图 4.4 Triangle proof 输出结果

【解题思路】

定义静态字符串变量、静态整型变量和非静态变量，使用直接调用静态变量和定义类的对象调用非静态变量，输出两者的调用结果。

【实现步骤】

（1）创建一个空白解决方案“Triangle proof”。

（2）在解决资源方案管理器单击右键，选择【添加】→【新建项目】→【控制台应用程序】，取名为 Triangle proof。

（3）在解决方案资源管理器中右击引用，在弹出的快捷菜单中选择【添加引用】。

（4）选择.net 选项卡，拖动滚动条找到 System.Windows.Forms，单击确定。

```
    using System.Windows.Forms；
public class TrigonProof
    {
        public static string sideA = "三角形"；
        public static int sideB = 2；
        public string sideC = "三角形边"；
    }
    public class Program
    {
        static void Main( )
```

```
        {
            MessageBox.Show(TrigonProof.sideA)；//静态变量不用实例化便可以直接拿来使用
           //MessageBox.Show(TrigonProof.sideC)；而非静态变量则需要把类实例化才能使用）
MessageBox.Show(TrigonProof.sideB.ToString())；//以上相同的
            TrigonProof ex = new TrigonProof ()；
            MessageBox.Show(ex.sideC)；// 这就是调用非静态变量的方法
        }
 }
```

（5）在解决方案资源管理器中，右击 Triangle proof 项目，并在弹出的快捷菜单中选择【属性】→【输出类型】，选择 Windows 应用程序

（6）按“F5”键运行该项目，即可得到如图 4.4 所示的输出结果。

【工程师提示】

这个实例主要说明静态变量的使用，静态变量可以直接调用，非静态变量不能被直接调用。静态变量调用方式是类名.静态变量名，非静态变量则是对象.非静态变量名。

2. 非静态变量

非静态变量是不带“static”修饰符声明变量。非静态变量一定是建立在所属类型的对象后才开始存在于内存中。如果变量被定义在类中，那么当对象被建立时，变量随之诞生；对象消失，变量也随之消失。如果定义在结构中，那么结构存在多久，变量也存在多久。

3. 局部变量

局部变量是指存储在独立的程序块中，如一个 for 语句、switch 语句或者是方法中声明的变量，它只在该范围中有效，离开该范围，则变量失效。

```
for（int i=0；i<4；i++）
{
MessageBox.Show（i.ToString()）；//此时变量 i 在循环中是有效的
```

```
}
MessageBox.Show ( i.ToString() ; //此时变量 i 已经失效了
```

【课堂练习】

（1）下面程序段哪些地方出错了？

```
class Test
{
    public int x = 10 ;
    public static int y = 20 ;
}
class Program
{
    static void Main(string[] args)
    {
        Test t = new Test() ;
        Console.WriteLine(t.x) ;
        Console.WriteLine(t.y) ;
    }
}
```

（2）下面程序段的输出结果是什么？

```
class Test
{
    private static int x = 0 ;
    public void Fun()
    {
        Console.WriteLine("x={0}" ,  x = x + 10) ;
    }
}
class Program
{
    static void Main(string[] args)
    {
        Test t = new Test() ;
        t.Fun() ;
        t.Fun() ;
    }
```

```
}
```

