

目 录

一、VBA 语言基础	1
第一节 标识符	1
第二节 运算符	1
第三节 数据类型	1
第四节 变量与常量	1
第五节 数组	2
第六节 注释和赋值语句	2
第七节 书写规范	2
第八节 判断语句	2
第九节 循环语句	3
第十节 其他类语句和错误语句处理	4
第十一节 过程和函数	4
一. Sub 过程	4
二. Function 函数	5
三. Property 属性过程和 Event 事件过程	5
第十二节 内部函数	5
一. 测试函数	5
二. 数学函数	5
三. 字符串函数	6
四. 转换函数	6
五. 时间函数	6
第十三节 文件操作	6
文件	6
删除	7
打开	7
读入	7
写入	7
关闭	7
其他文件函数	7
二、VISUAL BASIC 程序设计网络教学	1
第一课 VBA 是什么	1
1.1 VBA 是什么	1
1.2 EXCEL 环境中基于应用程序自动化的优点	1
1.3 录制简单的宏	2
1.4 执行宏	2
1.5 查看录制的代码	2
1.6 编辑录制的代码	3
1.7 录制宏的局限性	4
1.8 小结	4
第二课 处理录制的宏	4
2.1 为宏指定快捷键	4
2.2 决定宏保存的位置	5
2.3 个人宏工作簿	5
2.3.1 保存宏到个人宏工作簿	5
2.3.2 使用并编辑个人宏工作簿中的宏	5
2.4 将宏指定给按钮	6
2.5 将宏指定给图片或其他对象	6
2.6 小结	7

第三课 学习控件.....	7
3.1 EXCEL 开发过程简介	7
3.2 认识不同的控件.....	7
3.3 向工作表添加控件.....	7
3.4 设置控件的特性.....	8
3.5 给控件命名.....	8
3.6 使用用户窗体.....	8
3.7 疑难解答.....	9
第四课 理解变量和变量的作用.....	9
4.1 代码存在的位置：模块.....	9
4.2 对模块的概览.....	10
4.2.1 创建过程.....	10
4.2.2 运行宏.....	11
4.3 保存对模块所做的改变.....	11
4.4 变量.....	11
4.4.1 变量的数据类型.....	11
4.4.2 用 Dim 语句创建变量(声明变量).....	13
4.4.3 变量命名的惯例.....	13
4.4.4 使用数组.....	14
4.4.5 变量赋值.....	15
第五课 利用 VBA 设置工作表使用权限	15
1. 使用 WITH 语句。	17
2. 使用对象变量。	17
方法 3：减少对象的激活和选择	17
方法 4：关闭屏幕更新	18
第六课 提高 EXCEL 中 VBA 的效率	19
方法 1：尽量使用 VBA 原有的属性、方法和 WORKSHEET 函数.....	19
方法 2：尽量减少使用对象引用，尤其在循环中.....	19
1. 使用 With 语句。	19
2. 使用对象变量。	20
3. 在循环中要尽量减少对象的访问。	20
方法 3：减少对象的激活和选择	20
方法 4：关闭屏幕更新	20
第七课 如何在 EXCEL 里使用定时器.....	20
三、学习微软 EXCEL 2002 VBA 编程和 XML，ASP 技术.....	22
第一章 电子表格自动化简介和了解宏命令.....	22
1 了解宏	22
2 宏命令的普通应用.....	22
3 写宏之前的计划	23
4 录制宏	24
5 运行宏	26
6 修改宏代码	26
7 添加注释.....	29
8 分析宏代码	29
9 清除宏代码	30
10 测试修改好的宏	31
11 两个层面运行宏的方法	32
12 完善你的宏代码	32

13 重新命名宏	34
14 运行宏的其它方法	34
15 使用键盘快捷键运行宏	34
16 通过菜单运行宏	35
17 通过工具栏按钮运行宏	37
18 通过工作表里面的按钮运行宏	38
19 保存宏	39
20 打印宏	39
21 保存宏在个人宏工作簿	40
22 打开含有宏的工作簿	41
23 VB 编辑窗口	42
24 了解工程浏览窗口	42
25 了解属性窗口	43
26 了解代码窗口	44
27 VB 编辑器里的其它窗口	45
28 接下来	46
第二章 VBA 第一步	46
1 了解指令, 模块和过程	46
2 VBA 工程命名	46
3 模块重命名	47
4 从其它工程调用过程	48
5 了解对象, 属性和方法	49
6 学习对象, 属性和方法	50
7 句法和文法	52
8 打断很长的 VBA 语句	54
9 了解 VBA 错误	54
10 查找帮助	56
11 语法和编程快捷助手	57
12 属性/方法列表	58
13 常数列表	58
14 参数信息	59
15 快速信息	59
16 自动完成关键字	59
17 缩进/凸出	60
18 设置注释块/解除注释块	60
19 使用对象浏览器	60
20 使用 VBA 对象库	65
21 用对象浏览器来定位过程	66
22 使用立即窗口	66
23 获取立即窗口里的信息	68
24 学习对象	69
25 电子表格单元格操作	69
26 使用 RANGE 属性	69
27 使用 CELLS 属性	69
28 使用 OFFSET 属性	70
29 选择单元格的其它方法	71
30 选择行和列	71
31 获取工作表信息	72
32 往工作表输入数据	72
33 返回工作表中的信息	72
34 单元格格式	72
35 移动, 复制和删除单元格	73

36 操作工作簿和工作表	74
37 操作窗口 (WINDOWS)	74
38 管理 EXCEL 应用程序	75
39 接下来.....	75
第三章 了解变量, 数据类型和常量.....	76
1 保存 VBA 语句的结果	76
2 变量是什么	76
3 数据类型.....	76
4 如何产生变量.....	77
5 如何声明变量.....	78
6 明确变量的数据类型	79
7 变量赋值.....	80
8 强制声明变量.....	82
9 了解变量范围.....	83
10 过程级别 (当地) 变量	83
11 模块级别变量	84
12 工程级别变量.....	85
13 变量的存活期.....	85
14 了解和使用静态变量	85
15 声明和使用对象变量	86
16 使用明确的对象变量	87
17 查找变量定义.....	87
18 在 VB 过程里面使用常量	87
19 内置常量.....	88
20 接下来.....	89
第四章 VBA 过程: 子程序和函数	89
1.关于函数过程.....	89
2.创建函数过程.....	89
3.执行函数过程.....	91
4.从工作表里运行函数过程	91
5.从另外一个 VBA 过程里运行函数过程.....	93
6.传递参数.....	93
7.明确参数类型.....	94
8.按地址和按值传递参数.....	95
9.使用可选的参数	96
10.定位内置函数.....	97
11.使用 MsgBox 函数	98
12.MsgBox 函数的运行值	101
13.使用 INPUTBOX 函数	102
14.数据类型转变.....	103
15.使用 INPUTBOX 方法	104
16.使用主过程和子过程	107
17.接下来.....	109
第五章 基于 VBA 做决定	109
1.关系和逻辑运算符.....	109
2.If...THEN 语句.....	110
3.基于多于一个条件的决定	112
4.The If...Then...Else 语句.....	113
5.If...Then...ElseIf 语句	116
6.嵌套的 If...Then 语句	117

7.SELECT CASE 语句.....	118
8.和 CASE 子句一起使用 Is	119
9.确定 CASE 子句里数值的范围	120
10.在 CASE 子句里确定多个表达式	121
11.接下来.....	121
第六章 在 VBA 中重复操作	121
1.Do Loops: Do...While 和 Do...Until	121
2.观察过程执行.....	124
3.While...Wend 循环.....	125
4.For...Next 循环.....	126
5.For Each...Next 循环.....	127
7.提前跳出循环.....	128
8.循环嵌套.....	129
9.接下来.....	129
第七章 利用 VBA 数组管理数据清单和表格	129
1.了解数组.....	130
2.声明数组.....	131
3.数组的上界和下界.....	131
4.在 VBA 过程里使用数组.....	131
5.数组和循环语句.....	132
6.使用二维数组.....	134
7.静态和动态数组	135
8.数组函数.....	137
9.Array 函数	137
10.IsArray 函数	137
11.Erase 函数	138
12.Lbound 函数和 Ubound 函数	139
13.数组中的错误.....	139
14.数组作为参数.....	141
15.接下来.....	141
第八章 利用 VBA 操纵文件和文件夹	141
1.获取当前文件夹的名称 (CurDir 函数)	142
2.更改文件或文件夹名称 (Name 函数)	142
3.检查文件或文件夹是否存在 (Dir 函数)	143
4.获得文件修改的日期和时间 (FileDateTime 函数)	144
5.获得文件大小 (FileLen 函数)	145
6.返回和设置文件属性 (GetAttr 函数和 SetAttr 函数)	145
7.更改缺省文件夹或驱动器 (ChDir 语句和 ChDrive 语句)	146
8.创建和删除文件夹 (MkDir 语句和 Rmdir 语句)	147
9.复制文件 (FileCopy 语句)	147
10.删除文件 (Kill 语句)	149
11.从文件读取和写入数据 (Input/Output)	149
12.文件访问类型.....	149
13.使用顺序文件.....	150
14.读取储存于顺序文件里的数据	150
15.逐行读取文件.....	150
16.从顺序文件中读取字符.....	151
17.读取分隔文本文件.....	152
18.往顺序文件里写数据	153
19.使用 Write # 和 Print # 语句.....	154

20.操作随机文件.....	155
21.创建用户定义的数据类型.....	155
22.操作二进制文件.....	158
23.操作文件和文件夹的时髦方法.....	159
24.使用 WSH 获取文件信息.....	162
25.FILESYSTEMOBJEC 的方法和属性.....	163
26.对象 FILE 的属性.....	166
27.文件夹对象属性.....	167
28.驱动器对象属性.....	167
29.使用 WSH 创建文本文件.....	168
30.使用 WSH 进行其它操作.....	170
31.运行其它应用程序.....	170
32.创建快捷方式.....	171
33.接下来.....	172
第九章 利用 VBA 控制其它应用程序.....	172
1.启动应用程序.....	172
2.在应用程序之间切换.....	175
3.控制其它应用程序.....	175
4.控制应用程序的其它方法.....	177
5.了解自动控制.....	177
6.了解链接和嵌入.....	178
7.使用 VBA 进行链接和嵌入.....	179
8.COM 和自动控制.....	180
9.了解绑定.....	180
10.后期绑定.....	180
11.早期绑定.....	180
12.建立到对象库的引用.....	181
13.创建自动控制对象.....	182
14.使用 CREATEOBJECT 函数.....	182
15.使用自动控制创建一个新的 WORD 文档.....	183
16.使用 GETOBJECT 函数.....	183
17.打开存在的 WORD 文档.....	184
18.使用关键字 NEW.....	185
19.使用自动控制访问 MICROSOFT OUTLOOK.....	186
20.接下来.....	187
第十章 对话框和自定义窗体.....	187
1.文件打开和另存为对话框.....	190
2.GETOPENFILENAME 和 GETSAVEASFILENAME 方法.....	194
3.创建窗体.....	195
4.创建用户窗体的工具.....	197
5.标签.....	198
6.文字框.....	198
7.框架.....	198
8.选项按钮.....	199
9.复选框.....	199
10.切换按钮.....	199
11.列表框.....	199
12.复合框.....	199
13.滚动条.....	199
14.旋转按钮.....	199
15.图像.....	199

16. 多页控件	200
17. TABSTRIP 控件	200
18. REFEDIT 控件	200
19. 在窗体上放置控件	200
20. 应用程序示例 1: 信息调查	200
21. 在窗体上添加按钮、选项框和其它控件	202
22. 更改控件名称	204
23. 设置其它控件属性	204
24. 准备工作表以储存窗体数据	205
25. 显示自定义窗体	206
26. 设置 TAB 顺序	206
27. 了解窗体和控件事件	207
28. 编写 VBA 过程对窗体和控件事件反应	209
29. 编写过程来初始化窗体	209
30. 编写过程填充列表框控件	211
31. 编写过程控制选项按钮	211
32. 编写过程同步文字框和旋转按钮	212
33. 编写过程关闭用户窗体	212
34. 转移窗体数据到工作表	213
35. 使用 INFO SURVEY 应用程序	214
36. 应用程序示例 2: 学生和考试	214
37. 使用多页和 TABSTRIP 控件	214
38. 给窗体 STUDENTSAND EXAMS 自定义窗体编写 VBA 过程	216
39. 使用自定义窗体 STUDENTSAND EXAMS	221
40. 接下来	223
第十一章 自定义集合和类模块	223
1. 使用集合	224
2. 声明自定义集合	225
3. 给自定义集合添加对象	225
4. 从自定义集合移出对象	226
5. 创建自定义对象	227
6. 创建类	227
7. 变量声明	227
8. 定义类的属性	228
9. 创建 PROPERTY GET 过程	228
10. 创建 PROPERTY LET 过程	229
11. 创建类方法	230
12. 创建类的示例	230
13. 类模块里的事件过程	230
14. 创建用户界面	231
15. 观察 VBA 过程的执行	240
16. 接下来	243
第十二章 使用 VBA 创建自定义菜单和工具栏	243
1. 工具栏	243
2. 创建自定义工具栏	245
3. 删除自定义工具栏	247
4. 使用 COMMANDBAR 的属性	247
5. 使用 COMMANDBAR 控件	247
6. 理解和使用控件属性	249
7. 控件方法	251
8. 使用菜单	252

9. 菜单编程.....	254
10. 创建子菜单	256
11. 修改内置快捷菜单	258
12. 创建快捷菜单.....	260
13. 接下来.....	262
第十三章 调试 VBA 过程和处理错误	262
1. 测试 VBA 过程.....	262
2. 终止过程.....	263
3. 使用断点.....	264
4. 在中断模式下使用立即窗口.....	267
5. 使用 STOP 语句.....	269
6. 添加监视表达式	269
7. 清除监视表达式	272
8. 使用快速监视.....	272
9. 使用本地窗口和调用堆栈对话框.....	273
10. 逐句运行 VBA 过程	274
11. 逐句运行过程	275
12. 逐过程执行过程	275
13. 设置下一条语句	276
14. 显示下一条语句	277
15. 终止和重新设置 VBA 过程	277
16. 了解和使用条件编译	277
17. 操纵书签.....	279
18. 捕捉错误.....	279
17. 接下来.....	283
第十四章 微软 EXCEL 2002 中的事件编程.....	283
1. 事件过程介绍.....	283
2. 激活和失活事件	285
3. 事件次序.....	287
4. 工作表事件	287
5. 工作簿事件	292
6. 图表事件.....	310
7. 内嵌图表事件.....	314
8. 可为应用软件对象识别的事件	315
9. 查询表时间	318
10. 接下来.....	321
第十五章 在 EXCEL 里使用 ACCESS.....	321
1. 对象库	321
2. 建立对对象库的引用	327
3. 链接到 ACCESS	327
4. 使用 AUTOMATION 链接到 ACCESS 数据库	329
5. 使用 DAO 链接到 ACCESS 数据库.....	332
6. 使用 ADO 链接到 ACCESS 数据库.....	332
7. 从 EXCEL 执行 ACCESS 任务	334
8. 创建新 ACCESS 数据库	334
9. 打开 ACCESS 窗体.....	336
10. 打开 ACCESS 报表	341
11. 运行 ACCESS 查询	343
12. 运行选择查询.....	345
13. 运行参数查询.....	346

14.调用 ACCESS 函数	348
15.获取 ACCESS 数据到 EXCEL 工作表	348
16.使用 GETROWS 方法获取数据	348
17.使用 COPYFROMRECORDSET 方法获取数据	352
18.使用 TRANSFERSPREADSHEET 方法获取数据	352
19.使用 OPENDATABASE 方法	353
20.从 ACCESS 数据创建文本文件	357
21.从 ACCESS 数据创建查询表	360
22.在 EXCEL 里使用 ACCESS 数据	362
23.用 ACCESS 数据创建内嵌图表	362
24.传输 EXCEL 电子表格到 ACCESS 数据库	364
25.将 EXCEL 电子表格链接到 ACCESS 数据库	366
26.将 EXCEL 电子表格导入 ACCESS 数据库	367
27.放置 EXCEL 数据到 ACCESS 表中	367
28.接下来..... ..	369

一、VBA 语言基础

Zhou Jibin 2004-11-30

第一节 标识符

一. 定义

标识符是一种标识变量、常量、过程、函数、类等语言构成单位的符号，利用它可以完成对变量、常量、过程、函数、类等的引用。

二. 命名规则

- 1) 字母打头，由字母、数字和下划线组成，如 A987b_23Abc
- 2) 字符长度小于 40，(Excel2002 以上中文版等，可以用汉字且长度可达 254 个字符)
- 3) 不能与 VB 保留字重名，如 public, private, dim, goto, next, with, integer, single 等

第二节 运算符

定义：运算符是代表 VB 某种运算功能的符号。

- 1) 赋值运算符 =
- 2) 数学运算符 &、+ (字符连接符)、+ (加)、- (减)、Mod (取余)、\ (整除)、* (乘)、/ (除)、- (负号)、^ (指数)
- 3) 逻辑运算符 Not (非)、And (与)、Or (或)、Xor (异或)、Eqv (相等)、Imp (隐含)
- 4) 关系运算符 = (相同)、<> (不等)、> (大于)、< (小于)、>= (不小于)、<= (不大于)、Like、Is
- 5) 位运算符 Not (逻辑非)、And (逻辑与)、Or (逻辑或)、Xor (逻辑异或)、Eqv (逻辑等)、Imp (隐含)

第三节 数据类型

VBA 共有 12 种数据类型，具体见下表，此外用户还可以根据以下类型用 Type 自定义数据类型。

数据类型	类型标识符	字节
字符串型 String	\$	字符长度 (0-65400)
字节型 Byte	无	1
布尔型 Boolean	无	2
整数型 Integer	%	2
长整数型 Long	&	4
单精度型 Single	!	4
双精度型 Double	#	8
日期型 Date	无	8 公元 100/1/1-99/12/31
货币型 Currency	@	8
小数点型 Decimal	无	14
变体型 Variant	无	以上任意类型，可变
对象型 Object	无	4

第四节 变量与常量

- 1) VBA 允许使用未定义的变量，默认是变体变量。
- 2) 在模块通用说明部份，加入 Option Explicit 语句可以强迫用户进行变量定义。
- 3) 变量定义语句及变量作用域

Dim 变量 as 类型 ' 定义为局部变量，如 Dim xyz as integer
Private 变量 as 类型 ' 定义为私有变量，如 Private xyz as byte
Public 变量 as 类型 ' 定义为公有变量，如 Public xyz as single
Global 变量 as 类型 ' 定义为全局变量，如 Global xyz as date

Static 变量 **as** 类型 ' 定义为静态变量, 如 `Static xyz as double`

一般变量作用域的原则是, 那部份定义就在那部份起作用, 模块中定义则在该模块起作用。

4) 常量为变量的一种特例, 用 **Const** 定义, 且定义时赋值, 程序中不能改变值, 作用域也如同变量作用域。如下定义: `Const Pi=3.1415926 as single`

第五节 数组

数组是包含相同数据类型的一组变量的集合, 对数组中的单个变量引用通过数组索引下标进行。在内存中表现为一个连续的内存块, 必须用 **Global** 或 **Dim** 语句来定义。定义规则如下:

`Dim 数组名([lower to]upper [, [lower to]upper, ...]) as type`; Lower 缺省值为 0。二维数组是按行列排列, 如 XYZ(行, 列)。

除了以上固定数组外, VBA 还有一种功能强大的动态数组, 定义时无大小维数声明; 在程序中再利用 **Redim** 语句来重新改变数组大小, 原来数组内容可以通过加 **preserve** 关键字来保留。如下例:

`Dim array1() as double : Redim array1(5) : array1(3)=250 : Redim preserve array1(5,10)`

第六节 注释和赋值语句

1) 注释语句是用来说明程序中某些语句的功能和作用; VBA 中有两种方法标识为注释语句。

✓ 单引号 ' ; 如: ' 定义全局变量; 可以位于别的语句之尾, 也可单独一行

✓ **Rem** ; 如: **Rem** 定义全局变量; 只能单独一行

2) 赋值语句是进行对变量或对象属性赋值的语句, 采用赋值号 =, 如 `X=123: Form1.caption=" 我的窗口"`

对对象的赋值采用: `set myobject=object` 或 `myobject:=object`

第七节 书写规范

1) VBA 不区分标识符的字母大小写, 一律认为是小写字母;

2) 一行可以书写多条语句, 各语句之间以冒号 : 分开;

3) 一条语句可以多行书写, 以空格加下划线 _ 来标识下行为续行;

4) 标识符最好能简洁明了, 不造成歧义。

第八节 判断语句

1) **If...Then...Else** 语句

`If condition Then [statements][Else elstatements]`

如 1: `If A>B And C<D Then A=B+2 Else A=C+2`

如 2: `If x>250 Then x=x-100`

或者, 可以使用块形式的语法:

`If condition Then`

`[statements]`

`[ElseIf condition-n Then`

`[elseifstatements] ...`

`[Else`

`[elstatements]]`

`End If`

如 1:

`If Number < 10 Then`

`Digits = 1`

`ElseIf Number < 100 Then`

`Digits = 2`

`Else`

`Digits = 3`

`End If`

2) **Select Case...Case...End Case** 语句

如 1:

```

Select Case Pid
    Case "A101"
        Price=200
    Case "A102"
        Price=300
    .....
    Case Else
        Price=900
End Case

```

3) **Choose** 函数

choose(index, choce-1, choice-2, ..., choice-n), 可以用来选择自变量串列中的一个值, 并将其返回, index 必要参数, 数值表达式或字段, 它的运算结果是一个数值, 且界于 1 和可选择的项目数之间。choice 必要参数, Variant 表达式, 包含可选择项目的其中之一。如:

```
GetChoice = Choose(Ind, "Speedy", "United", "Federal")
```

4) **Switch** 函数

```
Switch(expr-1, value-1[, expr-2, value-2 _ [, expr-n, value-n]])
```

switch 函数和 Choose 函数类似, 但它是以两个一组的方式返回所要的值, 在串列中, 最先为 TRUE 的值会被返回。expr 必要参数, 要加以计算的 Variant 表达式。value 必要参数。如果相关的表达式为 True, 则返回此部分的数值或表达式, 没有一个表达式为 **True**, **Switch** 会返回一个 Null 值。

第九节 循环语句

1) **For Next** 语句 以指定次数来重复执行一组语句

```
For counter = start To end [Step step] ' step 缺省值为 1
```

```
[statements]
```

```
[Exit For]
```

```
[statements]
```

```
Next [counter]
```

如 1:

```

For Words = 10 To 1 Step -1          ' 建立 10 次循环
    For Chars = 0 To 9                  ' 建立 10 次循环
        MyString = MyString & Chars    ' 将数字添加到字符串中
    Next Chars                          ' Increment counter
    MyString = MyString & " "          ' 添加一个空格
Next Words

```

2) **For Each...Next** 语句 主要功能是对一个数组或集合对象进行, 让所有元素重复执行一次语句

```
For Each element In group
```

```
Statements
```

```
[Exit for]
```

```
Statements
```

```
Next [element]
```

如 1:

```

For Each rang2 In rang1
    With rang2.interior
        .colorindex=6
        .pattern=xlSolid
    End with
Next

```

这上面一例中用到了 **With...End With** 语句, 目的是省去对象多次调用, 加快速度; 语法为:

```
With object
[statements]
End With
```

3) Do...loop 语句 在条件为 true 时，重复执行区块命令

Do {while |until} condition' while 为当型循环，until 为直到型循环，顾名思义，不多说啦

```
Statements
```

```
Exit do
```

```
Statements
```

```
Loop
```

或者使用下面语法

```
Do ' 先 do 再判断，即不论如何先干一次再说
```

```
Statements
```

```
Exit do
```

```
Statements
```

```
Loop {while |until} condition
```

第十节 其他类语句和错误语句处理

一. 其他循环语句

结构化程序使用以上判断和循环语句已经足够，建议不要轻易使用下面的语句，虽然 VBA 还支持。

1) Goto line 该语句为跳转到 line 语句行

2) On expression gosub destinationlist 或者 on expression goto destinationlist 语句为根据 expression 表达式值来跳转到所要的行号或行标记

3) Gosub line...line...Return 语句，Return 返回到 Gosub line 行，如下例：

```
Sub gosubtry()
    Dim num
    Num=inputbox("输入一个数字，此值将会被判断循环")
    If num>0 then Gosub Routine1 : Debug.print num: Exit sub
    Routine1:
    Num=num/5
    Return
```

```
End sub
```

4) while...wend 语句，只要条件为 TRUE，循环就执行，这是以前 VB 老语法保留下来的，如下例：

```
while condition 'while I<50
[statements]    'I=I+1
wend            'Wend
```

二. 错误语句处理

执行阶段有时会有错误的情况发生，利用 On Error 语句来处理错误，启动一个错误的处理程序。语法如下：

```
On Error Goto Line    '当错误发生时，会立刻转移到 line 行去
On Error Resume Next  '当错误发生时，会立刻转移到发生错误的下一行去
On Error Goto 0        '当错误发生时，会立刻停止过程中任何错误处理过程
```

第十一节 过程和函数

过程是构成程序的一个模块，往往用来完成一个相对独立的功能。过程可以使程序更清晰、更具结构性。VBA 具有四种过程：Sub 过程、Function 函数、Property 属性过程和 Event 事件过程。

一. Sub 过程

Sub 过程的参数有两种传递方式：按值传递(ByVal)和按地址传递(ByRef)。如下例：

```
Sub password (ByVal x as integer, ByRef y as integer)
    If y=100 then y=x+y else y=x-y
    x=x+100
```

```
End sub
```

```
Sub call_password ()
    Dim x1 as integer
    Dim y1 as integer
    x1=12
    y1=100
    Call password (x1,y1) ‘调用过程方式: 1. Call 过程名(参数 1, 参数 2...); 2. 过程名 参
    数 1, 参数 2...
    debug.print x1,y1 ‘结果是 12、112, y1 按地址传递改变了值, 而 x1 按值传递, 未改变原值
End sub
```

二. Function 函数

函数实际是实现一种映射, 它通过一定的映射规则, 完成运算并返回结果。参数传递也两种: 按值传递(ByVal)和按地址传递(ByRef)。如下例:

```
Function password(ByVal x as integer, byref y as integer) as boolean
    If y=100 then y=x+y else y=x-y
    x=x+100
    if y=150 then password=true else password=false
End Function
```

```
Sub call_password ()
    Dim x1 as integer
    Dim y1 as integer
    x1=12
    y1=100
    if password then ‘调用函数: 1. 作为一个表达式放在=右端 ; 2. 作为参数使用
    debug.print x1
    end if
End sub
```

三. Property 属性过程和 Event 事件过程

这是 VB 在对象功能上添加的两个过程, 与对象特征密切相关, 也是 VBA 比较重要组成, 技术比较复杂, 可以参考相关书籍。

第十二节 内部函数

在 VBA 程序语言中有许多内置函数, 可以帮助程序代码设计和减少代码的编写工作。

一. 测试函数

IsNumeric(x)	‘是否为数字, 返回 Boolean 结果, True or False
IsDate(x)	‘是否是日期, 返回 Boolean 结果, True or False
IsEmpty (x)	‘是否为 Empty, 返回 Boolean 结果, True or False
IsArray(x)	‘指出变量是否为一个数组。
IsError(expression)	‘指出表达式是否为一个错误值
IsNull(expression)	‘指出表达式是否不包含任何有效数据 (Null)。
IsObject(identifier)	‘指出标识符是否表示对象变量

二. 数学函数

Sin(X)、Cos(X)、Tan(X)、Atan(x) 三角函数, 单位为弧度
 Log(x) 返回 x 的自然对数
 Exp(x) 返回 e^x
 Abs(x) 返回绝对值
 Int(number)、Fix(number) 都返回参数的整数部分, 区别: Int 将 -8.4 转换成 -9, 而 Fix 将 -8.4 转换成 -8
 Sgn(number) 返回一个 Variant (Integer), 指出参数的正负号
 Sqr(number) 返回一个 Double, 指定参数的平方根

VarType(varname) 返回一个 Integer，指出变量的子类型

Rnd(x) 返回 0-1 之间的单精度数据，x 为随机种子

三. 字符串函数

Trim(string)	去掉 string 左右两端空白
Ltrim(string)	去掉 string 左端空白
Rtrim(string)	去掉 string 右端空白
Len(string)	计算 string 长度
Left(string, x)	取 string 左段 x 个字符组成的字符串
Right(string, x)	取 string 右段 x 个字符组成的字符串
Mid(string, start, x)	取 string 从 start 位开始的 x 个字符组成的字符串
Ucase(string)	转换为大写
Lcase(string)	转换为小写
Space(x)	返回 x 个空白的字符串
Asc(string)	返回一个 integer，代表字符串中首字母的字符代码
Chr(charcode)	返回 string，其中包含有与指定的字符代码相关的字符

四. 转换函数

CBool(expression)	转换为 Boolean 型
CByte(expression)	转换为 Byte 型
CCur(expression)	转换为 Currency 型
CDate(expression)	转换为 Date 型
Cdbl(expression)	转换为 Double 型
CDec(expression)	转换为 Decemal 型
CInt(expression)	转换为 Integer 型
CLng(expression)	转换为 Long 型
CSng(expression)	转换为 Single 型
CStr(expression)	转换为 String 型
CVar(expression)	转换为 Variant 型
Val(string)	转换为数据型
Str(number)	转换为 String

五. 时间函数

Now 返回一个 Variant (Date)，根据计算机系统设置的日期和时间来指定日期和时间。

Date 返回包含系统日期的 Variant (Date)。

Time 返回一个指明当前系统时间的 Variant (Date)。

Timer 返回一个 Single，代表从午夜开始到现在经过的秒数。

TimeSerial(hour, minute, second) 返回一个 Variant (Date)，包含具有具体时、分、秒的时间。

DateDiff(interval, date1, date2[, firstdayofweek[, firstweekofyear]]) 返回 Variant (Long) 的值，表示两个指定日期间的时间间隔数日

Second(time) 返回一个 Variant (Integer)，其值为 0 到 59 之间的整数，表示一分钟之中的某个秒

Minute(time) 返回一个 Variant (Integer)，其值为 0 到 59 之间的整数，表示一小时中的某分钟

Hour(time) 返回一个 Variant (Integer)，其值为 0 到 23 之间的整数，表示一天之中的某一钟点

Day(date) 返回一个 Variant (Integer)，其值为 1 到 31 之间的整数，表示一个月中的某一日

Month(date) 返回一个 Variant (Integer)，其值为 1 到 12 之间的整数，表示一年中的某月

Year(date) 返回 Variant (Integer)，包含表示年份的整数。

Weekday(date, [firstdayofweek]) 返回一个 Variant (Integer)，包含一个整数，代表某个日期是星期几

第十三节 文件操作

文件

Dir[(pathname[, attributes])] ; pathname 可选参数，用来指定文件名的字符串表达式，可能包含目

录或文件夹、以及驱动器。如果没有找到 pathname，则会返回零长度字符串 (""); attributes 可选参数。常数或数值表达式，其总和用来指定文件属性。如果省略，则会返回匹配 pathname 但不包含属性的文件。

删除

Kill pathname 从磁盘中删除文件，pathname 参数是用来指定一个文件名

Rmdir pathname 从磁盘中删除删除目录，pathname 参数是用来指定一个文件夹

打开

Open pathname For mode [Access access] [lock] As [#]filenumber [Len=reclength] 能够对文件输入/输出 (I/O)。

pathname 必要。字符串表达式，指定文件名，该文件名可能还包括目录、文件夹及驱动器。

mode 必要。关键字，指定文件方式，有 Append、Binary、Input、Output、或 Random 方式。如果未指定方式，则以 Random 访问方式打开文件。

access 可选。关键字，说明打开的文件可以进行的操作，有 Read、Write、或 Read Write 操作。

lock 可选。关键字，说明限定于其它进程打开的文件的操作，有 Shared、Lock Read、Lock Write、和 Lock Read Write 操作。

filenumber 必要。一个有效的文件号，范围在 1 到 511 之间。使用 **FreeFile** 函数可得到下一个可用的文件号。reclength 可选。小于或等于 32,767 (字节) 的一个数。对于用随机访问方式打开的文件，该值就是记录长度。对于顺序文件，该值就是缓冲字符数。

说明 对文件做任何 I/O 操作之前都必须先打开文件。Open 语句分配一个缓冲区供文件进行 I/O 之用，并决定缓冲区所使用的访问方式。如果 pathname 指定的文件不存在，那么，在用 Append、Binary、Output、或 Random 方式打开文件时，可以建立这一文件。如果文件已由其它进程打开，而且不允许指定的访问类型，则 Open 操作失败，而且会有错误发生。如果 mode 是 Binary 方式，则 Len 子句会被忽略掉。

重要 在 Binary、Input 和 Random 方式下可以用不同的文件号打开同一文件，而不必先将该文件关闭。在 Append 和 Output 方式下，如果要用不同的文件号打开同一文件，则必须在打开文件之前先关闭该文件。

读入

Input #filenumber, varlist 从已打开的顺序文件中读出数据并将数据指定给变量

Get [#]filenumber, [recnumber], varname 将一个已打开的磁盘文件读入一个变量之中。

写入

Write #filenumber, [outputlist] 将数据写入顺序文件

Print #filenumber, [outputlist] 将格式化显示的数据写入顺序文件中

Put [#]filenumber, [recnumber], varname 将一个变量的数据写入磁盘文件中。

关闭

Close [filenumberlist] 关闭 Open 语句所打开的输入/输出 (I/O) 文件

注意 如果今后想用 Input # 语句读出文件的数据，就要用 Write # 语句而不用 Print # 语句将数据写入文件。因为在使用 Write # 时，将数据域分界就可确保每个数据域的完整性，因此可用 Input # 再将数据读出来。使用 Write # 还能确保任何地区的数据都被正确读出。Write 与 Print # 语句不同，当要将数据写入文件时，Write # 语句会在项目和用来标记字符串的引号之间插入逗号。Write # 语句在将 outputlist 中的最后一个字符写入文件后会插入一个新行字符，即回车换行符，(Chr(13) + Chr(10))。

其他文件函数

LOF(filenumber) 返回一个 Long，表示用 Open 语句打开的文件的大小，该大小以字节为单位。

EOF(filenumber) 返回一个 Integer，它包含 Boolean 值 True，表明已经到达为 Random 或顺序 Input 打开的文件的结尾。

Loc(filenumber) 返回一个 Long，在已打开的文件中指定当前读/写位置

Seek(filenumber) 返回一个 Long，在 Open 语句打开的文件中指定当前的读/写位

二、Visual BASIC 程序设计网络教学

第一课 VBA 是什么

1.1 VBA 是什么

直到 90 年代早期,使应用程序自动化还是充满挑战性的领域.对每个需要自动化的应用程序,人们不得不学习一种不同的自动化语言.例如:可以用 EXCEL 的宏语言来使 EXCEL 自动化,使用 WORD BASIC 使 WORD 自动化,等等.微软决定让它开发出来的应用程序共享一种通用的自动化语言-----Visual Basic For Application(VBA),可以认为 VBA 是非常流行的应用程序开发语言 VASUAL BASIC 的子集.实际上 VBA 是“寄生于”VB 应用程序的版本.VBA 和 VB 的区别包括如下几个方面:

1. VB 是设计用于创建标准的应用程序,而 VBA 是使已有的应用程序(EXCEL 等)自动化
2. VB 具有自己的开发环境,而 VBA 必须寄生于已有的应用程序.
3. 要运行 VB 开发的应用程序,用户不必安装 VB,因为 VB 开发出的应用程序是可执行文件(*.EXE),而 VBA 开发的程序必须依赖于它的“父”应用程序,例如 EXCEL.

尽管存在这些不同,VBA 和 VB 在结构上仍然十分相似.事实上,如果你已经了解了 VB,会发现学习 VBA 非常快.相应的,学完 VBA 会给学习 VB 打下坚实的基础.而且,当学会在 EXCEL 中用 VBA 创建解决方案后,即已具备在 WORD ACCESS OUTLOOK FOXPRO PROWERPOINT 中用 VBA 创建解决方案的大部分知识.

- * VBA 一个关键特征是你所学的知识在微软的一些产品中可以相互转化.
- * VBA 可以称作 EXCEL 的“遥控器”.

VBA 究竟是什么?更确切地讲,它是一种自动化语言,它可以使常用的程序自动化,可以创建自定义的解决方案.

此外,如果你愿意,还可以将 EXCEL 用做开发平台实现应用程序.

1.2 EXCEL 环境中基于应用程序自动化的优点

也许你想知道 VBA 可以干什么?使用 VBA 可以实现的功能包括:

1. 使重复的任务自动化.
2. 自定义 EXCEL 工具栏,菜单和界面.
3. 简化模板的使用.
4. 自定义 EXCEL,使其成为开发平台.
5. 创建报表.
6. 对数据进行复杂的操作和分析.

用 EXCEL 作为开发平台有如下原因:

1. EXCEL 本身功能强大,包括打印,文件处理,格式化和文本编辑.
2. EXCEL 内置大量函数.
3. EXCEL 界面熟悉.
4. 可连接到多种数据库.

用其他语言开发应用程序,一半的工作是编写一些基本功能的模块,包括文件的打开和保存,打印,复制等.而用 EXCEL 作为开发平台,则由于 EXCEL 已经具备这些基本功能,你要做的只是使用它.

