第5章 VBA 应用基础

VBA（Visual Basic for Applications）是 Microsoft Office 集成办公软件的内置编程语言，是新一代标准宏语言。它基于 VB（Visual Basic）发展起来，与 VB 有很好的兼容性。VBA 寄生于 Office 应用程序，是 Office 的重要组件，利用它可以将烦琐、机械的日常工作自动化，从而极大提高用户的办公效率。

VBA 与 VB 主要有以下区别：

（1）VB 用于创建标准的应用程序，VBA 是使已有的应用程序（Office）自动化。

（2）VB 具有自己的开发环境，VBA 寄生于已有的应用程序（Office）。

（3）VB 开发出的应用程序可以是可执行文件（EXE 文件），VBA 开发的程序必须依赖于它的父应用程序（Office）。

尽管存在这些不同，VBA 和 VB 在结构上仍然十分相似。如果已经掌握 VB，就会发现学习 VBA 非常容易。反过来，学完 VBA 也会给学习 VB 打下很好的基础。

用 VBA 可以实现如下功能：

（1）使重复的任务自动化。

（2）对数据进行复杂的操作和分析。

（3）将 Office 作为开发平台，进行应用软件开发。

用 Office 作为开发平台有以下优点：

（1）VBA 程序只起辅助作用。许多功能 Office 已经提供，可以直接使用，简化了程序设计。例如，打印、文件处理、格式控制和文本编辑等功能不必另行设计。

（2）通过宏录制，可以部分地实现程序设计的自动化，大大提高软件开发效率。

（3）便于发布。只要发布含有 VBA 代码的文件即可，不需要考虑运行环境，因为 Office 是普遍配备的应用软件。不需要安装和卸载，不影响系统配置，属于绿色软件。

（4）Office 界面对于广大计算机应用人员来说比较熟悉，符合一般操作人员的使用习惯，便于软件推广应用。

（5）用 VBA 编程比较简单，即使非计算机专业人员，也可以很快编出自己的软件。而且 Office 应用软件及其 VBA 内置了大量函数、语句、方法等，功能非常丰富。

在 Office 2016 各个应用程序（如 Word、Excel、PowerPoint 等）中使用 VBA 的方式相同，语言的操作对象也大同小异。因此，只要学会在一种应用程序（如 Excel）中使用 VBA，就能在其他应用程序中使用 VBA。

本章只介绍在 Excel 环境下 VBA 的应用，包括宏的录制、编辑与使用，VBA 语法基础，过程以及面向对象程序设计的有关知识。

5.1　用录制宏的方法编写 VBA 程序

宏（Macro）是一组 VBA 语句。可以理解为一个程序段，或一个子程序。在 Office 2016 中，宏可以直接编写，也可以通过录制形成。录制宏，实际上就是将一系列操作过程记录下来并由系统自动转换为 VBA 语句。这是目前最简单的编程方法，也是 VBA 最具特色的地方。用录制宏的办法编制程序，不仅能简化编程过程，还可以提示用户使用什么语句和函数，帮助用户学习程序设计。当然，实际应用的程序不能完全靠录制宏，还需要对宏进一步加工和优化。

5.1.1　准备工作

1. 在功能区中显示“开发工具”选项卡

在 Excel 2016 中，为了使用 VBA，需要在功能区中显示“开发工具”选项卡，因为在默认情况下，不会显示这个选项卡。用以下方法可以将“开发工具”选项卡添加到功能区中：

（1）在“文件”选项卡中选择“选项”命令，在“Excel 选项”对话框选择“自定义功能区”项；或者右击功能区，在弹出的快捷菜单中选择“自定义功能区”命令。

（2）在对话框“自定义功能区”选项组的“主选项卡”列表框中选中“开发工具”复选框。

2. 设置宏安全性

有一种计算机病毒叫作“宏病毒”，它是利用“宏”来传播和感染的病毒。为了防御这种计算机病毒，Office 软件提供了一种安全保护机制，即设置“宏”的安全性。

在“开发工具”选项卡的“代码”选项组中单击“宏安全性”按钮，即可在弹出的对话框中设置不同的安全级别。安全级别越高，对宏的限制越严。

由于宏就是 VBA 程序，因此限制使用宏，实际上就是限制 VBA 代码的执行。这从安全角度考虑是应该的，但是如果这种限制妨碍了软件功能的发挥就值得考虑了。

其实，宏病毒只是众多计算机病毒的一种，可以同其他计算机病毒同样对待，用统一的防护方式和杀毒软件进行防治，而不必太在意 Office 本身“宏”的“安全性”。尤其是需要频繁使用带有 VBA 代码的应用软件时，完全可以把“宏设置”的安全性设置为“启用所有宏”；在“开发人员宏设置”选项组中选中“信任对 VBA 工程对象模型的访问”复选框。

5.1.2　宏的录制与保存

首先在 D 盘根目录下创建一个“照片”文件夹，将 8 个图片文件 0433101.jpg、0433102.jpg、…、0433108.jpg 复制到该文件夹。然后录制一个简单的宏，它的功能是在 Excel 工作表中选定单元格，设置单元格的行高和列宽，在单元格中插入图片并调整大小。步骤如下：

（1）启动 Excel，在“开发工具”选项卡的“代码”选项组中单击“录制宏”按钮。

（2）在“录制宏”对话框中输入宏名“插入图片”，单击“确定”按钮。此时，功能区

上的“录制宏”按钮切换为“停止录制”按钮。

（3）选定任意一个单元格（如 G1 单元格），在“开始”选项卡的“单元格”选项组中单击“格式”按钮。设置行高为 100、列宽为 12。

（4）在“插入”选项卡的“插图”选项组中单击“图片”按钮。在“插入图片”对话框中选择指定文件夹中的图片文件“0433101.jpg”并插入。

（5）右击图片，在弹出的快捷菜单中选择“大小和属性”命令。在“设置图片格式”对话框中取消“锁定纵横比”复选框的选择，设置高度为 3.5 厘米、宽度为 2.7 厘米，单击“关闭”按钮。

（6）单击“开发工具”选项卡“代码”选项组的“停止录制”按钮，结束录制宏过程。

注意：在录制宏之前，要计划好操作步骤和命令。如果在录制宏的过程中进行了错误操作，则错误的操作也会被录制。

要执行刚才录制的宏，可以在“开发工具”选项卡的“代码”选项组中单击“宏”按钮。在“宏”对话框中选择“插入图片”项，单击“执行”按钮。

在“录制宏”对话框中，可以指定将宏保存在当前工作簿、新工作簿或个人宏工作簿。

将宏保存在当前工作簿或新工作簿，则只有该工作簿被打开时，相应的宏才可以使用。

个人宏工作簿是为宏而设计的一种特殊的具有自动隐藏特性的工作簿。如果需要让某个宏在多个工作簿都能使用，就应当将宏保存到个人宏工作簿中。要将宏保存到个人宏工作簿，在“录制宏”对话框的“保存在”下拉列表中选择“个人宏工作簿”即可。

5.1.3 宏代码的分析与编辑

对已经存在的宏，可以查看代码，也可以进行编辑。

在“开发工具”选项卡的“代码”选项组中单击“宏”按钮。在“宏”对话框中选择“插入图片”，单击“编辑”按钮，进入 VB 编辑环境，显示出如下代码：

```
Sub 插入图片()
'
' 插入图片 宏
'

'
    Range("G1").Select
    Selection.RowHeight = 100
    Selection.ColumnWidth = 12
    ActiveSheet.Pictures.Insert("D:\照片\0433101.jpg").Select
    Selection.ShapeRange.LockAspectRatio = msoFalse
    Selection.ShapeRange.Height = 99.2125984252
    Selection.ShapeRange.Width = 76.5354330709
End Sub
```

上述代码包括以下几部分：

（1）宏（子程序）开始语句。

每个宏都以 Sub 开始，Sub 后面紧接着是宏的名称和一对括号。

（2）注释语句。

从单引号开始直到行末尾是注释内容。注释的内容是给人看的，与程序执行无关。

给程序加注释是应该养成的良好习惯，这对日后的维护大有好处。假如没有注释，即使是自己编写的程序，过一段时间以后，要读懂它也并非一件容易的事。

（3）实现具体功能的语句。

对照先前的操作，不难分析出各语句的功能：

Range("G1").Select 用来选定“G1”单元格。

Selection.RowHeight = 100 和 Selection.ColumnWidth = 12 用来设置选中单元格的行高和列宽。

ActiveSheet.Pictures.Insert("D:\照片\0433101.jpg").Select 的功能是在当前单元格中插入一个指定的图片。

Selection.ShapeRange.LockAspectRatio = msoFalse 取消图片的“锁定纵横比”复选框的选择。

Selection.ShapeRange.Height = 99.2125984252 和 Selection.ShapeRange.Width = 76.5354330709 用来设置图片的高度和宽度（以磅为单位）。

（4）宏结束语句。

End Sub 是宏的结束语句。

了解了代码中各语句的作用后，便可以在 VBA 的编辑器窗口修改宏。将前面的几行注释语句删除，再加入循环语句，将宏改为：

```
Sub 插入图片()
  For r = 1 To 8
    Range("G" & r).Select
    Selection.RowHeight = 100
    Selection.ColumnWidth = 12
    ActiveSheet.Pictures.Insert("D:\照片\043310" & r & ".jpg").Select
    Selection.ShapeRange.LockAspectRatio = msoFalse
    Selection.ShapeRange.Height = 99.2125984252
    Selection.ShapeRange.Width = 76.5354330709
  Next
End Sub
```

这里，在原来基础上做了 3 点改动：

（1）加入 For 循环语句，将原来的程序段作为循环体，使之能够被执行 8 次；

（2）将字符串常量"G1"改成由字符串连接运算符“&”、字符串常量"G"以及变量 r 所构成的字符串表达式"G" & r ，使得每次循环选定的单元格不同；

（3）用变量 r 的值作为图片文件名的最后一个字符。

再次运行“插入图片”宏，可以看到当前工作表的 G 列从 1 行到 8 行自动依次插入了 8 张图片，每个图片的大小都相同。

循环控制语句 For…Next 的语法形式如下：

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

循环语句执行时，首先为循环变量置初值。如果循环变量的值没有超过终值，则执行循环体，运行到 Next 时把步长加到循环变量上。若该值仍没有超过终值，则继续循环，直至循环变量的值超过终值时，才结束循环。

步长可以是正数、可以是负数，为 1 时可以省略。

遇到 Exit For 时，退出循环。

可以将一个 For…Next 循环放置在另一个 For…Next 循环中，组成嵌套循环。每个循环中要使用不同的循环变量名。下面的循环结构是正确的：

```
For I = 1 To 10
   For J = 1 To 10
      For K = 1 To 10
         ...
      Next K
   Next J
Next I
```

许多过程都可以用录制宏来完成。但录制的宏不具备判断或循环功能，人机交互能力差。因此，需要对录制的宏进行加工。

在“开发工具”选项卡的“代码”选项组中单击“Visual Basic”按钮，或用 Alt+F11 快捷键，可以直接打开 Visual Basic 编辑器。Visual Basic 编辑器也叫 VBE，实际上是 VBA 的编辑环境。

在 VBE 中可以编辑、调试和运行宏，也可以定义模块、用户窗体和过程。

如果要删除宏，可在“开发工具”选项卡的“代码”选项组中单击“宏”按钮，然后在“宏名”列表框中选定要删除的宏，再单击“删除”按钮。

5.1.4 用其他方式执行宏

除了用“开发工具”选项卡“代码”选项组的“宏”和在 Visual Basic 编辑环境中运行宏外，还可以用以下几种方式运行宏。

1. 用快捷键运行宏

在 Excel 中，可以在创建宏时指定快捷键，也可以在创建后再指定。录制宏时，在“录制宏”对话框中可以直接指定快捷键。录制宏后指定快捷键也很简单：单击“开发工具”选项卡“代码”选项组中的“宏”按钮，在“宏”对话框中选择要指定快捷键的宏，再单击“选项”按钮，通过“宏选项”对话框进行设置。

为宏指定了快捷键后，就可以用快捷键来运行宏。

注意：打开包含宏的工作簿时，为宏指定的快捷键会覆盖原有的快捷键功能。例如，

把 Ctrl+C 快捷键指定给某个宏后，Ctrl+C 快捷键就不再执行复制命令了。因此，在定义新的快捷键时，应尽量避开系统已定义的常用快捷键。

2. 用按钮运行宏

通过快捷键可以快速执行某个宏，但是宏的数量多了，快捷键就不那么方便记忆了。而且，如果宏是由其他人来使用，快捷键就更不合适了。

作为 VBA 应用软件开发者，应该为使用者提供一个易于操作的界面。“按钮”是最常见的界面元素之一。

例如，在 Excel 中录制一个名为“填充颜色”的宏，用来向当前选定的单元格填充某种颜色。然后在当前工作表中添加一个按钮，并将“填充颜色”这个宏指定给该按钮，步骤如下：

（1）在 Excel 中录制一个名为“填充颜色”的宏，向当前选定的单元格填充绿色。

（2）在 Excel 功能区的“开发工具”选项卡中单击“控件”选项组的“插入”按钮，再选择“表单控件>按钮（窗体控件）”命令，此时光标变成十字形状。

（3）在当前工作表的适当位置按住鼠标左键并拖动画出一个矩形，这个矩形的大小即为按钮的大小。得到满意的大小后松开鼠标，这样一个命令按钮就被添加到了工作表中，同时 Excel 会自动打开“指定宏”对话框。

（4）在“指定宏”对话框中选择“填充颜色”，单击“确定”按钮，就把该宏指定给按钮了。

（5）右击按钮，在弹出的快捷菜单中选择“编辑文字”命令，将按钮的标题改为“填充颜色”。

（6）单击按钮外的任意位置，结束按钮设计。

此后，单击按钮就可以运行该宏。

3. 用图片运行宏

将宏指定给图片十分简单：在“插入”选项卡的“插图”选项组中单击“图片”“形状”等按钮，在当前工作表置入插图后，右击插图，在弹出的快捷菜单中选择“指定宏”命令，并为图片指定宏。

