# 第 2 章　VB 语言基础

## 2.1　VB 语句的一般规则

### 2.1.1　VB 语句的特点

要用设计出来的应用程序完成一定任务，就必须为事件编写程序代码。程序代码由语句构成。应用程序的主要任务都是对数据进行处理，而处理的具体操作由语句来完成。了解 VB 语句的基本特点有助于比较快速地熟悉使用 VB 编程。在 VB 的程序代码窗口中，每输入完一个语句，就要按 Enter 键以示结束。出现在语句中的对象名、属姓名、方法名等，称为命令词。在语句的输入过程中，VB 会对它们做简单的格式化处理，让命令词的第一个字母大写，后面跟着小写，VB 会在运算符的前后自动加上空格。

VB 具有自动缩进功能，即如果按 Tab 键向右缩进一个制表符后输入一条语句，按 Enter 键后下一条语句仍保持缩进一个制表符，与上一语句对齐，除非重新做调整。利用这种功能可使输入的程序层次分明，便于阅读和查错。除此之外我们还有必要了解 VB 语句的以下特点。

#### 1. VB 代码不区分字母的大小写

VB 系统不区分大小写字母，在同一个模块中小写“a”变量与大写的变量“A”是同一变量。同时，系统对用户输入的程序代码会进行自动转换，转换规则如下。

（1）代码中输入的关键字，首字母总被转换成大写，其余字母被转换成小写。

（2）若关键字由多个英文单词组成，则将每个英文单词的首字母转换成

大写。

（3）对于用户自己定义的变量、过程名，VB 以第一次定义的为准，以后输入的相同的变量、过程名向第一次输入的形式转换。

2. 语句书写自由

（1）在同一语句行上，可以是一条语句，也可以书写多个语句，语句之间用冒号“:”分隔。

（2）一个单行语句可以分为若干行书写，书写时在本行后面加上续行符，即加上空格和下画线“_”。

（3）一行最多允许书写 255 个字符。

3. 可以在语句中给出注释

为了提高程序的可读性，通常应在程序的适当位置加上必要的注释。VB 中的注释语句以撇号“'”开头。

4. VB 提供了部分语法检测功能，能帮助使用者及时发现错误

例如，假定原打算输入的语句为：

```
Command1. Caption = "欢迎使用 Visual Basic 6. 0!"
```

但不小心在控件名 Command1 与属性名 Caption 之间输入了两个点。这样，在语句输入完成后按 Enter 键时，VB 会立即给出提示性的警告信息，如图 2. 1 所示。

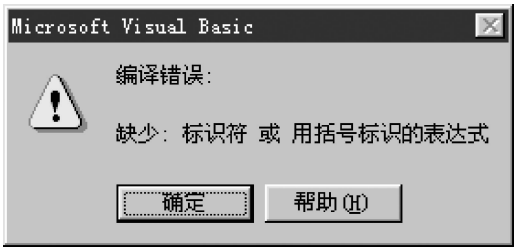



图 2. 1　警告信息

这是自动校验输入程序行语法的功能，受控于“选项”对话框“编辑器”页签中的设置。如果没有这个功能，是因为没有设置，这时通过菜单“工具|选项”命令可进入“选项”对话框，选中“编辑器”页签里的“自动语法检测(K)”复选框，就能使你的 VB 具有这一功能。

## 2. 1. 2　标识符命名规则

标识符是程序设计者为常量、变量、数组、控件、函数、过程等定义的标

识，利用它可以引用相应的常量、变量、数组等。在系统中有很多预先定义的标识符，即保留字。保留字是 VB 预先定义的标识符，本身具有特定的含义和用途，不能再用作一般的标识符。例如，保留字 Integer 表示数据类型是整型，Print 用于在指定对象上输出数据，Call 表示过程调用等。

用户定义标识符时必须以字母（A ~ Z，a ~ z）、数字（0 ~ 9）或下画线组成，也可以包含汉字，但第一个字符一定要是字母或汉字；标识符长度不能超过 255 个字符，其中窗体、控件和模块的标识符不能超过 40 个；标识符不能与 VB 的关键字同名。

### 2.1.3 赋值语句

所谓“赋值语句”，就是让变量或对象属性获得取值的操作，这里的“取值”是源操作数，它可以是变量、常量、表达式或一个对象的属性值；而获得该值的变量或对象属性，则是目标操作数。赋值语句的一般格式是：

```
目标操作数 = 源操作数
```

其中等号“ = ”为赋值运算符。要注意的是，等号两边的操作数应保持数据类型一致。

给变量赋值以及重新设置对象属性值，是编写 VB 程序代码中的两个最常见的任务，可以这样认为：赋值语句是 VB 变量获得新值或将信息从信息源复制到信息目的地的基本方式。

在书写赋值语句时，等号左边只能是变量名或对象属性，等号右边为常数值、变量或表达式。当等号右边为变量名时，必须保证它在此时已经有了明确的取值。下面是一些合法和不合法赋值语句的例子。

合法语句：

```
Total = 99
Amount = 100 + Total
Gcd $ = "早上好"
Text1. Text = "Visual Basic 6. 0 欢迎您!"
Text1. Text = Text2. text
```

不合法语句：

```
99 = Total
Gcd $ = 早上好!
Text1. Text = Visual Basic 6. 0 欢迎您!
```

合法赋值语句是显而易见的，不合法语句：99 = Total 违反了左边不能为常数的赋值原则；Gcd $ = 早上好！以及 Text1. Text = Visual Basic 6. 0 欢迎您！中

Gcd $ 表明是一个字符串变量，Text1. Text 是文本框 Text1 的 Text 属性，它们接受字符串常量值，而字符串常量必须用引号括起来。没有引号就不是字符串，这违反了数据类型一致的赋值原则。

通过属性窗口设置属性值，是静态改变对象属性取值的方法。通过赋值语句，则可以在程序执行时动态改变对象的属性取值。给对象的属性赋值，一般格式为：

对象名．属性名 = 属性的新值

### 2.1.4 注释语句和结束语句

开发出来的应用程序，如果没有注释，会影响它的可读性，过一段时间再去阅读，就会出现困难；另外，开发规模较大的应用程序时，往往是由一个团队来开发，由若干个人分工编写程序，这样就更需要加上注释，以便能互相阅读理解。因此在编写程序代码时注释是必不可少的。

在 VB 里，注释的写法是在注释文字前加上一个单引号（'），注释语句是非执行语句，只是对有关内容作解释。习惯上在程序代码的开头加上若干注释语句，以便给出标题和程序总体功能说明；对执行语句作注释则把注释语句在该执行语句的后面，用以解释语句的作用。

在 VB 里用 End 作结束语句。在执行了 End 语句后，所有被打开的窗口以及程序都会从内存中清除。在 VB 中结束语句还有其他的形式：

- ✧ End Sub　　结束一个过程
- ✧ End Function　　结束一个函数过程
- ✧ End If　　结束一个 If 判断语句
- ✧ End Type　　结束一个自定义的数据类型
- ✧ End Select　　结束一个 Select Case 分支语句

## 2.2 基本数据类型

描述客观事物的数字、字符以及所有能输入到计算机中并被计算机程序加工处理的符号的集合称为数据。数据有多种类型，每种类型都有一定的数据结构特点。在 VB 中，提供了许多基本数据类型，同时用户也可以自定义数据类型。表 2.1 列出 VB 所允许使用的基本数据类型。

