第 3 章　程序设计基本结构

计算机程序执行的控制流程只能由三种基本控制结构组成，即顺序结构、选择结构和循环结构。Visual Basic 虽然采用事件驱动，由用户激发某事件去执行相应的事件处理过程，这些处理过程之间并不形成特定的执行次序，但对每一个事件处理过程内部而言，又总包含这三种基本结构。事件驱动的方式无须程序员管理，但内部的语句流程是由程序员控制的。

3.1　顺序结构

所谓顺序结构，就是按照语句的书写顺序执行，即语句的执行顺序与书写顺序一致，但仅有顺序结构不能处理复杂的问题。本节介绍构成 Visual Basic 顺序结构中使用的基本语句、输入和输出操作、顺序结构程序设计的具体实例。

3.1.1　赋值语句 Let

赋值语句是程序设计中最基本、最常用的语句，是为变量和对象属性设置新值的最主要方法。其作用是把一个表达式的值赋给一个变量或一个对象的属性。

赋值语句的一般格式：

```
[Let]  <变量名>=<表达式>        '等号为赋值号
[Let]  [<对象名>.]<属性名>=<表达式>
```

赋值语句的功能：首先计算赋值号右边表达式的值，然后将此值赋给赋值号左边的变量或属性。

说明：

（1）关键字 Let 为可选项，通常都省略该关键字，“=”号称为赋值号。

（2）赋值语句具有计算和赋值的双重功能，它首先计算“=”号右边的表达式，然后把结果赋给“=”号左边的变量。

（3）给对象的属性赋值时，应指明对象名和属性名，系统默认的对象是当前窗体。在程序中如果给当前窗体的属性赋值，则可以省略对象名。

（4）赋值号跟数学中的等号具有不同的含义。例如，赋值语句“a=3”应读作“将数值 3 赋给变量 a”；赋值语句“x=x+6”，表示把变量 x 的当前值加 6 后将结果赋给变量 x，如果 x 的当前值为 2，执行这条语句后，x 的值为 8。

例如，下面两个语句的含义是不同的：

```
x=y   ' 将 y 的值赋给 x
y=x   ' 将 x 的值赋给 y
```

（5）赋值语句要求右端表达式计算结果的数据类型与左端变量的数据类型相容。如果用字符串的形式表示数值，则可以将字符串赋给数值变量，也可将数值赋给字符串变量。但如果把非数值形式的字符串赋给数值变量，将会在编译时出现错误。只有数据类型相容时才可以赋

值，例如，可以把计算结果为单精度的表达式赋给整型变量。

【例 3-1】赋值相容问题。

（1）设计如图 3-1 所示的用户界面。在窗体中建立一个命令按钮 Command1、两个标签 Label1 和 Label2、两个文本框 Text1 和 Text2。

（2）编写命令按钮 Command1 的 Click 事件过程。

```
Private Sub Command1_Click()
   Dim x As Integer
   Dim y As String
   y = "100.23"       ' 将数值形式的字符串赋给 y
   x = y              ' 将字符串变量 y 赋给整型变量 x
   y = y + 20
   Text1.Text = x
   Text2.Text = y
End Sub
```

（3）运行程序并单击“显示”按钮，其结果如图 3-1 所示。由于字符串常量“100.23”是数值形式，所以对整型变量 x 和字符串变量 y 都是赋值相容。

【例 3-2】交换两个标签中显示的文本内容。

分析：将两个标签比喻为两个瓶子 A 和 B，A、B 中分别装有不同颜色的液体。要交换两个瓶子中的液体，可以再取一个空瓶子 C，先将瓶 A 中的液体倒入空瓶 C，再将瓶 B 中的液体倒入 A，最后将瓶 C 中的液体倒入 B 中。用语句实现如下：

C=A:A=B:B=C

选择“新建工程”选项进入窗体设计器，建立一个命令按钮 Command1、4 个标签 Label1～Label4，如图 3-2 所示。

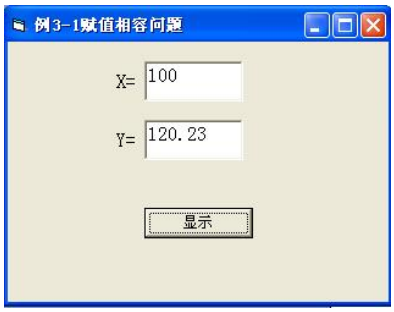


图 3-1　执行结果

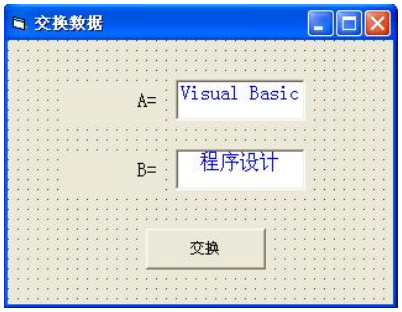


图 3-2　交换两个标签内容

编写代码如下：

```
Private Sub Command1_Click()
  C = Label2.Caption
  Label2.Caption = Label4.Caption
  Label4.Caption = C
End Sub
```

3.1.2　数据输出

1. 使用标签控件（Label）输出数据

标签（Label）是 Visual Basic 中最常用的输出文本信息的工具。在程序运行后，对于 Label

控件显示的文本用户不能直接修改。有些没有标题（Caption）属性的控件（如 TextBox）可以用 Label 标识。在 Label 中显示的文本是由 Label 控件的 Caption 属性控制的，该属性可以在设计时通过“属性”窗口设置或在运行程序时赋值。

【例 3-3】用标签输出多行文本。

（1）建立应用程序界面。选择“新建工程”选项，进入窗体设计器，添加一个命令按钮 Command1、一个标签 Label1，如图 3-3（a）所示。

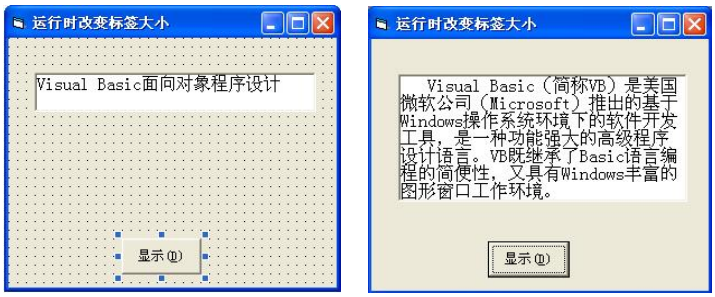
（2）在属性窗口中设置对象属性。

```
Label1.Caption="Visual Basic 面向对象程序设计"
Label1.BorderStyle =1          '有边框标签
Label1.WordWrap= True          '标签纵向扩充，并自动换行
Label1.Autosize=True           '标签横向扩充
```

（3）编写程序代码。

```
Private Sub Command1_Click()
  Label1.Caption = " Visual Basic（简称 VB）是 Microsoft 推出的"& _
  "基于 Windows 操作系统环境下的软件开发工具，是一种功能强大的高级程序"& _
  "设计语言。VB 既继承了 BASIC 语言编程的简便性，又具有 Windows 丰富的"& _
  "图形窗口工作环境。"
End Sub
```

程序运行后，单击“显示（D）”按钮，得到如图 3-3（b）所示的结果。其中可以看出，当将标签的水平自动扩充 AutoSize 和垂直自动扩充 WordWrap 属性都设置为 True 时，标签将自动适应其中的内容而改变大小。



（a）　　（b）

图 3-3　自动改变标签的尺寸

【例 3-4】模拟秒表计时。

（1）建立应用程序界面。在窗体上分别显示开始时间、结束时间和经过时间。它们都是不允许编辑的数据，所以用三个标签来显示（Label4～Label6）；另外用三个标签作文字说明（Label1～Label3）；再用两个命令按钮（Command1 和 Command2）实现秒表的开始、停止操作。

（2）设置对象属性。各控件的主要属性如图 3-4 所示。

（3）编写程序代码。

```
Dim startime, endtime, pastime                    ' 在通用段说明可变类型变量
  Private Sub Command1_Click()                    '“开始”按钮的单击事件过程
  startime = Now                                  ' 获得系统的时间
  Label4.Caption = Format(startime, "hh:mm:ss") ' 使用格式函数显示时间
  Label5.Caption = ""
```

```
  Label6.Caption = ""
  Command2.Enabled = True
  Command1.Enabled = False
End Sub
Private Sub Command2_Click()     '“停止”按钮的单击事件过程
  endtime = Now
  passtime = endtime - startime
  Label5.Caption = Format(endtime, "hh:mm:ss")
  Label6.Caption = Format(passtime, "hh:mm:ss")
  Command2.Enabled = False
  Command1.Enabled = True
End Sub
```

说明：变量 startime、endtime 和 pastime 分别表示开始时间、结束时间和经过时间，它们为两个事件过程共用，所以必须在通用段加以说明。程序运行结果如图 3-4 所示。

图 3-4　模拟秒表

2. 使用 Print 方法输出数据

（1）Print 方法。

Print 方法用于在窗体（Form）、立即窗口（Debug）、图片框（PictureBox）或打印机（Printer）等对象中显示或打印输出字符串或表达式的值，其语法格式如下：

```
[<对象名称>.]Print [<表达式列表>][,|;]
```

说明：

1）<对象名称>可以是窗体、立即窗口、图片框或打印机。如果省略了<对象名称>，则在当前窗体上输出。例如：

```
Print "Visual Basic"            ' 在当前窗体中显示"Visual Basic"
form2.Print "Visual Basic"       ' 在 form2 窗体中显示"Visual Basic"
Picture.Print "Visual Basic"     ' 在图片框 Picture 中显示"Visual Basic"
Debug.Print "Visual Basic"       ' 在立即窗口中显示"Visual Basic"
Printer.Print " Visual Basic"    ' 在打印机上打印"Visual Basic"
```

2）<表达式列表>可以是一个或多个表达式，如果省略，则输出一个空行。Print 方法具有计算和输出的双重功能，先计算表达式的值后输出。

当输出多个表达式时，各表达式之间用逗号“,”或分号“;”隔开。使用逗号分隔符，则各输出项按标准输出（分区输出）格式显示，即每隔 14 列为一个输出区，逗号后面的表达式的值将在下一个输出区显示；使用分号分隔符，则按紧凑格式输出，即后一项紧跟前一项输出。在一个 Print 语句中，可以将逗号和分号混合使用。

3）如果语句末尾没有分隔符，则执行 Print 方法输出当前输出项后将自动换行，下面的 Print 输出的内容将在新的一行上显示。为了使上下两个 Print 语句输出内容显示在同一行上，