5.2　变量和运算符

前面用录制宏的方法编写了一个简单的 VBA 程序。通过对程序的分析，了解了一些基本知识和几个语句的功能。为进一步开发 Excel 的功能，编写各种需求的程序，还应掌握 VBA 的语法、变量、数据类型、运算符等知识。

5.2.1　变量与数据类型

1. 变量

变量用于临时保存数据。程序运行时，变量的值可以改变。在 VBA 代码中可以用变量来存储数据或对象。例如：

```
MyName="北京"          '为变量赋值
MyName="上海"          '修改变量的值
```

5.1 节已经在宏的代码中使用了变量，下面再举一个简单的例子说明变量的应用。

【例 5-1】 在宏代码中使用变量。

在 Excel 功能区的“开发工具”选项卡“代码”选项组中单“宏”按钮，在“宏”对话框中输入宏名“Hello”，然后单击“创建”按钮，进入 Visual Basic 编辑器环境。输入如下代码：

```
Sub Hello()
  s_name = InputBox("请输入您的名字:")
  MsgBox "Hello," & s_name & "! "
End Sub
```

其中，Sub、End Sub 两行代码由系统自动生成，不需要手动输入。

在上述代码中，InputBox 函数显示一个信息输入对话框，输入的信息作为函数值返回，赋值给变量 s_name。MsgBox 显示一个对话框，用来输出信息，其中包含变量 s_name 的值。关于函数的详细内容请查看系统帮助信息。

在 Visual Basic 编辑器中，按 F5 键，或者单击工具栏中的 按钮，运行这个程序，显示一个图 5-1 所示的信息输入对话框。输入“LST”并单击“确定”按钮，显示图 5-2 所示的输出信息对话框。

图 5-1　输入信息对话框图

图 5-2　输出信息对话框

2．变量的数据类型

变量的数据类型决定变量允许保存何种类型的数据。表 5-1 列出了 VBA 支持的数据类型，同时列出了各种类型的变量所需要的存储空间和能够存储的数据范围。

表 5-1　数据类型

数据类型	存储空间	数值范围
Byte（字节）	1 字节	0～255
Boolean（布尔）	2 字节	True 或 False
Integer（整型）	2 字节	−32768～32767
Long（长整型）	4 字节	−2147483648～2147483647
Single（单精度）	4 字节	负值范围：−3.402823E38～−1.401298E−45 正值范围：1.401298E−45～3.402823E38
Double（双精度）	8 字节	负值范围：−1.79769313486232E308～−4.94065645841247E−324 正值范围：4.94065645841247E−324～1.79769313486232E308

续表

数据类型	存储空间	数值范围
Currency（货币）	8 字节	−922337203685477.5808～922337203685477.5807
Decimal（小数）	12 字节	不包括小数时：±79228162514264337593543950335 包括小数时：±7.9228162514264337593543950335
Date（日期时间）	8 字节	日期：100 年 1 月 1 日～9999 年 12 月 31 日 时间：00:00:00～23:59:59
Object（对象）	4 字节	任何引用对象
String（字符串）	字符串的长度	变长字符串：0～20 亿个字符 定长字符串：1～64K 个字符
Variant（数字）	16 字节	Double 范围内的任何数值
Variant（文本）	字符串的长度	数据范围和变长字符串相同

3．声明变量

变量在使用之前，最好进行声明，即定义变量的数据类型，这样可以提高程序的可读性和节省存储空间。当然这也不是绝对的，在不关心存储空间，而注重简化代码、突出重点的情况下，可以不声明直接使用变量。变量不经声明直接使用，系统会自动将变量定义为 Variant 类型。

通常使用 Dim 语句来声明变量。声明语句放到过程中，则该变量在过程内有效。声明语句放到模块顶部，则变量在模块中有效（过程、模块和工程等知识将在 5.4 节介绍）。

下面语句创建了变量 strName 并且将其指定为 String 数据类型。

```
Dim strName As String
```

为了使变量可被工程中所有的过程使用，要用如下形式的 Public 语句声明公共变量：

```
Public strName As String
```

变量的数据类型可以是表 5-1 中的任何一种。如果未指定数据类型，则默认为 Variant 类型。

变量名必须以字母开始，并且只能包含字母、数字和某些特定的字符，最大长度为 255 个字符。

可以在一个语句中声明几个变量。如在下面的语句中，变量 intX、intY、intZ 被声明为 Integer 类型。

```
Dim intX As Integer, intY As Integer, intZ As Integer
```

在下面的语句中，变量 intX 与 intY 被声明为 Variant 型，intZ 被声明为 Integer 型。

```
Dim intX, intY, intZ As Integer
```

可以使用 Dim 和 Public 语句来声明变量的对象类型。下面的语句为工作表的新建实例声明了一个变量。

```
Dim X As New Worksheet
```

如果定义对象变量时没有使用 New 关键字，则在使用该变量之前，必须使用 Set 语句将该引用对象的变量赋值为一个已有对象。

4．声明数组

数组是具有相同数据类型并共用一个名字的一组变量的集合。数组中的不同元素通过下标加以区分。

若数组的大小固定不变，则它是静态数组。若数组的大小在程序运行时可变，则它是动态数组。

数组的下标从 0 还是从 1 开始，可用 Option Base 语句进行设置。如果 Option Base 没有指定为 1，则数组下标默认从 0 开始。

数组要先声明后使用。下面这行代码声明了一个固定大小的数组，它是个 11 行乘以 11 列的 Integer 型二维数组：

```
Dim MyArray(10,10) As Integer
```

其中，第 1 个参数表示第 1 个下标的上界，第 2 个参数表示第 2 个下标的上界，默认的下标下界为 0，数组共有 11×11 个元素。

在声明数组时，不指定下标的上界，即括号内为空，则该数组为动态数组。动态数组可以在执行代码时改变大小。下面语句声明的就是一个动态数组：

```
Dim sngArray() As Single
```

动态数组声明后，可以用 ReDim 语句重新定义数组的维数以及每个维的上界。重新声明数组时，数组中存在的值一般会丢失。若要保存数组中原先的值，可以使用 ReDim Preserve 语句来扩充数组。例如，下面的语句将 varArray 数组扩充了 10 个元素，而数组中原来值并不丢失。

```
ReDim Preserve varArray(UBound(varArray) + 10)
```

其中，UBound(varArray)函数返回数组 varArray 原来的下标上界。

5．为变量赋值

为变量或数组元素赋值，通常使用赋值语句。

【例 5-2】 变量和数组的赋值。

以下程序首先声明了变量 rs、zf、k 和动态数组 cj。然后用 InputBox 函数输入学生人数并赋值给变量 rs，按人数重新定义数组 cj 的下标上界，用循环程序输入每个学生的成绩并赋值给 cj 数组下标变量。最后，用循环程序对 cj 数组下标变量的值求和，输出平均分。

```
Sub pjf()
  Dim cj() As Integer
  Dim rs As Integer
  Dim zf As Integer
  Dim k As Integer
  rs = InputBox("输入学生的人数：")
```

```
    ReDim cj(rs)
    For k = 1 To rs
       cj(k) = InputBox("输入考试成绩" & k)
    Next
    zf = 0
    For k = 1 To rs
       zf = zf + cj(k)
    Next
    MsgBox rs & "位学生的平均分是：" & zf / rs
End Sub
```

在 Excel 中创建一个宏，将代码输入，程序运行后，如果输入人数为 5，成绩分别为：80、90、85、88、75，最后得到的结果如图 5-3 所示。

图 5-3　程序的输出结果

在编写程序时，一般每个语句占一行，但有时候可能需要在一行中写几个语句。这时需要用“：”来分开不同的语句。例如 a=1:b=2。

有时一个语句太长，看上去不整齐，可以用空格加下画线“_”作为断行标记，将其分开写成几行。

5.2.2　运算符

VBA 中的运算符有 4 种：算术运算符、比较运算符、逻辑运算符和连接运算符，可用来组成不同类型的表达式。

1．算术运算符

算术运算符用于构建数值表达式或返回数值运算结果，各运算符的作用和示例见表 5-2。

表 5-2　算术运算符

符号	作　用	示　例	符号	作　用	示　例
+	加法	3+5=8	\	整除	19\6=3
−	减法、一元减	11-6=5、-6*3=-18	mod	取模	19 mod 6=1
*	乘法	6*3=18	^	指数	3^2=9
/	除法	10/4=2.5			

2．比较运算符

比较运算符用于构建关系表达式，返回逻辑值 True、False 或 Null（空）。常用的比较运算符名称和用法见表 5-3。

表 5-3　常用的比较运算符

符　号	名　称	用　法	符　号	名　称	用　法
<	小于	〈表达式 1〉<〈表达式 2〉	>=	大于或等于	〈表达式 1〉>=〈表达式 2〉
<=	小于或等于	〈表达式 1〉<=〈表达式 2〉	=	等于	〈表达式 1〉=〈表达式 2〉
>	大于	〈表达式 1〉>〈表达式 2〉	<>	不等于	〈表达式 1〉<>〈表达式 2〉

在由比较运算符组成的关系表达式中，当符合相应的关系时，结果为 True，否则为 False。如果参与比较的表达式有一个为 Null，则结果为 Null。

例如：

当变量 A 的值为 3、B 的值为 5 时，关系表达式 A>B 的值为 False，A<B 的值为 True。

3．逻辑运算符

逻辑运算符用于构建逻辑表达式，返回逻辑值 True、False 或 Null(空)。常用的逻辑运算符名称和语法见表 5-4。

表 5-4　常用的逻辑运算符

符　号	名　称	语　　法
And	与	〈表达式 1〉And〈表达式 2〉
Or	或	〈表达式 1〉Or〈表达式 2〉
Not	非	Not〈表达式〉

例如：

```
A = 10: B = 8: C = 6: D = Null      '设置变量初值

MyCheck = A > B And B > C           '返回 True
MyCheck = B > A And B > C           '返回 False
MyCheck = A > B And B > D           '返回 Null

MyCheck = A > B Or B > C            '返回 True
MyCheck = B > D Or B > A            '返回 Null

MyCheck = Not(A > B)                '返回 False
MyCheck = Not(B > A)                '返回 True
MyCheck = Not(C > D)                '返回 Null
```

4．连接运算符

字符串连接运算符有 2 个：“&”和“+”。

其中“+”运算符既可用来计算数值的和，也可以用来做字符串的串接操作。不过，最好还是使用“&”运算符来做字符串的连接操作。如果“+”运算符两边的表达式中混有字符串及数值的话，其结果会是数值的求和。如果都是字符串“相加”，则返回结果才与“&”相同。

例如：

```
MyStr = "Hello" & " World"          '返回 "Hello World"
MyStr = "Check" & 123               '返回 "Check 123"
MyNumber = "34" + 6                 '返回 40
MyNumber = "34" + "6"               '返回 "346"（字符串被串接起来）
```

5．运算符的优先级

按优先级由高到低的次序排列的运算符如下：

括号→指数→一元减→乘法和除法→整除→取模→加法和减法→连接→比较→逻辑（Not、And、Or）。

【例 5-3】 百钱买百鸡问题。

假设公鸡每只 5 元，母鸡每只 3 元，小鸡 3 只 1 元。要求用 100 元钱买 100 只鸡，问公鸡、母鸡、小鸡可各买多少只？请编一个 VBA 程序求解。

分析：设公鸡、母鸡、小鸡数分别为 x、y、z，则可列出方程组。

$$\begin{cases} x+y+z=100 \\ 5x+3y+z/3=100 \end{cases}$$

这里有 3 个未知数、2 个方程式，说明有多个解。可以用穷举法求解。

编程：进入 Excel 2010，在“开发工具”选项卡的“代码”选项组中单击“宏”按钮，在打开的“宏”对话框中输入宏名“百钱百鸡”，指定宏的位置为“当前工作簿”，单击“创建”按钮，进入 VB 编辑环境。

然后，输入如下代码。

```
Sub 百钱百鸡()
  For x = 0 To 19
    For y = 0 To 33
      z = 100 - x - y
      If 5 * x + 3 * y + z / 3 = 100 Then
        g = g & "公鸡" & x & ",母鸡" & y & ",小鸡" & z & Chr(10)
      End If
    Next
  Next
  MsgBox g
End Sub
```

因为公鸡和母鸡的最大数量分别为 19 和 33，所以采用双重循环结构，让 x 从 0 到 19、y 从 0 到 33 进行循环。每次循环求出一个 z 值，使得 x+y+z=100。如果满足条件 5x+3y+z/3=100，则 x、y 和 z 就是一组有效解，把这个解保存到字符串变量 g 中。循环结束后，用 MsgBox 函数输出全部有效解。

图 5-4　程序输出结果

程序运行后的结果如图 5-4 所示。

在上面这段程序中，使用了 Chr 函数，把 ASCII 码 10 转换为对应的回车符。

程序中还用到了 If 语句。If 是最常用的一种分支语句。它符合人们通常的语言和思维习惯。例如，if（如果）绿灯亮，then(那么)可以通行，else(否则)停止通行。

If 语句有 3 种语法形式：

```
（1）if <条件> then <语句 1> [else <语句 2>]
```

```
（2）if <条件> then
        <语句组 1>
    [else
        <语句组 2>]
    end if
（3）if <条件 1> then
        <语句组 1>
    [elseif <条件 2> then
        <语句组 2> ...
    else
        <语句组 n>]
    end if
```

<条件>是一个关系表达式或逻辑表达式。若值为 True，则执行紧接在关键字 then 后面的语句组。若<条件>的值为 False，则检测下一个 elseif<条件>或执行 else 关键字后面的语句组，然后继续执行下一个语句。