表 2.1 VB 基本数据类型

| 数据类型 | | 类型声明符 | 存储位数 | 取值范围 |
|---|---|---|---|---|
| Numeric（数据型） | Byte（字节型） | 无 | 8 位（1 字节） | 0～255 |
| | Integer（整型） | % | 16 位（2 字节） | －32768 ～ ＋32767 |
| | Long（长整型） | & | 32 位（4 字节） | －2147483648 ～ ＋2147483647 |
| | Single（单精度浮点型） | ! | 32 位（4 字节） | －3.401E45 ～ －1.401E－45<br>＋1.401E－45 ～ ＋3.403E38 |
| | Double（双精度浮点型） | # | 64 位（8 字节） | －1.987E308 ～ －4.941E－324<br>＋4.941E－324 ～ ＋1.987E308 |
| | Currency（货币型） | @ | 64 位（8 字节） | －922337203685477.5808<br>＋922337203685477.5808 |
| String（字符串长度） | String（定长）<br>String（变长） | $ | 字符串长度<br>字符串长度加 10 | |
| Data（日期类型） | | 无 | 8 字节 | 公元 100 年 1 月 1 日 ～ 公元 9999 年 12 月 31 日 |
| Boolean（布尔类型） | | 无 | 16 位（2 字节） | True（－1）或 False（0）<br>默认值为 False |
| 对象型 | Object | 无 | 32 位（4 字节） | 任何对象的引用 |
| Variant（变体类型） | Variant<br>Variant | 无 | 16 字节<br>字符串长度加 22 | 任何数值 |

## 2.2.1 数值数据类型

VB 支持 6 种数值数据类型，包括字节型、整型、长整型、单精度浮点型、双精度浮点型及货币型。

### 1. 字节型数据类型（Byte）

Byte 数据类型在内存中占 1 个字节，用来表示一个无符号整数，范围是 0～255。

### 2. 整数数据类型（Integer）

整数数据类型在内存中用 2 个字节（16 位），范围是－32768 ～ ＋32767 中不带小数点的数据，在－32678 ～ ＋32767 中的数字在尾部加一个“%”符号也表示整型数据，如 135%，－45% 等。

### 3. 长整型数据类型（Long）

长整型数据类型是超过－32768 ～ ＋32767，而在－214743648 ～ ＋2147483647 不带小数点的数。一个长整型数在内存中占 4 个字节（32 位）。在－214743648 ～ ＋2147483647 的数字尾部带一个“&”符号也表示为长整数。

### 4. 单精度数浮点类型（Single）

单精度数是带小数点的实数，有效值数位为 7 位。在内存中用 4 个字节（32 位）存放一个单精度数。通常以指数形式（科学记数法）来表示，以“E”或“e”表示指数部分。

5. 双精度数据类型（Double）

双精度数据也是带小数点的实数，有效数位为 15 位。在内存中用 8 个字节（64 位）存放一个双精度数。双精度数通常以指数形式（科学记数法）来表示，以“D”或“d”表示指数部分。

6. 货币型数据类型（Currency）

货币类型是为计算货币而设置的定点数据类型，它的精度要求高，规定精确到小数点后 4 位，一般的数值型数据在计算机内是通过二进制方式进行运算的，因而有误差，而货币型数据用用十进制方式进行运算的，所以具有比较高的精确度，取值范围在 -922337203685477. 5805 ~ +922337203685477. 5805。

## 2. 2. 2 字符串数据类型（String）

字符串类型用来定义一个字符序列，字符由字母、数字及其他符号构成，用双引号括起。在内存中一个字符用一个字节（8 位的二进制）来存放。

## 2. 2. 3 日期类型（Data）

日期类型用来表示日期、时间，在内存中一个日期型数据用 8 个字节来存放。在计算机系统中的系统时钟以双精度数存储，每毫秒更新一次。日期范围从公元 100 年 1 月 1 日到公元 9999 年 12 月 31 日，时间从 0：00：00 到 23：59：59，日期常数要用“#”开始并且用“#”结束，如#05/01/2004#，表示 2004 年 5 月 1 日，不能写成“05/01/2004”，此为字符串格式常量的表示方法。

## 2. 2. 4 通用数据类型（Variant）

通用数据类型也称变体类型，是一种通用的、可变的数据类型，它可以表示存储上述任何一种数据类型，根据所进行的操作自动进行数据转换。

但是这种灵活的操作方式也会因它的数据不明确性而产生不易觉察的错误，其次这种数据类型要占用大量内存，建议初学者不要使用该数据类型。

假设定义 var1 为通用型变量，格式为：

```
Dim Var1 As Variant
```

在变量 Var1 中可以存放任何类型的数据，如：

```
Var1 = 12345    （存放一个数字）
Var1 = "BASIC"（存放一个字符串）
Var1 = #05/01/2004#（存放一个日期型数据）
```

当一个变量未定义类型时，系统自动将该变量定义为 Variant 类型。不同类型的数据在 Variant 变量中是按其实际类型存放的（例如将一个整数赋给 Var1，在内存区中按整型数方式存放），用户不必作任何转换，转换的工作由系统自动完成。

VB 提供一种 VarType 函数，用来测试一个 Variant 变量的实际数据类型。VarType 函数的返回值是一个数值，其含义见表 2.2。

表 2.2 VarType 函数返回值的类型

| VarType 函数值 | 数值类型 | VarType 函数值 | 数值类型 | VarType 函数值 | 数值类型 |
|---|---|---|---|---|---|
| 0 | 空 | 3 | 长整型 | 6 | 货币型 |
| 1 | Null | 4 | 单精度 | 7 | 日期型 |
| 2 | 整型 | 5 | 双精度 | 8 | 字符串型 |

例如：

```
Dim Var1 as variant
Int1 = 123
Long1 = 186&
Single1 = 12.6!
Double1 = 34.5
Str1 = "abcd"
Cur = 8886@
Da = #10/21/1997#
Print vartype(Var1), vartype(Int1), vartype(Long1), vartype(Single1)
Print vartype(Str1), vartype(Cur), vartype(Double1),vartype(Da)
```

Var1 被定义成为 Variant 型变量，程序中未对它赋值，其他各变量均未声明为何种类型，也一样按 Variant 型对待。分别对 7 个 Varint 型变量赋值，然后用 Vartype 函数测试这 8 个变量的实际类型，可以从输出结果中看到它们的实际数据类型。

运行此段程序，输出结果如下：

```
0(未赋值),2(整型),3(长整型),4(单精度型)
8(字符串型),6(货币型),5(双精度型),7(日期型)
```

## 2.3 常量与变量

### 2.3.1 常量与变量

计算机在处理数据时，必须将其装入内存。在高级语言中，需要对存放数据的内存单元进行命名，通过内存单元名来访问其中的数据，变量或常量就是被命

名的内存单元。取值不变的量称为常量，取值可以改变的量称为变量。

在程序运行过程中，其值不能被改变的量称为常量。VB 常量有普通常量、符号常量、系统常量 3 种。普通常量一般从字面上区分其数据类型；符号常量是用一个字符串代替程序中的常数；系统常量是系统定义的常量，存放于 VB 系统库中。

在设计应用程序过程中，为常量、变量起名，必须遵循在上一节中所说的标识符命名原则：

- 名字的第一个字符必须是英文字母或汉字；
- 名字只能由字母、数字、下画线或汉字组成；
- 名字中最多可包含 255 个字符；
- 不用 VB 的保留字和函数作为变量的名字；
- 在同一个模块中名字应该是唯一的，同一模块是指能够引用此量的区域，如一个过程、一个窗体等。