1.3 录制简单的宏

在介绍学习 VBA 之前，应该花几分钟录制一个宏。

新术语：“宏”，指一系列 EXCEL 能够执行的 VBA 语句。

以下将要录制的宏非常简单，只是改变单元格颜色。请完成如下步骤：

1) 打开新工作簿，确认其他工作簿已经关闭。

2) 选择 A1 单元格。调出“常用”工具栏。

3) 选择“工具”-“宏”-“录制新宏”。

4) 输入“改变颜色”作为宏名替换默认宏名，单击确定，注意，此时状态栏中显示“录制”，特别是“停止录制”工具栏也显示出来。替换默认宏名主要是便于分别这些宏。

★ 宏名最多可为 255 个字符，并且必须以字母开始。其中可用的字符包括：字母、数字和下划线。宏名中不允许出现空格。通常用下划线代表空格。

5) 选择“格式”的“单元格”，选择“图案”选项中的红色，单击“确定”。

6) 单击“停止录制”工具栏按钮，结束宏录制过程。

※ 如果“停止录制”工具栏开始并未出现，请选择“工具”-“宏”-“停止录制”。

录制完一个宏后就可以执行它了。

1.4 执行宏

当执行一个宏时，EXCEL 按照宏语句执行的情况就像 VBA 代码在对 EXCEL 进行“遥控”。但 VBA 的“遥控”不仅能使操作变得简便，还能使你获得一些使用 EXCEL 标准命令所无法实现的功能。而且，一旦熟悉了 EXCEL 的“遥控”，你都会奇怪自己在没有这些“遥控”的情况下，到底是怎么熬过来的。要执行刚才录制的宏，可以按以下步骤进行：

1) 选择任何一个单元格，比如 A3。

2) 选择“工具”-“宏”-“宏”，显示“宏”对话框。

3) 选择“改变颜色”，选择“执行”，则 A3 单元格的顏色变为红色。试着选择其它单元格和几个单元格组成的区域，然后再执行宏，以便加深印象。

1.5 查看录制的代码

到底是什么在控制 EXCEL 的运行呢？你可能有些疑惑。好，让我们看看 VBA 的语句吧。

1) 选择“工具”-“宏”-“宏”，显示“宏”对话框。

2) 单击列表中的“改变颜色”，选择“编辑”按钮。

此时，会打开 VBA 的编辑器窗口（VBE）。关于该编辑器，以后再详细说明，先将注意力集中到显示的代码上。代码如下：（日期和姓名会有不同）

```
Sub 改变颜色()  
,  
  
, 改变颜色 Macro  
, xw 记录的宏 2000-6-10  
,
```

```
,
With Selection.Interior
.ColorIndex = 3
.Pattern = xlSolid
.PatternColorIndex = xlAutomatic
End With
End Sub
```

将来会十分熟悉这种代码，虽然现在它们看上去像一种奇怪的外语。学习 VBA 或编程语言在某种程度上比较像在学习一种外语。

Sub 改变颜色():这是宏的名称。

中间的以“'”开头的五行称为“注释”，它在录制宏时自动产生。

以 With 开头到 End With 结束的结构是 With 结构语句，这段语句是宏的主要部分。注意单词“selection”，它代表“突出显示的区域”（即：选定区域）。With Selection.Interior：它读作“选择区域的内部”。这整段语句设置该区域内部的一些“属性”。

其中：

.ColorIndex = 3: 将该内部设为红色。注意：有一小圆点，它的作用在于简化语句，小圆点代替出现在 With 后的词，它是 With 结构的一部分。另外：红色被数字化为 3。（红色警戒是否可称作：3 号警戒，嗯？）有兴趣的话，你将 3 改为其他数字试试看。

.Pattern = xlSolid: 设置该区域的内部图案。由于是录制宏，所以，虽然你并未设置这一项，宏仍然将其记录下来（因为在“图案”选项中有此一项，只是你未曾设置而已）。xlSolid 表示纯色。

.PatternColorIndex = xlAutomatic: 表示内部图案底纹颜色为自动配色。

End With:结束 With 语句。

End Sub:整个宏的结束语

1.6 编辑录制的代码

在上一节，我们录制了一个宏并查看了代码，代码中有两句实际上并不起作用。哪两句？现在，在宏中作一个修改，删除多余行，直到和下面代码相同：

```
Sub 改变颜色()
,
' 改变颜色 Macro
' xw 记录的宏 2000-6-10
,
,
```

```
With Selection.Interior
.ColorIndex = 3
End With
End Sub
```

完成后，在工作表中试验一下。你会发现结果和修改前的状况一样。在 With 语句前加入一行：

```
Range("A5").Select
```

试着运行该宏，则无论开始选择哪个单元格，宏运行结果都是使 A5 单元格变红。

现在可以看到，编辑录制的宏同样非常简单。需要编辑宏是因为以下三个方面的原因。一：在录制中出错而不得不修改。二：录制的宏中有多余的语句需要删除，提高宏的运行速度。三：希望增加宏的功能。比如：加入判断或循环等无法录制的语句。

1.7 录制宏的局限性

希望自动化的许多 EXCEL 过程大多都可以用录制宏来完成，但是宏记录器存在以下局限性。通过宏记录器无法完成的工作有：

- 1) 录制的宏无判断或循环能力。
- 2) 人机交互能力差，即用户无法进行输入，计算机无法给出提示。
- 3) 无法显示 EXCEL 对话框。
- 4) 无法显示自定义窗体。

1.8 小结

本课中，你已经掌握了 VBA 的一些基础知识，你会录制宏、编辑宏而且了解了录制宏的局限性。你很努力，并且已经为将来学习 VBA 甚至 VB 等编程语言打下了基础。关键是你已经了解了一个谜底，就是说，你了解了什么是编程。下面是些小练习，做完后才可以去玩哟。

思考：

- 1) VBA 只能用于 EXCEL 吗？
- 2) VBA 是基于哪种语言？
- 3) 说说 EXCEL 和 VBA 的关系。
- 4) 为什么要用宏？

第二课 处理录制的宏

2.1 为宏指定快捷键

你也许希望为经常使用的宏指定快捷键。快捷键是指键的组合，当其按下时执行一条命令。例如：CTRL+C

在许多程序中代表“复制”命令。当给宏指定了快捷键后，就可以用快捷键来执行宏，而不必通过“工具”菜单。

注意：当包含宏的工作簿打开时间，为宏指定快捷键会覆盖 EXCEL 默认的快捷键。例如：

把 CTRL+C 指定给某个宏，那么 CTRL+C 就不再执行复制命令。用以下方法可以打印出 EXCEL 的快捷键清单（用 A4 纸打印共有 24 页之多）：

- 1) 打开 EXCEL 帮助文件并选择“目录”选项。
- 2) 从“使用快捷键”文件夹中选择““快捷键”标题。
- 3) 右击该标题，从快捷菜单中选择“打印”。
- 4) 选择“打印所选标题和所有子主题”，单击“确定”。

可以在创建宏时指定快捷键，也可以在创建后再指定。要在创建（录制）宏时指定快捷键，只须在录制宏时在输入宏名后，在“快捷键”文本框中输入相应的键。录制宏后指定快捷键也很简单，只需选择“工具”“宏”，显示“宏”对话框，选择要指定快捷键的宏，再单击“选项”按钮，通过“选项”对话框进行设置。

2.2 决定宏保存的位置

宏可保存在三种可能的位置：

- 1) 当前工作簿。（只有该工作簿打开时，该宏才可用。）
- 2) 新工作簿。
- 3) 个人宏工作簿。

2.3 个人宏工作簿

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。第一次将宏创建到个人宏工作簿时，会创建名为“PERSONAL.XLS”的新文件。如果该文件存在，则每当 EXCEL 启动时会自动将此文件打开并隐藏在活动工作簿后面（在“窗口”菜单中选择“取消隐藏”后，可以很方便地发现它的存在。）如果你要让某个宏在多个工作簿都能使用，那么就应当创建个人宏工作簿，并将宏保存于其中。个人宏工作簿保存在“XLSTART”文件夹中。具体路径为：C:\WINDOWS\Profiles\Application Data\Microsoft\Excel\XLSTART。可以以单词“XLSTART”查询。

注意：如果存在个人宏工作簿，则每当 EXCEL 启动时会自动将此文件打开并隐藏。因为它存放在 XLSTART 文件夹内。

2.3.1 保存宏到个人宏工作簿

本练习，将保存一个简单的宏到个人宏工作簿，该宏为文本加下划线并改为斜体，步骤如下：

- 1) 建立一个名为“HOUR2”的工作簿，选择“工具”-“宏”-“录制新宏”，显示“录制新宏”对话框。
- 2) 输入“格式化文本”作为宏名。
- 3) 从“保存在”下拉框中选择“个人宏工作簿”。
- 4) 单击“确定”按钮。现在进入录制模式。
- 5) 单击“斜体”工具栏按钮。一段时间内，鼠标出现沙漏，特别是在第一次创建个人宏工作簿时，因为 EXCEL 在创建该工作簿。
- 6) 单击“下划线”按钮。
- 7) 停止录制。

2.3.2 使用并编辑个人宏工作簿中的宏

刚才已经保存了一个宏到个人宏工作簿，现在可以在任何工作簿中使用该宏。可按如下步骤操作：

- 1) 关闭所有 EXCEL 工作簿。

- 2) 任意打开一个 EXCEL 文件. (EXCEL 自动将个人宏工作簿同时打开并隐藏.)
 - 3) 在 A3 中输入你的名字.
 - 4) 选择“工具”-“宏”, 显示宏对话框. 现在可以在宏列表中看到“格式化文本”这个宏.
 - 5) 选择“格式化文本”宏, 并执行. 现在 A3 单元格中, 你的名字变为斜体字还带有下列线.
- 选择“窗口”-“取消隐藏”, 可以将 PERSONAL.XLS 显示出来, 其中没有任何文字, 但通过 VBA 编辑器可以在其中的模块中找到“格式化文本”这个宏. 在 VBA 编辑器中可以直接编辑或者删除. 如果 PERSONAL.XLS 中一个宏都没有, 在启动 EXCEL 时仍会打开 PERSONAL.XLS, 这也许是 EXCEL 存在的一个小毛病.

2.4 将宏指定给按钮

即使通过快捷键可以是宏的执行变快, 但是一旦宏的数量多了也难于记忆, 而且, 如果宏是由其他人来使用, 难道你要他们也记住那么多的快捷键吗?

作为 EXCEL 开发者, 一个主要的目标是为自动化提供一个易于操作的界面. “按钮”是最常见的界面组成元素之一. 通过使用“窗体”工具栏, 可以为工作簿中的工作表添加按钮. 在创建完一个按钮后, 可以为它指定宏, 然后你的用户就可以通过单击按钮来执行宏. 在本练习中, 将创建一个按钮, 并为它指定一个宏, 然后用该按钮来执行宏. 具体步骤如下:

- 1) 打开“HOUR2”工作簿.
- 2) 调出“窗体”工具栏.
- 3) 单击“窗体”工具栏中的“按钮”控件, 此时鼠标变成十字形状.
- 4) 在希望放置按钮的位置按下鼠标左键, 拖动鼠标画出一个矩形, 这个矩形代表了该按钮的大小. 对大小满意后放开鼠标左键, 这样一个命令按钮就添加到了工作表中, 同时 EXCEL 自动显示“指定宏”对话框.
- 5) 从“指定宏”对话框中选择“格式化文本”, 单击“确定”. 这样, 就把该宏指定给命令按钮.
- 6) 在按钮的标题“按钮 1”前单击鼠标左键, 按下 DELETE 直到删除所有文本, 输入“格式化”作为标题.
- 7) 单击按钮外的任意位置, 现在该按钮的标题由默认的“按钮 1”变为“格式化”而且被指定了一个宏.
- 8) 试着在某个单元格中输入文本, 单击按钮运行该宏.

当鼠标移动至该按钮时自动变成手的形状, 如果要改变其大小或标题, 只需用右键单击该按钮就可以进行修改和设置. 很明显, 你再也不需记住宏的名字或快捷键了, 只需按一下按钮.

2.5 将宏指定给图片或其他对象

要执行宏有多种方法可以选择, 可以将宏指定给按钮等控件, 还可以指定给图片、自定义工具栏、窗体甚至可以将宏指定给某个“事件”, 比如单击工作表, 双击工作表, 激活工作表, 打开工作簿等等, “事件”是一个重要的概念, 除此而外“方法”“对象”都是将来你会经常接触到的. 现在它们看来十分抽象, 但是将来你会很熟悉这些词语. 指定宏到图片十分简单, 只需单击某个图片, 单击快捷菜单中的“指定宏”进行设置即可.

如果不希望在工作表上添加控件或图片执行宏, 还有一种方法可以选择: 将宏指定给“工具栏按钮”, 可按如下步骤进行:

- 1) 打开“HOUR2”工作簿, 选择“工具”-“定义”, 显示“自定义工具栏”对话框.
- 2) 从“类别”列表框中选择“宏”, 从“命令”列表框中选择“自定义按钮”.
- 3) 将“自定义按钮”拖动到工具栏.
- 4) 右键单击该按钮, 选择“指定宏”, 显示“指定宏”对话框.

- 5) 选择“格式化文本”并确定。
- 6) 单击“关闭”按钮，关闭“自定义工具栏”对话框。
- 7) 试着在某个单元格中输入文本，单击工具栏按钮运行该宏。

2.6 小结

小结与思考：宏存放于三个可能的位置。个人宏工作簿存放的位置和特性。执行宏的方式。指定宏是为某个对象的事件指定一个程序，一旦这个对象以该事件激活，系统将运行指定的程序。

常用的对象有：workbook, worksheet, range, cells, 图表, 图片, 数据透视表, 控件, 窗体, 工具栏。每一个对象都有其可以响应的特殊事件（也有一些通用事件如单击或双击等）。如有兴趣，可以通过 EXCEL 帮助文件查询这几个词条。在 EXCEL 中看到的几乎都是属于某个对象，而在 EXCEL 中所做的许多工作，如移动一下鼠标等等，都可能触发了一个事件。下一学时我们将共同学习“控件”。

第三课 学习控件

3.1 EXCEL 开发过程简介

需要对以下问题有个大致的概念。

- 1) 谁使用——这决定了程序的操作难度及界面感观。
- 2) 数据来源和保存在哪里——这决定了程序的结构。
- 3) 如何操作——这将决定程序的界面和细节。
- 4) 数据处理的结果——最终决定程序的价值。

3.2 认识不同的控件

开始时请关闭所有工作簿，打开一个新工作簿并另存为“HOUR3”。在工具栏上单击鼠标右键，从快捷菜单中选择“窗体”，显示“窗体”工具栏。其中有 16 个控件，只有 9 个可放到工作表内。

- 1) 标签：它用于表现静态文本。
- 2) 分组框：它用于将其他控件进行组合。
- 3) 按钮：用于执行宏命令。
- 4) 复选框：它是一个选择控件，通过单击可以选择和取消选择，可以多项选择。
- 5) 选项按钮：通常几个选项按钮组合在一起使用，在一组中只能选择一个选项按钮。
- 6) 列表框：用于显示多个选项并从中选择。只能单选。
- 7) 组合框：用于显示多个选项并从中选择。可以选择其中的项目或者输入一个其它值。
- 8) 滚动条：不是你常见的来给很长的窗体添加滚动能力的控件，而是一种选择机制。例如调节过渡色的滚动条控件。包括水平滚动条和垂直滚动条。
- 9) 微调控件：也是一种数值选择机制，通过单击控件的箭头来选择数值。例如改变 Windows 日期或时间就会使用到微调控件。

3.3 向工作表添加控件

用 EXCEL 设计界面十分简单，要将控件添加到工作表上，可以按以下步骤操作：

- 1) 创建新工作簿并另存为“HOUR3”，显示“窗体”工具栏。
- 2) 选择“标签”控件。
- 3) 将鼠标定位到 E1，此时鼠标变成小十字。
- 4) 按下左键，拖动大约四个单元格长度，放开鼠标左键。如果希望控件大小易于控制，可

在创建该控件时按下 ALT 拖动。

5) 在标签 1 上单击右键, 选择“编辑文字”, 现在可以输入文字. 完成后, 单击任何单元格退出文字编辑。

6) 通过以上步骤可以添加其它控件到工作表中, 不再赘述。

3.4 设置控件的特性

设置控件的特性, 可以按以下步骤操作:

- 1) 选中先前创建的复选框控件, 如果没有马上创建一个。
- 2) 右击该控件, 选择“控制”选项卡。
- 3) 在“单元格链接”中输入 A1 并确定。
- 4) 单击任意单元格, 退出设置。
- 5) 用鼠标左键单击复选框, A1 出现 TRUE, 这意味着该控件被选中. 再次单击该控件, A1 出现 FALSE。
- 6) 选择刚才创建的滚动条控件. 并调出“设置控件格式”对话框。
- 7) 在“单元格链接”中输入 A3 并确定。
- 8) 在滚动条外任意单元格单击鼠标左键, 使滚动条不被选择。
- 9) 用鼠标单击滚动条上的箭头, 则 A1 的数值增加 1, 继续单击则 A1 的数值继续增加。
- 10) 保存并关闭该工作簿。

3.5 给控件命名

当创建一个控件时 EXCEL 会自动给它指定一个名字, 但不便于理解和记忆, 为控件取名的方法基本和给单元格或区域取名的方法相同. 选中某个控件, 再在位于公式栏上的“名字”编辑框输入控件名字. 这样就给控件更改了名字。

3.6 使用用户窗体

如果希望创建专业级的应用程序, 并且方便用户输入数据, 那么应该使用用户窗体. 用户窗体可以作为程序的对话框和窗口. 向用户窗体添加控件基本类似于向工作表添加控件, 然而第一步要创建一个用户窗体. 这可以通过 VBA 编辑器实现. 具体按以下步骤操作:

- 1) 打开“HOUR3”工作簿, 选择“工具”-“宏”-“VBA 编辑器”, 打开 VBA 编辑器。
- 2) 在 VBA 编辑器中选择工具栏上的“插入用户窗体”按钮或者选择“插入”菜单, 从下拉菜单中选择“用户窗体”。

现在, VBA 编辑器中出现一个名为“USERFORM1”的窗体, “控件工具箱”同时出现, 在其中有许多已经熟悉的控件, 另外还有一些新的控件。

这些新的控件是:

A) 切换按钮: 该控件如果被选中, 那么会保持被按下的状态. 如果再次单击它就恢复为没有按下的状态. EXCEL 工具栏中有几个这样的按钮, 例如: “全屏显示”, “加粗”, “下划线”以及“窗体”工具栏中的“切换网格”等。

B) 选项卡条 (TabStrip): 它是包含多个选项卡的控件. 通常用来对相关的信息进行组织或分类. 例如: 你也许希望用选项卡条来显示各个地区的销售信息, 可以给每个地区设置一个选项卡. 在默认时, 选项卡包含两页, 分别叫做 TAB1 和 TAB2, 可以添加更多的选项卡。

C) 多页: 外观类似选项卡条, 是包含一页或多页的控件. 选项卡条给人相似的外观, 而多页控件的各页包含各自不同的控件, 有各自不同的布局. 多页的例子很多, 例如: “设置控件格式”对话框和“工具”菜单中的“选项”对话框. 以及“格式”菜单中的“单元格...”对话框。

D) 图像控件: 它允许向窗体上放置图片. 图片格式须

为 *.bmp, *.cur, *.gif, *.ico, *.jpg, *.wmf.

F)RefEdit:这是工具箱中默认情况下的最后一个控件。它外观象文本框,通过这个控件可以将用户窗体折叠起来,以便选择单元格区域。还记得在使用 fx“粘贴函数”时的情况吗?

通过实践,我们会逐渐掌握每个控件的特性,这的确需要花时间,但不必死记硬背。

在对用户窗体设计得满意时,可以对其进行预览,方法是在 VBA 编辑器中选择该窗体,单击“运行”菜单中的三角符号“运行子过程/用户窗体”,三角符号在 VBA 工具栏上也可能看得到,旁边是一个垂直的等于符号,最右边是个小正方形符号,它们类似于录音机上的按钮。运行窗体的另一个方法是按 F5 键。

小结:学习完本学时后,我们具备了用于程序界面设计的基本知识。我们对控件不在感到陌生,也明白如何向工作表和窗体添加控件,但控件的内容很多,需要边用边理解。此后,我们将从界面转移到学习编写代码,并最终将二者融合。让我们准备好学习编程吧!

3.7 疑难解答

问题 1. 怎样决定控件的位置?如何选择添加到工作表还是添加到用户窗体?

解答:这完全取决于个人的爱好和应用程序的用户。如果用户对 EXCEL 非常熟悉,那么他们也许更希望以工作表的方式操作。在这种情况下不妨直接在工作表上创建控件;如果你的用户对 EXCEL 不熟悉或者你需要给用户一个专业的界面感觉,那么应该使用用户窗体。

问题 2. 什么情况下该用选项卡条而不是多页控件?

解答:如果每一页具有相同布局,则应选择选项卡条,否则应该选择多页。

本节作业

1. 思考:

- 1) 列举两种可以让用户进行多选一的控件。
- 2) 如何将控件与单元格链接起来。

2. 判断:

- 1) 只有在 VBA 编辑器中才能添加用户窗体。
- 2) 在 VBA 编辑器中看到的窗体网格线在运行时会自动显示。

3. 填空:()是显示静态文本的控件。

第四课 理解变量和变量的作用

4.1 代码存在的位置: 模块

VBA 代码必须存放在某个位置,这个地方就是模块。有两种基本类型的模块:标准模块和类模块。模块中的每个过程或者是函数过程,或者是子程序概念。本课的最后部分将讨论函数过程和子程序的区别。

新术语:

模块:它是作为一个单元保存在一起的 VBA 定义和过程的集合。

类模块:VBA 允许你创建自己的对象,对象的定义包含在类模块中。

你的大部分工作集中在标准模块中(简称为模块)当录制宏时如果不存在模块,EXCEL 自动创建一个。EXCEL 和 VBA 不关心代码存放在哪一个模块中,只要代码存在于打开的工作簿中即可。

4.2 对模块的概览

过程被定义为 VBA 代码的一个单元，过程中包括一系列用于执行某个任务或是进行某种计算的语句。工作簿的每个过程都有唯一的名字加以区分。

有两种不同的过程：子程序和函数过程。子程序只执行一个或多个操作，而不返回数值。当录制完宏查看代码时，所看到的就是子程序。宏只能录制子程序，而不能录制函数过程。一个子程序的例子如清单 4-1 所示。

程序清单 4-1 子程序的例子

```
Sub cmdSmallFont_Click()
With Selection.Font
.Name="Arial"
.FontStyle="Regular"
.Size=16
End With
End sub
```

上面列出的过程实际上是一个事件过程。通过它的名字，就可以知道这是一个事件过程。这个过程的名字是由一个对象的名字 CmdSmallFont 和一个事件的名字 Click 组成的，两者之间用下划线分开。如果还不明白，可以告诉你，CmdSmallFont 是一个命令按钮的名字。也就是说，当单击这个命令按钮时，就会运行这个事件过程。

函数过程通常情况下称为函数，要返回一个数值。这个数值通常是计算的结果或是测试的结果，例如 False 或 True。正如前面所说，可以用 VBA 创建自定义函数。实际上可以在工作表上使用你创建的函数。程序清单 4-2 是一个计算价格的 10%为运费的简单例子。

程序清单 4-2 简单的用户定义函数示例。

```
Public Function Shipping(Price)
Shipping = Price * 0.1
End Function
```

请注意，这个函数使用一个参数 (Price)。子程序和函数都可以使用参数。不论 Price 的值是多少，它都将决定运费额。Price 可以是数字和单元格引用。函数返回计算出来的运费，这个函数可以用在单元格中。

A B

1 Price 100

2 Shipping =shipping(B1)

4.2.1 创建过程

创建第一个过程需要两个基本步骤。首先，需要向工作簿中添加一个模块。接着需要向模块中添加一个工程。对于创建的每一个应用程序，只需添加一次模块。可以使用多个模块，但这是不必要的。某些开发者喜欢使用多个模块，以便根据他们的目的或者窗体对过程进行组织。在本练习中，创建的过程只显示一个消息框。

在本练习中创建的过程只显示一个消息框。在本练习中使用 MsgBox 是为了提供一个可见的例子，虽然我们还没有介绍过 MsgBox 语句，但是在本例中将使用它。要创建该过程，请按如下步骤进行：

1) 打开一个新工作簿。

2) 选择“工具”-“宏”-“Visual Basic 编辑器”，打开 VBA 编辑器窗口。

3) 在 VBA 编辑器的左面，可以看到“工程资源管理器”窗口。在工程资源管理器窗口的“Thisworkbook”上单击鼠标右键，选择“插入”-“模块”，这样就将一个模块添加到应用程序中了。(如果你没有看见“工程资源管理器”窗口，可以按 Ctrl+R)

4) 选择“插入”“过程”，显示“添加过程”对话框。

5) 输入“第一个工程”作为过程名字。在“类型”分组框中，确认选择了“子程序”。单击“确定”按钮。这样一个新的过程就添加到模块中了。可以在模块中看到以 Public Sub 第一个过程（）开始，以 End Sub 结束的语句结构。

6) 在过程中插入光标，输入以下语句并回车：

```
Msgbox "这是我的第一个过程"
```

在输入 Msgbox 后，会自动弹出一个消息框告诉你有关这条命令的信息，称之为自动列表技术。输入完成的过程如下所示：

```
Public Sub 第一个过程（）
Msgbox "这是我的第一个过程"
End Sub
```

VBA 对子程序和函数有如下的命名规则：

- * 名字中可以包含字母数字和下划线。
- * 名字中不能包含空格句号惊叹号，也不能包含字符@ & \$ #.
- * 名字最多可以包含 255 个字符。

4.2.2 运行宏

创建这个过程后，可以运行一下。运行一个过程有几种方法：可以直接使用“运行”菜单，“运行子程序/用户窗体”工具栏按钮或按下 F5 键。要运行一个过程，可以按照如下步骤：

- 1) 单击“运行子程序/用户窗体”工具栏按钮，过程执行并显示一个消息框。
- 2) 单击消息框之中的“确定”按钮，关闭该消息框。

4.3 保存对模块所做的改变

要保存新过程，需要保存过程所驻留的工作簿。可以用 VBA 编辑器保存工作簿。具体步骤如下：

1) 选择“文件”-“保存工作簿”。因为本工作簿还没有保存过，所以要给它命名。

2) 输入“HOUR4”作为文件名并按回车键，则工作簿和模块与过程都保存下来了。第四课理解变量和变量的作用

Excel Home

4.4 变量

变量是用于临时保存数值的地方。每次应用程序运行时，变量可能包含不同的数值，而在程序运行时，变量的数值可以改变。

为了说明为什么需要变量，可以按照如下步骤创建一个简单的过程：

1) 创建一个名为“你叫什么名字”的过程。

2) 在过程中输入如下代码：

```
Inputbox "输入你的名字:"
```

现在不要担心 inputbox 语句的语法，将在第六学时中了解到有关这条命令的更多信息。

3) 按下 F5 键运行过程，这时会显示一个输入框，要求输入你的名字。

4) 输入你的名字并按“确定”按钮，则结束该过程。

你输入的名字到哪里去了？如何找到用户在输入框中输入的信息？在这种情况下，需要使用变量来存储用户输入的结果。

4.4.1 变量的数据类型

使用变量的第一步是了解变量的数据类型。变量的数据类型控制变量允许保存何种类型

的数据. 表 4-1 列出了 VBA 支持的数据类型, 还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值范围.

数据类型

存储空间

数值范围

Byte

1 字节

0 - 255

Booleam

2 字节

True 或者 False

Integer

2 字节

-32768 - 32767

Long(长整型)

4 字节

-2147483648 - 2147483647

Single

4 字节

负值范围:

-3.402823E38 - -1.401298E-45

正值范围:

1.401298E-45 - 3.402823E38

Double

8 字节

负值范围:-1.79769313486232E308 - -494065645841247E-324

正值范围:4.94065645841247E-324 - 1.79769313486232E308

Currency

8 字节

-922337203685477 - 922337203685477

Decimal

14 字节

不包括小数时:

+/-79228162514264337593543950335

包括小数时:

+/-7.9228162514264337593543950335

Date

8 字节

1000 年 1 月 1 日 - 9999 年 12 月 31 日

Object

4 字节

任何引用对象

String(长字符串)

10 字节+1 字节/字符

0 - 约 20 亿

String(固定长度)

字符串的长度

1 - 约 65400

Variant (数字)

16 字节

Double 范围内的任何数值

Variant (文本)

22 字节+1 字节/字符

数据范围和变长字符串相同

表 4-1 VBA 数据类型

作为 ABV 程序员, 一个目标是选择需要存储空间尽量小的数据类型来保存所需要的数据, 这正是表 4-1 提供各种数据类型存储空间的原因。例如, 要保存诸如班级学生总数这样的小数字, 那么 Byte 数据类型就足够了。在这种情况下, 使用 Single 数据类型只是对计算机存储空间的浪费。

4.4.2 用 Dim 语句创建变量(声明变量)

现在, 你对变量可以使用的数据类型已经比较熟悉了, 以下我们将创建变量. 创建变量可以使用 Dim 语句, 创建变量通常成为“声明变量” Dim 语句的基本语法如下:

Dim 变量名 AS 数据类型

这条语法中的变量名代表将要创建的变量名. 对变量的命名规则和对过程的命名规则相同. 这条语句中的数据类型部分可以是表 4-1 中的任何一种数据类型.

变量名必须以字母开始, 并且只能包含字母数字和特定的特殊字符, 不能包含空格句号惊叹号, 也不能包含字符@ & \$ #. 名字最大长度为 255 个字符

在接下来的练习中将说明如何在 VBA 中使用变量, 你将要输入你的名字, 并用一个消息框将其显示出来. 具体步骤如下:

1) 创建一个名为“显示你的名字”的子程序.

2) 输入以下代码:

```
Public Sub 显示你的名字()
```

```
Dim s 名字 As String
```

```
s 名字 = Inputbox("请输入你的名字:")
```

```
Msgbox "你好"& s 名字
```

```
End Sub
```

3) 将鼠标放到过程中的任何地方, 按下 F5 键运行过程, 会显示一个输入框.

4) 输入你自己的名字并按回车键, 会显示一个消息框, 显示的文字中包含你自己的名字.

5) 单击“确定”按钮, 返回过程中.

在 Dim 语句中不必提供数据类型. 如果没有数据类型, 变量将被定义为 Variant 类型, 因为 VBA 中默认的数据类型是 Variant. 你知道这一点后, 最初的反应也许是觉得应该不用自己决定数据类型, 而将一切抛给 VBA. 这种观念是完全错误的. 你必须决定选择使用何种数据类型. 因为 Variant 数据类型占用存储空间较大 (16 或 22 字节) 而且它将影响程序的性能. VBA 必须辨别 Variant 类型的变量中存储了何种类型的数据。

4.4.3 变量命名的惯例

下表给出了推荐的变量命名惯例

数据类型

短前缀

长前缀

Array

a

ary

```

Boolean
f
bin
Byte
b
bit
Currency
c
cur
Double
d
dbl
Date/Time
dt
dtm/dat
Integer
i
int
Long
l
lng
Object
o
obj
Single

sng
String
s
str
Variant
v
var

```

表 4-2 变量命名的前缀

4.4.4 使用数组

如果你使用过其他编程语言,可能对数组已经比较熟悉了.数组是具有相同数据类型并共同享有一个名字的一组变量的集合.数组中的元素通过索引数字加以区分,定义数组的方法如下:

Dim array_name(n) As type (其中 n 是数组元素的个数)

例如,如果要创建保存 10 个学生名字的数组,可以用以下语句:

```
Dim s 学生名字(9) As Integer
```

注意,括号中的数字是 9 而不是 10.这是因为在默认的情况下,第一个索引数字是 0.数组在处理相似信息时非常有用.假设要处理 15 门考试成绩,可以创建 15 个独立的变量,这意味着要使用 15 个 Dim 语句.也可以创建一个数组来保存考试成绩,具体如下:

```
Dim s 考试成绩(14) As Integer
```

声明数组时的另一种方法是不给定大小.可以在程序运行时定义其大小.通过创建动态数组就可以做到.例如,你的程序要创建一表格,可以提示用户输入表格的行和列的数目。

声明动态数组的语法如下：

```
Dim dyn_array() As type
```

对数组声明后可以在程序运行时用：ReDim 语句指定数组的大小：

```
ReDim dyn_array() (array_size)
```

参数 array_size 代表数组的新大小。如果要保留数组的数值，请在 ReDim 语句后使用保留字 Preserve，具体语法如下：

```
ReDim Preserve dyn_array(array_size)
```

4.4.5 变量赋值

声明变量后就可以给变量赋值。请注意下列语句中为数组变量赋值时索引数字的使用。
程序清单 4-4

```
Dim i 人数 As Integer
```

```
Dim i 考试成绩 As Integer
```

```
Dim i As Integer
```

```
i 人数 = inputbox("输入学生的人数：")
```

```
ReDim Preserve i 考试成绩(i 数量)
```

```
For i = 1 to i 人数
```

```
i 考试成绩(i) = inputbox("输入考试成绩"& i )
```

```
Next
```

第五课 利用 VBA 设置工作表使用权限

Excel Home

一般保护工作表采取的方法是用 EXCEL 菜单中的“保护”命令，有时这尚嫌不足，比如一些机密文件根本要让某些使用者无法看到，但又需要他来操作工作簿中的其他表，怎么办？

可以打开 VBA 编辑器，打开“工程资源管理器”，双击该工作表，现在出现的是设置该表的属性的编辑窗口，单击窗口左上的下拉列表框，选择 worksheet，这时再从该窗口右上方的列表框中选择 Active(“激活”)，这时自动显示如下的语句块：

```
Private Sub Worksheet_Activate()
```

```
End Sub
```

在其中加入代码：（假设用“123”作为密码，Sheet“机密文档”为限制权限文档，sheet“普通文档”为工作簿中你认为任何适合的工作表）

```
If Application.InputBox("请输入操作权限密码:") = 123 Then
```

```
Range("A1").Select
```

```
Else
```

```
Msgbox "密码错误, 即将退出!"
```

```
Sheets("普通文档").Select
```

```
End if
```

程序如下：

```
Private Sub Worksheet_Activate()
```

```
If Application.InputBox("请输入操作权限密码:") = 123 Then
```

```
Range("A1").Select
Else
MsgBox "密码错误, 即将退出!"
Sheets("普通文档").Select
End If
End Sub
```

这样做仍有一个问题, 就是越权使用者仍会看到一些文件的片段, 即在提示密码的那段时间。好, 你可以这样做, 用上述方法选择工作表的 Deactivate 事件, 输入以下代码:

```
Sheets("机密文档").Cells.Font.ColorIndex = 2
```

这段程序使得此工作表在不被激活时, 所有文字为白色。然后, 在第一个程序中的 Range("A1").Select 后插入一行, 写入以下代码:

```
ActiveSheet.Cells.Font.ColorIndex = 56
```

这段程序, 在你输入正确密码后, 将该表所有文字转变为深灰色。

完整的程序如下:

```
Private Sub Worksheet_Activate()
If Application.InputBox("请输入操作权限密码:") = 123 Then
Range("A1").Select
Sheets("机密文档").Cells.Font.ColorIndex = 56
Else
MsgBox "密码错误, 即将退出!"
Sheets("普通文档").Select
End If
End Sub
```

Excel Home

由于 Microsoft Office 办公套件的广泛应用, 以及该软件版本的不断提升, 功能不断完善, 在 Office 办公套件平台上开发出的 VBA 应用程序越来越多, 而 VBA 是一种宏语言, 在运行速度上有很大的限制。因此 VBA 编程的方法直接关系到 VBA 程序运行的效率, 本文列举了一些提高 VBA 程序运行效率的方法。

方法 1: 尽量使用 VBA 原有的属性、方法和 Worksheet 函数

由于 Excel 对象多达百多个, 对象的属性、方法、事件多不胜数, 对于初学者来说可能对它们不全部了解, 这就产生了编程者经常编写与 Excel 对象的属性、方法相同功能的 VBA 代码段, 而这些代码段的运行效率显然与 Excel 对象的属性、方法完成任务的速度相差甚大。例如用 Range 的属性 CurrentRegion 来返回 Range 对象, 该对象代表当前区。(当前区指以任意空白行及空白列的组合为边界的区域)。同样功能的 VBA 代码需数十行。因此编程前应尽可能多地了解 Excel 对象的属性、方法。

充分利用 Worksheet 函数是提高程序运行速度的极度有效的方法。如求平均工资的例子:

```
For Each c In
Worksheet(1).Range(" A1:A1000" )
TotalValue = TotalValue + c.Value
Next
AverageValue = TotalValue / Worksheet(1).Range(" A1:A1000" ).Rows.Count
```

而下面代码程序比上面例子快得多:


```
AverageValue=Application.WorksheetFunction.Average(Worksheets(1).Range("A1:A1000"))
```

其它函数如 Count, Counta, Countif, Match, Lookup 等等, 都能代替相同功能的 VBA 程序代码, 提高程序的运行速度。

方法 2: 尽量减少使用对象引用, 尤其在循环中

每一个 Excel 对象的属性、方法的调用都需要通过 OLE 接口的一个或多个调用, 这些 OLE 调用都是需要时间的, 减少使用对象引用能加快 VBA 代码的运行。例如

1. 使用 With 语句。

```
Worksbooks(1).Sheets(1).Range(" A1:A1000" ).Font.Name=" Pay"
Worksbooks(1).Sheets(1).Range(" A1:A1000" ).Font.FontStyle=" Bold" ...
```

则以下语句比上面的快

```
With Worksheets(1).Sheets(1).Range(" A1:A1000" ).Font
    .Name = " Pay"
    .FontStyle = " Bold"
...
End With
```

2. 使用对象变量。

如果你发现一个对象引用被多次使用, 则你可以将此对象用 Set 设置为对象变量, 以减少对对象的访问。如:

```
Worksbooks(1).Sheets(1).Range(" A1" ).Value = 100
Worksbooks(1).Sheets(1).Range(" A2" ).Value = 200
```

则以下代码比上面的要快:

```
Set MySheet = Worksheets(1).Sheets(1)
MySheet.Range(" A1" ).Value = 100
MySheet.Range(" A2" ).Value = 200
```

3. 在循环中要尽量减少对象的访问。

```
For k = 1 To 1000
    Sheets(" Sheet1" ).Select
    Cells(k, 1).Value = Cells(1, 1).Value
Next k
```

则以下代码比上面的要快:

```
Set TheValue = Cells(1, 1).Value
Sheets(" Sheet1" ).Select
For k = 1 To 1000
    Cells(k, 1).Value = TheValue
Next k
```

方法 3: 减少对象的激活和选择

如果你的通过录制宏来学习 VBA 的, 则你的 VBA 程序里一定充满了对象的激活和选择, 例如 Worksheets(XXX).Activate、Sheets(XXX).Select、Range(XXX).Select 等, 但事实上大多数情况下这些操作不是必需的。例如

```
Sheets(" Sheet3" ).Select
Range(" A1" ).Value = 100
Range(" A2" ).Value = 200
可改为:
```

```
With Sheets(" Sheet3" )
    .Range(" A1" ).Value = 100
    .Range(" A2" ).Value = 200
End With
```

方法 4: 关闭屏幕更新

如果你的 VBA 程序前面三条做得比较差, 则关闭屏幕更新是提高 VBA 程序运行速度的最有效的方法, 缩短运行时间 2/3 左右。关闭屏幕更新的方法:

```
Application.ScreenUpdate = False
```

请不要忘记 VBA 程序运行结束时再将该值设回来:

```
Application.ScreenUpdate = True
```

以上是提高 VBA 运行效率的几种方法。

第七课 如何在 Excel 里使用定时器

Excel Home

用过 Excel 97 里的加载宏“定时保存”吗? 可惜它的源程序是加密的, 现在就上传一篇介绍实现它的文档。

在 Office 里有个方法是 application.ontime , 具体函数如下:

```
expression.OnTime(EarliestTime, Procedure, LatestTime, Schedule)
```

如果想进一步了解, 请参阅 Excel 的帮助。

这个函数是用来安排一个过程在将来的特定时间运行, (可为某个日期的指定时间, 也可指定的时间段之后)。通过这个函数我们就可以在 Excel 里编写自己的定时程序了。下面就举两个例子来说明它。

1. 在下午 17:00:00 的时候显示一个对话框。

```
Sub Run_it()
```

```
Application.OnTime TimeValue("17:00:00"), "Show_my_msg"
```

' 设置定时器在 17:00:00 激活, 激活后运行 Show_my_msg 。

```
End Sub
```

```
Sub Show_my_msg()
```

```
msg = MsgBox("现在是 17:00:00 ! ", vbInformation, "自定义信息")
```

```
End Sub
```

2. 模仿 Excel 97 里的“自动保存宏”, 在这里定时 5 秒出现一次

```
Sub auto_open()
```

```
MsgBox "欢迎你, 在这篇文档里, 每 5 秒出现一次保存的提示!", vbInformation, "请注意!"
```

```
Call runtimer ' 打开文档时自动运行
```

```
End Sub
```

```
Sub runtimer()
```

```
Application.OnTime Now + TimeValue("00:00:05"), "saveit"
```

' Now + TimeValue("00:15:00") 指定在当前时间过 5 秒钟开始运行 Saveit 这个过程。

```
End Sub
```

```
Sub SaveIt()
```

```
msg = MsgBox("朋友, 你已经工作很久了, 现在就存盘吗?" & Chr(13) _
```

```

& "选择是：立刻存盘" & Chr(13) _
& "选择否：暂不存盘" & Chr(13) _
& "选择取消：不再出现这个提示", vbYesNoCancel + 64, "休息一会吧！")
' 提示用户保存当前活动文档。
If msg = vbYes Then ActiveWorkbook.Save Else If msg = vbCancel Then Exit Sub
Call runtimer ' 如果用户没有选择取消就再次调用 Runtimer
End Sub

```

以上只是两个简单的例子，有兴趣的话，可以利用 Application.OnTime 这个函数写出更多更有用的定时程序。

第六课 提高 Excel 中 VBA 的效率

由于 Microsoft Office 办公套件的广泛应用，以及该软件版本的不断提升，功能不断完善，在 Office 办公套件平台上开发出的 VBA 应用程序越来越多，而 VBA 是一种宏语言，在运行速度上有很大的限制。因此 VBA 编程的方法直接关系到 VBA 程序运行的效率，本文列举了一些提高 VBA 程序运行效率的方法。

方法 1：尽量使用 VBA 原有的属性、方法和 Worksheet 函数

由于 Excel 对象多达百多个，对象的属性、方法、事件多不胜数，对于初学者来说可能对它们不全部了解，这就产生了编程者经常编写与 Excel 对象的属性、方法相同功能的 VBA 代码段，而这些代码段的运行效率显然与 Excel 对象的属性、方法完成任务的速度相差甚大。例如用 Range 的属性 CurrentRegion 来返回 Range 对象，该对象代表当前区。（当前区指以任意空白行及空白列的组合为边界的区域）。同样功能的 VBA 代码需数十行。因此编程前应尽可能多地了解 Excel 对象的属性、方法。

充分利用 Worksheet 函数是提高程序运行速度的极度有效的方法。如求平均工资的例子：

```

For Each c In
    Worksheet(1).Range(" A1:A1000" )
    TotalValue = TotalValue + c.Value
Next
AverageValue = TotalValue / Worksheet(1).Range(" A1:A1000" ).Rows.Count

```

而下面代码程序比上面例子快得多：

```

AverageValue=Application.WorksheetFunction.Average(Worksheets(1).Range( "
A1:A1000" ))

```

其它函数如 Count, Counta, Countif, Match, Lookup 等等，都能代替相同功能的 VBA 程序代码，提高程序的运行速度。

方法 2：尽量减少使用对象引用，尤其在循环中

每一个 Excel 对象的属性、方法的调用都需要通过 OLE 接口的一个或多个调用，这些 OLE 调用都是需要时间的，减少使用对象引用能加快 VBA 代码的运行。例如

1. 使用 With 语句。

```

Workbooks(1).Sheets(1).Range(" A1:A1000" ).Font.Name=" Pay"
Workbooks(1).Sheets(1).Range(" A1:A1000" ).Font.FontStyle=" Bold" ...

