Lab6 C#基础

学习目标：

1、掌握接口的定义方法

2、掌握结构与枚举的定义方法

一、接口

接口是指描述可属于任何类或结构的一组相关功能。接口可由属性、方法、事件、索引器任意组合构成，但接口不能包含字段。接口不能单独存在，不能实例化，只能在实现接口的类中实现。

接口与类的区别：比喻来说，接口类似做指示的高层领导，而类是一线工作的人。因为接口从来做具体的实现，只是指导性的东西(例如方法的声明等，即定义出基本的功能框架)。而类一旦接受某个接口的领导（即实现某个接口），那么就得遵循指导性方案，把领导说过的每一件事办妥（即实现接口中的方法、属性、索引等，因为接口中只能大概声明，没有具体实现，等着一线工作人员来完成）。

接口是使用关键字 interface 定义的，声明接口的语法格式如下：

[访问修饰符] interface 接口名称

{

接口成员;

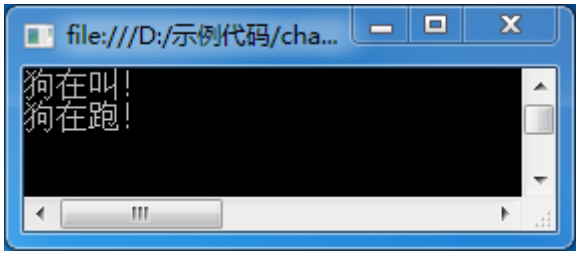
}

【实例】声明一个接口 Animal，在接口中定义了一个抽象方法 Bark()，并定义一个类 Dog 来实现接口中的所有方法。

程序代码如下：

```
//定义测试类
class Program
{
    static void Main(string[] args)
    {
        Dog dog = new Dog();
        dog.Bark();
        dog.Run();
        Console.ReadLine();
    }
}
//定义Animal接口
interface Animal
{
    void Bark();
}
//定义Dog类实现Animal接口
class Dog : Animal
{
    public void Bark()
    {
```

```
            Console.WriteLine("狗在叫！");
        }
        public void Run()
        {
            Console.WriteLine("狗在跑！");
        }
    }
```



【分析】

本实例中定义了 Animal 接口，并定义 Dog 类继承 Animal 接口，Dog 类的 Bark()方法实现了 Animal 接口的 Bark()方法。同时，Dog 类中也定义了自己的方法 Run()。

【注意】

对于接口中定义的成员有如下要求：

（1）接口的成员必须是方法、属性、事件或索引器。接口不能包含常量、字段、运算符、构造方法以及任何类的静态成员等；

（2）接口不提供对它所定义成员的实现，实现由继承的类来完成；

（3）接口成员都是 public 类型的，但不能使用 public 修饰符；

（4）一个类虽然只能继承一个基类，但可以实现任意数量的接口。

二、结构

结构类型（struct）通常用来封装小型相关变量组，如学生的成绩、商品的类型等。在 C#中，作为一个整体的学生（例如名称为 Student），称为结构型，而学生的姓名、学号、成绩等数据项称为结构型的成员。

结构类型是使用关键字 struct 定义的，声明结构类型的语法格式如下：

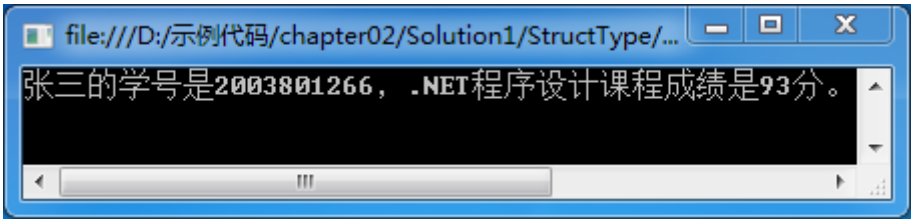
```
[访问修饰符] struct  结构名称
{
    结构体;
}
```

【实例】声明一个结构 Student，其中 name、id、score 是结构的数据成员，并在 Main()方法中使用该结构赋值并输出。

程序代码如下：

```
//定义测试类
class Program
{
    static void Main(string[] args)
    {
        Student s;   //使用结构类型
        s.name = "张三";
        s.id = 2003801266;
        s.score = 93;
```

```
            Console.WriteLine("{0}的学号是{1}，.NET程序设计课程成绩是{2}分。", s.name, s.id,
s.score);
            Console.Read();
        }
        //定义结构Student
        struct Student
        {
            public string name;
            public int id;
            public int score;
        }
    }
```



