互动出版网 China-Pub.com 专业图书网上第一专营店 www.china-pub.com 独家提供样章

第3章　变量和常量

想要成功，就要学习在机遇从头顶上飞过时跳起来抓住他，这样逮到机遇的概率大。

——比尔·盖茨

本章主要知识点

- 值类型
- 引用类型
- 类型的转换
- 装箱和拆箱
- 变量的概念及应用
- 常量的概念及应用

3.1　变量和常量的数据类型

教学录像：光盘\mr\video\第3章\变量和常量的数据类型.exe

在编写*C#*程序时，无论是声明变量还是常量，都必须用到数据类型，*C#*中的数据类型根据其定义可以分为两种：一种是值类型，另一种是引用类型。这两种类型的差异在于数据的存储方式，值类型的变量本身直接存储数据。而引用类型则存储实际数据的引用，程序通过此引用找到真正的数据，在以下内容中将会对这些类型进行详细讲解。

3.1.1　值类型

值类型变量直接存储其数据值，主要包含整数类型、浮点类型以及布尔类型等。值类型变量在堆栈中进行分配，因此效率很高，使用值类型主要目的是为了提高性能。值类型具有如下特性：

- 值类型变量都存储在堆栈中。
- 访问值类型变量时，一般都是直接访问其实例。
- 每个值类型变量都有自己的数据副本，因此对一个值类型变量的操作不会影响其他变量。
- 复制值类型变量时，复制的是变量的值，而不是变量的地址。
- 值类型变量不能为*null*，必须具有一个确定的值。

值类型是从*System.ValueType*类继承而来的类型，下面详细介绍值类型中包含的几种数据类型。

1. 整数类型

整数类型代表一种没有小数点的整数数值，在*C#*中内置的整数类型如表3.1所示。

互动出版网 China-Pub.com 专业图书网上第一专营店 www.china-pub.com 独家提供样章

表3.1　C#内置的整数类型

类　型	说　明	范　围
sbyte	8位有符号整数	- 128~127
short	16位有符号整数	- 32 768~32 767
int	32位有符号整数	- 2 147 483 648~2 147 483 647
long	64位有符号整数	- 9 223 372 036 854 775 808~9 223 372 036 854 775 807
byte	8位无符号整数	0~255
ushort	16位无符号整数	0~65 535
uint	32位无符号整数	0~4 294 967 295
ulong	64位无符号整数	0~18 446 744 073 709 551 615

说明：***byte***类型以及***short***类型是范围比较小的整数，如果正整数的范围没有超过65 535，声明为***ushort***类型即可，当然更小的数值直接以***byte***类型作处理即可。只是使用这种类型时必须特别注意数值的大小，否则可能会导致运算溢出的错误。

【例3.1】创建一个控制台应用程序，在其中声明一个***int***类型的变量***mr***，并初始化为2010；声明一个***byte***类型的变量***kj***，并初始化为255，最好输出。代码如下。（实例位置：光盘***mr******example***\\第3章\\3.1）

代码位置：光盘\mr\example\第3章\3.1\Test02\Test02\Program.cs

```
01    static void Main(string[] args)
02    {
03        int mr = 2010;                          //声明一个int类型的变量mr
04        byte kj = 255;                          //声明一个byte类型的变量kj
05        Console.WriteLine("mr={0}", mr);        //输出int类型变量mr
06        Console.WriteLine("kj={0}", kj);        //输出byte类型变量kj
07        Console.ReadLine();
08    }
```

程序的运行结果如图3.1所示。

图3.1　整数类型实例运行结果图

图3.2　错误提示

注意：如果将***byte***类型的变量mr赋值为266，重新编译程序，就会出现错误提示，如图3.2所示。这是因为***byte***类型的变量是8位无符号整数，其范围在0～255之间，266已经超出了***byte***类型的范围，所以编译程序会出现错误提示。

试一试：修改上面的程序，计算这两个变量（***mr***和***kj***）的差，并输出到控制台。

2. 浮点类型

浮点类型变量主要用于处理含有小数的数值数据，浮点类型主要包含了***float***、***double***和***decimal***这3种数值类型。表3.2列出了这3种数值类型的描述信息。

互动出版网 China-Pub.com 专业图书网上第一专营店 www.china-pub.com 独家提供样章

表3.2 浮点类型及描述

类 型	说 明	精 度	范 围
float	4字节IEEE单精度浮点数	精确到7位数	1.5×10^{-45}~3.4×10^{38}
double	8字节IEEE双精度浮点数	精确到15~16位数	50×10^{-324}~1.7×10^{308}
decimal	16字节浮点数	精确到28~29位数	1.0×10^{-28}~1.7×10^{28}