下面是一些正确的和错误的名字例子。

正确的名字：

ApplePrice

CmdStartStop_Flag

X1

Ye23p

错误的名字：

8Base　第 1 个字符不是字母

Base. base 名字中使用了不允许出现的句号

因为在 VB 中不区分英文字母的大小写，即 VB 把 BASE，Base，base 都视为是代表一个变量的名字。在正确命名的前提下，引入面向对象程序设计的观点，名字最好应体现一个变量的作用和数据类型，保持程序中前后风格的统一，这样可以增强程序的可读性和可维护性。例如为体现数据类型，可在变量命名时，为它们加上 L 类型前缀，如下所示。

| 数据类型 | 标准前缀 |
|---|---|
| Integer（整型） | Int |
| Long（长整型） | Long |
| Single（单精度浮点型） | Sng |
| Double（双精度浮点型） | Db1 |
| Currency（货币型） | Cur |
| Byte（字节型） | Byt |
| String（字符串型） | Str |
| Boolean（布尔型） | Bln |
| Date（时期型） | Date |
| Variant（变体型） | Vart |

同其他语言一样，变量命名时不能使用保留字。例如在 VB 中对象的属性名（Caption，FontBold，…）、运算符（AND，OR，…）、方法名（move，print，…）等，如果把它们作为变量名加以使用，VB 就会给出错误信息。

## 2.3.2 变量说明语句

在使用变量前一般都要对变量进行说明，指定该变量的数据类型，这样系统才能为这个变量分配适量的存储位置。

### 1. 用类型定义符进行变量说明

VB 对常用的数据类型设有类型定义符。把类型定义符放在一个变量的末尾，就可以标明该变量属于什么数据类型，表 2.3 列出类型定义符与数据类型之间的对应关系。

**表 2.3 类型定义符与相应的数据类型**

| 类型定义符 | 数据类型 |
|---|---|
| % | Integer（整型） |
| & | Long（长整型） |
| ! | Single（单精度浮点型） |
| # | Double（双精度浮点型） |
| @ | Currency（货币型） |
| $ | String（字符串型） |

例如：int1% 表示 int1 是一个整型变量，dob1#表示 dob1 是一个双精度浮点型变量。str1 $ 表示 str1 是一个字符串型变量。

### 2. 用 Dim 语句进行变量说明

用 Dim 语句可以对过程级窗体级的变量说明，一般格式是：

DIM < 变量名 > As < 数据类型 >

例如：

```
Dim Var1 As Integer          把名为 Var1 的变量定义为整型。
Dim Total As Double          把名为 Total 的变量定义为双精度浮点型。
Dim BookName As String       把名为 BookName 的变量定义为变长字符串型。
Dim MyName As String * 10    把名为 MyName 的变量定义为定长(10 个字节)字符串型。
```

VB 虽然提供通用数据类型，从表面上看不定义数据类型，系统将根据所进行的操作自动进行数据转换。这并不表明使用一个变量之前不要对它进行说明。事实上在大多数情况下，不对所使用的变量进行必要的说明，应用程序将不能得到正确的运行，甚至无法运行。因此在使用变量之前应先对它说明，特别对初学者来说更应如此，对变量应该遵循先定义再使用，并养成良好的编程习惯。

不定义变量在编写程序代码时容易发生变量名错误。假如程序设计员把已有的一个变量名写错了，系统是无法分辨是采用了隐式说明的新的变量，还是程序设计员把已有的一个变量名写错了。因此对 VB 系统来说是作为一个新的变量名处理。

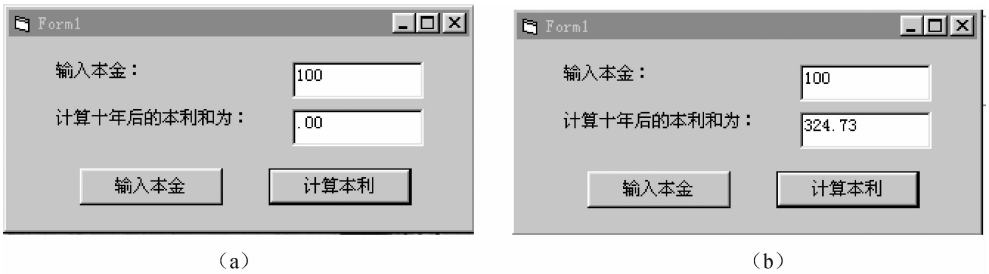
【例 2.1】如下例用来计算存款的本利，窗体上安放两个按钮，一个用来“输入本金”，另一个用来“计算本利”，窗体上对象的基本属性设置为：

| 对象 | Name | Caption | AutoSize | Alignment | BorderStyle | Appearance |
|---|---|---|---|---|---|---|
| 窗体 | form1 | 计算本利 | — | — | — | — |
| 标签 | label1 | 你的本金是： | true | — | — | — |
| 标签 | label2 | 空 | — | 2 | 1 | 0 |
| 标签 | label3 | 十年的本利是： | true | — | — | — |
| 标签 | label4 | 空 | — | 2 | 1 | 0 |
| 命令按钮 | command1 | 输入本金 | — | — | — | — |
| 命令按钮 | command2 | 计算本利 | — | — | — | — |

为 command1 和 command2 编写 Click 事件过程代码如下：

```
Private Sub command1_Click ( )
   bj = inputbox( " 请输入你的本金数" ," 输入框" ,0)   ' 输入本金
   label2. caption = bj
End sub
Private sub command2_Click ( ) ' 计算本利
For t = 1 to 10
I = bj * 0. 125
bj = bj + I
Next t
label4. Caption = Format ( bj ," ####. 00" )
End sub
```

这里，command1_ Click 代码中用到了一个变量 bj；command2_Click 代码中用了一个变量 t，还引用了 command1_Click 代码中的变量 bj。如果把这个设计投入运行，结果如图 2.2（a）所示，出现了十年后连本带利全无的奇怪现象。




（a）　　（b）

图 2.2　运行结果

这一结果的出现是由使用的变量引起的。VB 在处理每一个变量时还遵循两个原则：一是任何变量都有自己的作用范围（过程、窗体、工程，习惯上我们把变量按它的作用范围分为过程级、窗体级、工程级变量），超出了作用范围，它的值是无意义的；二是一个变量一旦定义但未被赋值时，VB 就将自动为之设置初值而不像有些高级语言一样是一个随机数据，见表 2.4。

表 2.4 各种数据类型变量的初值表

| 数据类型 | 默认初值 |
|---|---|
| 数值型变量 | 0 |
| 字符串型变量 | 空字符串 " " |
| 布尔型变量 | False（或 0） |
| 日期型变量 | 上午十二点 |
| 变体型变量 | 空值（Emply），既不是 0，也不是空字符串" " |

上面例题改为如下先定义再使用就不会出错，其正确结果如图 2.2（b）所示。

```
Dim bj As Single
Private Sub command1_Click( )
    Dim str As String
    str = InputBox("请输入你的本金数", "输入框", 0)        '输入本金
    label2. caption = str
    bj = Val(str)
End Sub
Private Sub command2_ Click( )    '计算本利
  Dim i As Single
    For j = 1 To 10
       i = bj * 0. 125
       bj = bj + i
    Next j
label4. Caption = Format(bj, "####. 00")
End Sub
```

