实验 5　类的设计和实现

实 验 目 的

- 掌握类的声明和对象的创建；
- 掌握对象的封装性；
- 掌握类的访问修饰符；
- 掌握类的继承的实现；
- 掌握派生类、抽象类、抽象方法的使用；
- 掌握类的多态性；
- 掌握运算符的重载；
- 了解接口的实现；
- 了解事件的实现。

实 验 内 容

实验 5-1　创建类 MyMath，计算圆的周长、面积和球的体积

实验要求：创建类 MyMath，包含常量 PI；静态方法 Perimeter（周长）、Area（面积）、Volume（体积）。运行效果如图 5-1 所示。

图 5-1　实验 5-1 运行效果

操作提示：程序代码如图 5-2 所示。

实验 5-2　创建表示摄氏温度的类 TemperatureCelsius

实验要求：创建类 TemperatureCelsius，包含实例字段 degree（表示摄氏温度）和实例方法 ToFahrenheit（将摄氏温度转换为华氏温度）。运行效果如图 5-3 所示。

操作提示：程序代码如图 5-4 所示。

实验 5-3　类的继承的实现

实验要求：创建基类 Person 和派生类 Teacher。基类 Person 包含实例字段 name 和

```
sy5_1.MyMath                                  Volume(double r)
using System;
namespace sy5_1
{
    class MyMath
    {
        public const double PI = 3.1415926;  //常量
        public static double Perimeter(double r) //静态方法
        {
            double p = 2 * PI * r;
            return p;
        }
        public static double Area(double r) //静态方法
        {
            double a = PI * r * r;
            return a;
        }
        public static double Volume(double r) //静态方法
        {
            double v = 4 * PI * Math.Pow(r, 3) / 3;
            return v;
        }
    }
    class Program
    {
        static void Main(string[] args)
        {
            Console.Write("请输入半径： ");
            double r = double.Parse(Console.ReadLine());
            Console.WriteLine("圆的周长 = {0}", MyMath.Perimeter(r));
            Console.WriteLine("圆的面积 = {0}", MyMath.Area(r));
            Console.WriteLine("球的体积 = {0}", MyMath.Volume(r));
            Console.ReadKey();
        }
    }
}
```

图 5-2　实验 5-1 程序代码

```
请输入摄氏温度：15
摄氏温度 = 15，华氏温度 = 59
```

图 5-3　实验 5-2 运行效果

```
sy5_2.Program                                 Main(string[] args)
using System;
namespace sy5_2
{
    public class TemperatureCelsius
    {
        protected double degree; //数据：摄氏温度
        public TemperatureCelsius(double d) //构造函数
        {
            this.degree = d;
        }
        public double ToFahrenheit() //实例方法
        {    // 摄氏温度到华氏温度的转换
            return (degree * 9 / 5) + 32;
        }
    }
    class Program
    {
        static void Main(string[] args)
        {
            Console.Write("请输入摄氏温度： ");
            double d = Double.Parse(Console.ReadLine());
            TemperatureCelsius celsius = new TemperatureCelsius(d);
            Console.WriteLine("摄氏温度 = {0}，华氏温度 = {1} ", d, celsius.ToFahrenheit());
            Console.ReadKey();
        }
    }
}
```

图 5-4　实验 5-2 程序代码

age；虚函数 GetInfo()显示个人信息。派生类 Teacher 除了包含基类的 name 和 age 字段，还包含自己的 TeacherID 字段，并使用关键字 override 来重写方法 GetInfo()。运行效果如图 5-5 所示。

```
Name: Mr. YU
Age : 40
teacherID: 1990108001
```

图 5-5　实验 5-3 运行效果

操作提示：程序代码如图 5-6 所示。

```
sy5_3.Person                                    GetInfo()
using System;
namespace sy5_3
{
    public class Person  //基类 等同于public class Person:Object
    {
        public string name;
        public uint age;
        public Person(string name, uint age) //基类的构造函数
        {
            this.name = name; //this关键字引用类的当前实例
            this.age = age;  //this关键字引用类的当前实例
        }
        public virtual void GetInfo()
        {
            Console.WriteLine("Name: {0}", name);
            Console.WriteLine("Age : {0}", age);
        }
    }
    public class Teacher : Person  //派生类
    {
        public string teacherID;
        //派生类构造函数并用 ":base" 调用基类构造函数
        public Teacher(string name, uint age, string id):base(name, age)
        {
            this.teacherID = id;
        }
        public override void GetInfo()
        {  //调用基类的方法
            base.GetInfo();
            Console.WriteLine("teacherID: {0}", teacherID);
        }
    }
    public class TestPersonTeacher
    {
        static void Main(string[] args)
        {
            //构造
            Teacher objteacher = new Teacher("Mr. YU", 40, "1990108001");
            objteacher.GetInfo();
            Console.ReadLine();
        }
    }
}
```