如果不做任何设置，包含小数点的数值都被认为是*double*类型，例如9.27，没有特别指定的情况下，这个数值是*double*类型。如果要将数值以*float*类型来处理，就应该通过强制使用*f*或*F*将其指定为*float*类型。

【例3.2】下面的代码就是将数值强制指定为*float*类型。实现代码如下。

```
01   float theMySum = 9.27f;                 //使用f强制指定为float类型
02   float theMuSums = 1.12F;                //使用F强制指定为float类型
```

说明：如果要将数值强制指定为*double*类型，则需要使用*d*或*D*进行设置。

【例3.3】下面的代码就是将数值强制指定为*double*类型。实现代码如下。

```
01   double myDou = 927d;                    //使用d强制指定为double类型
02   double mudou = 112D;                    //使用D强制指定为double类型
```

注意：如果需要使用*float*类型变量时，必须在数值的后面跟随*f*或*F*，否则编译器会直接将其作为*double*类型处理；也可以在*double*类型的值前面加上（*float*），对其进行强制转换。

3. 布尔类型

布尔类型主要用来表示*true/false*值，一个布尔类型的变量，其值只能是*true*或者*false*，不能将其他的值指定给布尔类型变量，布尔类型变量不能与其他类型之间进行转换。

【例3.4】将927赋值给布尔类型变量*x*，代码如下。

```
bool x = 927;
```

这样赋值显然是错误的，编译器会返回错误提示“常量值927无法转换为*bool*”。布尔类型变量大多数被应用到流程控制语句当中，例如循环语句或者*if*语句等。

说明：在定义全局变量时，如果没有特定的要求不用对其进行初始化，整数类型和浮点类型的默认初始化为0，布尔类型的初始化为*false*。

3.1.2 引用类型

引用类型是构建*C#*应用程序的主要对象类型数据。引用类型的变量又称为对象，可存储对实际的引用。*C#*支持两种预定义的引用类型，即*object*和*string*，其说明如表3.3所示。

表3.3 引用类型及说明

设 置 值	描 述
object	object类型在.NET Framework中是Object的别名。在C#的统一类型系统中，所有类型（预定义类型、用户定义类型、引用类型和值类型）都是直接或间接从Object继承的
string	string类型表示零或更多Unicode字符组成的序列

在应用程序执行的过程中，预先定义的对象类型以*new*创建对象实例，并且存储在堆栈中。堆栈是一种由系统弹性配置的内存空间，没有特定大小及存活时间，因此可以被弹性地运用于

互动出版网 China-Pub.com 专业图书网上第一专营店 www.china-pub.com 独家提供样章

对象的访问。

引用类型具有如下特征。

- 必须在托管堆中为引用类型变量分配内存。
- 必须使用*new*关键字来创建引用类型变量。
- 在托管堆中分配的每个对象都有与之相关联的附加成员，这些成员必须被初始化。
- 引用类型变量是由垃圾回收机制来管理的。
- 多个引用类型变量可以引用同一对象，这种情形下，对一个变量的操作会影响另一个变量所引用的同一对象。
- 引用类型被赋值前的值都是*null*。

【例3.5】创建一个控制台应用程序，在其中创建一个类*C*，在此类中建立一个字段*Value*，并初始化为0，然后在程序的其他位置通过*new*创建对此类的引用类型变量，最后输出，代码如下。（实例位置：光盘\mr\example\第3章\3.5）

代码位置：光盘\mr\example\第3章\3.5\Test03\Test03\Program.cs

```
01  class Program
02  {
03      class C                                  //创建一个类C
04      {
05          public int Value = 0;                //声明一个公共int类型的变量Value
06      }
07      static void Main(string[] args)
08      {
09          int mr = 0;                          //声明一个int类型的变量mr，并初始化为0
10          int kj = mr;                         //声明一个int类型的变量kj，并将v1赋值给v2
11          kj = 2011;                           //重新将变量kj赋值为2011
12          C MR = new C();                      //使用new关键字创建引用对象
13          C KJ = MR;                           //使MR等于KJ
14          KJ.Value = 112;                      //设置变量KJ的Value值
15          Console.WriteLine("Values:mr={0},kj={1}", mr, kj);//输出变量mr和kj
16          Console.WriteLine("Refs:MR={0},KJ={1}", MR.Value, KJ.Value);//输出引用类型对象的Value值
17          Console.ReadLine();
18      }
19  }
```

