# 第3章

# 数据类型、常量与变量

这里的数据指的是可以被计算机处理的信息。为了快速地对数据进行运算并有效地利用存储空间，Visual Basic 把各种不同的数据归纳为多种数据类型。每种数据类型都有类型名称，每种类型的数据占用一定数量的存储空间，可以表示的值也有一定的范围限制。

数据类型可以用来定义变量、数组、常量、过程的参数与返回值、类和结构体的数据成员等。

## 3.1 基本数据类型

### 3.1.1 数值型

数值型是数据类型中的一个大类，包括如表 3.1 所示的 11 种具体类型。

**表 3.1 数值型数据类型**

| 序号 | 类型名称 | 中文名称 | 字节数 | 可表示值的范围 | 别称 |
|---|---|---|---|---|---|
| 1 | SByte | 有符号字节型 | 1 | 整数，－128～127 | |
| 2 | Byte | 字节型 | 1 | 整数，0～255 | |
| 3 | Short | 短整型 | 2 | 整数，－32 768～32 767 | Int16 |
| 4 | UShort | 无符号短整型 | 2 | 整数，0～65 535 | UInt16 |
| 5 | Integer | 整型 | 4 | 整数，－2 147 483 648 ～ 2 147 483 647 | Int32 |
| 6 | UInteger | 无符号整型 | 4 | 整数，0～4 294 967 295 | UInt32 |
| 7 | Long | 长整型 | 8 | 整数，－9 223 372 036 854 775 808～9 223 372 036 854 775 807 | Int64 |
| 8 | ULong | 无符号长整型 | 8 | 整数，0～18 446 744 073 709 551 615 | UInt64 |
| 9 | Single | 单精度浮点型 | 4 | 实数，$-3.402\,823\times10^{38}$～$3.402\,823\times10^{38}$，6～7 位有效数字 | |
| 10 | Double | 双精度浮点型 | 8 | 实数，$-1.79\,769\,313\,486\,232\times10^{308}$～$1.79\,769\,313\,486\,232\times10^{308}$，14～15 位有效数字 | |
| 11 | Decimal | 实型 | 16 | 定点实数，＋/－79 228 162 514 264 337 593 543 950 335，29 个有效位数 | |

这 11 种数据类型都是用来表示数值的,前 8 种表示整数,使用二进制补码形式存储数据。在实际编程时,应根据具体用途决定应该选用什么类型来表示数值。表示范围越大、精度越高的数据类型所占用的存储空间也越大、运算速度越慢。

### 3.1.2 Char(字符型)、String(字符串型)

Char 型使用 2 字节保存 16 位编码,可以表示 0～65 535 之间的一个整数,也可以表示单个 Unicode 字符。Unicode 编码的前 128 个码位(0～127)对应于标准美式键盘上的字母和符号,与 ASCII 字符集(见附录 B)定义的码位相同;随后的 128 个码位(128～255)表示特殊字符,如拉丁字母、重音符号、货币符号以及分数。其余的码位(256～65 535)用于表示不同种类的符号,包括世界范围的各种文本字符、音调符号以及数学和技术符号。

String 为字符串类型。字符串是指由多个字符组成的有序队列,实际上是多个 16 位(2 字节)编码序列。字符串数据类型是专门用来存放文字信息的。字符串占用的内存大小根据其存储的字符多少而定。一个字符串可包含从 0 到将近 20 亿($2^{31}$)个 Unicode 字符。String 类型是所有 Visual Basic 数据类型中唯一占用内存大小与所存数据有关的数据类型,实际上是一种可变长度字符串类型。

无论是 Char 类型的两个字节,还是 String 类型的多个字节,存储的只是字符的"内码",也就是某个字符在字符集中的编号。字符的外观(划画、字体等)是字符在需要显示时(显示在显示器,或输出到打印机),使用字符内码从字体文件中检索其字模再描绘出来的,字符串中并不包括外观信息。

### 3.1.3 Boolean(逻辑型)

Boolean 类型(也称为"布尔型")的数据只可能有两个值:True(逻辑"真")和 False(逻辑"假"),用来表示"是"与"否"、"开"与"关"、"对"与"错"这类只有两种取值的情况。一般情况下,逻辑型数据占 2 字节的存储空间。

### 3.1.4 Date(日期时间型)

Date 类型(又名 DateTime 类型)可以表示日期与时间信息。Date 类型数据用 8 个字节来表示日期(公元 100 年 1 月 1 日～9999 年 12 月 31 日)和时间(12:00:00AM～11:59:59.9999999PM,即 0:00:00～23:59:59.9999999),精度为 100 纳秒。

> 除了上述这些类型之外,Visual Basic 的基本数据类型还包括通用类型:Object。另外枚举类型、结构体、类等自定义数据类型将在 3.5 节和第 8 章中介绍。

## 3.2 直接常量

常量(Constant)是指在程序运行过程中其值始终保持不变的量。常量有两种:直接常量与符号常量。下面是各种数据类型直接常量的表示法。

## 3.2.1　整型常量

### 1. 十进制表示法

整型常量的十进制表示法与人们的日常书写方法相同，下面是一些十进制整型常量：

0　　10　　－12　　2000　　－1234

默认情况下，Visual Basic 认为整型常量是 Integer 类型的常量，如果超出 Integer 类型表示范围的常量，认为是 Long 类型的常量。

### 2. 八进制表示法

以“&O”(字母 O)开头，后面接由 0～7 组成的八进制数。下面是一些八进制常量：

&O11　　&O123　　&O7777　　&O176340

分别等于十进制的 9、83、4095、64736。

### 3. 十六进制表示法

以“&H”开头，后面接由 0～9、A～F 组成的十六进制数。下面是一些十六进制常量：

&H11　　&HFF　　&HFFFF　　&HFF76340

分别等于十进制的 17、255、65535、267871040。

## 3.2.2　实数常量

实数常量可以使用日常的书写方法。如果整数部分或小数部分为 0，则可以省略这一部分，但要保留小数点。下面是几个实数常量：

3.14159　　0.23　　24.　　－.45　　－0.05

也可以用指数形式表示实数常量，用 mEn 来表示 $m\times10^n$，其中的 m 是一个整数常量或实数常量，n 必须是整数常量，m 和 n 均不能省略。例如：

1E2 表示 $1\times10^2$　　－.5E-2 表示 $-0.5\times10^{-2}$　　13.22E0 表示 $13.22\times10^0$

实数常量中的“E”可以写为小写“e”。在 Visual Basic 中，实数常量被认为是 Double 类型的。

## 3.2.3　字符串型常量

字符串常量必须使用两个英文的双引号“”把实际的文本括起来。双引号称为字符串的**“界定符”**，表示字符串的开始与结束。下面是几个字符串常量：

"你好!"、" "、"12.34"、"a0123"、"Let It Be"、""