例如，根据一个字符串是否以字母 A 到 F、G 到 N 或 O 到 Z 开头来设置整数值。程序段如下：

```
Dim strMyString As String, strFirst As String, intVal As Integer
strFirst = Mid(strMyString, 1, 1)
If strFirst >= "A" And strFirst <= "F" Then
  intVal = 1
ElseIf strFirst >= "G" And strFirst <= "N" Then
  intVal = 2
ElseIf strFirst >= "O" And strFirst <= "Z" Then
  intVal = 3
Else
  intVal = 0
End If
```

其中，用 Mid 函数返回 strMyString 字符串变量从第 1 个字符开始的一个字符。假如 strMyString="VBA"，则该函数返回"V"。

5.3 面向对象程序设计

VBA 是面向对象的编程语言和开发工具。在编写程序时，经常要用到对象、属性、事件、方法等知识。下面介绍这些概念、它们之间的关系以及在程序中的用法。

1. 对象

客观世界中的任何实体都可以被看成对象。对象可以是具体的物，也可以指某些概念。从软件开发的角度来看，对象是一种将数据和操作过程结合在一起的数据结构，或者是一种具有属性和方法的集合体。每个对象都具有描述它特征的属性和附属于它的方法。属性用来表示对象的状态，方法是描述对象行为的过程。

在 Windows 软件中，窗口、菜单、文本框、按钮、下拉列表等都是对象。有的对象可

容纳其他对象，被称为容器对象，有的对象要放在别的对象当中，被称为控件。

VBA 中绝大多数对象具有可视性（Visual），即，有能看得见的直观属性，如大小、颜色、位置等。在软件设计时就能看见运行后的样子，即“所见即所得”。

对象是 VBA 程序的基础，几乎所有操作都与对象有关。Excel 的工作簿、工作表、单元格、图表都是对象。

VBA 将 Office 中的每个应用程序都看成一个对象。每个应用程序都由各自的 Application 对象代表。

2. 属性

属性就是对象的性质，如大小、位置、颜色、标题、字体等。为了实现软件的功能，也为了软件运行时界面美观、实用，必须设置对象的有关属性。

每个对象都有若干个属性，每个属性都有一个预先设置的默认值，多数不需要改动，只有部分属性需要修改。同一种对象在不同地方应用，需要设置或修改的属性也不同。

某些属性可以用鼠标拖动设置，如大小、位置等，也可以在属性窗口中设置。另一些则必须在属性窗口或程序中设置，如字体、颜色、标题等。

若要用程序设置属性的值，可在对象的后面紧接小数点、属性名称、赋值号及新的属性值。

下面语句的作用是为 Sheet1 工作表的 F8 单元格内部填充蓝色。

```
Sheets("Sheet1").Range("F8").Interior.ColorIndex = 5
```

其中，Sheet1 是当前工作簿中的一个工作表对象，F8 是工作表中的单元格对象，Interior 是单元格的内部（也是对象），ColorIndex 是 Interior 的一个属性，“=”是赋值号，5 是要设置的属性值。

读取对象的属性值，可以获取有关该对象的信息。

例如，下面的语句返回活动单元格的地址。

```
addr = ActiveCell.Address
```

3. 事件

所谓事件，就是可能发生在对象上的事情，是由系统预先定义并由用户或系统发起的动作。事件作用于对象，对象识别事件并做出相应的反应。事件可以由系统引发，例如生成对象时，系统引发一个 Initialize 事件。事件也可以由用户引发，例如单击按钮，拖动对象、改变大小，都会引发相应的事件。

在软件运行过程中，若对象发生某个事件，则需要做出相应的反应。例如单击“退出”按钮，则软件结束运行。

为了使对象在某一事件发生时能够做出预定的反应，必须针对这一事件编写相应的代码。这样，在软件运行时，只要事件发生，就执行对应的代码，完成相应的动作。事件不发生，则不执行。

【例 5-4】 自动填写单元格列标和行号。

在 Excel 中编写一个程序，实现以下功能：当选定当前工作表的任意一个单元格时，

该单元格将自动填入其列标和行号。

首先，创建一个 Excel 工作簿，保存为“自动填写单元格列标和行号.xlsm”。

然后，进入 VB 编辑环境，双击 Microsoft Excel 对象中的 Sheet1 工作表对象。在代码编辑器窗口上方的“对象”下拉列表中选择 Worksheet、“过程”下拉列表中选择 SelectionChange，对工作表的 SelectionChange 事件编写如下代码：

```
Private Sub Worksheet_SelectionChange(ByVal Target As Range)
  a = Target.Address(True, False)       '绝对行、相对列地址
  cn = Left(a, InStr(a, "$") - 1)       '取出列标
  rn = Target.Row                       '取出行号
  Target.Value = cn & "列" & rn & "行"  '填写到当前单元格
End Sub
```

这是一个子程序过程，过程名为“Worksheet_SelectionChange”，其中 Worksheet 表示工作表对象，SelectionChange 是一个事件名，该事件在工作表单元格焦点改变时发生。当用鼠标或键盘改变单元格焦点时，系统会执行上面过程中的代码。

过程的参数 Target 表示当前单元格对象。

用 Target.Address(True, False)可以求出单元格地址，送给变量 a。其中第 1 个参数 True 表示绝对行，第 2 个参数 False 表示相对列。即地址字符串的行号前带有“$”符、列标前不带“$”符。例如，“E$8”“F$20”“AK$136”等。

函数 InStr(a, "$")求出字符串 a 中“$”的位置，Left(a, InStr(a, "$") −1)从字符串 a 中取出“$”左边的子串，也就是列标。

行号可以用 mid 函数从字符串 a 中提取，但直接用 Target.Row 提取更简单些。

分解出当前单元格的列标和行号后，重新组合成一个提示字符串填写到当前单元格。

打开“自动填写单元格列标和行号”工作簿，选中 Sheet1 工作表，用鼠标或键盘选中任意一个单元格，该单元格将自动填入其列标和行号，如图 5-5 所示。

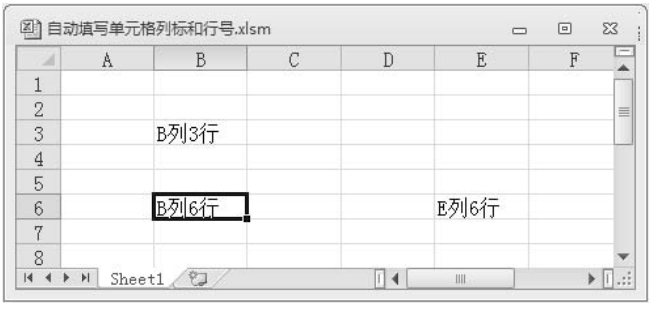

图 5-5　自动填写的列标和行号信息

4. 方法

方法是对象可以执行的动作。例如，Worksheet 对象的 PrintOut 方法用于打印工作表的内容。

方法通常带有参数，以限定执行动作的方式。

例如，下面的语句可将活动工作表的 1～2 页打印 1 份。

```
ActiveWindow.SelectedSheets.PrintOut 1, 2, 1
```

下面的语句通过使用 Save 方法保存当前工作簿。

```
ActiveWorkbook.Save
```

通常，方法是动作，属性是性质。使用方法将导致发生某些事件，使用属性则会返回对象的信息或改变对象的某个性质。

所谓面向对象程序设计，就是要设计一个个对象，再把这些对象用某种方式联系起来构成一个系统，即软件系统。

每个对象需要设计的不外乎属性，针对需要的事件编写程序代码，在编写代码时使用系统提供的语句、命令、函数、事件和方法。

【例 5-5】 在 Excel 中实现定时提醒。

Office 的 Application 对象中有个 OnTime 方法，用来触发一个程序在特定时刻运行。特定的时刻可以是某个日期的某个时间，也可以是相对某个时刻的时间值。通过这个方法可以在 Excel 中编写定时程序。

在 Excel“开发工具”选项卡的“代码”选项组中单击“宏”按钮，在“宏”对话框中输入宏名“ds”，然后单击“创建”按钮，进入 VB 编辑环境，输入如下宏代码：

```
Sub ds()
  Application.OnTime TimeValue("8:50:00"), "my_msg"
End Sub
```

用同样的方式，创建另一个宏：

```
Sub my_msg()
  MsgBox "现在是 8 点 50 分,9 点钟您有个约会!
", vbInformation, "提醒"
End Sub
```

宏 ds 设置定时器在 8:50:00 激活。激活后运行宏 my_msg，弹出一个对话框并显示提示信息。

运行宏 ds 后，当指定的时刻到来时，屏幕会显示图 5-6 所示的信息。

图 5-6　定时提醒对话框

5.4　过程

前面录制或手动编写的“宏”是“过程”的一种，叫子程序。还有一种常用的“过程”叫函数。它们可能放在对象当中，也可能放在独立的模块当中。放在对象当中的“过程”可能和某个事件相关联。对象、模块又属于“工程”的资源。

本节研究工程、模块、过程之间的关系，过程的创建，子程序和自定义函数的用法，代码的调试方法。

5.4.1 工程、模块与过程

每个 VBA 应用程序都存在于一个“工程”中。工程下面可分为若干个“对象”“窗体”“模块”“类模块”。在录制宏时，如果原来不存在模块，Office 就自动创建一个。

在“开发工具”选项卡的“代码”选项组中单击“Visual Basic”按钮，或者按 Alt+F11 快捷键，进入 VB 编辑环境。

在“视图”菜单中选择“工程资源管理器”命令，或在“标准”工具栏中单击“工程资源管理器”按钮，都可以打开“工程”任务窗格。这时，在“标准”工具栏中单击“用户窗体”“模块”或“类模块”按钮，或在“插入”菜单中选择相应的菜单命令，便可在“工程”中插入相应的项目。

双击任意一个项目，可在右边的窗格中查看或编写程序代码。VB 编辑器中的工程和代码界面如图 5-7 所示。

图 5-7 VB 编辑器窗口

模块中可以定义若干个“过程”。每个过程都有唯一的名字，过程中包含一系列语句。过程可以是函数、子程序或属性。

函数过程通常要返回一个值。这个值是计算的结果或是测试的结果，例如 False 或 True。可以在模块中创建和使用自定义函数。

但子程序过程只执行一个或多个操作，不返回数值。前面录制的宏，实际上就是子程序过程，宏名就是子程序名。用宏录制的方法可以得到子程序过程，但不能得到函数或属性过程。

属性过程由一系列语句组成，用来为窗体、标准模块以及类模块创建属性。

创建过程通常有以下两种方法。

【方法 1】 直接输入代码。

（1）打开要编写过程的模块。

（2）键入 Sub、Function 或 Property，分别创建 Sub、Function 或 Property 过程。系统会在后面自动加上一个 End Sub、End Function 或 End Property 语句。

（3）在其中键入过程的代码。

【方法 2】 用“添加过程”对话框。

（1）打开要编写过程的模块。

（2）在“插入”菜单中选择“过程”命令，显示图 5-8 所示的“添加过程”对话框。

（3）在“添加过程”对话框的“名称”文本框键入过程的名称。选择要创建过程的类型，设置过程的范围。如果需要，还可以选中“把所有局部变量声明为静态变量”复选框。最后，单击“确定”按钮，进行代码编写。

图 5-8 “添加过程”对话框

进入 Excel 或打开一个工作簿，系统都会自动创建一个工程，工程中自动包含工作簿对象、工作表对象。过程可以在对象中创建，也可以在模块或类模块中创建。如果模块不存在，首先需要向工程添加一个模块。

【例 5-6】 创建一个显示消息框的过程。

（1）在 Excel 中，单击“开发工具”选项卡“代码”选项组中的“Visual Basic”按钮，打开 VB 编辑器窗口。

（2）在工具栏中单击“工程资源管理器”按钮，或按 Ctrl+R 快捷键，在 VB 编辑器的左侧打开“工程”窗格。

（3）在“工程”窗格的任意位置右击，在弹出的快捷菜单中选择“插入>模块”命令，或在“标准”工具栏中单击“模块”按钮，或选择“插入>模块”菜单命令，将一个模块添加到工程中。

（4）在“插入”菜单中选择“过程”命令，打开图 5-8 所示的“添加过程”对话框。在对话框中输入“显示消息框”作为过程名。在“类型”选项组中选择“子程序”单选按钮。单击“确定”按钮。这样一个新的过程就添加到模块中了。

（5）在过程中输入语句，得到下面的代码段：

```
Public Sub 显示消息框()
  Msgbox "这是一个测试用的过程"
End Sub
```

在输入 Msgbox 命令过程中，系统会自动提示有关参数信息。

要运行一个过程，可以使用“运行”菜单中的“运行子程序/用户窗体”命令，也可以使用工具栏按钮或按 F5 快捷键。

模块与过程随工作簿一起保存。在工作簿窗口可以通过“文件”选项卡保存工作簿，保存类型应为“Excel 启用宏的工作簿”。

5.4.2　子程序

每个子程序都以 Sub 开头，End Sub 结尾。

语法格式如下：

```
[Public|Private] Sub 子程序名([<参数>])
  [<语句组>]
  [Exit Sub]
  [<语句组>]
End Sub
```

Public 关键字可以使子程序在所有模块中有效。Private 关键字使子程序只在本模块中有效。如果没有指定，默认情况是 Public。

子程序可以带参数。

Exit Sub 语句的作用是退出子程序。

【例 5-7】 下面是一个求矩形面积的子程序。它带有两个参数 L 和 W，分别表示矩形的长和宽。

```
Sub mj(L, W)
  If L = 0 Or W = 0 Then Exit Sub
  MsgBox L * W
End Sub
```