根据这样两个原则，在具体过程中说明的变量，只能在过程中使用，只能在该过程中起作用，窗体中的其他过程不能使用。在窗体程序代码“通用”项中说明的变量，就可以在窗体的所有过程中起作用了。

注意，在同一行内进行多个变量说明时，可以运用逗号将多个变量独立说明，但不能像 C 语言等高级语言一样互相借用。例如，把 x，y，z 三个变量在一行上说明为 Integer 型，要写成：

```
Dim x As Integer, y As Integer, z As Integer
```

而不能写成：

```
Dim x,y,z As Integer
```

后一种写法对 VB 来说，只把变量 z 视为是 Integer 型的，而变量 x 和 y 则成了 Variant（变体）型的了。又如说明语句：

```
Dim a As Integer,b,c As Currency
```

其意为变量 a 为整型的，变量 c 为货币型的，变量 b 则为变体型的。

### 2.3.3 静态变量说明语句 Static

如前所述，过程变量说明位于过程内部，可以用 Dim 语句完成对变量的说明。用 Dim 说明的过程级变量的特点是在退出过程时，这个变量就被清除，因此再一次调用这个过程时，变量不会保留上一次调用时的值，是动态变量。如果希望一个过程级变量能保留上一次调用的值，再一次调用该过程，变量的值保留并可以继续使用，那么就应该在过程内部用 Static 语句来对这一变量进行说明。用 Static 对过程级变量进行说明的一般格式是：

```
Static <变量名> As <数据类型>
```

用 Static 语句说明的变量，通常称为静态变量。

下面以窗体 form1 的 Click 事件过程为例，来理解 Dim 和 Static 说明语句的本质区别：

```
Private Sub Form_ Click( )
   Dim x As Integer
   Static y As Integer
      x = x + 1
      form1. Print Tab(5);   "x 的值是:";x
      y = y + 1
      form1. Print Tab(15);  "y 的值是:";y
End Sub
```

在代码中，x 用 Dim 语句说明是 Integer 型的，y 用 Static 语句也说明是 Integer 型。在此过程里，对 x，y 都是做加 1 操作。运行工程，用鼠标单击窗体后，窗体上将立即显示：

```
x 的值是:1
y 的值是:1
```

第 1 次用鼠标单击窗体后，x 和 y 的值都是 1（初始都取默认值 0），第 2 次用鼠标单击窗体，结果将显示：

```
x 的值是:1
y 的值是:2
```

由此可以看出用 Dim 和 Static 说明变量的不同。对于过程级变量 x 来说，在

第 1 次单击窗体引发了 Form_Click 事件过程而在窗体上输出信息“x 的值是：1”之后，变量 x 就不复存在了。因此在第 2 次单击窗体而又一次引发 Form_Click 事件过程时，x 仍是取默认值 0，因此它仍是在窗体上输出“x 的值是：1”。但对于过程级变量 y 来说，在第 1 次单击窗体引发了 Form_Click 事件过程而在窗体上输出信息“y 的值是：1”之后，虽然退出了该事件过程，但 y 却保存着原来的痕迹：1。因此在第 2 次单击窗体而又一次引发 Form_Click 事件过程，y 是以 1 的姿态进入的，所以执行语句“y = y + 1”，y 的值就是 2，在窗体上输出的信息就成为“y 的值是：2”了。

### 2.3.4 强制显式变量说明的语句 Options Explicit

在 VB 中，既允许用类型定义符或 Dim 等变量说明语句为变量进行显式说明，也允许在使用一个变量之前，先不对其说明（即隐式说明）。因为在编写程序代码时最难以察觉的错误之一就是拼错变量名，VB 一旦遇到一个新的变量名，它无法分辨这是一个采用了隐式说明的新的变量，还是程序设计员把已有的一个变量名写错了，VB 只能将其视为新变量名来对待。为了避免这种事情发生，VB 提供了一个强制要求进行显式说明变量的语句：

Option Explicit

该语句的功能是在它的作用范围内强制程序员一定要先定义再使用，只要 VB 发现未经显式说明的变量，就发出错误警告信息，不得通过，从而强制对所有变量都进行显式说明。

Option Explicit 语句的使用方法是，如果把它写在某个过程里，只对此过程起检查作用。如果希望在应用程序开发过程中，总要求进行显式的变量说明，其做法是选菜单“工具 | 选项”命令，弹出“选项”对话框，在“编辑器”页签中选中“要求变量声明”项（默认是不选）。这样设置完毕，以后每次新建工程时，VB 就会把 Option Explicit 自动写到模块级的变量声明部分，而无须再一次次输入该语句了。不过要注意的是，这一设置对正在进行设计的应用程序不起作用。

### 2.3.5 常量说明语句 Const

在编写程序代码过程中，经常会碰到需要反复使用同一个数值量。这时常用一个字符串来代替这个数值量，即我们说的符号常量。这至少有 3 个方面的好处：第一，记住一个有一定意义的词比记住一个很长的量要容易。例如，在程序里如要用到重量磅（pount）到公斤（kg）之间的换算，1 公斤 = 2.204 6 磅，那

么最好定义一个名字 PountToKg 来代表 2.204 6。这样，在程序里全部都用 PountToKg 来代替算值 2.204 6 不容易把 2.204 6 写错。第二，如果常数很长，出现频繁，那么一次次输入不但麻烦，且易出错。例如圆周率 3.141 592 6，习惯上用名字 PI 代之，即 PI = 3.141 592 6，这样可以减少很多的输入麻烦。第三，使用符号常量可以提高程序的可移植性，当另有程序功能相似时，只需修改其数值即可以，而不要大量的修改程序。

常量说明语句的一般格式是：

```
Const <常量名>=<常量>
```

就上面的例子而言，其对应的常量说明语句应该是：

```
Const PountToKg = 2.2046
Const PI = 3.1415926
```

<常量名>的命名规则与变量名的规则相同，但习惯上用大写来表示符号常量。常量的作用范围也与变量相同，窗体模块级常量的说明应该在程序代码窗口对象框的“通用”和“声明”栏里进行。在说明了常量之后，就可以用这些常量名来代替具体的常量值了。例如在说明了 Const PI = 3.141 592 6 后，就可以用下面的语句来计算半径为 5 的圆面积：

```
s = 5 * 5 * PI
```

## 2.4 运算符与表达式

表达式由常量、变量、运算符、函数和圆括号按一定的规则组成，通过运算后有一个结果，运算结果的类型由数据和运算符共同决定。本节主要介绍算术表达式、字符串表达式、关系表达式、逻辑表达式。数据的运算包括算术运算、关系运算、逻辑运算和连接运算。被运算的对象，如常量、常数、变量、函数等称为操作数，运算符是用来对操作数进行各种运行的操作符号，VB 提供了丰富的运算符，包括算术运算符、关系运算符、逻辑运算符和连接运算符 4 种。不同操作数通过运算符连接成一个整体，即为表达式。按照值的不同类型，可以把表达式分为算术表达式（得到数值）、字符串表达式（得到字符串）和逻辑表达式（得到逻辑值）3 种。