4.2.2　成员访问控制符

成员访问控制符可以定义方法、变量、类，但不同的成员访问控制符有不同的访问级别。

1. 公有成员（public）

公有成员提供了类的外部界面，允许类使用者从外部进行访问，公有成员的修饰符“public”，这是限制最少的一种方式。

2. 私有成员（private）

私有成员仅限于在该类中访问，不允许类以外的使用者访问，私有成员的修饰符“private”。

3. 保护成员（protected）

保护成员既在类中可以访问，派生类也可以对其访问，保护成员的修饰符“protected”。

4. 内部成员（internal）

使用“internal”修饰符修饰的成员是一种特殊成员。这种成员对于同一包中的应用程序或库是透明的，而在包“.NET”之外是禁止访问的。

【实例 4-3】　实例使用成员访问控制符。其运行效果如图 4.5 所示。

图 4.5　People 输出结果

【解题思路】

创建两个类分别取名 People 和 Program，分别对两个类写入不同的修饰符的方法、变量，然后将其实例化，看看输出的结果。

【实现步骤】

（1）创建一个空白解决方案“People”。

（2）在解决资源方案管理器单击右键，选择【添加】→【新建项目】→【控制台应用程序】，取名为 People。

（3）在解决方案资源管理器中右击引用，在弹出的快捷菜单中选择【添加引用】。

（4）选择.net 选项卡，拖动滚动条找到 System.Windows.Forms，单击确定。

```
using System.Windows.Forms;
public class People
                {
                    protected string name;     //保护成员
    private int age;              // 私有变量
        public double height;     // 公有变量
        public void PeopleInfo(double myheight,int myage, string myname)//公共方法
        {
            height = myheight;
            age = myage;
            name = myname;
       MessageBox.Show("姓名："+name+"\n身高："+height.ToString()+"\n年龄：
    "+age.ToString());//调用//一个公共的方法
        }
    }
    public class Tom:People
    {
        public void TomInfo()
        {
```

```
            People ex = new People ();
            height = 1.75;
            // age = 23;错误的，不可以访问私有变量
            name = "小红";//正确的，可以访问保护变量
            MessageBox.Show("姓名：" + name + "\n身高：" +height.ToString());
        }
    }
    public class Program
    {
        static void Main( )
        {
            People ex = new People ();
            ex.PeopleInfo (1.80,25,"阿明");//调用公共方法Test
            MessageBox.Show(ex.height.ToString());//输出公共变量
            //MessageBox.Show(ex. age.ToString());这样调用是错误的，整型变量age是属于
             Example类//的私有变量，则不能被调用
            ///MessageBox.Show(ex.name.ToString());作为保护变量Name，只能在派生类中使用
            Tom t = new Tom ();
            t.TomInfo ();
        }
    }
```

（5）在解决方案资源管理器中，右击 People 项目，并在弹出的快捷菜单中选择【属性】→【输出类型】，选择 Windows 应用程序。

（6）按“F5”键运行该项目，即可得到如图 4.5 所示的输出结果。

【工程师提示】

实例 4-3 讲解的是修饰符的使用，public、private、protected 各种访问级别，最常用的是 public 和 private，而在网页中 protected 使用比较多。

4.2.3　域或属性

1. 域

域是类和对象的属性，它可以是基本数据类型的变量，也可以是其他类的对象。若将类中的某个域声明为 static，那该域称为静态域，否则为非静态域。一般来说，静态成员是属于类所有的，非静态成员则属于类的实例——对象。

【实例 4-4】 了解如何使用静态域和非静态域。其运行效果如图 4.6 所示。

图 4.6 Media 输出结果

【解题思路】

定义一个 Media 类，定义一个静态变量和非静态变量，对象访问其非静态变量，可不建对象而直接访问静态变量，分别输出访问的有关属性。

【实现步骤】

（1）创建一个空白解决方案“Media”。

（2）在解决资源方案管理器单击右键，选择【添加】→【新建项目】→【控制台应用程序】，取名为 Media。

（3）在解决方案资源管理器中右击引用，在弹出的快捷菜单中选择【添加引用】。

（4）选择.net 选项卡，拖动滚动条找到 System.Windows.Forms，单击确定。

```
using System.Windows.Forms;
```

```
public class Media
    {
        public static string mediaName;//定义一个静态变量
        public string news;                    //定义一个非静态变量
        public void Test()
        {
            mediaName = "新华社";//正确，访问静态变量
            news = "2012年伦敦奥运会于7月28日开幕";//正确，访问非静态变量
        }
        public static void Test2()
        {
            mediaName = " The Associated Press ";
        //静态方法只能访问本类静态变量
            //news = " British royal family welcomes Olympic Games";
            MessageBox.Show(mediaName);
        }
    }
    public class Program
    {
        static void Main( )
        {
            Media me = new Media ();
            me.news = " Bo's wife charged with homicide ";
            Media. mediaName = " The Associated Press ";//正确，类可不建对象而直接访问静态变量
            MessageBox.Show(Media. mediaName +"\n"+ me.news);
            me.Test();//正确，对象可以调用非静态方法
            MessageBox.Show(Media.mediaName +"\n"+ me.news);
            Media.Test2();//正确,类可以调用静态方法
        }
    }
```

（5）在解决方案资源管理器中右击 Media 项目，并在弹出的快捷菜单中选择【属性】→【输出类型】，选择 Windows 应用程序。

（6）按“F5”键运行该项目，即可得到如图 4.6 所示的输出结果。

【工程师提示】

都知道域是类和对象的属性，域中还包含静态域和非静态域，那么就有静态变量、方法和非静态变量、方法，而静态的方法特殊一些，静态方法中只能调用静态变量值；非静态方法都可以调用。调用静态方法的方式为：对象名.方法名( )，调用非静态方法必须将方法所在的那个类实例化成对象调用：对象名.方法名( )。