上述子程序首先判断两个参数，如果任意一个参数值为零，则直接退出子程序，不做任何操作。否则，计算出矩形面积 L*W，并将面积显示出来。

调用子程序用 Call 语句。对上述子程序执行

```
Call mj(8,9)
```

其输出结果为 72。而执行

```
Call mj(8,0)
```

则不输出任何结果。

Call 语句用来调用一个 Sub 过程。语法形式如下：

```
[Call] <过程名> [<参数列表>]
```

其中，关键字 Call 可以省略。如果指定了这个关键字，则<参数列表>必须加上括号。如果省略 Call 关键字，也必须要省略<参数列表>外面的括号。

因此，Call mj(8,9)可以改为 mj 8,9

【例 5-8】 输出“玫瑰花数”。

所谓“玫瑰花数”，也叫“水仙花数”，指一个三位数，其各位数字立方和等于该数本身。

进入 Excel，在 VB 编辑环境中，插入一个模块，创建如下子程序过程：

```
Sub 玫瑰花数()
  c = 1
  For n = 100 To 999
    i = n \ 100
```

```
    j = n \ 10 - i * 10
    k = n Mod 10
    If (n = i * i * i + j * j * j + k * k * k) Then
      Cells(1, c) = n
      c = c + 1
    End If
  Next
End Sub
```

上述子程序首先用赋值语句设置列号变量 c 的初值为 1。

然后用循环语句对所有三位数，分别取出百、十、个位数字保存到变量 i、j、k 中，如果各位数字立方和等于该数本身，则将该数填写到当前工作表第 1 行 c 列单元格，并调整列号 c。

其中：

n\100，将 n 除以 100 取整，得到百位数；

n\10–i*10，得到十位数；

n Mod 10，将 n 除以 10 取余，得到个位数。

Cells(1，c)表示 1 行 c 列单元格对象。

赋值语句 Cells(1, c) = n 设置该单元格对象的 Value 属性值为 n。Value 是单元格对象的默认属性，可以省略不写。

在 Visual Basic 编辑器中，按 F5 键运行这个程序后，在当前工作表中得到图 5-9 所示的结果。

	A	B	C	D	E
1	153	370	371	407	

图 5-9　在工作表中输出的“玫瑰花数”

5.4.3　自定义函数

VBA 提供了大量的内置函数。例如字符串函数 Mid、统计函数 Max 等。在编程时可以直接引用，非常方便。但有时也需要按自己的要求编写函数，即自定义函数。

用 Function 语句可以定义函数，其语法形式如下：

```
[Public|Private] Function 函数名([<参数>]) [As 数据类型]
  [<语句组>]
  [函数名=<表达式>]
  [Exit Function]
  [<语句组>]
  [函数名=<表达式>]
End Function
```

定义函数时用 Public 关键字，则所有模块都可以调用它。用 Private 关键字，函数只可用于同一模块。如果没有指定，则默认为 Public。

函数名末尾可使用 As 子句来声明返回值的数据类型，参数也可指定数据类型。若省略

数据类型说明，系统会自动根据赋值确定。

Exit Function 语句的作用是退出 Function 过程。

下面这个自定义函数可以求出半径为 R 的圆的面积：

```
Public Function area(R As Single) As Single
  area = 3.14 * R ^ 2
End Function
```

该函数也可简化为：

```
Function area(R)
  area = 3.14 * R ^ 2
End Function
```

如果要计算半径为 5 的圆的面积，可调用函数 area(5)。假设 A 是一个已赋值为 3 的变量，area(A+5)将求出半径为 8 的圆的面积。

【例 5-9】 求最大公约数。

下面编写一个 VBA 程序，对给定的任意两个正整数，求它们的最大公约数。

求最大公约数的方法有多种，这里使用一种被称为“辗转相除”的方法。用两个数中较大的数除以较小的数取余，如果余数为零，则除数即为最大公约数；若余数大于零，则将原来的除数作为被除数，余数作为除数，再进行相除、取余操作，直至余数为零。

可以在 Excel 中编写一个自定义函数，求两个数的最大公约数，并在工作表中测试这个函数。

（1）设计工作表

创建一个 Excel 工作簿，保存为“求最大公约数.xlsm”。

在 Sheet1 工作表的 A、B、C 列创建一个表格，设置表头、边框线，及最适合的列宽、行高，输入一些用于测试的数据。得到图 5-10 所示的界面。

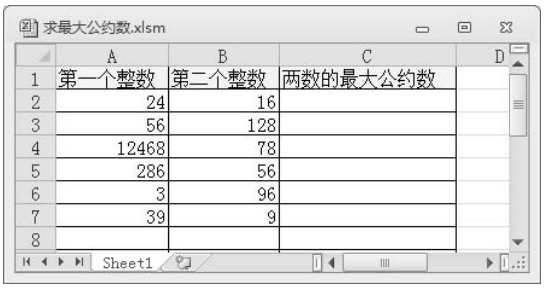

图 5-10　工作表界面

（2）编写自定义函数

进入 VB 编辑环境，插入一个模块，编写一个自定义函数 hcf，代码如下：

```
Function hcf(m, n)
  If m < n Then
    t = m: m = n: n = t      '让大数在 m、小数在 n 中
  End If
```

```
    r = m Mod n                    '对 m 和 n 取模，结果放到 r 中
    Do While r > 0                 '辗转相除
      m = n
      n = r
      r = m Mod n
    Loop
    hcf = n                        '返回最大公约数 n
End Function
```

上述自定义函数的两个形参 m 和 n，为要求最大公约数的两个正整数。

在函数体中，首先对两个形参进行判断，让大数在 m 中、小数在 n 中。其实，这个判断过程是可以省略的，因为即便 m 小于 n，第一轮循环后，m 也会自动与 n 互换位置。

然后，用 m 除以 n 得到余数 r。如果余数 r 大于零，则将原来的除数 n 作为被除数 m，余数 r 作为除数 n，再重复上述过程，直到余数 r=0 为止。此时，除数 n 就是最大公约数，作为函数值返回。

（3）测试自定义函数

函数 hcf 定义后，在当前工作表的 C2 单元格输入公式“=hcf(A2,B2)”，如图 5-11 所示。按 Enter 键后得到结果 8，即 24 和 16 的最大公约数为 8。

将 C2 单元格的公式向下填充到 C7 单元格，将会得到其余几组数值的最大公约数，如图 5-12 所示。

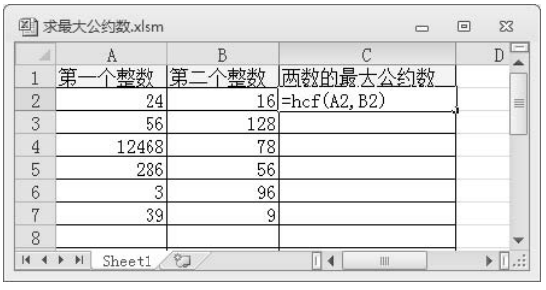

图 5-11　在 C2 单元格输入公式

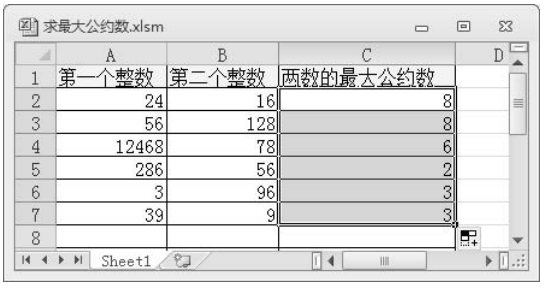

图 5-12　将 C2 单元格公式填充到 C7

下面，对 Do…Loop 语句进一步说明。

Do…Loop 语句提供了一种结构化与适应性更强的方法来执行循环。

它有以下两种形式：

```
（1）Do[{While|Until}<条件>]
       [<过程语句>]
       [Exit Do]
       [<过程语句>]
     Loop
（2）Do
       [<过程语句>]
       [Exit Do]
       [<过程语句>]
     Loop [{While|Until}<条件>]
```

上述格式中，While 和 Until 的作用正好相反。使用 While，当<条件>为 True 时继续循环。使用 Until，当<条件>为 True 时，结束循环。

把 While 或 Until 放在 Do 子句中，则先判断后执行。把一个 While 或 Until 放在 Loop 子句中，则先执行后判断。

5.4.4 代码调试

1. 代码的运行、中断和继续

在 VB 编辑环境中运行一个子程序过程或用户窗体，有以下几种方法：

【方法 1】 选择“运行”菜单中的“运行子过程/用户窗体”命令。

【方法 2】 单击工具栏中的“运行子过程/用户窗体”按钮。

【方法 3】 用 F5 快捷键。

在执行代码时，可能会由于以下原因而中断执行：

（1）发生运行时错误。

（2）遇到一个断点或 Stop 语句。

（3）人为中断执行。

如果要人为中断执行，可用以下几种方法：

【方法 1】 选择“运行”菜单中的“中断”命令。

【方法 2】 用 Ctrl+Break 快捷键。

【方法 3】 单击工具栏中的“中断”按钮。

【方法 4】 选择“运行”菜单中的“重新设置”命令。

【方法 5】 单击工具栏中的“重新设置”按钮。

要继续执行，可用以下几种方法：

【方法 1】 在“运行”菜单中选择“继续”命令。

【方法 2】 按 F5 快捷键。

【方法 3】 单击工具栏中的“继续”按钮。

2. 跟踪代码的执行

为了分析代码，查找逻辑错误原因，需要跟踪代码的执行。跟踪的方式有以下几种：

（1）逐语句。跟踪代码的每一行，并逐语句跟踪过程。这样就可查看每个语句对变量的影响。

（2）逐过程。将每个过程当成单个语句。使用它代替“逐语句”以跳过整个过程调用，而不是进入调用的过程。

（3）运行到光标处。允许在代码中选定想要中断执行的语句。这样就允许“逐过程”执行代码区段，例如循环。

要跟踪执行代码，可以在“调试”菜单中选择“逐语句”“逐过程”“运行到光标处”命令，或使用相应的快捷键（F8、Shift+F8、Ctrl+F8）。

在跟踪过程中，只要将鼠标指针移到任意一个变量名上，就可以看到该变量当时的值，由此分析程序是否有错。也可以选择需要的变量，添加到监视窗口进行监视。

3. 设置与清除断点

若估计代码的某处可能存在问题，可在特定语句上设置一个断点以中断程序的执行，不需要中断时再清除断点。

将光标定位在需要设置断点的代码行，然后用以下方法可以设置或清除断点：

【方法 1】 在“调试”菜单中选择“切换断点”命令。

【方法 2】 按 F9 快捷键。

【方法 3】 在对应代码行的左边界标识条上单击。

以上方法均会在代码行和左边界标识条上设置断点标记。清除断点则标记消失。

如果在一个包含多个语句的（用冒号分隔的）行上面设置一个断点，则中断会发生在程序行的第一个语句。

要清除应用程序中的所有断点，可在“调试”菜单中选择“清除所有断点”命令。

5.5　工作簿、工作表和单元格

在实际应用中，经常要对 Excel 工作簿、工作表、单元格区域进行操作。本节介绍用 VBA 代码对它们进行操作的方法，之后给出一个自动生成年历的应用案例。

5.5.1　工作簿和工作表操作

利用 VBA 代码新建工作簿，可使用 Add 方法。

【例 5-10】 下述过程创建一个新的工作簿，系统自动将该工作簿命名为“工作簿 N”，其中“N”是一个序号。新工作簿将成为活动工作簿。

```
Sub AddOne()
  Workbooks.Add
End Sub
```

新建工作簿时，最好将其分配给一个对象变量，以便控制新工作簿。

【例 5-11】 下述过程将 Add 方法返回的工作簿对象分配给对象变量 NewBook。然后，对 NewBook 进行操作。

```
Sub AddNew()
  Set NewBook = Workbooks.Add
  NewBook.SaveAs Filename:="Test.xlsx"
End Sub
```

其中，Set 语句用来为对象变量赋值。

用 Open 方法可以打开一个工作簿。

【例 5-12】 下述过程打开 D 盘根目录中的 Test.xlsx 工作簿。

```
Sub OpenUp()
  Workbooks.Open ("D:\Test.xlsx")
End Sub
```

工作簿中每个工作表都有一个编号，它是分配给工作表的连续数字，按工作表标签位置从左到右编排序号。利用编号可以实现对工作表的引用。

【例 5-13】 下述过程激活当前工作簿上的第 1 张工作表。

```
Sub FirstOne()
  Worksheets(1).Activate
End Sub
```

也可以使用 Sheets 引用工作表。

【例 5-14】 下述过程激活工作簿中的第 4 张工作表。

```
Sub FourthOne()
  Sheets(4).Activate
End Sub
```

注意：如果移动、添加或删除工作表，则工作表编号顺序将会更改。

还可以通过名称来标识工作表。

下面这条语句激活工作簿中的 Sheet1 工作表。

```
Worksheets("Sheet1").Activate
```

5.5.2 单元格和区域的引用

在 Excel 中，经常要指定单元格或单元格区域，然后对其进行某些操作，如输入公式、更改格式等。

Range 对象既可表示单个单元格，也可表示单元格区域。下面是标识和处理 Range 对象的常用方法。

1．用 A1 样式记号引用单元格和区域

Range 对象中有一个 Range 属性。使用 Range 属性可引用 A1 样式的单元格或单元格区域。

【例 5-15】 下面的程序将工作表“Sheet1”的单元格区域 A1:D5 的字体设置为加粗。

```
Sub test()
  Sheets("Sheet1").Range("A1:D5").Font.Bold = True
End Sub
```

表 5-5 给出了使用 Range 属性的 A1 样式引用示例。

表 5-5 使用 Range 属性的 A1 样式引用示例

引　　用	含　　义
Range("A1")	单元格 A1
Range("A1:B5")	从单元格 A1 到单元格 B5 的区域
Range("C5:D9,G9:H16")	多块选定区域
Range("A:A")	A 列
Range("1:1")	第 1 行