程序的运行结果如图3.3所示。

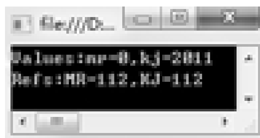

图3.3 引用类型的实现实例运行结果图

试一试：修改上面的程序，在“*KJ.Value = 112;*”语句的下一行添加“*mr= KJ.Value;*”这条语句，然后再输出相关变量和字段的信息。

3.1.3 值类型与引用类型的区别

从概念上看，值类型直接存储其值，而引用类型存储对其值的引用。这两种类型存储在内

互动出版网 China-Pub.com 专业图书网上第一专营店 www.china-pub.com 独家提供样章

存的不同地方。在*C#*中，必须在设计类型的时候就决定类型实例的行为。如果在编写代码时不能理解引用类型和值类型的区别，那么将会给代码带来不必要的异常。

从内存空间上看，值类型是在栈中操作，而引用类型则在堆中分配存储单元。栈在编译的时候就分配好内存空间，在代码中有栈的明确定义，而堆是程序运行中动态分配的内存空间，可以根据程序的运行情况动态地分配内存的大小。因此，值类型总是在内存中占用一个预定义的字节数。而引用类型的变量则在栈中分配一个内存空间，这个内存空间包含的是对另一个内存位置的引用，这个位置是托管堆中的一个地址，即存放此变量实际值的地方。

也就是说值类型相当于现金，要用就直接用，而引类型相当于存折，要用得先去银行取。

但值类型在栈上分配内存，而引用类型在托管堆上分配内存，只是一种笼统的说法。下面对其进行详细描述。

（1）对于值类型的实例，如果作为方法中的局部变量，则被创建在线程栈上；如果该实例作为类型的成员，则作为类型成员的一部分，连同其他类型字段存放在托管堆上。

每种值类型均有一个隐式的默认构造函数来初始化该类型的默认值。例如：

```
int i = new int();
```

等价于：

```
Int32 i = new Int32();
```

等价于：

```
int i = 0;
```

等价于：

```
Int32 i = 0;
```

使用***new***运算符时，将调用特定类型的默认构造函数并对变量赋以默认值。在上例中，默认构造函数将值0赋给了***i***。

说明：***C#***的所有值类型均隐式派生自***System.ValueType***，而***System.ValueType***直接派生于***System.Object***。即***System.ValueType***本身是一个类类型，而不是值类型。其关键在于***ValueType***重写了***Equals***方法，从而对值类型按照实例的值来比较，而不是引用地址来比较。

（2）引用类型的实例创建在托管堆上。

【例3.6】下面以一段代码来详细讲解一下值类型与引用类型的区别，代码如下。（实例位置：光盘***mr******example***\\第3章\\3.6）

代码位置：光盘\mr\example\第3章\3.6\Test04\Test04\Program.cs

```
01   namespace Test03
02   {
03           class Program
04           {
05               static void Main(string[] args)
06               {
07                   //调用ReferenceAndValue类中的Demonstration方法
08                   ReferenceAndValue.Demonstration();
09                   Console.ReadLine();
10               }
```

互动出版网 China-Pub.com 专业图书网上第一专营店 www.china-pub.com 独家提供样章


```
11          }
12          public class stamp                              //定义一个类
13          {
14              public string Name { get; set; }            //定义引用类型
15              public int Age { get; set; }                //定义值类型
16          }
17          public static class ReferenceAndValue           //定义一个静态类
18          {
19              public static void Demonstration()          //定义一个静态方法
20              {
21                  stamp Stamp_1 = new stamp { Name = "Premiere", Age = 25 }; //实例化
22                  stamp Stamp_2 = new stamp { Name = "Again", Age = 47 }; //实例化
23                  int age = Stamp_1.Age;                  //获取值类型Age的值
24                  Stamp_1.Age = 22;                       //修改值类型的值
25                  stamp guru = Stamp_2;                   //获取Stamp_2中的值
26                  Stamp_2.Name = "Again Amend";           //修改引用的Name值
27                  Console.WriteLine("Stamp_1's age:{0}", Stamp_1.Age); //显示Stamp_1中的Age值
28                  Console.WriteLine("age's value:{0}", age); //显示age值
29                  Console.WriteLine("Stamp_2's name:{0}", Stamp_2.Name); //显示Stamp_2中的Name值
30                  Console.WriteLine("guru's name:{0}", guru.Name); //显示guru中的Name值
31              }
32      }
33  }
```