则需要在上一个 Print 语句的最后加上逗号或分号。

【例 3-5】用 Print 方法在窗体中直接输出字符串或表达式的值。编写窗体的单击事件 Form_Click 事件代码如下，程序执行结果如图 3-5 所示。

```
Private Sub Form_Click()
  Print "3*4="; 3 * 4          ' 使用";"分隔
  Print "3*4=", 3 * 4          ' 使用","分隔
  Print                        ' 输出一个空行
  Print Date                   ' 输出当前日期
  Print 2 > 3                  ' 输出关系表达式的值 False
  Print
  FontSize = 18                ' 设置字体大小
  FontBold = True              ' 字体加粗
  Print "欢迎学习";            ' 在行尾使用分号
  Print "VB 程序设计"
End Sub
```

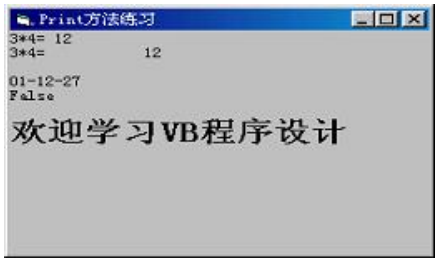


图 3-5　程序执行结果

（2）与 Print 方法有关的函数。

1）Tab 函数。在 Print 方法中，可以使用 Tab 函数对输出结果进行定位。其格式为：

```
Tab(n)
```

其中，n 为数值表达式，其值为整数，它用来指定表达式输出的起始列号。要输出的内容放在 Tab 函数的后面，并用分号隔开。例如：

```
Print Tab(5); "计算机";Tab(15); "世界"
```

当在一个 Print 方法中有多个 Tab 函数时，每个 Tab 函数对应一个输出项，各输出项之间用分号隔开。

2）Spc 函数。在 Print 方法中，用 Spc 函数跳过 n 列。其格式如下：

```
Spc(n)
```

其中，n 是在显示或打印下一个表达式之前插入的空格数。Spc 函数与输出项之间用分号隔开。例如：

```
print "计算机";Spc(5); "世界"
```

Spc 函数与 Tab 函数的作用类似，可以互相代替。但需要注意，Tab 函数从对象的左端开始计数，而 Spc 只表示两个输出项之间的间隔。

3. 使用信息框函数（MsgBox）输出数据

使用信息框函数 MsgBox 可以产生一个对话框来显示信息，并且在用户单击某个按钮后返回一个整数值以表明用户单击了哪个按钮。MsgBox 函数的使用格式为：

```
<变量名>=MsgBox(<提示信息> [,<对话框类型>  [,<对话框标题>]])
```

说明：

（1）<提示信息> 指定在对话框中显示的文本，最大长度为 1024 个字符，对话框的高度和宽度随<提示信息>的增加而增加。在<提示信息> 中使用回车符 Chr(13)、换行符 Chr(10)使显示的文本换行。

（2）<对话框类型>用于控制对话框中出现的按钮数目和图标样式，一般有四个参数，其取值和含义如表 3-1 至表 3-4 所示。下述四种参数值用加号连接共同作为<对话框类型>，这四种参数中的每一类都对应有几种取值情况，每个取值既可以用具体数值表示，也可以用系统定义的常量来表示。<对话框类型>可省略，若省略默认值为 0，即只显示一个“确定”按钮，而且此按钮为默认按钮，此时逗号分隔符不能省略。

表 3-1　参数 1——命令按钮种类

数值	常量	说明
0	vbOKOnly	“确定”按钮
1	vbOKCancel	“确定”和“取消”按钮
2	vbAbortRetryIgnore	“终止”、“重试”和“忽略”按钮
3	vbYesNoCancel	“是”、“否”和“取消”按钮
4	vbYesNo	“是”和“否”按钮
5	vbRetryCancel	“重试”和“取消”按钮

表 3-2　参数 2——图标类型

数值	常量	说明
16	vbCritical	停止图标“×”
32	vbQuestion	问号图标“？”
48	vbExclamation	警告图标“!”
64	vbInformation	信息图标“i”

表 3-3　参数 3——默认按钮

数值	常量	说明
0	vbDefaultButton1	指定默认按钮为第一按钮
256	vbDefaultButton2	指定默认按钮为第二按钮
512	vbDefaultButton3	指定默认按钮为第三按钮

表 3-4　参数 4——等待模式

数值	常量	说明
0	vbApplicationModal	当前应用程序挂起，直到用户对信息框做出响应才继续工作
4096	vbSytemModal	所有应用程序挂起，直到用户对信息框做出响应才继续工作

（3）<对话框标题>指定对话框的标题，该选项可以省略。

（4）Msgbox()函数的返回值指明了用户在对话框中选择了哪一个按钮，在表 3-1 中共提到了可能出现的 7 种按钮，它们是：“确认”、“取消”、“终止”、“重试”、“忽略”、“是”和“否”按钮。函数的返回值分别与这 7 种按钮相对应，为从 1 到 7 的 7 个数值，具体对应情况如表 3-5 所示。

表 3-5　MsgBox 函数返回值

返回值	返回常量	操作说明
1	vbOK	选择了“确定”按钮
2	vbCancel	选择了“取消”按钮
3	vbAbort	选择了“终止”按钮
4	vbRetry	选择了“重试”按钮
5	vbIgnore	选择了“忽略”按钮
6	vbYes	选择了“是”按钮
7	vbNo	选择了“否”按钮

（5）<变量名>用来存放用户关闭信息框时所选择的命令按钮的返回值，若不需要返回值，则可省略该项。即信息对话框可以写成语句形式。例如，如图 3-6 所示是信息对话框 MsgBox 语句形式的用法和执行结果。

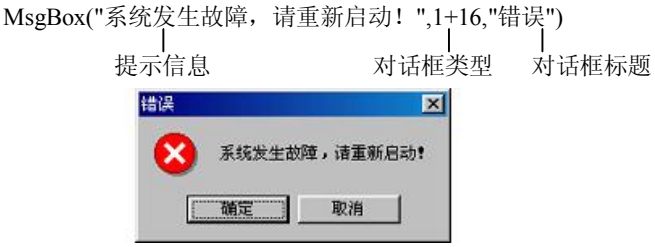


图 3-6　语句形式和执行结果

【例 3-6】信息对话框 MsgBox 函数形式举例。在任意程序中，可增加一个窗体关闭时确认是否退出程序的功能，如图 3-7 所示。

图 3-7　运行结果

在窗体的 Unload（卸载）事件代码中增加如下代码：

```
Private Sub Form_Unload(Cancel As Integer)
  X = MsgBox(" 确实要退出吗？", 4 + 32 + 256, "关闭窗口")
  If  X = 6  Then
      Cancel = 0
```

```
    Else
       Cancel = 1
    End If
End Sub
```

说明：

（1）当使用窗体右上角的“×”（关闭）按钮命令或用 Unload 语句关闭该窗体时，此事件被触发。

（2）参数 Cancel 是一个整数，用来确定窗体是否从屏幕删除。如果 Cancel 为 0，则窗体被删除。将 Cancel 设置为任何一个非 0 值可防止窗体被删除。

（3）本例中使用了条件语句 If…Then…EndIf，该语句的详细用法请参见下一节。

（4）信息对话框中的各参数也可以使用如表 3-1 至表 3-5 所示的常量代替数值。例如，在此例中 MsgBox 函数可以写成如下常量形式：

x = MsgBox(" 确实要退出吗？",vbYesNo+ vbQuestion+vbDefaultButton2,"关闭窗口")

3.1.3　数据输入

1. 使用文本框控件（TextBox）输入数据

文本框是一种最常用的控件，可以方便地由用户输入或显示文本。

【例 3-7】在文本框中输入三种家电商品的单价、销售数量，计算并输出总销售额。

（1）建立应用程序界面并设置对象属性。

在新窗体中建立 7 个标签 Label1～Label7，其中 Label7 用来显示计算结果，其他 6 个标签用于提示；6 个文本框和 3 个命令按钮，其中 6 个文本框用来输入商品的单价和数量。设计界面及运行结果如图 3-8 所示。

图 3-8　用文本框输入数据

（2）编写代码。

```
Private Sub Command1_Click()          ' 计算按钮的单击事件
  Dim a1 As Single, b1 As Single, c1 As Single
  Dim a2 As Single, b2 As Single, c2 As Single
  Dim x As Single
  a1 = Val(Text1.Text)                ' 电视机的单价
  b1 = Val(Text2.Text)                ' 洗衣机的单价
  c1 = Val(Text3.Text)                ' 电冰箱的单价
  a2 = Val(Text4.Text)                ' 电视机的数量
  b2 = Val(Text5.Text)                ' 洗衣机的数量
```

```
    c2 = Val (Text6.Text)                    ' 电冰箱的数量
    x = a1 * a2 + b1 * b2 + c1 * c2          ' 计算三种家电的总销售额
    Label7.Caption = x
End Sub
Private Sub Command2_Click()                 ' 清除按钮的单击事件
    Text1.Text = ""                          ' 以下语句使各文本框清空
    Text2.Text = ""
    Text3.Text = ""
    Text4.Text = ""
    Text5.Text = ""
    Text6.Text = ""
    Label7.Caption = ""
    Text1.SetFocus                           ' Text1 获得焦点
End Sub
Private Sub Command3_Click()                 ' “关闭”按钮的单击事件
    Unload Me                                ' 关闭当前窗体
End Sub
```

【例 3-8】在文本框中输入任意一串英文字符，将它们进行大小写转换。

（1）在新窗体中建立 1 个标签 Label1、1 个文本框 Text1 和 3 个命令按钮 Command1～Command3。各对象的字号、字体等属性由读者自行设计。

（2）编写代码。

本题主要使用大小写转换函数 UCase()和 LCase()，还用到了 KeyUp（键盘抬起）事件，并将最初输入的字符存入文本框的 Tag 属性中。

```
Private Sub Text1_KeyUp(KeyCode As Integer, Shift As Integer)
    Text1.Tag = Text1.Text               ' 键盘抬起时将 Text1 中的文本存入 Tag 属性
End Sub
Private Sub Form_Activate()              ' 窗体的激活事件
    Text1.SetFocus                       ' Text1 获得焦点
End Sub
Private Sub Command1_Click()             ' “大写”按钮的单击事件
    Text1.Text = UCase(Text1.Tag)        ' 转换成大写
End Sub
Private Sub Command2_Click()             ' “小写”按钮的单击事件
    Text1.Text = LCase(Text1.Tag)        ' 转换成小写
End Sub
Private Sub Command3_Click()             ' “还原”按钮的单击事件
    Text1.Text = Text1.Tag               ' 将 Text1.Tag 中存放的原始数据显示在文本框中
End Sub
```