续表

引　用	含　义
Range("A:C")	从 A 列到 C 列的区域
Range("1:5")	从第 1 行到第 5 行的区域
Range("1:1,3:3,8:8")	第 1 行、第 3 行和第 8 行
Range("A:A,C:C,F:F")	A、C 和 F 列

可用方括号将 A1 引用样式或命名区域括起来，作为 Range 属性的快捷方式。这样就不必键入单词 Range 和引号了。

【例 5-16】 下面的程序可清除工作表“Sheet1”的单元格区域“A1:B5”的内容。

```
Sub ClearRange()
  Worksheets("Sheet1").[A1:B5].ClearContents
End Sub
```

如果将对象变量设置为 Range 对象，则可通过变量引用单元格区域。

【例 5-17】 下述过程创建了对象变量 myRange，并将活动工作簿中 Sheet1 的单元格区域 A1:D5 赋予该变量。随后的语句用该变量代替该区域对象，填充随机函数值并设置该区域的格式。

```
Sub Random()
  Dim myRange As Range
  Set myRange = Worksheets("Sheet1").Range("A1:D5")
  myRange.Formula = "=RAND()"
  myRange.Font.Bold = True
End Sub
```

2. 用行列编号引用单元格

Range 对象有一个 Cells 属性，该属性返回代表单元格的 Range 对象。可以使用 Cells 属性的行列编号来引用单元格。

【例 5-18】在下面的程序中，Cells(6,1)返回 Sheet1 的 6 行 1 列单元格（即 A6 单元格），然后将 Value 属性设置为 10。

```
Sub test()
  Worksheets("Sheet1").Cells(6, 1).Value = 10
End Sub
```

【例 5-19】 下面的程序用变量替代编号，在单元格区域中循环处理。将 Sheet1 工作表第 3 列的 1～20 行单元格填入自然数 1～20。

```
Sub test()
  Dim Cnt As Integer
  For Cnt = 1 To 20
    Worksheets("Sheet1").Cells(Cnt, 3).Value = Cnt
  Next Cnt
```

```
End Sub
```

如果对工作表应用 Cells 属性时不指定编号，则该属性将返回代表工作表所有单元格的 Range 对象。

【例 5-20】 下述过程将清除活动工作簿中 Sheet1 的所有单元格的内容。

```
Sub ClearSheet()
  Worksheets("Sheet1").Cells.ClearContents
End Sub
```

Range 对象也可以由 Cells 属性指定区域。例如，Range(Cells(1,1),Cells(6,6))表示当前工作表由 1 行 1 列到 6 行 6 列所构成的区域。

3. 引用行和列

用 Rows 或 Columns 属性可以引用整行或整列。这两个属性返回代表单元格区域的 Range 对象。

【例 5-21】 下面的程序用 Rows(1)返回 Sheet1 的第 1 行，然后将单元格区域的 Font 对象的 Bold 属性设置为 True。

```
Sub test()
  Worksheets("Sheet1").Rows(1).Font.Bold = True
End Sub
```

表 5-6 列举了 Rows 和 Columns 属性的几种用法。

表 5-6 Rows 和 Columns 属性的应用示例

引　用	含　义	引　用	含　义
Rows(1)	第 1 行	Columns("A")	第 1 列
Rows	工作表上所有的行	Columns	工作表上所有的列
Columns(1)	第 1 列		

若要同时处理若干行或列，可创建一个对象变量并使用 Union 方法将 Rows 或 Columns 属性的多个调用组合起来。

【例 5-22】 下面的程序将活动工作簿中 Sheet 1 的第 1 行、第 3 行和第 5 行的字体设置为加粗。

```
Sub SeveralRows()
  Dim myUn As Range
  Worksheets("Sheet1").Activate
  Set myUn = Union(Rows(1), Rows(3), Rows(5))
  myUn.Font.Bold = True
End Sub
```

4. 引用命名区域

为了通过名称来引用单元格区域，首先要对区域命名。方法是选定单元格区域后，单击编辑栏左端的名称框，键入名称后，按 Enter 键。

【例 5-23】 下面的程序将当前工作表中名为“AA”的单元格区域内容设置为 30。

```
Sub SetValue()
    [AA].Value = 30
End Sub
```

【例 5-24】下面的程序用 For Each…Next 循环语句在命名区域中的每个单元格上循环。如果该区域中的某个单元格的值超过 25，就将该单元格的颜色更改为黄色。

```
Sub ApplyColor()
  Const Limit As Integer = 25
  For Each c In Range("AA")
    If c.Value > 25 Then
       c.Interior.ColorIndex = 27
    End If
  Next c
End Sub
```

For Each…Next 语句针对一个数组或集合中的每个元素（可以把 Excel 工作表区域单元格作为集合的元素），重复执行一组语句。语法形式如下：

```
For Each <元素> In <集合或数组>
  [<语句组>]
  [Exit For]
  [<语句组>]
Next [<元素>]
```

其中，<元素>是用来遍历集合或数组中所有元素的变量。

如果集合或数组中至少有一个元素，就会进入 For…Each 的循环体执行。一旦进入循环，便针对集合或数组中每个元素执行循环体中的所有语句。当集合或数组中的所有元素都执行完了，便会退出循环，执行 Next 之后的语句。

可以在循环体中的任何位置放置 Exit For 语句，退出循环。

5. 相对引用与多区域引用

处理相对于某个单元格的其他单元格的常用方法是使用 Offset 属性。

【例 5-25】 下面的程序将位于活动工作表活动单元格下 1 行和右 3 列的单元格设置为双下画线格式。

```
Sub Underline()
  ActiveCell.Offset(1, 3).Font.Underline = xlDouble
End Sub
```

通过在两个或多个引用之间放置逗号，可使用 Range 属性引用多个单元格区域。

【例 5-26】 下面的过程清除当前工作表上 3 个区域的内容。

```
Sub ClearRanges()
  Range("C5:D9,G9:H16,B14:D18").ClearContents
```

```
End Sub
```

假如上述 3 个区域分别被命名为 MyRange、YourRange 和 HisRange，则也可用下面的语句清除当前工作表这 3 个区域的内容。

```
Range("MyRange,YourRange,HisRange").ClearContents
```

用 Union 方法可将多个单元格区域组合到一个 Range 对象中。

【例 5-27】 下面的过程创建了名为 myMR 的 Range 对象，并将其定义为单元格区域 A1:B2 和 C3:D4 的组合，然后将该组合区域的字体设置为加粗。

```
Sub MRange()
  Dim r1, r2, myMR As Range
  Set r1 = Sheets("Sheet1").Range("A1:B2")
  Set r2 = Sheets("Sheet1").Range("C3:D4")
  Set myMR = Union(r1, r2)
  myMR.Font.Bold = True
End Sub
```

5.5.3 对单元格和区域的操作

知道如何引用单元格和区域后，就可以对它们进行操作了。下面介绍选定和激活单元格、处理活动单元格、在单元格区域中循环等方法。

1. 选定和激活单元格

使用 Excel 时，有时要选定单元格或区域，然后执行某一操作。例如设置单元格的格式或在单元格中输入数值等。

用 Select 方法可以选中工作表和工作表上的对象，而 Selection 属性则可返回代表活动工作表上的当前选定的区域对象。

宏录制器经常创建使用 Select 方法和 Selection 属性的宏。下述子程序过程是用宏录制器创建的，其作用是在工作表 Sheet1 的 A1 和 B1 单元格输入文字“姓名”和“地址”，并设置为粗体。

```
Sub 宏1()
    Sheets("Sheet1").Select
    Range("A1").Select
    ActiveCell.FormulaR1C1 = "姓名"
    Range("B1").Select
    ActiveCell.FormulaR1C1 = "地址"
    Range("A1:B1").Select
    Selection.Font.Bold = True
End Sub
```

完成同样的任务，也可以使用下面过程：

```
Sub Labels()
  With Worksheets("Sheet1")
```

```
    .Range("A1") = "姓名"
    .Range("B1") = "地址"
    .Range("A1:B1").Font.Bold = True
  End With
End Sub
```

第二种方法没有选定工作表或单元格，因而效率更高。

在 VBA 程序中，使用单元格之前，既可以先选中它们，也可以不经选中而直接进行某些操作。

【例 5-28】 要用 VBA 程序在单元格 D8 中输入公式，不必先选定单元格 D8，而只需要将 Range 对象的 Formula 属性设置为需要的公式。代码如下：

```
Sub EnterFormula()
  Range("D8").Formula = "=SUM(D1:D7)"
End Sub
```

可用 Activate 方法激活工作表或单元格。

【例 5-29】 下述过程选定了一个单元格区域，然后激活该区域内的一个单元格，但并不改变选定区域。

```
Sub MakeActive()
  Worksheets("Sheet1").Activate
  Range("A1:D4").Select
  Range("B2").Activate
End Sub
```

2. 处理活动单元格

ActiveCell 属性返回代表活动单元格的 Range 对象。可对活动单元格应用 Range 对象的任何属性和方法。例如，语句 ActiveCell.Value = 35 将当前工作表活动单元格的内容设置为 35。

用 Activate 方法可以指定活动单元格。

【例 5-30】 下述过程激活 Sheet1 工作表，并使单元格 B5 成为活动单元格，然后将其字体设置为加粗。

```
Sub SetA()
  Worksheets("Sheet1").Activate
  Range("B5").Activate
  ActiveCell.Font.Bold = True
End Sub
```

注意：选定区域用 Select 方法，激活单元格用 Activate 方法。

【例 5-31】 下述过程在选定区域内的活动单元格中插入文本，然后通过 Offset 属性将活动单元格右移一列，但并不更改选定区域。

```
Sub MoveA()
```

```
  Worksheets("Sheet1").Activate
  Range("A1:D10").Select
  ActiveCell.Value = "姓名"
  ActiveCell.Offset(0, 1).Activate
End Sub
```

CurrentRegion 属性返回由空白行和空白列所包围的单元格区域。

【例 5-32】 下面的程序将选定区域扩充到与活动单元格相邻的包含数据的单元格中。其中，CurrentRegion 属性返回由空白行和空白列包围的单元格区域。

```
Sub Region()
  Worksheets("Sheet1").Activate
  ActiveCell.CurrentRegion.Select
End Sub
```

3. 在单元格区域中循环

在 VBA 程序中，经常需要对区域内的每个单元格进行同样的操作。为达到这一目的，可使用循环语句。

在单元格区域中循环的一种方法是将 For…Next 循环语句与 Cells 属性配合使用。使用 Cells 属性时，可用循环计数器或其他表达式来替代单元格的行、列编号。

【例 5-33】 下述过程在单元格区域 C1:C20 中循环，将所有绝对值小于 10 的数字都设置为红色。其中用变量 cnt 代替行号。

```
Sub test()
  For cnt = 1 To 20
    Set curc = Worksheets("sheet1").Cells(cnt, 3)       '设置对象变量
    curc.Font.ColorIndex = 0                            '先置成黑色
    If Abs(curc.Value) < 10 Then curc.Font.ColorIndex = 3'若小于10则改成红色
  Next cnt
End Sub
```

很多时候，需要用 VBA 程序求出 Excel 数据区尾端的行号和列号。

求数据区尾端行号常用的方法有以下几种：

```
r = Range("A1").End(xlDown).Row                 '求A1单元格数据区尾端行号
r = Cells(1, 1).End(xlDown).Row                 '求A1单元格数据区尾端行号
r = Range("A1048576").End(xlUp).Row             '求A列数据区尾端行号
r = Cells(1048576, 1).End(xlUp).Row             '求A列数据区尾端行号
r = Columns(1).End(xlDown).Row                  '求A列数据区尾端行号
```

求数据区尾端列号常用的方法有以下几种：

```
c = Range("A1").End(xlToRight).Column           '求A1单元格数据区尾端列号
c = Cells(1, 1).End(xlToRight).Column           '求A1单元格数据区尾端列号
c = Cells(1, 16384).End(xlToLeft).Column        '求第1行数据区尾端列号
c = Rows(1).End(xlToRight).Column               '求第1行数据区尾端列号
```

在单元格区域中循环的另一种简便方法是使用 For Each…Next 循环语句和由 Range 属性指定的单元格集合。

【例 5-34】 下述过程在单元格区域 A1:D10 中循环，将所有绝对值小于 10 的数字都设置为红色。

```
Sub test()
  For Each c In Worksheets("Sheet1").Range("A1:D10")
    If Abs(c.Value) < 10 Then c.Font.ColorIndex = 3
  Next
End Sub
```

如果不知道要循环的区域边界，可用 CurrentRegion 属性返回活动单元格周围的数据区域。

【例 5-35】下述过程在当前工作表上运行时，将在活动单元格周围的数据区域内循环，将所有绝对值小于 10 的数字都设置为红色。

```
Sub test()
  For Each c In ActiveCell.CurrentRegion
    If Abs(c.Value) < 10 Then c.Font.ColorIndex = 3
  Next
End Sub
```

5.5.4　自动生成年历

下面给出一个在 Excel 中生成年历的程序，它可为任意指定的年份生成完整的年历，结果如图 5-13 所示。

首先，创建一个 Excel 工作簿，在任意一个工作表中，按图 5-13 所示的样式设置单元格区域的字体、字号、字体颜色、填充颜色、边框、列宽、行高等格式。

然后，进入 VB 编辑环境，插入一个模块，在模块中编写一个“生成年历”子程序，代码如下：

```
Sub 生成年历()
  '指定年份
  y = InputBox("请指定一个年份：")
  '清除原有内容
  Range("1:1,4:11,14:21,24:31,34:41").ClearContents
  '设置标题
  Cells(1, 1) = y & "年历"
  '将每个月的天数存放到数组 dm（下标从 0 开始）
  Dim dm As Variant
  dm = Array(31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31)
  '处理闰年，修正 2 月份天数
  If ((y Mod 400 = 0) Or (y Mod 4 = 0 And y Mod 100 <> 0)) Then
    dm(1) = 29
  End If
```