字符串常量中可以包括任何可输入的字符，包括汉字、中文标点、英文、英文符号。空格也是合法的字符，上面的第二个字符串就是由空格组成的。如果两个引号之间没有任何字符，表示一个**空字符串**(上面的最后一个)。空字符串是特殊的字符串，下面的语句使用空字符串常量清空文本框的内容：

```
TextBox1.Text = ""              '清空文本框内容
```

因为双引号当作了字符串常量的界定符,所以如果要在字符串中包含双引号,就要使用两个连续的双引号表示一个双引号,例如,下面的语句在文本框中只显示一个双引号。

```
Text1.Text = "这是一个双引号: """
```

总之,一个字符串中的双引号应该成对使用。如果字符串中包括汉字的全角双引号,与普通字符相同,不必进行特殊处理。

### 3.2.4 逻辑型常量

逻辑型常量只有两个: True 和 False。注意,它们没有任何界定符。"True"与"False"不是逻辑值,而是字符串常量。

### 3.2.5 日期时间型常量

日期时间型常量表示一个确切日期与时间,使用“#”号作界定符。一般可辨认的表示日期时间的文本都可以作为日期时间型常量。如果只有日期部分,则时间部分默认为#00:00:00#; 如果只有时间部分,则默认日期部分为#0001/1/1#。例如下面 4 个常量都表示同一个日期:“2008 年 1 月 2 日”。

#1/2/2008#　　#2008-1-2#　　#Jan 2,2008#　　#January 2,2008#

下面是一些时间常量:

#12:00:00 PM#(中午 12 点)　　#12:00:00 AM#(午夜 12 点)

#8:20:20 PM#　　#2:00:00 PM#　　#2:14:02#

下面是两个日期时间常量:

#1/2/2008 2:14:02 AM#　　#11/12/2008 6:00:00 PM#

分别表示“2008 年 1 月 2 日凌晨 2 点 14 分零 2 秒”和“2008 年 11 月 12 日下午 6 点整”。

在“代码”窗口中输入日期时间常量时,Visual Basic 会自动转换为内部统一形式。1930—2029 年之间日期的年份,会被省略掉前两位数,如“08”表示“2008”年。时间值的表示采用 12 小时制(上午“AM”、下午“PM”)。

### 3.2.6 类型字符与类型符号*

表 3.2 列出了 Visual Basic 提供的类型字符和类型符号,将类型字符或类型符号放到常量后面可强制该常量为指定的类型。

下面的两个常量分别是长整型(Long)常量、短整型(Short)常量和整型(Integer)常量(尽管它们的数值大小是相同的,但占用内存的字节数不同):

10&　　10S　　10

下面是一个字符型常量,而不是字符串常量:

"A"C

表 3.2　类型字符与类型符号

| 数据类型 | 类型字符 | 类型符号 |
|---|---|---|
| Short | S | |
| Integer | I | % |
| Long | L | & |
| Decimal | D | @ |
| Single | F | ! |
| Double | R | # |
| UShort | US | |
| UInteger | UI | |
| ULong | UL | |
| Char | C | |
| String | | $ |

类型字符只能用于修饰常量，类型符号还可以用于变量的定义。例如，下面语句定义了一个 Single 类型的变量 d：

```
Dim d!
```

## 3.3　变　　量

变量(Variable)是程序运行过程中用来保存临时数据所占用的内存空间，它的值是可变的，所以也称为“内存变量”。程序通过变量名来操作变量，每个变量有一定的数据类型、作用范围，占用一定字节的内存空间。熟练地使用变量是学习编程的必经之路。

图 3.1 展示了一个变量从开辟内存开始，直到被从内存中清除为止，中间的这一段时间是被反复赋值和取值的过程。

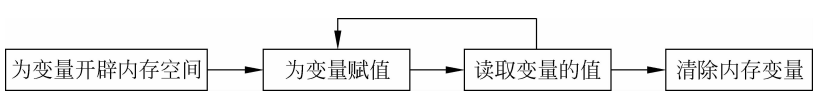


图 3.1　变量的生命周期

### 3.3.1　变量命名规则

Visual Basic 中的变量名要满足对标识符的要求，见 2.9 节。下面列出的是一些合法的变量名：

Abc　　Name　　intAge　　x12　　My_var1　　PI　　_X2

下面是一些非法的变量名：

12ab　　_　　ab.cd　　$MyVar　　Call　　x[1]　　a+b　　:

当变量名不符合规则时，Visual Basic 编辑器会显示错误信息。

与对象名同理，给变量命名时也应该注意它的描述性。每一种数据类型都有一个约定前缀(见附录 D)，使用前缀与英文单词或汉语拼音形成有意义的变量名。例如，可以使用 strUserName 作字符串类型变量名来保存用户姓名。

## 3.3.2 定义变量

定义变量也称为“声明变量”,是为变量指定变量名、数据类型以及作用域。一般情况下,定义变量同时为变量分配内存空间。定义变量的统一语法格式如下:

```
Public|Private|Dim|Static 变量名[As 数据类型名] [ = 初值]
```

其中的“Public|Private|Dim|Static”4 个关键字指定变量的作用范围(即作用域)。

> **变量的作用域**是指变量生效的范围,既能够对该变量赋值,又能读取该变量值的代码范围。在 Visual Basic 中,变量有四种作用域:块级、过程级、模块级和全局级。

下面分别介绍四种作用域变量的定义方法。

### 1. 块级变量

在过程中以下的控制结构中定义的变量为块级变量:Do…Loop、For[Each] …Next、If…End If、Select…End Select、With…End With、Try…End Try、While…End While,以后章节将介绍这些控制结构的用法。

```
Dim 变量名[As 数据类型名] [ = 初值]
```

块级变量只能在所定义的语句块中访问。因为这些语句块属于过程的一部分,所以块级变量的作用于小于过程级变量。

实质上,块级变量是特殊的过程级变量,原因有两个:

(1) 块级变量不能和所在的过程中过程级变量同名;

(2) 语句块运行结束时,执行所在过程的其他部分时,块级变量虽不能被访问,但仍占内存,仍保持其值,直到下一次再进入该语句块。只有所在过程结束时,块级变量才被从内存中清除。

### 2. 过程级变量

过程级变量又称为局部(Local)变量,定义于过程内、过程中所有的控制结构语句块之外,它的作用域是它所在的过程。也就是说,它在哪个过程中定义就只能在这个过程中使用。过程级变量只有在定义之后的语句才能使用。定义过程级变量的语句为:

```
Dim|Static 变量名[As 数据类型名] [ = 初值]
```

使用 Dim 关键字定义的过程级变量的特点是当所在过程执行完毕,变量就会消失,释放所占用的内存。下一次执行该过程时,重新给变量分配内存空间。

使用 Static 关键字定义的过程级变量被称为“静态变量”。静态过程级变量在程序启动时即被分配内存空间,程序结束时清除,所以在每次过程执行完毕后变量的值仍被保留,下一次该过程被执行时变量的值仍然可用。