程序的执行结果如图 3-9（b）所示。

（a）　　（b）

图 3-9　大小写转换

2. 使用输入框（InputBox）函数输入数据

InputBox 函数用于将用户从键盘输入的数据作为函数的返回值返回到当前程序中，此函数采用对话框界面，可以提供一个良好的交互环境。其语法格式为：

```
变量名=InputBox(<提示内容>,[<对话框标题>],[<默认值>])
```

说明：

（1）<提示内容>指定在对话框中显示的文本。要使<提示内容>换行显示，可以在换行处插入回车符 Chr(13)、换行符 Chr(10)。对话框的高度和宽度随着<提示内容>而增加，最多可有 1024 个字符。

（2）<对话框标题>指定对话框的标题。

（3）<默认值>用于指定输入内容的文本框中显示的初始文本。

（4）在输入框内输入信息后，若用户单击“确定”按钮，将把输入信息返回到变量中；若单击“取消”按钮，返回的将是一个空字符串。

例如，如图 3-10 所示是输入框 InputBox 函数的用法和执行结果。

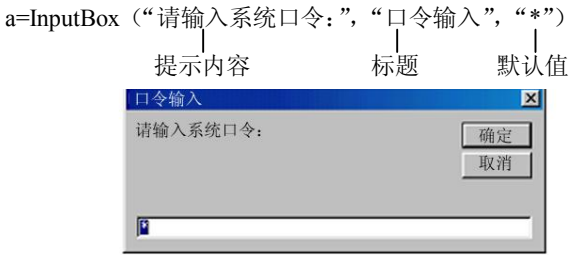


图 3-10　InputBox 用法举例

3. 焦点和 Tab 键顺序

（1）焦点（Focus）。

一个应用程序可以有多个窗体，每个窗体又可以有很多对象，但用户任何时候只能操作一个对象。焦点是对象接收鼠标或键盘输入的能力，当对象具有焦点时才能接受用户的输入，因此当前被操作的对象我们称它获得了焦点。一个窗体上如果有多个文本框，只有具有焦点的文本框才能接受和显示由键盘输入的文本内容。

窗体和多数控件都可以接受焦点，某些控件不具有焦点，如标签、框架、计时器等。仅当控件的 Visible 和 Enabled 属性被设置为真（True）时，控件才能接受焦点。焦点在任何时候只能有一个，当对象获得焦点时发生 GetFocus 事件，当对象失去焦点时发生 LostFocus 事件。可以用以下方法将焦点赋给对象：

1）在程序运行时，用鼠标或快捷键选择对象，按 Tab 键或 Shift+Tab 组合键在当前窗体的各对象之间切换焦点。

2）在程序代码中用 SetFocus 方法来设置焦点。

如在例 3-8 中，编写窗体的 Activate 事件代码，其中调用了 SetFocus 方法，使得程序开始时光标（焦点）位于文本框 Text1 中。

```
Private Sub Form_Activate()
  Text1.SetFocus
End Sub
```

（2）Tab 键顺序。

Tab 键顺序是指用户按 Tab 键时，焦点在窗体上的控件之间移动的顺序。一般情况下，Tab 键顺序由向窗体中建立控件的先后顺序确定。

例如，假定在窗体上按以下顺序建立了 5 个控件：Text1、Text2、Text3、Command1 和 Command2。程序运行时，Text1 首先获得焦点，每按一次 Tab 键，焦点按 Text2、Text3、Command1、Command2 的顺序移动。当焦点位于 Command2 时，如果再按 Tab 键，焦点又回到 Text1。

设置控件的 TabIndex 属性可以改变控件的 Tab 键顺序。当在窗体上建立第一个控件时，控件的 TabIndex 属性默认设置为 0，第二个控件的 TabIndex 属性默认值为 1，依此类推。程序运行中按 Tab 键，焦点将根据 TabIndex 属性值所指定的焦点移动顺序移动到下一个控件上。通过设置控件的 TabIndex 属性值，可以改变焦点的移动顺序。不能获得焦点的控件，以及无效的和不可见的控件，不具有 TabIndex 属性，因而不包含在 Tab 键顺序中，按 Tab 键时，这些控件将被跳过。通常在程序运行时按 Tab 键可以选择 Tab 键顺序中的每一个控件，若控件的 TabStop 属性设置为 False，按 Tab 键时就会跳过该控件，但该控件的 TabIndex 顺序值仍然保留。

3.1.4 常用基本语句

1. 卸载对象语句（Unload）

从内存中卸载指定的窗体或控件，可以使用 Unload 语句。Unload 语句的语法格式：

```
Unload 对象名
```

说明：

（1）对象名是要卸载的窗体对象或控件的名称。

（2）如果卸载的对象是程序中唯一的窗体，将终止程序的执行。例如：

```
Private Sub Command1_Click()
  Unload Me          '卸载当前窗体，Me 是系统关键字，代表当前窗体
End Sub
```

2. 结束语句（End）

语法格式：

```
End
```

End 语句能够强行终止程序代码的执行，清除所有变量。在集成开发环境下运行程序的过程中，用户也可以单击工具栏上的“结束”按钮来强行结束程序的执行。

3.2 选择结构

用顺序结构只能编写简单的程序，处理一些简单的问题。在实际应用中，有许多问题需要判断条件，根据判断的结果来控制程序的流程。选择结构就是能够根据不同的情况做出不同的选择，执行不同的操作，它是程序设计中的基本结构之一。

3.2.1 条件语句（If 语句）

Visual Basic 提供了两种格式的条件语句：单行条件语句和块结构条件语句。

1. 单行条件语句

单行条件语句比较简单，其语法格式如下：

```
If <条件> Then <语句块 1> [ Else <语句块 2> ]
```

功能：如果“条件”成立（值为 True），则执行语句块 1，否则执行语句块 2。

说明：

（1）“条件”通常是关系表达式或逻辑表达式。

（2）单行结构条件语句要求在一行内书写完毕，不能超过一行 255 个字符的限度。

（3）语句块 1 和语句块 2 可以是简单语句，也可以是用冒号分隔的多个语句。

（4）[Else <语句块 2>] 是可选项，当省略该项时，If 语句简化为：

```
If <条件> Then <语句块 1>
```

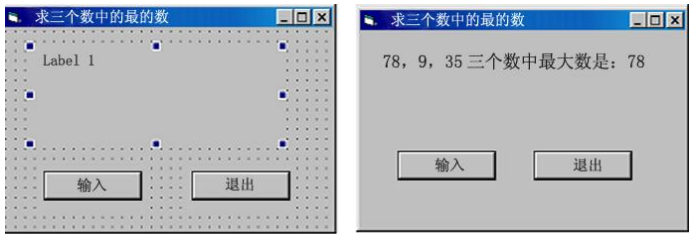
功能：如果“条件”成立（值为 True），则执行语句块 1，否则直接执行 If 语句的下一条语句。

【例 3-9】输入三个数 a、b、c，输出三者之中的最大数。

（1）建立应用程序界面与设置属性。参照前一节的方法建立用户界面和设置属性，在窗体上设置一个标签 Label1、两个命令按钮 Command1 和 Command2，如图 3-11（a）所示。

（2）编写程序代码。首先使用 InputBox 函数把三个数值输入到变量 a、b、c 中，假设 a 最大，放在变量 max 中，然后分别与 b 和 c 比较，把最大数放在变量 max 中。编写 Command1 的单击事件，将结果输出在 Label1 中。程序执行结果如图 3-11（b）所示。

```
Private Sub Command1_Click()
  Dim a As Single, b As Single, c As Single, max As Single
  Label1.Caption = ""
  a = Val(InputBox("请输入第 1 个数:", "输入 a", 0))
  b = Val(InputBox("请输入第 2 个数:", "输入 b", 0))
  c = Val(InputBox("请输入第 3 个数:", "输入 c", 0))
  p = a & "," & b & "," & c
  p = p & " 三个数中最大数是:"
  max = a                    '设 a 为最大值
  If b > max Then max = b
  If c > max Then max = c
  Label1.Caption = p & max
End Sub
Private Sub Command2_Click()
  Unload Me
End Sub
```



（a）　　（b）

图 3-11　比较三个数大小

2. 块结构条件语句

在单行结构条件语句中，如果条件分支执行的操作比较复杂，不能在一个逻辑行内书写

完毕时，可以使用块结构条件语句。块结构条件语句又称为多行条件语句，语法格式如下：

```
If <条件> Then
  <语句块 1>
[Else
  <语句块 2>]
End If
```

功能：首先判断<条件>，如果条件为 True，则执行 Then 后面的语句块 1，否则执行 Else 后面的语句块 2。当执行完 Then 或 Else 之后的语句块后，继续执行 End If 后面的语句。

说明：

（1）在块 If 语句中，If 语句必须是第 1 行语句，End If 语句是 If 块的最后一个语句。

（2）语句块可以是单条语句，也可以是多条语句。当有多条语句时，可以分别写在多行里；如果写在一行中，则各语句之间用冒号隔开。

（3）Else 子句是可选项，若有该项，Else 必须单独占一行。

【例 3-10】火车站托运行李，按规定当行李重量不超过 50 公斤时，每公斤运费 0.75 元，超过 50 公斤后，超过部分按每公斤 0.9 元收费。输入行李重量，计算出应付运费。

分析：设行李重量为 w 公斤，应付运费为 y 元，则运费公式为：

$$y=\begin{cases}0.75w & w\leqslant 50\\ 50\times 0.75+(w-50)\times 0.9 & w>50\end{cases}$$

（1）建立用户界面并设置对象的属性。程序运行后，通过文本框输入行李的重量，计算结果输出在标签中。当用户输入重量值时，单击“计算”按钮或按回车键后，在 Label3 中显示出应付的运费金额；当用户单击“清除”按钮或按 Esc 键后，清除重量和运费；用户修改重量时，Label3 中的数据自动清除，如图 3-12 所示。

图 3-12　行李托运界面

（2）编写程序代码。

```
Private Sub Command1_Click()          '“计算”按钮的 Click 事件
Dim w As Single, y As Single
w = Val(Text1.Text)
If w <= 50 Then
  y = w * 0.75
 Else
  y = 50 * 0.75 + (w - 50) * 0.9
End If
Label3.Caption = y
End Sub
```

```
Private Sub Command2_Click()     '“清除”按钮的 Click 事件
Text1.SetFocus
Text1.Text = ""
Label3.Caption = ""
End Sub

Private Sub Text1_Change()       ' 当 Text1 中的内容变化时，响应 Change 事件
Label3.Caption = ""
End Sub
```

【例 3-11】设计一个接受口令的窗体，单击“确定”按钮后，如果输入的口令正确则进入一个新窗口，如图 3-13（b）所示；若输入的口令错误，则给出错误提示。无论用户输入什么字符，文本框中只显示相同数量的“*”，如图 3-13（a）所示。