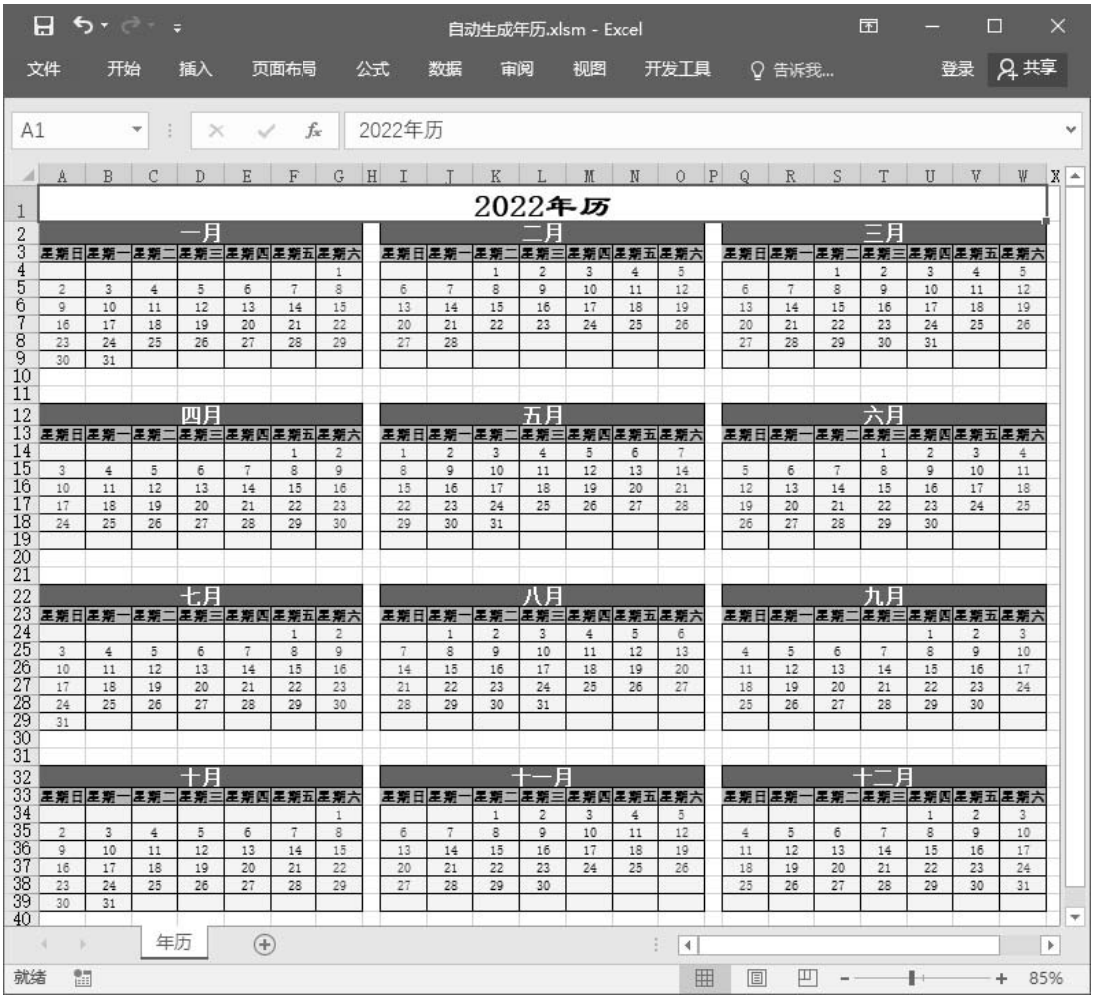

图 5-13　在 Excel 中生成的年历

```
For m = 0 To 11
  '计算每月第一天的星期数（1 日、2 一、3 二、4 三、5 四、6 五、7 六）
  d = DateSerial(y, m + 1, 1)
  w = Weekday(d)
  '计算每月起始的行号和列号
  r = (m \ 3) * 10 + 4
  c = (m Mod 3) * 8
  '排出一个月的日期
  For d = 1 To dm(m)
    Cells(r, c + w) = d
    w = w + 1
    If w > 7 Then
      w = 1
      r = r + 1
    End If
  Next
```

```
  Next
End Sub
```

上述程序首先用 InputBox 函数输入一个年份送给变量 y，清除表格中原有的内容，设置年历标题。然后将每个月的天数存放到数组 dm（下标从 0 开始），如果是闰年，则将 2 月份天数修正为 29。最后用循环语句将 12 个月的数据填充到相应的单元格。

在填充每个月的数据时，先用函数 DateSerial 生成该月第 1 天的日期型数据，用函数 Weekday 计算该日期是星期几，保存到变量 w 中。这里用 1 表示星期日、2 表示星期一、3 表示星期二、4 表示星期三、5 表示星期四、6 表示星期五、7 表示星期六。然后计算该月份数据在工作表中的起始行号和列号，并根据起始行、列号和变量 w 的值依次填写该月的日期。

5.6　工作表函数与图形

本节先介绍在 VBA 程序中使用 Excel 工作表函数和处理图形对象的方法，之后给出两个应用案例。

5.6.1　在 VBA 中使用 Excel 工作表函数

直接在 Excel 编辑栏中输入的函数叫工作簿函数，其在 VBA 中使用时则被称为工作表函数（Worksheet Function）。

在 VBA 程序中，可以使用大多数 Excel 工作簿函数。各函数功能、参数和用法等详细内容可参考帮助信息。

1. 在 VBA 中调用工作表函数

在 VBA 程序中，通过 WorksheetFunction 对象可使用 Excel 工作表函数。

【例 5-36】 以下 Sub 过程使用 Min 工作表函数求出某个区域中的最小值。

在这段程序中，先将变量 myR 声明为 Range 对象，然后将其设置为 Sheet1 的 A1:C10 单元格区域。指定另一个变量 answer 为对 myR 应用 Min 函数的结果。最后将 answer 的值显示在消息框中。

```
Sub UF()
  Dim myR As Range
  Set myR = Worksheets("Sheet1").Range("A1:C10")
  answer = Application.WorksheetFunction.Min(myR)
  MsgBox answer
End Sub
```

注意： VBA 函数和 Excel 工作表函数可能同名，但作用和引用方式是不同的。例如，工作表函数 Log 和 VBA 函数 Log 是两个不同的函数。

2. 在单元格中插入工作表函数

若要在单元格中插入工作表函数，需指定函数作为相应的 Range 对象的 Formula 属

性值。

【例 5-37】 以下程序将 RAND 工作表函数（可生成随机数）赋给活动工作簿 Sheet1 上 A1:B3 区域的 Formula 属性。

```
Sub Fml()
  Worksheets("Sheet1").Range("A1:B3").Formula = "=RAND()"
End Sub
```

5.6.2 处理图形对象

图形对象包括 3 种类型：Shapes 集合、ShapeRange 集合和 Shape 对象。

通常，用 Shapes 集合可创建和管理图形，用 Shape 对象可修改单个图形或设置属性，用 ShapeRange 集合可同时管理多个图形。

若要设置图形的属性，必须先返回代表一组相关图形属性的对象，然后设置对象的属性。

【例 5-38】 下面的程序先使用 Fill 属性返回 FillFormat 对象，该对象包含指定图表或图形的填充格式属性。然后再使用 FillFormat 对象的 ForeColor 属性来设置指定图形的前景色。

```
Sub test()
  Worksheets(1).Shapes(1).Fill.ForeColor.RGB = RGB(255, 0, 0)
End Sub
```

通过选定图形，然后使用 ShapeRange 属性来返回包含选定图形的 ShapeRange 对象，可创建包含工作表上所有 Shape 对象的 ShapeRange 对象。

【例 5-39】 下面的程序创建选定图形的 ShapeRange 对象，然后填充绿色。

注意：要先选中一个或多个图形。

```
Sub test()
  Set sr = Selection.ShapeRange
  sr.Fill.ForeColor.SchemeColor = 17
End Sub
```

【例 5-40】 假设在 Excel 当前工作簿的第 1 张工作表上创建了 2 个图形，并分别命名为“Spa”和“Spb”。下面的程序在工作表上构造包含图形“Spa”和“Spb”的图形区域，并对这 2 个图形应用渐变填充格式。

```
Sub test()
  Set myD = Worksheets(1)
  Set myR = myD.Shapes.Range(Array("Spa", "Spb"))
  myR.Fill.PresetGradient msoGradientHorizontal, 1, msoGradientBrass
End Sub
```

在 Shapes 集合或 ShapeRange 集合中循环，也可以对集合中的单个 Shape 对象进行处理。

【例 5-41】 下面的程序在当前工作簿的第 1 张工作表上对所有图形进行循环，更改每个自选图形的前景色。

```
Sub test()
  Set myD = Worksheets(1)
  For Each sh In myD.Shapes
    If sh.Type = msoAutoShape Then
       sh.Fill.ForeColor.RGB = RGB(255, 0, 0)
    End If
  Next
End Sub
```

【例 5-42】 下面的程序对当前活动窗口中所有选定的图形构造一个 ShapeRange 集合，并设置每个选定图形的填充色。

注意：事先要选中一个或多个图形。

```
Sub test()
  For Each sh In ActiveWindow.Selection.ShapeRange
    sh.Fill.Visible = msoTrue
    sh.Fill.Solid
    sh.Fill.ForeColor.SchemeColor = 57
  Next
End Sub
```

5.6.3　多元一次方程组求解

下面在 Excel 中设计一个小软件，它可对任意一个多元一次方程组求解。

1．界面初始化程序设计

可以把任意一个多元一次联立方程组分为 3 部分：系数矩阵 a、向量 b、解向量 x。例如，二元一次联立方程式

$$\begin{cases} X+Y=16 \\ 2X+4Y=40 \end{cases}$$

上述的系数矩阵 a、向量 b、解向量 x 如图 5-14 所示。

a		b	x
1	1	16	12
2	4	40	4

图 5-14　系数矩阵 a、向量 b、解向量 x

为了便于输入任意一个多元一次联立方程组的系数矩阵 a、向量 b，输出解向量 x，需要在 Excel 工作表中设置单元格区域、清除原有数据，并进行必要的属性设置。可用下面的初始化子程序实现：

```
Sub init()
  '指定阶数 n
  n = InputBox("请输入方程组的阶数：")
```

```
    '清除工作表内容和背景颜色
    Cells.ClearContents
    Cells.Interior.ColorIndex = xlNone
    '设置系数矩阵标题及背景颜色
    Cells(1, 1) = "A1"
    Cells(1, 2) = "A2"
    rg = "A1:" & Chr(64 + n) & 1
    Cells(1, 1).AutoFill Destination:=Range(rg)
    Range(rg).Interior.ColorIndex = 33
    '设置向量 B 标题及背景颜色
    Cells(1, n + 1) = "B"
    Cells(1, n + 1).Interior.ColorIndex = 46
    '设置解向量 X 标题及背景颜色
    Cells(1, n + 2) = "X"
    Cells(1, n + 2).Interior.ColorIndex = 43
    '设置系数矩阵区域背景颜色
    rg_a = "A2:" & Chr(64 + n) & (n + 1)
    Range(rg_a).Interior.ColorIndex = 35
    '设置向量 B 区域背景颜色
    rg_b = Chr(64 + n + 1) & "2:" & Chr(64 + n + 1) & (n + 1)
    Range(rg_b).Interior.ColorIndex = 36
    '设置解向量 X 区域背景颜色
    rg_x = Chr(64 + n + 2) & "2:" & Chr(64 + n + 2) & (n + 1)
    Range(rg_x).Interior.ColorIndex = 34
End Sub
```

上述子程序首先用 InputBox 函数将方程的阶数指定给变量 n，清除工作表所有内容和背景颜色。然后设置系数矩阵 a、向量 b、解向量 x 标题及背景颜色。最后设置系数矩阵 a 区域、向量 b 区域、解向量 x 区域的背景颜色。其中用到了 AutoFill 方法进行序列数据自动填充。例如，当指定方程的阶数为 4 时，得到的界面如图 5-15 所示。

	A	B	C	D	E	F
1	A1	A2	A3	A4	B	X
2						
3						
4						
5						

图 5-15　指定方程的阶数为 4 时的界面

2．求解方程组程序设计

求解的原理很简单：先计算系数矩阵 a 的逆矩阵，再与向量 b 进行矩阵相乘就得到了向量 x。而矩阵求逆和相乘的功能可分别由工作表函数 MInverse 和 MMult 直接完成。

为了实现对任意一个多元一次方程组求解，还需要考虑方程组无解的情况，这可以通过检查系数矩阵的行列式值是否为零来判断。矩阵行列式求值可由工作表函数 MDeterm 来完成。

求解方程组子程序的具体代码如下：

```
Sub calc()
  n = Range("A1").End(xlDown).Row - 1                          '方程的阶数
  rg_a = "A2:" & Chr(64 + n) & (n + 1)                         '系数矩阵区域
  rg_b = Chr(64+n+1) & "2:" & Chr(64 + n + 1) & (n + 1)        '向量 B 区域
  rg_x = Chr(64+n+2) & "2:" & Chr(64 + n + 2) & (n + 1)        '解向量 X 区域
  a = WorksheetFunction.MDeterm(Range(rg_a))                   '求矩阵行列式的值
  If a = 0 Then
    MsgBox "方程组无解！"
  Else
    b = WorksheetFunction.MInverse(Range(rg_a))                '求矩阵的逆矩阵
    c = WorksheetFunction.MMult(b, Range(rg_b))                '求两矩阵乘积
    Range(rg_x).Value = c
  End If
End Sub
```