下面定义了三个不同类型的过程级变量:

```
Dim intMyVar1 As Integer = 2          '定义整型变量 intMyVar1,赋初值 2
Dim blnSex As Boolean                 '定义逻辑型变量 blnSex
```

```
Static strName As String                    '定义字符串型静态变量 strName
```

### 3. 模块级变量

定义模块级变量的语句必须放在窗体模块(或标准模块、结构体和类模块)的内部、模块内所有的过程外面。定义模块级变量的语法格式为：

**Private|Dim 变量名[As 数据类型名] [ = 初值]**

其中，关键字 Private 和 Dim 是等效的。

模块级变量的作用域是所在的模块，也就是说，定义变量的这个模块中的所有过程代码都可以访问该变量。模块级变量在程序启动时被创建，程序结束时被清除。

### 4. 全局变量

全局变量也称为公有变量，是指在本程序的所有模块中都可以对其值进行存取的变量。全局变量定义的位置与模块级变量相同，使用的是 Public 关键字：

**Public 变量名[As 数据类型名] [ = 初值]**

与模块级、静态过程级变量相同，全局变量也是在程序启动时创建，程序结束时被清除。

### 5. 变量的作用域

由前面的叙述可知，定义变量时使用的关键字和定义变量的位置决定了变量的作用域。图 3.2 形象地说明了全局变量、模块级变量、过程级变量和块级变量作用域的覆盖范围。

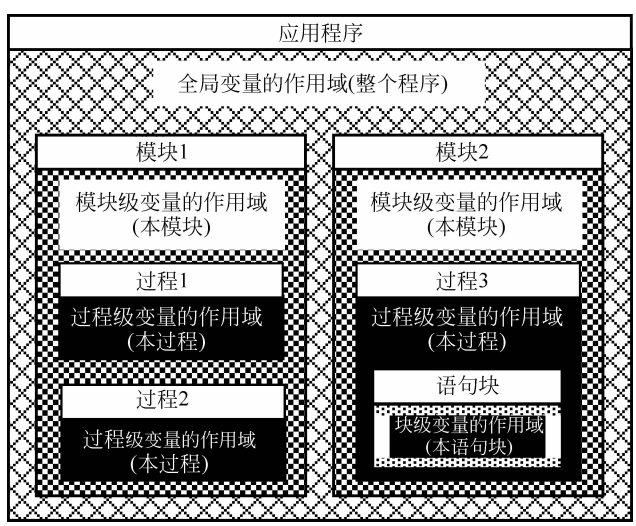


**图 3.2　变量的作用域**

程序中某个过程中的语句可以存取本过程中定义的过程级变量、所在模块定义的模块级变量、任意模块定义的全局变量，不能存取其他过程中定义的过程级变量和其他模块中定义的模块级变量。

请看下面的示意代码，第 1 行处不能定义任何变量，因为它不属于任何模块(在模块外)；第 3、4 行处可定义全局和模块级变量，因为在窗体模块内，在所有过程之上；第 6、7 行处可定义过程级变量，因为在过程内，控制结构语句块外；第 10、11 行处可定义块级变量，

只用于该 Select Case 结构中。第 16 行处也不能定义任何变量,它在模块之外;第 18、19 行可定义全局变量和模块级变量,这是标准模块;第 21、22 行定义过程级变量;第 24、25 行可定义块级变量。

```
1  '此处不能定义变量
2  Public Class Form1                '窗体模块
3     Public i As Integer            '此处可定义全局变量或模块级变量
4              …
5     Private Sub Button1_Click() Handles Button1.Click
6          Dim i As Integer          '此处可定义过程级变量
7              …
8          Select Case i
9              Case 20
10                    '此处可定义块级变量
11             …
12         End Select
13         …
14    End Sub
15 End Class
16    '此处不能定义变量
17 Public Module module1             '标准模块
18    Public i As Integer            '此处可定义全局变量或模块级变量
19             …
20    Private Sub sub1()
21             '此处可定义过程级变量
22             …
23         For i As Integer = 1  To 2
24             '此处可定义块级变量
25             …
26         Next
27         …
28    End Sub
29    …
30 End Module
```

实际上,可以不在模块开头或过程开头处定义变量,如可在两个过程之间或所有过程之后(如第 29 行处)定义模块级变量或全局变量,不影响程序运行。也可在第 13 行或 27 行处定义过程级变量,但只有在定义语句之后才能使用该变量。

**访问变量** 在过程中访问过程级变量、模块级变量、本模块定义的全局变量、标准模块中定义的全局变量时,可以直接使用变量名。如:

```
i = 1                    '为变量赋值
```

访问其他窗体模块中的全局变量时,必须在变量名前加模块名和“.”加以限定。例如,在窗体模块 Form1 的过程中给窗体模块 Form2 中定义的全局变量 i 赋值,必须使用如下的语句:

```
Form2.i = 2              '为其他窗体模块中定义的全局变量赋值
```

### 6. 一条语句定义多个变量

Visual Basic 允许使用一条语句定义多个变量。语法格式如下：

**Public|Private|Dim|Static 变量 1 [As 类型][,变量 2 [As 类型],…]**

如果相邻的变量数据类型相同，则可以省略前面的“[As 类型]”。例如，下面两条变量定义语句功能相同：

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

下面两条语句等价：

```
Dim a As Boolean, b As Date, c As Date
Dim a As Boolean, b, c As Date
```

### 7. 变量的初值与默认值

可以在定义变量的语句中直接给变量赋初值，如：

```
Dim a As Integer = 9
```

如果定义变量时未赋初值，则根据其数据类型不同，系统自动赋给变量默认值，以下是不同数据类型变量的默认值：

(1) 所有数值型变量的默认值为 0(或 0.0)；

(2) 逻辑型变量的默认值为 False；

(3) 日期时间型变量的默认值为“#0001/1/1　0:00:00#”；

(4) 字符串变量的默认值为 Nothing，不是空字符串；

(5) 字符型变量的默认值是内码为 0 的特殊字符；

(6) Object 变量的默认值为 Nothing。

## 3.3.3　变量的赋值与取值

变量是一个内存区域，程序代码通过变量名对其进行赋值和取值操作。给变量赋值使用的是如下的赋值语句：

**变量名 = 表达式**

上述赋值语句中，“=”是赋值号，而不是等于号；“表达式”指的是单个右值元素或用运算符连接的多个“右值元素”。赋值语句把赋值号“=”右边“表达式”的值写到左边“变量名”所代表的内存空间中，变量原有的值被覆盖，不复存在。**如果表达式值的数据类型与变量的数据类型不同，在赋值时会进行数据类型的默认转换，不能转换会引发错误**。

> 为了便于叙述，本书把对象属性、常量、变量、数组与集合元素、类或结构体数据成员、函数调用等具有“值”的项目称为“**右值元素**”，因为它们可以放在赋值号的右侧；多个右值元素构成的表达式也是右值元素。把可写对象属性、变量、数组与集合元素、类或结构体的可写数据成员等可以被赋值的项目称为“**左值元素**”，因为它们可以放在赋值号的左侧。