图 5-6　实验 5-3 程序代码

实验 5-4　抽象类、抽象方法、多态性的实现

实验要求：创建抽象基类 Shape 和派生类 Rectangle、Circle。利用多态性实现 Area(计算面积)和 Show(显示图形名称和面积)抽象方法。运行效果如图 5-7 所示。

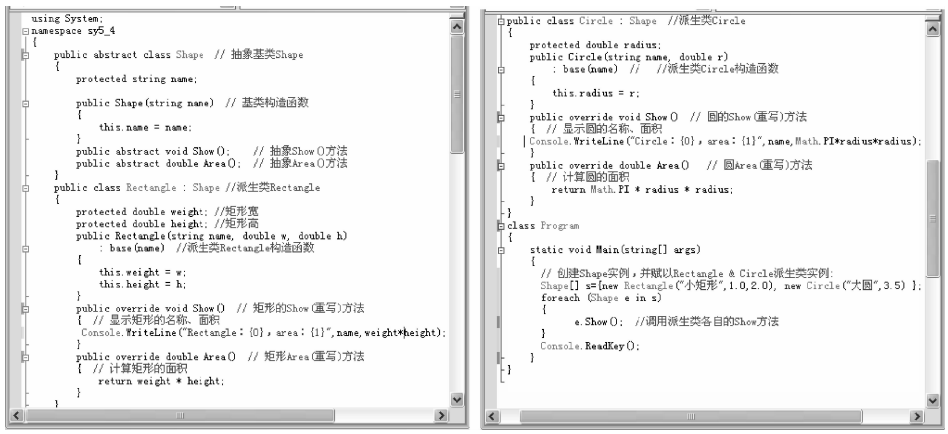
```
Rectangle: 小矩形, area: 2
Circle: 大圆, area: 38.4845100006475
```

图 5-7　实验 5-4 运行效果

操作提示：程序代码如图 5-8 所示。

实验 5-5　运算符重载

实验要求：参照课本例 7.24，使用运算符重载创建定义复数相加、相减和相乘的复数类 Complex。运行效果如图 5-9 所示。

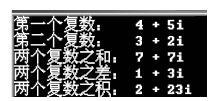


```
using System;
namespace sy5_4
{
    public abstract class Shape  // 抽象基类Shape
    {
        protected string name;

        public Shape(string name)  // 基类构造函数
        {
            this.name = name;
        }
        public abstract void Show();    // 抽象Show()方法
        public abstract double Area();  // 抽象Area()方法
    }
    public class Rectangle : Shape //派生类Rectangle
    {
        protected double weight; //矩形宽
        protected double height; //矩形高
        public Rectangle(string name, double w, double h)
            : base(name)  //派生类Rectangle构造函数
        {
            this.weight = w;
            this.height = h;
        }
        public override void Show()  // 矩形的Show(重写)方法
        {  // 显示矩形的名称、面积
            Console.WriteLine("Rectangle：{0}，area：{1}",name,weight*height);
        }
        public override double Area()  // 矩形Area(重写)方法
        {  // 计算矩形的面积
            return weight * height;
        }
    }
```

```
public class Circle : Shape  //派生类Circle
{
    protected double radius;
    public Circle(string name, double r)
        : base(name)  //   //派生类Circle构造函数
    {
        this.radius = r;
    }
    public override void Show()  // 圆的Show(重写)方法
    {  // 显示圆的名称、面积
        Console.WriteLine("Circle：{0}，area：{1}",name,Math.PI*radius*radius);
    }
    public override double Area()   // 圆Area(重写)方法
    {  // 计算圆的面积
        return Math.PI * radius * radius;
    }
}
class Program
{
    static void Main(string[] args)
    {
      // 创建Shape实例，并赋以Rectangle & Circle派生类实例:
      Shape[] s={new Rectangle("小矩形",1.0,2.0), new Circle("大圆",3.5) };
      foreach (Shape e in s)
      {
          e.Show();  //调用派生类各自的Show方法
      }
      Console.ReadKey();
    }
}
```