```

则以下语句比上面的快

```

With Workbooks(1).Sheets(1).Range(" A1:A1000" ).Font
    .Name = " Pay"
    .FontStyle = " Bold"

```

```
...
End With
```

2. 使用对象变量。

如果你发现一个对象引用被多次使用，则你可以将此对象用 Set 设置为对象变量，以减少对对象的访问。如：

```
Workbooks(1).Sheets(1).Range(" A1" ).Value = 100
Workbooks(1).Sheets(1).Range(" A2" ).Value = 200
```

则以下代码比上面的要快：

```
Set MySheet = Workbooks(1).Sheets(1)
MySheet.Range(" A1" ).Value = 100
MySheet.Range(" A2" ).Value = 200
```

3. 在循环中要尽量减少对象的访问。

```
For k = 1 To 1000
    Sheets(" Sheet1" ).Select
    Cells(k, 1).Value = Cells(1, 1).Value
Next k
```

则以下代码比上面的要快：

```
Set TheValue = Cells(1, 1).Value
Sheets(" Sheet1" ).Select
For k = 1 To 1000
    Cells(k, 1).Value = TheValue
Next k
```

方法 3：减少对象的激活和选择

如果你的通过录制宏来学习 VBA 的，则你的 VBA 程序里一定充满了对对象的激活和选择，例如 Workbooks(XXX).Activate、Sheets(XXX).Select、Range(XXX).Select 等，但事实上大多数情况下这些操作不是必需的。例如

```
Sheets(" Sheet3" ).Select
Range(" A1" ).Value = 100
Range(" A2" ).Value = 200
```

可改为：

```
With Sheets(" Sheet3" )
    .Range(" A1" ).Value = 100
    .Range(" A2" ).Value = 200
End With
```

方法 4：关闭屏幕更新

如果你的 VBA 程序前面三条做得比较差，则关闭屏幕更新是提高 VBA 程序运行速度的最有效的方法，缩短运行时间 2/3 左右。关闭屏幕更新的方法：

```
Application.ScreenUpdate = False
```

请不要忘记 VBA 程序运行结束时再将该值设回来：

```
Application.ScreenUpdate = True
```

以上是提高 VBA 运行效率的几种方法。

第七课 如何在 Excel 里使用定时器

用过 Excel 97 里的加载宏“定时保存”吗？可惜它的源程序是加密的，现在就上传一篇介绍实现它的文档。

在 Office 里有个方法是 application.ontime ，具体函数如下：

expression.OnTime(EarliestTime, Procedure, LatestTime, Schedule)

如果想进一步了解，请参阅 Excel 的帮助。

这个函数是用来安排一个过程在将来的特定时间运行，（可为某个日期的指定时间，也可为指定的时间段之后）。通过这个函数我们就可以在 Excel 里编写自己的定时程序了。下面就举两个例子来说明它。

1. 在下午 17:00:00 的时候显示一个对话框。

```
Sub Run_it()
```

```
Application.OnTime TimeValue("17:00:00"), "Show_my_msg"
```

```
' 设置定时器在 17:00:00 激活，激活后运行 Show_my_msg 。
```

```
End Sub
```

```
Sub Show_my_msg()
```

```
msg = MsgBox("现在是 17:00:00 ! ", vbInformation, "自定义信息")
```

```
End Sub
```

2. 模仿 Excel 97 里的“自动保存宏”，在这里定时 5 秒出现一次

```
Sub auto_open()
```

```
MsgBox "欢迎你，在这篇文档里，每 5 秒出现一次保存的提示！ ", vbInformation, "请注意！ "
```

```
Call runtimer ' 打开文档时自动运行
```

```
End Sub
```

```
Sub runtimer()
```

```
Application.OnTime Now + TimeValue("00:00:05"), "saveit"
```

```
' Now + TimeValue("00:15:00") 指定在当前时间过 5 秒钟开始运行 Saveit 这个过程。
```

```
End Sub
```

```
Sub SaveIt()
```

```
msg = MsgBox("朋友，你已经工作很久了，现在就存盘吗？ " & Chr(13) _
```

```
& "选择是：立刻存盘" & Chr(13) _
```

```
& "选择否：暂不存盘" & Chr(13) _
```

```
& "选择取消：不再出现这个提示", vbYesNoCancel + 64, "休息一会吧！ ")
```

```
' 提示用户保存当前活动文档。
```

```
If msg = vbYes Then ActiveWorkbook.Save Else If msg = vbCancel Then Exit Sub
```

```
Call runtimer ' 如果用户没有选择取消就再次调用 Runtimer
```

```
End Sub
```

以上只是两个简单的例子，有兴趣的话，可以利用 Application.OnTime 这个函数写出更多更有用的定时程序。

三、学习微软 Excel 2002 VBA 编程和 XML, ASP 技术

作者: Julitta Korol 翻译: Tiger Chen Nov 28' 2004

本书展示了Excel 2002 在标准用户界面之外什么是可行的。如果你曾经想不使用菜单来打开一个新的工作表, 或者创建一个充分自动化的自定义窗体来收集数据和在工作表里存储结果, 那么你必须学习一些编程。本书教你如何通过将一些费时的和重复的工作交给Excel, 从而更加成果丰富。使用Excel内置语言, VBA (Visual Basic for Applications), 你将给自己或他人带来非常高的自动化程度的电子表格。通过使用许多内置的编程工具, 你做得可以比想象中容易得多。你不要增加额外费用, 除非你想熟悉Excel背后的秘密。在Excel窗口下, 同时按下Alt+F11, 你将进入VB编辑器界面——Excel的编程界面。既然这个保护得很好的秘密已经公开了, 就让我告诉你更多一些。除了VBA之外, 本书还介绍了两种可以和Excel并用的热门英特网技术。一种是ASP (Active Server Pages), 另一种是XML (Extensible Markup Language)。你也可以学习到许多其它的支持技术。因此, 如果你真正想要获得一些热门技术, 请立即购买本书, 并且不要浪费时间, 马上开始学习。*Learn Microsoft Excel 2002 VBA Programming with XML and ASP* 带领你从始至终创建VBA过程, VBScripts, ASP 页面, XML 文件和XSL 工作表。沿着这条路, 有许多详细的, 适用的“如何做”例子和插图。本书的方法是“由做而学”。本书的前面几章介绍了一些基本的VBA概念, 循序渐进, 复杂的主题在后面的章节。十七章中的每一章是按循序的。此外, 本书还由四章附录, 讨论在Excel里针对一些特殊方面的操作和编程。本书可以当作是一种在办公室或家里学习的课程。许多课程都有前提条件, 本书也不例外。*Learn Microsoft Excel 2002 VBA Programming with XML and ASP* 不会向你介绍Excel的基本东西, 例如菜单和快捷键。我们假设你已经喜欢使用Excel, 并且有兴趣学习如何与Excel在它自己的语言里交流, 学习如何将它与现在的英特网技术结合。

第一章 电子表格自动化简介和了解宏命令

<http://club.excelhome.net/dispbbs.asp?boardID=2&ID=72173&page=5>

你准备好了增进你的微软 Excel 2002 电子表格的智能吗? 通过将日常工作自动化, 你可以使你的电子表格更快, 更有效。本章带领你步入使用宏命令来加速电子表格的过程。你将学到宏是什么, 如何以及什么时候使用它们, 乃至如何编写和修改宏代码。开始学习宏命令很简单。创建宏并不需要什么, 只是一些你已经拥有的知识——基本的微软Excel 2002 菜单知识和电子表格概念。你准备好开始了吗? 确保你坐在计算机前并且打开了Excel 2002。

1 了解宏

宏是一些储存了一系列命令的程序。当你创建一个宏命令的时候, 你只是将一系列的键盘输入结合成一个简单的命令, 你以后可以“回演”这个命令。因为宏命令可以减少复杂任务的步骤, 使用宏命令可以显著得减少你花在创建, 设置格式, 修改和打印工作表的时间。你可以通过Excel内置的录制工具来创建宏命令, 也可以在代码编辑器里面直接写代码。微软 Excel 2002 电子表格具有强大的编程功能。

技巧1—1: 普通语言

Excel 5 是市场上第一个使用VBA的软件。从那以后, VBA开拓了在所有微软办公应用软件中的应用。这意味着你从本书中学习的VBA将来同样可以应用到其它微软办公软件中, 例如: Word, PowerPoint, Outlook or Access

2 宏命令的普通应用

微软 Excel 2002 带来了许多内置, 节省时间的特点, 这些使你工作得更快更聪明。在你决定用宏命令来自动化工作表任务前, 确保没有现成的内置工具来做这项任务。然而, 当你发现你需要反复地做一些动作, 或者Excel没有提供一个内置工具帮你完成该任务, 那么创建一个宏命令吧。宏命令可以使你能够将工作表的任何部分工作自动化。例如, 你可以自动化数据录入——创建一个宏命令在工作表输入标题或者用新标签取代列标题。宏命令可以帮你检查选中的工作表区域里的重复值。你可以通过宏命令快速地将格式应用到多个工作表, 并且可以结合不同的格式, 例如字体, 颜色, 边框和阴影等。尽管如此, Excel 还拥有非常强大的图表功能, 如果你想要将图表创建和格式设置自动化, 那么宏命令是一个好方法。宏命令也可以帮助你设置打印区域, 页边距, 页眉, 页脚, 以及选择特殊的打印选项。

3 写宏之前的计划

在你创建一个宏命令之前，花几分钟来考虑你究竟想做什么。因为宏命令是一大堆键盘输入的集合，事先计划你的行动非常重要。最早的计划宏命令的方法是手动地将宏命令需要做的事情做一遍。在你做键盘输入的同时，在一张纸上记录下他们实际发生的情况，不要漏掉任何东西。象录音机一样，Excel 可以将你的所有动作录制下来（译者：事实上并非如此，有些操作是无法录制的）。如果在录制宏之前，你没有很好地计划，你会录制很多不必要的步骤，而这些都会影响运行速度。尽管修改宏代码比去除录制宏里面不必要的步骤容易，但是，仅仅录制必要的步骤会节省你修改代码的时间和以后的麻烦。

假设你想看一眼哪些区域是文本，哪些是数字以及哪些是公式，图1-1显示了使用不同的字体颜色和样式来区分这些单元格里潜在的内容。

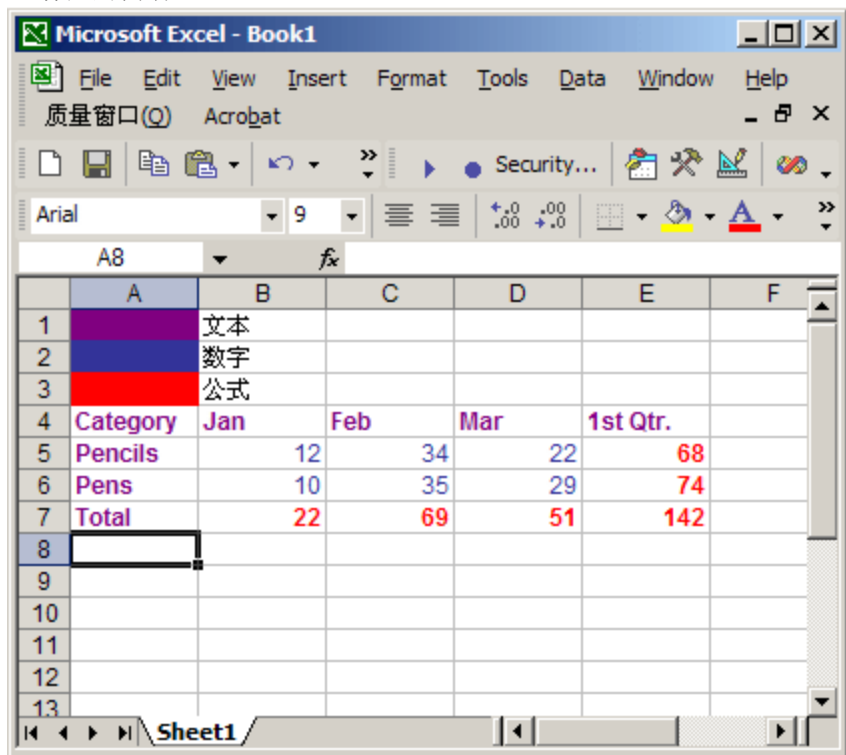


图1-1

如何设置如图1-1所示的格式？打开一个含有公式计算的工作表，或者创建一个如图所示的例子。如果你用图示例子，请确保用SUM函数计算每月和季度的总结。

在录制宏之前，请做如下操作：

1. 选取一个单元格
2. 选择“编辑”——“定位”
3. 在“定位”对话框中，点击“特殊”按钮
4. 在“特殊”对话框中，勾选“常数”——勾选“文本”，同时去除“数字”，“逻辑值”和“错误值”的勾选
5. 按“确定”返回工作表。注意，这时含有文本的单元格已经被选中了。小心，不要改变选中区域，直到你在下一步做一些必要的格式设置
6. 对选中区域。选择“格式”——“单元格”
7. 在单元格格式设置对话框，选择“字体”——设置字体为“粗体”，颜色为“紫色”。然后点击“确定”关闭对话框。注意，含有文本的单元格显示了不同的颜色。

步骤1到7教你定位到文本单元格，要定位数字单元格，请按如下操作：

8. 选取一个单元格
9. 选择“编辑”——“定位”
10. 在“定位”对话框中，点击“特殊”按钮
11. 在“特殊”对话框中，勾选“常数”——勾选“数字”，同时去除“文本”，“逻辑值”和“错误值”的勾选
12. 按“确定”返回工作表。注意，这时含有数字的单元格已经被选中了。小心，不要改变选中区域，直到你在下一步做一些必要的格式设置

13. 对选中区域。选择“格式”－“单元格”
14. 在单元格格式设置对话框，选择“字体”－设置字体颜色为“暗蓝色”。然后点击“确定”关闭对话框。注意，含有数字的单元格显示了不同的颜色。

步骤8到14教你定位到数字单元格，要定位公式单元格，请按如下操作：

15. 选取一个单元格
16. 选择“编辑”－“定位”
17. 在“定位”对话框中，点击“特殊”按钮
18. 在“特殊”对话框中，勾选“公式”
19. 按“确定”返回工作表。注意，这时含有公式计算结果的单元格已经被选中了。小心，不要改变选中区域，直到你在下一步做一些必要的格式设置
20. 对选中区域。选择“格式”－“单元格”
21. 在单元格格式设置对话框，选择“字体”－设置字体为“粗体”，颜色为“红色”。然后点击“确定”关闭对话框。注意，含有公式的单元格显示了不同的颜色。

步骤15到21教你定位到公式单元格。你可以在工作表中加上图例，以便容易理解。

22. 选取区域A1:A3，选择“插入”－“行”
23. 选择单元格A1
24. 选择“格式”－“单元格”，并且在“填充色”页点击“紫色”，点击“确定”返回工作表
25. 选择B1，输入“文本”
26. 选择单元格A2
27. 选择“格式”－“单元格”，并且点击“暗蓝色”，“确定”返回
28. 选择B2，输入“数字”
29. 选择A3
30. 选择“格式”－“单元格”，点击“红色”，“确定”返回
31. 选择B3，输入“公式”
32. 选择A1

完成步骤22到32后，A1:A3单元格会显示一个简单的颜色图例，如图1－1所示。

正如你所看到的，不管工作表显示的任务多么简单，你却可能需要很多步骤来达到预期效果。创建一个可以重复你刚才操作的宏命令，真的很省时间，特别是当你需要重复这个相同的工作到很多工作表中去。

4 录制宏

既然你已经知道了你需要做哪些操作，是时候打开你的宏录制器来创建你的宏了。在你依照下面的录制步骤之前，请确保你已经清除了刚才例子里的格式。按下Ctrl+A以选中整个工作表，选择“编辑”－“清除”－“格式”选择A1:A3并且选择“编辑”－“删除”。在“删除”对话框，选择“整行”然后点击“确定”。

依照以下步骤来创建你的第一个宏命令：

1. 选择一个单元格

录制宏之前，你应该想好是否要录制当前单元格的位置。如果你需要宏总是从一个特定的位置开始，那么先打开宏录制器再选择你想要宏开始选中的特定位置。如果当前单元格的位置无关紧要，那么先选择一个单元格，然后才打开宏录制器。

选择“工具”－“宏”－“录制新宏”。出现一个录制宏对话框。

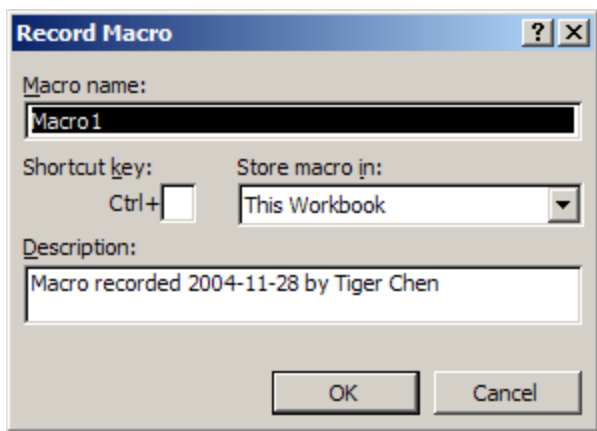


图1-2 当你录制新宏的时候，必须有名字。你也可以给它设置一个快捷键，储存地方以及描述。

2. 给宏取个名字 “WhatsInACell”

技巧1-2：宏命名

如果你忘记给宏命名，Excel 会给出一个默认的宏名，例如：Macro1，Macro2，等等。宏名可以包含字母，数字和下划线，但是第一个字必须是字母（译者：中文亦可，建议用英文）。例如：Report1是有效的宏名，然而1Report则是非法的。宏名里不能含有空格。如果你隔开宏名中的每个词，可以使用下划线。例如：WhatsInACell，改为Whats_In_A_Cell。

3. 在宏的存贮位置里，选择“当前工作簿”

技巧1-3：保存宏

Excel 让你可以将宏保存在三个地方：

个人宏工作簿——如果你将宏保存在这里，你每次使用Excel 的时候都可以使用这个宏。个人宏工作簿在XLStart文件夹中。如果这个工作簿不存在，那么当你第一次使用这个选项的时候，Excel会自动生成这个工作簿。

新工作簿——Excel将宏放在一个新工作簿里。

当前工作簿——宏将被保存在你正在使用的工作簿里面。

4. 在描述框里输入：显示单元格里潜在的内容：文本，数字，公式

5. 点击“确定”关闭宏录制对话框，并开始录制。这时，出现了“停止录制”工具栏。Excel下面的状态栏显示“准备录制”



图1-3 停止录制工具栏提供按钮来停止录制，以及让Excel如何在录制宏的时候处理单元格地址

技巧1-4：单元格地址：相对或绝对？

绝对——如果在执行宏命令的过程中，无论哪些单元格被选中了，你都希望宏在特定的单元格执行这些录制的操作，那么使用绝对单元格地址。绝对单元格引用具有如下形式：\$A\$1，\$C\$5等。Excel宏录制器默认使用绝对引用。在录制前，确保停止录制工具栏上的第二个按钮没有被按下去。当鼠标指向这个按钮，工具提示“相对引用”。

相对——如果你想要宏可以用在任何区域，就打开相对引用。相对引用具有如下形式：A1，C5等。在录制前，确保停止录制工具栏上的第二个按钮已经被按下去了。记住，Excel 将会继续使用相对引用，直到退出Excel或者再次点击相对引用按钮。在录制宏的过程中，你可以使用这两种引用方法。例如：你可以选择一个特定单元格（如\$A\$4），做一个操作，然后选择另外一个相对于当前位置的单元格（如C9，他在当前单元格\$A\$4往下5行和向右2列的位置）。当你复制单元格时，相对引用会自动调整引用，而绝对引用则不会。

6. 从新进行刚才你手动完成的那些操作（参见“写宏之前的计划”）
录制宏的时候，只有当你按下了回车键或者点击了确定之后，你的操作才会被录制。如果你按下了Esc键或者点击了取消，宏录制器不会录制任何操作。
7. 完成所有操作后，点击停止录制工具栏上的“停止录制”按钮，或者选择“工具”－“宏”－“停止录制”

5 运行宏

你创建了一个宏命令后，至少要运行一次以确保它运行正确。在本章的后面，你将学到好几种运行宏的方法，不过，现在，使用菜单命令。要看到你的成果，确保清除了例子的格式。按下Ctrl+A以选中整个工作表，选择“编辑”－“清除”－“格式”选择A1:A3并且选择“编辑”－“删除”。在“删除”对话框，选择“整行”然后点击“确定”。稍后，你将在另外一个宏里面录制清除工作表格式的步骤。

1. 打开任何包含文本，数字和公式的工作表
2. 选择“工具”－“宏”－“运行宏”来打开宏对话框
3. 点击你要运行的宏的名称（参见图1-4）
4. 选择“运行”，执行宏

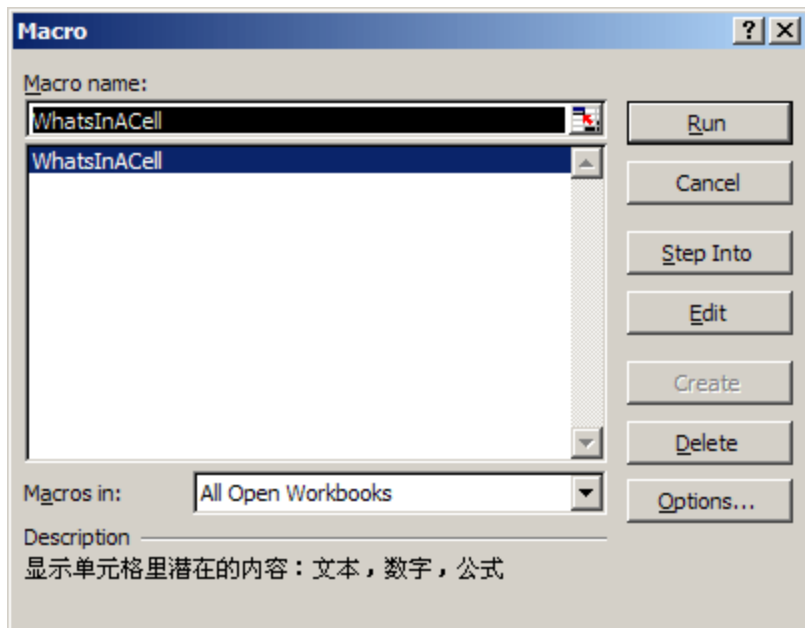


图1-4 在宏对话框，你可以选择一个宏，运行，编辑或者删除它

你也许经常会发现录制的宏不会按你预期的和你第一次操作那么运行。也许在录制宏的时候，你选择了错误的字体，或者忘记改变单元格颜色，或者你临时发现最好加上一个步骤。不必惊慌。Excel允许你修改代码，而不会强迫你重新录制那些单调的操作。

6 修改宏代码

你必须知道你的宏代码放在哪里，你才能找到并修改它。回想你打开宏录制器的时候，你选择了“当前工作簿”作为存储地址。最容易找到宏的方法是打开宏对话框，如图1-4所示。

1. 选择“工具”－“宏”
2. 选择宏名（本例中为WhatsInACell）
3. 点击“编辑”按钮

Excel 打开一个专门的窗口，叫做Visual Basic Editor（VBE）如图1-5所示。利用快捷键Alt+F11可快速地在Excel表格界面和代码窗口切换。选择VBE菜单上的关闭选项可以关闭VBA代码窗口，返回到电子表格界面。

代码窗口暂时看上去有些令人迷惑，不必担心。只要你开始录制宏，以及尝试写一些代码，你终将这个屏幕所有的组件。现

在，看一下代码窗口的菜单和工具栏。这两个工具栏和Excel窗口的菜单完全不同。代码窗口的菜单和工具包含一些编程和测试代码所需要的工具。只要你彻底地学习本书的每一章，你就会成为使用这些工具的专家。

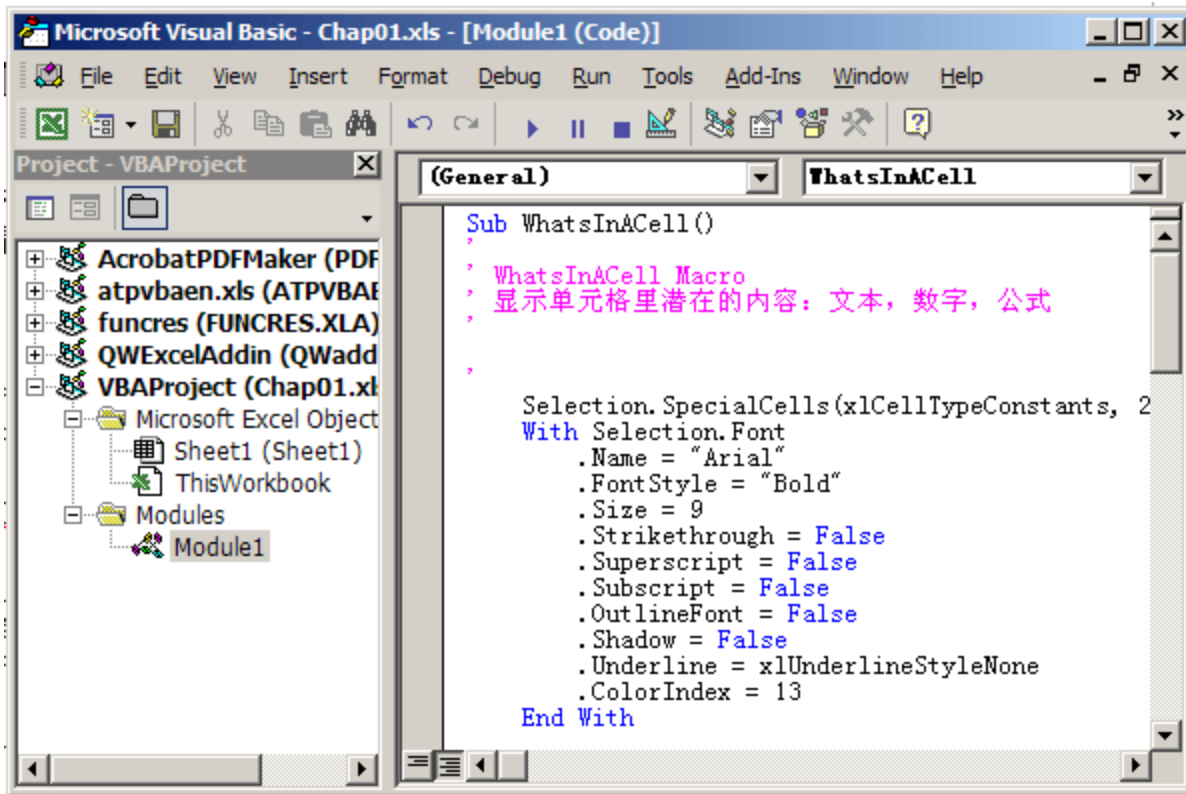


图1-5 VBE窗口是编辑宏命令和书写新的VBA代码的地方

VBE窗口的主要部分是多个窗口的集合界面，这些窗口在你创建和测试VBA过程的时候是及其有用的。图1-5显示了三个集合在一起的窗口：工程窗口，属性窗口和代码窗口。工程窗口显示一个开启的模块文件夹，在这里，模块1被选中了。Excel录制你在工作表里的操作叫做模块1，模块2，等等。在本书接下来的章节里，你将利用模块来编写你自己的过程代码。模块类似于Word中的一个空白文档。储存每个单独模块的文件夹称为“模块”

技巧1-5：宏还是过程？

宏是通过内置宏录制器录制的，或者在VB编辑器里手动输入的一系列指令或函数。从Excel 5.0开始，“宏”经常被“过程”这个更广的概念所代替。尽管这两个词可以交替互换使用，但是，许多编程者更喜欢“过程”。虽然宏可以让你模仿键盘操作，真正的过程则还可以执行一些不能通过鼠标，键盘或者菜单来做的操作。换句话说，过程是一个更复杂的宏，它结合了传统编程语言的言语结构。

代码窗口（参见图1-5）显示了下列由宏录制器录制的代码：

```
Sub WhatsInACell()
'
' WhatsInACell Macro
' Macro recorded 5/31/2002 by Julitta Korol
' Indicates the contents of the underlying cells: text, numbers, formulas.
'
Selection.SpecialCells(xlCellTypeConstants, 2).Select
With Selection.Font
    .Name = "Arial"
    .FontStyle = "Bold"
    .Size = 10
End With
```

```
.Strikethrough = False
.Superscript = False
.Subscript = False
.OutlineFont = False
.Shadow = False
.Underline = xlUnderlineStyleNone
.ColorIndex = 13
End With
Range("B6").Select
Selection.SpecialCells(xlCellTypeConstants, 1).Select
With Selection.Font
    .Name = "Arial"
    .FontStyle = "Regular"
    .Size = 10
    .Strikethrough = False
    .Superscript = False
    .Subscript = False
    .OutlineFont = False
    .Shadow = False
    .Underline = xlUnderlineStyleNone
    .ColorIndex = 11
End With
Range("C6").Select
Selection.SpecialCells(xlCellTypeFormulas, 23).Select
With Selection.Font
    .Name = "Arial"
    .FontStyle = "Bold"
    .Size = 10
    .Strikethrough = False
    .Superscript = False
    .Subscript = False
    .OutlineFont = False
    .Shadow = False
    .Underline = xlUnderlineStyleNone
    .ColorIndex = 3
End With
Range("A1:A3").Select
Selection.EntireRow.Insert
Range("A1").Select
With Selection.Interior
    .ColorIndex = 13
    .Pattern = xlSolid
    .PatternColorIndex = xlAutomatic
End With
Range("B1").Select
ActiveCell.FormulaR1C1 = "Text"
Range("A2").Select
With Selection.Interior
    .ColorIndex = 5
    .Pattern = xlSolid
```

```

        .PatternColorIndex = xlAutomatic
End With
Range("B2").Select
ActiveCell.FormulaR1C1 = "Numbers"
Range("A3").Select
With Selection.Interior
    .ColorIndex = 3
    .Pattern = xlSolid
    .PatternColorIndex = xlAutomatic
End With
Range("B3").Select
ActiveCell.FormulaR1C1 = "Formulas"
Range("B4").Select
End Sub

```

从现在开始，让我们注重于寻找下面两个问题的答案：如何阅读宏代码？如何修改宏代码？

7 添加注释

看一下录制的宏代码，请注意那些开头带单引号的行。这些行就是注释。注释默认显示为绿色。执行宏代码时，VB会忽略这些注释行。注释经常和宏代码放在一起，来整理那些意义不甚明显的语句。现在，我们来给宏WhatsInACell 添加注释。

1. 激活VBE窗口
2. 在Selection.SpecialCells(xlCellTypeConstants, 2).Select前面点击一下，将光标移至该语句开头，回车
3. 将光标往上移一行到空白处，并且添加如下注释，注意前面有单引号（译者：英文状态下的单引号）
‘ Find and format cells containing text
4. 在Selection.SpecialCells(xlCellTypeConstants, 1).Select前面点击一下，将光标移至该语句开头，回车
5. 将光标往上移一行到空白处，并且添加如下注释，注意前面有单引号（译者：英文状态下的单引号）
‘ Find and format cells containing numbers
6. 在Selection.SpecialCells(xlCellTypeFormulas, 23).Select前面点击一下，将光标移至该语句开头，回车
7. 将光标往上移一行到空白处，并且添加如下注释，注意前面有单引号（译者：英文状态下的单引号）
‘ Find and format cells containing formulas
8. 在Range("A1:A3").Select前面点击一下，将光标移至该语句开头，回车

技巧1—6：关于注释

在VBE代码窗口里，以单引号开头的都是注释。注释的默认颜色是绿色。可以通过“选项”对话框（“工具”—“选项”—“编辑器格式”）更改注释颜色。注释也可以写在代码的后面。例如，在语句.ColorIndex = 11之后添加注释。点击该语句句末，按下Tab键，输入单引号，然后输入注释。显示如下：

```
.ColorIndex = 11 ' Sets the font color to Violet
```

注释除了给用户代码目的信息之外，没有任何作用。请别忘记给你的代码写上注释。如果你几个月后还要回到你的代码，那么注释将帮你大忙。同样，注释可以使别人很快理解你的程序。

9. 将光标往上移一行到空白处，并且添加如下注释，注意前面有单引号（译者：英文状态下的单引号）
‘Create legend

8 分析宏代码

所有宏过程都以关键词“Sub”开始，以关键词“End Sub”结束。在关键词“Sub”之后是宏的真正的名字，然后紧跟着是一对括号。在关键词Sub 和End Sub之间是那些你每次运行宏代码时VB执行的语句。VB从上到下读取语句，忽略那些句前带单引号的语句（参见上节关于注释的内容），读到End Sub时停止。请注意，录制的宏代码里包含许多停顿（译者：英文模式下的句号）。每行代码中都有停顿，用来连接VBA语言中不同的要素。如何阅读这种语言的用法呢？要从最后一个停顿的右边向左

读。看看WhatsInACell里的一些语句：

```
Range("A1:A3").Select
```

选择A1到A3单元格

```
Selection.EntireRow.Insert
```

往选中的区域中插入行。因为前面你选中的的是三个单元格（译者：应该说是占据了三行的单元格），VB将插入三行。

```
ActiveCell.FormulaR1C1 = "Text"
```

往当前单元格里输入“Text”。因为，之前的代码是Range("B1").Select，选择单元格B1，B1是当前激活的单元格，所有VB往B1单元格里面输入文本。

```
With Selection.Interior
```

```
.ColorIndex = 3
```

```
.Pattern = xlSolid
```

```
.PatternColorIndex = xlAutomatic
```

```
End With
```

这是一段特别的代码块，解释如下：给当前选中的单元格设置单元格填充色为红色（ColorIndex = 3），设置填充模式为实心（xlSolid），并且给当前单元格明确为默认的填充模式（xlAutomatic）。这个代码块以关键词With开始，End With结束，它将加速宏代码的执行。宏代码知道走捷径，而不会每次都重复下面的说明：

```
Selection.Interior.ColorIndex = 3
```

```
Selection.Interior.Pattern = xlSolid
```

```
Selection.Interior.PatternColorIndex = xlAutomatic
```

在关键词With后面紧跟重复的Selection.Interior，再以End With结尾。

9 清除宏代码

你已经逐行解析了你宏代码，你会发现Excel录制了许多你并不想要包含进去的信息。例如，在选中了文本单元格后，除了设置字体为粗体和颜色为紫色之外，Excel 还录制了其它在字体页的选项——字体名称，字体大小，删除线，上标，下标，阴影和下划线。请看下列代码片断：

```
With Selection.Font
```

```
.Name = "Arial"
```

```
.FontStyle = "Bold"
```

```
.Size = 10
```

```
.Strikethrough = False
```

```
.Superscript = False
```

```
.Subscript = False
```

```
.OutlineFont = False
```

```
.Shadow = False
```

```
.Underline = xlUnderlineStyleNone
```

```
.ColorIndex = 13
```

```
End With
```

如果你使用了对话框，Excel总会录制所有的设定。这些多余的代码使得你的宏代码冗长而难以理解。因此，你完成录制宏后，最好检查一遍你录制的代码并删除不必要的行。

1. 在下面的代码中，删除带删除线的行：

```
With Selection.Font
```

```
.Name = "Arial"
```

```
.FontStyle = "Bold"
```

```
.Size = 10
```

```
.Strikethrough = False
```

```
.Superscript = False
```

```
.Subscript = False
```

```
.OutlineFont = False
```



```
.Shadow = False  
.Underline = xlUnderlineStyleNone  
.ColorIndex = 13
```

```
End With
```

清除后，在关键词With和End With之间只剩下了两句代码，这些才是你在录制宏的时候真正做的设置：

```
With Selection.Font  
.FontStyle = "Bold"  
.ColorIndex = 13
```

```
End With
```

2. 找到设置数字单元格格式的代码，依照下面的例子修改代码：

```
' Find and format cells containing numbers  
With Selection  
.SpecialCells(xlCellTypeConstants, 1).Select  
.Font.ColorIndex = 11 ' Sets the font color to Violet  
End With  
Range("C6").Select
```

3. 找到设置公式单元格格式的代码，依照下面的例子修改代码：

```
' Find and format cells containing formulas  
Selection.SpecialCells(xlCellTypeFormulas, 23).Select  
With Selection.Font  
.FontStyle = "Bold"  
.ColorIndex = 3  
End With
```

4. 找到下述两行代码：

```
Range("A1:A3").Select  
Selection.EntireRow.Insert
```

5. 用下面的一句代码取代上面的两句代码：

```
Range("A1:A3").EntireRow.Insert
```

注意，Excel使用了R1C1形式来设置选中单元格个公式：

```
ActiveCell.FormulaR1C1 = "Text"  
ActiveCell.FormulaR1C1 = "Numbers"  
ActiveCell.FormulaR1C1 = "Formulas"
```

宏录制器使用了一次“ActiveCell”和一次“Selection”来选择当前单元格。这两个词都称为属性。你将在第二章里学习属性。当仅有一个单元格被选中时，你可以随意使用“ActiveCell”或者“Selection”。

10 测试修改好的宏

当你修改宏的时候，很可能会带入一些错误。例如，你可能会删除一行重要的代码，或者可能不注意清除或忽略了一个逗号（停顿）。为了确保你的宏在你修改之后还能正确地工作，你必须运行它。

在VBE窗口，将光标放在宏代码WhatsInACell的任意行，选择“运行”—“运行模块/窗体”

如果你在修改代码的时候，没有带入任何问题，那么宏将顺利运行，而不会有任何报错。你需要切换到Excel界面取看你的宏运行的结果。你可以点击任务栏，或者按Alt+F11回到Excel界面。

如果宏在运行的过程中遇到错误，你将会看到一个对话框显示发现的错误类型。在你运行宏命令之前，你必须确保你的宏可以在当前激活的工作表里面运行。例如，你当前电脑上激活的时一个空的Excel工作表，你试图运行WhatsInACell，这时你将看到一个错误信息：“运行时间错误‘1004’—找不到单元格”。点击“结束”按钮，在你重新运行宏之前，确保选择了正确的工作表。

如果选择的工作表只含有文本，你在运行WhatsInACell的时候，VB试图选择含有数字的单元格时会遇到同样的“找不到单元格”的错误。

如果你忽略了With Selection.Font中的逗号，VB会出现“运行时间错误‘424’—需要对象”的信息。点击信息框上的“调

试”按钮，你将进到代码窗口。同时，VB会进入“中断”模式，并且将有问题的行用黄色突出出来。在你更正代码后，VB可能会弹出信息：“这个操作将会重置你的工程，继续？”点击确定。尽管你可以在中断模式修改代码，但是，有些修改会终止宏的继续执行。更正宏代码后，重新运行它，也许你会需要解决更多的错误，之后才能顺利地运行它。你将在第二章和第十三章里面学到更多的关于如何处理VBA错误的方法。

11 两个层面运行宏的方法

你既可以在Excel界面运行宏，也可以在VB编辑器界面运行它。当你从VB编辑器屏幕执行WhatsInACell时，VB在屏幕之后执行这些代码。你看不到VB选择和设置格式，也看不到VB插入三空行做图例。为了观察到VB的执行情况，你必须在Excel界面，通过选择“工具”－“宏”，或者将你Excel界面和VB编辑器界面同时显示在电脑屏幕上（参见图1－6）