【分析】

本实例中定义了一个结构类型，类型名称为 Student，它包含 3 个成员，其中 public 为各成员的访问权限。为了在程序中使用该结构类型的数据，定义了 Student 类型的变量 s，并分别为变量 s 的 name、id 和 score 赋值并输出。

在程序中进行输出时，采用了一种新的方式，即占位符的方式。{0}、{1}、{2}即是点位符，这两个位置会被后续的变量列表依次替代。例如此例中，{0}将被变量 s 的 name 值替代, {1}将被变量 s 的 id 值替代，{2}将被变量 s 的 score 值替代。

【注意】

C#内置的结构类型主要有 DateTime 和 TimeSpan 等。DateTime 表示某个时间点，其成员主要有：Year、Month、Day、Hour、Minute、Second、Today、Now 等，分别表示年、月、日、时、分、秒、今天、当前时间。TimeSpan 表示某个时间段，其成员主要有：Days、Hours、Minutes、Seconds 等，分别表示某个时间段的天数、小时数、分数、秒数。

三、枚举

枚举类型（enum）是一种由一组被称为枚举数列表的常数所组成的独特类型。每种枚举类型都有对应的数据类型，可以是除 char 以外的任何简单数据类型。

枚举类型是使用关键字 enum 定义的，声明枚举类型的语法格式如下：

[访问修饰符] enum 枚举类型名称

{

枚举列表

};

例如，当数字 0、1、2、3、4、5、6 表示星期时，为直观起见，我们可以先使用一组中文符号来表示它们，依次为：星期日、星期一、星期二、星期三、星期四、星期五、星期六，并给它们取一个统一的名称如 Weekdays，使用 enum 来标记，完整代码如下：

```
enum Weekdays { 星期日, 星期一, 星期二, 星期三, 星期四, 星期五, 星期六 };
```

其中，Weekdays 是枚举型的名称，而花括号中的中文字符分别表示 7 个不同的枚举元素。

【实例】声明一个枚举类型 Weekdays（枚举列表中成员的默认类型为 int，其中星期一

是第一个枚举数，值为 0，后面每个枚举数的值依次递增 1），并使用该枚举类型并输出。

程序代码如下：

```
//定义测试类
class Program
{
    static void Main(string[] args)
    {
        Weekdays myday = (Weekdays)DateTime.Now.DayOfWeek;//使用枚举类型
        Console.WriteLine("今天是：{0}",myday);
        Console.ReadLine();
    }
}
//定义枚举类型Weekdays
enum Weekdays { 星期日, 星期一, 星期二, 星期三, 星期四, 星期五, 星期六 };
}
```

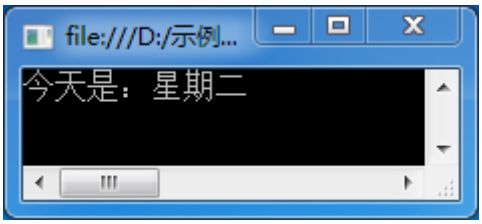


图 2-2 EnumType 项目运行结果

【分析】

本实例中定义了枚举类型，枚举名为 Weekdays，枚举值一共七个，即一周中的七天，所有被声明为 Weekdays 类型的变量值只能是七天中的某一天。

【注意】

（1）枚举元素的数据值是确定的，一旦声明，就不能在程序的运行过程中更改；

（2）枚举元素的个数是有限的，同样一旦声明，就不能在程序的运行过程中增加或减少；

（3）默认情况下，枚举的值是一个整数，第一个枚举数的值默认为 0，也可以自定义更改，后面每个枚举数的值依次递增 1；

（4）如果需要修改默认的规则，则重写枚举元素的值即可，例如：

enum EnumType {a=7,b,c=12,d,e};

在此枚举型中，a 的值为 7，b 为 8，c 为 12，d 为 13，e 为 14。

（5）枚举型与结构型是有区别的。结构实质上是若干个数据成员与数据操作的组合，一个结构型数的值是由各个成员的值组合而成的，结构型的各个数据成员的数据类型可以是不相同的，例如在实例2-1中的结构型变量s的值是由"张三"、2003801266与93这3个数据构成的。而枚举型的各个枚举元素的数据类型是相同的，枚举数只能代表某一个枚举元素的值，例如在实例2-2中的枚举变量myday在程序中只代表枚举元素星期二，其值为2。