（a）　　（b）

图 3-13　输入口令

（1）建立应用程序界面。建立一个新工程，在其中添加两个窗体 Form1 和 Form2。在 Form1 中添加两个标签、一个文本框和一个命令按钮。在 Form2 中添加一个标签。程序执行结果如图 3-13 所示，其中（a）图为 Form1，（b）图为 Form2。

（2）设置对象属性，如表 3-6 所示。

表 3-6　各对象的属性

窗体	对象	属性	属性值	说明
Form1	Form1	Caption	输入系统口令	
	Label1	Caption	请输入系统的口令	
	Label2	Caption	空	
	Command1	Caption	确认	
	Text1	Text	空	
		PasswordChar	*	Text1 中的字符
Form2	Form2	Caption	口令正确	
	Label1	Caption	欢迎使用工资管理系统	

（3）编写程序代码。将 Form1 设置为启动窗口，启动 Form1 后，在文本框中输入口令，单击“确定”按钮后，若输入的口令正确（“abc”，大小写均可），则显示 Form2 窗体；否则，

在 Form1 的 Label2 中显示“对不起，口令错!”等。

编写“确认”按钮的 Click 事件如下：

```
Private Sub Command1_Click()
  a = UCase(Text1.Text)              ' 将 Text1 中输入的字符转换成大写字母
  If a = "ABC" Then                  ' 口令正确
    Unload Form1                     ' 卸载窗体 Form1
    Form2.Show                       ' 显示窗体 Form2
  Else
    Label2.Caption = "对不起，口令错！" & Chr(13)& "请重新输入"
    Text1.SetFocus                   ' Text1 获得焦点
    Text1.SelStart = 0               ' 将 Text1 的插入点置于最前端
    Text1.SelLength = Len(a)         ' 选中用户上次输入错误的所有字符
  End If                             ' 自动进入改写状态
End Sub
```

编写窗体 Form1 的装载事件如下：

```
Private Sub Form_Load()
   Text1.SelStart = 0                ' 将 Text1 的插入点置于最前端
   Text1.SelLength = Len(Text1.Text)
End Sub
```

编写 Text1 的 Change 事件如下：

```
Private Sub Text1_Change()
   Label2.Caption = ""
End Sub
```

3. 条件语句的嵌套

（1）一般格式的条件语句嵌套。

在条件语句中，如果 Then 后面的语句块 1 或 Else 后面的语句块 2 中还包含另一个条件语句，则称为条件语句的嵌套。其一般格式如下：

```
If <条件 1> Then
  …
  If <条件 2> Then
     …
  Else
     …
  End  If
  …
Else
  …
  If <条件 2> Then
     …
  Else
     …
  End  If
  …
End  If
```

语句块 1

语句块 2

【例 3-12】从键盘输入某课程的百分制的成绩 mark，要求转换成五级制的 grade，转换条件为：

$$\text{grade}=\begin{cases}\text{优秀} & \text{mark}\geqslant 90\\ \text{良好} & 80\leqslant \text{mark}\leqslant 90\\ \text{中等} & 70\leqslant \text{mark}\leqslant 80\\ \text{及格} & 60\leqslant \text{mark}\leqslant 70\\ \text{不及格} & \text{mark}<60\end{cases}$$

程序代码如下：

```
Private Sub Form_Load()
 Dim mark As Integer, grade As String
 mark = InputBox("请输入百分制成绩")
 Show
 If mark >= 90 Then
   grade = "优秀"
 Else
   If mark >= 80 Then
     grade = "良好"
   Else
     If mark >= 70 Then
       grade = "中等"
     Else
       If mark >= 60 Then
         grade = "及格"

       Else
         grade = "不及格"
       End If
     End If
   End If
 End If
 Print "百分制成绩是："; mark
 Print "转换成等级是："; grade
End Sub
```

使用一般格式条件语句嵌套时，一定要注意 If 与 Else、If 与 End If 的配对关系。在上面所给的程序代码中，由于采取了按层次凹进的写法，还是比较容易看出条件语句之间嵌套的逻辑关系，以及 If 与 Else、If 与 End If 的配对关系的。

（2）ElseIf 格式的条件语句嵌套。

多层条件语句嵌套容易造成程序冗长，难以确定 If 与 Else、If 与 End If 的配对关系，给编写和阅读程序造成困难。为了解决这些问题，Visual Basic 提供了带有 ElseIf 的条件语句嵌套结构，格式为：

```
If <条件 1> Then
  <语句块 1>
Else If  <条件 2> Then
  <语句块 2>
Else If <条件 3> Then
  <语句块 3>
…
[Else
```

```
    <语句块 n>]
End If
```

功能：该语句执行时先判断<条件 1>，如果为 True，执行<语句块 1>；如果为 False 就判断<条件 2>，依此类推，直到找到为 True 的条件。一旦找到一个为 True 的条件，就执行相应的语句块，然后执行 End If 语句后面的代码；如果所有的条件都是 False，就执行 Else 后面的<语句块 n>，然后执行 End If 语句后面的代码。

【例 3-13】使用带 ElseIf 的条件语句实现例 3-12 的题目要求。

```
Private Sub Form_Load()
Dim mark As Integer, grade As String
 mark = InputBox("请输入百分制成绩")
 Show
 If mark >= 90 Then
   grade = "优秀"
 ElseIf mark >= 80 Then
   grade = "良好"
 ElseIf mark >= 70 Then
   grade = "中等"
 ElseIf mark >= 60 Then
   grade = "及格"
 Else
   grade = "不及格"
 End If
 Print "百分制成绩是："; mark
 Print "转换成等级是："; grade
End Sub
```

由此可见，使用 If…Then…ElseIf 结构简单、清晰，省去了嵌套结构的多个 End If 语句。

注意：①论有多少分支，程序执行一个满足条件的分支后退出分支结构，其余分支不再执行；②ElseIf 不能写成 Else If；③注意条件表达式的书写顺序，否则可能会出现逻辑错误。例如上例中的条件语句写成：

```
If mark >= 60 Then
   grade = "及格"
 ElseIf mark >= 70 Then
   grade = "中等"
 ElseIf mark >=80 Then
   grade = "良好"
 ElseIf mark >=90 Then
   grade = "优秀"
 Else
   grade = "不及格"
 End If
```

以上程序段虽然没有语法错误，但不能按要求获得正确的结果。只能得到“及格”或“不及格”两种情况，原因请读者自行分析。

4. 使用 IIf 函数

在程序设计中可以使用 IIf 函数来实现一些比较简单的条件判断操作，IIf 函数的语法格式为：

```
IIf (<条件表达式>,<条件为真时的值>,<条件为假时的值>)
```

说明：

（1）“条件表达式”可以是关系表达式、布尔表达式或数值表达式。如果用数值表达式作条件，则非 0 为真，0 为假。

（2）“条件为真时的值”是当条件表达式为真时函数返回的值，“条件为假时的值”是当条件表达式为假时函数返回的值，它们可以是任何表达式。

（3）语句 y=IIf(<条件表达式>,<条件为真时的值>,<条件为假时的值>)相当于：If <条件表达式> Then y=<条件为真时的值> Else y=<条件为假时的值>。

例如数学中的分段函数 $y=\begin{cases}1+x \geqslant 0\\ x^2<0\end{cases}$，使用 If 语句描述为：

```
If x>=0 Then y=1+x Else y=x^2
```

使用 IIf 函数描述为：

```
y=IIf(x>=0, 1+x, x^2)
```

3.2.2　Select Case 语句

在程序设计中，经常会有多种情况的选择，虽然可以使用 If 语句的嵌套形式实现多分支选择，但结构不够简明。使用 Select Case 语句也可以实现多分支选择，而且语句更为简单、易读，它根据表达式的值来决定执行多组语句中的哪一组。Select Case 语句的语法格式为：

```
Select Case <测试表达式>
    Case <表达式列表 1>
       <语句块 1>
    [Case <表达式列表 2>
       <语句块 2>]
     …
    [Case <表达式列表 n>
       <语句块 n>]
    [Case Else
     <语句序列 n+1>]
End Select
```

功能：多分支选择语句以 Select Case 开头，以 End Select 结束，根据“测试表达式”的值从多个 Case 中选择符合条件的一个语句块执行。其执行过程是：先对“测试表达式”求值，然后顺序测试该值与哪个 Case 子句中的“表达式列表”相等，如果找到了则执行该 Case 分支后的语句块，然后执行 End Select 后面的语句；如果没有找到则执行 Case Else 后的语句序列，然后执行 End Select 后面的语句。

说明：

（1）<测试表达式>通常是数值表达式或字符表达式，一般为一个变量。

（2）<表达式列表>值的数据类型必须与<测试表达式>值的数据类型相同。

（3）<表达式列表>的表达形式有以下三种：

- 表达式：该种表达方式通常列出一些具体的取值，例如 Case 1,3,5,7。
- 表达式 To 表达式：该种表达方式常用来表示一个数值范围，较小的数应该放在关键字 To 的前面，例如 Case 3 To 9。
- Is 比较运算表达式：Is 关键字与比较运算符结合，也常用来表达一个数值范围，例如 Case Is<10。

（4）在每个 Case 子句中还允许使用表达式的混合形式。例如 Case 2,4,6, 8 To 10 ,Is>15。

（5）若多个 Case 子句中有同一种取值重复出现，则只执行第一个出现此取值的 Case 语句后的相应语句块。

【例 3-14】编写程序计算货物运费。设货物运费单价 y 元/每吨/每公里，运输距离为 s 公里。单价 y 与距离 s 的关系为：

$$Y=\begin{cases}30 & s<200\\ 28.5 & 200\leqslant s<300\\ 26 & 300\leqslant s<400\\ 24.5 & 400\leqslant s<500\\ 22 & s\geqslant 500\end{cases}$$

输入要托运的货物重量 w 吨、托运的距离 s 公里，计算总运费 f：

f=y*w*s（元）

（1）建立应用程序界面并设置对象属性。如图 3-14 所示，在窗体上添加 3 个标签、3 个文本框和两个命令按钮。其中 Text1 和 Text2 分别用来输入货物重量 w 和托运距离 s，Text3 用来显示总金额，并将 Text3 的 Locked 属性设置为 True，使之成为只读不能改写。

