第 1 章　VB 6.0 数值计算程序设计基础

1.1　VB 应用程序的基本结构

利用 VB 开发应用程序时，通过工程来管理构成应用程序的所有不同文件。一个工程主要包括：

（1）跟踪所有部件的工程文件（.vbp）；

（2）窗体文件（.frm）；

（3）标准模块文件（.bas）；

（4）类模块文件（.cls）；

（5）资源文件（.res）。

1.1.1　模块及按模块划分的层次结构

Visual Basic 中的程序代码存储在模块中，模块是相对独立的程序单元。VB 系统提供的模块包括窗体模块、标准模块、类模块。三种模块都可以包含声明和过程，它们形成了如图 1.1 所示工程的模块层次结构。

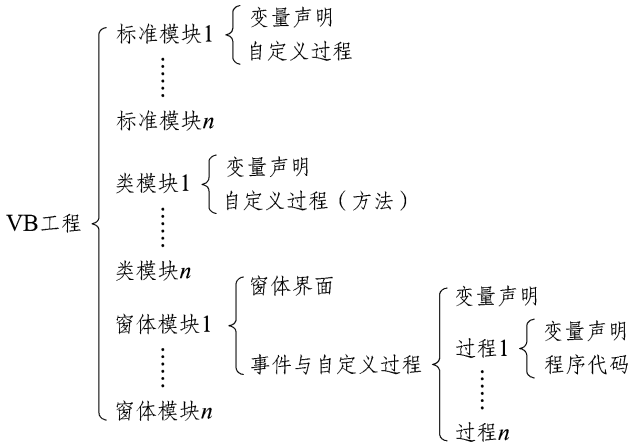

图 1.1　VB 应用程序的组织结构

1. 窗体模块

窗体模块包含窗体及其控件的属性设置、事件过程（代码部分）、窗体内自定义过程的窗体级声明等。每个窗体对应一个窗体模块，其文件扩展名为.frm。

一个 VB 应用程序至少应包含一个窗体模块。每个窗体模块包含菜单、命令按钮等控件。每个控件都有一个对应的事件过程，事件过程中有响应该事件执行的程序段。除了事件过程，窗体模块还可包含通用过程、函数过程，这些过程对来自该窗体中任何事件过程、通用过程或函数过程的调用指令作出响应。

2. 标准模块

简单的应用程序可以只有一个窗体，所用的程序都驻留在窗体模块中，而当应用程序比较复杂时，往往会采用多个窗体。复杂多窗口应用程序可能存在几个窗体模块共同执行的代码，为了在各窗体中不产生重复代码，VB 系统提供一种独立的模块，用于保存公用程序代码，该独立模块就称为标准模块。

标准模块是 VB 程序中的一个独立模块，包含模块级或全局变量声明、函数过程和通用过程，标准模块的文件扩展名为.bas。一个应用程序可以没有标准模块，也可以包含多个标准模块。

为了提高开发效率，通常将那些与特定窗体或控件无关的代码放入标准模块，写入标准模块的代码不与特定应用程序绑定。标准模块中除包含允许其它模块访问的过程，还包括变量、常量、数据类型、自定义过程的全局或模块级声明。

（1）使用标准模块。

在缺省状态下，变量对于过程是局部的，即仅能在创建这些变量的过程中读取或者修改。与之相似，过程对于创建它们的窗体来说也是局部的。为了在工程中的所有窗体中共享变量和过程，需要在该工程的一个或多个标准模块中对它们进行声明和定义。

正如窗体一样，标准模块被单独列在 Project（工程）窗口内，并可通过使用 File（文件）菜单中的“Save Module As”菜单项存盘。与窗体模块不同的是，标准模块不包含对象及其属性设置，而只包含可在代码窗口中显示和编辑的代码。