下面是一条定义变量的语句和一条赋值语句：

```
Dim a As Integer                    '定义变量 a,开辟 4 个字节内存空间,默认值为 0
a = 10                              '赋值语句,把整型常量 10 赋给变量 a,a 的值变为 10
```

要读取变量的值,只需将变量名写到表达式中。在计算表达式时,变量名即代表变量的当前值。读取变量的值时,并不会改变变量的值。

```
b = a                               '读取变量 a 的值赋给变量 b,变量 a 的值不变
a = b                               '读取变量 b 的值赋给变量 a,变量 b 的值不变
a = a + 1                           '把变量 a 的值与 1 相加后再赋给 a,即 a 的值增加 1
a = a + b - c                       '把变量 a 的值与变量 b 的值相加再减去变量 c 的值,
                                    '计算结果赋给变量 a,a 的值发生变化,b 和 c 的值不变
```

> 当给数值型变量赋一个超出其表示范围的值,会导致“溢出”错误。

值得注意的是,如果要把相同的值赋给两个变量,应该使用两个赋值语句,如：

```
a = 2 : b = 2
```

不能写成这样：

```
a = b = 2                           '错误,不能以这种方式将 2 同时赋给变量 a 和 b
```

上式也是一条赋值语句,但是意义不同,将在第 4 章讲解。

赋值号的左边只能是单个“左值元素”,不能是表达式。例如,下面的语句是错误的：

```
a + b = 2                           '错误,赋值号左边不能是表达式,无法给表达式赋值
```

**【例 3.1】** 使用过程级变量。

创建项目,建立如图 3.3 所示的窗体界面,窗体和两个按钮的对象名分别是 Form1、Button1 和 Button2。在“代码”窗口中编写如下的两个按钮的事件过程：

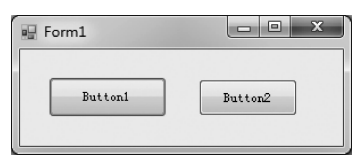


**图 3.3 窗体界面(例 3.1)**

```
1  Public Class Form1
2      Private Sub Button1_Click() Handles Button1.Click
3          Dim i As Integer            '动态过程级变量
4          i = i + 1
5          Button1.Text = i
6      End Sub
7
8      Private Sub Button2_Click() Handles Button2.Click
9          Static i As Integer         '静态过程级变量
10         i = i + 1
11         Button2.Text = i
12     End Sub
13 End Class
```

启动程序,连续多次单击两个按钮,看看程序是如何执行的。

连续单击按钮 Button1,它上面显示的数字一直是“1”。因为每次单击按钮,都会引

发 Click 事件，执行一次 Button1_Click 事件过程。首先，执行 Dim i As Integer 语句，为变量 i 开辟内存，这时它的初始值为 0；然后，执行 i＝i＋1 语句，变量 i 自增 1，它的值为 1；最后，执行 Command1. Text＝i 语句，按钮的 Text 属性被赋值为 1（由整数 1 转换为字符串“1”），按钮表面上显示“1”。每次过程执行完毕，自动释放非静态过程级变量所占用的内存空间，下一次执行该过程的情况与上一次完全相同，因此此按钮表面上的数字一直是“1”。

单击按钮 Button2 的情况则不同。因为 Button2_Click 事件过程中使用了 Static 关键字将变量 i 定义为静态过程级变量；第一次单击 Button2 之前变量 i 已被创建（默认值为 0），单击后增加 1，然后赋值给按钮 Button2 的 Text 属性显示在按钮表面；过程结束时变量 i 并不会被清除，下一次单击执行事件过程时不再重新开辟内存，它的值继续增加 1。所以，连续单击 Button2，会使它显示的数字以 1、2、3、……不断递增。整个程序结束时，静态过程级变量才会被从内存中清除。

注意，虽然 Button1_Click 和 Button2_Click 两个事件过程中都有名为 i 的过程级变量，但它们却是两个完全不同的变量（占用不同的内存空间），互不干扰，分别在所处的过程中起作用。

**【例 3.2】** 使用模块级变量。

创建与例 3.1 相同的窗体界面（如图 3.3 所示），在“代码”窗口编写以下程序代码：

```
Public Class Form1
    Dim i As Integer                    '模块级变量
    Private Sub Button1_Click() Handles Button1.Click
        Static i As Integer             '静态过程级变量
        i = i + 1
        Button1.Text = i
    End Sub

    Private Sub Button2_Click() Handles Button2.Click
        Static i As Integer             '静态过程级变量
        i = i + 1
        Button2.Text = i
    End Sub

    Private Sub Form1_Click() Handles Me.Click
        i = i + 1
        Me.Text = i
    End Sub
End Class
```

启动程序，连续多次单击窗体客户区和两个按钮，看看程序是如何执行的。

连续单击两个按钮控件，每个按钮上的数字都会以 1、2、3、……不断递增。连续单击窗体窗户区，标题栏上的数字也会以 1、2、3、……不断递增。两个按钮的事件过程中分别定义了名为 i 的静态过程级变量，能够维持各自的递增。因为 Form1_Click 事件过程中未定义任何过程级变量，因此过程中的 i 代表的是模块级变量 i。每次单击窗体，Form1_Click 事件过程即被执行一次，模块级变量 i 会增加 1，并显示到标题栏上。

**【例 3.3】** 为模块级变量赋初值。

创建与例 3.1 相同的窗体界面(如图 3.3 所示)。在“代码”窗口编写以下程序代码:

```
Public Class Form1
    Dim i As Integer                '模块级变量
    Private Sub Button1_Click() Handles Button1.Click
        i = i + 1
        Button1.Text = i
    End Sub

    Private Sub Button2_Click() Handles Button2.Click
        i = i + 1
        Button2.Text = i
    End Sub

    Private Sub Form1_Click() Handles Me.Click
        i = i + 1
        Me.Text = i
    End Sub

    Private Sub Form1_Load() Handles Me.Load
        i = 10
    End Sub
End Class
```

程序代码中有 4 个事件过程,都未定义局部变量,所以它们中的 i 都是指的同一个变量:即模块级变量 i。程序启动时,首先执行窗体的 Load 事件处理过程,为变量 i 赋初值 10。然后,无论单击的是按钮还是窗体,都会为 i 加 1,然后由被单击的对象显示。所示,单击窗体和按钮,显示的数值会相互攀升,而不是单独递增。