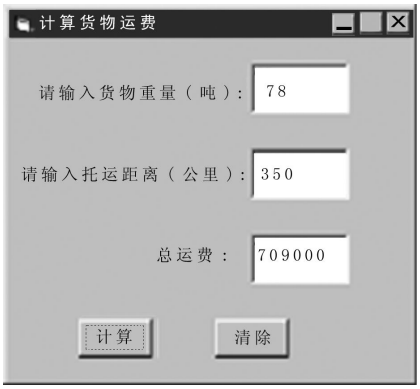


图 3-14　计算运费程序结果

（2）编写程序代码。

1）编写“计算”按钮的单击事件。

```
Private Sub Command1_Click()
  Dim w As Single, s As Single
  Dim y As Single, f As Single
  w = Val(Text1.Text)
  s = Val(Text2.Text)
  Select Case S
    Case Is < 200
      y = 30
    Case Is < 300
      y = 28.5
    Case Is < 400
      y = 26
```

```
    Case Is < 500
      y = 24.5
    Case Else
      y = 22
  End Select
  f = y * w * s
  Text3.Text = f
End Sub
```

2）编写“清除”按钮的单击事件。

```
Private Sub Command2_Click()          ' “清除”按钮的单击事件
  Text1.SetFocus                      ' Text1 获得焦点
  Text1.Text = ""
  Text2.Text = ""
  Text3.Text = ""
End Sub
```

3）编写 Text1 的键盘事件。

```
Private Sub Text1_KeyPress(KeyAscii As Integer)
  If KeyAscii = 13 Then               ' 在 Text1 中回车后 Text2 获得焦点
    Text2.SetFocus
  End If
End Sub
```

4）编写 Text2 的键盘事件。

```
Private Sub Text2_KeyPress(KeyAscii As Integer)
  If KeyAscii = 13 Then
    Command1.SetFocus                 ' 在 Text2 中回车后 Command1 获得焦点
  End If
End Sub
```

【例 3-15】给定年份和月份，判断该年是否为闰年，并根据给定的月份判断该月有多少天（闰年的条件是年份能被 4 整除但不能被 100 整除，或者能被 400 整除）。

分析：根据给定条件可以写出判断闰年的布尔表达式：

```
(y Mod 4 = 0 And y Mod 100 < > 0)Or (y Mod 400 = 0)
```

程序界面及程序执行结果如图 3-15 所示，程序代码如下：

```
Private Sub Command1_Click()
  Dim y As Integer, m As Integer, days As Integer
  Dim leap As Boolean                 ' 闰年标记
  If  Text1.Text = "" Then MsgBox "请输入正确的年份，重试！": Exit Sub
  If  Text2.Text = "" Or Val(Text2.Text)< 0 Or Val(Text2.Text)> 12 Then MsgBox _
    "请输入正确的月份，重试！": Exit Sub
    y = Val(Text1.Text)
    m = Val(Text2.Text)
  If (y Mod 4 = 0 And y Mod 100 <> 0)Or (y Mod 400 = 0)Then
    leap = True                       ' leap 的值为 True 时，表示该年为闰年
  Else
    leap = False
  End If
  Select Case m
   Case 1, 3, 5, 7, 8, 10, 12
     days = 31
```

```
    Case 4, 6, 9, 11
      days = 30
    Case 2
      If leap Then
        days = 29
      Else
        days = 28
      End If
  End Select
  Label4.Caption = y & "年" & IIf(leap, "是", "不是")& "闰年，" _
              & Chr(13)& m & "月有" & days & "天"
  Text1.SetFocus
End Sub
Private Sub Command2_Click()
  Text1.SetFocus
  Text1.Text = ""
  Text2.Text = ""
  Label4.Caption = ""
End Sub
```

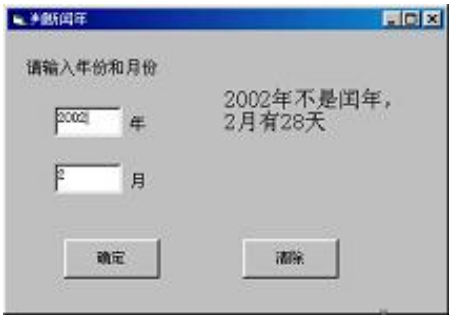


图 3-15　判断闰年

3.3　循环结构

循环结构是指在执行程序语句时，需要对其中的某些语句重复执行多次，被重复执行的程序段称为循环体。使用循环结构可以简化程序，提高程序执行效率。

Visual Basic 提供的循环结构语句有 For … Next、Do … Loop、For Each … Next 和 While … Wend 等。其中最常用的是 For … Next 和 Do … Loop 语句，本节主要讲解这两个循环语句的用法。

3.3.1　Do…Loop 语句

Do … Loop 语句是通过检测循环条件决定循环的语句，Do … Loop 语句具有很强的灵活性，它既可以构成先判断条件再决定是否执行循环体的形式，也可以构成先执行循环体后判断条件的形式。

1. 先判断条件形式的 Do…Loop 语句

语句格式为：

```
Do [While|Until <条件>]
   [循环体]
Loop
```

说明：

（1）Do While … Loop 是“当型循环”形式，即<条件>为真（True）时，执行循环体；条件为假（False）时，终止循环。

（2）Do Until … Loop 是“直到型循环”形式，即<条件>为假（False）时，执行循环；直到条件为真（True）时，终止循环。

（3）<条件>即循环条件，是一个关系表达式或逻辑表达式，其值为 True 或 False。

（4）在 Do … Loop 中，可以在循环体的适当位置使用 Exit Do 语句，以便随时退出。当有多个循环嵌套使用时，Exit Do 语句只能跳出当前的 Do … Loop 循环。

【例 3-16】计算 1+2+3+…+100。

分析：采用累加求和的方法，用变量 s 存放累加和，s 的初值为 0。用变量 n 计数，n 从 1 开始变化到 100。

（1）用当型循环语句 Do While … Loop，程序如下：

```
Private Sub Form_Load()
  Dim s As Integer, n As Integer
  Show
  s = 0: n = 1
  Do While n <= 100
    s = s + n
    n = n + 1
  Loop
  Print "1+2+3+…100="; s
End Sub
```

（2）用直到型循环语句 Do Until … Loop，程序如下：

```
Private Sub Form_Load()
  Dim s As Integer, n As Integer
  Show
  s = 0: n = 1
  Do Until n > 100
    s = s + n
    n = n + 1
  Loop
  Print "1+2+3+…100="; s
End Sub
```

（3）Do…Loop 循环语句也可以省略 While/Until，将循环结束条件写在循环体内，用 Exit Do 语句退出循环。程序如下：

```
Private Sub Form_Load()
  Dim s As Integer, n As Integer
  Show
  s = 0: n = 1
  Do
    s = s + n
    n = n + 1
    If n > 100 Then Exit Do
```

```
  Loop
  Print "1+2+3+…100="; s
End Sub
```

【例 3-17】若我国现有人口 13 亿，按人口年增长 0.8%计算，多少年后我国人口翻番。

分析：当前人口 13 亿，多少年后人口达 26 亿，即人口翻番。利用循环语句，可求得逐年增长的人口数量，当达到 26 亿时，停止循环。

程序如下：

```
Private Sub Form_Click()
 x = 13                  ' 当年的人数
 n = 0
 Do While x < 26         '当人数还未达到 26 亿
  x = x + (x * 0.008)    '根据增长率计算当年的人数
  n = n + 1              ' 年份加 1
 Loop
 Print "求得的年数为："; n
 Print "人数为："; Round(x, 3)   '保留 3 位小数
End Sub
```

2. 后判断条件形式的 Do … Loop 语句

语句格式为：

```
Do
  [循环体]
Loop [While|Until <条件>]
```

后判断条件形式的 Do…Loop 语句与先判断条件形式的区别是：首先执行循环体，然后判断条件的真假，决定是否继续循环。所以，后判断条件形式的 Do…Loop 语句至少执行一次循环体。

在例 3-16 中我们编写了求 1～100 累加和的程序，现将该程序改写成后判断条件形式的 Do…Loop 语句。

```
Private Sub Form_Load()
  Dim s As Integer, n As Integer
  Show
  s = 0: n = 1
  Do
    s = s + n
    n = n + 1
  Loop While n <= 100
  Print "1+2+3+…100="; s
End Sub
```

也可以改成使用直到型的后判断形式的 Do…Loop 语句，即 Do...Loop Until n>100，请读者自行改写程序。

3.3.2 For … Next 语句

For … Next 循环结构是实现循环的又一种形式，它属于计数型循环，通常用于循环次数已知的程序结构中。语法格式如下：

```
For <循环变量> =<初值> To <终值> [Step <步长>]
  [<循环体>]
Next [<循环变量>]
```

说明：

（1）< 循环变量>是一个数值型变量，用作循环计数器控制循环次数。

（2）<初值>为循环变量的初始取值，<终值>为循环变量的最后取值，它们均可以是数值型的常量、变量或表达式，它们的值可以是整数和实数。

（3）<步长>用于决定循环变量每次增加的数值，即变量在变化时的增值，也可为数值型的常量、变量或表达式。步长的取值可以根据初值和终值的关系分为正数和负数两种，若初值大于终值，则必须将步长设为负数；若初值小于终值，则必须将步长设为正数，才有可能执行内部循环体。当步长为 1 时，可以省略 Step 1，步长可以为实数。

（4）可以在循环体中的适当位置放置 Exit For 语句，以便随时退出循环。

（5）For 语句的执行过程是：首先把初值赋给循环变量，接着检查循环变量是否超过终值，如果超过，就停止执行循环体跳出循环，执行 Next 后面的语句；否则执行一次循环体，然后把“循环变量+步长”的值再赋给循环变量，重复上述过程。For 语句的流程图如图 3-16 所示。

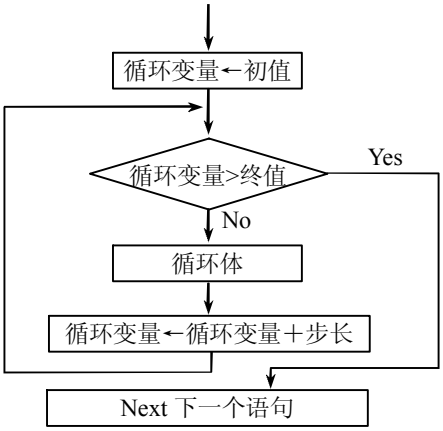


图 3-16　or 语句执行流程图

从流程图可以看到，For 语句是先判断条件的循环，如果把 For 语句的“初值”设定为超过“终值”，则不执行循环体，而直接执行 Next 后面的语句。例如：

```
For I=5 To 1
   <循环体>
Next I
```

这是一个没有意义的语句。这里所说的“超过”有两种含义，即大于或小于。当步长为整数时，检查循环变量是否大于终值；当步长为负数时，判断循环变量是否小于终值。

（6）循环的次数由初值、终值和步长三个因素决定，计算公式为：

循环次数=Int((终值−初值)/步长+1)