图 5-8　实验 5-4 程序代码



```
第一个复数：    4 + 5i
第二个复数：    3 + 2i
两个复数之和：  7 + 7i
两个复数之差：  1 + 3i
两个复数之积：  2 + 23i
```

图 5-9　实验 5-5 运行效果

操作提示：程序代码如图 5-10 所示。

```
public struct Complex       // 复数
{
    public int real;        // 实部
    public int imaginary;   //  虚部
    public Complex(int real, int imaginary)  //构造函数
    {
        this.real = real;
        this.imaginary = imaginary;
    }
    // 重载运算符(+)
    public static Complex operator +(Complex c1, Complex c2)
    {
        return new Complex(c1.real + c2.real, c1.imaginary + c2.imaginary);
    }
    // 重载运算符(-)
    public static Complex operator -(Complex c1, Complex c2)
    {
        return new Complex(c1.real - c2.real, c1.imaginary - c2.imaginary);
    }
    // 重载运算符(*)
    public static Complex operator *(Complex c1, Complex c2)
    {
     return new Complex(c1.real*c2.real-c1.imaginary*c2.imaginary,c1.real*c2.imaginary+c1.imaginary*c2.real);
    }
    //重载ToString方法以显示复数的实部和虚部
    public override string ToString()
    {
        return (String.Format("{0} + {1}i", real, imaginary));
    }
}
class TestComplex
{
    static void Main()
    {
        Complex num1 = new Complex(4, 5);
        Complex num2 = new Complex(3, 2);
        Complex sum = num1 + num2;//使用重载运算符(+)
        Complex sub = num1 - num2;//使用重载运算符(-)
        Complex mul = num1 * num2;//使用重载运算符(*)
        //调用重载的ToString方法
        Console.WriteLine("第一个复数：   {0}", num1);
        Console.WriteLine("第二个复数：   {0}", num2);
        Console.WriteLine("两个复数之和：  {0}", sum);
        Console.WriteLine("两个复数之差：  {0}", sub);
        Console.WriteLine("两个复数之积：  {0}", mul);
        Console.ReadKey();
    }
}
```

图 5-10　实验 5-5 程序代码

实验 5-6　接口的实现

实验要求：声明一个接口 ICDPlayer，包含 Play()、Stop()、NextTrack() 和 PreviousTrack()四个接口方法，以及一个只读属性 CurrentTrack。创建类 CDPlayer 实现该接口，模拟 CD 的播放、停止、下一音轨、上一音轨的操作。运行效果如图 5-11 所示。

图 5-11　实验 5-6 运行效果

操作提示：程序代码如图 5-12 所示。

```
public interface ICDPlayer
{
    void Play();  //方法签名
    void Stop();  //方法签名
    void PreviousTrack(); //方法签名
    void NextTrack();     //方法签名
    int CurrentTrack  // 只读属性
    {
        get;
    }
}
public class CDPlayer : ICDPlayer
{
    private int currentTrack = 0;
    // 实现接口中定义的方法
    public void Play()
    {
        Console.WriteLine("启动CD...");
    }
    public void Stop()
    {
        Console.WriteLine("停止CD...");
    }
    public void PreviousTrack()
    {
        Console.WriteLine("前一个音轨...");
        if (currentTrack >= 1) currentTrack--;
    }
    public void NextTrack()
    {
        Console.WriteLine("后一个音轨...");
        currentTrack++;
    }
    public int CurrentTrack  // 只读属性
    {
       get
        {
            return currentTrack;
        }
    }
}
class TestCDPlayer
{
    static void Main(string[] args)
    {
        CDPlayer myCD = new CDPlayer();
        myCD.Play();
        Console.WriteLine("myCD.CurrentTrack={0}", myCD.CurrentTrack);
        myCD.NextTrack();
        myCD.NextTrack();
        Console.WriteLine("myCD.CurrentTrack={0}", myCD.CurrentTrack);
        ICDPlayer myICD = (ICDPlayer)myCD;
        myICD.PreviousTrack();
        Console.WriteLine("myICD.CurrentTrack={0}", myICD.CurrentTrack);
        myICD.Stop();
        Console.ReadKey();
    }
}
```