> 窗体的 Load 事件过程是为模块级、全局变量赋初值,为控件进行初始化的最佳位置。

## 3.3.4 变量的同名问题

### 1. 不允许同名的情况

一般情况下,在同一作用域内不能定义重名的变量。

(1) 同一个过程中不能定义同名过程级变量。

(2) 块级变量不能与所在过程的过程级变量同名,因为块级变量实质上是特殊的过程级变量。同一过程中的多个块中不能定义同名的块级变量。

(3) 同一模块中不能定义同名的模块级变量。

(4) 同一模块中不能定义同名的全局变量。

(5) 同一模块中定义的模块级变量和全局变量不能同名。

### 2. 允许同名的情况

(1) 不同的过程中可以定义同名的过程级变量和块级变量。

(2) 不同的模块中可以定义同名的模块级变量。

(3) 过程中可以定义与所在模块中模块级变量同名的过程级变量。

(4) 过程中可以定义与全局变量同名的过程级变量。

(5) 模块中可以定义与其他模块定义的全局变量同名的模块级变量。

(6) 不同的模块中可以定义同名的全局变量。

### 3. 变量同名时的情况

(1) 不同作用域的变量同名时,作用域小的变量会屏蔽作用域大的变量,即过程级变量屏蔽模块级和全局变量,模块级变量屏蔽全局变量。例如,在例 3.2 中,按钮事件过程中的变量 i 屏蔽模块级变量 i,过程中被访问的 i 实际上是过程级变量。

(2) 如果不同模块中全局变量同名,访问其他模块中定义的全局变量时应添加模块名进行限定(形式为"模块名.变量名")。访问本模块或标准模块中定义的全局变量时不必进行限定。如果本模块与标准模块中的全局变量同名,访问标准模块中的全局变量时也应加模块名进行限定。

(3) 当过程级变量与本模块定义的全局变量或模块级变量同名时,因屏蔽作用,在过程中直接使用这个变量名时,指的是过程级变量。如果要在此过程中使用本模块的全局变量或模块级变量,应使用关键字 Me 限定变量名,则可访问该全局变量或模块级变量。

(4) 如果本模块中的模块级变量与其他模块中的全局变量同名,可以在变量名前加模块名来访问其他模块的全局变量。

例如下面的程序,当单击按钮 Button1 时,它上面显示的是"10"。如果将"Me.a"改为"a",则显示"0"。

```
1  Public Class Form1
2      Dim a As Integer = 10          '模块级变量 a
3      Private Sub Button1_Click() Handles Button1.Click
4          Dim a As Integer           '过程级变量 a
5          Button1.Text = Me.a        '访问模块级变量
6      End Sub
7  End Class
```

### 4. 变量与对象的同名情况*

在窗体模块中,窗体的属性名、方法名、窗体模块中的事件过程和通用过程名、窗体上的控件名与模块级别定义的变量(全局和模块级变量)被认为是同一层次的,它们之间不能同名。所以,模块级变量、全局变量名不能与前面提到的所有的标识符相重复。

例如,在窗体模块中,不能定义一个名为 Text 或 Hide 的模块级或全局变量,因为它们是窗体的属性名和方法名。

如果因故必须定义与窗体的属性或方法(如 Text)同名的模块级或全局变量,可以使用 shadows 关键字:

```
Private Shadows Text As String
```

这样就不会出错了,Text 此时是模块级变量,窗体的 Text 属性被强制屏蔽了,程序中不能访问了。

过程中的局部变量(过程级变量和块级变量)比模块级低一级,会屏蔽模块级标识符。所以,与控件同名的过程级变量会屏蔽控件,如要访问该控件,应用 Me 关键字来限定。

例如,在窗体 Form1 上有文本框 TextBox1 与按钮 Button1。如果在 Button1_Click 事件过程中定义了一个名为 TextBox1 的过程级变量。那么,在这个事件过程中,语句:

```
TextBox1 = "您好"
```

是为过程级变量 TextBox1 赋值,而

```
Me.TextBox1 = "您好"
```

才是为控件 TextBox1 的默认属性 Text 属性赋值。

以下语句会产生语法错误,因为 Visual Basic 把 TextBox1 理解为变量,而不是控件,它并没有 Text 属性。

```
TextBox1.Text = "您好"
```

> 在定义变量时,尽量使用作用范围小的变量,也就是说:能使用过程级变量解决问题,就不使用模块级变量;能使用模块级变量完成任务,就不使用全局变量。

### 3.3.5 通用对象型(Object,Control)*

在 Visual Basic 中,可以说是“一切皆对象”,与各种对象相对应的是各种“类”,例如 Button、TextBox、Label 等是控件类,第 8 章将介绍自定义类(Class),第 9 章会介绍其他控件类。甚至数组、集合、字符串以及各种基本数据类型,在 Visual Basic 中都是用“类”实现的。

Visual Basic 提供了 Object 类型,它是通用对象类型,是所有类的基类;还提供了 Control 类型,它是通用控件类型,是所有控件的基类。

Visual Basic 允许定义对象型变量,对象型变量占用 4 字节的内存空间,保存的是对某个对象的引用,而不是保存对象本身(对象保存在内存的其他位置)。程序通过对象型变量可以间接地对它所引用对象的进行操作。

定义对象型变量的语句是:

```
Public|Private|Dim|Static 变量名 As Object|Control|对象类型
```

使用 Object 关键字定义的变量可以引用任何一种类型的对象。使用 Control 关键字定义的变量能够引用所有的控件对象。而使用一个具体的“对象类型”名(如 Button、TextBox 等)定义的对象型变量只能引用相同类型的对象。

给对象型变量赋值与给其他类型变量赋值方式相同:

```
对象型变量名 = 对象型表达式
```

赋值号“=”右边的“对象型表达式”可以是对象名,也可以是另一个对象型变量,还可以是返回对象型值的方法或函数。

例如,假设有一个名为 bntOK 的按钮,可以通过以下的三条语句来设置其 Text 属性:

```
Dim objButton As Object                    '定义对象型变量
objButton = bntOK                          '把按钮对象赋给对象型变量
objButton.Text = "OK"                      '通过对象型变量设置按钮对象的 Text 属性
```

对象型变量的默认值是一个特殊值：Nothing，表示未引用任何对象。可以通过给对象型变量赋 Nothing 值使之不引用任何对象。

```
objButton = Nothing                        '使对象型变量不再引用任何对象
```

> 与 Integer、Date、Boolean 等“值类型”不同，Object、Control 是典型的“引用类型”，对象型变量保存的不是一个具体的值，而是其他对象的“引用”，或其他对象的“指针”、“地址”。
>
> 在有些使用场合，如过程的形式参数或返回值类型，Object 还代表“所有类型”或“不确定的类型”。

## 3.3.6　类型转换

类型转换是指把数据从一种类型转换为另一种类型。类型转换发生在以下三种情况：

(1) 赋值时，如果左值元素的类型与被赋的值类型不一致，会发生类型转换。

(2) 计算表达式时，如果运算量的类型与运算符的要求不符，则进行类型转换。

(3) 在调用过程参数传递时，提供的参数与要求的类型不一致时，进行类型转换。

类型转换有可能扩大(widening)，也有可能收缩(narrowing)，扩大转换指的是从表示值范围小的类型转为表示值范围大的类型，而收缩转换正好相反。如由 Integer 类型向 Long 类型转换为扩大转换，由 Single 向 Long 转换为收缩转换。扩大转换永远不会出现溢出错误，并且总是成功的；而收缩转换必然伴有信息的丢失(数值精度降低)，并有可能失败。