### 2.4.1 算术运算符

算术运算符是最为常见和常用的运算符，VB 提供了 7 个算术运算符，见表 2.5。在进行算术运算时一般要遵循以下 4 个方面的问题。

（1）每个符号占 1 个字符位置，所有符号都必须一个一个并排写在同一横线

上，不能在右上角或右下角写方次或下标。例如：$2^3$ 要写成 2^3。

（2）原来在数学表达式中省略的内容必须重新写上。例如：2x 要写成 2 * x。

（3）所有括号都用小括号（），括号必须配对。例如：3[x+2(y+z)] 必须写成 3 *（x+2 *（y+z））。

（4）要把数学表达式中的有些符号，改成 VB 中可以表示的符号。例如：要把 2πr 改为 2 * pi * r，其中字符常量 pi 要先定义。

**表 2.5 VB 的算术运算符**

| 优先级 | 运算符 | 运算 | 举例 | 结果 |
|---|---|---|---|---|
| 1 | ^ | 指数运算 | x=5^3 | 125 |
| 2 | - | 求负 | -5 | -5 |
| 3 | * | 乘 | x=2*40 | 80 |
| 3 | / | 浮点除法 | x=5/3 | 1.666 667 结果为浮点数 |
| 4 | \ | 整数除法 | x=20\6 | 2 |
| 5 | Mod | 取模（即取余） | x=20 Mod 3 | 2 |
| 6 | - | 减 | x=20-5 | 15 |
| 6 | + | 加 | x=20+8 | 28 |

### 1. 求指数运算符“^”

指数运算符可以计算一个数的乘方或开方。当指数大于 1 时做乘方运算，当指数小于 1 时做开方运算，下面的几个例子说明指数运算的过程。

2^2　　2 的平方，即 2 * 2，结果为 4

2^3　　2 的立方，即 2 * 2 * 2，结果为 8

4^0.5　4 的平方根，结果为 2

125^(1/3)　125 的立方根，结果为 5

10^-2　10 的平方的倒数，即 1/100，结果为 0.01。

### 2. 浮点除法运算符“/”和整数除法运算符“\”

浮点除法，就是通常意义下的除法，其结果为一个浮点数。所谓整数除法，即整除运算的结果只取整数，具体地说是先把有小数的操作数四舍五入后成为整型数或长整型数，然后执行除法运算，在结果中截取整数部分，不进行舍入处理。如下例所示。

10/4=2.5

10/4=2

25.63/6.78=3.7802359

25.63\6.78=3

3. 取模运算符 mod

整数除法是在结果中取其整数部分，而取模运算同是取其余数部分，所以取模运算实际是先把带有小数的操作数四舍五入后执行整除运算，取余数作为结果。如下例所示。

10/4　结果为2(商为2)

10.2 Mod 4　结果为2(余数为2)

7 Mod 4　结果为3

25.68 Mod 6.45　结果为2

## 2.4.2 关系运算符

关系运算符又称比较运算符，用来对两个表达式的值进行比较，运算结果只可能取逻辑值True(-1)或False(0)。VB把任何非0值都认为是“真”，但一般以-1表示真，以0表示假。关系表达式成立时，取值True；不成立时，取值False。由于是根据所给关系的成立与否产生出逻辑值（真或假）结果，所以由关系运算符连接而成的式子是一个逻辑表达式。VB提供了6种常用的关系运算符，见表2.6。

表2.6　VB的关系运算符

| 运算符 | 运算 | 举例 | 结果 |
|---|---|---|---|
| < | 小于 | N1 < N2 | -1 |
| <= | 小于等于 | N1 <= N2 | -1 |
| > | 大于 | N1 > N2 | 0 |
| >= | 大于等于 | N1 >= N2 | 0 |
| = | 等于 | N1 = 3，N2 = 5，N1 = N2 | 0 |
| < > | 不等 | N1 < > N2 | -1 |
| Like | 字符串匹配 | "aBBBa" Like "a * a" | True |
| Is | 对象引用比较 | | |

关系运算的规则如下：

(1) 当两个操作式均为数值型，按数值大小比较。

(2) 字符串比较，则按字符的ASCII码值从左到右一一比较，直到出现不同的字符为止。例如，表达式"a" > "b"的结果是False，因为a的ASCII(97)小于b的ASCII(98)。表达式"ABCDE" > "ABRA"的结果为False。

(3) 数值型与可转换为数值型的数据比较。例如，表达式29 > "189"是按数

值比较，结果为 False。

（4）数值型与不能转换成数值型的字符型数据不能比较。例如，表达式 77 > " sdcd" 不能比较，系统提示“类型不匹配”错误。

由关系运算符构成的表达式经常用在选择结构的条件语句或循环结构的循环条件语句中。

## 2.4.3 逻辑运算符

逻辑运算符也称布尔运算。用逻辑运算符连接两个或多个关系式，组成一个布尔表达式，结果为逻辑值：True 或 False。VB 的逻辑运算符有以下 6 种，参见表 2.7。逻辑非运算符 Not 为单目运算符（要求一个操作数），其他为双目运算符。由逻辑运算可以构造出复杂的条件运算。逻辑运算的一般原则如下。

（1）逻辑运算符的优先级不相同，Not（逻辑非）最高，但它低于关系运算，Imp（逻辑蕴含）最低。

（2）VB 中常用的逻辑运算符是 Not、And 和 Or。它们用于对多个关系表达式进行逻辑判断。例如，数学上表示某个数在某个区域时用表达式 $10 \leqslant X < 20$，在 VB 程序中应写成：

```
X >= 10 And X < 20
```

（3）参与逻辑运算的量一般都应是逻辑型数据，如果参与逻辑运算的两操作数是数值量，则以数值的二进制值逐位进行逻辑运算（0 当 False，1 当 True）。

表 2.7 VB 的逻辑运算符

| 运算符 | 运算 | 意义 |
|---|---|---|
| NOT | NOT A | 对 A 求反 |
| AND | A AND B | A 与 B |
| OR | A OR B | A 或 B |
| XOR | A XOR B | A 异或 B |
| EQV | A EQV B | A 等于 B |
| IMP | A IMP B | A 蕴含 B |

### 1. AND（逻辑与运算符）

逻辑与运算是双目运算，只有两个表达式的值都为 True 时，运算结果才为 True。设 A 和 B 代表两个逻辑表达式，那么 A AND B 是一个新的逻辑表达式，它的运算结果取值见表 2.8。

表 2.8　逻辑运算的取值表

| A | B | A AND B | A OR B | NOT A | A XOR B | A EQV B | A IMP B |
|---|---|---|---|---|---|---|---|
| True | True | True | True | False | False | True | True |
| True | False | False | True | False | True | False | False |
| False | True | False | True | True | True | False | True |
| False | False | False | False | True | False | True | True |

### 2. OR（或运算符）

逻辑或也是双目运算符，只要这两个表达式中有一个取值为 True，运算结果就为 True。若以 A 和 B 代表两个逻辑表达式，那么 A OR B 的运算结果取值见表 2.8。

### 3. NOT（逻辑非运算符）

非运算是一个单目运算符，如果该表达式的值为 True，那么运算结果为 False；如果该表达式的值为 False，那么运算结果为 True。

### 4. XOR（逻辑异或运算符）

异或运算是一个双目运算符，习惯上叫做逻辑加法，当参与运算的两个表达式取不同值时，运算结果为 True；当参与运算的两个表达式取相同值时，运算结果为 False。若以 A 和 B 代表两个逻辑表达式，那么 A XOR B 的值见表 2.8。