（2）创建标准模块。

在工程中创建一个空的标准模块的步骤如下：

① 启动 VB 打开一个新的标准工程，单击工程菜单中的“添加模块”菜单项，选择“新建”并单击“打开”按钮。执行上述操作后，VB 即在工程中增加一个默认名为“Module1”的标准模块，并且模块代码窗口被自动打开，可在窗口中编辑修改程序代码。

② 在“文件”菜单中，单击“保存 Module1”。

3. 类模块

类模块是面向对象编程的基础，VB 系统允许通过在类模块中编写代码建立新对象，这些新对象可以包含自定义的属性和方法。每个类模块只能定义一个对象。类模块定义的对象不可视，应用需要通过声明对象型变量的方法。

类模块与标准模块的区别在于：标准模块仅包含代码，而类模块既含代码又含数据，类模块可视为没有物理表示的对象。

VB 中对象是用类定义的，工具箱上每个控件都是一个类，但在窗体上引用一个控件之前，以该控件命名的对象是不存在的。具体的、可以引用的对象实际上是类的一个拷贝或实例。

类与过程有共同之处，但有本质的区别：过程是将逻辑上有关的语句与数据集合在一起，主要用于执行；而类则是逻辑上有关的过程及其数据的集合，主要不是用于执行而是提供所需的资源。

1.1.2 VB 应用程序运行流程

VB 应用程序呈层次结构，典型应用程序包括若干个模块：应用程序中每个窗体的窗体模块、共享代码的标准模块和自定义对象的类模块。每个模块包含若干含有代码的过程，过程分为三类：事件过程、通用过程、函数过程。过程是划分 VB 代码的最小单元，每个过程是一个可执行的代码片段。

VB 程序的运行通过事件来驱动，程序运行的流程完全取决于事件发生与否及发生的顺序。VB 定义了众多的事件，用户程序设计者通常只需对所选择的事件设计一段响应程序（事件过程），由用户操作对象驱动相应的事件发生来完成设定的功能，或由事件过程中的指令调用通用、函数过程来执行指定的操作。事件指窗口或控件能识别的活动，通常发生在用户与应用程序交互时，但也有一些事件由系统自行产生，如计时器事件。

事件驱动是图形操作界面程序设计的本质，即用户控制事件产生，而代码做出反应。事件在 Visual Basic 中是指由 IDE 或者系统指定的，能够被窗体或控件所响应和识别的动作。例如，Form_Load 就是一个事件，是指窗口在内存中加载完毕后，所触发的一个动作。事件可分为“用户事件”和“系统事件”。例如，鼠标点击事件 MouseDown 就是一个用户事件；时钟控件 Timer 是一个典型的系统事件。一般来说，应用程序中最早触发、必然发生的事件是 Form_Load 事件。

事件的命名是 VB 系统设定的，命名格式为：

```
Private Sub 控件名_事件名（）
```

例如：Private Sub Command1_Click（）是指在控件（按钮）Command1 上发生点击事件。

1. 典型事件

(1)窗体和图像框类事件。

Paint 事件：当某一对象在屏幕中被移动，改变尺寸或清除后，程序会自动调用 Paint 事件。注意：当对象的 AutoDraw 属性为 True(-1)时，程序不会调用 Paint 事件。

Resize 事件：当对象的大小改变时触发 Resize 事件。

Load 事件：仅适用于窗体对象，当窗体被装载时运行。

Unload 事件：仅适用于窗体对象，当窗体被卸载时运行。

(2)当前光标(Focus)事件。

GotFocus 事件：当光标聚焦于该对象时发生事件。

LostFocus 事件：当光标离开该对象时发生事件。

注意：Focus 英文为“焦点”、“聚焦”之意。最直观的例子是：设有两个窗体，一个窗体被另一窗体部分遮盖。当点击下面的窗体时，其即被激活并全部显示出来，这就是 GotFocus 事件；而另外一个窗体则被遮盖，并且标题条变灰，称为 LostFocus 事件。