程序的运行结果如图3.4所示。

图3.4　值类型与引用类型

在图3.14中可以看出，当改变了***Stamp_1.Age***的值时***age***没跟着变，而在改变了***anders.Name***的值后，***guru.Name***却跟着变了，这就是值类型和引用类型的区别。在声明***age***值类型变量时，将 ***Stamp_1.Age***的值赋给它，这时，编译器在栈上分配了一块空间，然后把***Stamp_1.Age***的值填进去，二者没有任何关联，就像在计算机中复制文件一样，只是把***Stamp_1.Age***的值拷贝给***age***了。而引用类型则不同，在声明***guru***时把***Stamp_2***赋给它，前面说过，引用类型包含的只是堆上数据区域地址的引用，其实就是把***Stamp_2***的引用也赋给***guru***，因此它们指向了同一块内存区域。既然是指向同一块区域，不管修改谁，另一个的值都会跟着改变，就像信用卡跟亲情卡一样，用亲情卡取了钱，与之关联的信用卡账上也会跟着发生变化。

试一试：修改上面的程序，在“***stamp guru = Stamp_2;***”语句的下一行添加“***Stamp_2= Stamp_1;***”这条语句，然后再输出相关属性和变量的值。

3.1.4　枚举类型

枚举类型是一种独特的值类型，它用于声明一组具有相同性质的常量，编写与日期相关的应用程序时，经常需要使用年、月、日、星期等日期数据，可以将这些数据组织成多个不同名

互动出版网 China-Pub.com 专业图书网上第一专营店 www.china-pub.com 独家提供样章

称的枚举类型。使用枚举可以增加程序的可读性和可维护性。同时，枚举类型可以避免类型错误。

> 说明：在定义枚举类型时，如果不对其进行赋值，默认情况下，第一个枚举数的值为0，后面每个枚举数的值依次递增1。

在*C#*中使用关键字*enum*类声明枚举，其形式如下。

```
01   enum 枚举名
02   {
03       list1=value1,                        //给枚举元素1赋初始值
04       list2=value2,                        //给枚举元素2赋初始值
05       list3=value3,                        //给枚举元素3赋初始值
06       ...
07       listN=valueN,                  //给枚举元素N赋初始值
08   }
```

其中，大括号{}中的内容为枚举值列表，每个枚举值均对应一个枚举值名称，***value1~valueN***为整数数据类型，***list1~listN***则为枚举值的标识名称。下面通过一个实例来演示如何使用枚举类型。

【例3.7】本示例首先应用关键字***enum***定义一个枚举类型***MonthOfYear***，用来存储12个月份的名称，然后在***Program***类中分别对***MonthOfYear***枚举类型变量***month***赋初值。代码如下。（实例位置：光盘***mr******example***\\第3章\\3.7）

代码位置：光盘\mr\example\第3章\3.7\Test05\Test05\Program.cs

```
01   namespace Test05
02   {
03       enum MonthOfYear                     //定义一个枚举类型，描述月份
04       {
05           January,Feburary,March,April,May,June,July,Aguest,September,October,November,December
06       }
07       class Program
08       {
09           static void Main(string[] args)              //入口方法
10           {
11               MonthOfYear month;                       //定义一个枚举变量
12               month = MonthOfYear.May;                 //引用一个枚举值
13               Console.WriteLine("本月是{0}",month);     //输出本月的值
14               Console.ReadLine();
15           }
16       }
17   }
```

程序的运行结果如图3.5所示。

图3.5 枚举类型的实现实例运行结果图

互动出版网 China-Pub.com 专业图书网上第一专营店 www.china-pub.com 独家提供样章

试一试：仿照上面的实例，定义一个存储星期几的名称（如***Monday***、***Tuesday***、***Wednesday***等）的枚举***OneOfWeek***，然后在***Main***方法中输出指定的某个枚举值。

3.1.5　类型转换

类型转换就是将一种类型转换成另一种类型。转换可以是隐式转换或者显式转换，本节将详细介绍这两种转换方式，并讲解有关装箱和拆箱的内容。

1. 隐式转换

所谓隐式转换就是不需要声明就能进行的转换。进行隐式转换时，编译器不需要进行检查就能安全地进行转换。表3.4列出了可以进行隐式转换的数据类型。

表3.4　隐式类型转换表

源 类 型	目标类型
sbyte	short、int、long、float、double、decimal
byte	short、ushort、int、uint、long、ulong、float、double或decimal
short	int、long、float、double或decimal
ushort	int、uint、long、ulong、float、double或decimal
int	long、float、double或decimal
uint	long、ulong、float、double 或 decimal
char	ushort、int、uint、long、ulong、float、double或decimal
float	double
ulong	float、double或decimal
long	float、double或decimal

从***int***、***uint***、***long***或***ulong***到***float***，以及从***long***或***ulong***到***double***的转换可能导致精度损失，但是不会影响其数量级。其他的隐式转换不会丢失任何信息。