【例 3-18】求 n!（n 为自然数）。

由阶乘的定义可知：n!=1×2×3×…×(n−2)×(n−1)×n =(n−1)!×n

也就是说，一个自然数的阶乘等于该自然数与前一个数的阶乘的乘积，即从 1 开始连续地与下一个自然数相乘，直到 n 为止。

程序界面和属性设置请读者自行设计，编写窗体的激活事件，利用输入框由用户输入 n

的值，循环变量为 i，用 k 存储阶乘值，将结果显示在窗体上。程序代码如下：

```
Private Sub Form_Activate()
   Dim n As Integer, i As Integer
   Dim k As Long       '防止乘积结果溢出，将 k 定义为长整型或实型
   n = InputBox("请输入 n", "求 n! ")
   k = 1
   For i = 1 To n
    k = k * i
   Next i
   Print n; "!="; k
End Sub
```

其中 For 循环语句也可以写成：

```
For i=n To 1 Step -1
  k=k+1
Next i
```

【例 3-19】求 $e=\frac{1}{1!}+\frac{1}{2!}+\frac{1}{3!}+...+\frac{1}{n!}$，直到第 50 项。

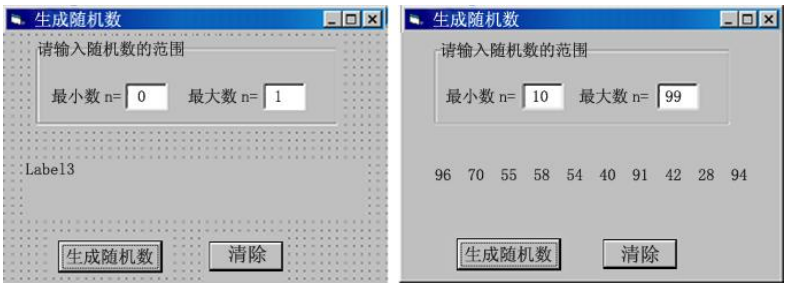
```
Private Sub Form_Click()
Dim i As Integer, t As Double, s As Single
t = 1: e = 0
For i = 1 To 50
 t = t * i
 e = e + 1 / t
Next i
Print "e="; e
End Sub
```

说明：为防止乘积结果数据溢出，将变量 t 定义为 Double 型。

【例 3-20】使用随机函数产生 10 个在指定范围内的随机数，随机数的范围在文本框内输入确定。

分析：在第 2 章中介绍的随机函数 Rnd()可以产生一个[0,1）区间内的随机小数，那么，Rnd*a 可以产生[0,a）区间中的随机实数。若 n 和 m 均为整数，则表达式 Int((m+1−n)*Rnd）+n 的值是闭区间[n,m]中的一个随机整数。

窗体的设计及对象属性如图 3-17（a）所示，在本例中使用了框架对象，框架作为容器在其中添加了两个标签 Label1 和 Label2、两个文本框 Text1 和 Text2。将产生的随机数显示在 Label3 中。有关框架控件的详细内容将在第 4 章中介绍。



（a）　　（b）

图 3-17　产生 10 个随机数

程序代码如下：

```
Private Sub Command1_Click()
   Dim n As Integer, m As Integer
   Randomize                                  '产生随机数种子
   n = Val(Text1.Text)
   m = Val(Text2.Text)
   For i = 1 To 10
    p = p & Int((m + 1 – n)* Rnd)+ n & "   "
   Next i
   Label3.Caption = p
End Sub
Private Sub Command2_Click()
   Text1.SetFocus                             ' Text1 获得焦点
   Text1.SelStart = 0                         ' 将光标置于 Text1 的起始位置
   Text1.SelLength = Len(Text1.Text)          ' 将 Text1 中的内容全部选中
   Text2.Text = ""
   Label3.Caption = ""
End Sub
```

3.3.3　多重循环

在循环体内可以包含 Visual Basic 的任何语句，当在循环体内包含另外一个循环语句时就形成了循环嵌套，又叫多重循环。

下面用一个简单的例子来分析多重循环的执行情况。

【例 3-21】双重循环举例。

编写窗体的激活事件，程序如下：

```
Private Sub Form_Activate()
  Dim i As Integer, j As Integer
  For i = 1 To 3
    For j = 4 To 5
      Print i, j
    Next j
  Next i
End Sub
```

程序的执行结果如图 3-18 所示。从图中可以看出，当外循环变量 i 每取一个值，其内循环将循环一个周期，即 j 将从 4 变化到 5。当 i=1 时，j 分别取 4 和 5；当 i=2 时，j 又分别取 4 和 5，依此类推。

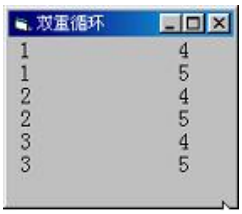


图 3-18　双重循环

$i=1$ 时 $\begin{cases} j=4 \\ j=5 \end{cases}$　　$i=2$ 时 $\begin{cases} j=4 \\ j=5 \end{cases}$　　$i=3$ 时 $\begin{cases} j=4 \\ j=5 \end{cases}$

【例 3-22】设计一个窗体打印乘法九九表。

```
Private Sub Form_Click()
  FontSize = 10
  Print Tab(31); "乘法九九表"
  Print Tab(30); "------------"
  For i = 1 To 9
    For j = 1 To i
      Print Tab((j - 1)* 8); i & "*" & j & "=" & i * j;
    Next j
    Print                    ' 每行之后产生一个换行
  Next i
End Sub
```

在窗体的任意位置单击鼠标会得到如图 3-19 所示的结果。

图 3-19　乘法九九表

【例 3-23】在窗体上输出如图 3-20 所示的图形。

图 3-20　输出三角形

程序代码如下：

```
Private Sub Form_Click()
  Cls
  Print: Print: Print
  For i = 1 To 8                ' 输出 8 行
    Print Tab(20 - i);          ' 控制每行的起始位置
    For j = 1 To 2 * i - 1      ' 每行输出 2*i-1 个“*”
      Print "*";
    Next
    Print                       ' 换行
  Next
End Sub
```

3.4　多重窗体程序设计

在一个复杂的应用程序中，一个工程可以包含多个窗体，即多重窗体（MultiForm），指的是应用中有多个窗口界面，这些窗口分别显示在屏幕上，它们之间没有绝对的从属关系，根据完成的任务相互联系在一起。

3.4.1　建立多重窗体应用程序

1. 多重窗体的添加

在多重窗体的应用程序中，要建立的界面由多个窗体组成。要向当前工程中添加新的窗体有以下三种方法：

（1）选择“工程”→“添加窗体”命令，可实现新窗体的添加。这些窗体的默认名称为 Form1，Form2 等。

（2）单击工具栏中的“添加窗体”按钮，完成添加新窗体。

（3）在“工程资源管理器”窗口中，右击要添加新窗体的“工程”，在弹出的快捷菜单中选择“添加”→“添加窗体”选项，也可以创建新窗体。

2. 移除窗体

要将一个窗体从工程中移除，有以下两种方法：

（1）在“工程资源管理器”窗口中选定要移除的窗体，选择“工程”→“移除”命令。

（2）在“工程资源管理器”窗口中，右击要移除的窗体，在弹出的快捷菜单中选择“移除”选项。

注意：移除的窗体并没有被删除，只是从该工程中移除走了。

3. 设置启动窗体

一个具有多个窗体的应用程序中，必须有一个启动窗体。默认情况下，应用程序会把设计阶段建立的第一个窗体作为启动窗体，在应用程序开始运行时，此窗体首先被显示出来。实际应用中如果要设置其他窗体为启动窗体，可以采用以下操作方法：

（1）选择“工程”→“工程属性”命令，弹出“工程属性”对话框；或者在“工程资源管理器”窗口中选定工程并右击，在弹出的快捷菜单中选择“工程属性”选项，弹出“工程属性”对话框，如图 3-21 所示。

图 3-21　“工程属性”对话框

（2）选择“通用”选项卡，在“启动对象”下拉列表框中选择要作为启动窗体的窗体。

（3）单击“确定”按钮。

3.4.2 多重窗体程序设计常用的方法

前面介绍的窗体的属性和方法同样适用于多重窗体程序设计。在多重窗体的程序设计中，经常要用到四种方法：Load 方法、Show 方法、Hide 方法、Unload 方法。

1. Load *方法*

Load 方法的语法格式：

```
Load [窗体名称]
```

使用 Load 方法调用的窗体只是被装入内存，并不会显示出来，与此同时会产生一个 Form_Load()事件。

2. Show *方法*

Show 方法的语法格式：

```
[窗体名称.] Show [窗体模式值]
```

使用 Show 方法可以显示被调用的窗体，如果在调用 Show 方法之前没有把窗体调入内存，那么 Show 方法会自动地把窗体调入内存。

窗体模式值是可选项，取值为 0 或 1，对应 Show 方法的两种模式。当取值为 0 或被省略时，执行 Show 方法显示的窗体是“无模式的”，用户可以激活其他窗体，对其他窗体进行操作；当取值为 1 时，执行 Show 方法显示的窗体是“模式的”，用户不能对其他窗体进行操作，直到该窗体关闭为止。一般应用程序的各种对话框大多是“模式的”，使用这种模式，只有关闭了对话框才能进行其他操作。

3. Hide *方法*

Hide 方法的语法格式：

```
[窗体名称.]Hide
```

使用 Hide 方法可以隐藏窗体，实际上是把窗体的 Visible 属性值设为 False。窗体被隐藏之后，虽然在屏幕上不可见，但它并没有从内存中移走。如果在调用 Hide 方法时窗体还没有加载，那么 Hide 方法将加载该窗体。

4. Unload *方法*

Unload 方法的语法格式：

```
Unload [窗体名称]
```

其功能是从内存中清除指定的窗体。

另外在多重窗体程序设计中，经常用到关键字 Me，它代表当前窗体的默认名称，Unload Me 语句可将当前窗体从内存中卸载。

3.4.3 多重窗体程序设计举例

多重窗体的应用程序设计过程中，首先应分析应用需求，将其功能划分为不同的部分。在创建窗体时，除了设计好各窗体自身要完成的功能外，还要考虑窗体之间的内在联系与调用关系，指定启动窗体，安排好各窗体之间的衔接。程序运行后显示的第一个窗体通常称为启动窗体。

【例 3-24】为了方便调试程序，将本章所有例题放在一个工程中，再增加一个启动窗体（假设为 Form24），并将该窗体设置为启动窗体，设置方法如图 3-21 所示，然后在该窗体中增加若干命令按钮，如图 3-22 所示。程序执行后单击任意一个按钮则启动该按钮所对应的窗体，避免了每次设置启动窗体的麻烦。