图 5-12　实验 5-6 程序代码

实验 5-7　事件的实现

实验要求：参照课本例 9.9 的综合示例，实现事件处理。运行效果如图 5-13 所示。

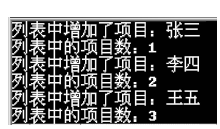
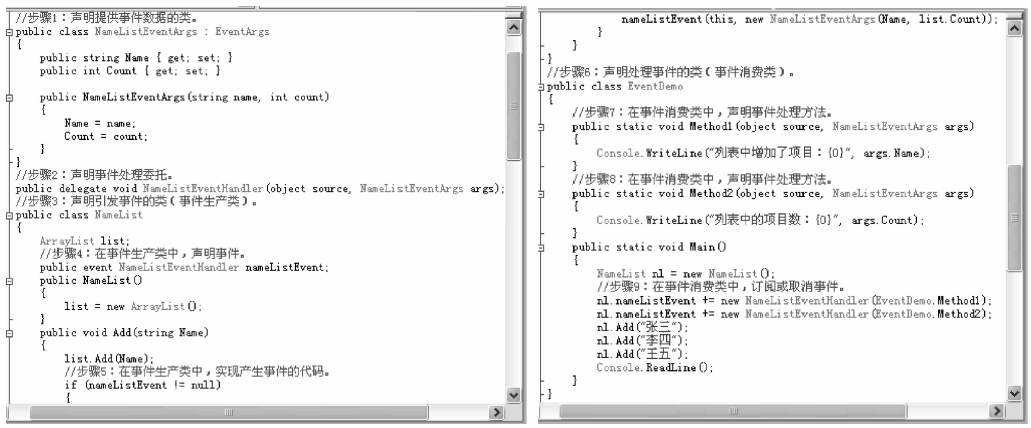


图 5-13　实验 5-7 运行效果

操作提示：程序代码如图 5-14 所示。



```
//步骤1：声明提供事件数据的类。
public class NameListEventArgs : EventArgs
{
    public string Name { get; set; }
    public int Count { get; set; }

    public NameListEventArgs(string name, int count)
    {
        Name = name;
        Count = count;
    }
}
//步骤2：声明事件处理委托。
public delegate void NameListEventHandler(object source, NameListEventArgs args);
//步骤3：声明引发事件的类（事件生产类）。
public class NameList
{
    ArrayList list;
    //步骤4：在事件生产类中，声明事件。
    public event NameListEventHandler nameListEvent;
    public NameList()
    {
        list = new ArrayList();
    }
    public void Add(string Name)
    {
        list.Add(Name);
        //步骤5：在事件生产类中，实现产生事件的代码。
        if (nameListEvent != null)
        {
            nameListEvent(this, new NameListEventArgs(Name, list.Count));
        }
    }
}
//步骤6：声明处理事件的类（事件消费类）。
public class EventDemo
{
    //步骤7：在事件消费类中，声明事件处理方法。
    public static void Method1(object source, NameListEventArgs args)
    {
        Console.WriteLine("列表中增加了项目：{0}", args.Name);
    }
    //步骤8：在事件消费类中，声明事件处理方法。
    public static void Method2(object source, NameListEventArgs args)
    {
        Console.WriteLine("列表中的项目数：{0}", args.Count);
    }
    public static void Main()
    {
        NameList nl = new NameList();
        //步骤9：在事件消费类中，订阅或取消事件。
        nl.nameListEvent += new NameListEventHandler(EventDemo.Method1);
        nl.nameListEvent += new NameListEventHandler(EventDemo.Method2);
        nl.Add("张三");
        nl.Add("李四");
        nl.Add("王五");
        Console.ReadLine();
    }
}
```

图 5-14　实验 5-7 程序代码

实验 6　结构和枚举

实 验 目 的

- 掌握结构的声明和调用；
- 掌握枚举的声明和使用；
- 了解系统提供的枚举类型 System. ConsoleColor 的使用；
- 了解系统提供的 System. Enum 类的使用。

实 验 内 容

实验 6-1　创建并使用日期结构体

实验要求：定义一个日期结构体 MyDate，包含字段 year、month、day，以及方法 DisplayWeek()和 DisplayDate()。输入年、月、日，显示日期(年/月/日)及日期所对应的星期。运行效果如图 6-1 所示。