### 5. EQV（逻辑等价运算符）

逻辑等价运算也是一个双目运算符，如果这两个表达式取值相同，运行结果为 True；否则为 False。A EQV B 的值见表 2.8。

### 6. TIMP（逻辑蕴含运算符）

逻辑蕴含运算也是一个双目运算符，除了左侧逻辑表达式取真（True）值而右侧逻辑表达式取假（False）值的情况外，运算结果总取值 True。A IMP B 的值见表 2.8。

## 2.4.4　连接运算符

字符串运算符有两个：“&”“+”，它们的作用都是将两个字符串连接起来，合并成一个字符串。但使用“+”进行字符串连接时，有时会出现混乱，因此在进行字符串连接时，建议尽量采用运算符“&”。如"stu"&"dent"，运算结果为："student"；"1"+"2"的运算结果为"12"。

## 2.4.5 运算符的优先级

当一个表达式中出现多种运算符时，首先进行算术运算符，接着处理字符串连接运算符，然后处理比较运算符，最后处理逻辑运算符。用括号可以改变优先顺序，强令表达式的某些部分优先运行。括号内的运算总是优先于括号外的运算。对于多重括号，总是由内到外。在实际书写表达式时要注意以下几点。

（1）运算符不能相邻。例如，a + * b 是错误的。

（2）乘号不能省略。例如，x 乘以 y 应写成：x * y。

（3）括号必须成对出现，且均使用圆括号。例如，( -b + sqr(b * b - 4 * a * c))/(2 * a)。

（4）表达式从左到右在同一行上并排书写，不能出现上下标。

（5）要注意各种运算符的优先级别，为保持运算顺序，在写 VB 表达式时需要适当添加括号（），若用到库函数必须按库函数要求书写。运算的优先顺序（从高到低）可以归纳如下。

函数→算术运算（乘方→取负→乘/除→整除→求余→加/减）
→字符串运算符→关系运算符→逻辑运算符（Not→And→Or）

**【例 2.2】** 某校期末考试三门课，其中两门主课，按照学校规定，凡是满足下列条件之一者可当优秀学生：①三门总分在 270 分以上者；②两门主课均在 95 分以上者；③一门主课为 100 分，其他两门均在 80 分以上者。

设三门课程分别为 A，B，C，其中 A，B 为主课，优秀学生的逻辑表达式为

```
A + B + C > 270  or  A > 95  and  B > 95 or A = 100 and B > 80 and C > 80  or B = 100 and A > 80 and C > 80
```

**【例 2.3】** 求判断任一年份 Y 是不是闰年的逻辑表达式。

**解：** 判断闰年的 3 个条件如下。

①年份不能被 4 整除，则必不是闰年；

②年份能被 4 整除，但不能被 100 整除，则必是闰年

③年份能被 100 整除，且能被 400 整除，是闰年。

判断 Y 能否被 4 整除的方法：

```
Y Mod 4 = 0  ?
Y\4 = Y/4  ?
Int(Y/4) = Y/4  ?
```

是闰年的逻辑表达式为：

```
Y Mod 4 = 0  And Y Mod 100 < > 0  Or Y Mod 400 = 0
```

## 2.5 函 数

VB 提供了大量的内部函数（也称 VB 库函数），内部函数是 VB 中预先设置好的完成某一特定功能的函数，通常带有一个或几个参数，并返回一个返回值。通过使用内部函数，可以方便地完成各种复杂运算。VB 既为用户预定义了内部函数，供用户随时调用，同时也可以允许用户自定义函数过程。

函数的一般调用格式为：

＜函数名＞ （[ ＜参数表＞]）

VB 提供的内部函数大致可分为如下几类：数学函数、字符串函数、转换函数、日期和时间函数及其他函数。下面分别做简单介绍。

### 1. 基本数学函数

Sqr(x) 求平方根，如 Sqr(9) 值为 3；

Log(x)，x＞0 求自然对数，如 Log(10) 值为 2.3；

Exp(x) 求以 e 为底的幂值，如 Exp(3) 值为 20.086；

Abs(x) 求 x 的绝对值，如 Abs(－2.5) 值为 2.5；

Hex[$](x) 求 x 的十六进制数，返回的是字符型值，如 Hex[$](28) 值为"1C"；

Oct[$](x) 求 x 的八进制数，返回的是字符型值，如 Oct[$](10) 结果为"12"；

Sgn(x) 求 x 的符号，当 x＞0 返回 1；当 x＝0 返回 0；当 x＜0 返回 －1；

Sin(x) 求 x 的正弦值，Cos(x) 求 x 的余弦值，Tan(x) 求 x 的正切值，x 的单位是弧度；

Atn(x) 求 x 的反正切值，函数返回的是弧度值。

Rnd[(x)] 产生一个在（0，1）均匀分布区间的随机数，每次的值都不同；若 x＝0，则给出的是上一次本函数产生的随机数；x＜0，则返回相同的随机数；x＞0 或者缺省，产生序列中的下一个随机数。

如果要随机产生 1～N 的整数，可以用表达式 Int(N＊Rnd＋1) 来表达。例如，Int(10＊Rnd＋1)，就会产生出 1～10 的随机整数。

如果随机产生 a～b 的随机整数，可以用公式 Int(Rnd＊(b－a＋1))＋a 来表达。例如，Int(Rnd＊81)＋10，就会产生出 10～90 的随机整数。

### 2. 常用字符函数

ASC(X) 返回字符串 X 的第一个字符的字符码。如 Asc("A")，返回值为 65。

Chr(X) 返回字符码等于 X 的字符。如 Chr(65)，返回值 A。

Len(X) 计算字符串 X 的长度。如 Len("vb 教程")，值为 4。

Mid(X，n，m) 由 X 的第 n 个字符读起，读取后面的 m 个字符，如 Mid("abcdefg"，2，4)结果为"bcde"。

Replace(X，S，R) 将字符串 X 中的字符串 S 替换为字符串 R，然后返回。如 X = "VB is very good"，P = Replace(X，good，nice)，计算后 P = "VB is very nice"。

StrReverse(X) 返回 X 参数反转后的字符串。如 X = " abc "，p = StrReverse(X)，计算后 P = "cba"。

Ucase(X) 将 X 字符串中的小写字母转换成大写。

Lcase(X) 将 X 字符串中的大写字母转换成小写。

InStr(n，X，Y) 从 X 第 *n* 个字符起找出 Y 出现的位置。如 InStr("abc123def123"，"12")，那么返回值为 4。

### 3. 主要转换函数

转换函数用于数据类型或形式的转换，包括整型、实型、字符串型之间以及 ASCII 码字符之间的转换。常用类型转换函数如下所示。

Str(x) 将数值数据 x 转换成字符串。

Val(x) 将字符串 x 中的数字转换成数值。

Chr(x) 返回以 x 为 ASCII 码的字符。如 Chr(65) 的计算结果为"A"。

Asc(x) 给出字符 x 的 ASCII 码值，十进制数。

Int(x) 取小于等于 x 的最大整数。如 Int(－3.5) 的值为 －4；Int(3.5) 的值为 3。

Fix(x) 将数值型数据 x 的小数部分舍去。如 Fix(－3.5) 计算后结果为 －3。

CDate(x) 将有效的日期字符串转换成日期。如 CDate(#1990，2，23#) 结果为：1990－2－23。

CCur(x) 将数值数据 x 转换成货币型。

Round(x，N) 在保留 *N* 位小数的情况下四舍五入取整。

### 4. 常用测试函数

测试函数的返回值都是布尔值。常用测试函数如下。

IsArray(变量) 测试变量是否是一个数组。

IsDate(日期或字符串的表达式) 测试表达式值是否可以转换成日期。日期的范围介于公元 100 年 1 月 1 日与公元 9999 年 12 月 31 日之间。