说明：对于不同值类型之间的转换，如果从低精度、小范围的数据类型转换为高精度、大范围是数据类型，可以使用隐式转换。

【例3.8】将***int***类型的值隐式转换成***long***类型，代码如下。

```
01    int i=2011;                  //定义整形变量并初始化
02    long l=1;                    //定义长整形变量并初始化
03    l=i;                         //隐式转换
```

2. 显式转换

显式转换也可以称为强制转换，需要在代码中明确地声明要转换的类型。如果要在不存在隐式转换的类型之间进行转换，就需要使用显式转换。表3.5列出了需要进行显式转换的数据类型。

表3.5　显式类型转换表

源 类 型	目标类型
sbyte	byte、ushort、uint、ulong或char
byte	sbyte和char
short	sbyte、byte、ushort、uint、ulong或char
ushort	sbyte、byte、short或char
int	sbyte、byte、short、ushort、uint、ulong或char

互动出版网 China-Pub.com 专业图书网上第一专营店 www.china-pub.com 独家提供样章

（续）

源 类 型	目标类型
uint	sbyte、byte、short、ushort、int或char
char	sbyte、byte或short
float	sbyte、byte、short、ushort、int、uint、long、ulong、char或decimal
ulong	sbyte、byte、short、ushort、int、uint、long或char
long	sbyte、byte、short、ushort、int、uint、ulong或char
double	sbyte、byte、short、ushort、int、uint、ulong、long、char或decimal
decimal	sbyte、byte、short、ushort、int、uint、ulong、long、char或double

由于显式转换包括所有隐式转换和显式转换，因此总是可以使用强制转换表达式从任何数值类型转换为任何其他的数值类型。

【例3.9】创建一个控制台应用程序，将*double*类型的*x*进行显式类型转换，代码如下。

```
01   static void Main(string[] args)
02   {
03       double x = 19810927.0112;                   //建立double类型变量x
04       int y = (int)x;                             //显示转换成整型变量y
05       Console.WriteLine(y);                       //输出整型变量y
06       Console.ReadLine();
07   }
```

程序运行结果为19810927。

也可以通过*Convert*关键字进行显式类型转换，上述例子还可以通过下面代码实现。

【例3.10】创建一个控制台应用程序，通过*Convert*关键字进行显式类型转换，代码如下。

```
01   double x = 19810927.0112;                       //建立double类型变量x
02   int y = Convert.ToInt32(x);                     //通过Convert关键字转换
03   Console.WriteLine(y);                           //输出整型变量y
04   Console.ReadLine();
```

3. 装箱和拆箱

将值类型转换为引用类型的过程叫做装箱，相反，将引用类型转换为值类型的过程叫做拆箱，下面将通过例子详细介绍装箱与拆箱的过程。

（1）装箱

装箱允许将值类型隐式转换成引用类型，下面通过一个实例演示如何进行装箱操作。

【例3.11】创建一个控制台应用程序，声明一个整型变量*i*并赋值为2011，然后将其赋值到装箱对象*obj*中，最后再改变变量*i*的值，代码如下。（实例位置：光盘*mr**example*\第3章\3.11）

代码位置：光盘\mr\example\第3章\3.11\Test06\Test06\Program.cs

```
01   static void Main(string[] args)
02   {
03       int i = 2011;                  //声明一个int类型变量i，并初始化为2011
04       object obj = i;                //声明一个object类型obj，其初始化值为i
05       Console.WriteLine("1、i的值为{0}，装箱之后的对象为{1}", i, obj);
06       i = 927;                       //重新将i赋值为927
07       Console.WriteLine("2、i的值为{0}，装箱之后的对象为{1}", i, obj);
08       Console.ReadLine();
09   }
```

互动出版网 China-Pub.com 专业图书网上第一专营店 www.china-pub.com 独家提供样章

运行结果如图3.6所示。

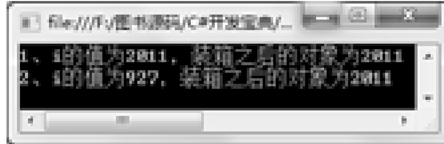

图3.6　装箱运行效果图

从程序运行结果可以看出，值类型变量的值复制到装箱得到的对象中，装箱后改变值类型变量的值，并不会影响装箱对象的值。

试一试：修改上面的实例，把“*i = 927;*”这条语句修改为“*obj=927;*”，然后在输出相关变量的值。

（2）拆箱

拆箱允许将引用类型显式转换为值类型，下面通过一个示例演示拆箱的过程。

【例3.12】创建一个控制台应用程序，声明一个整型变量***i***并赋值为2011然后将其复制到装箱对象***obj***中，最后，进行拆箱操作将装箱对象***obj***赋值给整型变量***j***，代码如下。（实例位置：光盘***mr******example***\\第3章\\3.12）