(3)鼠标操作事件。

Click 事件：鼠标单击事件。

DblClick 事件：鼠标双击事件。

MouseDown、MouseUp 事件：鼠标键按下/放开事件。

MouseMove 事件：鼠标移动事件。

DragDrop 事件：拖放事件，相当于 MouseDown、MouseMove 和 MouseUp 的组合。

DragOver 事件：鼠标在拖放过程中就会产生 DragOver 事件。

(4)键盘操作事件。

KeyDown、KeyUp 事件：键盘按键的按下/放开事件。

KeyPress 事件：键盘按键事件。

(5)改变控制项事件。

Change 事件：当对象的内容发生改变时，触发 Change 事件，最典型的例子是文本框(TextBox)的 text 属性值被改变时，就会触发 Change 事件。

DropDown 事件：下弹事件，仅用于组合框(ComboBox)对象。

PathChange 事件：路径改变事件，仅用于文件列表框(FileBox)对象。

2. 其他事件

Timer 事件：仅用于计时器，每隔一段时间被触发一次。

一个对象的 Enable 属性为 False 时，用户不能通过鼠标或键盘操作，仍可通过程序控制。

1.1.3 过程中代码执行流程控制

封装在事件、通用或函数过程中的程序代码，执行顺序是三种基本程序结构的组合：

顺序结构：程序按语句顺序由上而下逐句执行。

选择结构：程序按设定的条件实现程序语句的选择执行。

循环结构：程序按给定的条件重复地执行设定的程序段或过程。

1. 顺序结构

从上往下按顺序执行的语句结构。

2. 选择结构

根据设定的条件分析，比较和判断，选择性地执行不同的程序代码。

(1) If 语句结构。

If 语句分单分支结构、双分支结构和多分支结构三种情况。If 语句的“条件”是一个逻辑表达式，VB 系统根据条件判断返回 True 或 False，选择执行不同的程序语句块。

① 单分支结构(图 1.2)。

其语法结构为：

```
If 条件 Then
   语句或语句块
End If
```

② 双分支结构(图 1.3)。

其语法结构为：

```
If 条件 Then
   语句或语句块 1
Else
   语句或语句块 2
End If
```

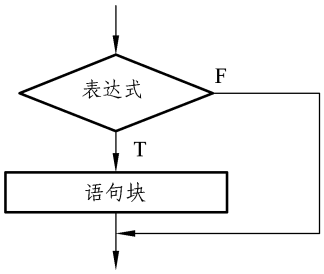

图 1.2　单分支结构

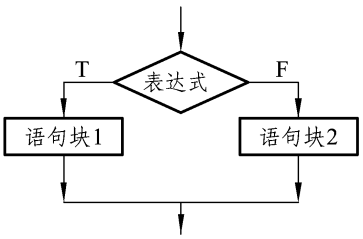

图 1.3　双分支结构

③ 多分支结构(图 1.4)。

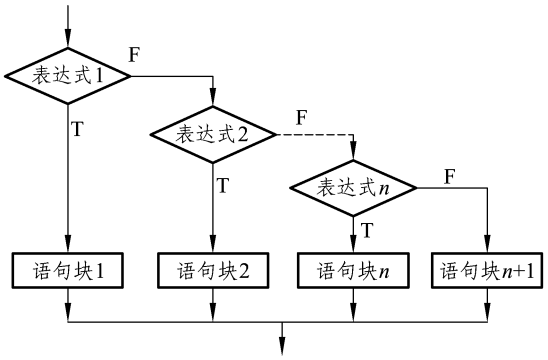

图 1.4　多分支结构

其语法结构为：

```
If 条件 1 Then
   语句或语句块 1
Else If 条件 2 Then
   语句或语句块 2
     …
Else
   语句或语句块 n
End If
```

（2）Select Case 语句。

Select Case 语句首先判断一个表达式的值，然后根据表达式的值选择执行不同的程序语句块。

其语法结构如下：

```
Select Case 表达式
    Case 表达式 1
        语句或语句块 1
    Case 表达式 2
        语句或语句块 2
    …
    Case Else
        语句或语句块 n
End Select
```