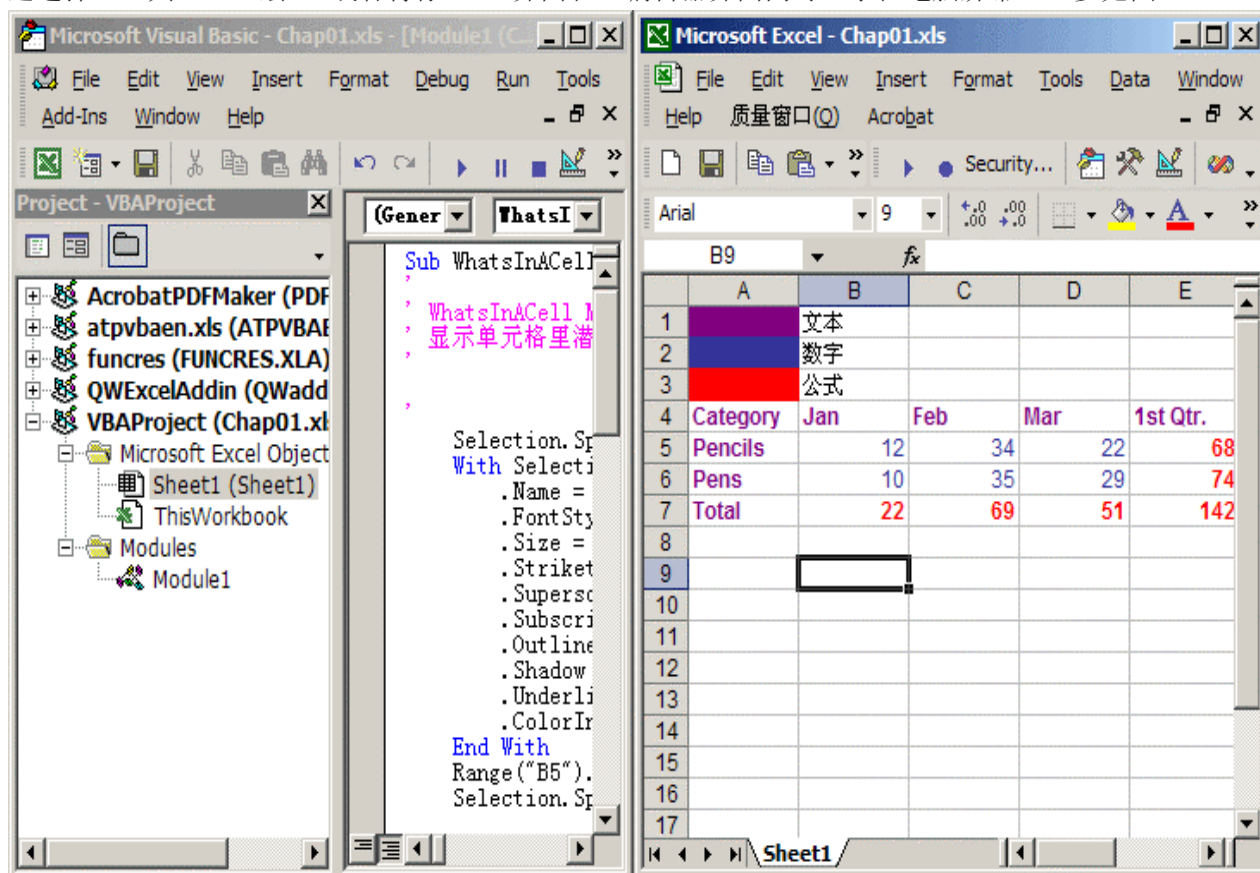


图1－6 如果你从VB编辑器运行宏时，想观察宏的运行情况，你必须将Excel界面和VB编辑器并排地布置在一起

按照下列步骤来并排布置你的Excel界面和VB编辑器界面：

1. 在任务栏上的空白处单击右键。任务栏在屏幕的下端，“开始”按钮的位置。
2. 下列菜单中，选择“纵向平铺窗口”
3. 最小化那些不需要的窗口，重复步骤1
4. 现在，两个窗口并排显示了，点击代码的任意位置，然后按下“F5”（或者选择“运行”－“运行模块/窗体”）。坐好，观察你录制的宏在运行，不是很激动吗？稍后，你将学习如何将VB慢慢运行，这样你就可以一步一步地观察宏代码的运行情况。

12 完善你的宏代码

录制宏后，你可能会发现宏可以进行一些别的操作。如果你已经熟悉了VB语言，要往宏里面加指令并不是一件困难的事情。然而，在绝大多数情况下，你可以将这些工作交给Excel宏录制器，从而更有效地完成这项工作。你也许会说，Excel录制了太多的不需要的指令。但是，肯定的是，宏录制器不会犯错，你完全可以依赖于它。

如果你想要通过宏录制器在你的代码里添加指令，那么你必须录制一个新宏，然后复制需要的部分再粘贴到原来代码的正确

位置。

我们来给A1:B3添加粗边框：

1. 激活图1—6看到的Excel界面
2. 选择“工具”—“宏”—“录制新宏”
3. 在宏对话框点击确定，接受默认的宏名并开始录制
4. 选择区域A1:B3
5. 选择“格式”—“单元格”，点击“边框”页
6. 在“边框样式”部分，点击“外部”按钮
7. 在边框粗细列表，点击最粗的，再点击确定关闭对话框
8. 点击单元格A1。注意，A1:B3区域有了粗边框。
9. 点击“停止录制”按钮，或者选择“工具”—“宏”—“停止录制”

切换至VB编辑器窗口，查看你录制的宏。给A1:B3（译者：原文为A1:A3）添加粗边框的代码如下：

```
Sub Macro2()  
,  
,  
, Macro2 Macro  
, Macro recorded 5/31/2002 by Julitta Korol  
,  
,  
  
    Range("A1:B3").Select  
    Selection.Borders(xlDiagonalDown).LineStyle = xlNone  
    Selection.Borders(xlDiagonalUp).LineStyle = xlNone  
    With Selection.Borders(xlEdgeLeft)  
        .LineStyle = xlContinuous  
        .Weight = xlThick  
        .ColorIndex = xlAutomatic  
    End With  
    With Selection.Borders(xlEdgeTop)  
        .LineStyle = xlContinuous  
        .Weight = xlThick  
        .ColorIndex = xlAutomatic  
    End With  
    With Selection.Borders(xlEdgeBottom)  
        .LineStyle = xlContinuous  
        .Weight = xlThick  
        .ColorIndex = xlAutomatic  
    End With  
    With Selection.Borders(xlEdgeRight)  
        .LineStyle = xlContinuous  
        .Weight = xlThick  
        .ColorIndex = xlAutomatic  
    End With  
    Selection.Borders(xlInsideVertical).LineStyle = xlNone  
    Selection.Borders(xlInsideHorizontal).LineStyle = xlNone  
    Range("A1").Select  
End Sub
```

现在，我们来分析一下这些录制的代码。你认为你可以去掉其中的一些指令吗？在你删除这些不必要的代码之前，考虑使用注释。在你删除任何代码之前，请将它们注释掉，然后运行宏。如果VB没有出现任何错误信息，那么你就可以安全地删除这些被注释了的代码。如果每次都按照这个指导思想，你就不会重复录制相同的操作了。如果这个宏命令没有正确地运行，那么你需要去掉刚才的注释，毕竟，这些代码可能是必须的。关于注释的详细信息，参见第二章。

当你使用宏录制器来创建宏的时候，你可以很快地掌握Excel菜单选项和对话框设置在VBA里的等同方法。然后，你可以在在

线帮助里面查找这些VB指令的意思和用法。很显然，VB要执行越多的指令，宏运行的速度就越慢。去掉那些无关紧要的命令会加速宏的运行。然而，为了使你的代码容易理解，你需要戴上你的侦探帽子，寻求最佳途径。例如，看一下你录制的给选中的单元格加外框的宏。看上去，宏录制器是在分别地给每一条线进行设置。VB没有一个简单的一句命令来给选中的区域加外边框，这似乎很难理解。学习任何语言中正确的词语和表达是很费时的。时间一长，你会发现VB实际上有一个另外的方法BorderAround让你在单元格区域添加边框和设置颜色，线型和新边框的粗细。下面的语句是VBA中给选中的单元格设置外围粗边框的最佳方法：

```
Range("A1:B3").BorderAround Weight:=xlThick
```

上面的指令使用Range对象的BorderAround方法。它给A1:B3区域添加了一个粗线外框。（下一章涵盖了VB对象，属性和方法）。现在我们将上面的指令加到宏WhatsInACell里面去：

1. 激活含有宏WhatsInACell的代码窗口
2. 在ActiveCell.FormulaR1C1 = "Formulas"之后插入一行
3. 在空白行加入以下指令：

```
Range("A1:B3").BorderAround Weight:=xlThick 4
```
4. 光标放在宏代码的任何位置，按F5运行修改好的代码。

技巧1—7：附加指令

要在现存的代码中添加指令的话，通过在需要的位置按回车键加入空白行，并且输入必要的VB语句。如果附加指令是键盘操作或菜单命令的话，你可以使用宏录制器来创建必要的代码，然后将它们复制粘贴到原来的宏里面。

假设你想要VB在执行完最后一行代码时给你提示，这种操作是不可能被录制下来的，因为Excel没有相应的菜单选项。但是，你可以手动使用VB语言在你的代码里面添加指令。

1. 在代码窗口下，在End Sub前回车
 2. 光标放在空白行，输入下列语句：

```
MsgBox "所有操作都已完成。"（译者：英文状态下的引号。）
```
 3. 确保光标在代码里，按下F5
 4. VB执行完最后一个指令后，弹出这个信息。点击确定。你现在知道宏已经运行完成。
- MsgBox是用得非常频繁的VBA函数之一，你将在第四章中学习它的使用。

13 重新命名宏

在代码里面添加了一些代码后，为了更好地反映这个宏的目的，你需要将其改名。过程的名称应该越接近它的功能越好。你不需要按任何键就可以更改宏名。在代码窗口，你将关键词Sub后面老的宏名清除，并且打入新的名称即可。

14 运行宏的其它方法

到现在为止，你已经学习了运行宏的方法。你已经知道通过选择“工具”—“宏”—“运行宏”来运行宏。不幸的是，如果你需要经常运行宏，这种方法是不方便的。你也可以在VB编辑器窗口使用快捷键F5或者通过选择“运行”—“运行模块/窗体”来运行宏。此外，你还可以在VB编辑器窗口点击标准工具栏上的按钮来运行宏（见图1—7）。

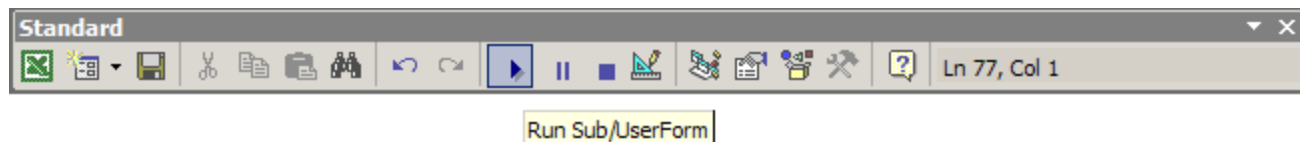


图1—7 VB过程可以通过标准工具栏来运行

15 使用键盘快捷键运行宏

流行的方法是通过设置一个快捷键来运行宏。按Ctrl+Shift+D比从宏对话框激活宏要容易得多。你必须给宏设置一个快捷键，之后才能使用它。

1. 按Alt+F8快速打开宏对话框

2. 点击宏清单里的WhatsInACell，然后选择选项按钮
3. 弹出宏选项对话框，如图1—8。光标定位在快捷键文本框里
4. 按下Shift键和键盘是的字母I。Excel录制下了快捷键Ctrl+Shift+I
5. 点击确定以关闭宏选项对话框
6. 点击取消返回工作表。试试用你刚设置的快捷键来运行宏，确保激活了Excel窗口，然后按下Ctrl+Shift+I

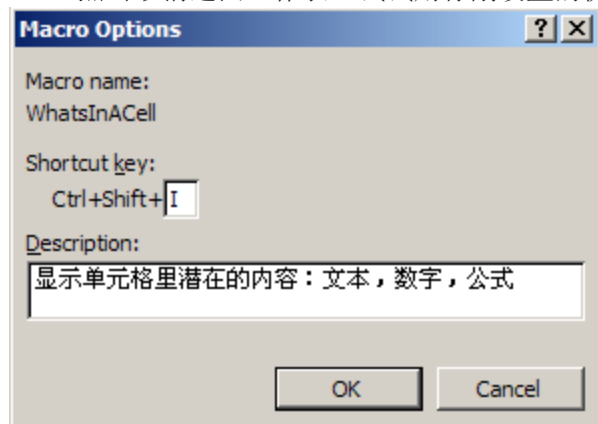


图1—8 使用宏选项对话框设置键盘快捷键来运行宏

技巧1—8：避免快捷键冲突

如果你给宏设置的快捷键和Excel内置的快捷键冲突，而且你打开的又正是含有那个宏的工作表，那么按下该快捷键后Excel会运行你自己的宏。

16 通过菜单运行宏

如果你宁愿通过菜单来运行宏，那么你可以将你的宏做成一个菜单选项。使用“自定义菜单”对话框，你可以快速的将你的宏命令加入到任何Excel的内置菜单中。

1. 在Excel界面工具栏的空白处，单击右键，选择“自定义菜单”
2. 在自定义菜单对话框选择“命令”页
3. 在“类别”清单里选择“宏”

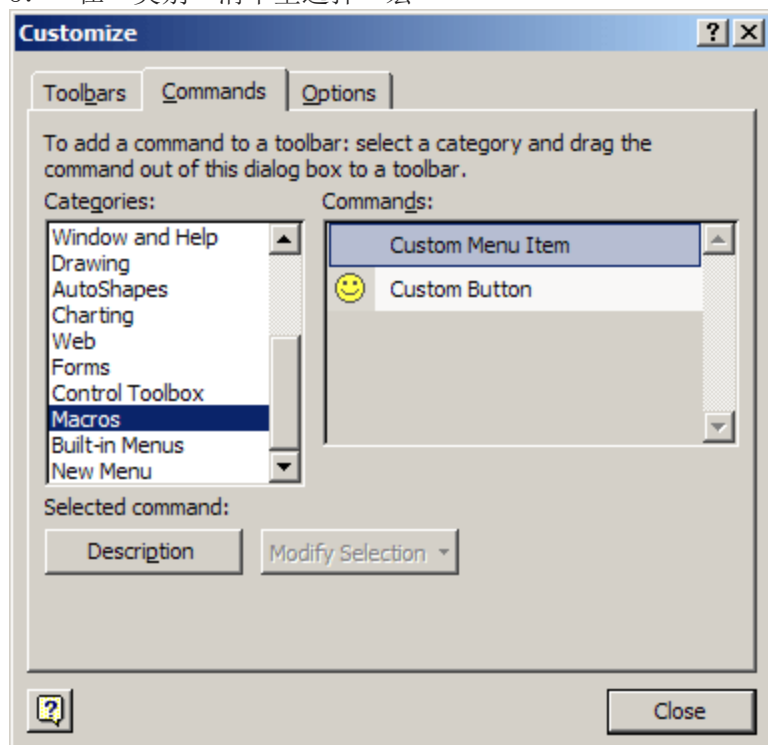


图1-9 创建自定义菜单（第一步）

4. 将“自定义菜单”拖曳至工具菜单里去。当工具菜单展开时，你可以将按钮放在任意地方。图1-10显示了自定义菜单在工具菜单的最下面。

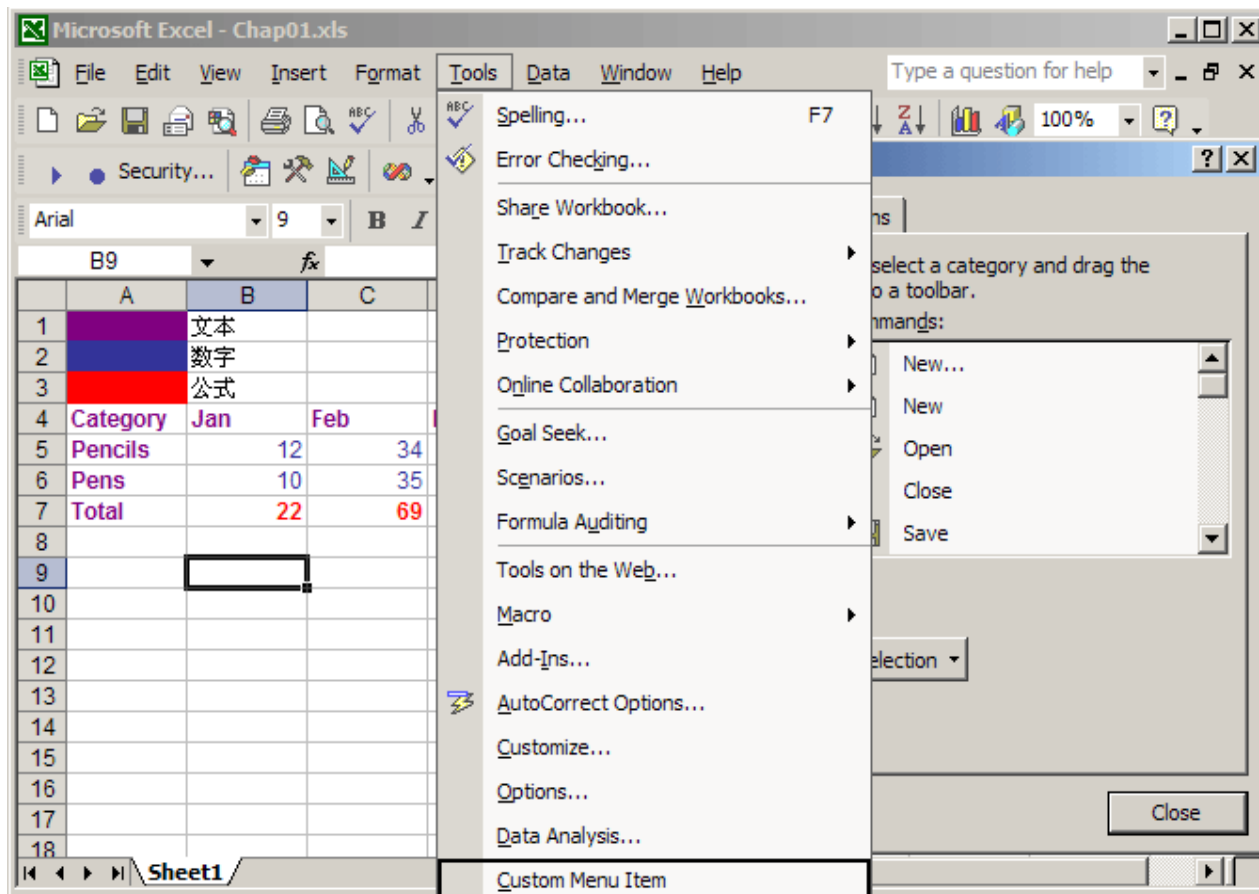


图1-10 创建自定义菜单（第二步），你可以将自定义菜单放在Excel菜单里，也可以放在子菜单里

5. 在菜单项上单击右键，并且在快捷菜单“名称”的文本框里，将其改成你想要的名字（参见图1-11）。例如，将名称改为“Contents of Cells”。连接符用以表示键盘快捷键。将连接符放在你想显示下划线的字符之前。这个自定义菜单将会显示为“Contents of Cells”，注意，菜单里面字与字之间可以有空格。
6. 选择最后一个选项（快捷菜单上）——“指定宏”（参见图1-11）。在宏对话框，选择宏“WhatsInACell”，点击确定，关闭自定义菜单对话框。
- 现在，你的宏可以通过自定义菜单来运行了。如果你没有给自定义菜单选项指定宏就关闭了这个快捷菜单，Excel在你第一次试图使用这个自定义菜单选项时会提示你要宏名。
7. 选择“工具”——“Contents of Cells”，或者按Alt+T和l来运行宏。如果你在做上述操作时，清除了内置菜单或菜单选项，可以打开自定义菜单对话框，点击工具页，然后选择“重置”按钮就可以恢复了。然而，这样操作后，会恢复Excel默认设置，你的自定义菜单选项也不复存在了。

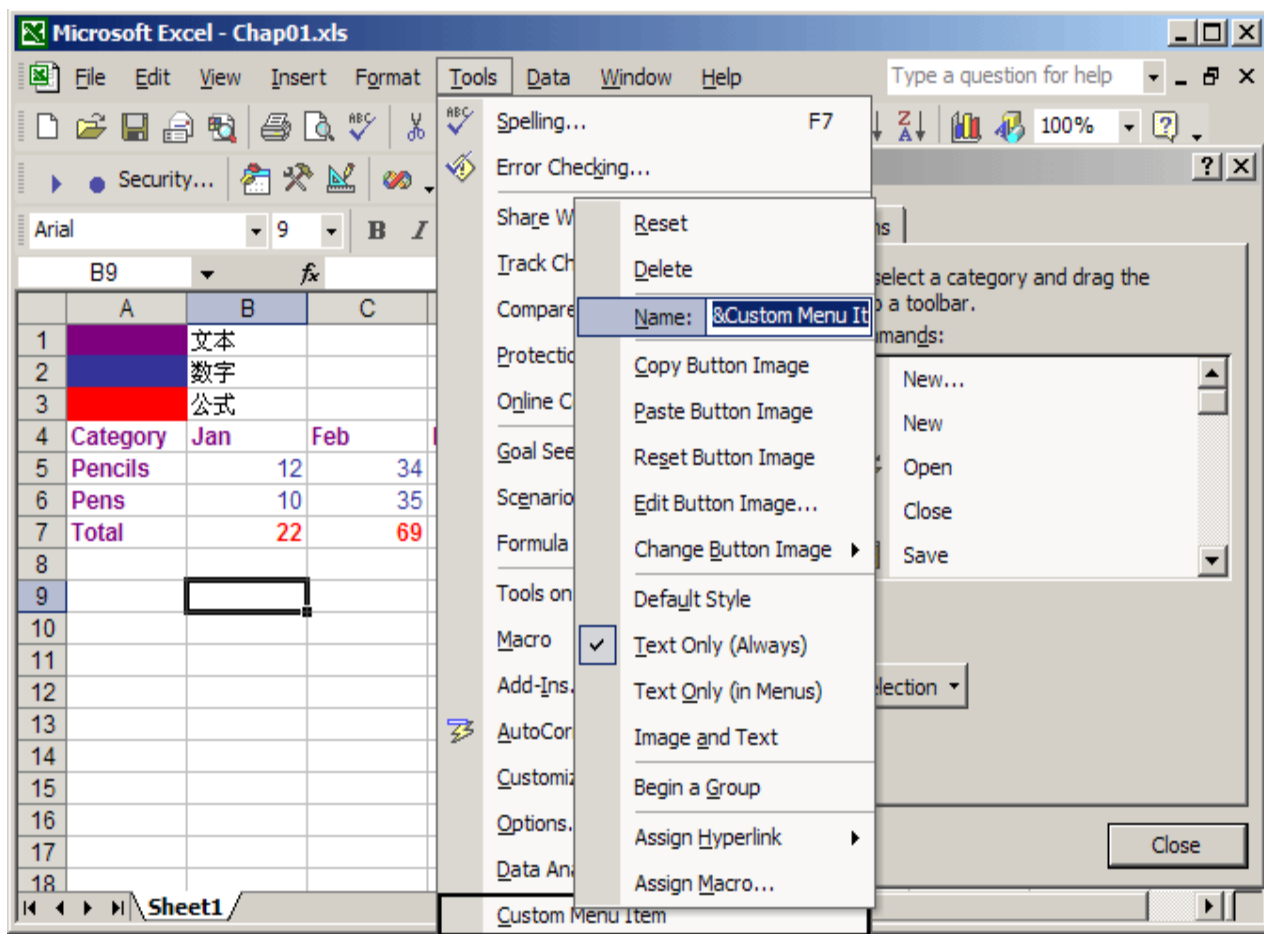


图1-11 创建自定义菜单（第三步）你可以使用快捷菜单给菜单选择重命名，已经设置你自己的宏。你必须先打开自定义菜单，才能使用该快捷菜单

17 通过工具栏按钮运行宏

如果你喜欢使用工具栏里的按钮，你可以轻易地在任何工具栏里添加按钮，并且指定你自己的宏。我们来添加WhatsInACell到工具栏去。

1. 选择“工具”－“自定义”
2. 在自定义对话框，点击“命令”页
3. 在类别清单里选择宏
4. 拖曳“自定义按钮”图标到工具栏的任何地方。在本例中，这个按钮放在标准工具条中格式刷的右边。
5. 修改按钮的工具提示：在按钮上单击右键，然后在出现的快捷菜单的名称选项中，编辑名称文本。本例中，将工具提示改为“Contents of Cells”
6. 修改按钮图标：在按钮上单击右键，并且选择“修改按钮图标”，出现42个Excel预先设计的图标供你选择。本例中，用铅笔图标取代了默认的图标
7. 给按钮指定宏：在按钮上单击右键，并且选择“指定宏”
8. 选择“WhatsInACell”点击确定
9. 点击关闭，关闭自定义对话框
10. 光标指向你刚才创建的自定义按钮上，按钮的旁边显示工具提示“Contents of Cells”（参见图1-12）。点击按钮运行宏

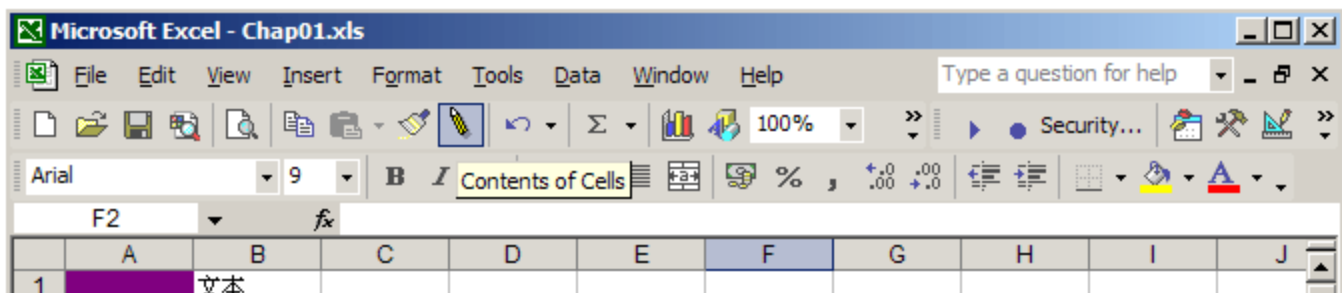


图1-12 你可以在任何工具栏添加自定义按钮来运行宏

18 通过工作表里面的按钮运行宏

在本书后面，你将学习如何在工作表中添加按钮，帮助Excel初学者做数据输入。现在，我们来过一遍如何将宏WhatsInACell指定在一个工作表的按钮上。

1. 激活含有数据的工作表
2. 选择“视图”－“工具栏”，并且选择“窗体”。窗体工具栏出现了，如图1-13

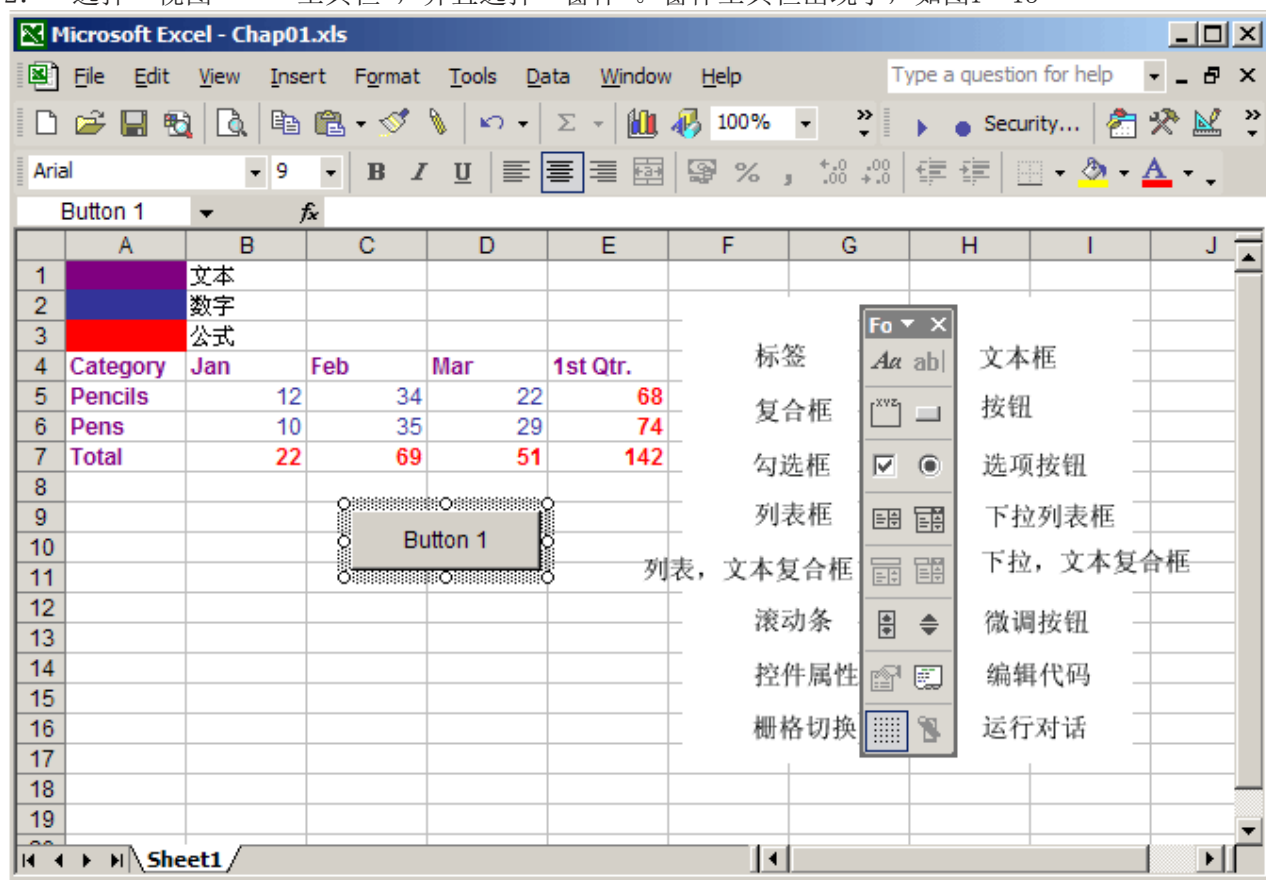


图1-13 你可以将宏指定给一个工作表里的按钮

3. 在窗体工具栏上点击按钮
4. 在工作表任意地方点击一下
5. 当出现指定宏对话框时，选择宏名（WhatsInACell）然后点击确定
6. 改变按钮1的名称：确保选中了按钮，并且输入名称“Contents of Cells”。按钮被选中后，它就像图1-13里显示的一样。如果选择的符号没有显示，在按钮上单击右键，并且在快捷菜单上选择“编辑文本”，选择默认的文字，然后输入新的名称
7. 按钮重命名后，在工作表按钮之外的任何地方点击一下退出按钮编辑状态
8. 点击你刚才创建的按钮，运行宏

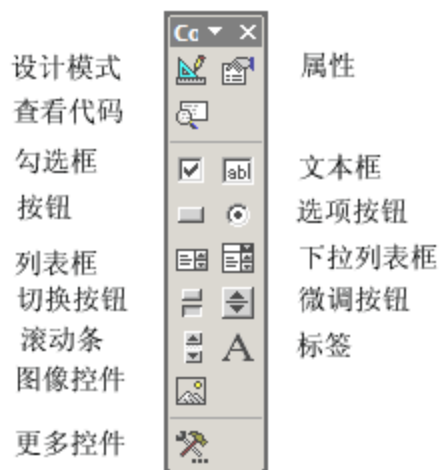


图1-14 控件工具箱的默认工具

技巧1-9 往工作表里添加控件

你可以使用窗体工具栏往工作表里添加控件（参见图1-13），也可以使用控件工具箱（参见图1-14）。两种工具栏都可以通过视图选择工具栏选项来获得。

窗体里的控件和Excel的早期版本（5.0，7.0和97）兼容，并且可以用在图表，老的XLM宏表和所有你想通过点击控件来运行宏的工作表里。

控件工具箱里的控件就是人们熟知的ActiveX控件。你可以将ActiveX控件放在工作表或者你用VB编辑器创建的窗体上。然而，窗体工具栏上的控件只对点击（Click）事件反应，ActiveX控件则有许多行为，或者说事件，发生于你使用它的时候。

当你使用窗体控件时，你给它指定宏。这个宏时储存在本工作表，新工作表或者个人宏工作簿的一个模块里。当你使用ActiveX控件时，书写的宏代码时储存在控件本身的。

19 保存宏

在这章中，你创建的宏WhatsInACell位于一个Excel工作表中。你需要保存这个开启了的工作表来保存这个宏。我建议你将其保存为Chap01.xls。保存后，关闭它，然后打开一个新工作表。注意，你工具栏上的自定义按钮还在那儿，正如工具菜单里的Contents of Cells样还在那儿一。在你使用这些工具运行宏之前，请在单元格A1里输入“Addition”，A2里输入数字2，A3里输入数字4，已经A4里输入“=SUM(A2:A3)”。当你运行这个宏时，Excel会打开适当的工作表并且执行这个指定给自定义工具的过程。

20 打印宏

如果你要将你的宏归档起来，或者在你离开电脑的时候研究宏代码，你就需要打印宏。你可以打印你储存宏的整个模块，也可以打印选择的行。

打印含有宏的整个模块：

1. 将光标放在模块的任意地方
2. 选择“文件”－“打印”
3. 在打印－VBA对话框，选择“当前模块”
4. 点击确定打印模块

打印选中的文本：

1. 在模块里，选择你要打印的文本
2. 选择“文件”－“打印”
3. 在打印－VBA对话框，选择“选择”
4. 点击确定打印选中的文本

21 保存宏在个人宏工作簿

当你录制宏时，可以将它保存在个人宏工作簿里面。当你储存宏在个人宏工作簿里时，Excel创建一个名为“Personal.xls”的文件并且放在“Program Files\Microsoft Office\Office”的子文件夹——XLStart文件夹里。保存在XLStart文件夹的文件每次在Excel启动的时候都会自动打开。个人宏工作簿是一个保存通用宏代码的方便的地方，就像下面这个宏。现在来录制一个通用的宏“FormulasOnOff”。这个宏的目的是设置是否显示工作簿的公式。

1. 选择“工具”－“宏”－“录制新宏”
2. 在录制宏对话框，输入宏名“FormulasOnOff”
3. 在保存宏的下拉菜单里选择“个人宏工作簿”
4. 点击快捷键文本框，并且按下“Shift+F”
5. 选择确定退出录制宏对话框
6. 按下“Ctrl+~”打开公式的显示，或者选择“工具”－“选项”并且点击“视图”页上“窗口选项”中的“公式”检验盒。当你打开公式显示时，工作簿单元格里显示的是公式，而非这个公式计算出来的数值。如果你是在一个空白工作表中录制这个宏的，那么你将注意到的唯一变化是工作表的列宽。
7. 点击“停止录制”，或者选择“工具”－“宏”－“停止录制”
8. 查看代码：按下Alt+F11，或者选择“工具”－“宏”－“VB编辑器”

这时，VB编辑器屏幕上的工程窗口里显示了一个多出来的VBA工程（Personal.xls）。点击这个过程名左边的加号来打开这个工程。这个VBA工程包含两个文件夹：Excel对象和模块。点击模块文件夹的加号来打开它，然后双击模块1。这时代码窗口显示了宏FormulasOnOff的内容（参见图1-15）。每个Excel工作表只有一个工程。你第一次录制宏的时候，Excel创建一个模块文件夹，并且将你的代码储存在模块1里面。如果你在相同的工作表里录制另一个宏，Excel将其放在前一个录制的宏的同一个模块1的下面。相同工作时间录制的所有的宏都储存在相同的模块里面。但是，如果你关闭Excel，然后再重新这个工作簿，Excel就会将它储存在一个新的模块。

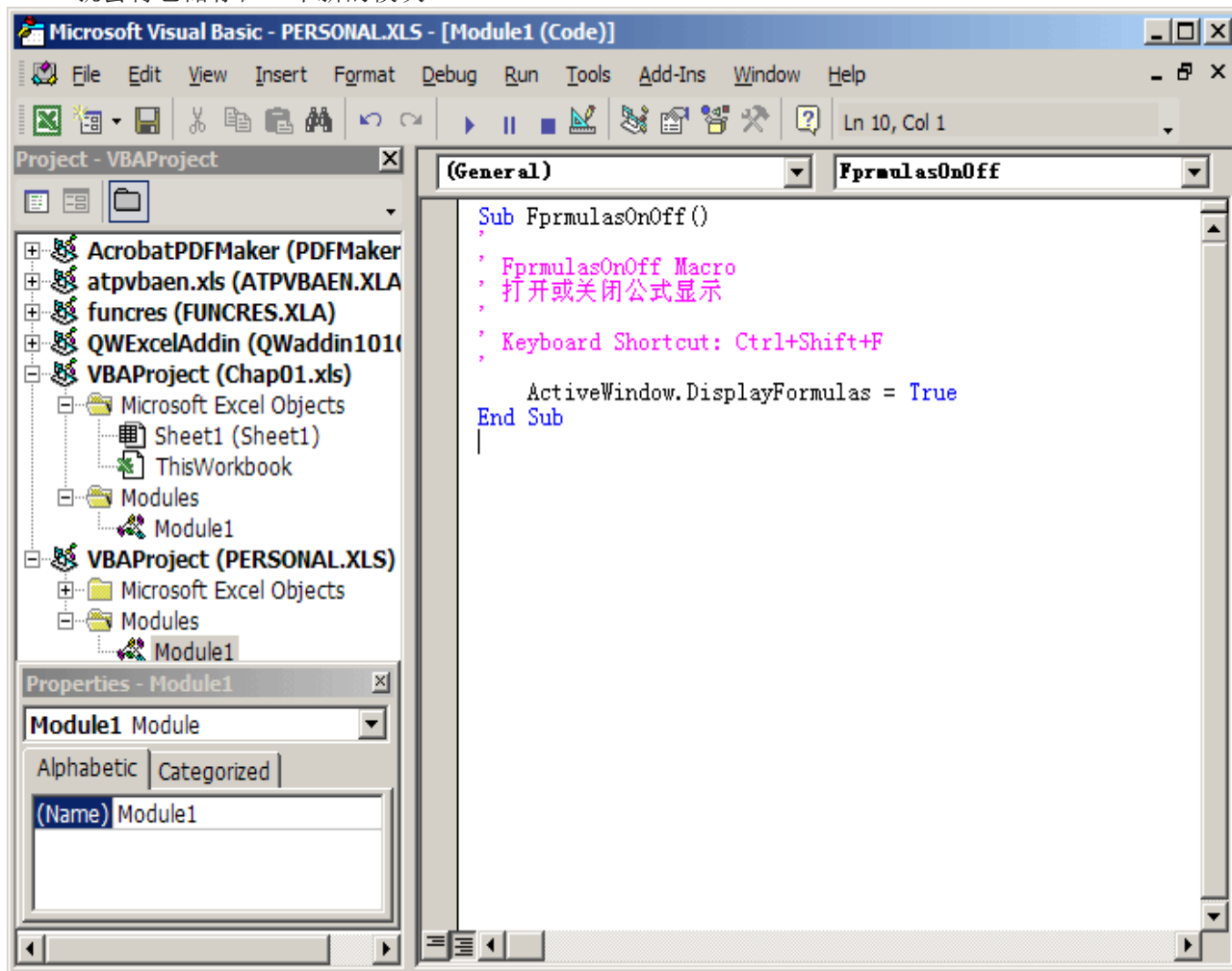


图1-15 在工程浏览器窗口，你可以选择你需要的工程

录制宏的时候，你打开了公式的显示。这个宏的名称表明可以切换公式显示的开和关。你必须修改代码才能确保它按照这种方式运行。

录制的宏设置当前窗口显示公式为真：

```
ActiveWindow.DisplayFormulas = True
```

设置为“False”将关闭公式的显示：

```
ActiveWindow.DisplayFormulas = False
```

为了在VBA里设置转换，你需要按照下面的方法来连接两语句：

```
ActiveWindow.DisplayFormulas = Not ActiveWindow.DisplayFormulas
```

用上面的语句代替你录制的代码，并且运行这个宏。无论你运行多少次，这个宏总是知道做什么。你可以使用相同的思路来创建代码以切换格式线或其它Excel特点的显示与否。当你关闭Excel时，它会提示你保存个人宏工作簿的变化，点击确定以保存变化。当你重启Excel，个人宏工作簿会在后台自动开启。

如果你想要在个人宏工作簿里保存其它的宏，你可以选择下列方法中的一个：

- 录制一个新宏，并且选择个人宏工作簿来储存
- 切换到VB编辑器，打开你要移动到个人宏工作簿里去的宏，剪切这个宏，并且打开个人宏工作簿。将宏粘贴到已经存在的模块中，或者创建一个新模块再粘贴
- 选择“文件”－“导入文件”……从另外一个VB工程 (*.frm, *.bas, *.cls) 导入宏代码