2. 属　性

属性在 C#中充分的体现出了封装性 :不直接操作类的数据内容 ,通过访问器进行访问。

属性的访问声明有三种方式：

(1) 只有 Set 访问器，属性的值只能写入不能读出，Set 访问器通过 value 来写入值。

(2) 只有 Get 访问器 , 属性的值只能读不能写 , Get 访问器是通过 Return 来读取属性值。

(3) 同时具有 Get 和 Set 访问器，属性的值既能读也能写。

【实例 4-5】　输出一个员工的基本信息，输出他的姓名、性别、电话号码和通信地址，须用属性来确定哪些信息只读、只写或者是可读可写。其运行效果如图 4.7 所示。

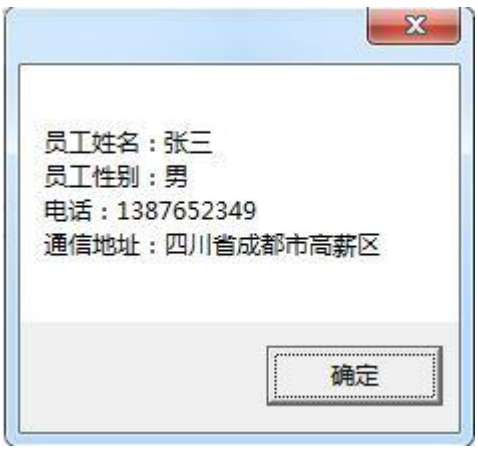

图 4.7　Personnel 输出结果

【解题思路】

一个员工的姓名、性别是不可修改的，这两个属性就只能是只读，而电话和通信地址是可以修改的，那么属性则是可读可写的。

【实现步骤】

（1）创建一个空白解决方案“Personnel”。

（2）在解决资源方案管理器单击右键，选择【添加】→【新建项目】→【控制台应用程序】，取名为 Personnel。

（3）在解决方案资源管理器中右击引用，在弹出的快捷菜单中选择【添加引用】。

（4）选择.net 选项卡，拖动滚动条找到 System.Windows.Forms，单击确定。

```
using System.Windows.Forms;
public class Personnel
    {
        private string name="张三";
        private string sex = "男";
        private string phone;
        private string address;
        public string Name          //只读属性
        {
            get { return name; }
        }
        public string Sex
        {
            get { return sex; }
        }
        public string Phone       //可读可写属性
        {
            get { return phone; }
            set { phone = value; }
        }
```

```
        public string Address
        {
            get { return address; }
            set { address = value; }
        }
        static void Main( )
        {
            Personnel ex = new Personnel ();
            ex.Phone = "1387652349";
            ex.Address = "四川省成都市高薪区";
            //ex.Sex = "女";不可写，这是一个只读属性
            MessageBox.Show("员工姓名：" + ex.Name + "\n" + "员工性别：" + ex.Sex + "\n" +
            "电话：" + ex.Phone + "\n" + "通信地址：" + ex.Address);
        }
    }
```

（5）在解决方案资源管理器中右击 Personnel 项目，并在弹出的快捷菜单中选择【属性】→【输出类型】，选择 Windows 应用程序。

（6）按“F5”键运行该项目，即可得到如图 4.7 所示的输出结果。

4.2.4 方法（method）

1. 方法的定义

方法在面向对象程序语言设计中经常都会用到，对类的数据成员操作都封装在类的成员方法中。而方法的声明主要包括修饰符、返回值数据类型、方法名、入口参数和方法体。主要的方法修饰符 :new、public、protected、internal、private、static、virtual、sealed、override、abstract、extern 。在学习中通常都会用到 public 和 private 修饰符。

方法的返回值类型必须是合法的 C#数据类型，并且是与方法返回值相同的数据类型，在方法体里，用“return”得到返回值。如果没有返回值，则声明时用关键字“void”, 并且方法

体里中不能使用“return”来返回数值。比如：

```
Public int A ( )
{
    int b=1，a=1;
    return a+b;          //return 返回数据值
}
Public void B ( )
{
    int b=1，a=1;
    a=a+b;               //没有 return 返回值
}
```

C＃的类定义的方法有两种：静态和非静态。使用了 static 修饰符的方法为静态方法，否则为非静态。静态方法是一种特殊的成员方法，像静态变量一样，它不属于类的某一个具体的实例。非静态方法可以访问类中的任何成员，而静态方法只能访问类的静态成员。

【实例 4-6】　简单的方法创建与使用。具体运行效果如图 4.8 所示。

图 4.8　People 输出结果

【解题思路】

在 People 这个类中，声明一个方法叫 PeopleInfo 方法，里面包含定义的变量、要输出的结果、返回的类型值。