上述程序首先根据当前工作表有效数据区的行号求出方程的阶数 n，确定系数矩阵 a、向量 b、解向量 x 对应的单元格区域 rg_a、rg_b 和 rg_x。然后分别用工作表函数 MDeterm、MInverse 和 MMult 求矩阵行列式的值、逆矩阵和两矩阵乘积。最后将结果填写到解向量 x 对应的区域。

程序运行后的结果如图 5-16 和图 5-17 所示。

	A	B	C	D	E	F
1	A1	A2	A3	A4	B	X
2	1	1	1	1	5	1
3	1	2	-1	4	-2	2
4	2	-3	-1	-5	-2	3
5	3	1	2	11	0	-1

图 5-16　程序运行结果之一

	A	B	C	D
1	A1	A2	B	X
2	1	1	16	12
3	2	4	40	4

图 5-17　程序运行结果之二

5.6.4　创建动态三维图表

创建一个 Excel 工作簿，在第 1 张工作表中输入图 5-18 所示的数据。

选中 A3:D9 区域，在“插入”选项卡“图表”选项组中单击“柱形图”按钮，选择“三维柱形图”，将图表插入到当前工作表，如图 5-19 所示。

	A	B	C	D
1				
2				
3	年度	食品	服装	电器
4	2016年	3454	5554	6677
5	2017年	3450	4575	5678
6	2018年	4565	7667	8766
7	2019年	4557	6832	8766
8	2020年	5766	6543	9011
9	2021年	6900	7676	8766

图 5-18　创建图表需要的数据区

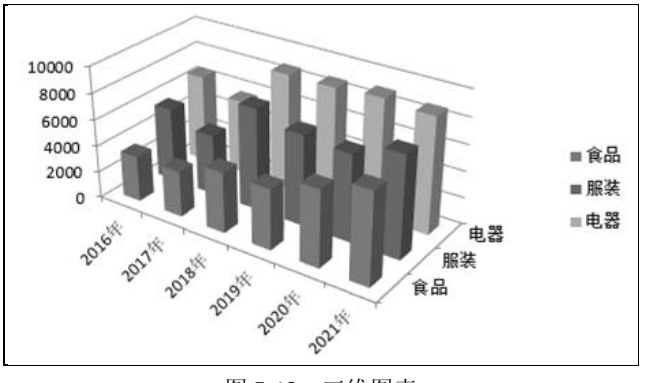

图 5-19　三维图表

进入 VB 编辑环境，编写如下子程序：

```
Sub 动态效果()
  Set Gbj = Sheets(1).ChartObjects(1).Chart
  RoSpeed = 0.3                        '设置步长
  For k = 0 To 35 Step RoSpeed         '正向旋转
    Gbj.Rotation = k: DoEvents
  Next
  For k = 0 To 45 Step RoSpeed         '正向仰角
    Gbj.Elevation = k: DoEvents
  Next
  For k = 35 To 0 Step RoSpeed * -1'反向旋转
    Gbj.Rotation = k: DoEvents
  Next
  For k = 45 To 0 Step RoSpeed * -1'反向仰角
    Gbj.Elevation = k: DoEvents
  Next
End Sub
```

上述程序首先将第 1 张工作表中第 1 个图表的图表区赋值给对象变量 Gbj，设置一个步长值并赋给变量 RoSpeed。然后分别用循环语句控制图表区进行正向旋转、正向仰角、反向旋转、反向仰角变换。其中，DoEvents 语句的作用是让出系统控制权，达到动态刷新图表的目的。

运行这个子程序将会看到图表的动态变化效果。

5.7 在工作表中使用控件

在 Excel 工作表中可以放置命令按钮、文本框、复选框、列表框等控件，也可以创建用户窗体，在用户窗体中放置需要的控件实现特定的功能。

本节介绍在 Excel 工作表中放置控件、设置控件属性以及用 VBA 程序对控件进行操作的方法。

在 Excel 2016 中，控件分为两种：表单控件和 ActiveX 控件。在“开发工具”选项卡“控件”选项组中单击“插入”按钮，可以看到图 5-20 所示的控件列表，其中上半部分为表单控件，下半部分为 ActiveX 控件。

图 5-20 控件列表

1. 表单控件

在 Excel 2016 中，表单控件有 12 个，其中 9 个是可以放到工作表上的控件，分别是：

- “标签”，表示静态文本。
- “分组框”，用于组合其他控件。
- “按钮”，用于运行宏命令。
- “复选框”，是一个选择控件，通过单击可以选中和取消选中，可以多项选择。
- “选项按钮”，通常几个组合在一起使用，在一组中只能选择一个选项按钮。

 “列表框”，用于显示多个选项供选择。

 “组合框”，用于显示多个选项供选择。可以选择其中的项目或者输入一个其他值。

 “滚动条”，是一种选择控制机制。包括水平滚动条和垂直滚动条。

 “数值调节钮”，是一种数值选择机制。通过单击控件的箭头来选择数值。

要将表单控件添加到工作表，可以单击需要的控件，待光标变成十字形状时，在当前工作表的适当位置按住鼠标左键并拖动，画出一个代表控件大小的矩形，大小满意后松开鼠标，这样一个控件就被添加到工作表上了。

右击控件，在弹出的快捷菜单中选择“设置控件格式”命令，可设置控件的格式。不同控件格式各不相同。

例如，滚动条控件的“设置控件格式”对话框中有一个“控制”选项卡，在“单元格链接”编辑框中输入或选中一个单元格地址，单击“确定”按钮后，再单击其他任意单元格，即可退出设计状态。接下来单击滚动条上的微调按钮，则指定单元格的数值会随之改变。

复选框控件的“设置控件格式”对话框中还有一个“控制”选项卡，在“单元格链接”编辑框中输入或选中一个单元格地址，单击“确定”按钮后，再单击其他单元格，即可退出设计状态。接下来单击复选框，对应的单元格出现 TRUE，表示该控件被选中，再次单击该控件，出现 FALSE，表示该控件未被选中。

创建控件时，Excel 会自动给它指定一个名字。为便于理解和记忆，可以给它重新起一个名字。要给控件改名，只需要右击选中控件，在弹出的快捷菜单中选择“编辑文字”命令，即可编辑控件名字。

右击控件，在弹出的快捷菜单中选择“指定宏”命令，可以为控件指定宏。这样在控件上单击就可以执行相应的 VBA 程序了。

2. ActiveX 控件

其中，“命令按钮”相当于表单控件的“按钮”，数值调节钮、复选框、选项按钮、列表框、组合框、滚动条、标签与表单控件作用相同。

 “文本框”用来输入或显示文本信息。

 “切换按钮”可以在“按下”和“抬起”两种状态中切换和锁定，不像普通“命令按钮”那样只能锁定一种状态，但作用与“命令按钮”相似。

 “图像”用来放置图片。

在“开发工具”选项卡的“控件”选项组中有一个“设计模式”按钮，它有两种状态：该按钮被按下时，工作表上的控件处于设计模式，可以对控件的属性、代码等进行设计；该按钮抬起时，工作表上的控件为运行模式，可执行代码，完成相应的动作。

在“开发工具”选项卡的“控件”选项组中单击“属性”按钮，可以打开“属性”窗口，设置或显示控件的属性。在设计模式下，右击某一控件，在弹出的快捷菜单中选择“属性”命令，也可以打开“属性”窗口，而且直接列出该控件的属性。

在“开发工具”选项卡的“控件”选项组中单击“查看代码”按钮，可以进入 VB 编辑环境，查看或编写控件的代码。在设计模式下，右击某一控件，在弹出的快捷菜单中选择“查看代码”命令，也可以直接查看或修改该控件的代码。

单击“其他控件”按钮，可以在列表框中选择更多的控件。

3. 在工作表上处理控件

Excel 中用 OLEObjects 集合的 OLEObject 对象代表 ActiveX 控件。若要用编程的方式向工作表添加 ActiveX 控件，可用 OLEObjects 集合的 Add 方法。

【例 5-43】 下面的程序向当前工作簿的第 1 张工作表添加命令按钮。

```
Sub acb()
  Worksheets(1).OLEObjects.Add "Forms.CommandButton.1", _
  Left:=200, Top:=200, Height:=20, Width:=100
End Sub
```

大多数情况下，VBA 代码可用名称引用 ActiveX 控件。例如，下面的语句可更改控件的标题。

```
Sheet1.CommandButton1.Caption = "运行"
```

下面的语句可设置控件的左侧定位。

```
Worksheets(1).OLEObjects("CommandButton1").Left = 10
```

下面的语句也可设置控件的标题。

```
Worksheets(1).OLEObjects("CommandButton1").Object.Caption = "run me"
```

工作表上的 ActiveX 控件具有两个名称。一个是可以在工作表“名称”框中看到的图形名称，另一个是可以在“属性”窗口中看到的代码名称。在控件的事件过程名称中使用的是控件代码名称，从工作表的 Shapes 或 OLEObjects 集合中返回控件时，使用的是图形名称。二者通常情况下保持一致。

例如，假定要向工作表中添加一个复选框，其默认的图形名称和代码名称都是 CheckBox1。如果在“属性”窗口中将控件名称改为 CB1，那么图形名称也会同时改为 CB1。此后，在事件过程名称中需用 CB1，也要用 CB1 从 Shapes 或 OLEObject 集合中返回控件，语句如下：

```
ActiveSheet.OLEObjects("CB1").Object.Value = 1
```

5.8 使用 Office 命令栏

在 Microsoft Office 中，工具栏、菜单栏和快捷菜单都可由同一种类型的对象进行编程控制，这类对象就是命令栏（CommandBar）。

通过 VBA 程序，可以为应用程序创建和修改自定义工具栏、菜单栏和快捷菜单栏，还可以为命令栏添加按钮、文字框、列表框和组合框等控件。

命令栏控件和 ActiveX 控件尽管具有相似的外观和功能，但两者并不相同。所以既不能在命令栏中添加 ActiveX 控件，也不能在文档或表格中添加命令栏控件。

5.8.1　自定义工具栏

利用 VBA 代码可以创建和修改工具栏。例如，改变按钮的状态、外观、功能，添加或修改组合框控件等。

每个按钮控件都有两种状态：按下状态（True）和未按下状态（False）。要改变按钮控件的状态，可为 State 属性赋予适当的值。也可以改变按钮的外观或功能。要改变按钮的外观而不改变其功能，可用 CopyFace 和 PasteFace 方法。CopyFace 方法将某个特殊按钮的图符复制到剪贴板，PasteFace 方法将按钮图符从剪贴板粘贴到指定的按钮上。要将按钮的动作改为自定义的功能，可为该按钮的 OnAction 属性指定一个自定义过程名。

表 5-7 列举了命令栏按钮常用的属性和方法。

表 5-7　命令栏按钮常用的属性和方法

属性或方法	说　　明
CopyFace	将指定按钮的图符复制到“剪贴板”上
PasteFace	将“剪贴板”上的图符粘贴到指定按钮上
Id	代表按钮内置函数的值
State	按钮的外观或状态
Style	按钮图符显示其图标还是显示其标题
OnAction	指定在单击按钮、显示菜单或更改组合框控件的内容时运行的过程
Visible	对象是否可见
Enabled	对象是否有效

1. 改变按钮外观

【例 5-44】 创建包含一个命令按钮的命令栏，用代码改变命令栏按钮外观。

进入 Excel 的 VB 编辑环境，插入一个模块，在模块中输入如下 3 个过程：

```
Sub CreateCB()
  Set myBar = CommandBars.Add(Name:="cbt")
  myBar.Visible = True
  Set oldc = myBar.Controls.Add(Type:=msoControlButton, ID:=23)
  oldc.OnAction = "ChangeFaces"
End Sub
Sub ChangeFaces()
  Set newc = CommandBars.FindControl(Type:=msoControlButton, ID:=19)
  newc.CopyFace
  Set oldc = CommandBars("cbt").Controls(1)
  oldc.PasteFace
End Sub
Sub DelCB()
  CommandBars("cbt").Delete
End Sub
```

另创建两个窗体控件按钮，分别为“创建工具栏”——指定 CreateCB 过程；“删除工

具栏”——指定 DelCB 过程，用来配合命令按钮的运行。

过程 CreateCB 首先用 Add 方法创建一个工具栏，命名为 cbt。然后让工具栏可见。接下来在工具栏中添加一个按钮，设置按钮的 ID 值为 23（对应于“打开”按钮）。最后通过命令栏按钮对象的 OnAction 属性，指定其执行的过程为 ChangeFace。

ChangeFace 过程首先找到 Excel 系统中 ID 为 19 的工具栏按钮，然后用 CopyFace 方法将该按钮的图符复制到“剪贴板”上，再用 PasteFace 方法将其粘贴到 cbt 工具栏的按钮上。这样就在运行时修改了命令栏按钮的外观。

过程 DelCB 用 Delete 方法删除工具栏 cbt。

运行 CreateCB 过程，Excel 功能区中会增加一个“加载项”选项卡，其中有一个“自定义工具栏”选项组，上面有一个按钮  。单击这个按钮，外观变为  。

运行 DelCB 过程，功能区上的“加载项”选项卡消失。

2. 使用图文按钮

【例 5-45】 创建一个自定义工具栏，添加两个图文型按钮。

创建一个 Excel 工作簿，进入 VB 编辑环境。在当前工程的 Microsoft Excel 对象中，双击 ThisWorkbook。在代码编辑窗口上方的“对象”下拉列表中，选择 Workbook，在“过程”下拉列表中选择 Open，对工作簿的 Open 事件编写如下代码：

```
Private Sub Workbook_Open()
  Set tbar = Application.CommandBars.Add(Temporary:=True)
  With tbar.Controls.Add(Type:=msoControlButton)
    .Caption = "统计"                    '按钮文字
    .FaceId = 16                         '按钮图符
    .Style = msoButtonIconAndCaption     '图文型按钮
    .OnAction = "tj"                     '执行的过程
  End With
  With tbar.Controls.Add(Type:=msoControlButton)
    .Caption = "增项"
    .FaceId = 12
    .Style = msoButtonIconAndCaption
    .OnAction = "zx"
  End With
  tbar.Visible = True
End Sub
```