22 打开含有宏的工作簿

无论何时你打开一个含有宏的工作簿，Excel显示一个警告信息，如图1-16。为了避免显示这个警告信息，你可以通过安全对话框关闭病毒保护（参见图1-17）。

当病毒信息出现时，你可以选择：

- 取消宏——当你打开一个来源不熟悉的含有宏的工作簿，例如因特网，电子邮件，为了保护你的电脑不被宏病毒破坏，你应该选择“取消宏”。工作簿打开时不会运行它里面的任何宏。如果没有密码保护的话，你就可以切换到VB编辑窗口查看代码。查看代码后（译者：如果代码安全），你可以关闭该工作簿，然后重新打开它并且启用宏。

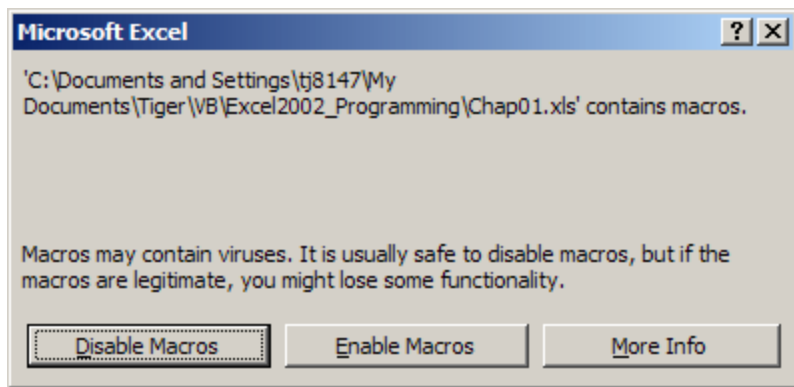
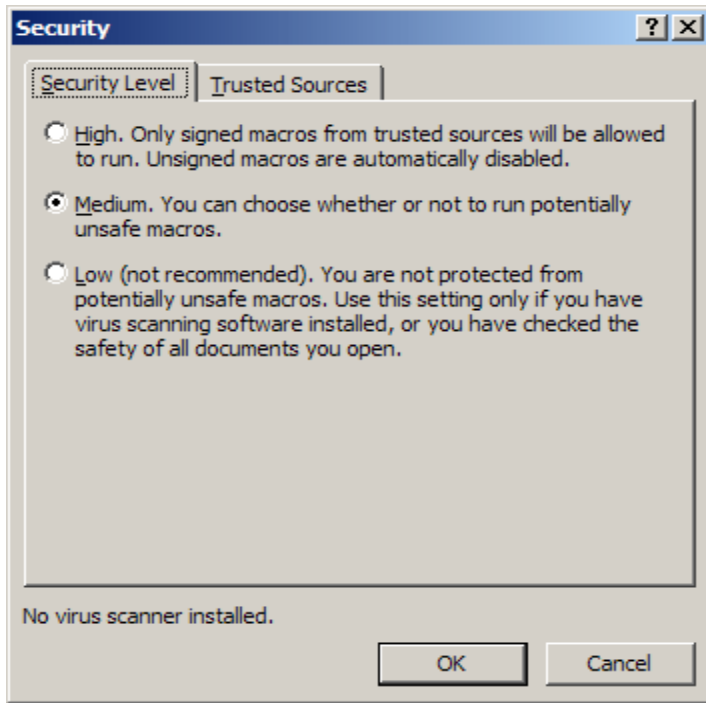


图1-16 如果你打开了病毒保护，当工作簿含有宏时，Excel 会弹出一个警告信息

- 启用宏——你如果指定这个工作簿来自于一个可靠的来源，也含有有用的宏，点击启用宏按钮。
- 更多信息——在你决定取消或者启用宏时，如果你需要了解更多的信息，那么点击这个按钮。Excel 2002 有一个有用的功能让你自动取消所有没有签名并且来源不明的宏。选择“工具”－“宏”－“安全”进入这个功能。

当你创建一个需要给别人使用的宏时，你可以使用VB编辑器工具菜单里的数字签名来确认这个宏不会带来病毒。宏的数字签名正如在纸上的签名。请在Excel在线帮助里搜索如何安装和创建你自己的数字签名。输入“数字签名”就可以获得相关主题。



图一17 选择中间的选项，让你根据工作簿决定是否取消或者启用宏

23VB 编辑窗口

现在，你已经知道如何录制，运行和修改宏了，让我们花些时间来熟悉VB编辑器的一些特点。使用VB编辑器上的工具，你能够：

- 编写你自己的宏过程
- 创建自定义窗体
- 查看和修改对象属性
- 测试VBA过程和定位错误

有两种方法进入VB编辑器：

- 从Excel界面的工具菜单：选择“工具”－“宏”－“宏编辑器”
- 从键盘：按下Alt+F11
- （译者：在工作表标签上单击右键，然后选择查看代码）

24 了解工程浏览窗口

工程窗口显示当前打开的工程和它的组成部分清单。VBA工程包括下列组成：

- 工作表
- 图表
- 当前工作簿——工程储存的工作簿
- 模块
- 类模块——特殊的模块让你可以创建自己的对象
- 窗体
- 引用到其它工程

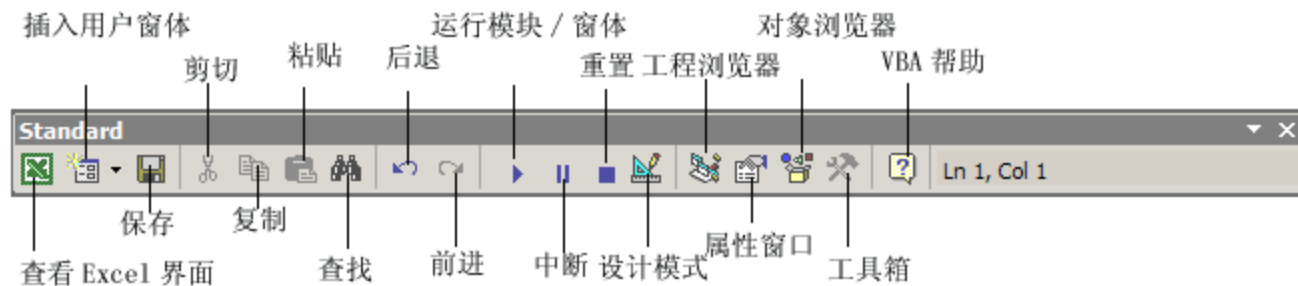
通过工程浏览器，你可以管理你的工程，容易地在当前打开的工程中切换。

你可以通过三种途径激活工程浏览器：

- 通过视图菜单，选择工程浏览器
- 通过键盘，按下Ctrl+R
- 通过工具栏，点击工程浏览器按钮（参见图1－18）

工程浏览器有三个按钮。左边第一个按钮（查看代码）显示当前选中的模块（译者：或者窗体）里的代码窗口。中间那个按

钮（查看对象）显示Excel界面当前工作表，或者窗体文件夹里面的窗体。右边的按钮（切换文件夹）隐藏或者显示工程浏览器里的文件夹。



图一18 标准工具栏上的按钮提供了快速的方式进入许多VB特征

25 了解属性窗口

属性窗口让你查看你工程里的对象和设置它们的属性。当前选中的对象的名称就显示在属性窗口的标题栏下面的对象栏。对象的属性可以按照字母顺序查看，也可以按类别查看（参见图1—19）。

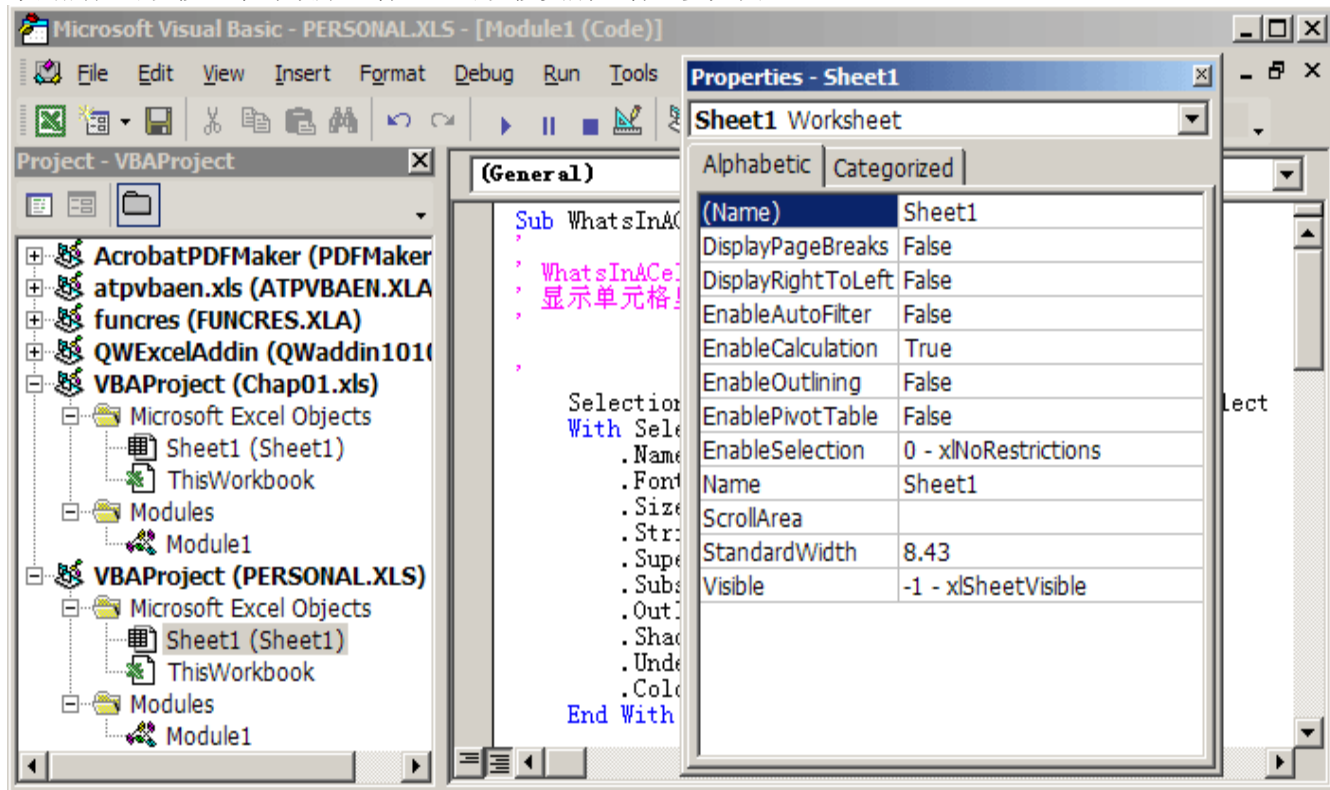


图1—19 属性窗口显示的是当前被选中的对象的属性设置

- 字母顺序——按字母顺序地列出被选择的对象的所有属性。通过选择属性名，并且输入或者选择新的设置，来更改属性设置。
- 类别——按类别列出被选中的对象的所有属性。你可以将清单折叠起来，查看类别，你也可以展开类别查看属性。类别名称左边的加号（+）说明这个类别可以展开。减号（-）说明这个类别已经展开。

有三种方式可以进入属性窗口：

- 视图菜单，选择属性窗口
- 从键盘，按下F4
- 从工具栏，点击属性窗口按钮（参见图1—18）

26 了解代码窗口

代码窗口是用来VB编程的，也是用来查看，修改录制的宏代码和现存的VBA工程的。每个模块会以一个专门的窗口打开。有好几个方法可以激活代码窗口：

- 从工程浏览器窗口，选择你要的用户窗体或者模块，然后点击查看代码按钮
- 从菜单，选择“视图”－“代码”
- 从键盘，按下F7

在代码窗口的上面，有两个下拉清单列表（参见图1-20），方便你快速地移动到任意代码处。在代码窗口左上角的对象列表框，你可以选择你想查看代码的对象。你可以在代码窗口右上角的列表框里选择一个过程或者事件过程查看代码。当你打开这个列表框，这个模块里的所有过程名按字母顺序排列在那儿。如果你选择了一个过程，光标就会跳到那个过程的第一行处。

将列分工具条（参见图1-20）拖曳下列，你就可以将代码窗口分为两半了（参见图1-21）。

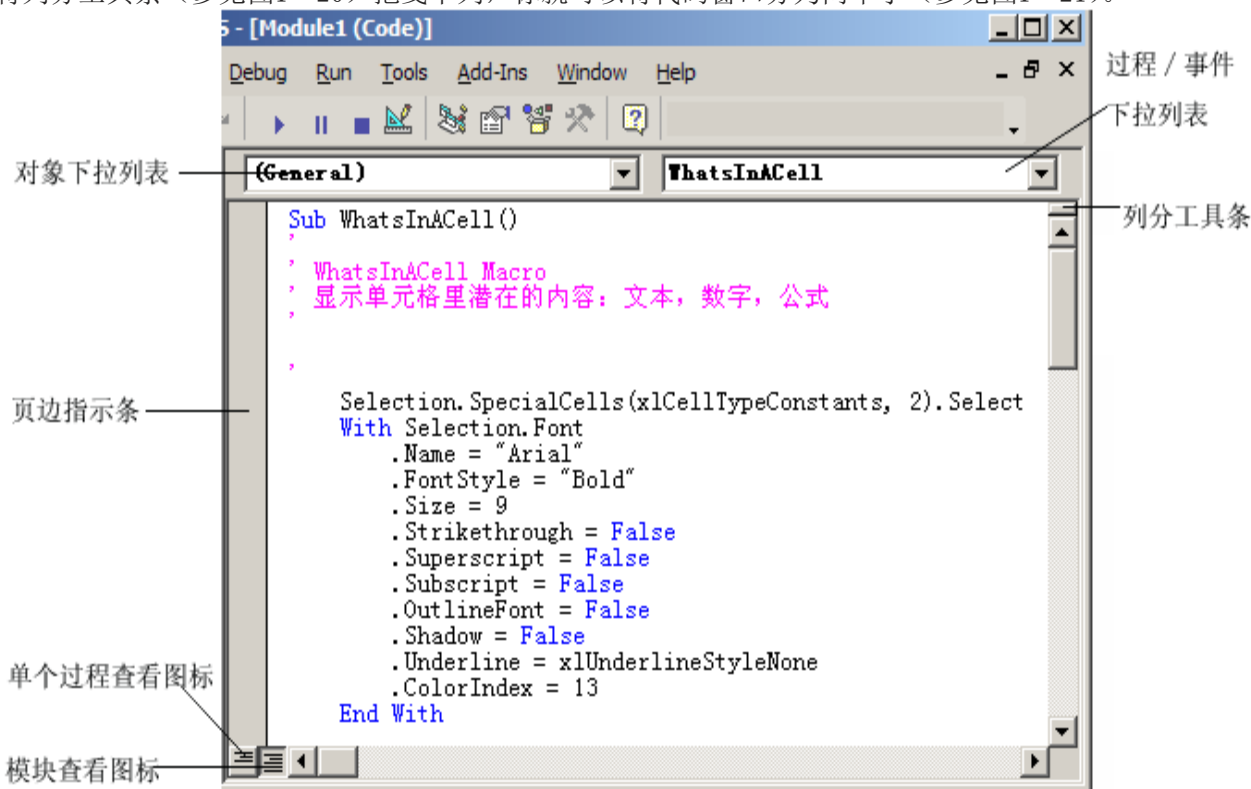


图1-20 代码窗口有几个部分，使得定位过程和查看代码变得很轻松

你可以查看不同的代码部分了。这样设置代码窗口，目的是方便在同一个模块内的过程里复制或剪切，并且粘贴代码片断。只要简单地将列分工具条拖曳至代码窗口上面就行。代码窗口的底部有两个图标。点击“过程查看”图标，代码窗口里一次只显示一个过程，可以通过过程/事件列表框选择另外的过程。点击“全部模块查看”则可以显示这个模块里的所有过程。使用竖向滚动条可以在代码中滚动。

页边指示工具条是在修改代码和调试是提供一些帮助指示的。如果你想快速查看关于指示的信息，请看第十三章。

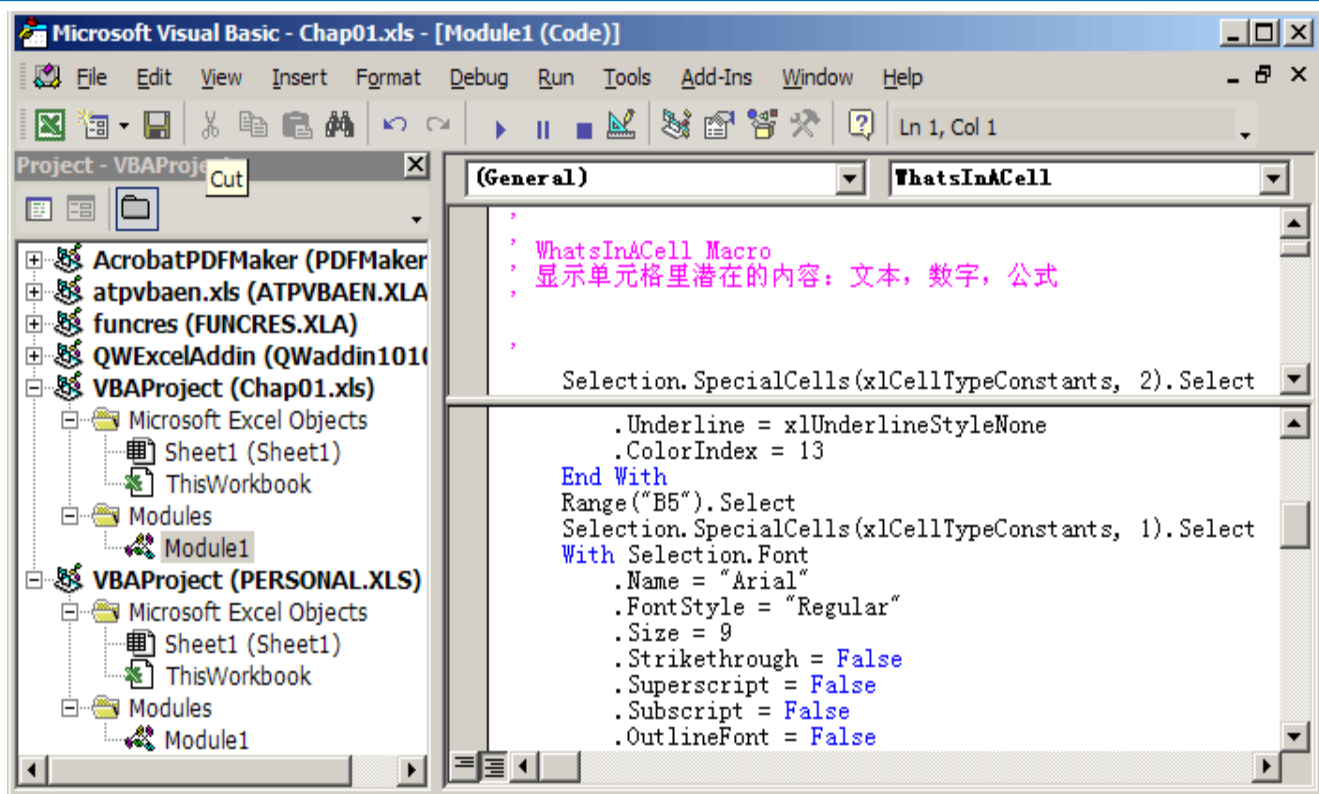


图1-21 你可以将代码窗口列分为两个窗口来查看长过程

27 VB 编辑器里的其它窗口

除了代码窗口，VB环境下还有很多其它窗口频繁地被使用：

窗体窗口用来创建自定义对话框和用户窗体。

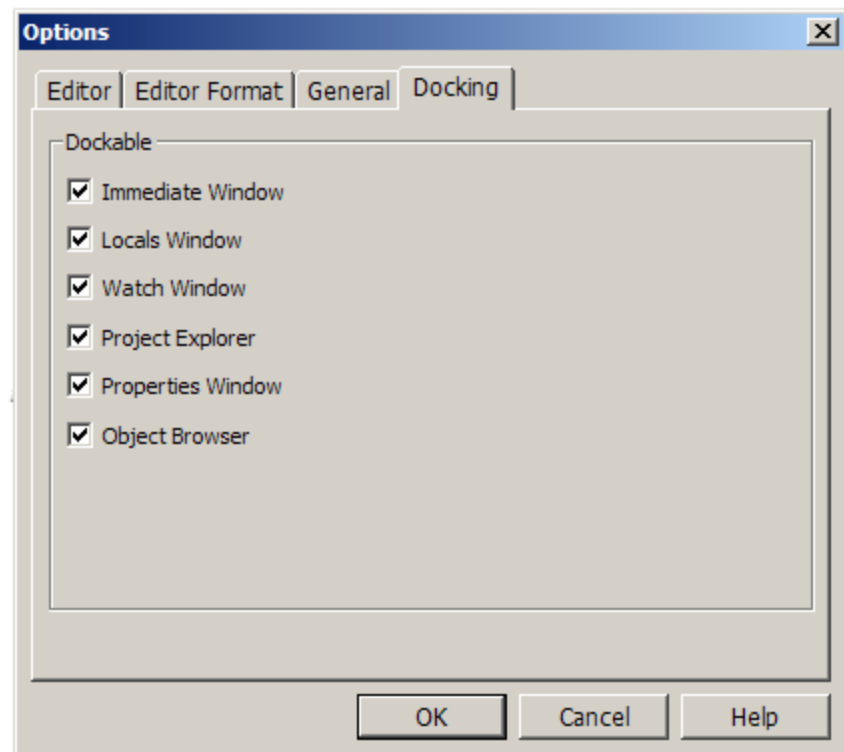


图1-22 在选项里，你可以选择显示哪些窗口

你将在第十章里学习到更多的这类知识。

图1-22显示了一些可以显示了VB编辑器里的窗口清单。你将在第二章里（对象浏览器，立即窗口）和在第三章里（当地窗口，观察窗口）学习如何使用这些窗口。

28 接下来.....

通过一些电子表格自动化的例子，你不但已经如何录制宏，而且也已经学习了如何查看，阅读和修改VB代码。还有，你也尝试了用许多方法运行。最后，你还快速过了一遍VB编辑器窗口。

在下一章，我们将会给你介绍VBA基础，你将学习许多新术语，更重要的是，你将获得有用的VBA词汇表，这将帮你将更多的任务交给Excel帮你完成。

第二章 VBA 第一步

作者: Julitta Korol 翻译: Tiger Chen Nov 28' 2004

语言学习是一个长期的活动，在此过程中，你会经历不同的阶段，由生疏到熟悉。学习VBA编程也是一样的，没有捷径。要想精通VBA，你必须从一个初级阶段起步（第二至四章）。只有当你对VBA后面的一些基本的东西有了很好的理解之后，才能提高到中级阶段（第五至七章）和高级阶段（第八至十七章）。但是，重要的事情先来。在你能够用VBA自由自在地控制Excel之前，你需要获得一些你的术语和语法。在VB里，如何表达这些？“在工作簿里添加一个新工作表”，“删除单元格A5的内容”，“将单元格A1的公式复制到单元格B1”？你也许知道这些单个的词语，然而，你怎么知道如何将它们正确地组合在一起，Excel才能执行这些任务？你将在本章中学习VBA的术语和规则。

1 了解指令，模块和过程

在第一章，你学习到了Excel宏录制器创建的一系列指令是和你实际进行的操作完全等同的。这些指令自动地放在工作簿里一个叫做“模块”的表里。Excel将模块储存在模块文件夹里，这个文件夹在当前工作簿，新工作簿或者个人宏工作簿里面。你必须激活VB编辑器窗口，并且双击工程浏览器里的模块文件夹才能查看到这些模块。当模块表在代码窗口里打开了后，你才能最后分析你的过程代码。

所有录制的指令都包括在“过程”里面。过程里面的每一行都是一个“指令”。指令的类型有很多中，例如，关键词，运算符，或者其它过程的调用。“关键词”代表VB中的一个特殊的意义。在第一章中，你学习了最常见的VBA关键词——Sub 和End Sub，它们表示一个过程的开始和结束。关键词默认地显示为蓝色。你不要将这些术语做其它的目的，因为关键词已经被VB保护了。除了关键词，VB指令里还可以有运算符。运算符有四种类型：算术运算，字符串连接，逻辑运算和比较运算。“运算符”允许你将某些值结合起来。例如，减号运算符 (/) 可以用来计算总数的百分比。本书中，你有很多机会看到如何在VBA过程中使用运算符。

VB指令的另外一种类型是过程调用。过程调用让你快速地跳到其它过程并且执行其它指令。是不是很难想象？让我们看一下你在第一章中录制的宏WhatsInACell。假设你也需要包含同一模块中宏FormulasOnOff中的一些语句。怎么做呢？你可以复制需要的代码行，再粘贴过去。然而，有一种更简单快速的方法。你可以调用这个过程，而不需要在两个过程中复制。例如，你想VB在遇到指令MsgBox “所有操作都已完成”之前执行宏FormulasOnOff里面的指令，只要添加下面一句代码就行：

FormulasOnOff

当VB到达这一行，它就会立即跳到FormulasOnOff过程并且执行它的代码。之后，它会回到宏WhatsInACell去执行剩下的代码，遇到关键词End Sub时则停止。

在你尝试这个例子之前，你必须学会如何给VBA过程和模块命名，已经如何调用不同工程里的过程。

2 VBA 工程命名

工程是一套Excel对象，模块，窗体和引用。除了VBAProject这个位于工程浏览器中工作簿名称之前默认名称，每个工程需要一个独特的名称。我们来给VBAProject (Chap01.xls) 和 VBAProject (Personal.xls) 命名：

1. 启动Excel，打开Chap01.xls，这里储存了宏WhatsInACell代码。你录制了宏FormulasOnOff的个人宏工作簿会自动开启
2. 切换到VB编辑器窗口

3. 选择VBAProject (Chap01.xls)
4. 双击属性窗口里的名称属性，这个操作选中了默认的工程名称VBAProject
5. 输入“FirstSteps”作为该VBA工程的名称，回车。注意，工程浏览器现在显示的是名称是FirstSteps (Chap01.xls)
6. 在工程浏览窗口选择VBAProject (Personal.xls)
7. 双击属性里的名称属性
8. 输入“Personal”作为它的名称，回车

技巧2—1 避免名称冲突

为了避免VBA工程之间的命名冲突，请给你的工程独特的名称。你可以使用下述方法之一来更改工程名称：

- 在工程浏览器窗口，选择工程名称，双击属性窗口里的名称属性，再输入新的名称
- 在工程浏览器窗口，在工程名称上单击右键，并且选择“工程名称属性”。出现如图2—1显示的工程属性对话框，在工程名称文本框里面输入新的名称



图2—1 工程属性窗口可以用来更改当前被选中的工程名称和描述

3 模块重命名

当你录制宏或者创建新的过程时，VB会创建一个模块文件夹来储存你的VBA 代码。第一个文件夹叫“模块1”，第二个叫“模块2”，等等。你打开一个新的工作簿并且创建VBA工程时，新VBA工程里的模块文件夹又会命名为“模块1”，“模块2”，等等。模块拥有相同的名称不但对你，而且对VB造成很大混淆，因为，它要在一个打开许多工程的环境中执行你的宏或工程。为了避免模块混淆，给FirstSteps (Chap01.xls) 工程和 Personal (Personal.xls) 工程里的“模块1”重新命名：

1. 在工程浏览器窗口，选择FirstSteps (Chap01.xls)工程，并且选择“模块1”
2. 双击属性窗口里的名称属性，这个动作选中了模块的默认名称“模块1”
3. 输入“WorksheetFormatting”作为模块1的名称，并且回车。注意，工程浏览器窗口现在显示的模块名称是“WorksheetFormatting”
4. 在工程浏览器窗口选择Personal (Personal.xls)
5. 双击属性窗口里的名称属性
6. 输入“Switches”作为模块1的名称，回车



图2-2 工程浏览器窗口显示通过属性窗口给工程和模块设置的独特名称

4 从其它工程调用过程

你只要明确过程名称，就可以调用这个在同一个工程里任何模块里的过程。假设过程FormulasOnOff和宏WhatsInACell在同一个工程里的不同模块（或者同一个模块），在WhatsInACell宏里面调用过程FormulasOnOff，你所要做的所有工作只是明确过程名称，示例如下：

```
Sub WhatsInACell()  
<这里是你录制的指令>  
FormulasOnOff  
End Sub
```

然而，如果两个或者两个以上的模块含有这个相同的过程名称，你除了要明确过程名称外，还必须包括模块名称。假设工程FirstSteps (Chap01.xls)有三个模块。模块FormulaFormatting包含宏WhatsInACell，但是，模块Switches和模块Formulas都含有一个叫FormulasOnOff的宏。如何在WhatsInACell调用FormulasOnOff（模块Switches里面的）？请看下面例子：

```
Sub WhatsInACell()  
<这里是你录制的指令>  
Switches.FormulasOnOff  
End Sub
```

要调用其它工程里的过程，你必须建立对该工程的引用。你可以在“引用”对话框进行这些操作。因为FormulasOnOff在Personal (Personal.xls)工程里，在你能够从WhatsInACell调用它之前，你需要添加对“Personal”的引用。下面是几种建立引用的方法：

1. 在工程浏览器窗口，点击FirstSteps (Chap01.xls)
2. 选择“工具”－“引用”
3. 在引用对话框，选中“Personal”旁边的勾选框（参见图2-3），然后点击确定（译者：在截图前，我并没有保存Personal，所以在附图里没有Personal一行，如果你依照书中一步一步走下来，应该没有问题）

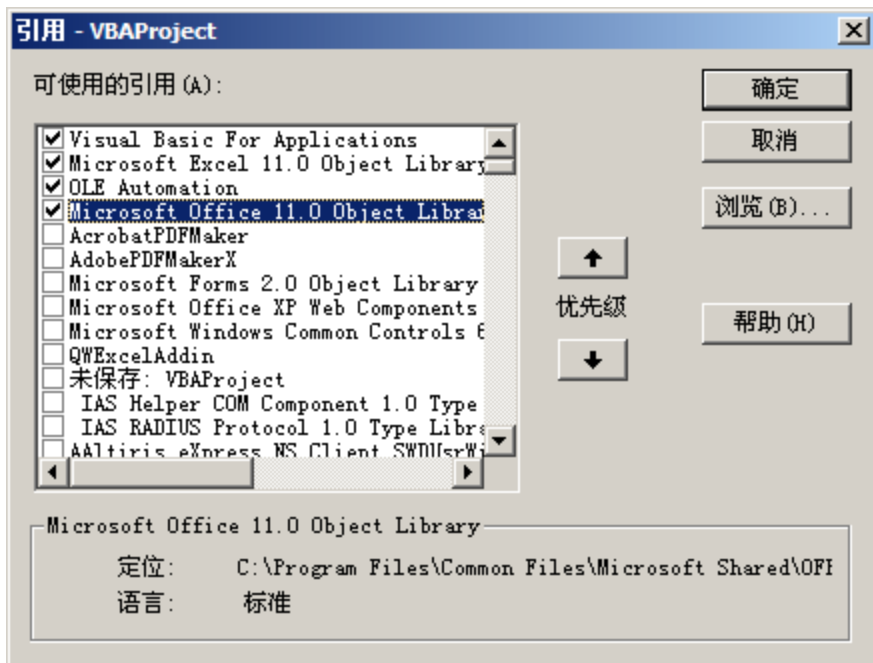


图2-3 引用对话框列出了所有这个工程可以引用的工程。如果你想要执行其它工程里的过程，你就必须建立对这个工程的引用

既然对“Personal”工程的引用已经建立了，我们就来从WhatsInACell里调用FormulasOnOff吧。

1. 在工程浏览器窗口，选择FirstSteps (Chap01.xls)并且定位到含有WhatsInACell的模块
2. 在MsgBox “所有操作都已完成”之前增加一空行，并且输入代码：FormulasOnOff
3. 返回到Excel界面，确保当前工作表是这个例子数据（参见第一章的图1-1）
4. 使用任何你在第一章里学到的方法来运行宏WhatsInACell

如果你给两个不同工程里的不同过程以相同的名称，那么你必须明确工程名称才能调用它。

我们假设FirstSteps (Chap01.xls)工程和Personal (Personal.xls) 工程里都有叫FormulasOnOff的宏，要调用Personal (Personal.xls)工程里的FormulasOnOff（记住，你已经必须先建立对Personal的引用），必须包括工程名称：

```
Sub WhatsInACell ( )
    <这里是录制的指令>
    Personal.Switches.FormulasOnOff
End Sub
```

技巧2-2 VB如何定位被调用的过程

当你调用一个过程，VB先在主调方（WhatsInACell）的同一个模块里查找。如果没有找到被调过程（FormulasOnOff），VB就会在同一个工程的其它模块里查找。如果还是找不到，VB则会检查对其它工程的引用。

技巧2-3 工程名称不在引用对话框

如果你想要调用一个当前关闭的工程里的过程，当你打开引用对话框试图建立引用时，这个过程名称不在清单中。点击“浏览”，并且打开被调用过程所在的文件夹。添加引用的对话框默认地列出数据库文件 (*.olb, .tlb, .dll)。从文件类型的下拉清单中选择Excel文件 (*.xls, *.xla)，选择并打开含有你要调用过程的文件。这个工程的名称将会加在引用对话框的最后一行。

5 了解对象，属性和方法

使用VBA，你可以创建工程控制Excel的许多东西，你同样也可以控制很多其它的应用程序。VB的伟大来自于它的控制和管理各种各样的对象的能力。但是，“对象”是什么呢？“对象”是你通过VBA控制的东西。工作簿，工作表，工作表里的单元格区域，图表或者工具条，这些只是一些用Excel时想要控制的东西的举例。这些东西就是对象。Excel含有超出一百种你可以

通过不同方式操作的对象。所有的VB对象都被分层归类。一些对象本身又可能含有其它的对象，例如，Excel时一个应用对象，这个应用对象包含其它对象，例如工作簿或者命令条。工作簿对象可能包含其它对象，如工作表或者图表。你将在本章中学习如何控制以下Excel对象：区域，窗口，工作表，工作簿和应用。我将“区域”列在了第一位置，有一个非常重要的原因，如果你不知道如何操作单元格区域的话，你基本上不能用电子表格来做什么。

某些对象看上去相似。如果你打开一个新工作簿，检查它的工作表，你不会发现什么不同。一组相似的对象被称为“集合”。例如，工作表的集合包含所有具体工作簿中的工作表；命令条的集合包含所有的工具条和菜单。集合同样是对象。Excel中使用得最频繁的集合是表（Sheets）集合，它代表所有的工作表和图表，还有工作簿集合，工作表集合以及窗口集合。当你使用集合时，相同的动作可以在这个集合中所有的对象上执行。

每一种对象都有一些特征供你描述。在VB里，这些对象的特征被称为“属性”。例如，工作簿对象有名称属性；区域对象有列，字体，公式，名称，行，样式和值等属性。这些对象属性是可以设置的。你通过设置对象的属性控制对象的外观和位置。对象属性一次只能设置为一个特定的值。例如，当前工作簿不可能同时有两个不同的名称。VB中最难理解的部分是有些属性同时又可以是对象。想想区域（Range）对象，你可以通过设置字体颜色来改变选定单元格的外观。但是，字体（Font）可以有不同的名称（Times New Roman, Arial, ...），不同的字号（10, 12, 14, ...）和不同的样式（粗体，斜体，下划线，...）。这些是字体的属性。如果字体有属性，那么字体也是对象。

属性真是了不起，让你改变对象的外观，但是，如何控制这些操作呢？你在使Excel为你执行任务之前，你需要知道另外一个术语。对象有方法。每一种你想要对象做的操作都被称为“方法”。最重要的VB方法是Add方法。你可以使用这个方法添加一个新工作簿或者工作表。对象可以使用不同的方法。例如，区域（Range）对象有专门的方法让你清除单元格内容（ClearContents方法），清除格式（ClearFormats方法）以及同时清除内容和格式（Clear方法）。还有让你选择，复制或移动对象的方法。方法有可选择的参数确定方法执行的具体方式。例如，工作簿（Workbook）对象有一个叫关闭（Close）的方法。你可以使用它关闭任何打开了的工作簿。如果工作簿有改动，Excel会弹出一个信息，问你是否要保存变化。你可以使用关闭方法和设定它的保存变化（SaveChanges）参数为假（False）来关闭这个工作簿并且不管它的任何变化。正如例子：

```
Workbooks("Chap01.XLS").Close SaveChanges:=False
```

6 学习对象，属性和方法

当你学习新的事物时，理论会给你必须的背景，但是，你如何真正知道那是什么呢？大多数人习惯形象思维，为了使Excel对象易于理解，VB在线帮助提供了一个对象模型，请看接下来的附图。注意，Application对象位于树型图的最上端，它实际上代表Excel本身。其它对象在较低的层次。

假设你想要控制Range对象，在你能够控制任何Excel对象之前，你必须对它创建引用。为了获得下图中的Range对象，只要遵照下面几行代码。每次看到树型图中的线指向不同的层时，你只要巧妙地将线换成一个逗点运算符（停顿，英文状态下的句号）。这样，最终你会以下面的方式到达Range对象：

```
Application.Workbook.Worksheet.Range
```

你可以使用Excel对象树型图来寻找其它对象的路径，例如窗口（Window），批注（Comment），自动筛选（AutoFilter）或者绘图区（ChartArea）。分析对象模型是一个学习Excel对象的非常好的方法。你花在这里的时间，以后你开始编写VBA过程的时候，会给你加倍的回报。通常，你需要明确你引用的对象的名称。

现在，我们来点更具体的。假设你要清除单元格A4里的内容。手动做这个时，只要选择单元格A4然后按下键盘上的Delete键就可以了。用VB做同样的操作，你首先需要知道如何使Excel选中了正确的单元格。单元格A4和其它的工作表单元格一样，是Range对象。VB没有Delete方法来清除单元格内容，取而代之的是ClearContents方法，例如：

```
Range("A4").ClearContents
```

注意在对象名称和方法之间的逗点运算符。这个指令去除单元格A4里的内容。然而，如何使Excel清除工作簿Chap02.xls第一个工作表里单元格A4的内容呢？我们仍然假设打开了好几个工作簿。

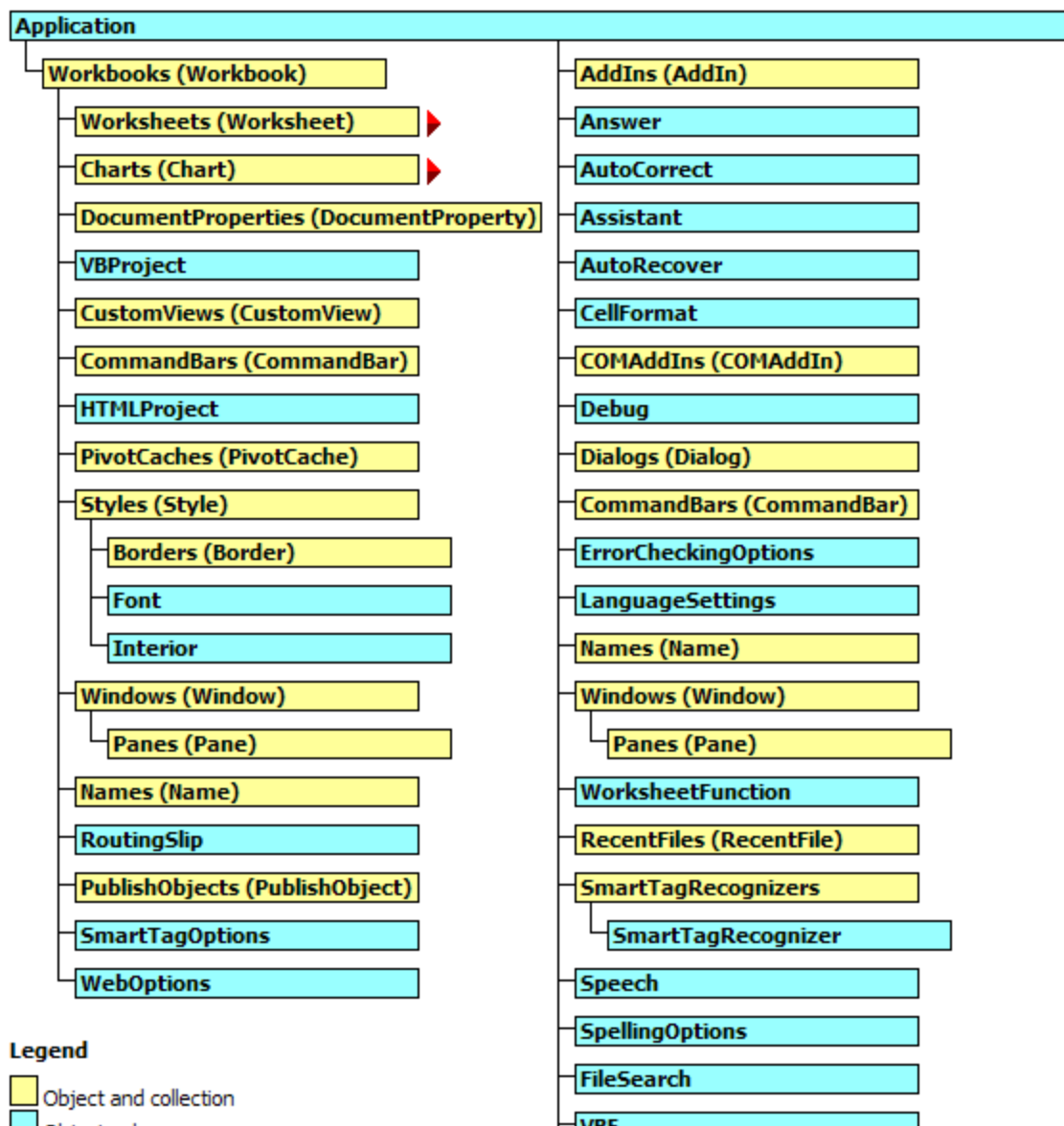


图2-4 Excel对象树型图（第一页）（译者：原书图2-5（工作簿对象树型图）已包含在内，故在此略过）

如果你不希望最后在错误的工作簿或工作表里删除了A4里的内容，那么你必须写下详细的指令，这样VB就知道在哪里找这个单元格：

```
Application.Workbooks("Chap02.xls").Worksheets("Sheet1").Range("A4").ClearContents
```