### 1. 隐式转换

隐式转换，也称为“默认转换”，是指不使用专门的类型转换函数，按默认规则进行转换。

1) 数值型之间的转换

不同的数值类型之间可以互相转换。把整数转换为浮点类型时，存储格式转换，数值的大小不变。当把浮点数转换为整型数时，小数部分要“四舍五入”为整数，如果小数部分恰好是 0.5，则要向最近的偶数靠拢。

例如，下面语句将浮点类型的常量 4.56 赋给整型变量 i，在赋值过程会发生默认类型转换，变量的值变为 5：

```
i = 4.56                                   '浮点数转换为整数,变量 i 的值为 5
```

如果被赋的值为 4.5，则转换为整数 4：

```
i = 4.5                                    '转换时向最近的偶数靠拢,变量 i 的值为 4
```

如果被赋的值为 5.5，则转换为整数 6：

```
i = 5.5                                    '转换时向最近的偶数靠拢,变量 i 的值为 6
```

因为一种类型可以表示的值,另一种类型可能不能表示,所以在类型转换时,应注意避免出现“溢出”错误。

2) 字符串类型与数值型之间的转换

所有的数值都可以转换为字符串类型,反之则不然,只有字符串内容全部是数值信息的才可以转换为数值型。

例如,下面的语句向字符串变量 s1 赋一个数值:

```
s1 = 12.34                          's1 的值为"12.34"
```

下面的语句向整型变量 j 赋字符串值:

```
j = "1.2e3"                         'j 的值为 1200
```

如果无法进行默认转换,则会出错,程序不能运行。例如,下面的语句将包含字母的字符串赋给数值型变量 k,会出错。

```
k = "abc123"        '出错!无法转换
```

注意,空字符串不能赋给任何的数值类型,否则也会出现“无法转换”的错误。如:

```
k = ""              '出错!无法转换
```

3) 逻辑型值的转换

由逻辑型转换为有符号数值型的规则是:False 转换为 0,True 转换为-1。逻辑型转换为无符号数值型时,False 转换为 0,True 转换为该类型的最大值(对于 Byte 为 255,对于 UShort 为 65 535)。由数值型转换为逻辑型时:0 转换为 False,非 0 值均转换为 True。

逻辑型值转换为字符串时,True 和 False 分别转换为"True"和"False"。

把字符串转换为逻辑型时,只有"True"和"False"(或其他大小写形式,如"TRUE"和"FALSE")可以分别被转换为 True 和 False。其他任何字符串都不能转换为逻辑值,会出现“无法转换”错误。

4) 日期时间类型的转换

日期时间型值转换为字符串时,会按日期的短格式(可以在 Windows 控制面板的“区域设置”中设置)转换为相应的字符串。例如:

```
s2 = #2/1/2013 8:20:00#             '字符串变量 s2 的值为"2013-2-1 8:20:00"
```

表示有效时间的字符串可以转换为日期时间型值。例如:

```
d1 = "13:23:34"                     '日期型变量 d1 的值为#0001/1/1 13:23:34 #
```

日期时间型数据不能转换为数值型和逻辑型。

**2. 显式转换**

显式转换是指使用 Visual Basic 提供的类型转换函数进行转换(如表 3.3 所示)。使用这些类型转换函数可使程序的可读性增强,并且能进行强制类型转换,避免可能出现的歧义性。

**表 3.3　类型转换函数**

| 函数 | 转换为 | 函数 | 转换为 | 函数 | 转换为 |
|---|---|---|---|---|---|
| CBool( ) | Boolean | CByte( ) | Byte | CChar( ) | Char |
| CDate( ) | Date | CDbl( ) | Double | CDec( ) | Decimal |
| CInt( ) | Integer | CLng( ) | Long | CObj( ) | Object |
| CSByte( ) | SByte | CShort( ) | Short | CSng( ) | Single |
| CStr( ) | String | CUInt( ) | UInteger | CULng( ) | ULong |
| CUShort( ) | UShort | | | | |

显式转换的规则和隐式转换相同，不能隐式转换的，也不能显式转换。例如：

```
i3 = CInt(123.5)                    '整型变量 i3 的值是 124
f1 = CDbl("x1.2")                   '出错!以字母开头的字符串无法转换为数值
```

除了表 3.3 中列出的类型转换函数外，VB 还提供了其他的一些与类型转换有关的内部函数，如 Int、Fix、Val、Floor、Ceiling、Format 等，具体将在第 10 章中介绍。

**3. 不能进行转换的情况**

在数据类型默认转换过程中可能出错的情况可归纳为以下 4 类：

(1) 包含非数值字符(字母和符号)的字符串向数值型转换时出现“无法转换”错误。

(2) 非"True"或"False"的字符串向逻辑型转换时，出现“无法转换”错误。

(3) 非日期内容的字符串向日期型转换时，出现“无法转换”错误。如试图将"abc"转换为日期型。

(4) 转换时超出目标类型的表示范围，出现“溢出”错误。

## 3.3.7　Option 设置语句*

Visual Basic 中的 Option 语句实际上是一些开关，控制着所在代码文件(.vb)的相关设置。Option 语句共有 4 个，必须用在代码文件(.vb)开头、所有模块之前，设置只在所在文件中生效。

**1. Option Explicit On|Off**

此语句决定变量是否必须显式定义。默认设置为 On，要求变量必须先定义后使用。如果使用以下语句将此项设置为 Off，则变量可不用定义，直接使用，变量的数据类型因被赋值的类型而定或是 Object 类型(受 Option Infer 设置决定)。

```
Option Explicit Off
```

**2. Option Strict On|Off**

此设置的默认值为 Off，如将此项设置为 On，则默认数据类型转换时只允许取值范围小的类型向取值范围大的转换，反之则不可。如，当设置是

```
Option Strict Off
```

时，以下语句是正确的：

```
Dim x As Integer = 1.5
```

当设置是

```
Option Strict On
```

时,以下语句是错误的:

```
Dim x As Integer = 1.5
```

因为 1.5 默认为 Double 类型的常量,向 Integer 类型转换属于收缩转换。可用以下语句代替:

```
Dim x As Integer = CInt(1.5)
```

3. Option Infer On|Off

此设置的默认值为 On,如果在定义变量时未指定数据类型,则根据初值推断数据类型;如果设置为 Off,则没有指定数据类型的变量被定义为 Object 类型。如,当设置是

```
Option Infer On
```