当工作簿打开时，产生 Open 事件，执行上述代码。

这段代码首先创建一个自定义工具栏，设置临时属性（关闭当前工作簿后，工具栏自动删除）。然后在工具栏上添加两个图文型按钮，分别设置按钮的标题、图符和要执行的过程。

插入一个模块。在模块中编写以下两个过程：

```
Sub tj()
  MsgBox "统计功能！"
End Sub
```

```
Sub zx()
  MsgBox "增项功能！"
End Sub
```

这样，当打开该工作簿时，Excel 功能区中会自动出现一个“加载项”选项卡，其中有一个“自定义工具栏”选项组，上面有两个图文按钮 统计 和 增项，单击按钮，可显示相应的提示信息。

3. 使用组合框

编辑框、列表框和组合框都是功能强大的控件，可以添加到 VBA 应用程序的工具栏中，这通常需要用 VBA 代码来完成。

要设计一个组合框，需要用到表 5-8 所示的属性和方法。

表 5-8　组合框常用的属性和方法

属性或方法	说　明
Add	在命令栏中添加控件，可设置 Type 参数为：msoControlEdit、msoControlDropdown 或 msoControlComboBox
AddItem	在列表框或组合框中添加列表项
Caption	为组合框控件指定标签。Style 属性设置为 msoComboLabel，则该标签在控件旁显示
Style	确定指定控件的标题是否在该控件旁显示：msoComboLabel 显示；msoComboNormal 不显示
OnAction	指定当用户改变组合框控件的内容时要运行的过程

【例 5-46】 在自定义工具栏中添加一个组合框。

创建一个 Excel 工作簿，进入 VB 编辑环境，插入一个模块。在模块中，首先用下面的语句声明一个模块级对象变量 newCombo，用来表示自定义工具栏上的组合框。

```
Dim newCombo As Object
```

然后，编写如下过程：

```
Sub 创建工具栏()
  Set myBar = CommandBars.Add(Temporary:=True)
  myBar.Visible = True
  Set newCombo = myBar.Controls.Add(Type:=msoControlComboBox)
  With newCombo
    .AddItem "Q1"
    .AddItem "Q2"
    .AddItem "Q3"
    .AddItem "Q4"
    .Style = msoComboLabel
    .Caption = "请选择一个列表项："
    .OnAction = "stq"
  End With
End Sub
```

上述过程首先创建一个自定义工具栏，设置临时属性，使其可见。然后在工具栏中创建一个组合框，添加 4 个列表项，在旁边显示标题，指定当用户改变组合框控件的内容时要运行的过程 stq。

最后，编写 stq 过程如下：

```
Sub stq()
  k = newCombo.ListIndex
  MsgBox "选择了组合框的第" & k & "项！"
End Sub
```

上述子程序，通过模块级变量 newCombo 引用工具栏上的组合框，由组合框的 ListIndex 属性得到选项的序号，用 MsgBox 显示相应的信息。

运行“创建工具栏”过程，在 Excel 功能区中会自动出现一个“加载项”选项卡，其中有一个“自定义工具栏”选项组，上面有一个组合框，组合框的左边显示标题“请选择一个列表项：”。在组合框中选择任意一个列表项，将会显示相应的提示信息。

5.8.2 选项卡及工具栏按钮控制

下面，创建一个具有 3 张工作表的 Excel 工作簿，通过 VBA 程序实现以下功能：

当工作簿打开时，自动创建一个临时自定义工具栏。工具栏上放置 1 个组合框、2 个按钮。选中第 1 张工作表时，激活功能区的“开始”选项卡；选中第 2 张工作表时，激活功能区的“加载项”选项卡，组合框和第 1 个按钮可用，第 2 个按钮不可用；选中第 3 张工作表时，激活功能区的“加载项”选项卡，组合框和第 2 个按钮可用，第 1 个按钮不可用。选择组合框的任意一个列表项，该列表项文本将被添加到当前单元格区域。单击两个按钮，分别显示不同的提示信息。

首先创建一个 Excel 工作簿，保存为“选项卡及工具栏按钮控制.xlsm”。

然后，在 VB 编辑环境中，单击工具栏中的“工程资源管理器”按钮，在当前工程中的“Microsoft Excel 对象”中双击“ThisWorkbook”，对当前工作簿进行编程。

在代码编辑窗口上方的“对象”下拉列表框中选择 Workbook，在“过程”下拉列表框中选择 Open，对工作簿的 Open 事件编写如下代码：

```
Private Sub Workbook_Open()
  Set tbar = Application.CommandBars.Add(Temporary:=True)
  Set combx1 = tbar.Controls.Add(Type:=msoControlComboBox)
  With combx1
    .Width = 200
    .DropDownLines = 8
    .OnAction = "fill"
    .AddItem ("信息科学技术")
    .AddItem ("软件工程")
    .AddItem ("电子信息工程")
  End With
  Set butt1 = tbar.Controls.Add(Type:=msoControlButton)
  With butt1
```

```
    .Caption = "各省学生人数"
    .Style = msoButtonCaption
    .OnAction = "gsrs"
  End With
  Set butt2 = tbar.Controls.Add(Type:=msoControlButton)
  With butt2
    .Caption = "教材发放情况"
    .Style = msoButtonCaption
    .OnAction = "jcff"
  End With
  tbar.Visible = True
  Worksheets(1).Activate
End Sub
```

当工作簿打开时，上述程序被自动执行，完成以下操作：

（1）创建一个临时自定义工具栏，用对象变量 tbar 表示。设置自定义工具栏的临时属性，是为了不影响 Excel 系统环境，在工作簿打开时创建，工作簿关闭时删除。

（2）在工具栏中添加一个组合框，保存到对象变量 combx1 中。设置组合框的宽度、列表项目数，添加 3 个列表项，指定要执行的过程为 fill。

（3）在工具栏中添加 2 个按钮，保存到对象变量 butt1 和 butt2 中。标题分别为“各省学生人数”和“教材发放情况”。为按钮分别指定要执行的过程为 gsrs 和 jcff。

（4）让自定义工具栏可见，选中第 1 张工作表。

为了在选中不同工作表的情况下，激活不同的选项卡，控制工具栏按钮的可用性，可对工作簿的 SheetActivate 事件编写如下代码：

```
Private Sub Workbook_SheetActivate(ByVal Sh As Object)
  Select Case Sh.Index
    Case 1
      Application.SendKeys "%H{F6}"
    Case 2
      Application.SendKeys "%X{F6}"
      butt1.Enabled = True
      butt2.Enabled = False
    Case Else
      Application.SendKeys "%X{F6}"
      butt1.Enabled = False
      butt2.Enabled = True
  End Select
End Sub
```

上述代码在工作簿的当前工作表改变时被执行。

如果当前选中的是第 1 张工作表，激活功能区的“开始”选项卡；是第 2 张工作表，激活功能区的“加载项”选项卡，第 1 个按钮可用，第 2 个按钮不可用；是第 3 张工作表，激活功能区的“加载项”选项卡，第 2 个按钮可用，第 1 个按钮不可用。组合框的 Enabled

属性默认值为 True，因此始终可用。

Microsoft 没有提供直接用 VBA 激活功能区选项卡的方法。但是，可以使用 SendKeys 方法模拟按键，来激活需要的选项卡。

例如，按 Alt 键，然后按 H 键，可激活“开始”选项卡。在功能区中会有这些按键的提示。如果要隐藏按键提示，只需要按 F6 键。

语句 Application.SendKeys "%H{F6}"发送按键信息，激活“开始”选项卡。

其中，“%H”相当于 Alt+H 键，“{F6}”相当于 F6 键。

同样道理，语句 Application.SendKeys "%X{F6}"可以激活“加载项”选项卡。

由于对象变量 combx1、butt1 和 butt2 在工作簿的 Open 事件中被赋值，而在其他过程中引用，因此要把它们声明为全局变量。

在 VB 编辑环境中，用“插入”菜单插入一个模块。在模块的顶部用下面语句声明全局型对象变量：

```
Public combx1, butt1, butt2 As Object
```

最后，在模块中编写以下 3 个过程：

```
Sub fill()
  Selection.Value = combx1.Text
End Sub
Sub gsrs()
  MsgBox "统计各省学生人数模块"
End Sub
Sub jcff()
  MsgBox "统计教材发放情况模块"
End Sub
```

这样，当选择组合框的任意一个列表项时，该列表项文本都会被添加到当前单元格区域中。单击 2 个按钮，将分别显示不同的提示信息。

5.8.3 自定义菜单

本小节在 Excel 工作簿中创建一个图 5-21 所示的自定义菜单。工作簿打开时，“维护”按钮自动出现在“加载项”选项卡中，选择“输入”“修改”“删除”命令时可显示出相应的信息，选择“退出”命令，则删除“加载项”选项卡。

图 5-21 自定义菜单

实现方法如下：

（1）在 Excel 环境中，选择“开发工具”选项卡“代码”选项组中的“Visual Basic”命令，或按 Alt+F11 键，打开 VB 编辑器。

（2）打开“工程资源管理器”，双击“Microsoft Excel 对象”的“ThisWorkbook”，打开代码编辑器窗口，在上面的“对象”下拉列表中选择“Workbook”，在“过程”下拉列表中选择“Open”，输入代码，得到如下过程：

```
Private Sub Workbook_Open()
  Set mb = MenuBars.Add("MyMenu")                          '创建菜单栏
  Set mt = mb.Menus.Add("维护")                             '添加水平菜单项
  mt.MenuItems.Add Caption:="输入", OnAction:="in_p"       '添加竖直菜单项
  mt.MenuItems.Add Caption:="修改", OnAction:="modi"
  mt.MenuItems.Add Caption:="删除", OnAction:="dele"
  mt.MenuItems.Add Caption:="退出", OnAction:="quit"
  mb.Activate                                              '激活自定义菜单
End Sub
```

（3）在 VB 编辑环境的“标准”工具栏中单击“模块”按钮，或选择“插入”菜单中的“模块”命令，插入一个模块。在模块中输入如下 4 个过程：

```
Sub in_p()
  MsgBox ("执行输入功能")
End Sub
Sub modi()
  MsgBox ("执行修改功能")
End Sub
Sub dele()
  MsgBox ("执行删除功能")
End Sub
Sub quit()
  MenuBars("MyMenu").Delete '删除自定义菜单
End Sub
```

（4）保存工作簿。

再次打开这个工作簿时，“加载项”选项卡中将出现自定义菜单。选择“输入”“修改”“删除”命令时，会显示相应的提示信息，选择“退出”命令，则删除“加载项”选项卡。

上机练习

1. 在 Excel 中编写程序，自动生成指定年月的月历。例如，指定 2022 年 2 月，得到图 5-22 所示的月历。

2022年2月

星期日	星期一	星期二	星期三	星期四	星期五	星期六
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28					

图 5-22　月历样板

2. 在 Excel 工作表中，设计图 5-23 所示的界面。然后，编写一个求任意一元二次方程根的子程序并指定给“求解”按钮，编写一个清除方程系数和根的子程序并指定给“清除”按钮。例如，输入方程的系数为 5、8、6，单击“求解”按钮，应得到图 5-24 所示的结果。单击“清除”按钮，界面恢复到图 5-23 所示的情形。

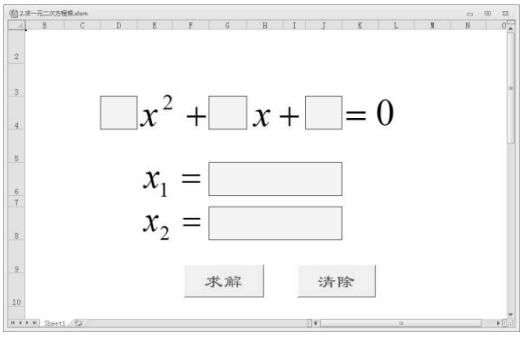

图 5-23　工作表界面

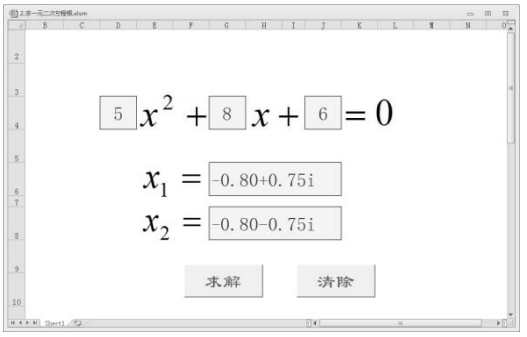

图 5-24　方程求解后的界面

3. 在 Excel 工作簿中编写程序，将当前工作表第 1 行从指定位置 m 开始的 n 个数按相反顺序重新排列。例如，原数列为：1，2，3，4，5，6，7，8，9，10，11，12，13，14，15，16，17，18，19，20。从第 5 个数开始，将 10 个数进行逆序排列，则得到新数列为：1，2，3，4，14，13，12，11，10，9，8，7，6，5，15，16，17，18，19，20。

4. 编写一个程序，提取字符串中的数字符号。例如，程序运行后输入字符串“abc123edf456gh”，则输出“123456”。

5. 在 Excel 中编写一个函数，返回指定区域中多个最大值地址。例如，图 5-25 所示的 B3:K3 区域及其数值对应的函数返回值应为“C1,F3,I3”。

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
1																						
2		数据区											地址区									
3		8	12	7	6	12	8	7	12	3	7		C3, F3, I3									
4		9	7	5	5	9	7	6					B4, F4									
5		3	1	2	8	7	5	6	7	8			E5, J5									
6																						

图 5-25　Excel 单元格区域及其数值