上面的指令应该写成一行，并且应该从右到左阅读：清除单元格A4里的内容，这个单元格在一个叫“Sheet1”的工作表里，而这个工作表又在一个叫“Chap02.xls”的工作簿里面，工作簿“Chap02.xls”又应该是Excel应用程序的一部分。注意，集合名称的后面带有一个字母“s”：Workbooks和Worksheets。所有引用的工作簿，工作表或单元格名称都必须用引号（译者：英文状态的引号）包括起来。

如何找到Excel对象树型图？选择Excel界面上的“帮助”——“Excel帮助”——“编程信息”——“微软Excel VB参考”——“Excel对象模型”。（译者：2002版有全局的对象模型，2003版好像没有这个。）

除了Excel对象，你还可以使用Office, Forms和DAO, ADO对象模型。属于这些数据库（library）中的对象都可以用在Excel中，也可以用在Office家族中的其它应用软件中。在十五章，你可以看到使用DAO和ADO对象，从Excel进入Access数据库的例子。

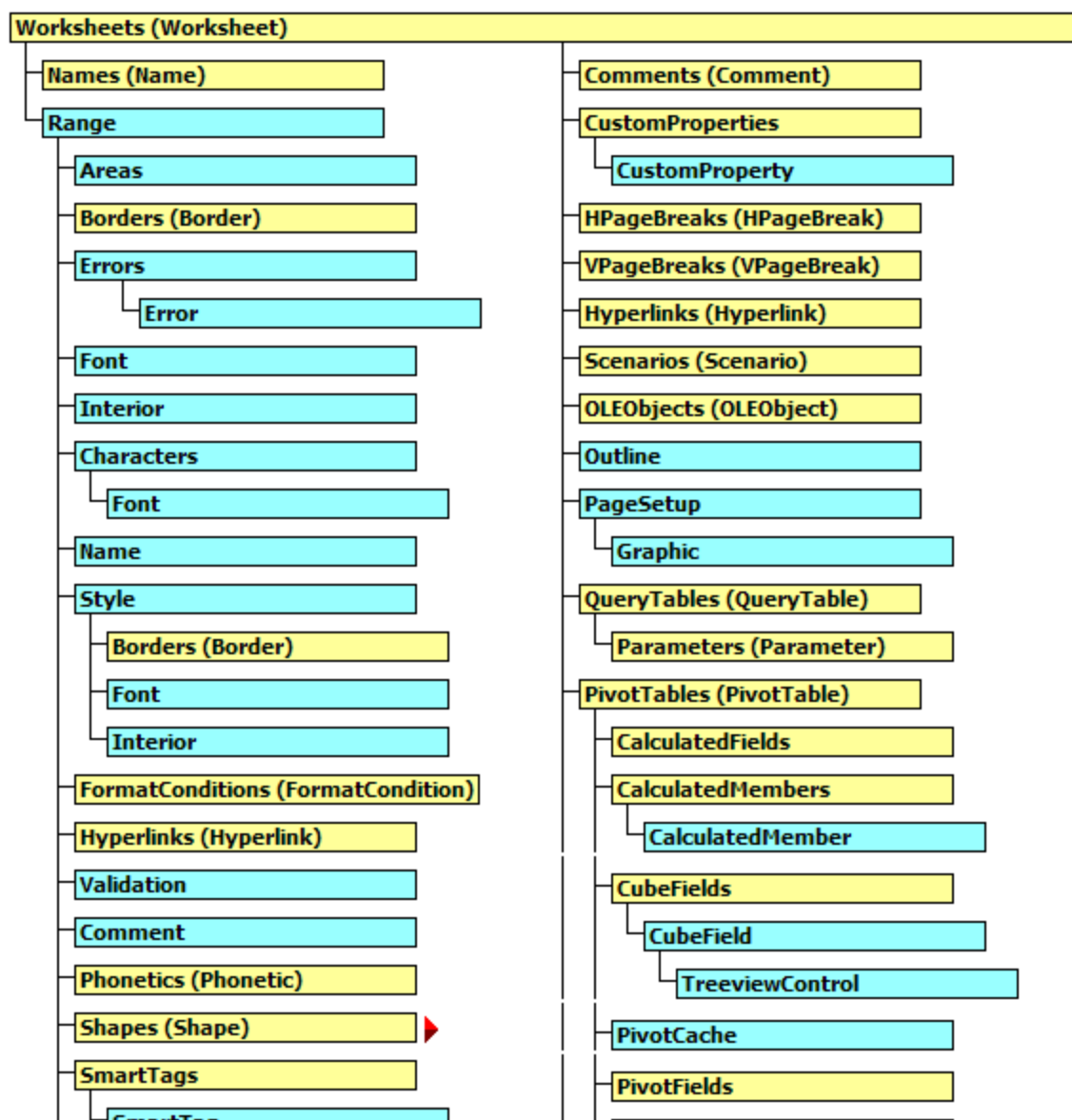


图2—6 Excel对象(Worksheet)

技巧2—4 VBA 和 Excel 的早期版本

Excel在线帮助列出了Excel对象模型从早期版本以来的变化。许多对象，属性和方法都已经被更新的，改进的特色所代替了。为了提供兼容性，被取代的对象已经被隐藏起来了（译者：它们仍然是可用的）。打开VB编辑器窗口上的在线帮助，隐藏的对象同样可以在对象浏览器里找到。如果你在对象浏览器窗口上单击右键，你可以选择显示隐藏成员的选项。你将在以后的章节中学习如何使用对象浏览器。

7 句法和文法

既然现在你已经知道了VBA的一些基本组成要素（对象，属性和方法），是时候开始使用它们了。但是，你怎么将对象，属性和方法连接成正确的语言结构呢？每种语言都有语法规则，人们必须遵循语法以确保他们被理解了。无论你说的是英语，西班牙语，法语还是其它语言，你在读，写的时候都必须遵从一定的规则。在编程中，我们使用“句法”（syntax）这个术语来更确切地明确它的语言规则。你可以在在线帮助或者在对象浏览器窗口查找每个对象，属性或方法的句法。下面列出一些你必须的VB常用规则：

- 规则1：引用对象的属性

如果这个属性没有自变量，使用下面的句法：

Object.Property

对象是一个占位符，是你放置你想要进入的实际对象名称的地方。属性同样也是一个占位符，你可以在这里放置该对象的特点。例如：指向工作表中单元格A4中输入的值，见下述指令：

Range("A4").Value

注意对象名称和属性之间的句号。当你需要进入一个存在于多个其它对象里的对象的属性时，你必须按顺序地写上所有对象的名称，并且用 句号运算符分开。例如：

ActiveSheet.Shapes(2).Line.Weight

这个例子指向当前工作表里图形（Shapes）集合里的第二个对象里的直线（Line）对象的粗细（Weight）属性。

有些属性要求一个或多个自变量。例如，使用偏移（Offset）属性，你可以选择一个和当前单元格相对位置的单元格。Offset属性需要两个自变量，第一个自变量为行号（rowOffset）第二个是列号（columnOffset）。

对象 属性 自变量

ActiveCell.Offset(3, 2)

在上面的例子中，假设当前单元格是A1，Offset(3, 2)将会指向往下3行和往右两列的单元格，也就是单元格C4。因为，在括号内的自变量总是很难理解，通常操作是将它们的名称也列上，见下例：

ActiveCell.Offset(rowOffset:=3, columnOffset:=2)

注意，自变量名称后面总是跟着一个冒号和一个等于号（:=）。如果你使用带名称的自变量，你可以任意顺序地列出它们，上面的指令也可以写成这样：

ActiveCell.Offset(columnOffset:=2, rowOffset:=3)

改后的指令没有改变意思，你仍然指向单元格C4。然而，如果你改变ActiveCell.Offset(3, 2)中自变量的次序，结果你会指向D3，而不是C4。

■ 规则2：改变对象的属性

Object.Property = Value

Value是一个新的你要赋给该对象属性的值。这个值可以是：

✧ 一个数字

Range("A4").Value = 25

上面的指令在当前工作表的单元格A4里输入数字25

✧ 在引号里的文本

ActiveCell.Font.Name = "Times New Roman"

上面的指令将当前单元格字体改为Times New Roman

✧ 逻辑值（True或False）

ActiveCell.Font.Bold = True

上面的指令设置当前单元格的字体为粗体。

■ 规则3：返回对象属性的当前值

Variable = Object.Property

Variable（变量）是VB将要储存属性设置的地方的名称，你将在第三章里学习关于变量的知识。

变量 对象 属性

CellValue = Range("A4").Value

上面的指令将当前A4单元格的值保存到变量CellValue。

■ 规则4：指向对象的方法

如果该方法没有自变量，那么句法应该是：

Object.Method

对象是一个占位符，是你放置你想要进入的实际对象名称的地方。方法同样也是一个占位符，你可以在这里放置要对该对象的进行的操作的名称。例如，可以使用下述指令来清除单元格A4的格式（译者：应该是内容）：

对象 方法

Range("A4").ClearContents

如果该方法可以使用自变量来限制，那么句法为：

Object.Method (argument1, argument2, ... argumentN)

例如，使用GoTo方法，你可以快速地选择工作表里的任何区域。GoTo方法的句法为：

Object.GoTo(Reference, Scroll) ‘对象.GoTo(参照，窗口滚动)

Reference自变量是目标单元格或者区域，Scroll自变量可以设定为真（True）让Excel窗口滚动到该目标地址出现在窗口的左上角；或者设定为假（False），窗口不滚动（译者：系统默认为False）。例如，下面的VBA语句选择工作表Sheet1里的单元格P100，并且窗口滚动：

```
Application.GoTo _
    Reference:=Worksheets("Sheet1").Range("P100"), _
    Scroll:=True
```

上面的指令没有固定在一行，使用了一条特殊的线（下划线）将它分为几段。下面的部分将讲述这个。

8 打断很长的 VBA 语句

尽管一行VBA代码最多可以包含1024个字母，但是，为了使你个过程容易阅读，最好将长的语句打断为两行甚至多行。VB使用一个专门的连续线（下划线）置于一行代码的末尾，表明下一行是这行的连续。例如：

```
Selection.PasteSpecial _
    Paste:=xlValues, _
    Operation:=xlMultiply, _
    SkipBlanks:=False, _
    Transpose:=False
```

这个连续符是下划线，你必须在下划线之后带一个空格。

你可以在下述几种情况中使用连续符：

- 运算符之前或者之后。例如：&, +, Like, NOT, AND
- 逗号之前或者之后
- 冒号和等号（:=）之前或者之后
- 等号之前或者之后

你不可在冒号和等于号之间使用连续符，例如，下面的代码VB是不认的：

```
Selection.PasteSpecial Paste: _
    =xlValues, Operation: _
    =xlMultiply, SkipBlanks: _
    =False, Transpose: _
    =False Selection.PasteSpecial Paste: _
    =xlValues, Operation: _
    =xlMultiply, SkipBlanks: _
    =False, Transpose: _
    =False
```

同样，在引号之内的文本之间加连续符也是不对的，例如，下面的下划线的使用是无效的：

```
MsgBox "To continue the long instruction, use the _
    line continuation character."
```

上面的指令应该打断为如下代码：

```
MsgBox "To continue the long instruction, use the " & _
    "line continuation character."
```

9 了解 VBA 错误

在编写或编辑VBA过程之中，无论你多小心，出错的可能性还是很打的。例如，你可以错误拼写一语句，放错了一个逗号或引号，或者忘记了一个句号或右括号。这些类型的错误称为句法错误。幸运的是，VB比较容易帮助你发现这种类型的错误。为了让VB在你输入一行代码后，自动帮你检测语法的正确性，你需要在VB窗口的“工具”—“选项”里，确保勾选了“编辑器”页上的“自动语法检测”。

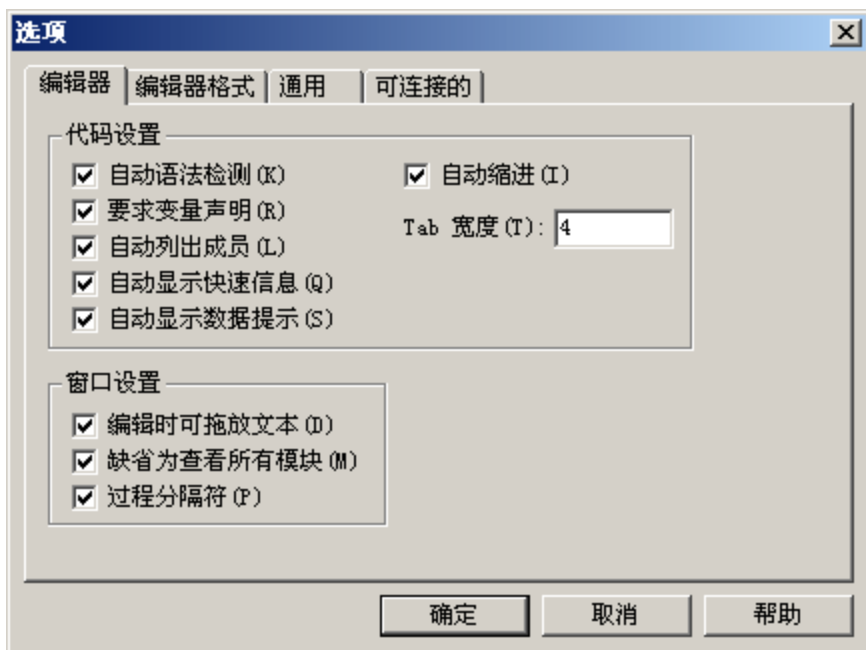


图2-7 选项对话框的编辑器上的“自动语法检测”帮你检查VBA过程里的打字错误

当VB发现语法错误时，弹出一个错误信息框，并且将有误的代码行颜色变为红色（参见图2-8）或者其它在选项对话框“编辑器格式”页设定的颜色。如果错误信息框上的解释不够清楚，你总是可以点击“帮助”按钮寻求更多的帮助。再如果VB在线帮助无法给你提供正确的方向，那么返回你的程序，仔细检查有误的那行代码是否漏掉了字母，引号，句号，冒号，等于号，左括号和右括号等。

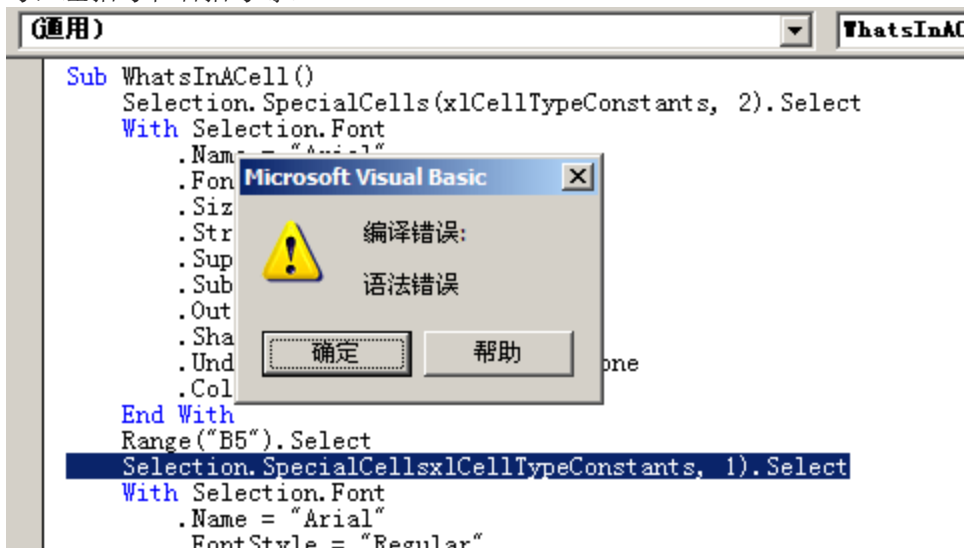


图2-8 这个错误由于漏掉了常数xlCellType前面的括号而产生



图2-9 当VB试图在工作表或单元格区域里选择一个并不存在的单元格时，就会产生一个运行时间错误

查找语法错误可能是烦人的并且费时的事情。有些语法错误只有在过程运行的时候才能被发现。在试图运行你的过程的时候，VB可能找到那种因为使用了无效的自变量，或者是漏掉了那些需要成对使用的指令如If语句和循环结构，而造成的错误。

技巧2-5：程序调试

你很可能不只一次听过“计算机程序里充满了错误”。在编程里，错误就被称为“bug”（错误，漏洞），而“调试”（debug）则是给你的程序除错的过程。调试的第一步就是改正所有的语法错误。VB提供了无数种工具，你可以使用它们来追踪和消除错误。在本章中，你将知道如何使用VB助手帮助你在编写程序时出现尽可能少的错误；在第十三章中，你将学习如何使用专门的调试工具来捕获你VBA程序里的错误。

除了语法错误外，还有其它两种错误：运行时间和逻辑。运行时间错误发生在过程运行的时候。图2-9显示了一种典型的运行时间错误。运行时间错误经常发生在那些程序员在编写代码的时候没有想到的情况。例如，当程序试图访问一个用户电脑上并不存在的驱动器或者文件，或者没有首先检查是否用户插入软盘并关闭软驱口而试图复制一个文件到软盘，这时就会发生运行时间错误。

第三种错误——逻辑错误，通常不会发出明确的错误信息。过程可能没有语法错误，甚至运行无误，然而，得到的却是错误的结果。逻辑错误通常非常难以查找，并且它藏得很隐秘，间歇发生，你不能指望花几个小时，甚至几天，就能找到错误源。

10 查找帮助

当你使用宏录制器时，你所有的操作都被翻译成VBA指令，并且放置在一个模块里。你在研究这些录制的过程时，不要忘记帮助随时可用。你会发现有些代码的意思可能会非常易懂，然而，有效却不怎么明白。这时候，你就需要寻求帮助了。当你独自工作时，只要轻轻一点或者轻轻一按就可以请教你的VBA老师了。使用VB在线帮助比使用词典或参考手册要快捷和容易得多。如果你讨厌一页一页地在词典里找你需要的术语，你将惊讶于你如何快地从VB代码窗口找到需要的帮助页面。

让我们来检查你如何通过内置的VBA老师的帮助，将WhatsInACell过程里的第一句变成你自己的VBA词汇：

```
Selection.SpecialCells(xlCellTypeConstants, 2).Select
```

上面的指令可以打断为三部分，哪些部分？Selection是对象还是属性？SpecialCells是什么？Select是什么？要回答这些问题，请依照下面的操作：

1. 激活你要分析的过程的代码窗口
2. 点击你不懂的词语
3. 按下F1



图2—10 VBA的对象，属性和方法在在线帮助里解释得很详细

帮助就会显示相应的页面。如果你的光标放在词语“Selection”中间，你就会知道Selection可以是一个应用程序的属性，也可以是一个窗口对象。如果你的光标在下一个不明白的术语（SpecialCells）并且按上面的步骤再做一遍后，你将看到SpecialCells帮助屏幕（参见图2—10）。注意，每个帮助主题包含许多信息。被查找的指令类型显示在帮助窗口的上面；指令的类型允许词语分类。例如，SpecialCells是方法，可以使用这个方法的对象名称列在“应用于”下面（译者：点击“应用于”，出现对象名称列表）。指令名称下面的“参阅”和“示例”让你快速地跳到其它应用或意义相似的指令，以及查看使用这个指令的例子。指令的意义显示在“参阅”和“示例”标题的下面。接下来则是语法和需要的自变量和其它参数。“说明”部分给出一些推荐使用该指令的情形。

你可以很容易地将示例中的代码复制到你的过程中去：选中你要复制的代码行，按下Ctrl+C，或者单击右键，选择快捷菜单上的“复制”，然后切换到VB代码窗口，点击你需要粘贴代码的地方，再按下Ctrl+V或者选择“编辑”—“粘贴”。

11 语法和编程快捷助手

VB编辑器窗口上的“编辑”工具条上有很多按钮，帮助你快速容易地输入正确格式的VBA指令。如果“编辑”工具条目前在VB编辑器窗口上不可连接，那么你可以选择“视图”—“工具条”



图2—11 编辑工具条上的按钮使得编写程序和设置指令格式变得很容易

在VB中编写过程需要你使用成百上千的内置指令和函数。因为大多数人们无法学习所有VBA里可用指令的语法，IntelliSense提供了你所需求的语法和编程帮助。当你在代码窗口编程时，经常会有专门的窗口弹出来，引导你完成正确的VBA代码。

12 属性/方法列表

每个对象都会有许多属性和方法。当你输入一个对象名称和一个句号以分开这个对象名称和它的属性或方法时，弹出一个列表菜单。这个菜单列出了在这个句号之前的对象的所有可用的属性和方法（参见图2-12）。如何打开这个自动工具？选择“工具”－“选项”，单击选项对话框上的“编辑器”页，确保勾选上“自动列出成员”。

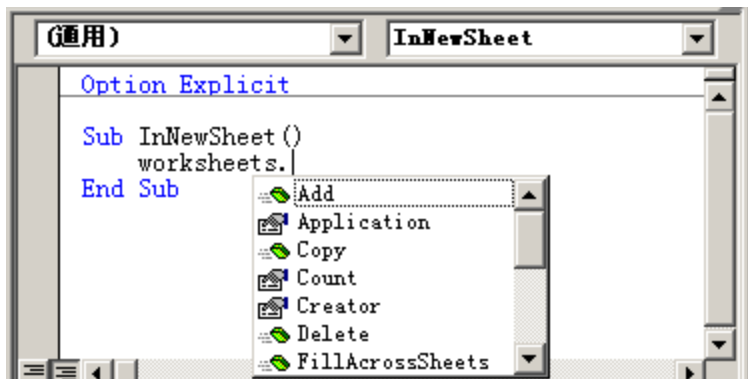


图2-12 在输入VBA指令时，VB会建议可以用于该对象的属性和方法

如何从弹出菜单（图2-12）上选择项目？试图输入你需要的属性或方法名称的前几个字母，当Excel突出显示了正确的项目名称，则回车，插入该项目到你的代码中去，并且开始新的一行；或者，你需要在同一行继续写代码，那么就按Tab键代替回车。你也可以双击该项目来插入到代码中去。只要按Esc键就可以关闭这个弹出菜单，而不插入任何项目。当你按Esc键取消了弹出菜单后，VB将不会对同样的对象显示该菜单。你可以通过以下方法来再次显示属性/方法弹出菜单：

- 按Ctrl+J
- 使用“backspace”（后退）键删除句号，然后重新输入句号
- 在代码窗口上单击右键（该对象，句号后），并且在快捷菜单上选择“属性/方法列表”
- 选择“编辑”－“属性/方法列表”
- 点击“编辑器”工具条上的“属性/方法列表”按钮

13 常数列表

本章的前面，你学习了给属性赋值，需要使用下面规则：

Object.Property = Value

如果选项对话框（编辑器页）已经勾选了“自动列出成员”，Excel就会在等号前的属性弹出一个菜单，列出该对属性有效的常数。常数是表明确切的描述或者结果的值。Excel和Office里面的其它应用软件都有很多预先定义的内置常数。你将在第三章中学习常数，常数类型和使用。

假设你需要你的程序打开Excel工作表上的分页预览。“编辑”（译者：视图）菜单上有两个选择：普通视图和分页预览。普通视图是绝大多数Excel任务的默认视图模式；分页预览则是编辑视图，显示工作表中有内容的区域。这两种选项都有相应的内置常数来表示。Excel常数起名总是以“xl”开头。你一旦在代码窗口里输入指令：

ActiveWindow.View =

就会弹出一个菜单，列出这个属性的有效常数名称。使用在上节中“属性/方法列表”弹出菜单同样的技术，也可以处理“常数列表”弹出菜单。按下Ctrl+Shift+J或者点击“编辑器”工具条上的“常数列表”按钮，可以激活常数列表菜单。

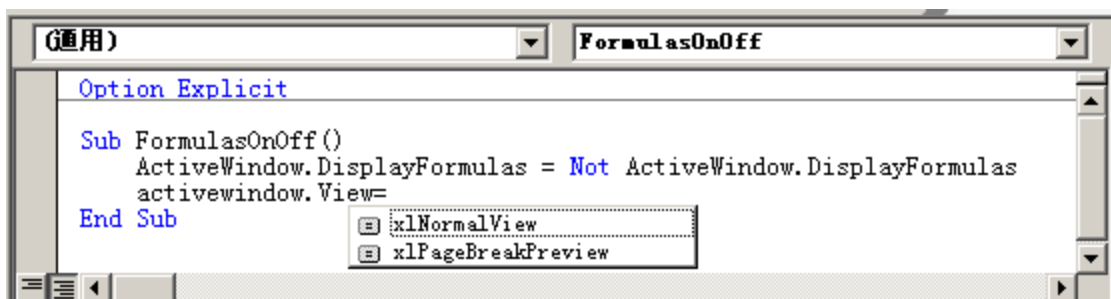


图2-13 常数列表弹出菜单显示了对敲入的属性有效的常数清单

14 参数信息

如果你有过使用Excel函数的经历，你就会知道许多函数需要一个或者多个自变量（或者参数）。如果VB函数要求自变量，你可以在输入左括号后在光标下面看到一个提示框，显示必要的或可选的自变量的名称（参见图2-14）。参数信息帮你很容易地给VBA函数设置参数。另外，它提醒你其它两件对函数运行正确至关重要的事情：自变量的顺序和自变量要求的数据类型。你将在下一章里学习数据类型。

在代码窗口里面输入下述代码，看看参数信息是如何工作的：

ActiveWorkbook.SaveAs(

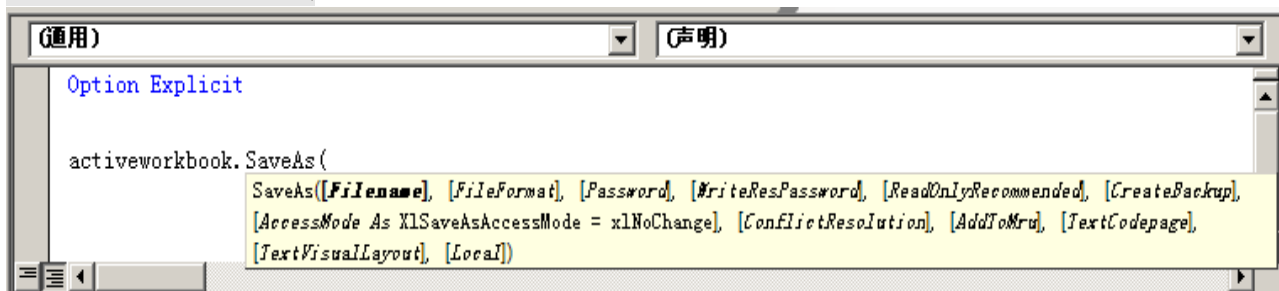


图2-14 A tip window displays a list of arguments utilized by a VBA function or instruction.

你一旦输入了开始的括号，光标的下面就会出现一个提示框，当前的自变量会显示为粗体；当你输入完第一个自变量并且输入了逗号，VB会将下一个自变量为粗体。可选的自变量会用中括号[]括起来。只要按下Esc键就可以关闭参数信息窗口。如何使用键盘来打开提示窗口？输入指令或函数，紧接着是左括号，然后按下Ctrl+Shift+I。你也可以点击编辑菜单上的参数信息按钮或者选择“编辑”-“参数信息”。

15 快速信息

当你选择了代码窗口里的指令，函数，方法，过程名称或者常数，然后点击编辑工具条上的“快速信息”按钮（或者按下Ctrl+I），VB将会显示突出显示项目的语法和常数的值。快速信息可以通过选项对话框来打开或者关闭。在编辑器页，勾选“自动显示快速信息”，打开快速信息功能。

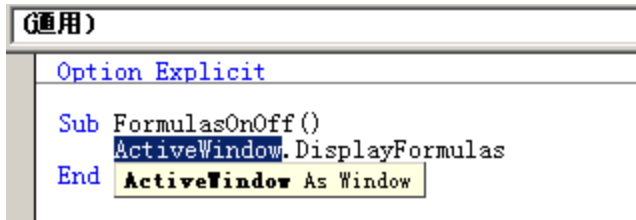


图2-15 快速信息提供函数参数清单，也可以是常数值和VBA语句语法

16 自动完成关键字

加速在代码窗口编写VBA程序的另一种方法是使用“自动完成关键字”功能。当你输入一个关键字的前几个字母，然后按下Ctrl+空格键，或者点击编辑工具条上的“自动完成关键字”按钮，VB会帮你输入这个关键字的剩余字母，节约你的时间。例如，

在代码窗口里输入关键字“Application”的前四个字母，并且按下Ctrl+空格键：

Appl

VB将会完成剩余的字母，在Appl地方，你将看到整个关键字Application。如果有好几个关键字具有相同的开头字母，当你按下Ctrl+空格键后，VB会显示一个弹出菜单，列出所有关键字。测试这个例子，可以输入关键字Application的前三个字母，按工具条上的自动完成关键字按钮，然后在弹出菜单上选取合适的关键字。

17 缩进/凸出

也许你已经看到，在选项对话框的编辑器页上有许多设置你可以打开以使用代码窗口许多可用的自动功能。如果勾选了“自动缩进”选项，你就可以自动缩进所选的代码行，缩进的量为“Tab宽度”文本框里的数字。默认的自动缩进量是4个字母，你也可以在文本框里输入一个新的数字来改变Tab宽度。你为什么需要在代码里使用缩进？缩进可以使你的代码更容易阅读和理解。特别是输入一些做决定或重复性工作的代码行时，更建议使用缩进。你将在第五和第六章中学习如何创建这种类型的VB指令。现在，我们来练习使用缩进和凸出代码行，用第一章里的宏WhatsInACell作为例子：

1. 在工程浏览器窗口，选择FirstSteps (Chap01.xls)工程，并且激活含有WhatsInACell代码的WorksheetFormatting模块
2. 选择开始为关键字With和结束为End With的一段代码
3. 点击编辑工具条上的缩进按钮，或者按键盘上的Tab键（译者：按Tab键需要将光标放在行首，而非选中指令）
4. 选中的指令会向右移动4个字母的位置，如果使用了Tab宽度的默认设置。
5. 点击编辑工具条上的“凸出”按钮，或者按Shift+Tab将选中的指令行返回原来的位置。

缩进和凸出同样可以在编辑菜单里找到。

18 设置注释块/解除注释块

在第一章中，你学习了一行代码前面加一个引号表示注释。注释不但使代码更容易理解，而且它在VBA过程的测试和处理问题中都是很有用的。例如，你执行一个过程时，它可能和期望的运行不一致，对于那些可能产生问题的代码行，你现在想略过它们，但是以后可能还需要用到它们，你就可以在它们前面加一个引号注释掉它们，而不必删除它们。对大多数人来说，需要注释掉一行代码的话，只有在它前面敲入一个引号就可以了，但是，如果要注释掉整块代码，使用“编辑”工具条上的“设置注释块”和“解除注释块”按钮则是很方便的。要注释掉几行代码，只要选中这些代码行并且点击“设置注释块”按钮。点击“解除注释块”按钮，将注释掉的代码恢复到代码里。

如果你没有选中文本，就点击了“设置注释块”按钮，只有在光标所在的代码行前面加入引号。

19 使用对象浏览器

如果你想要在VBA众多的组件和功能中自由切换，那么使用对象浏览器。这个专门的内置工具在VB编辑器窗口是可用的。使用下面任何一种方法都可以访问对象浏览器：



图2-16 对象浏览器让你在当前VBA工程里可用的所有对象，属性和方法里浏览

- 按F2
- 选择“视图”—“对象浏览器”
- 点击工具条上的“对象浏览器”按钮

对象浏览器让你浏览VBA过程中可用的对象，也可以查看它们的属性，方法和事件。在对象浏览器的帮助下，你可以在VBA工程的多个过程之间快速移动。

对象浏览器分为三部分（参见图2-16）。窗口的上部显示“工程/库”下拉列表，这里列出了所有库名称以及当前VBA工程里可用的所有工程名称。库是包含一个应用程序里对象的信息的专门文件。新的库可以通过“引用”对话框（“工具”—“引用”）来添加。<所有库>列出了你电脑上安装了的所有库中的所有对象。当你选择一个叫“Excel”的库时，仅仅能在Excel里执行的对象名称才能被看到。和Excel库相反，VBA库列出了所有能在VBA里执行的对象名称。

在“工程/库”下拉列表框下面，有一个“搜索”文本框，让你可以快速地在某个库里搜索你的信息。这个地方会记住你最近搜索的四个项目。在对象浏览器的任何地方单击右键，在快捷菜单上选择“全字匹配”，你就可以只搜索匹配整个字的内容。对象浏览器上的“搜索结果”（参见图2-16和2-17）显示符合搜索条件的库，类和成员。当你输入搜索文本并且单击搜索按钮，VB展开对象浏览器对话框以显示搜索结果。你可以点击“望远镜”按钮右边的“显示/隐藏搜索结果”来显示或者隐藏搜索结果。

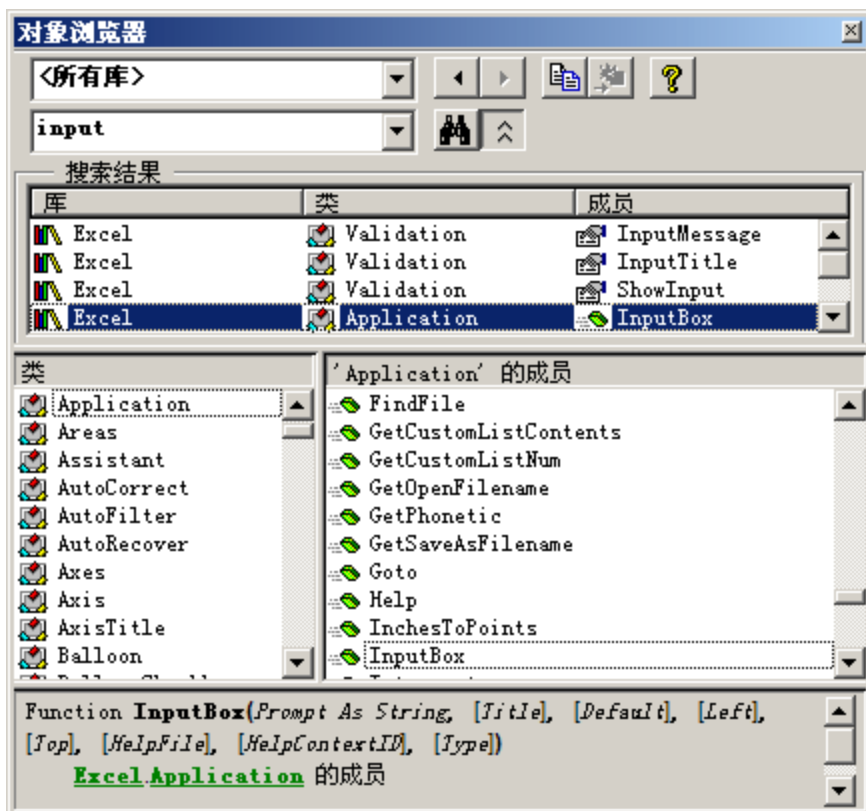


图2—17 在对象浏览器里搜索答案

类列表框显示所选中的库里面所有可用的对象类，如果你选择VBA工程，列表显示该工程里的对象。在图2—16里，CommandBarComboBox 对象类被选中了。当你选中一个类，右边的列表（成员）显示该类可用的属性，方法和事件。图2—16上显示了类CommandBarComboBox的一些成员。成员默认地按字母顺序列出。然而，你可以使用对象浏览器快捷菜单上的“组成成员”来组织这些成员列表（属性，方法或事件）。如果你选择“工程/库”里面的VBA工程，成员列表框列出该工程里的所有过程。双击该过程名称，就可以进入该过程并检查其代码。如果你选择类“VBA”，你将看到VB内置的函数和常数。如果你对所选类或成员需要更多的信息，可以点击对象浏览器窗口上面的问号按钮。对象浏览器下面的窗口显示所选成员的代码格式。如果你点击代码格式里绿色的连接部分，你将跳到对象浏览器窗口的所选成员类或库。代码格式里的代码可以被复制到Windows剪贴板并且粘贴到代码窗口里去。如果对象浏览器和代码窗口都是可见的，你只要选中代码格式里的文本，直接拖曳到代码窗口就行了。

通过对象浏览器窗口上的横竖分割线，你可以轻易地改变各个窗口的大小。

你已经发现了对象浏览器，你也许在想如何才能让它帮助你进行VBA编程。假设你在工作表中央放置了一个文本框，你如何让Excel将这个文本框移动到工作表的左上方？

1. 打开一个新工作表
2. 选择“视图”—“工具栏”，然后点击“绘图”
3. 点击“绘图”上的文本框，在工作表中央画一个文本框，并且随便输入什么文字
4. 选择文本框之外的任意单元格
5. 按下Alt+F11激活VB编辑器窗口，并且选择工程浏览器窗口的Personal (Personal.xls)
6. 选择“插入”—“模块”，增加一个新的模块
7. 在属性窗口，给该模块重命名：Manipulations
8. 选择“视图”—“对象浏览器”，或按F2
9. 在“工程/库”下拉列表框里选择“Excel”类
10. 在搜索框里输入“textbox”并点击搜索按钮。确保你没有在文字间敲入空格。