时,以下语句定义的变量 x 是 Integer 类型:

```
Dim x = 5
```

否则,x 是 Object 类型。

4. Option Compare Binary|Text

此选项用于字符串的比较,如果设置为 Binary,按字符串字符的二进制内码比较,则"A"<"a";如果为 Text,则按字符不区分大小写比较,则"A"="a"。

如图 3.4 所示,这 4 个 Option 选项的默认值可通过"工具"菜单下的"选项"对话框修改。因默认的设置比较合理、比较常用,对于初学者不建议更改。

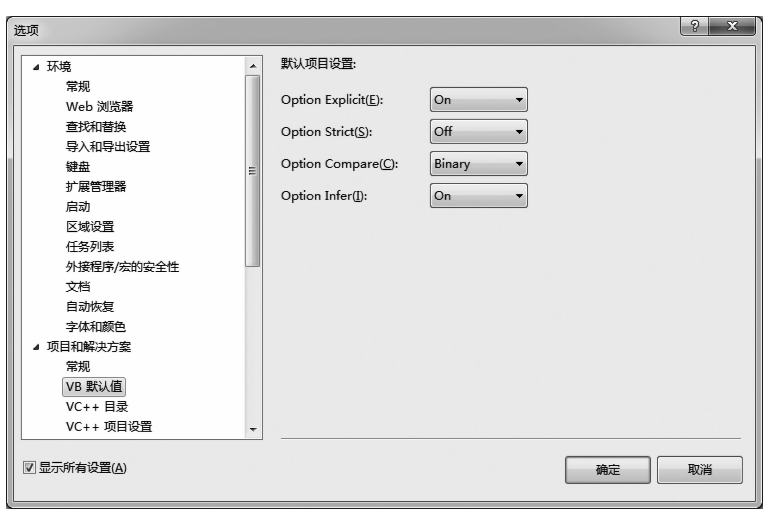


**图 3.4 Option 选项默认设置**

实际上，显式地定义每一个变量、明确指定变量的数据类型，赋值时了解发生的类型转换，是很好的习惯。为了省一点事，不定义变量或不指定数据类型是得不偿失的。Visual Basic 提供这几个选项更主要的目的是与老版本兼容。

## 3.4　符 号 常 量

符号常量是常量的一种，区别于直接常量。符号常量与变量相似，属于某一数据类型并有名称，先定义后使用。定义时必须指定符号常量的值，在程序运行过程中它的值不能改变（即不能再被赋值）。

符号常量与直接常量相比，具有如下优点：

（1）便于记忆与识别，可以使用具有描述性的名字替代抽象的值，如使用 PI 代表数值 3.141 592 6；

（2）便于程序的修改，如果要改变常量所代表的值，只需在定义常量的位置修改一次即可。

符号常量也有作用域，由定义时使用的语句和位置决定。定义符号常量的方法如下：

（1）过程级常量。在过程中定义，作用域为所在的过程。语法为：

**Const 常量名[As 数据类型名] = 表达式**

（2）模块级常量。在模块的通用声明段中定义，可在本模块的所有过程中使用。语法为：

**[Private] Const 常量名[As 数据类型名] = 表达式**

（3）全局常量。在标准模块的声明段中定义，程序的所有模块均可使用。语法为：

**Public Const 常量名[As 数据类型名] = 表达式**

常量名只要满足标识符的要求即可。为了与变量有所区别，一般可为常量名加上“con”前缀，或使用大写字母。

例如，下面语句定义了过程级常量 PI 来代替数值 3.141 593，并用来计算圆的面积：

```
Const PI As Single = 3.141593          '定义单精度浮点类型符号常量 PI
s = PI * r * r                         '通过半径 r 计算圆的面积，然后赋给变量 s
```

定义符号常量时，可以使用不包括函数和变量的常量表达式来赋值（表达式中可以使用算术运算符与逻辑运算符），也可以使用其他已定义的常量来赋值。例如：

```
Const con1 As Integer = PI + 2.5       '正确，如果 PI 是已定义的符号常量
Const con2 = i + Math.Sin(j)           '错误！不能使用函数 Sin 和变量 i、j
```

可以使用一条语句定义多个符号常量，方法与定义变量相似。

## 3.5　枚举常量与枚举类型

枚举（Enum）是批量定义相关常量的一种方法，被定义的枚举类型由多个枚举常量组成，枚举常量由一个有意义的名称代表一个整数。

定义枚举常量的语法格式为：

```
[Private|Public] Enum 枚举类型名 [As 数据类型]
    枚举常量 1[ = 整数值 1]
    枚举常量 2[ = 整数值 2]
    …
End Enum
```

“枚举类型名”和“枚举常量”的命名规划遵循 Visual Basic 对标识符的要求。

“数据类型”可以是 Byte、Integer、Long、SByte、Short、UInteger、ULong 或 UShort 等整型类型之一，默认为 Integer 类型。

例如，以下的语句定义了枚举类型 WeekDays，包括了 7 个枚举常量，对应一个星期中的七天。因未显式地指定整数值，默认地被从上到下依次赋予了 0、1、2、…、6。

```
Public Enum WeekDays
  Sunday
  Monday
  Tuesday
  Wednesday
  Thursday
  Friday
  Saturday
End Enum
```

即等价于：

```
Public Enum WeekDays
  Sunday = 0
  Monday = 1
  Tuesday = 2
  Wednesday = 3
  Thursday = 4
  Friday = 5
  Saturday = 6
End Enum
```

定义枚举常量之后，便可用该常量代替相应的整数值，如用 Sunday 代替数值 0，必要时(如有重名)可使用类型名修饰，如 WeekDays. Sunday。

定义枚举常量时，可显式地全部或部分为枚举常量赋值，可以是负整数。未被赋值的，按默认值从上到下自动赋值为 0、1、2、3、……例如，下面的定义 Saturday 将是 0，Monday 为 1，Tuesday 为 2、…… 允许两个以上的枚举常量拥有相同的值(如这里的 Saturday 和 Sunday 均为 0)。

```
Public Enum WeekDays
  Saturday
  Sunday = 0
  Monday
  Tuesday
  Wednesday
  Thursday
```

```
    Friday
    Invalid = -1
End Enum
```

使用枚举常量的好处是可以用一些便于记忆的、有特定意义的“单词”代替一些整数。在实际编程时甚至可以“淡忘”枚举常量实际代表的整数。

可以使用枚举类型来定义变量，如以下语句定义 WeekDays 类型的变量 workday：

```
Dim workday As WeekDays                    '定义枚举类型的变量
```

可以使用枚举常量为变量赋值：

```
workday = WeekDays.Friday                  '将枚举常量赋值给枚举类型变量
workday = WeekDays.Monday
```

需要指出的是，枚举类型的变量的值并不仅限于它的类型所包括的枚举常量，也可以被赋值为其他整数值，如：

```
workday = 20
```

除了自定义的枚举常量外，Visual Basic 中还有大量预定义的枚举常量，由系统提供。这些预定义的枚举常量，可以直接使用。在“属性”窗口中以列表形式提供可选项的属性，其值的类型一般均为枚举常量。如表 2.1 所示的窗体的 FormBorderStyle 属性值为 FormBorderStyle 类型的枚举常量，StartPosition 属性也是这样的。枚举常量还经常作函数的参数。

# 习　题　3

## 一、选择题

1. Short 类型的变量可存放的最大整数为(　　)。

   (A) 255　　(B) 256　　(C) 32 768　　(D) 32 767

2. 下面的 4 对数据类型中，哪一对所占的内存字节数相等？(　　)

