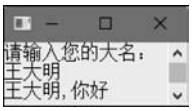

图 1-4　CONSOLE 程序

（4）掌握 Windows 窗体应用（下称 WinForm）类项目的新建与运行。

WinForm 项目的创建与 Console 类项目新建方式一样，只是需要选择“Windows 窗体应用”即可，其他与控制台类项目完全一样，如图 1-5 所示。

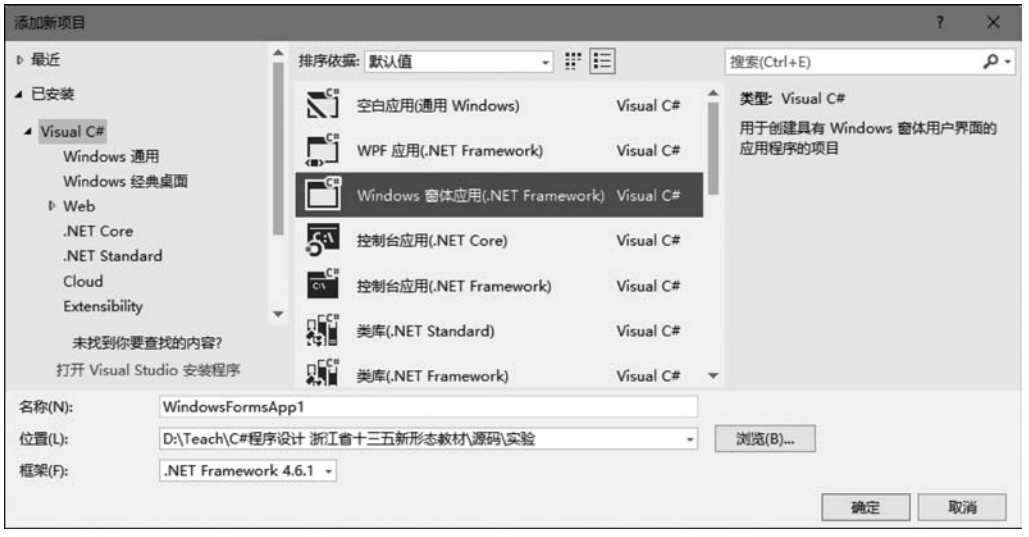

图 1-5　选择 Windows 窗体应用

（5）WinForm 下的 MessageBox、Label、TextBox、Button 等的简单使用。

首先新建 WinForm 项目，往窗体上拖曳一个 Label 控件，然后选中并右击该 Label 控件，在快捷菜单中选择“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 控件的后面拖入一个 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。

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。

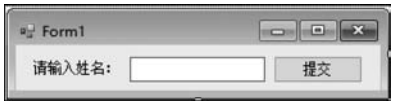

图 1-6　WinForm 界面

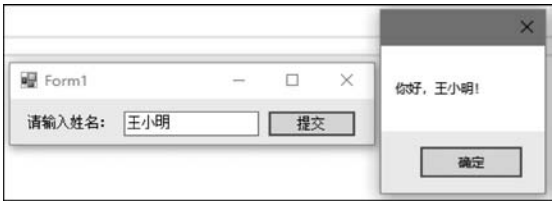

图 1-7　WinForm 程序运行效果

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。

 分析：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()接受的输入为字符串，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：

```
static void Main(string[] args)
{
    const double PI = 3.14;
    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;

        //计算外接圆的面积
        a = PI * r;
        c = 2 * PI * Math.Sqrt(r);
        Console.WriteLine("矩形外接圆周长为:" + c.ToString() + ";面积为:" + a.ToString());

        Console.ReadKey();
    }
```

程序运行结果如图 2-1 所示。

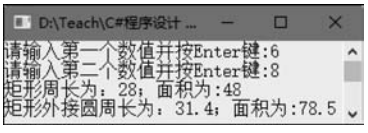

图 2-1　矩形及外接圆的面积和周长计算

上述程序也可以进行适当简化，例如，下列代码

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

可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为：

```
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

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。当回答并反馈完毕，询问用户是否继续，若用户回答 Y，则继续。

 分析：该题主要考查了算术运算符及 if 语句和循环语句。代码如下：

```
static void Main(string[] args)
{
    string sContinue = null;
    do
    {
        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(Math.Abs((double)i/j - double.Parse(Console.ReadLine()))< 0.0001)
                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.Write("需要继续吗?若要继续,请按 Y");
            sContinue = Console.ReadLine();
        } while (sContinue == "Y");

        Console.ReadKey();
    }
```

程序运行结果如图 3-1 所示。

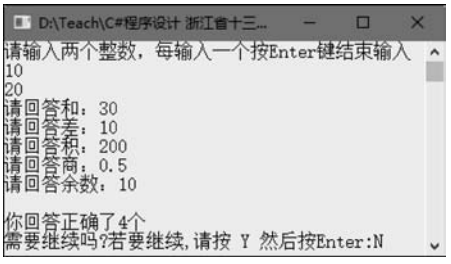

图 3-1　和、差、积、商和取余运算

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

上述程序的逻辑是：输入 Y 继续，输入任意其他内容都将退出，你能改造程序，使其输入 Y 继续，输入 N 退出，输入其他则提醒用户重新输入吗？

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、重载等术语含义。

4.2 实 验 内 容

(1) 实现一个简单的数学运算类 SimpleMath，要求如下。

- 定义 2 个属性，分别代表两个操作数。
- 定义 4 个普通的无参方法，分别完成加减乘除运算。
- 为类编写静态方法，分别完成两个数的加减乘除运算。
- 需要的话可以自行选择构造函数或者重载。

 分析：该题比较简单，主要练习了属性、方法、构造函数的使用。参考代码如下：