代码位置：光盘\mr\example\第3章\3.12\Test07\Test07\Program.cs

```
01    static void Main(string[] args)
02    {
03        int i = 2011;                    //声明一个int类型的变量i，并初始化为2011
04        object obj = i;                  //执行装箱操作
05        Console.WriteLine("装箱操作：值为{0}，装箱之后对象为{1}", i, obj);
06        int j = (int)obj;                //执行拆箱操作
07        Console.WriteLine("拆箱操作：装箱对象为{0}，值为{1}", obj, j);
08        Console.ReadLine();
09    }
```

运行结果如图3.7所示。

图3.7　拆箱运行效果图

说明：装箱是将一个值类型转换为一个对象类型（***object***），而拆箱则是将一个对象类型显式转换为一个值类型。对于装箱而言，它是将被装箱的值类型复制一个副本来转换，而对于拆箱而言，需要注意类型的兼容性，例如，不能将一个值为***string***的***object***类型转换为***int***类型。

试一试：修改上面的实例，在“*int j = (int)obj;*”语句的上一行添加“*i=927;*”这条语句，然后再输出相关变量的值。

互动出版网 China-Pub.com 专业图书网上第一专营店 www.china-pub.com 独家提供样章

上机练习

上机练习1　使用值类型和引用类型输出不同的字段

创建一个控制台应用程序，演示值类型与引用类型，新建两个引用类型的实例***obj1***、***obj2***并分别为其赋初值“婷子”、“英豪”，定义两个值类型变量***v1***、***v2***并为其赋初值2和5，然后分别输出值类型和引用类型的字段的值。将***v1***的值赋给***v2***，***obj1***的值赋给***obj2***，将***obj2***的值改为“喜婷”，再分别输出值类型和引用类型字段的值。重新对***v2***赋值为10，***obj2***得到一个新的实例“小雪”，再次输出输出值类型和引用类型字段的值，运行结果如图3.8所示。

图3.8　值类型和引用类型实例的运行结果图

上机练习2　判断当前系统日期是星期几

程序首先通过***enum***关键字建立一个枚举，枚举值名称分别代表一周的七天，如果枚举值名称是***Sun***，说明其代表的是一周中的星期日，其枚举值为0，依此类推。然后，声明一个***int***类型的变量***k***，用于获取当前表示的日期是星期几。最后，调用***swith***语句，输出当天是星期几。例如今天是2011/5/7星期六，应输出“今天是星期六”。运行结果如图3.9所示。

图3.9　判断当前系统日期是星期几

3.2　声明和使用变量

教学录像：光盘\mr\video\第3章\声明和使用变量.exe

变量用来表示一个数值、一个字符串值或者一个类的对象。变量存储的值可能会发生更改，但变量名称保持不变。

C#定以了7类变量，即静态变量、实例变量、数组元素、值参数、引用参数、输出参数和局部变量。

3.2.1　变量的基本概念

变量本身被用来存储特定类型的数据，可以根据需要随时改变变量中所存储的数据值。变量具有名称、类型和值。变量名是变量在程序源代码中的标识。变量类型确定它所代表的内存的大小和类型，变量值是指它所代表的内存块中的数据。在程序的执行过程中，变量的值可以发生变化。使用变量之前必须先声明变量，即指定变量的类型和名称。

3.2.2　声明变量

声明变量就是指定变量的名称和类型，变量的声明非常重要，未经声明的变量本身并不合

互动出版网 China-Pub.com 专业图书网上第一专营店 www.china-pub.com 独家提供样章

法，也因此无法在程序当中使用。在*C#*中，声明一个变量是由一个类型和跟在后面的一个或多个变量名组成，多个变量之间用逗号分开，声明变量以分号结束。

在声明变量时，要注意变量名的命名规则。*C#*的变量名是一种标识符，应该符合标识符的命名规则。变量名是区分大小写的，下面列出变量的命名规则。

- ❑ 变量名只能由数字、字母和下划线组成。
- ❑ 变量名的第一个符号只能是字母和下划线，不能是数字。
- ❑ 不能使用关键字作为变量名。
- ❑ 一旦在一个语句块中定义了一个变量名，那么在变量的作用域内都不能再定义同名的变量。

【例3.13】声明一个整型变量*mrkj*，代码如下。

```
int mrkj=2011;                                  //声明一个整型变量mrkj，并赋初值为2011
```

【例3.14】声明一个整型变量*mrkj*然后赋初值，代码如下。

```
01    int mrkj;                                 //声明一个整型变量mrkj
02    mrkj=2011;                                //为整型变量mrkj赋初值2011
```