图 6-1　实验 6-1 运行效果

操作提示：程序代码如图 6-2 所示。

```
public struct MyDate
{
    public int year;  //年
    public int month; //月
    public int day;   //日
    public MyDate(int y, int m, int d)//构造函数
    {
        year = y;
        month = m;
        day = d;
    }
    public void DisplayWeek()
    { // 显示所表示的日期是星期几
        Console.WriteLine("星期为：{0}", new DateTime(year, month, day).DayOfWeek);
    }
    public void DisplayDate()
    { // 显示日期(年/月/日)
        Console.WriteLine("日期为：{0}/{1}/{2}", year, month, day);
    }
}
class Program
{
    static void Main(string[] args)
    {
        Console.Write("请输入年： ");
        int y = int.Parse(Console.ReadLine());
        Console.Write("请输入月： ");
        int m = int.Parse(Console.ReadLine());
        Console.Write("请输入日： ");
        int d = int.Parse(Console.ReadLine());
        MyDate d1 = new MyDate(y, m, d);
        d1.DisplayDate();
        d1.DisplayWeek();
        Console.ReadKey();
    }
}
```

图 6-2　实验 6-1 程序代码

实验 6-2　学生成绩统计

实验要求：定义一个学生分数结构体 StudentGrade，包含字段 Name、Score，以及带两个参数的构造函数。利用结构体 StudentGrade，创建结构数组变量，存放若干学生的姓名和分数信息，计算平均分。运行效果如图 6-3 所示。

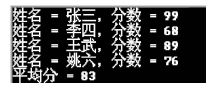


图 6-3　实验 6-2 运行效果

操作提示：程序代码如图 6-4 所示。

```
using System;
namespace sy6_2
{
    public struct StudentGrade
    {
        public string Name; //姓名
        public float Score; //分数
        public StudentGrade(string n, float s) //构造函数
        {
            Name = n;
            Score = s;
        }
    }
    class Program
    {
        static void Main(string[] args)
        {   // StudentGrade数组
            StudentGrade[] grades = { new StudentGrade("张三", 99), new StudentGrade("李四", 68),
                                      new StudentGrade("王武", 89), new StudentGrade("姚六", 76) };
            float sum = 0; //分数总和
            foreach (StudentGrade sg in grades)
            {
                Console.WriteLine("姓名 = {0}，分数 = {1}", sg.Name, sg.Score);
                sum += sg.Score;  //分数合计
            }
            Console.WriteLine("平均分 = {0}", sum / grades.Length);//平均分
            Console.ReadKey();
        }
    }
}
```

图 6-4　实验 6-2 程序代码

实验 6-3　计算 3 个坐标点构成的三角形的面积

实验要求：声明一个表示平面坐标系中的点的结构体 CoOrds，包含字段 x，y，以及带两个参数的构造函数。分别利用三种不同的方法（默认构造函数、有两个参数的构造函数、先声明再赋值），创建 3 个平面坐标点变量。求这 3 个坐标点所构成的三角形的面积。注意判断 3 个坐标点是否可以构成三角形。运行效果如图 6-5 所示。