IsEmpty(变量) 测试变量是否已经被初始化。

IsNull(变量) 测试变量是否为有效的数据。

IsNumeric(变量) 测试变量是否为可计算的数字型。

5. 日期时间函数

VB 提供了处理日期和时间的函数，日期和时间函数可以显示系统的日期和时间，提供某个事件何时发生及持续时间长短等信息。下面简单介绍几个取系统时间的函数。

Now( ) 读取系统当前日期、时间。

Date( ) 读取系统当前日期。

Time( ) 读取系统当前时间。

Timer( ) 计算午夜（即 0：00）起至当前系统时间所历经的秒数。

Year(日期字符串) 返回日期字符串中的年份。

Month(日期字符串) 月份。如 Month(Now( ))，将返回当前系统时间的月份。

Day(日期字符串) 日子。如 Day(Now( ))，将返回当前系统时间的日期。

Weekday(日期字符串，[指定一周的第一天]) 星期。

Hour(时间字符串) 返回时间字符串中的小时。

Minute(时间字符串) 分钟。

Second(时间字符串) 秒数。

## 2.6 VB 的输入/输出

### 2.6.1 VB 的输入

在 VB 中数据的输入通常用 InputBox 函数来实现。InputBox 函数运行时将弹出一个窗体，等待用户输入并按下按钮后返回之前在文本框输入的内容。函数使用格式如下。

InputBox(prompt[, title] [, default] [, xpos] [, ypos] [, helpfile, context])

参数说明：Prompt 必需的。显示说明对话框的作用的字符串表达式。prompt 的最大长度大约是 1 024 个字符，由所用字符的宽度决定。如果 prompt 包含多个行，则可在各行之间用回车符（Chr（13））、换行符（Chr（10））或回车换行符的组合（Chr（13）&Chr（10））来分隔。

Title 可选项。显示对话框标题栏字符串表达式。如果省略 title，则把应用程序名放入标题栏中。

Default 可选项。显示文本框中的字符串表达式，在没有其他输入时作为缺省值。如果省略 default，则文本框为空。

Xpos 可选项。数值表达式，成对出现，指定对话框的左边与屏幕左边的水平距离。如果省略 xpos，则对话框会在水平方向居中。

Ypos 可选项。数值表达式，成对出现，指定对话框的上边与屏幕上边的距离。如果省略 ypos，则对话框被放置在屏幕垂直方向距下边大约 1/3 的位置。

Helpfile 可选项。字符串表达式，识别帮助文件，用该文件为对话框提供上下文相关的帮助。如果已提供 helpfile，则也必须提供 context。

Context 可选项。数值表达式，由帮助文件的作者指定给某个帮助主题的帮助上下文编号。如果已提供 context，则也必须要提供 helpfile。

值得说明的是，如果同时提供了 helpfile 与 context，用户可以按 F1 来查看与 context 相应的帮助主题。

如 Message = "Enter a value between 1 and 3"，Title = "InputBox Demo"，Default = "1"。如果应用 MyValue = InputBox（Message，Title，Default，100，100），执行时将在（100，100）的位置显示对话框。如果用户单击“确定”按钮或按下“Enter”按键，则变量 MyValue 保存用户输入的数据。如果用户单击“取消”按钮，则返回一零长度字符串。

## 2.6.2 VB 中的输出

VB 的输出一般通过两种方式实现。

### 1. Print 方法

用来在窗体或其他控件上输出信息，格式如下：

Print［Spc(n) | Tab(n) |Expression Charpos］

其中，Spc(n) 可选项，表示输出空格的个数；

Tab(n) 可选项，用来指定输出数据的绝对列号；

Expression 可选，表示要打印的表达式；

Charpos 可选项，指定下个字符的插入点。

一般表达为：

［对象.］Print［输出项］［，| ;］

［对象］缺省为当前窗体。［输出项］缺省则打印一空白行，输出项之间可用分号或逗号分隔，使用“;”，其后的输出项紧接着前一个输出项输出；使用“,”，其后的输出项在下一个输出区输出。两个输出区默认相隔 14 列，若无“;”或“,”表示输出后换行。

### 2. MsgBox 函数

MsgBox 函数可以用来在对话框中显示消息，并等待用户单击按钮，然后返回一个整形的值，让程序了解用户单击的是哪一个按钮。语法格式如下：

MsgBox(prompt[, buttons] [, title] [, helpfile, context])

prompt 必要项。字符串表达式，作为显示在对话框中的消息。

Buttons 可选项。指定显示按钮的数目及形式。

Title 可选项。在对话框标题栏中显示的字符串表达式。

Button 参数说明中见表 2.9。Button 参数包含显示按钮数目及显示的图标。表示式为两项连接，如 vbOKOnly + VbCritical 或对应的值相加，如前面项等价于 0 + 16 = 16。例如：

```
r = MsgBox("Are you sure? ", vbYesNo)
```

表 2.9 Button 参数说明

| 常数 | 值 | 描述 |
|---|---|---|
| vbOKOnly | 0 | 只显示 OK 按钮 |
| VbOKCancel | 1 | 显示 OK 及 Cancel 按钮 |
| VbAbortRetryIgnore | 2 | 显示 Abort、Retry 及 Ignore 按钮 |
| VbYesNoCancel | 3 | 显示 Yes、No 及 Cancel 按钮 |
| VbYesNo | 4 | 显示 Yes 及 No 按钮 |
| VbRetryCancel | 5 | 显示 Retry 及 Cancel 按钮 |
| VbCritical | 16 | 显示  |
| VbQuestion | 32 | 显示  |
| VbExclamation | 48 | 显示  |
| VbInformation | 64 | 显示  |

如果不需要返回值，可使用没有括号的 MsgBox 语句，它的作用是弹出一个对话框用来提示用户，语法格式如下：

```
MsgBox prompt[, buttons] [, title]
```

例如：

```
MsgBox "输入有误！请重新输入", 3 + 32, "验证"
```

## 本章小结 <<<

本章着重介绍了编写 VB 程序所需要掌握的基本知识，要编写 VB 实用程序，必须熟练掌握 VB 语言的基础知识：主要包括数据类型、常量和变量、运算符、表达式、常用内部函数及编码规则；同时介绍了前期编写简单程序经常

要使用到的输入、输出方式。VB 提供了非常丰富的函数，使得编程变得非常轻松。

## 上机实训 <<<

### 实训二 数据类型、运算符和表达式

#### 一、实训目的

掌握变量的定义方法；熟悉 VB 数据类型，掌握常用内部函数的用法；掌握运算符和表达式的用法。

#### 二、实训内容

1. 静态变量的应用实训。
2. 练习 Mid、Len、InStr、Ucase 以及 Lcase 等字符串函数的使用。
3. 利用 Chr、Int 和 Rnd 函数随机生成大小写字母。
4. 生成指定范围的随机整数。
5. 在“立即”窗口练习内部函数、运算符和表达式的使用。