   (A) Integer 和 Boolean　　(B) Integer 和 Single

   (C) Date 和 Single　　(D) String 和 Double

3. 下列数据类型中，占用内存最小的是(　　)。

   (A) Boolean　　(B) Byte　　(C) Integer　　(D) Single

4. 使用 Public Const 语句定义全局常量，该语句可以放在下列什么位置？(　　)。

   (A) 过程中　　(B) 所有的模块之外

   (C) 过程的语句块中　　(D) 模块之内，过程之外

5. 下面的(　　)关键字不能在模块级别使用。

   (A) Dim　　(B) Private　　(C) Public　　(D) Static

6. 下列(　　)数据类型的变量不能存放负值。

   (A) Integer　　(B) Single　　(C) Byte　　(D) Long

7. 下面(　　)不是字符串常量。

(A) "你好"　　(B) "　"　　(C) "True"　　(D) #False#

8. 下面(　　)数据类型表示数值范围最大。

(A) SByte　　(B) Byte　　(C) Integer　　(D) Short

9. VB 中的标识符最多不能超过的字符个数为(　　)。

(A) 1023　　(B) 12　　(C) 40　　(D) 255

10. 下列(　　)是日期型常量。

(A) "2/1/99"　　(B) 2/1/99　　(C) #2/1/99#　　(D) {2/1/99}

11. 下面(　　)赋值语句不能使字节型变量 b 在内存中的二进制位成为 00001111。

(A) b=15　　(B) b=1111　　(C) b=&HF　　(D) b=&O17

12. 下列哪一组语句会产生错误?(　　)

(A) Dim i As Integer : i=True　　(B) Dim s As String : s="123.4.5"

(C) Dim i As Integer : i="123.4"　　(D) Dim b As Boolean : b="Yes"

**二、填空题**

1. 下列数据类型的变量各占多少字节的内存?

Byte:　(1)　; Integer:　(2)　; Long:　(3)　; Single:　(4)　; Double:　(5)　。

2. 把整数 1 赋给一个逻辑型变量,则逻辑型变量的值为　(6)　。

3. 刚被定义尚未赋值的日期型变量的值为　(7)　;逻辑型变量的值为　(8)　;整型变量的值为　(9)　;字符串变量的值为　(10)　。

4. 对象型变量可以引用一个对象。使用 Dim objFirst As Object 语句定义一个对象型变量,如果要把名称为 bntFirst 的按钮赋予它,应使用　(11)　语句。

5. 在一条 Dim 语句中可以定义多个变量,如 Dim strVar,intVar,sngVar As Integer,则 strVar、intVar 与 sngVar 的数据类型分别是　(12)　、　(13)　和　(14)　。

6. 把逻辑值 True 赋给 UShort 型变量之后,此变量的值会变为　(15)　;把逻辑值 True 赋给 Short 型变量之后,此变量的值会变为　(16)　。

7. Object 类型变量的默认值为　(17)　。

8. 如果要在文本框 TextBox1 中显示"He said, "Good morning!"."(注:不包括外层的中文双引号,内层是英文双引号),则应使用以下的赋值语句: TextBox1.Text = 　(18)　。

9. 新建 Windows 窗体应用程序项目,窗体界面上添加一个文本框和一个按钮,对象名分别是 TextBox1 和 Button1。

在"代码"窗口中输入以下程序行:

```
1  Public Class Form1
2     Public int1As Integer                          '①
3     Dim int1 As Integer                            '②
4     Private int1 As Integer                        '③
5     Private Sub Button1_Click() Handles Button1.Click
6         Dim int1 As Integer                        '④
7         Static int1 As Integer                     '⑤
8         int1 = int1 + 1
```

```
        TextBox1.Text = int1
    End Sub
End Class
```

其中语句①～⑤同时只用一条。如果运行程序，连续单击 Button1 按钮三次，则使用语句①时文本框中显示的是____(19)____；使用语句②时显示的是____(20)____；使用语句③时显示的是____(21)____；使用语句④时显示的是____(22)____；使用语句⑤时显示的是____(23)____。

**三、判断题**

1. Dim、Static、Private、Public 都是可以用来定义变量的关键字。（　　）

2. 变量在程序执行过程中可以改变数据类型。（　　）

3. 使用 Static 语句定义的静态过程级变量，能在该过程的多次调用之间保持它的值，并且其他的过程也可以使用这个变量的值。（　　）

4. 在定义符号常量的语句中可以先不赋值，在以后赋值；但是，一旦被赋值便不能再赋新值。（　　）

5. 定义符号常量时可以使用表达式给常量赋值，但不能包含变量和函数调用。（　　）

6. 因为 Single 类型的变量可表示的范围大于 Long 类型的变量，所以 Single 类型占用内存空间大于 Long 类型。（　　）

7. Date 类型变量既可以只保存日期值，也可以只保存时间值，或同时保存日期和时间值。（　　）

8. 在同一个过程中不能定义同名的变量；在过程中不能定义与同一模块的模块级变量同名的静态过程级变量。（　　）

9. 给字符串变量赋一个长整型数会产生“溢出”错误。（　　）

10. 如果一个变量在定义时未赋初值，则它在被赋值之前没有值。（　　）

11. 一个应用程序的不同模块可以定义同名的全局变量，但是在一个模块中存取另一个模块中的全局变量时，应在变量前加模块名来限定。使用本模块中定义的全局变量在没有重名的情况下不用加模块名。（　　）

12. 如果 A 和 B 都是短整型变量，A 的值为 1，B 的值为 256，则变量 A 所占用的内存空间比变量 B 小。（　　）

13. 因为程序级和模块级范围不同，所以可以在同一个窗体模块中定义同名的全局变量和模块级变量。（　　）

14. 可以使用 Const 关键字定义枚举常量。（　　）

15. 同一个枚举类型中的两个枚举常量可以代表相同的整数值。（　　）

16. 给字符型变量赋值一个长字符串常量会出错。（　　）

**四、改错题**

下面是窗体 Form1 的 Click 事件过程，要实现从第二次单击开始起，每次单击窗体时，窗体均向右移动 10 个像素。其中有几处错误，请改正。

```
Public Class Form1
    Private Sub Form_Click() Handles Form1.Click
        Dim intLeft As Integer
```

```
4          intLeft = intLeft + 10
5          Me.Left = intLeft
6      End Sub
7  End Class
```

为什么第一次单击窗体时,窗体不向右移动,而是跳到屏幕左边附近。如果希望无论窗体的初始位置在什么地方,每次单击(包括第一次)窗体都向右移 10 个单位,程序应如何编写?

**五、找出合法的直接常量**

$-0.0$,.0,0.,$2^3$,1.2 * 10 ^ 3,log3,π,α,e,35.7°,""",""",""abc"",3+5,12.3e,5e+0,±&h007,3/2/99,&H123A,False

**六、找出合法的变量名**

3M,$x_2$,π,[,e,PI,OK,DIM,dim,+a,we$,_name,a+b,a_b