Select Case 的执行过程是：在 Select Case 关键字后的测试条件中计算测试表达式，然后 VB 将表达式的值与结构中每一个 Case 关键字后的值进行比较，若相等就执行与该 Case 相关联的语句块。

在 Select Case 结构中，Case 关键字后表达式可以是几个值的列表，各值之间用逗号分隔。

如果有多个 Case 关键字后表达式值与测试表达式值匹配，则只执行第一个匹配的 Case 关键字后语句块。

Select Case 与 If...Then...Else 结构的区别在于 ,Select Case 结构只在开始处计算测试条件的值，而 If...Then...Else 结构为每个 Else If 语句计算不同的表达式。因此在处理多重选择问题时 ,If...Then...Else 结构能适应更复杂的选择性 ,而 Select Case 结构更具可读性并且执行效率更高。

3. 循环结构

循环结构是数值计算程序中最重要的结构之一，在复杂数值计算程序中起着不可替代的作用。循环结构是在给定条件成立时，反复执行某程序段，直到条件不成立为止。给定的条件称为循环条件 ,反复执行的程序段称为循环体。在 VB 中提供了多种循环语句供用户使用，下面介绍两种常用的形式：For 语句和 Do 语句。

（1）For 语句（计次循环语句）。

For 循环（For…Next）语句的基本格式为：

```
For 循环变量 = 初值 To 终值 Step 增量
    循环体
Next 变量
```

其中循环变量为数值型变量，初值、终值、增量均为数值型变量。“Step”部分可以省略，缺省时默认为 1。在循环体中使用 Exit For 语句可直接中止循环，跳出循环并执行 Next 后面的语句。正常循环结束后跳出循环时，循环变量值为循环变量终值 + 增量值，使用 Exit For 语句终止循环时，循环变量保持退出时的值。

For 循环的执行过程如下：

① 将循环变量赋初值。

② 比较循环变量是否小于或等于终值（如 Step 后增量值为负值，则比较是否大于或等于终值）,小于或等于（大于或等于）时循环条件成立 ,运行循环体中的语句 ,若条件不成立，结束循环执行 Next 后的语句。