#### 三、实验步骤及参考代码

1. 静态变量的应用实训，通过实训理解静态变量的生命周期。实验步骤如下：

（1）设计界面，如图 2.3 所示，在窗体中只定义了一个命令按钮。

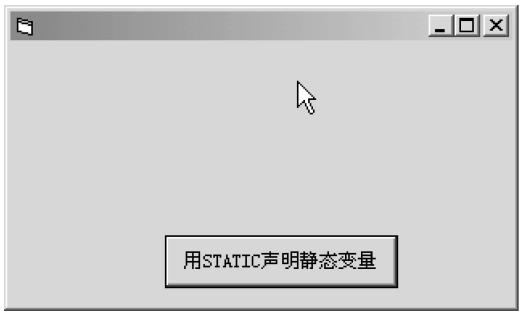



图 2.3 简单界面

（2）双击命令按钮，在代码窗口中编写事件过程代码。

事件过程代码如下：

```
Private Sub Command1_Click( )
    'A 和 I 变量定义成静态变量,程序执行完后,A 和 I 值保留到下一次,而 B 则不保留上次的值。
```

```
    Static A As Integer, I As Integer
    Dim B As Integer
    A = A + 1
    B = B + 1
    I = I + 1
    Print "第" + Str(I) + "次单击按钮,A =" + Str(A) + "  B =" + Str(B)
End Sub
```

（3）运行调试程序。

①单击保存工程图标存储工程，或执行“文件”|“保存工程”菜单。

②单击工具栏中央的小三角形按钮或执行“运行”|“启动”菜单。运行过程中可能出现各种语法错误或逻辑错误，需不断调试，直至完善。

③程序运行时，多次单击命令按钮。分析运行结果，Static 关键字声明的局部变量为静态局部变量，其值在应用程序整个运行期间一直存在，直到关闭程序为止，但只能在定义它的模块中使用；而 Dim 声明的变量只在过程的本次执行期间存在。

（4）运行调试程序。分析运行结果。

2. 字符串函数使用举例。通过输出字符串函数的值来学习掌握它的用法。

实验步骤：

（1）界面设计。（略）

（2）属性设置。（略）

（3）双击窗体，在代码窗口中编写事件过程代码。

添加事件过程代码如下：

```
Private Sub Form_Click()
  Print Asc("Apple")
  Print Chr(65)
  Print Str(123.5)
  Print Val("123.5")
  Print Len("王小红")
  Print Len(Space(6))
  Print Left("王小红", 2)
  Print Right("王小红", 2)
  Print Mid("王小红", 2, 1)
  Print LTrim("    abc")
  Print RTrim("abc    ")
  Print Trim("a bc    ")
End Sub
```

（4）运行调试程序。掌握基本字符串函数的用法。

## 本章习题 <<<

### 一、选择题

1. 下列选项中，合法的变量名是_____。

A. c% aaa　　B. sum_3　　C. Else　　D. 5persons

2. 有如下程序，该程序的执行结果是_____。

```
a = 20
b = 10
Print a > b
```

A. 1　　B. 0　　C. True　　D. False

3. 下列表达式不合法的是_____。

A. a = "123" + "abc"　　B. b = "123" & "abc"

C. c = 1 + 2\3　　D. d = [3 * (4 + 5) - 6]/7

4. 表达式 32/2^3 - 3 * 2^2 + 4^2 的值是_____。

A. 16　　B. 8　　C. 24　　D. 32

5. 逻辑数据类型由_____个字节组成。

A. 2　　B. 4　　C. 8　　D. 16

6. Timeover 是一个日期类型（Date）变量，以下赋值语句错误的是_____。

A. Timeover = #01/10/02#　　B. Timeover = #10：10：00#

C. Timeover = #01 - 02 - 04#　　D. Timeover = "1998,8,7"

7. 表达式 15 + 30 * 3/9 * 5 \ 5 mod 10 的值是_____。

A. 15　　B. 8　　C. 4　　D. 32

8. 以下常数中，_____占用存储空间最多。

A. 100　　B. -9.43E6　　C. -9.34D5　　D. 8989898

9. 下列选项中，为字符串常量的是_____。

A. 6/12/2001　　B. "6/12/2001"

C. #6，12，2001#　　D. 6，12，2001#

10. 有如下程序，该程序的执行后 b 的值是_____。

```
i = "12345"
j = "2345"
a = "10000"
b = i = j + a
```

A. 0　　B. 出错　　C. 12345　　D. False

11. 以下_____的执行结果是 0.6666667。

A. Dim x As Single
x = 2/3
Print x

B. Dim x As double
x = 2/3
Print x

C. Dim x As long
x = 2/3
Print x

D. x As integer
x = 2/3
Print x

12. 假设变量 boolVar 是一个布尔型变量，则下面正确的赋值语句是_____。

A. boolVar = 3 < 4　　B. boolVar = "True"

C. boolVar = . True.　　D. boolVar = #True#

13. 用于获取字符串 MyStr 最右端 3 个字符的函数是_____。

A. Right(MyStr, 3)　　B. RightTrim(MyStr, 3)

C. Rtrim(MyStr, 3)　　D. RightStr(MyStr, 3)

14. 执行语句 X = int(-5.6) + Abs(Fix(5.6)) 后，变量 X 的值是_____。

A. 1　　B. -1　　C. -11　　D. 0

15. 以下 VB 程序段的输出结果是_____。

```
A = Sqr(3) : B = Sqr(2) :C = A > B
Print C
```

A. -1　　B. 0　　C. False　　D. True

16. InputBox( ) 函数返回值的类型为_____。

A. 数值　　B. 字符串　　C. 遍体　　D. 数值或字符串

17. 语句 Print Sgn(100) 的输出结果为_____。

A. -1　　B. 0　　C. 1　　D. 100

18. 下面语句的输出结果是_____。

```
Print Format $(32548.5,"000,000.00")
```

A. 32548.5　　B. 32, 548.5

C. 032, 548.50　　D. 32, 548.50

19. 语句 PRINT"SGN(-5) ="; SGN(-5) 的输出结果为_____。

A. SGN(-5) = 0　　B. SGN(-5) = 1

C. SGN(-5) = -5　　D. SGN(-5) = -1

20. 能够产生 [10, 100] 随机整数的 VB 表达式是_____。

A. Int(Rnd(1) * 90) + 11　　B. Int(Rnd(1) * 91) + 11

C. Int(Rnd(1) * 90) + 10　　D. Int(Rnd(1) * 91) + 10

## 二、求表达式的值

1. 写出下列表达式的值。

①10 Mod 7

②7 mod 3 + 3^3/4 \ 5

③32/2^3 - 3 * 2^2 + 4 \ 3 - 6 mod 4

④"100" + "200"

⑤100 & "程序设计"

⑥(1 < 3) AND (5 < 7)

⑦NOT (2 < 4)

⑧NOT (5 < 8) OR < (2 < 6)

⑨(2 < 1) AND (10 < 20) or (11 < 10)

⑩NOT (11 > 12) AND (12 < 22) or (34 < 44)

2. 若 x = 5，y = 9，z = -1，下列各逻辑表达式的结果是什么？

①(x = 5)

②(x = 5) AND (x < y)

③(x < y)

④((x < 5) AND (z < x)) OR (y = 9)

⑤((x + z) >= y)

⑥NOT (y > z) AND (x + y > z)

⑦NOT (x + z < y) OR (y < z) AND (y + z < x)

⑧(x > z) XOR (x + y < z)

⑨(x + z > y) EQV (x < z)

⑩(y = 9) IMP (x + y) > z