【例3.15】用户一条语句声明变量并赋初值，各个变量之间用逗号分隔，代码如下。

```
int mr=2011,kj,mrkj=1,mrsoft;                   //声明变量并赋初值
```

3.2.3　变量的作用域

变量的作用域就是可以访问该变量的代码区域。一般情况下，可以通过以下规则确定变量的作用域。

- ❑ 只要字段所属的类在某个作用域内，其字段也在该作用域内。
- ❑ 局部变量存在于表示声明该变量的块语句或方法结束的封闭花括号之前的作用域内。
- ❑ 在*for*、*while*或类似语句中声明的局部变量存在于该循环体内。

变量按照作用域被划分为3种：全局变量、局部变量和循环内部变量。

【例3.16】创建一个控件台应用程序，在*Test*类中建立一个全局变量*Name*，调用方法*show*和*show2*都可以访问全局变量，代码如下。（实例位置：光盘*mr**example*\\第3章\\3.16）

代码位置：光盘\mr\example\第3章\3.16\Test08\Test08\Program.cs

```
01    class Program
02    {
03        static void Main(string[] args)
04        {
05            Test t = new Test();                         //创建一个实例
06            t.Name = "东方之珠";                          //给类的字段赋值
07            t.show();                                    //调用show方法
08            t.show2();                                   //调用show2方法
09            Console.ReadLine();
10        }
11    }
12    class Test
13    {
14        public string Name;
15        public void show()                               //输出名字
16        {
```

互动出版网 China-Pub.com 专业图书网上第一专营店 www.china-pub.com 独家提供样章


```
17            Console.WriteLine("姓名：{0}",Name);
18        }
19        public void show2()                                    //输出欢迎信息
20        {
21            Console.WriteLine("你好: {0}",Name);
22        }
23    }
```

运行效果如图3.10所示。

图3.10 全局变量的实现实例运行结果图

说明：全局变量是在程序加载时就分配了内存，整个程序运行完时才回收；局部变量是在程序运行到时才分配，这个方法执行完就回收，所以应尽可能地少定义全局变量。

试一试：修改上面的实例，把字段“ *Name* ”改成属性进行定义，然后再赋值和输出。

3.2.4 变量赋值

在*C#*中，使用赋值运算符“ = ”（等号）来给变量赋值，将等号右边的值赋给左边的变量。

【例3.17】首先声明两个变量***sum***和***num***，然后将变量***sum***赋值为2011，最后将变量***sum***赋值给变量***num***，代码如下。

```
01    int sum,num;                            //声明两个变量
02    sum =2011;                              //给变量sum赋值为2011
03    num = sum;                              //将变量sum赋值给变量num
```

技巧：在对多个同类型的变量赋同一个值时，为了节省代码的行数，可以同时对多个变量进行初始化：***int a, b, c, d, e; a = b = c = d = e = 0;***

上机练习

上机练习3 定义局部变量输出不同的字段

在***Test***类的***show***方法中定义一个局部变量***Name***，并赋初值。局部变量***Name***只能在***show***方法中可以访问。同样在***show2***方法中也定义一个局部变量。运行效果如图3.11所示。

图3.11 定义局部变量输出不同的字段

上机练习4 定义循环内部变量并输出变量的值

在***Test***类的***show***方法中定义一个局部变量***i***，局部变量***i***的作用域只限于***for***循环体内。运行效果如图3.12所示。

互动出版网 China-Pub.com 专业图书网上第一专营店 www.china-pub.com 独家提供样章

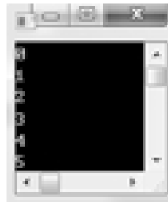

图3.12　定义循环内部变量并输出变量的值

3.3　声明和使用常量

教学录像：光盘\mr\video\第3章\声明和使用常量.exe

C#提供了两种类型的常量，一种是用***const***关键字声明，另一种是用***readonly***关键字声明。在数学上圆周率π的值为3.1415926...，这个值不会随着外界环境而改变。如果在代码中间接地更改了π的值，程序运行会出现未知的错误；如果将π的值设为常量，在某种程度上会增加程序的健壮性。

常量就是其值固定不变的量，而且常量的值在编译时就已经确定了。常量的类型只能为下列类型之一：***sbyte***、***byte***、***short***、***ushort***、***int***、***uint***、***long***、***ulong***、***char***、***float***、***double***、***decimal***、***bool***、***string***等。使用关键字***const***来创建常量，并且在创建常量时必须设置它的初始值。

在***C#***中，***const***常量是在编译时设置其值并且永远不能更改其值的字段，在对程序进行编译的时候，编译器会把所有***const***常量替换为常数。