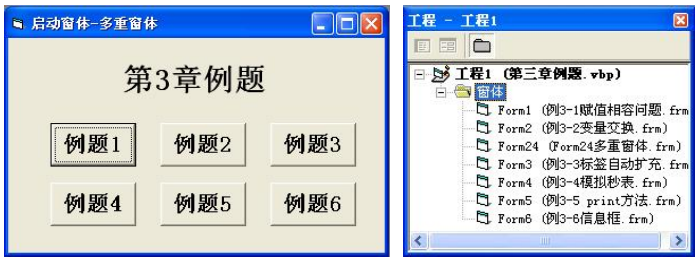


图 3-22　设置启动窗体

程序如下：

```
Private Sub Command1_Click(Index As Integer)    '按钮“例题 1”的单击事件
 Form1.Show          '显示“例题 1”窗体
 Form24.Hide
End Sub

Private Sub Command2_Click(Index As Integer)    '按钮“例题 2”的单击事件
 Form2.Show          '显示“例题 2”窗体
 Form24.Hide         '隐藏启动窗体
End Sub
```

此外，还需要编写“例题 1”中 Form1 的 Unload 事件。

```
Private Sub Form_Unload(Cancel As Integer)      '卸载“例题 1”窗体时显示启动窗体
Form24.Show
End Sub
```

其他窗体的显示和隐藏程序类似，请读者自行完成。

3.4.4　Sub Main 过程

有时一个应用程序启动执行时，不需要加载任何窗体，而是需要首先执行一段程序代码，完成一些初始化工作。如根据条件来决定加载哪一个窗体。要实现以上功能，可以在标准模块中建立一个名为 Main 的 Sub 过程，把首先需要执行的程序代码写在该 Sub Main 过程中，并且将 Sub Main 过程指定为启动对象，从而完成应用程序与窗体间的衔接。标准模块可以有多个，但 Sub Main 过程只能有一个，而且其名称必须是 Main。

当工程中含有已被设置为启动对象的 Sub Main 过程时，应用程序在运行时总是先执行 Sub Main 过程。

设置标准模块中的 Sub Main 过程为启动对象的方法是：在如图 3-21 所示的“工程属性”对话框中，选择“通用”选项卡，从“启动对象”下拉列表框中选择 Sub Main。

【例 3-25】Sub Main 过程示例。在例 3-24 的基础上添加一个标准模块，取名为“主程序.bas”。在标准模块的代码窗口中建立一个 Sub Main 程序，如图 3-23 所示。

选择“工程”→“添加模块”命令添加标准模块 Module1。在 Module1 的代码窗口中编写

代码（如图 3-23 所示），再设置 Sub Main 过程为“启动对象”。

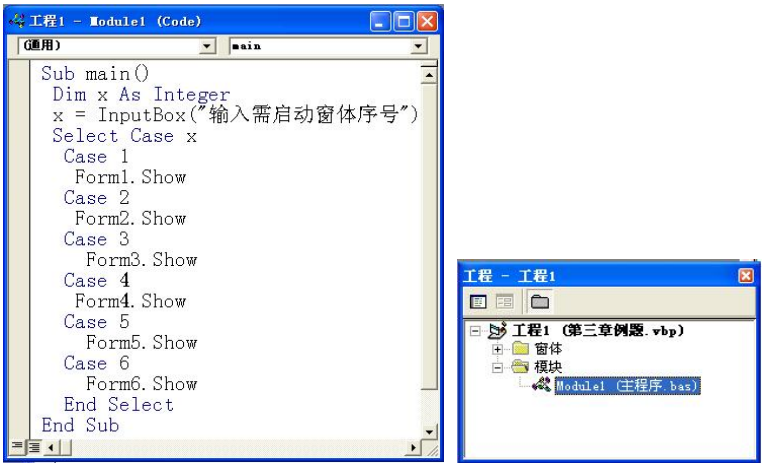


图 3-23　建立标准模块

启动程序后，自动调用标准模块，出现输入框后按照用户输入的序号调用不同的窗体。

3.5　综合程序举例

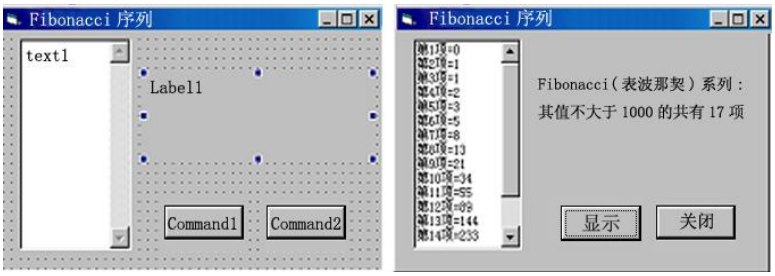
【例 3-26】Fibonacci（斐波那契）序列的前两项是 0 和 1，且每个后继项是前两项的和。因此 Fibonacci 序列为：0，1，1，2，3，5，8，13，21，34，…编写程序输出项的值不大于 1000 的 Fibonacci 序列。

（1）建立应用程序用户界面。在一个新窗体中添加一个文本框 Text1、一个标签 Label1、两个命令按钮 Command1 和 Command2，如图 3-24（a）所示。其中文本框 Text1 的 MultiLine 属性设置为 True（显示多行文本），ScrollBars 属性设置为 2-Vertical（显示垂直滚动条）。

（2）编写 Command1 和 Command2 的单击事件代码，执行结果如图 3-24（b）所示。

```
Private Sub Command1_Click()
  Dim i1 As Integer, i2 As Integer, i3 As Integer, n As Integer
  i1 = 0: i2 = 1: n = 1
  p = "第 1 项=" & i1
  Do While i2 <= 1000
     n = n + 1
     p = p & Chr(13)& Chr(10)& "第" & n & "项=" & i2
     i3 = i1 + i2
     i1 = i2              ' 得到新的 i1
     i2 = i3              ' 得到新的 i2
  Loop
  Text1.Text = p
  Label1.Caption = "Fibonacci(斐波那契)序列：" & Chr(13)& _
               "其值不大于 1000 的共有" & n & "项"
End Sub
Private Sub Command2_Click()
```

```
  Unload Me
End Sub
```



（a）　　（b）

图 3-24　Fibonacci 序列

说明：每执行一次循环体，由前两项求出第 3 项，即 i3=i1+i2，然后把原来的第 2 项赋给 i1，即 i1=i2，再将新求出的第 3 项赋给 i2，即 i2=i3，只要 i2≤1000，就不断重复执行循环体，并将每项的值累计到变量 p 中，用变量存储输出项的个数，退出循环后再一次将累计到变量 p 中的所有项的值显示在文本框中。函数 Chr(13)表示回车，Chr(10)表示换行。

【例 3-27】输入两个正整数，求它们的最大公约数，程序运行界面如图 3-25 所示。

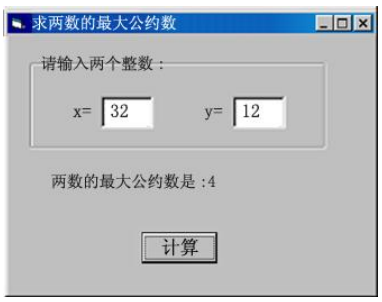


图 3-25　运行结果

求两个正整数的最大公约数可以采用“辗转相除法”，如下：

（1）以大数 x 作被除数，小数 y 作除数，相除后的余数为 r。

（2）若 r≠0，则 x←y，y←r，继续 x 与 y 相除得到新的 r，若仍有 r≠0，则重复此过程，直到 r=0 为止。

（3）最后的除数 y 就是最大公约数。

```
Private Sub Command1_Click()
  x = Val(Text1.Text)
  y = Val(Text2.Text)
  If x * y = 0 Then
    MsgBox "两个数都不能为 0！",  "错误！"
Exit Sub
  End If
  If x < y Then t = x: x = y: y = t
  Do
  r = x Mod y
  If r = 0 Then Exit Do
```

```
    x = y: y = r
  Loop
  Label3.Caption = "两数的最大公约数是：" & Trim(Str(y))
End Sub
```

此例中利用了 Exit Do 语句来退出 Do…Loop 循环语句。

【例 3-28】编写程序：为小学生随机出题，进行四则运算训练。要求自动产生 10 以内的随机整数，四种运算操作符也由随机函数产生。

（1）建立应用程序用户界面。在窗体中添加一个标签 Label1（显示题目）、一个文本框 Text1（输入计算结果）、一个命令按钮 Command1（计分）和一个列表框 List1（显示做题的对错），如图 3-26（a）所示。

（a）　　（b）

图 3-26　四则运算

（2）编写程序代码。

```
Option Explicit                          '在通用段强制变量声明
Dim Num1%, Num2%                         ' 两个操作数
Dim SExp$                                ' 存放产生的算术表达式
Dim Result!                              ' 计算机计算结果
Dim NOk%, NError%                        ' 统计计算正确与错误数
```

编写窗体的 Load 事件，随机产生两个操作数和运算符号：

```
Private Sub Form_Load()                  ' 通过产生随机数生成表达式
  Dim NOp%, Op As String * 1, t%         ' NOp 操作符代码
  Randomize                              '初始化随机数生成器
  Num1 = Int(10 * Rnd + 1)               '产生 1～10 之间的操作数 1
  Num2 = Int(10 * Rnd + 1)               '产生 1～10 之间的操作数 2
  NOp = Int(4 * Rnd + 1)                 '产生 1～4 之间的操作代码
  Select Case NOp
    Case 1                               '当 NOp=1 时，加法运算
      Op = "+"
      Result = Num1 + Num2               'Result 存放正确结果
    Case 2
```

```
      Op = "—"
      If Num1 < Num2 Then t = Num1: Num1 = Num2: Num2 = t
      Result = Num1 - Num2
    Case 3
      Op = "×"
      Result = Num1 * Num2
    Case 4
      Op = "÷"
      Do While Num1 Mod Num2 <> 0    '若不能整除，则产生下一组数
       Num1 = Int(10 * Rnd + 1)      '再次产生 1～10 之间的操作数 1
       Num2 = Int(10 * Rnd + 1)      '再次产生 1～10 之间的操作数 2
      Loop
      Result = Num1/Num2
   End Select
  SExp = Num1 & Op & Num2 & "="
  Label1 = SExp
End Sub
```

编写 Text1 的 KeyPress 事件，判断正确与否：

```
Private Sub Text1_KeyPress(KeyAscii As Integer)
  If KeyAscii = 13 Then
  If Val(Text1) = Result Then                          '检查计算是否正确
     List1.AddItem SExp & Text1 & Space(3) & "√ "  '计算正确
     NOk = NOk + 1
    Else
     List1.AddItem SExp & Text1 & Space(3) & "×"   '计算错误
     NError = NError + 1
    End If
    Text1 = ""                                         '下一个表达式生成
    Text1.SetFocus
    Form_Load
  End If
End Sub
```