```
class SimpleMath
{
    private double x;
    public double X
    {
        get { return x; }
        set { x = value; }
    }

    private double y;
    public double Y
    {
        get { return y; }
        set { y = value; }
    }
    public SimpleMath(double x,double y)
```

```
        {
            this.x = x;
            this.y = y;
        }

        public double Add()
        {
            return x + y;
        }

        public double Sub()
        {
            return x - y;
        }

        public double Mul()
        {
            return x * y;
        }

        public double Div()
        {
            return x / y;
        }


        public static double Add(double x, double y)
        {
            return x + y;
        }

        public static double Sub(double x, double y)
        {
            return x - y;
        }

        public static double Mul(double x, double y)
        {
            return x * y;
        }

        public static double Div(double x, double y)
        {
            return x / y;
        }
    }
```

调用代码如下：

```
static void Main(string[] args)
{
    Console.WriteLine("输入两个数:");
    double x = double.Parse(Console.ReadLine());
    double y = double.Parse(Console.ReadLine());
    SimpleMath math = new SimpleMath(x,y);
    Console.WriteLine("{0}+{1}={2}",x,y,math.Add());
    Console.WriteLine("{0}-{1}={2}", x, y, math.Sub());
    Console.WriteLine("{0}*{1}={2}", x, y, math.Mul());
    Console.WriteLine("{0}/{1}={2}", x, y, math.Div());

    Console.WriteLine("{0}+{1}={2}", x, y, SimpleMath.Add(x,y));
    Console.WriteLine("{0}-{1}={2}", x, y, SimpleMath.Sub(x, y));
    Console.WriteLine("{0}*{1}={2}", x, y, SimpleMath.Mul(x, y));
    Console.WriteLine("{0}/{1}={2}", x, y, SimpleMath.Div(x, y));

    Console.Read();
}
```

程序运行效果如图 4-1 所示。

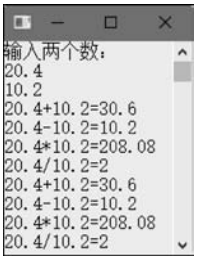

图 4-1　简单的数学运算类

(2) 实现一个矩形类 Rectangle，创建其实例并使用，要求如下。

- 字段成员为：Length，Width；且可供实例访问。
- 3 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。

分析：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;

    //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;
    }
    //接受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;
            }
        }
    }

    // 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;
    }
}
```

演示调用代码如下：

```
static void Main(string[] args)
{
    int iLength = 0, iWidth = 0, iPerimeter, iArea;
```

```
        Console.Write("请输入矩形的长:");
        iLength = Convert.ToInt32(Console.ReadLine());

        Console.Write("请输入矩形的宽:");
        iWidth = Convert.ToInt32(Console.ReadLine());

        Rectangle rect = new Rectangle(iLength, iWidth);
        if (rect.IsSquare() == true)
            Console.WriteLine("是正方形");

        iPerimeter = rect.Perimeter();
        Console.WriteLine("矩形的周长为:" + iPerimeter);
        iArea = rect.Area();
        Console.WriteLine("矩形的面积为:" + iArea);

        Console.Write("请输入指定矩形的面积:");
        iArea = Convert.ToInt32(Console.ReadLine());

        Rectangle rect2 = new Rectangle(iArea);
        iLength = rect2.Length;
        iWidth = rect2.Width;
        Console.Write("分解后得到的长和宽分别为:" + iLength + "和" + iWidth);

        Console.ReadLine();
    }
```

程序运行结果如图 4-2 所示。

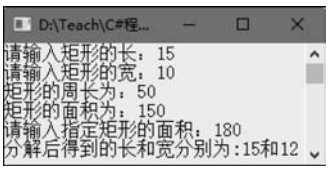

图 4-2　矩形类演示

(3) 编写一个类 Cal,要求如下。

- 此类具有 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 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。

分析：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。以下将分别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;
        }
    }
```

以下是 CalOprs 类，该类根据传入运算符，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：

```
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 Enter 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。

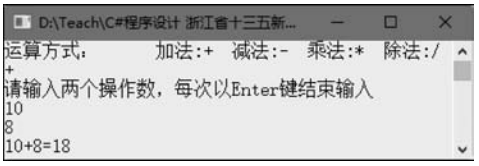

图 4-3　多态、继承等综合演示

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。

分析：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;
    }
}
```

事件委托及事件触发类代码如下：

```
public delegate void MyDelegate(object sender,EventArgs e);          //事件委托

//触发事件类
public class EventTrigger
{
    public event MyDelegate MyDelegateEvent;
    public void Fire(int max)
    {
        //按照指定的次数来触发事件,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);
            }
        }
    }
}
```

事件接收类代码如下：

```
// 事件接收类
public class EventSubscription
{
    public void SubscriptionEvent()
    {
        EventTrigger et = new EventTrigger();
        et.MyDelegateEvent += new MyDelegate(eventTrig_MyDelegateEvent);     //订阅
        et.Fire(10);                                                         //触发 10 次
    }
    //从 DataEventArgs 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：

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

程序执行效果如图 4-4 所示。

图 4-4　定时触发的事件