【例3.18】声明一个正确的常量，同时再声明一个错误的常量，以便读者对比参考，代码如下。

```
01    const double PI = 3.1415926;                    //正确的声明方法
02    const int MyInt;                                //错误：定义常量时没有初始化
```

注意：常量的值在编译时就已经确定了，***C#***中使用关键字***const***来声明常量，而且在声明常量时必须对其进行初始化。程序中使用常量时，实际不需要为常量分配内存，可以在程序集元数据中提取常量的值。

在***C#***中使用***readonly***关键字来声明只读字段，只读的字段与常量的用法很相似，都是为程序提供一个只读的值，不同之处在于，***readonly***字段无须在编译时初始化其值，可以在类型的构造方法中初始化***readonly***的值，这样***readonly***的值可以被写入一次。

【例3.19】创建一个控制台应用程序，定义一个***readonly***只读字段并使用类的构造方法对***readonly***字段初始化，代码如下。（实例位置：光盘***mr******example***\\第3章\\3.19）

代码位置：光盘\mr\example\第3章\3.19\Test09\Test09\Program.cs

```
01    class Program
02    {
03        static void Main(string[] args)
04        {
05         Test test = new Test(22);                  //创建类的实例
06            test.show();                            //调用show方法
07            Console.ReadLine();
08        }
```

互动出版网 China-Pub.com 专业图书网上第一专营店 www.china-pub.com 独家提供样章


```
09        class Test
10        {
11            public Test(int age)            //在类的构造方法中初始化只读字段Age
12            {
13                this.Age = age;             //初始化年龄
14            }
15            readonly int Age;               //定义只读的字段
16            public void show()
17            {
18                Console.WriteLine("年龄: {0}",Age);
19            }
20        }
21    }
```

本例运行结果为“ 年龄：22 ”。

试一试：修改上面的实例，再添加一个只读字段*Name*，然后在构造方法*Test*中初始化其值。

上机练习

上机练习5 定义常量计算圆的周长

创建一个控制台应用程序，定义一个常量字段*pi*赋值为3.14，根据圆的周长公式计算圆的周长，当圆的半径为20时，周长为125.663704，如图3.13所示。

图3.13 定义常量计算圆的周长

上机练习6 同时使用常量和只读字段输出信息

要求定义一个类，在该类中定义一个描述国籍的常量，其常量值为“ 中国 ”，定义一个描述姓名的*readonly*字段，并在该类的构造方法中初始化其值，本例运行效果如图3.14所示。

图3.14 同时使用常量和只读字段输出信息

3.4 术语

（1）值类型的变量本身直接存储数据，而引用类型则存储实际数据的引用。
（2）枚举类型是一种独特的值类型，它用于声明一组具有相同性质的常量。
（3）类型转换就是将一种类型转换成另一种类型，转换可以是隐式转换或者显式转换。
（4）将值类型转换为引用类型的过程叫做装箱。
（5）将引用类型转换为值类型的过程叫做拆箱。
（6）变量本身被用来存储特定类型的数据，可以根据需要随时改变变量中所存储的数据值。

互动出版网 China-Pub.com 专业图书网上第一专营店 www.china-pub.com 独家提供样章

（7）常量就是其值固定不变的量，而且常量的值在编译时就已经确定了。

3.5　小结

本章对*C#*程序设计的基础知识进行了详细讲解，并通过大量的举例说明，使读者更好地理解所学知识的用法。在阅读本章时，读者应该重点掌握变量的使用和类型的转换，同时对*C#*中的数据类型及常量也要有一定的了解。本章学习的难点是引用类型的使用及装箱、拆箱操作，引用类型其实就是各种类型的对象，它存储的是对实际数据的引用，*C#*中提供好了两个预定义的引用类型*object*和*string*，而装箱和拆箱操作实质上就是值类型和引用类型相互转换的过程。

3.6　练习

1. 尝试开发一个程序，定义一个*long*类型的变量*n*并附初值，然后再定义一个*int*类型的变量*m*，并将变量*n*赋值给变量*m*，最后输出变量*m*的值。

2. 尝试开发一个程序，定义一个描述月份的枚举类型，并在*Main*方法中引用这个枚举类型。

3. 尝试开发一个程序，该程序中建立一个静态方法，在静态方法中声明一个局部变量，并对其赋值，然后输出。

4. 尝试开发一个程序，要求声明一个常量，然后试着更改这个常量的值，看会引发什么错误。

5. 尝试开发一个程序，使用*const*声明自己的姓名，使用*readonly*声明自己的年龄，然后在*Main*方法中输出自己的姓名和年龄。

HZ BOOKS 华章图书