程序运行结果如图 3-26（b）所示，请有兴趣的读者修改程序，美化界面，编写出更适合小学生使用的四则运算应用程序。

【例 3-29】百元买百鸡问题。假设小鸡每只 5 角，公鸡每只 2 元，母鸡每只 3 元。现有 100 元钱要求买 100 只鸡，编写程序找出所有可能购买的方案，程序执行结果如图 3-27 所示。

分析：根据题意，设母鸡有 x 只，公鸡有 y 只，小鸡有 z 只，列出方程：

$$x+y+z=100$$

$$3x+2y+z=100$$

方法一：用三重循环，分别表示母鸡、公鸡和小鸡最大可能买的数目。

```
Private Sub Command1_Click()
Dim x%, y%, z%, n&
 Cls
 n = 0
 Print "方法一：三重循环"
```

```
Print "母鸡", "公鸡", "小鸡"
For x = 0 To 33          '母鸡可能买的数量范围
   For y = 0 To 50       '公鸡可能买的数量范围
     For z = 0 To 100    '小鸡可能买的数量范围
       n = n + 1         '计数，求出循环的次数
       If 3 * x + 2 * y + 0.5 * z = 100 And x + y + z = 100 Then
         Print x, y, z
     End If
    Next z
   Next y
 Next x
 Print "共计算了", n, "次"
End Sub
```

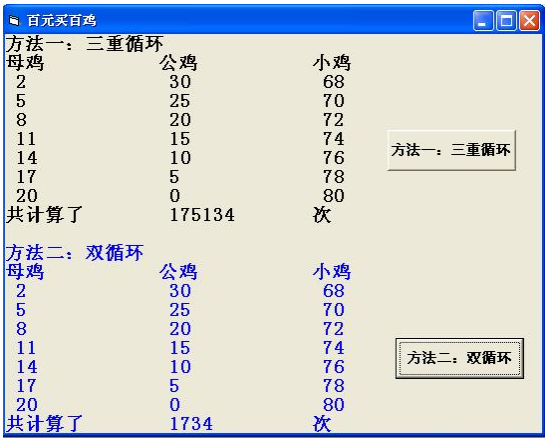


图 3-27　程序执行结果

方法二：用二重循环。

```
Private Sub Command2_Click()
Dim x%, y%, z%, n%
 'Cls
 Print
 Form29.ForeColor = RGB(0, 0, 255)
 Print "方法二：双循环"
 n = 0
 Print "母鸡", "公鸡", "小鸡"
 For x = 0 To 33
    For y = 0 To 50
      z = 100 - x - y      '小鸡的数量
      n = n + 1
      If 3 * x + 2 * y + 0.5 * z = 100 Then
         Print x, y, z
      End If
    Next y
  Next x
  Print "共计算了", n, "次"
End Sub
```

以上算法称为“穷举法”或“枚举法”，即利用计算机速度快的特点将可能出现的情况用循环结构逐一罗列，判断是否满足条件。

习题三

一、选择题

1．当文本框的（　　）属性为 True 时，Scroll 属性才有效。

A．Value　　B．MultiLine　　C．Index　　D．Tabindex

2．当某一按钮的（　　）属性设置为 False 时，该按钮不可见。

A．Enable　　B．Default　　C．Visible　　D．Cancel

3．通常，文本框的 SetFocus 方法不能使用在（　　）事件中。

A．Form_Click　　B．Form_Load　　C．Command_Click　　D．Label_click

4．下列语句可以将变量 X、Y 的值互换的是（　　）。

A．X=Y:Y=X　　B．T=X:Y=X:X=T

C．T=Y:Y=X:X=T　　D．X=T:T=X:Y=T

5.下列程序段求两个数中的最大数 max，不正确的是（　　）。

A．max=IIf(x>y,x,y)　　B．If x>y Then max=x　Else max=y

C．max=x　　D．If y>=x Then max=y

　　If y>=x Then max=y　　　　max=x

6．当 x=-5 时，下列语句执行后 y 的值是（　　）。

y=IIf(x>0,x^2+1,x-1)

A．-6　　B．26　　C．0　　D．4

7.“x 是小于 50 的非负数”，用 VB 表达式表示正确的是（　　）。

A．0<=x<50　　B．0<=x And x<50

C．0<=x<50　　D．0<=x Or x<50

8．以下叙述正确的是（　　）。

A．Do…While 语句构成的循环不能用其他循环语句代替

B．Do…While 语句构成的循环可以用 Break 语句退出

C．用 Do…While 语句构成的循环，在 While 后的条件不成立时结束循环

D．用 Do…Loop Until 语句构成的循环，在 Until 后的条件不成立时结束循环

9．关于语句 If x=1 Then y=1，下列说法正确的是（　　）。

A．x=1 和 y=1 均是赋值语句

B．x=1 和 y=1 均为关系表达式

C．x=1 为关系表达式，y=1 为赋值语句

D．x=1 为赋值语句，y=1 为关系表达式

10．循环嵌套应遵循的原则是（　　）。

A．内、外循环控制变量不能重名

B．内、外循环不能交叉

C．不能从循环体外跳到循环体内

D．以上都对

11．假定有以下循环结构：

```
Do Until <条件表达式>
   <循环体>
Loop
```

则以下描述正确的是（　　）。

A．如果“条件表达式”的值是 0，则一次循环体也不执行

B．如果“条件表达式”的值不是 0，则至少执行一次循环体

C．不论“条件表达式”的值是否为“真”，至少要执行一次循环体

D．如果“条件表达式”的值恒为 0，则无限次执行循环体

12．在窗体上画一个命令按钮，然后编写如下事件过程：

```
Private Sub Command1_Click()
   Dim I, Num
   Randomize
   Do
     For I = 1 To 1000
        Num = Int(Rnd * 100)
        Print Num
          Select Case Num
              Case 12
                 Exit For
              Case 58
                 Exit Do
              Case 65, 68, 92
                 End
          End Select
      Next I
     Loop
  End Sub
```

上述事件过程执行后，下列描述中正确的是（　　）。

A．Do 循环执行的次数为 1000 次

B．在 For 循环中产生的随机数小于或等于 100

C．当所产生的随机数为 12 时结束所有循环

D．当所产生的随机数为 65、68 或 92 时窗体关闭、程序结束

13．在窗体上画一个名为 Text1 的文本框和一个名为 Command1 的命令按钮，然后编写如下事件过程：

```
Private Sub Command1_Click()
Dim i As Integer,n As Integer
For i =0 To 50
   i=i+3
   n=n+1
   If i >10 Then Exit For
Next
Text1.Text=Str(n)
End Sub
```

程序运行后，单击命令按钮，在文本框中显示的值是（　　）。

A．3　　B．4　　C．5　　D．2

二、填空题

1．结构化程序设计的三种基本结构是_____、_____、_____。

2．VB 的赋值语句既可以给_____赋值，也可以给对象的_____赋值。

3．在 VB 中，用于产生一个消息框的语句或函数是_____，用于输入框的函数为_____。

4．循环语句 Do…Loop Until<条件>，当循环条件_____时退出循环。

5．在 For…Next 循环语句中，用_____语句中途退出循环。

6．下列程序段运行后显示的结果是_____。

```
Dim x
 x = Int(Rnd) + 5
 Select Case x
   Case 5
      Print "优秀"
   Case 4
      Print "良好"
   Case 3
      Print "通过"
   Case Else
      Print "不通过"
End Select
```

7．要使下列 For 语句循环执行 20 次，请填写循环变量的初值。

```
For k=_____To  -5  Step  -2
```

8．设有以下的循环，要求程序运行时执行 3 次循环体，请填空。

```
x=1
DO
   x=x+2
   Print x
Loop Until_____
```

三、程序填空题

1．下列程序的功能是计算 f=1-1/(2*3)+1/(3*4)-1/(4*5)+…+1/(19*20)。

```
Private Sub Form_Click()
  【1】
 f = 1
  【2】
  f = f + sign/(i * (i + 1))
  【3】
 Next i
 Print "f="; f
End Sub
```

2．输出 100 以内的所有 9 的倍数，并计算这些数的和。

```
Private Sub Form_Click()
Dim i As Integer, s As Integer
   s =【1】
   For i = 1 To 100
```

```
    If  【2】  Then
      s = 【3】
      Print  i
    End If
  Next  i
  Print  s
End  Sub
```

3．某大奖赛，有 7 个评委打分，对任意一位参赛者，输入 7 个评委的分数，去掉一个最高分、一个最低分后求出其余 5 个评委的平均分为该参赛者得分。

```
Private Sub Command1_Click()
  Dim mark!, aver!, i%, Max!, Min!
  aver = 0
  For i = 1 To 7
  mark = InputBox("请输入" & i & "位评委的打分")
  If i = 1 Then
    Max = mark: 【1】
  Else
    If mark < Min Then
       【2】
    ElseIf mark > Max Then
       【3】
    End If
  End If
  aver = 【4】
 Next i
 aver = 【5】
 Print aver, Min, Max
End Sub
```

4．找出能被 3、5、7 除余数为 1 的最小的 5 个正整数。

```
Private Sub Command1_Click()
  Dim countN%, n%
  countN = 0
  n = 1
  Do
    n = n + 1
    If 【1】 Then
      Print n
      countN = countN + 1
    End If
  Loop 【2】
End Sub
```

四、程序阅读题

1．运行下面的程序，单击窗体后屏幕上显示的结果是什么？

```
Private Sub Form_Click()
Dim x, y As Integer
x = 1: y = 0
Do While x < 3
   y = y + x
```

```
    x = x + 1
Loop
Print x, y
End Sub
```

2．运行下面的程序，单击窗体后屏幕上显示的结果是什么？

```
Private Sub Form_Click()
Cls
Print
For n = 1 To 4
 Print Tab(n + 20);
  For m = 1 To 10
   Print Spc(1); "*";
  Next m
  Print
Next n
End Sub
```

3．写出下列程序在窗体上的运行结果。

```
Private Sub Form_Click()
x = 13
If x Mod 2 = 0 Then
  s = 0
Else
  s = 1
End If
Print "s="; s
End Sub
```

4．写出下列程序单击窗体后的运行结果。

```
Private Sub Form_Click()
    c=1234
    c1=Trim(Str(c))
    For i=1 to 4
      Print right(c1,i)
    Next
   End Sub
```

5．在窗体上画一个命令按钮和一个标签，其名称分别为 Command1 和 Label1，有如下的事件过程：

```
Private Sub Command1_Click()
   conter = 0
   For i = 1 To 4
      For j = 6 To 1 Step -2
        conter = conter + 1
      Next j
   Next i
   Label1.Caption = Str(conter)
End Sub
```

程序运行后，单击命令按钮，标签中显示的内容是什么？