③ 循环体执行完毕后至 Next 语句，循环变量增加增量值，转到②继续比较执行。

（2）Do 语句。

For…Next 循环语句用于循环次数确定的循环问题，对于循环次数未知的循环问题，Visual Basic 设置了 Do 循环语句。

① 第一种循环结构（图 1.5）。

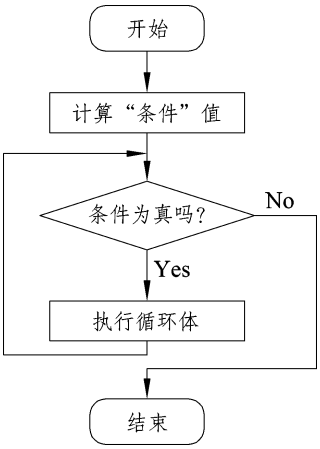

图 1.5　不计次循环结构（前测型）

其语法结构如下：

```
Do{While | Until}<条件>
    <语句块>
    [Exit Do]
    <语句块>
Loop
```

② 第二种循环结构（图 1.6）。

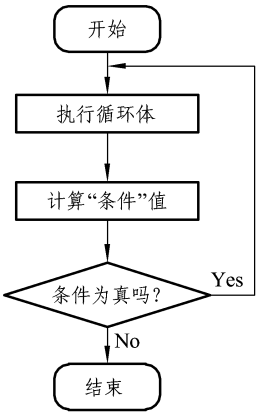

图 1.6　不计次循环结构（后测型）

其语法结构如下：

```
Do
    <语句块>
    [Exit Do]
    <语句块>
Loop{While | Until}<条件>
```

功能：当关键字 While 或关键字 Until 后条件为真（True）时，执行循环体。

说明：第一种格式为先判断后执行，有可能一次也不执行；第二种格式为先执行后判断，至少执行一次循环体；

Exit Do：类似于 Exit For 语句，执行该语句后，退出循环执行 Loop 后的语句。

说明：上述两种 Do 循环是以关键字 while 为例,条件成立进入循环；若选用关键字 until，则是条件成立时，结束循环。

1.2 VB 应用程序的对象

1.2.1 对象的概念

1. 对 象

将数据和处理数据的过程打包在一起而生成的新型数据类型称之为对象。对象中的数据称之为“属性”, 过程称之为“方法”。所有的窗体和控件都是对象。访问对象的属性和方法是在对象和属性、方法之间加一个“.”号。

```
Label1.caption = "Name"  访问属性
Text1.SetFocus  访问方法
```

2. 对象的操作

（1）访问对象属性。

访问对象的属性有两种方法，以标签控件 Label1 为例：

① 在属性栏里面直接定义。

设置标签 Label1 的 Caption 属性为 Visual Basic（图 1.7）。

图 1.7　属性栏

② 通过程序代码赋值。

在窗体加载事件 Form_Load 中写下代码：

```
Label1.Caption = "Visual Basic"
```

运行结果见图 1.8。

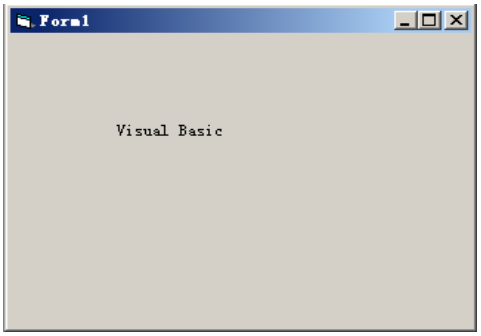

图 1.8　访问 Label1.Caption 属性运行结果

（2）访问对象的方法。

方法是 VB 为对象预设的、用户不能更改的事件过程，不同的对象具有不同的方法。访问对象方法的语法结构为：

```
对象名.方法名（[参数列]）
```

例如：在窗口 Form1 上显示字符“Hello World”的语句为：

```
Form1.Print "Hello World"
```

其中，对象为 Form1，方法为 Print，参数为“Hello World”。

3. 对象型变量

对象型变量是用于保存对象的变量，所谓保存对象，实际上是保存对象的内存地址（句柄）。和使用普通变量类似，对象型变量通过声明和赋值，就可以像普通变量一样操作。例如：

```
Dim obj as Command ' 声明 obj 为 Command 对象型变量
Set obj = Command1 ' 对象型变量赋值
obj.Caption = "ok" ' 访问对象型变量的属性
```

也可以清除对象型变量：

```
Set obj = Nothing
```

简单地说，对象型变量就是被声明成具体对象类型，并可以赋值于已定义对象（控件）的特殊变量。

4. VB 常用控件

控件是可视或具有图标的对象，如 Picture、Command、TextBox 等，VB 将常用的控件置于工具箱中，方便用户引用。常用控件主要有：

① 文本框 TextBox：是窗口中进行输入、输出操作的重要控件，文本框本身支持一般的文字编辑功能，这些功能已由 VB 封装在文本框对象中了。

方法：Move。

事件：Click、DblClick、Change 事件。注意在 Change 事件（文本框中文本内容发生改变）中，不可有改变自身 Text 属性值的语句。

② 标签 Label：主要用于对没有标题的控件进行说明，显示内容（caption 属性值）不可直接编辑，但可通过程序代码修改。

方法：Move。

事件：Click、DblClick、 Change 事件。标签显示内容发生改变时触发 Change 事件。

③ 图像控件 Image：用于在窗体上显示保存在图形文件中的图像。

Picture 属性：决定图像来源，可直接设定，也可在程序运行中用内部函数 LoadPicture 赋值，例如：

```
image.picture = loadpicture( "C:\windows\setup.bmp")
```