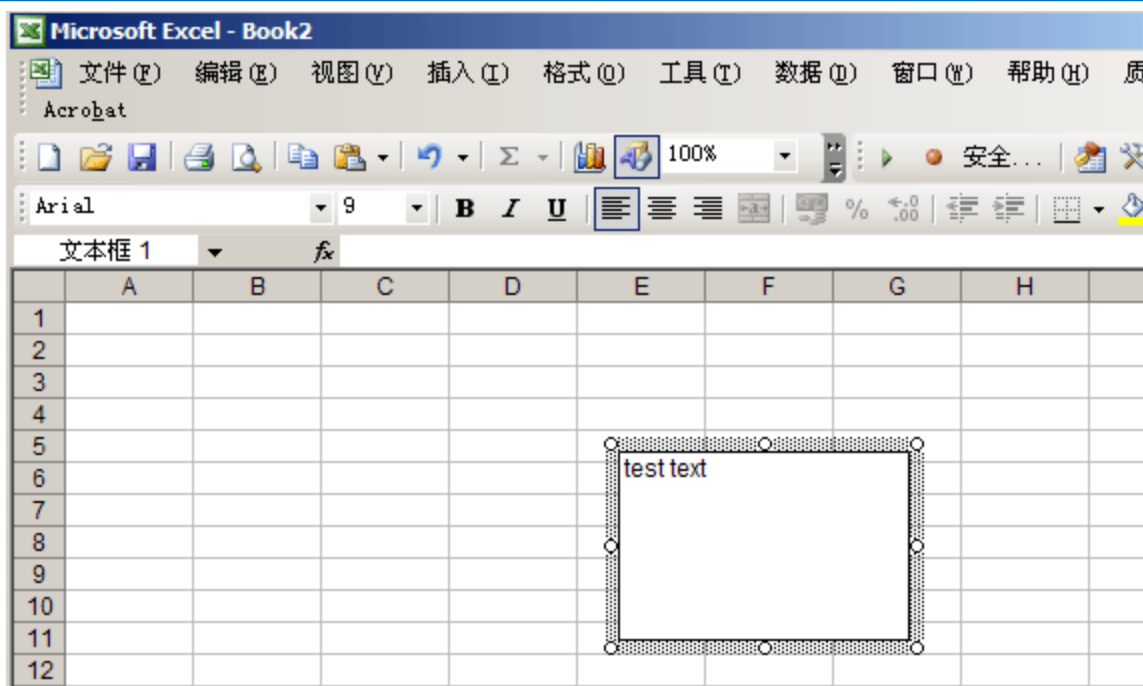


图2-18 Excel 在工作表上面的名称框里显示插入的对象名称

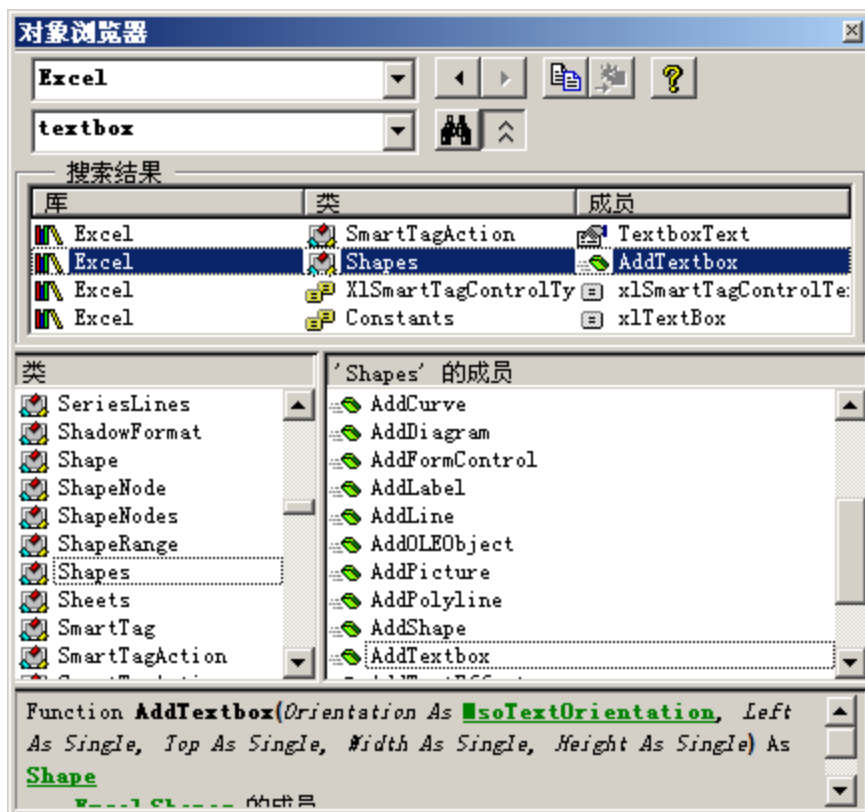


图2-19 使用对象浏览器窗口，你可以找到合适的VBA指令来编写你自己的过程

显示的结果表明对象Shapes掌管我们文本框操作（参见图2-19）。从成员列表清单上看，你可以很快就知道AddTextbox方法就是用来在工作表里添加文本框的方法。代码格式窗口显示了使用该方法的正确语法。如果你选择AddTextbox方法并且按F1，你将看到关于它的帮助窗口，有更详细的关于如果使用该方法的信息（参见图2-20）



图2-20 要获取对象浏览器找到的任何项目详细信息，只要选择整个项目并且按F1就可以了

当你细看AddTextbox方法的自变量和它们在帮助窗口上的解释时，你就可以很快地知道文本框在工作表中的位置是由Left和Top属性来决定的。你需要做的只是返回代码窗口，编写指令来移动文本框到工作表的左上方。

11. 关闭对象浏览器和帮助窗口（如果它们还是打开的）

12. 双击Manipulations模块，输入过程MoveTextBox：

```
Sub MoveTextBox()  
    With ActiveSheet.Shapes("Text box 1")           'Text box 1在中文版本里为“文本框 1”  
        .Select  
        .Left = 0  
        .Top = 0  
    End With  
End Sub
```

13. 选择“运行”－“运行宏”来测试这个过程

当你返回到放置该文本框的工作表时，该文本框已经移动到了工作表的左上方了。注意，MoveTextBox程序在Shapes集合里选择了“Text box 1”。Text box 1是工作表里第一个对象的默认名称。你每次增加新的对象后，Excel将给它安排新的好码（编号）。除了使用对象名称外，你还可以引用集合成员的编号。例如，你可以输入：

```
With ActiveSheet.Shapes(1)
```

来代替：

```
With ActiveSheet.Shapes("Text box 1")
```

我们用VB操纵另一个对象，你自己试试。在你放置文本框的工作表里再放置一个小圆圈。使用绘图工具上的椭圆工具画这个圆圈。在Manipulations模块里插入一个新的过程，并且编写代码来放置圆圈。记住，Excel连续地编号。第一个对象的编号为1，第二个则为2，等等，不管这个对象的类型是文本框，椭圆或者是矩形，没有关系。

下面的过程MoveCircle演示如何将当前工作表里的椭圆移动到左上方去：

```
Sub MoveCircle()  
    With ActiveSheet.Shapes(2)  
        .Select
```

```

        .Left = 0
        .Top = 0
    End With
End Sub

```

移动椭圆和移动文本框或者放在工作表里的其它对象类似。注意，过程中引用的是对象的编号，而非它的名称椭圆 2。当你运行MoveCircle时，Excel移动的是椭圆，而不是文本框了。

20 使用 VBA 对象库

在前面的例子里，你学习使用了Excel对象库里的Shapes（图形）集合成员的属性。Excel库包含专门使用Excel的对象，而VBA库则提供对许多内置VBA函数的访问，这些函数按类别分组。这些函数是通用的，它们使你能够管理文件，设置日期和时间，与用户交流，转换数据类型，处理文本串或者进行数学计算。在下面的练习中，你将学习如何使用内置的VBA函数来创建一个新文件夹，而不需要离开Excel界面：

1. 回到模块Manipulations，那里有你的MoveTextBox和MoveCircle过程
2. 输入一个新的过程：
Sub NewFolder()
3. 点击回车键，VB会自动输入结束关键词End Sub
4. 按下F2激活对象浏览器
5. 在“工程/库”列表框里选择VBA
6. 在搜索文本框里输入file并且回车
7. 滚动成员列表框，并且选中MkDir方法（参见图2-21）
8. 点击对象浏览器上的“复制”按钮，将被选择的方法名称复制到剪贴板



图2-21 编写过程时，向对象浏览器寻求帮助来找内置的VBA函数

9. 返回Manipulations窗口，并且将复制的指令粘贴到NewFolder过程
10. 输入一个空格，接着是”C:\Study”。确保你在引号里输入了整个路径名。NewFolder过程为：

```

Sub NewFolder()
    MkDir "C:\Study"
End Sub

```

11. 运行过程NewFolder

当你运行NewFolder过程，VB在C盘上创建了一个新的文件夹。激活Windows浏览器可以查看该新文件夹。创建一个新的文件夹后，你可能会发现你根本就不需要它，虽然你可以轻易地从Windows浏览器里删除该文件夹，但是，如何从编程上去掉它呢？对象浏览器上列出了许多对文件夹和文件操作很有帮助的其它方法。RmDir方法正如MkDir 方法一样使用简单。想要删除你硬盘上的“Study”文件夹，只要将MkDir方法换成RmDir方法然后重新运行NewFolder过程就可以了。或者，你也可以创建一个新的过程RemoveFolder，如下：

```
Sub RemoveFolder()  
    RmDir "C:\Study"  
End Sub
```

RmDir方法允许你从硬盘上删除不需要的文件夹。

21 用对象浏览器来定位过程

除了定位对象，属性和方法外，对象浏览器还是个定位在不同工程里面的过程非常方便的工具。下面的例子给你演示如何查看存在“Personal”工作簿里面的过程：

1. 激活对象浏览器并且选择工程/库下拉列表里的Personal。（译者：因为我没有Personal这个文件。。。）

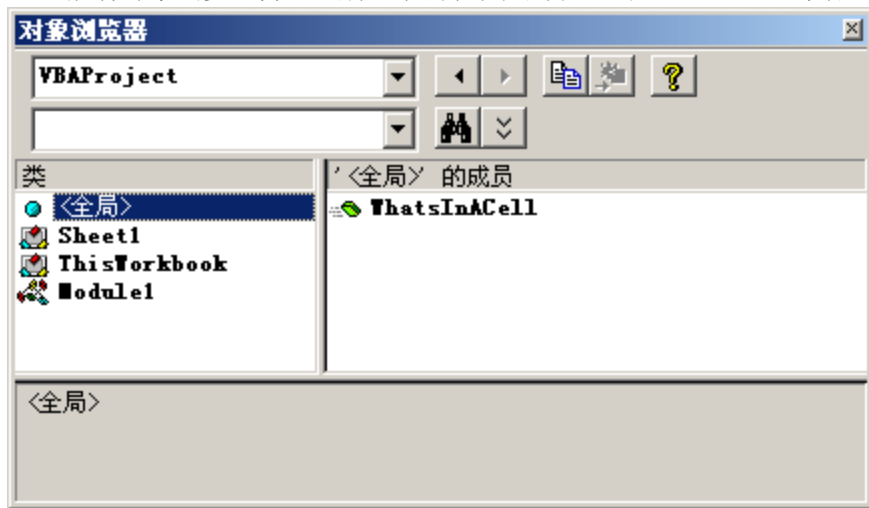


图2—22 对象浏览器列出所有在某个特定VBA工程里可用的过程

对象浏览器的左边显示所选工程里面对象名称，而右边则列出了所有可用的过程。

2. 双击NewFolder过程名称，VB将光标定位到该过程的第一行
3. 关闭对象浏览器

22 使用立即窗口

在你开始创建一个完善的VBA过程前（在下一章），先来做一些热身练习，增加你的VBA词汇。你怎样才能学得快而且没有痛苦？你如何试验一下一些新学的VBA指令？这里有一些简短的，互动的语言练习：输入一个简单的VBA指令，Excel会检查并且将结果显示在下一行。我们开始来设定你的练习屏幕：

1. 在VB编辑器窗口，选择“视图”－“立即窗口”

在决定使用在你自己的VBA过程之前，立即窗口可以用来试验VB语言中不同的指令，函数和运算符。这是一个非常好的调试新语言的工具，你输入在这个窗口里面的指令，将会立即显示结果。

立即窗口可以在VB编辑器窗口上任意移动，也可以设置为可连接的，这样它就会出现在相同的地方。可以通过选项对话框上的“可连接的”页来打开或关闭可连接设置。在VB编辑器窗口上按下Ctrl+G就可以快速访问立即窗口。立即窗口允许你输入VBA语句，并且测试它们的结果，而不需要写成一个过程。立即窗口就像一个草稿板，你可以用它测试你的语句。如果该语句输出了你希望的结果，你就可以将立即窗口上的语句复制到你的过程中去（或者，你也可以将语句拖曳到代码窗口，如果代码窗口是可见的）

2. 将Excel和VB编辑器窗口并排排列
3. 在立即窗口里输入下述指令，并且回车

```
Worksheets("Sheet2").Activate
```

当你按下回车键，VB开始工作，如果你上面输入的语句是正确的话，VBA激活当前工作簿里的第二个工作表。工作簿底部的Sheet2这时应该是突出显示的。

4. 在立即窗口，输入其它VBA语句并回车

```
Range("A1:A4").Select
```

你一旦按下回车，VB将选中当前工作表的A1，A2，A3和A4

5. 在立即窗口里输入下述指令：

```
[A1:A4].Value = 55
```

当你按下回车，VB在A1:A4中的每个单元格里放置数字55。上面的语句是引用Range对象的一种所写方式，完整的语法可读性更强：

```
Range("A1:A4").Value = 55
```

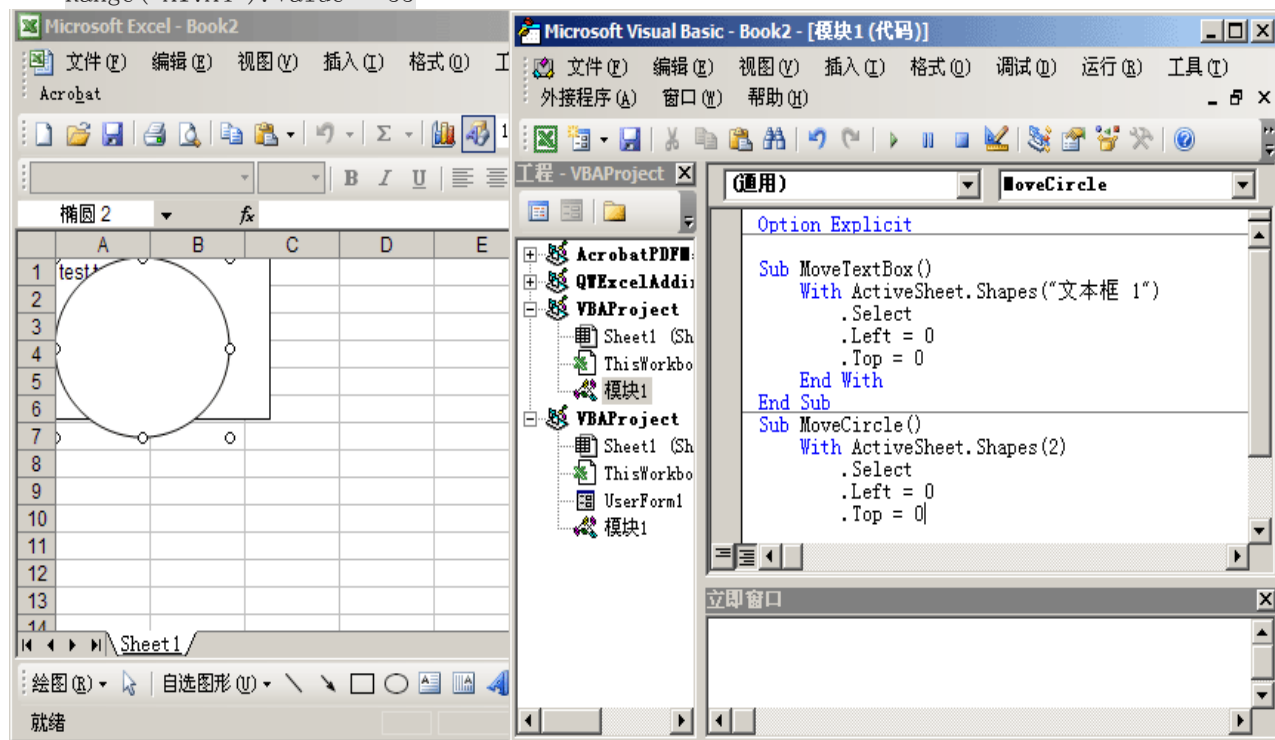


图2-23 将Excel和VB窗口并排排列让你可以观察指令的运行

6. 在立即窗口输入下述指令：

```
Selection.ClearContents
```

回车后，VBA清除所选单元格区域的内容，区域A1:A4现在是空的

7. 在立即窗口输入下述指令：

```
ActiveCell.Select
```

回车后，VB激活A1单元格

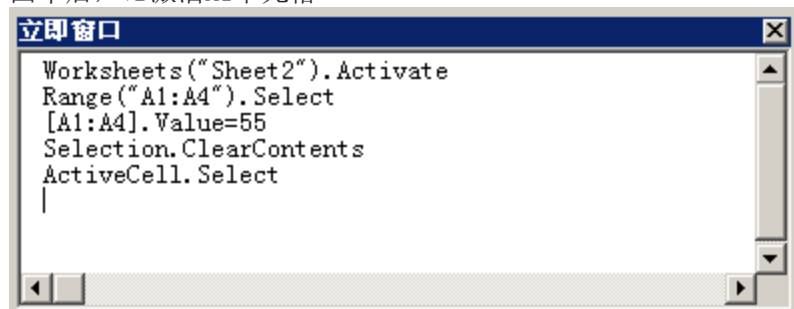


图2-24 在立即窗口里输入指令，一旦你按下回车键，指令就会被执行

图2-24显示了上面练习中在立即窗口里输入的所有指令。你每次按下回车键后，Excel总是执行光标所在行的语句。如果你想要再次执行同一指令，那么点击该指令行的任意位置，回车。为了更好的练习，重新运行图2-24里语句，从立即窗口的第二行指令开始，点击合适的地方并回车，一个一个地执行这些指令。

23 获取立即窗口里的信息

到目前为止，你已经使用立即窗口执行操作了，这些操作也可以是手动地在工作表的任意区域点击鼠标并且输入数据。立即窗口也允许你问问题。假设你想要找到下面问题的答案：“现在选中的是哪些单元格？”，“当前单元格的值是多少？”，“当前工作表的名称是什么？”，“当前窗口的编号是多少？”使用立即窗口，你可以轻易地找到这些问题，以及其它问题的答案。在前面的例子里，你输入了好几个指令，让我们返回立即窗口去问几个问题。Excel甚至在你关闭了立即窗口后还能记住你在立即窗口里输入的指令。当你退出Excel时，立即窗口的内容自动会被删除。

1. 鼠标点击立即窗口第二行你输入Range("A1:A4").Select的任意地方
2. 回车，让Excel再次选择单元格A1:A4
3. 在立即窗口新的一行输入下面的问题：

?Selection.Address

当你按回车，Excel不会选择工作表的任何东西，取而代之，立即窗口上会在另外一行显示该指令的结果。在该例中，Excel返回的是当前被选择的单元格的绝对地址（\$A\$1:\$A\$4）。问号（?）告诉Excel在立即窗口显示指令的结果。除了问号，你还可以使用Print关键字。让我们使用关键字Print问工作表名称

4. 在立即窗口新的一行，输入下述问题：

Print ActiveWorkbook.Name

回车后，Excel在立即窗口新的一行输入了当前工作簿的名称。

找找应用程序的名称如何？Chap02.xls的父对象是谁？

5. 在立即窗口新的一行，输入下述问题：

?Application.Name

Excel会显示它自己的全名：Microsoft Excel

立即窗口也可以用来做一个快速计算

6. 在立即窗口新的一行，输入下述问题：

?12/3

回车后，Excel会在下一行显示该除法运算的结果。但是，万一你想立即知道3+2和12*8的结果呢？你可以将它们输入在一行，而不必分成两行，例如：

?3+2:?12*8

注意，冒号将两个代码块分割开来。

当你按下回车键，Excel分别在立即窗口的两行显示结果5，96。

下面是你在立即窗口里输入的所有指令，以及Excel对你问题的回答：

```
Worksheets("Sheet2").Activate
```

```
Range("A1:A4").Select
```

```
[A1:A4].Value = 55
```

```
Selection.ClearContents
```

```
ActiveCell.Select
```

```
?Selection.Address
```

```
$A$1:$A$4
```

```
Print ActiveWorkbook.Name
```

```
Chap02.xls
```

```
?Application.Name
```

```
Microsoft Excel
```

```
?12/3
```

```
4
```

```
?3+2:?12*8
```

```
5
```

```
96
```

要清除立即窗口里的指令，只要选中所有指令并且按下Delete键

24 学习对象

要在Excel里创建一些自定义应用程序，需要一些常用对象或者对象集合的工作知识，例如Range，Workbook（Workbooks），Worksheet（Worksheets），Window（Windows）和 Application。在前面部分，你开拓了学习VB的许多方法。这里有一个总结关于什么时候使用什么工具：

- 当你在一个现行VBA过程，对对象，属性或方法有疑义时按F1打开在线帮助
- 如果你需要快速列出每个可用对象的属性和方法时，或者查找一个很难找到的过程时，使用对象浏览器
- 如果你想要测试VBA并且立即查看VBA命令的结果时，激活立即窗口。

本章剩余的几页里有一些VBA语言训练，可以帮助你更好地理解VBA语法。如果你花些时间在立即窗口过一遍这些语法训练，你将理解绝大部分。



图2—25 Excel对象模型里的Range对象

25 电子表格单元格操作

当你已经准备好编写你自己的VBA过程，将电子表格任务自动化的时候，你很可能是从寻求操作电子表格单元格的指令开始的。你需要知道如何选择单元格，如果在单元格输入数据，如何给单元格区域命名，如何设置单元格格式，以及如何移动，复制和删除单元格。虽然这些任务可以通过鼠标或键盘轻易执行，掌握VBA这方面的技术需要一些练习。你必须使用Range对象来引用单个单元格，单元格区域，行或列。如果你看了Excel对象模型，你会注意到Range对象是另外一个大对象——Worksheet对象——的一部分。有三种属性让你访问Range对象：Range属性，Cells属性和Offset属性。

26 使用 Range 属性

Range属性返回一个单元格或者单元格区域。引用必须是A1在引号里的样式（例如：“A1”）引用可以包括区域运算符冒号（例如：“A1:B2”）或者联合运算符逗号（例如：“A”，“B12”）

VBA操作	立即窗口输入
选择单个单元格（例如A5）	Range("A5").Select
选择一个单元格区域（例如A6:A10）	Range("A6:A10").Select
选择一些不相邻的单元格（例如A1，B6，C8）	Range("A1，B6，C8").Select
选择一些不相邻的单元格和单元格区域（例如A11:D11，C12，D3）	Range("A11:D11，C12，D3").Select

27 使用 Cells 属性

当你要选择一个确定的单元格时，Cells属性要求两个自变量，第一个是行号，第二个是列号或者列字母。自变量输入在括号中。如果忽略自变量，Excel将会选择当前工作表的所有单元格。

VBA操作	立即窗口输入
-------	--------

选择单个单元格（例如A5）	Cells(5, 1).Select或Cells(5, A).Select
选择一个单元格区域（例如A6:A10）	Range(Cells(6, 1), Cells(10, 1)).Select
选择工作表中所有单元格	Cells.Select

注意，在上面的例子中，你如何结合使用Range和Cells属性：

```
Range(Cells(6, 1), Cells(10, 1)).Select
```

在上面的例子里，第一个Cells属性返回单元格A6，而第二个返回单元格A10。Cells属性返回的单元格之后又当做Range对象的参数。结果Excel就选择了上面单元格为第一个Cells属性返回的结果和下面为第二个Cells属性返回单元格的区域了。工作表是单元格的集合，你也可以使用只带一个自变量的Cells属性来表示单元格在工作表所有单元集合中的位置。Excel按下列方式给单元格编号：单元格A1是工作表中的第一个单元格，B1是第二个，C1是第三个，等等。Cell1256是第一行中的最后一个单元格。你也许会想起Excel只有256列。

VBA操作	立即窗口输入
选择单元格A1	Cells(1).Select or Cells.Item(1).Select
选择单元格C1	Cells(3).Select or Cells.Item(3).Select
选择单元格IV1	Cells(256).Select or Cells.Item(256).Select
选择单元格A2	Cells(257).Select or Cells.Item(257).Select

注意，Item是返回一个集合成员的属性。因为Item是一个集合的默认成员，你可以直接引用工作表单元格，而不必明确地使用Item属性。

现在你发现了两种方法选择单元格（Range属性和Cells属性），你也许很迷惑为什么要使用更复杂的Cells属性呢？很明显Range属性更具有可读性，毕竟，你远在决定学习VBA之前就在Excel公式和函数里面使用了Range引用。然而，当需要将单元格当做集合操作的时候，Cells属性则使用更方便。使用这个属性去访问单元格集合中的所有单元格或者单个单元格。

28 使用 Offset 属性

另外一个引用工作表单元格非常灵活的方法是使用Offset属性。当工作表任务自动化时，你也许不知道某个单元格的确切地址。你如何能够选择一个你根本不知道地址的单元格？你可以让Excel基于当前选择的单元格来选择一个位置。Offset属性通过计算从开始选择的单元格向下或向上移动的具体行数，来得到新的区域。同样也可以从当前选择的单元格区域向右或向左移动具体的列数。Offset属性使用两个自变量来获得新单元格区域的地址。第一个自变量表示行偏移，第二个自变量则表示列偏移。我们来测试一下几个例子：

VBA操作	立即窗口输入
选择单元格A1下面一行和右边三列的单元格	Range("A1").Offset(1, 3).Select
选择单元格D15上面两行和左边一列的单元格	Range("D15").Offset(-2, -1).Select
选择当前单元格上面一行的单元格（同列）	ActiveCell.Offset(-1, 0).Select

上面的第一个例子里，Excel选择的时单元格D2。一旦你输入了第二个例子，Excel选择了单元格C13。

如果单元格A1和D15已经被选中了，你也可以将上面的两个例子改写为这样：

```
Selection.Offset(1, 3).Select
Selection.Offset(-2, -1).Select
```

注意，上面第三个例子里的第二个自变量是0，第一个或第二个自变量为0时，Offset属性相应表示当前行或当前列。如果当前活动单元格在第一行，那么指令ActiveCell.Offset(-1, 0).Select会导致错误。

当使用Offset属性时，你可能有时需要改变选择区域的大小。假设开始选择的区域是A5:A10，如何将选择区域向下移动两行，向右移动两列，然后再改变新选择区域的大小呢？假设新的选择区域应该是C7:C8。Offset属性只能完成前面部分，后面部分要求另外一个属性来完成。Excel有个专门的Resize属性，你可以结合Offset属性和Resize属性来回到上面的问题。在你结合这两个属性之前，我们先来看看如何独立地使用它们：

1. 将Excel窗口和VB窗口并排显示
2. 激活立即窗口，并且输入下述指令：

```
Range("A5:A10").Select
Selection.Offset(2, 2).Select
```

```
Selection.Resize(2, 4).Select
```

上面的第一条指令选择区域A5:A10，当前活动单元格是A5。第二条指令将选区偏移到C7:C12。单元格C7处于活动单元格A5的向下两行和向右两列。现在，活动单元格是C7。最后一条指令将当前选区改变大小，单元格区域C7:C8被选中了，而不再是C7:C12。象Offset属性一样，Resize属性也需要两个自变量。第一个是你要选取的行数，第二个则是要选取的具体列数因此，指令Selection.Resize(2, 4).Select将当前选择区域改为两行和四列

后面两行指令可以结合成下面方式：

```
Selection.Offset(2, 2).Resize(2, 4).Select
```

上面的例子，先是Offset属性计算得到新区域的起始点（译者：选区左上角的单元格），接着是Resize属性决定新选区的大小，然后是Select方法选取具体的单元格区域。

技巧2—6：录制单元格的选择

宏录制器默认地使用Range属性录制选择单元格。如果你打开宏录制器，并且选择单元格A2，输入“text”，再选择单元格A5，你将在VB编辑器窗口里得到下述代码：

```
Range("A2").Select
```

```
ActiveCell.FormulaR1C1 = "text"
```

```
Range("A5").Select
```

如果你使用相对引用方式，宏录制器会使用Offset属性。你可以在录制前，点击宏录制工具条上的相对引用按钮。宏录制器将得到如下代码：

```
ActiveCell.Offset(-3, 0).Range("A1").Select
```

```
ActiveCell.FormulaR1C1 = "text"
```

```
ActiveCell.Offset(3, 0).Range("A1").Select
```

当你使用相对引用方式录制宏时，过程总是会选择相对于当前活动单元格的单元格。注意，上面指令中的第一和第三行的引用单元格A1，即使我们没有涉及到A1的任何东西。你可能记得，在第一章中，宏录制器用它自己的方式将事情搞定。为了将上面的指令变简单一些，你可以删除对单元格A1的引用：

```
ActiveCell.Offset(-3, 0).Select
```

```
ActiveCell.FormulaR1C1 = "text"
```

```
ActiveCell.Offset(3, 0).Select
```

使用相对引用来录制过程后，不要忘记再次点击这个按钮，如果下次录制一个非相对地址的过程。

29 选择单元格的其它方法

如果你经常需要访问你工作表里某些遥远的单元格，你可能已经对下面的键盘快捷键很熟悉：End+上箭头，End+下箭头，End+左箭头和End+右箭头。在VBA中，你可以使用End属性快速地移动到遥远的单元格。

VBA操作	立即窗口输入
选择任何行的最后一个单元格	ActiveCell.End(xlright).Select
选择任何列的最后单元格	ActiveCell.End(xldown).Select
选择任何行的第一个单元格	ActiveCell.End(xleft).Select
选择任何列的第一个单元格	ActiveCell.End(xlup).Select

注意，End属性要求一个自变量来表示你要移动的方向。使用下列Excel内置的常数来跳到具体的方向：xlright, xleft, xup, xldown。

30 选择行和列

Excel使用EntireRow和EntireColumn属性来选择整行或整列。

VBA操作	立即窗口输入
选择当前活动单元格所在行的整行	Selection.EntireRow.Select
选择当前活动单元格所在列的整列	Selection.EntireColumn.Select

你选择了一个单元格区域，你也许想要知道选区包括多少行，多少列。我们来让Excel计算区域A1:D15中的行数和列数：

1. 在立即窗口里输入下述VBA语句

```
Range("A1:D15").Select
```

如果Excel窗口可见，当你按回车后，VBA会选中区域A1:D15

2. 输入下列语句来得到选区的行数

```
?Selection.Rows.Count
```

一旦你回车，VBA在下一行显示结果。你的选择包括15行

3. 输入下列语句来得到选区的列数

```
?Selection.Columns.Count
```

现在VBA告诉你，选中的区域A1:D15占据了四列的宽度。

4. 将光标放在关键字Rows或Columns中的任意位置，并且按下F1，获取这些有用属性的更多信息。

31 获取工作表信息

Excel工作表有多大？它有多少单元格，列和行？即使你忘记了这些细节，使用Count属性。

VBA操作	立即窗口输入
计算Excel工作表里总单元格数	?Cells.Count
计算Excel工作表里总行数	?Rows.Count
计算Excel工作表里总列数	?Columns.Count

Excel 2002工作表里有16,777,216个单元格，65,536行和256列。

32 往工作表输入数据

输入工作表里的信息可以是文本，数字或者公式。你可以使用Range对象的两种属性之一来往单元格或单元格区域里输入数据：Value属性或者Formula属性。

Value属性：

```
ActiveSheet.Range("A1:C4").Value = "=4 * 25"
```

Formula属性：

```
ActiveSheet.Range("A1:C4").Formula = "=4 * 25"
```

上面两种例子，A1单元格都显示4乘25的结果100。

VBA操作	立即窗口输入
在单元格A5里输入文本“Amount Due”	Range("A5").Formula = "Amount Due"
在单元格D21里输入数字“123”	Range("D21").Formula = 123
	Range("D21").Value = 123
在单元格B4里输入公式“=D21*3”	Range("B4").Formula = "=D21 * 3"

33 返回工作表中的信息

毫无疑问，你在某些VB过程中可能需要返回单元格或者单元格区域的内容。虽然你既可以使用Value属性也可以使用Formula属性，但是，这次，Range对象的这两个属性是不可互用的。

- Value属性显示具体单元格中公式的结果。例如，如果A1中含有公式“=4*25”，那么指令?Range("A1").Value将会返回值100
- 如果你想要显示公式，而不是结果，那么你必须使用Formula属性：?Range("A1").Formula。Excel将会显示公式“=4*25”而不是结果100

34 单元格格式

一个频繁的任务就是给选中的单元格或区域设置格式。你的VBA过程可能需要查明某个具体单元格的格式。我们可以使用

NumberFormat属性来找回单元格格式：

```
?Range("A1").NumberFormat
```

在立即窗口输入上面的问题后，Excel显示“General”（译者：中文版本是“常规”，G/通用格式），它表示所选的单元格没有设置任何特殊的格式。要用VBA改变单元格格式，输入下列指令：

```
Range("A1").NumberFormat = "$#,##0.00"
```

如果你在单元格A1里输入125，当你使用上面的指令给它设置格式后，单元格A1将显示“\$125.00”。你可以在Excel窗口的“设置单元格格式”对话框里查找必要的格式代码（“格式”－“单元格”）。如果你要的格式没有列在“设置单元格格式”对话框里，那么请参考在线帮助，查找创建用户定义的格式指导。

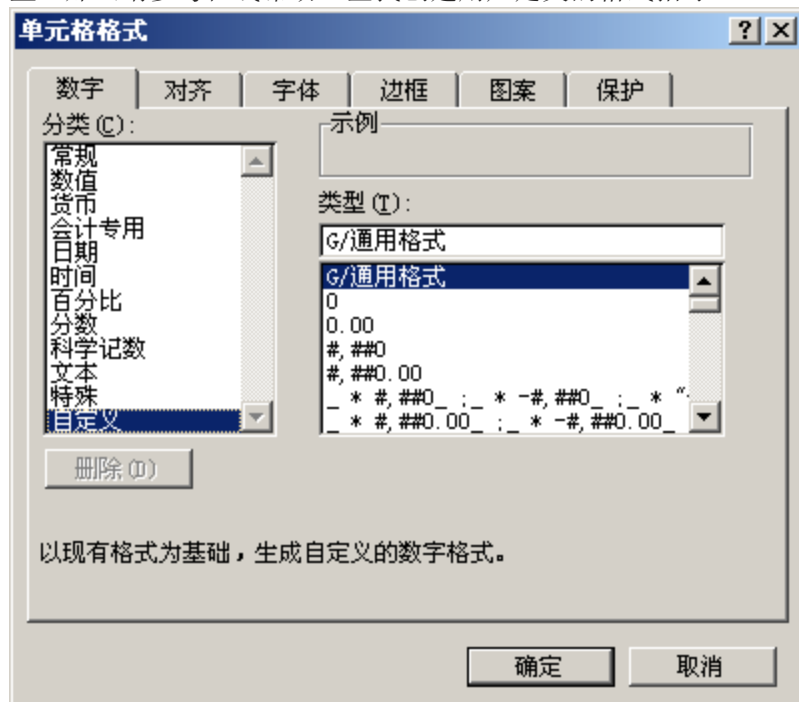


图2-26 你可以使用设置单元格格式对话框里的自定义给选择的单元格或单元格区域设置不同的格式

35 移动，复制和删除单元格

在你做一个新的工作表模板时，你会发现经常要移动，复制和删除单元格内容。VB让你的工作表编辑工作自动化变得可能，只要使用一些容易使用的方法就行：Cut，Copy和 Clear。

VBA操作	立即窗口输入
移动单元格A5的内容到单元格A4里面	Range("A5").Cut Destination:=Range("A4")
复制单元格A3里的公式到区域D5:F5中	Range("A3").Copy Destination:=Range("D5:F5")
清除单元格A4里的内容	Range("A4").Clear Range("A4").Cut

注意，使用在Range对象上的Cut和Copy方法都需要一个叫“Destination”的特殊自变量。这个自变量明确你要放置剪切或复制的数据的单元格或单元格区域地址。在最后一个例子中，使用了没有Destination自变量的Cut方法去除具体单元格的数据。Clear方法将删除具体单元格或单元格区域的所有内容，包括格式和批注。如果你想要明确你要删除什么，使用下列方法：

- ClearContents—仅清除单元格或单元格区域内的数据
- ClearFormats—仅清除格式
- ClearContents—清除区域里的所有批注

36 操作工作簿和工作表

既然你已经涉足操作工作表单元格和单元格区域，是时候上一个台阶，学习如何控制单个工作簿，已经整个工作簿集合了。如果你不知道如何打开一个新工作簿的话，你就不知道准备一个新的电子表格了；如果你不知道如何关闭工作簿，你就不知道如何将工作簿从屏幕上消除。这些重要的任务由两个VBA方法处理：Add和Close。下面的练习将给你必要的如何操作工作簿和工作表的语言技巧。

VBA操作	立即窗口输入
打开一个新工作簿	Workbooks.Add
获得第一个工作簿的名称	?Workbooks(1).Name
获得打开的工作簿数目	?Workbooks.Count
激活第二个打开的工作簿	Workbooks(2).Activate
激活工作簿Chap02.xls	Workbooks("Chap02.xls").Activate
当前活动的工作簿存盘为NewChap.xls	ActiveWorkbook.SaveAs Filename:="NewChap.xls"
关闭第一个工作簿	Workbooks(1).Close
关闭当前活动的工作簿，不保存变化	ActiveWorkbook.Close SaveChanges:=False
关闭所有打开的工作簿	Workbooks.Close

如果你运行了最后一个例子，那么现在你所有的工作簿都已经关闭了。在你要在工作表上使用，请确保先打开一个新工作簿。当你除了单个工作表时，你必须知道如何在工作簿里添加新的工作表，知道如何选择一个或一组工作表，知道如何命名、复制、移动和删除工作表。在VB里，每个任务都需要一个专门的方法或属性。

VBA操作	立即窗口输入
添加一个新工作表	Worksheets.Add
获得第一个工作表的名称	?Worksheets(1).Name
选择名为“Sheet3”的工作表	Worksheets(3).Select
选择第一，第三和第四个工作表	Worksheets(Array(1, 3, 4)).Select
激活名为“Sheet1”的工作表	Worksheets("Sheet1").Activate
将工作表“Sheet2”移动到工作表“Sheet1”之前	Worksheets("Sheet2").Move Before:=Worksheets("Sheet1")
重命名工作表“Sheet2”为“Expenses”	Worksheets("Sheet2").Name = "Expenses"
获得当前工作簿里的工作表数目	?Worksheets.Count
删除当前工作簿里的工作表“Expenses”	Worksheets("Expenses").Delete

注意Select方法和Activate方法之间的区别：

- 当只要一个工作表被选择时，Select和Activate方法可以互换使用
- 如果你要选择一组工作表，Activate方法将让你决定你选中的工作表中哪个要激活。我们知道，同时只能有一个工作表被激活。

技巧2-7：Sheets（译者简称为“表”）而不是Worksheets（译者简称为“工作表”）

除了工作表之外，工作簿集合里还包括图表。使用Add方法在工作簿里添加一个新图表：

Charts.Add

统计图表数目，使用：

?Charts.Count

在Excel 97之前的版本中，工作簿集合里包括两种额外的表：DialogSheets和Modules。Dialogs已经被更亲切的用户窗体（UserForms）所取代了。从Excel 97开始，对话框和模块都被创建在VB编辑器窗口里面了。

37 操作窗口（Windows）

当在好几个Excel工作簿上工作，并且需要比较或者巩固数据，或当你想要看同一个工作表里的不同部分时，你很可能要用到Excel“窗口”菜单里的选项：新建窗口和重排窗口。我们来看看如何通过VBA来安排窗口。

VBA操作	立即窗口输入
在新窗口里显示当前活动工作簿	ActiveWorkbook.NewWindow
在屏幕上显示所有打开了的工作簿	Windows.Arrange
激活第二个窗口	Windows(2).Activate
获得当前窗口的名称	?ActiveWindow.Caption
将当前窗口的名称改为“My Window”	ActiveWindow.Caption = “My Window”

当你在屏幕上显示窗口时，你可以决定如何排列它们。Arrange方法由许多自变量，让你如何放置窗口的自变量称为ArrangeStyle（排列方式）。如果你忽略ArrangeStyle自变量，Excel将平铺所有窗口。

常数	值	描述
xlArrangeStyleTiled	1	平铺窗口（默认模式）
xlArrangeStyleCascade	7	层叠窗口
xlArrangeStyleHorizontal	2	水平并排窗口
xlArrangeStyleVertical	3	垂直并排窗口

除了使用常数名称外，你也可以使用上面列出的等价值。要将所有窗口层叠起来，写下面的指令就可：

```
Windows.Arrange ArrangeStyle:=xlArrangeStyleCascade
```

或者更简单点：

```
Windows.Arrange ArrangeStyle:=7
```

38 管理 Excel 应用程序

在本章的开始部分，你学习了对象是组织在一个叫对象模型的专门结构。在应用程序的对象模型的最上面就是应用程序它本身。通过控制Application对象，你可以进行很多操作，例如将屏幕显示效果保存为当日最后显示的效果，或者退出该应用程序。你知道，Excel允许你使用“文件”菜单里的选项“保存工作区”来保存屏幕设定。在VBA里可以很容易地完成保存工作区的工作：

```
Application.SaveWorkspace “Project”
```

上面的指令将屏幕设置保存在名叫“Project”的工作区里。下次你要在相同的文件和窗口排列时，只要打开“Project”文件，Excel就会打开正确的文件和恢复你要的屏幕。

VBA操作	立即窗口输入
获取当前应用程序名称	?Application.Name
将Excel应用程序标题改为“My Application”	Application.Caption = “My Application”
将Excel应用程序标题改回为“Microsoft Excel”	Application.Caption = “Microsoft Excel”
获取你正在使用的操作系统	?Application.OperatingSystem
获取该应用程序注册的人名或公司名	?Application.OrganizationName
获取Excel.exe保存的文件夹路径	?Application.Path
退出Excel	Application.Quit

39 接下来……

在本章，你学习了很多基础的VBA术语和内置工具，可以用来使你的程序编写和调试更容易。你现在应该对绝大多数Excel对象是如何组织和控制的有了一个比较好的了解。我努力使描述尽量少，并且注重于教你如何使用你的新语言技巧立即控制Excel而不必先编写过程。正因为此，我注重于立即窗口。VB过程通常包含多于一行的代码，事实上，它们可能变得很复杂。在你开始创建完整的VBA过程之前，你仍然需要学习一些事情。例如，你如何保存目前Excel返回的信息，你的过程稍后可以使用？在立即窗口里输入指令的时候，你学习了如何问Excel一些重要的信息，你得到了类似“当前活动工作簿里有多少工作表？”或“单元格A4的内容是什么？”问题的答案。Excel不会在乎你的问题有多么烦，只要你输入符合严格VBA语法的问题，Excel就会给你答案。当你开始编写你自己的过程时，你会需要知道如何保存Excel的答案。在下章里，你将学习如何保存这种以后变量需要用到的信息。你也将展开数据类型和常数主题的学习。

第三章 了解变量，数据类型和常量

作者: Julitta Korol 翻译: Tiger Chen Dec 18' 2004

就象现实生活中一样，编程中也是有些事情必须马上做，而其它的事情可以稍后进行。当你推迟一件事情，你可以将它放入你心里的或纸上的“要做的事情”清单。清单上零零散散的事情，经常按照它们类型或者重要性来分类。当你将任务交给别人或者最终你要开始做了，你需要将该任务从清单里划掉。本章将演示你的VBA过程如何记住一些重要的信息，后面将用在你的语句或计算里。你将学习过程如何保留不断地往变量输入“要做的事情”，如何声明变量，以及它们和数据类型及常量有何关系。

1 保存 VBA 语句的结果

在第二章，你在立即窗口上输入一些VB指令，并且返回一些信息。例如，当你输入?Cells.Count，你发现工作表里有16,777,216个单元格。然而，当你在立即窗口之外的地方写VB过程时，你不能使用问号。当你忽略问号输入Cells.Count，VB不会突然停下来告诉你这个指令的结果。如果你想要知道某个指令执行后的结果，你就必须告诉VB记住它。在编程中，VB指令返回的结果可以赋值给变量。

2 变量是什么

变量是一个简单的用来引用一条数据的名称。你每次想要记住一个VBA指令的结果时，考虑用一个名称来代表它。例如，如果你必须用数字16,777,216来提醒你工作表中的总单元格数目，你可以使用一个名称，如AllCells, NumOfCells, TotalCells, 等等来代替。变量名称里可以包含字母，数字和一些标点符号，除了下面这些之外

, # \$ % & @ !