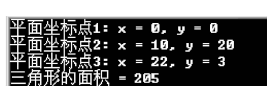


图 6-5　实验 6-3 运行效果

操作提示：

（1）判断 3 个坐标点是否可以构成三角形，先计算这 3 个坐标点所构成的 3 条边的长，如果任意两边之和大于第三边，则这 3 个坐标点可以构成三角形。

（2）程序代码如图 6-6 所示。

实验 6-4　使用系统提供的枚举类型 System.ConsoleColor

实验要求：使用系统提供的枚举类型 System.ConsoleColor，设置控制台的输出颜色（前景色、背景色）。运行效果如图 6-7 所示。

操作提示：

（1）ConsoleColor 枚举用于指定控制台的前景色和背景色。其取值如表 6-1 所示。

```
using System;
namespace sy6_3
{
    public struct CoOrds   // 平面坐标点
    {
        public int x, y;
        public CoOrds(int p1, int p2)  // 有2个参数的构造函数
        {
            x = p1;
            y = p2;
        }
    }
    class Program
    {
        static void Main(string[] args)
        {
            CoOrds coords1 = new CoOrds(); // 调用默认构造函数
            Console.WriteLine("平面坐标点1: x = {0}, y = {1}", coords1.x, coords1.y);
            CoOrds coords2 = new CoOrds(10, 20);  // 调用有2个参数的构造函数
            Console.WriteLine("平面坐标点2: x = {0}, y = {1}", coords2.x, coords2.y);
            CoOrds coords3;
            coords3.x = 22;
            coords3.y = 3;
            Console.WriteLine("平面坐标点3: x = {0}, y = {1}", coords3.x, coords3.y);
            double side1 = Math.Sqrt(Math.Pow(coords1.x - coords2.x, 2) + Math.Pow(coords1.y - coords2.y, 2));
            double side2 = Math.Sqrt(Math.Pow(coords1.x - coords3.x, 2) + Math.Pow(coords1.y - coords3.y, 2));
            double side3 = Math.Sqrt(Math.Pow(coords3.x - coords2.x, 2) + Math.Pow(coords3.y - coords2.y, 2));
            if (side1 + side2 > side3 && side1 + side3 > side2 && side3 + side2 > side1)
            {
                double h = (side1 + side2 + side3) / 2;
                double area = Math.Sqrt(h * (h - side1) * (h - side2) * (h - side3));
                Console.WriteLine("三角形的面积 = {0}", area);
            }
            else Console.WriteLine("不能构成三角形！");
            Console.ReadKey();
        }
    }
}
```

图 6-6　实验 6-3 程序代码

图 6-7　实验 6-4 运行效果

(2) Console.ForegroundColor 属性用于获取或设置控制台的前景色；也就是显示的每个字符的颜色。默认为灰色。

表 6-1　ConsoleColor 枚举成员

名　称	说　明	名　称	说　明
Black	黑色	Gray	灰色
DarkBlue	藏蓝色	DarkGray	深灰色
DarkGreen	深绿色	Blue	蓝色
DarkCyan	深紫色(深蓝绿色)	Green	绿色
DarkRed	深红色	Cyan	青色(蓝绿色)
DarkMagenta	深紫红色	Red	红色
DarkYellow	深黄色(赭色)	Magenta	紫红色

(3) Console.BackgroundColor 属性用于获取或设置控制台的背景色；也就是显示在每个字符后面的颜色。默认为黑色。

(4) Console.ResetColor 方法将控制台的前景色和背景色设置为默认值。

(5) 程序代码如图 6-8 所示。

```
using System;
namespace sy6_4
{
    class Program
    {
        static void Main(string[] args)
        {  // 指定控制台的前景色；也就是显示的每个字符的颜色。默认为灰色
            Console.ForegroundColor = ConsoleColor.Cyan;
            Console.WriteLine("前景色为Cyan");
            // 将控制台的前景色和背景色设置为默认值
            Console.ResetColor();
            // 指定控制台的背景色；也就是显示在每个字符后面的颜色。默认为黑色
            Console.BackgroundColor = ConsoleColor.Cyan;
            // 重新指定控制台的前景色
            Console.ForegroundColor = ConsoleColor.Red;
            Console.WriteLine("前景色为Red，背景色为Cyan");
            Console.ReadKey();
        }
    }
}
```

图 6-8　实验 6-4 程序代码

实验 6-5　定义和使用星期枚举类型

实验要求：定义一个表示星期的枚举类型。输入 1～7 中的任一数字，打印其所对应的星期；如果输入的数字不在 1～7 之间，则报错“输入数字范围不对!”；如果输入非数字字符，则利用异常处理捕捉异常信息。运行效果如图 6-9 所示。

(a) 数字对应的星期　　(b) 数字范围错误　　(c) 非数字字符错误

图 6-9　实验 6-5 运行效果

操作提示：程序代码如图 6-10 所示。

```
using System;
namespace sy6_5
{
    class Program
    {
        enum Days { Monday = 1, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday };
        static void Main(string[] args)
        {
            Console.Write("请输入一个数字（1~7）： ");
            try
            {
                int d = int.Parse(Console.ReadLine());
                if (d > 7 || d < 1) Console.WriteLine("输入数字范围不对!");
                else Console.WriteLine("{0} 对应于 {1}", d, (Days)d);
            }
            catch (FormatException e1)
            {
                Console.WriteLine(e1.Message);
            }
            Console.ReadKey();
        }
    }
}
```

图 6-10　实验 6-5 程序代码

实验 6-6　使用系统提供的 System.Enum 类

实验要求：参照课本例 10.5 枚举综合示例，定义一个星期枚举类型和一个颜色枚举类型，使用系统提供的 System.Enum 类对枚举进行基本操作，包括访问枚举成员的名称和