设计时赋予 Picture 属性的图形文件会被复制到二进制窗体文件（.frx）中，运行时不依赖原文件。

方法：Move。

事件：Click、DblClick 事件。

④ 图片框控件 PictureBox：与图像控件 Image 功能基本相同，在平差示例程序中，被用来容纳快捷按钮控件和绘制控制网图。

方法：Move。

事件：Change 事件。

⑤ 滚动条控件（HscrollBar/VscrollBar）：用于上下、左右的滚动文字或图形。

方法：Move。

事件：Change、Scroll 事件。前者由滚动条的 Value 值变化激发，后者在滚动框被拖动时引发。

滚动条有几个特殊的属性：

Value 属性：当滑块挪动后停在某一位置时，就改变了滚动条的 Value 属性。

Max 和 Min 属性：用来设置滚动条的最大值和最小值，譬如把 Max 设为 100，把 Min 设为 0 时，滚动条的 Value 就在 0 到 100 之间移动。当 Max 和 Min 倒置时，滚动条的方向也会相反，即当 Max < Min 时，原来 Value 增加的方向就会变成相反的方向。Max 和 Min 也可设为负值。

SmallChange 和 LargeChange 属性：当鼠标在滚动条两头的箭头上单击时，就会把滚动条向相应的方向挪动一点，这时用到的就是 SmallChange 属性。当把 SmallChange 属性设为 1 时，每点一下，滚动条的 Value 就改变 1；当鼠标在滚动条上点击时，用到的就是 LargeChange 属性，当设为 10 时，每点击一下滚动条，Value 就改变 10。

⑥ 框架控件 Frame :是左上角有标题的方框 ,目的是对窗体上的控件进行视觉上的分组，使窗体上的内容更有条理。

方法：Move。

事件：Click、DblClick 事件。

⑦ 复选框控件 CheckBox：提供选择项的控件，一个窗体可有多个复选框，并按功能进行分组，一组中可选中多个，也可一个不选。Value 属性值决定复选框的选中状态。

方法：Move。

事件：Click 事件。

⑧ 单选框控件 OptionBox：单选框控件与复选框的差别是，只能且必须有一个选项被选中。直接放在窗体上的所有单选框，无论其相互位置如何都被认为是一组。要在一个窗体上放置多个单选框，可使用容器控件——图片框或框架，其中框架较为常用。

方法：Move。

事件：Click、DblClick 事件。

⑨ 列表框控件 ListBox：同复选框、单选框一样，列表框也是提供选项的控件。

方法：

- AddItem 方法：往列表框中添加新条目。语法：

列表框对象名.AddItem 字符表达式[，序号]

- RemoveItem 方法：从列表框中指定序号位置删除条目。语法：

```
列表框对象名.RemoveItem 序号
```

- Clear 方法：清除列表框所有条目。语法：

```
列表框对象名.Clear
```

- Move 方法：移动列表框或改变其大小。语法：与其他控件相同。

事件：

- Click、DblClick 事件。
- Scroll 事件。
- ItemCheck 事件：ListBox 空间的 Style 属性设置为 1 时，若控件中一个条目复选框被选定或消除，则该事件发生。事件过程语法为：

```
Private Sub 列表框对象名.ItemCheck(Item As Integer)
```

⑩ 组合框控件 ComboBox：组合框可看作文本框和列表框的组合体，具有两者的事件和方法。

组合框不支持多选，故没有 MultiSelect，Selected，SelCount 属性，也无 ItemCheck 事件。

5. 控件数组

控件数组是一组具有相同名称和类型的控件，它们的事件名称也相同，即同类的控件若具有相同的 Name 属性，就称为控件数组。创建控件数组可以采用对一个控件做复制粘贴操作实现，也可以将已引用同类型控件名改为同一 Name 属性来完成。另外在菜单编辑器中，将某一菜单下的下拉子菜单名设置为同样的名称，这些子菜单也构成控件数组。

VB 还允许在程序运行状态下，通过程序语句创建控件数组，但是只支持在已有控件的基础上，新增控件数组成员。程序语句为：

```
Load object (index%)
Onload object (index%)
```

通过命令语句加载控件数组新成员时，新成员大多数属性值将由现有数组成员中 Index 属性值最小的控件复制。但是 VB 不会将 Visible、Index 等属性值复制到新控件成员，因此为了使新增控件成员可视，必须通过命令将属性值 Visible 设置为 True。

Onload 命令只能删除 Load 语句创建的控件，不能删除设计时在对象窗口创建的控件。

控件数组可以共享同一个事件过程，使用控件数组并以控件的 Index 属性值为选择条件，结合选择执行结构语句 Select Case，可将本需要封装在不同命令按钮或子菜单点击事件中的程序代码组合在一起，显著地简化程序结构。

【例 1】 设在一个窗体中有若干个 Command 按钮控件组成的数组，其 Index 属性值从 1 开始编号。现要让用户单击某按钮时，弹出一个窗口显示其 Index 属性值的平方。若不使用控件数组，则要设 Command 1，Command 2，…，Command *n*，*n* 个 Click 事件，并重复设计

程序代码因而非常烦琐。而采用控件数组，则只需采用下面一个事件即可完成。

```
Private Sub Commands1_Click(Index As Integer)
   MsgBox  Index^2
EndSub
```

控件数组的另一个功能是可使用 Index 属性循环访问控件组。例如，若要清除控件 TextBox 中各成员的 text 属性值，可将 Index 属性作为循环变量，在一个循环内访问控件数组全部成员。

控件数组中的成员可位于不同容器上，例如，两个不同 Form 控件上包含的 TextBox 控件，却是同一控件数组的元素。

【例 2】 使用控件数组，设计一个能进行加、 减、乘、除运算的运算器程序（程序界面见图 1.9)。

图 1.9 计算器程序界面

程序代码：

```
Dim Op As Integer,First As String ' Op 存放命令按钮序号,First 存放第一个操作数
Private Sub Command1_Click(Index As Integer) ' 数字按钮处理过程
   If Index = 10 Then
      Text1.Text = Text1.Text & "." ' 输入小数点"."
   Else
      Text1.Text = Text1.Text & CStr(Index) ' 将输入的数字转化为字符串并连接到
      ' 文本框中，各数字按钮 Index 参数值与按钮上的数字相同
   End If
   If Len(Text1.Text) = 2 And Left(Text1.Text, 1) = "0" And Mid(Text1,
```

```
    2, 1) <>
    "." Then ' 条件成立是只有两个字符，后一位不是小数点，而前一位是"0"
        Text1.Text = Mid(Text1.Text, 2) ' 去掉字符"0#"中的"0"
    End If
End Sub

Private Sub Command2_Click(Index As Integer) ' 命令按钮处理过程
    First = Val(Text1.Text) ' 将第 1 个操作数输入 First 变量
    Op = Index ' 将命令按钮序号输入 Op 变量
    Text1.Text = "" ' 文本框清空
End Sub

Private Sub Command3_Click() ' 清除文本框事件过程
    Text1.Text = ""
    First = 0
End Sub

Private Sub Command4_Click() ' 等号 "=" 键事件处理程序
    Dim Sec As Single ' 定义存放第 2 个操作数变量
    Sec = Val(Text1.Text)
    Select Case Op
        Case 0 ' 加法处理
            Text1.Text = Str(Format(First + Sec, "#######.##"))
        Case 1 ' 减法处理
            Text1.Text = Str(Format(First - Sec, "#######.##"))
        Case 2 ' 乘法处理
            Text1.Text = Str(Format(First * Sec, "#######.##"))
        Case 3 ' 除法处理
            If Sec <> 0 Then
               Text1.Text = Str(Format(First / Sec, "#######.##"))
            Else
               Text1.Text = " 除数为 0"
            End If
        Case 4 ' 取余处理
            If Sec <> 0 Then
               Text1.Text = Str(Format(First Mod Sec, "#######.##"))
```