变量的名称不可以以数字开始，也不可以含有空格。如果你想在变量名称里包含多于一个词语，可以使用下划线。虽然变量名称最多可以包含254个字母，但是，你最好使用短而简单的变量名称。使用短名称将会节省你的输入时间，如果你需要在你的VB过程里多次引用该变量的话。VB不管你在变量名称里使用大写字母还是小写字母，然而，大多数编程者使用小写字母，并且当变量名称包括一个或多个词语时，他们使用标题字母，那就是，象下面这样，他们将每个词语词头大写：NumOfCells, First_Name。（译者：中文也可以做为变量名称使用，但是，个人不建议使用中文）

技巧3—1 不能用作变量名称的词语

除了这些VBA占用了的词语之外，你可以使用任何你想用的标签作为变量名称。在VBA中有特定意义的VB语句以及其它某些词语不能用作变量名称。例如，词语Name, Len, Empty, Local, Currency或者Exit，如果你使用它们作为变量名，将会产生错误。

技巧3—2 富有意义的变量名称

给变量那种可以帮助你记住它们作用的名称。有些程序员使用前缀来识别变量类型。在你的代码中，一个以前缀“str”开头的变量名称（例如strName），很快就可以知道它是传递文本字符串的变量。

3 数据类型

当你创建VB过程时，你脑海里必然有个目的，你想要处理数据。因为你的过程要处理不同类型的信息，所以，你应该了解VB如何储存数据。“数据类型”这个术语决定了数据如何储存在电脑的内存里。例如，数据可以储存为数字，文本，日期，对象，等等。如果你忘了告诉VB你的数据类型，VB将分配数据类型为“Variant”。“Variant”类型有能力解决数据本身的操作类型并且使用该类型。表3—1里列出了VB数据类型。除了内置的数据类型之外，你还可以定义你自己的数据类型。（你将在第八章里看到用户自定义的数据类型的例子。）因为不同的数据类型占据电脑内存的空间是不一样的，一些类型比另外一些更“贵”些，因此，为了保存内存并确保你的过程运行更快，你应该选择占用字节最少的，同时又能够处理你数据的数据类型。

表3—1 VBA数据类型

数据类型 (名称)	大小 (字节)	描述
Boolean	2	逻辑值True或False
Byte	1	0到255的整数
Integer	2	- 32, 768到32, 767的整数
Long	4	- 2, 147, 483, 648到2, 147, 483, 647的整数
Single	4	单精度浮点数值 负数: - 3. 402823E38到 - 1. 401298E - 45 正数: 1. 401298E - 45到3. 402823E38
Double	8	双精度浮点数值 负数: - 1. 79769313486231E308到 - 4. 94065645841247E - 324 正数: 4. 94065645841247E - 324到1. 79769313486231E308
Currency	8	(放大的整数(译者: 整数除以10000得到的数值, 参见VBA帮助)) 使用在定点计算中: - 922, 337, 203, 685, 477. 5808到922, 337, 203, 685, 477. 5807 +/- 79, 228, 162, 514, 264, 337, 593, 543, 950, 335没有小数点; +/- 7. 9228162514264337593543950335小数点后有28位数字; 最小的非0数字是 +/- 0. 00000000000000000000000000000001
Decimal	14	从100年1月1日到9999年12月31日的日期
Date	8	变长字符串最多可包含大约 20 亿 (2^{31}) 个字符。
String (变长字符串)	10字节+字符串长度	
String (定长字符串)	字符串长度	定长字符串最多可包含大约65, 400 个字符。
Object	4	对象变量用来引用Excel中的任何对象
Variant (带数字)	16	最高范围到Double类型的任何数值
Variant (带字母)	22字节+字符串长度	和变长字符串的范围一样
用户定义类型 (使用Type)	成员所需的数值	每个成员的范围和它的数据类型的范围一致

4 如何产生变量

你可以通过一个专门的命令来声明变量从而产生一个变量, 或者也可以直接在语句里使用变量 (而不需要声明)。当你声明变量时, 你实际上让VB知道该变量的名称和数据类型, 这叫做“强制显式声明变量”。

如果你在使用变量前不告诉VB关于该变量的任何信息, 你这是在含蓄地告诉VBA你想要创建这个变量。没有明确声明的变量会自动地分配为Variant数据类型 (参见表3-1)。虽然不声明变量很方便 (你可以随意创建变量, 并且不用事先知道被赋值的数值的数据类型就可以赋值给该变量), 但是, 它会导致很多问题, 参见技巧3-4中要点。

技巧3-3 强制显式声明变量的好处

- 强制显式声明变量加速过程的执行。因为VB知道数据类型, 它只会占用实际储存数据需要的内存
- 强制显式声明变量使你的代码可读性和可理解性增加, 因为所有的变量都已列在过程的最前面
- 强制显式声明变量帮助预防由于变量名称拼写错误而导致的错误。VB根据变量声明里的拼写自动更正变量名称。

技巧3-4 隐式声明变量的坏处

- 如果你错误拼写了一个变量名称, VB会显示运行时间错误, 或者产生一个新的变量。你保证需要浪费很多时间来做故障排除, 然而, 如果在过程前声明了变量, 这些很容易避免
- 因为VB不知道你要保存的变量的数据类型, 它将分配给它Variant数据类型。这导致你的过程运行要慢一些, 因为VB每次在处理这个变量时不得不检查数据类型。因为Variant可以储存任何一种数据类型, VB不得不占用更多的内存来储存你的数据

5 如何声明变量

你可以使用关键字Dim来声明变量，Dim代表“Dimension”。关键字Dim后面紧跟变量名称，再后面就是数据类型。假设你想让过程显示员工的年龄，你计算年龄之前，必须给过程提供员工的生日。你可以这样做，声明一个叫DateOfBirth的变量：

```
Dim DateOfBirth As Date
```

注意，关键字Dim之后是变量名称（DateOfBirth）。如果你不喜欢这个名称，你可以自由地改为其它的，只有你想用的名称不是VBA关键字之一就行。关键字As以及后面的表3-1其中的一个数据类型，明确了该变量的数据类型。数据类型Date告诉VB变量DateOfBirth将会储存日期。

要储存员工的年龄，按下面方式声明变量Age：

```
Dim Age As Integer
```

变量Age将会储存今天和该员工生日之间年数的数字。因为年龄显示为整年，所以变量Age就被分配为Integer数据类型。

你可能还想要你的程序追踪员工的姓名，因此需要声明另一个变量来保存员工的名和姓：

```
Dim FullName As String
```

因为词语“Name”已经在VBA占用的清单上，在你的VBA程序里使用它的话保证会有错误。将变量命名为FullName并且将它声明为String类型（因为员工姓名是文本），来保存员工姓名。

技巧3-5 隐式声明变量

没有用Dim语句来明确声明的变量叫做隐式声明。这些变量自动会被分配一个数据类型Variant。它们可以保存数字，字符串和其它信息类型。你可以通过在你VBA程序的任何地方，简单地赋值给一个变量名称来创建一个变量。例如，你可以按下述方式来隐式声明变量：DaysLeft = 100

声明变量被认为是编程的好习惯，因为它使程序可读性增强并且帮助避免某些类型的错误。既然你已经知道了如何声明变量，我们就来看一下使用它的一个程序：

```
Sub AgeCalc( )
    'variable declaration (变量声明)
    Dim FullName As String
    Dim DateOfBirth As Date
    Dim Age As Integer
    'assign values to variables (赋值给变量)
    FullName = "John Smith"
    DateOfBirth = #01/03/1967#
    'calculate age (计算年龄)
    Age = Year(Now()) - Year(DateOfBirth)

    'print results to the Immediate window (在立即窗口里打印结果)
    Debug.Print FullName & " is " & Age & " years old."
End Sub
```

（译者：Debug是非常好的工具，它让对象在运行时将结果在立即窗口上显示）

变量在程序的开始部分就被声明了，从那里开始，它们就可以使用了。在上面的过程里，每个变量声明在分开的行。如果你想，你也可以同时在一行里声明好几个变量，用逗号分开每个变量，例如：

```
Dim FullName As String, DateOfBirth As Date, Age As Integer
```

注意，关键字Dim只在变量声明行的开头出现了一次。

当VB执行变量声明语句时，它产生了有确切名称的变量，并且占用内存空间来储存它们的值，然后，明确的值被赋给这些变量。如何给变量赋值？变量名称，之后是一个等号，等号的右边是你希望用该变量储存的数据。这里你输入的数据必须是该变量声明的数据类型。文本数据应该使用引号包括起来，而日期需要用井号#包括起来。VB使用DateOfBirth提供的数据来计算员工的年龄，并且将计算结果储存在Age这个变量。员工的姓名和年龄通过指令Debug.Print打印到立即窗口。当程序运行结束后，你必须打开立即窗口来查看结果。

我们来看看你声明了错误的数据类型，会发生什么情况。下面的过程是计算一个工作表里的总单元格数目，并且将结果使用一个对话框显示给用户。

```
Sub HowManyCells( )
```



```
Dim NumOfCells As Integer
NumOfCells = Cells.Count
MsgBox "The worksheet has " & NumOfCells & " cells."
```

```
End Sub
```

错误的数据类型会导致错误。在上面的过程里，当VB尝试将Cells.Count语句的结果赋给变量NumOfCells时失败，Excel显示信息“运行时间错误6——溢出”。这个错误的产生原因时变量的无效数据类型，工作表里单元格总数目不在Integer数据范围之内。要更正这个问题，你应该选择一个可以包含一个大数据的数据类型。然而，要快速地解决上面程序遇到的问题，你只要删除变量类型As Integer就行了。当你重新运行这个过程时，VB将给你的变量分配为Variant类型，尽管Variant比其它变量类型使用更多的内存和降低程序运行速度（因为VB不得不做额外的工作区检查变量类型），但是，如果这是在一个简短的程序里，使用Variant的代价也是难以觉察的。

技巧3-6 变量类型是什么？

通过下述方法，你可以快速地查明你程序里使用的变量的类型：在变量名称上单击右键，并且从快捷菜单上选择“快速信息”。

技巧3-7 串联

你可以将两个或多个字符串结合成为一个新的字符串。这个操作称为串联。你已经在AgeCalc和HowManyCells过程里看到了串联的例子。串联用&符号在表示。例如，“His name is ” & FirstName将会产生下述字符串：His name is John，或者His name is Michael。人名取决于变量FirstName的内容。注意，在is和结束引号之间有一个空格。字符串的串联也可以使用加号(+)来代表，然而，许多程序员为了消除混淆，宁愿将加号限制于数字的运算。

6 明确变量的数据类型

如果你在Dim语句里没有明确变量的数据类型，你最终将得到没有归类的变量。没有归类的包括，在VBA里，总是当成Variant数据类型。高度建议你产生归类了的变量。当你声明变量为某种数据类型，你的VBA程序会运行得更快一些，因为VB不需要停下来分析Variant变量到底是什么类型。

VB可以使用很多种数字变量。Integer变量只能保存从-32,768到32,767之间的所有整数。其它类型的数字变量有Long，Single，Double和Currency。Long变量可以保存从-2,147,483,648到2,147,483,647范围的所有整数。与Integer和Long相反，Single和Double变量可以保存小数。String变量用来引用文本。当你声明了一个String数据类型的变量时，你最好告诉VB这个字符串有多长，例如：

```
Dim extension As String * 3
```

声明变量extension字符串的长度为3个字符。如果你不给它分配一个明确的长度，这个字符串变量将是动态的。这意味着VB将会占用足够大的电脑内存来处理任意容量的文本。声明了变量后，你只能保存声明语句里显示的信息类型。给数字类型的变量赋文本值，或给文本类型变量赋数字值，都会导致“类型不匹配”的错误信息，或者导致VB修正该值。例如，如果你的变量声明为保存整数，而你的数据是小数，那么VB会忽略小数部分而只用数据的整数部分。试验一下下面的MyNumber过程，看看VB是如何修正数据以适合变量数据类型的：

```
Sub MyNumber()
    Dim myNum As Integer

    myNum = 23.11
    MsgBox myNum
End Sub
```

如果你不用Dim语句声明变量，你通过在变量名称后面加上一个特殊字符同样可以指明该变量的类型。例如下面，你可以在变量名称后面附上美元(\$)符号，来指明变量FirstName为字符串类型(String)：

```
Dim FirstName$
```

上面的声明和Dim FirstName As String是一样的。其它类型的声明字符列在表3-2里面。

表3-2 类型声明字符

数据类型	字符
Integer	%
Long	&
Single	!

Double	#
Currency	@
String	\$

注意，类型声明字符只能用于六种数据类型。将这些字符附在变量名称后面就可以使用这些类型声明字符了。过程AgeCalc2示范表3-2中类型声明字符的使用情况：

```
Sub AgeCalc2()
    'variable declaration (变量声明)
    Dim FullName$
    DateOfBirth As Date
    Dim Age%
    'assign values to variables (给变量赋值)
    FullName$ = "John Smith"
    DateOfBirth = #01/03/1967#
    'calculate age (计算年龄)
    Age% = Year(Now()) - Year(DateOfBirth)

    'print results to the Immediate window (在立即窗口里输出结果)
    Debug.Print FullName$ & " is " & Age% & " years old."
End Sub
```

技巧3-8 声明变量类型

变量类型可以用As后面的关键字来标示，也可以用后面附加的类型符号来标示。如果你既不加类型符号也不使用As命令，那么这个变量将为默认的类型，那就是VBA中的Variant类型。

7 变量赋值

既然你已经知道如何命名和声明变量了，是时候开始使用它们了。我们以学习如何创建变量开始。在VB中，你可以在你程序的任何地方创建变量，只有给它赋个值就行。

1. 打开一个新工作簿并且保存为Chap03.xls
2. 激活VB编辑器窗口
3. 在工程浏览器窗口，选择这个新的工程并在属性窗口里将它的名称改为Chap03
4. 选择“插入”－“模块”在工程Chap03里面添加一个新模块
5. 在属性窗口将该模块名Module1改为Variables
6. 在代码窗口，输入CalcCost过程，如下面所示。这个过程基于下述假设来计算购买一个计算器的价钱：计算器的价格为35美元，销售税为8.5%

```
Sub CalcCost()
    slsPrice = 35
    slsTax = 0.085
    Range("A1").Formula = "The cost of calculator"
    Range("A4").Formula = "Price"
    Range("B4").Formula = slsPrice
    Range("A5").Formula = "Sales Tax"
    Range("A6").Formula = "Cost"
    Range("B5").Formula = slsPrice * slsTax

    Cost = slsPrice + (slsPrice * slsTax)
    With Range("B6")
        .Formula = Cost
        .NumberFormat = "0.00"
    End With
End Sub
```

```
End With
strMsg = "The calculator total is " & "$" & Cost & "."
Range("A8").Formula = strMsg
```

End Sub

过程CalcCost使用了四个变量：slsPrice, slsTax, Cost和strMsg。因为这些变量都没有显式声明，所以它们的数据类型都是Variant。变量slsPrice和slsTax是在过程的开始时通过给它们赋值而产生的，变量Cost分配的值是下面计算的结果：slsPrice + (slsPrice * slsTax)。价格的计算是使用变量slsPrice和slsTax提供的值来进行的。变量strMsg将信息合并为一个文本信息给用户，然后这个信息是在工作表的一个单元格里输入一个完整的句子。

当你给变量赋值时，需要在变量名称后面输入一个等号，等号之后是你要输入的值。它可以是数字，公式或者带引号的文本。赋给变量slsPrice, slsTax和Cost的值比较容易理解，然而保存在变量strMsg的值则有些棘手。我来解释一下变量strMsg的内容吧。

```
strMsg = "The calculator total is " & "$" & Cost & "."
```

- 字符串“The calculator total is ”被引号包括起来了，注意，后面的引号前有个空格。
 - 字符串&让你将一个字符串附加在另一个字符串或者变量的内容后面
 - 在引号里面的美元符合（“\$”）用来表明货币类型。因为美元符合是字符，它需要用引号来包括起来
 - 字符串&必须用于每次你要在前面的字符串后加新信息的时候
 - 变量Cost是一个占位符，当过程运行时，计算器的实际价格将显示在这儿
 - 字符串&可以连接任何字符串
 - 句号用引号包括起来。当你需要在句子后面加句号时，如果它是在一个变量后面时，你必须单独再在后面加上它。
- 现在来运行它，将光标放在过程CalcCost的任何地方，并且选择“运行”－“运行宏”

技巧3—9 变量初始化

VB创建变量的时候就将其初始化了。变量假定为它们的默认值，数字型变量设置为0，布尔型变量初始化为False，字符串变量设置为空字符串（""），已经日期型变量则设置为1899年12月30日

注意，你在运行这个过程时，VB可能会弹出下面的信息：“编译错误：变量未定义”。如果这个情况发生了，点击确定以关闭这个信息框。VB将会选中变量slsPrice并且加亮过程名称Sub CalcCost，标题栏则显示“Microsoft Visual Basic－Chap03. xls [中断]”。VB中断模式允许你在继续之前更正错误。在本书的后面，你将学习如何在中断模式下解决问题。就现在而言，如果你遇到上面提及的错误时，通过选择“运行”－“重新设置”来退出中断模式；接下来，在代码窗口的上面删除显示在第一行的语句Option Explicit。Option Explicit语句意味着在本模块里使用的所有变量都必须经过正式声明。你将在下一节里学习这个语句。删除Option Explicit语句后，重新运行该过程，运行后，切换到Excel界面，过程运行的结果应该和图3—1一致。

	A	B	C
1	The cost of calculator		
2			
3			
4	Price	35	
5	Sales Tax	2.975	
6	Cost	37.98	
7			
8	The calculator total is \$37.975.		

图3—1 VBA过程可以在工作表里输入数据并计算结果

单元格A8显示变量strMsg的内容。注意，在单元格B6里面输入的价格有两位小数，而strMsg的价格却显示三位小数。要在单元格A8里显示带两位小数的计算器价格，你必须给变量Cost设置需要的格式，而不是给该单元格设置格式。VBA有专门的函数让你改变数据格式，你将使用Format函数来改变变量Cost的格式。该函数的语法是：

Format(expression, format)

Expression（表达式）是你要设置格式的值或者变量；format（格式）则是你要使用的格式类型。

1. 在CalcCost过程里更改变量Cost的计算:

```
Cost = Format(slsPrice + (slsPrice * slsTax), "0.00")
```

2. 将With...End With代码块取代为下述指令:

```
Range("B6").Formula = Cost
```

3. 将语句Range("B5").Formula = slsPrice * slsTax改为下面指令:

```
Range("B5").Formula = Format((slsPrice * slsTax), "0.00")
```

4. 重新运行修改后的程序

试验过程CalcCost之后,你可能会困惑,为什么我们要为声明变量烦恼,如果VB自己可以处理未声明的变量的话?因为过程CalcCost是如此之短,因此你不必担心VB每次使用这些Variant变量时会占用多少内存。然而,在短的过程中,内存问题不重要,但是当你输入变量名称时,你很可能出现错误。当你第二次使用Cost变量时,你忽略了“o”而写成“Cst”,后果会如何呢?

```
Range("B6").Formula = Cst
```

如果你使用了Tax在下面的公式中,而没有用slsTax,结果你将得到什么呢?

```
Cost = Format(slsPrice + (slsPrice * Tax), "0.00")
```

引入上面提及的错误后过程CalcCost的结果显示在图3—2。

	A	B	C
1	The cost of calculator		
2			
3			
4	Price	35	
5	Sales Tax	2.98	
6	Cost		
7			
8	The calculator total is \$35.00.		

图3—2 变量名称错误导致结果错误

注意,在图3—2里,单元格B6没有数值,因为VB没有找到变量Cst的任务语句。再因为VB不知道销售税,显示的计算器价格为总价(而没有加税金,见单元格A8)。VB不会猜测,它只是简单地做你告诉它的事情。这带我们到下一个部分,解释如何避免这类错误。在你继续之前,确保更正变量Cst和Tax为Cost和slsTax。

8 强制声明变量

VB使用Option Explicit语句自动提醒你正式地声明你的变量,这个语句必须放在每个模块的最上面。如果你试图运行一个含有未定义的变量的过程时,Option Explicit语句会让VB产生一个错误信息。

1. 返回代码窗口你输入过程CalcCost的地方
2. 在模块的最上面输入Option Explicit并回车,Excel将该语句显示为蓝色
3. 运行过程CalcCost,VB显示错误信息“编译错误:变量未定义”
4. 点击确定关闭信息框。

VB加亮变量名称slsPrice。现在你需要正式声明这个变量。当你声明了变量slsPrice并再次运行该过程时,VB一旦遇到另外一个未声明的变量时,将再次产生同样的错误。

5. 在CalcCost过程的开始部分输入下述声明:

```
'declaration of variables (声明变量)
Dim slsPrice as Currency
Dim slsTax as Single
Dim Cost as Currency
Dim strMsg as String
```

6. 按下F5来运行该过程,修改后的程序显示如下“

```
Option Explicit
```

```

Sub CalcCost()
' declaration of variables
    Dim slsPrice As Currency
    Dim slsTax As Single
    Dim Cost As Currency
    Dim strMsg As String

    slsPrice = 35
    slsTax = 0.085

    Range("A1").Formula = "The cost of calculator"
    Range("A4").Formula = "Price"
    Range("B4").Formula = slsPrice
    Range("A5").Formula = "Sales Tax"
    Range("A6").Formula = "Cost"
    Range("B5").Formula = Format((slsPrice * slsTax), "0.00")

    Cost = Format(slsPrice + (slsPrice * slsTax), "0.00")

    With Range("B6").Formula = Cost

    strMsg = "The calculator total is " & "$" & Cost & "."
    Range("A8").Formula = strMsg
    End Sub

```

在模块上面输入的Option Explicit语句强迫你声明变量，因为如果你需要声明变量的话，你必须每次在添加新模块时输入Option Explicit语句，所以你可以让VB帮你输入它。按照下述步骤来自动在新模块里添加Option Explicit语句：

1. 选择“工具”－“选项”
2. 确保选项对话框（编辑页）上的“要求变量声明”被勾选上了
3. 点击确定关闭对话框

从现在开始，每个新模块都会在第一行添加Option Explicit语句。如果你要在一起创建的模块里强制声明变量的话，你必须手动输入Option Explicit。

技巧3—10 Option Explicit更多信息

Option Explicit强迫正式（显式）声明模块里的所有变量。使用它的一个重大优点是，输入错误会在编译时（VB试图将源代码翻译为可执行代码）被检测到。Option Explicit语句必须在模块里的任何过程之前出现。

9 了解变量范围

不同的变量在VBA过程里有不同的影响范围。Scope（范围）这个术语定义某个特定的变量在同一个过程，其它过程，或者其它VBA工程里的可用性。变量在VBA里可以是下面三种级别的范围：

- 过程级别范围
- 模块级别范围
- 工程级别范围

10 过程级别（当地）变量

从本章起，你已经知道如何通过关键字Dim来声明变量，在关键字Dim在模块中的位置决定了该变量的范围。在VBA过程中用Dim关键字声明的变量拥有过程级别的范围，过程级别的变量经常被称为当地变量。当地变量只能使用在声明它的过程里面。未

声明的变量总是过程级别的变量。在它的范围内，变量的名称必须是唯一的，这意味着你不可在同一个过程里使用同样的名称来声明两个变量。然而，你可以在不同的过程里面使用同样的变量名称。换句话说，过程CalcCost里可以有slsTax变量，同一个模块里的过程ExpenseRep里页可以有它自己的变量slsTax，两个变量相互独立。

11 模块级别变量

当地变量有助于节省电脑内存，一旦该过程结束，该变量便立即消失，并且电脑释放该变量占用的内存空间。然而，在编程中，你经常需要变量在本过程结束后仍然在其它过程里可用，这种情形需要你改变变量的范围。你可能就需要定义一个模块级别的变量，而不是过程级别的了。要定义模块级别的变量的话，你必须将关键字Dim放在模块表里任何过程的上面（紧接着在关键字Option Explicit的下面）。例如，将slsTax设置为任何Variables模块里的过程都可以使用，按照下述方法声明slsTax变量：

```
Option Explicit
```

```
Dim slsTax As Single
```

```
Sub CalcCost()  
    <放置过程指令>
```

```
End Sub
```

在上面的例子里，关键字Dim在模块的上面，紧挨着Option Explicit语句。你需要另外一个过程来使用变量slsTax，这样你才能查看这是如何工作的。

1. 在代码窗口里，将变量声明行Dim slsTax As Single从Variables模块的CalcCost过程里剪切，并且粘贴到该模块的上面Option Explicit语句的下面
2. 在CalcCost过程的同一个模块里输入ExpenseRep过程代码：

```
Sub ExpenseRep()  
    Dim slsPrice As Currency  
    Dim Cost As Currency  
  
    slsPrice = 55.99  
    Cost = slsPrice + (slsPrice * slsTax)  
  
    MsgBox slsTax  
    MsgBox Cost  
End Sub
```

ExpenseRep过程里声明了两个货币类型的变量：slsPrice和Cost。slsPrice变量随后赋值55.99，slsPrice和CalcCost过程里声明的变量slsPrice是独立工作的。ExpenseRep过程计算采购的费用，该费用包括销售税，因为销售税和CalcCost过程里使用的是一样的，所以将slsTax变量声明为模块级别的变量。VB执行CalcCost过程后，变量slsTax的内容等于0.085。如果slsTax是当地变量的话，随着CalcCost过程的终结，变量slsTax的内容将被清空。过程ExpenseRep结束时显示两个信息框来输出变量slsTax和Cost的值。当你运行CalcCost后，VB清空除了slsTax之外的所有变量的内容，因为slsTax以被定义为模块级别的变量。一旦你试图通过运行ExpenseRep来计算价格，VB就会找到slsTax的值，并且将它用到计算中。

技巧3—11 私有变量

当你在模块级别声明变量时，除了关键字Dim之外，你还可以使用关键字Private。例如，

```
Private slsTax As Single
```

私有变量仅仅在声明该变量的模块里的过程中可用。私有变量总是在模块的上面Option Explicit语句之后声明。

技巧3—12 保持工程级别的变量为私有

为了避免工程级别的变量内容被工程之外使用，你可以在Option Explicit语句下面，模块的上面输入Option Private Module，例如：

```
Option Explicit  
Option Private Module  
Public slsTax As Single
```



```
Sub CalcCost( )
    <这里是过程代码>
End Sub
```

12 工程级别变量

模块级别的变量用关键字Public（而不是Dim）声明时，拥有工程级别范围。这意味着它们可以在VBA任何模块里使用。当你想要在一个打开的VBA工程的所有过程里使用某个变量时，必须用Public关键字来声明它，例如：

```
Option Explicit
Public sIsTax As Single
```

```
Sub CalcCost( )
    <过程代码>
End Sub
```

注意，变量sIsTax在模块上面以Public关键字声明的，它将会在该VBA工程里的任何过程里都可用。

13 变量的存活期

除了范围之外，变量还有存活期，变量的存活期决定了该变量能保存它的值有多久。一旦该工程打开，模块级别和工程级别的变量就会保留它们的值。然而，如果程序的逻辑需要，VB能够重新初始化这些变量。使用Dim语句声明的当地变量当过程结束时就会丢失值，当地变量的存活期是随着过程的运行的，并且它们在程序每次运行的时候可以被重新初始化。VB允许你通过改变声明方式延长当地变量的存活期。

14 了解和使用静态变量

用Static关键字声明的变量是特殊的当地变量，静态变量在过程级别声明。和那些用关键字Dim声明的当地变量相反，静态变量在程序已经不在它们的过程里时仍然不会丢失它们的内容。例如，当一个带有静态变量的VBA过程调用另外一个过程时，在VB执行完被调用的过程语句后返回主调过程时，静态变量仍然保留它原来的值。过程CostOfPurchase示范静态变量allPurchase的使用：

```
Sub CostOfPurchase()
    'declare variables
    Static allPurchase
    Dim newPurchase As String
    Dim purchCost As Single
    newPurchase = InputBox("Enter the cost of a purchase:")
    purchCost = CSng(newPurchase)
    allPurchase = allPurchase + purchCost

    'display results
    MsgBox "The cost of a new purchase is: " & newPurchase
    MsgBox "The running cost is: " & allPurchase
End Sub
```

上面的过程以一个名为allPurchase的静态变量和两个当地变量newPurchase和purchCost的声明开始。该过程中使用的InputBox函数显示一个对话框并且等着用户输入数值，一旦用户输入数值并且点击确定后，VB就会将该数值赋给变量newPurchase。在第四章中有InputBox函数的讨论。因为InputBox函数的结果总是字符串，变量newPurchase被声明为字符串数据类型了。然而，你不能在数学计算中使用字符串，这就是为什么需要在下一指令中使用一个类型转换函数（CSng）来将字符串值转换为单精度浮点类型的数字。函数CSng只需要一个自变量——你要转换的数值。函数CSng转换的数字结果保存在变量purchCost上。

技巧3—13 类型转换函数

在CSng上的任意地方按下F1，可以查看更多关于函数CSng的信息（也可以查看其它类型转换函数信息）

下一行指令：`allPurchase = allPurchase + purchCost`，将InputBox函数提供的新数值加和到目前的采购数值上。当你第一次运行这个过程的时候，变量`allPurchase`和变量`purchCost`的内容是一样的；当你第二次运行它的时候，这个静态变量的值由对话框提供的值增加了。你可以随意多次运行过程`CostOfPurchase`，只要该工程是开着的，变量`allPurchase`就会不断的变化。依照下述步骤来试验该过程：

1. 将光标放在过程`CostOfPurchase`里的任意地方并且按下F5
2. 当对话框出现时，输入一个数字，例如，输入100然后回车。VB显示信息““The cost of a new purchase is: 100.””
3. 点击确定，VB显示第二个信息“The running cost is: 100.”
4. 重新运行该程序，当对话框出现时，输入另外一个数字，例如输入50再回车。VB显示信息“The cost of a new purchase is 50.”
5. 点击确定，VB显示第二个信息“The running cost is: 150.”
6. 多次运行该程序，看看VB是如何追踪运行的总量的。

15 声明和使用对象变量

你已经学习的变量是用于储存数据的，储存数据是你使用“普通的”变量的主要原因。除了储存数据从普通变量之外，还有引用VB对象的特殊变量，这些变量称为对象变量。你在第二章中已经学习了多种对象，现在，你开始学习如何用对象变量来代表对象。对象变量不储存数据，它们告诉数据在哪儿。例如，你可以用对象变量告诉VB数据在当前工作表的单元格E10，对象变量使定位数据更容易。编写VB程序时，你经常需要写一些很长的指令，例如：

```
Worksheets("Sheet1").Range(Cells(1,1), Cells(10, 5)).Select
```

你可以声明一个对象变量来告诉VB数据在哪儿，而不必使用很长的指令来指向该对象。对象变量的声明和你已经学习的变量声明类似，唯一的不同是在关键字`As`后面，你输入词语`Object`作为数据类型，例如：

```
Dim myRange As Object
```

上面的语句声明了一个叫做`myRange`的对象变量。然而，实际上只声明对象变量是不够的，在使用这个变量于程序里之前，你还给这个对象变量赋上确定的值。使用关键字`Set`来给对象变量赋值，关键字`Set`后面是等号，再后面是该变量指向的值，例如：

```
Set myRange = Worksheets("Sheet1").Range(Cells(1,1), Cells(10, 5))
```

上面的语句给对象变量`myRange`赋值，这个值指向工作表`Sheet1`的单元格区域A1:E10。如果你忽略了关键字`Set`，VB将会显示一个错误信息——“运行时间错误91：对象变量或With块变量未设置”。

现在，又是来看看实例的时候了，下面的过程`UseObjVariable`示范一个叫`myRange`的对象变量的使用：

```
Sub UseObjVariable()  
    Dim myRange As Object  
    Set myRange = Worksheets("Sheet1").  
        Range(Cells(1, 1), Cells(10, 5))  
    myRange.BorderAround Weight:=xlMedium  
    With myRange.Interior  
        .ColorIndex = 6  
        .Pattern = xlSolid  
    End With  
    Set myRange = Worksheets("Sheet1").  
        Range(Cells(12, 5), Cells(12, 10))  
    myRange.Value = 54  
    Debug.Print IsObject(myRange)  
End Sub
```

我们来逐行分析一下过程`UseObjVariable`里的代码。过程开始的时候是对象变量`myRange`的声明，下一行将对象变量设置未`Sheet1`的区域A1:E10。从现在开始，你每次要引用这个区域时，你不需要写下整个地址，而只要使用这个捷径——该对象变量名称就可以了。这个过程的目的是在区域A1:E10外围设置边框，你不必使用下面这样冗长的指令：

```
Worksheets("Sheet1").Range(Cells(1, 1), _
```

```
Cells(10, 5).BorderAround Weight:=xlMedium
```

而可以使用一个捷径，使用对象变量名称：

```
myRange.BorderAround Weight:=xlMedium
```

下一节语句是将选区A1:E10设置底色。同样，你不需要使用你要操作的对象冗长地址，你可以使用简单的对象变量名称myRange。下一句代码是给对象变量myRange分配一个新的引用区域，VB将忘记老的引用，你下次使用myRange时，它会引用另一个区域E12:J12。在新区域（E12:J12）输入了54后，过程给你显示如何确定某个变量时对象类型。如果myRange是对象变量的话，指令Debug.Print IsObject(myRange)将在立即窗口里面输入True。IsObject是个VBA中指明某变量是否是对象变量的函数。

技巧3—14 使用对象变量的好处

使用对象变量的好处有：

- 它们可以代替真实对象使用
- 它们比真实对象更短更容易记住
- 当过程运行时，你可以改变它们的意义

16 使用明确的对象变量

对象变量可以引用任意一种对象，因为VB有很多种对象，所以，要让你的程序可读性更强，运行更快，最好创建引用到具体对象类型的对象变量。例如，在过程UseObjVariable中，你可以将myRange对象变量声明为Range对象，而不是通常的对象变量（Object）：

```
Dim myRange As Range
```

如果你要引用一个具体的工作表，你可以声明Worksheet对象：

```
Dim mySheet As Worksheet
```

```
Set mySheet = Worksheets("Marketing")
```

当对象变量不再需要时，你可以给它赋值Nothing，这将释放内存和系统资源：

```
Set mySheet = Nothing
```

你将在第九章里看到更多的使用对象变量的例子。

17 查找变量定义

当你在VBA过程里看到一行给变量赋值的指令时，你可以通过选择该变量名称并且按下Shift+F2，快速地定位到该变量的定义（声明）。或者，你也可以选择“视图”－“定义”，VB将跳到变量的声明行。要回到刚才的位置，只要按下Ctrl+Shift+F2或选择“视图”－“最后位置”。我们来试试：

1. 定位到过程CostOfPurchase的代码里
2. 定位到语句purchCost = CSng(newPurchase)
3. 在变量名称上单击右键，并在快捷菜单上选择“定义”
4. 通过按Ctrl+Shift+F2返回刚才位置
5. 试试在其它过程的其它变量上查找定义，每次使用不同的方法跳到声明位置。

技巧3—15 这个变量是什么类型？

你可以使用一个VB内置函数来查明变量的类型。参见第四章中使用函数VarType的例子。

18 在 VB 过程里面使用常量

当你的程序运行，变量的内容是可以变化的，如果你想要一次又一次地引用不变的值，那么你应该使用常量。常量就像一个指定的变量一样，总是引用这个相同的值。VB要求你在使用前要声明常量。正如下述例子，使用Const语句来声明常量：

```
Const dialogName = "Enter Data" As String
```

```
Const slsTax = 8.5
```

```
Const ColorIdx = 3
```

常量，象变量一样拥有范围。要让常量仅在一个过程里可用，将它声明为过程级别即可，例如：

```
Sub WedAnniv( )
```

```
    Const Age As Integer = 25
```

```
    <place procedure instructions here>
```

```
End Sub
```

如果你想要某个常量在一个模块的所以过程中都可用，则在Const语句前加上关键字Private就可（译者：写在所以过程之上），例如：

```
Private Const dsk = "B:" As String
```

私有常量必须在模块的上面，第一个Sub语句之上声明。

如果你要创建一个该工作簿所有模块都可用的常量时，在Const语句之前加上Public关键字就可以了，例如：

```
Public Const NumOfChar = 255 As Integer
```

公共常量必须在模块的上面，第一个Sub语句之上声明。

声明常量的时候，你可以使用下列数据类型之一：Boolean, Byte, Integer, Long, Currency, Single, Double, Date, String 或者Variant。

象变量一样，多个常量也可以在一行里声明，例如：

```
Const Age As Integer = 25, City As String = "Denver", PayCheck As Currency = 350
```

使用常量可以使你的VBA过程可读性强，容易维护。例如，你在程序里多次引用某个特定值，就可以使用常量，而不是这个值本身。这样，如果以后这个值变了（例如销售税率上升了），你只要简单地在Const语句里改变这个常量的声明就可以了，而不必追踪该值所有发生的地方。

19 内置常量

Excel和VBA都有一长列的预先定义的常量，并且不需要声明，这些内置常量可以通过对象浏览器查找，我们已经在第二章里详细讨论了对象浏览器。我们来打开对象浏览器并查找Excel常量清单：

1. 在VB编辑器窗口，选择“视图”－“对象浏览器”
2. 在“工程/库”下拉列表里选择Excel
3. 在搜索文本框里输入“Constants”并回车，VB显示搜索结果在“搜索结果”区域
4. 在“类”列表框拉下滚动条，选择“Constants”（参见图3-3）。对象浏览器右边区域显示所有Excel对象库里可用的内置常量。注意，所有常量的名称以前缀“xl”开头。

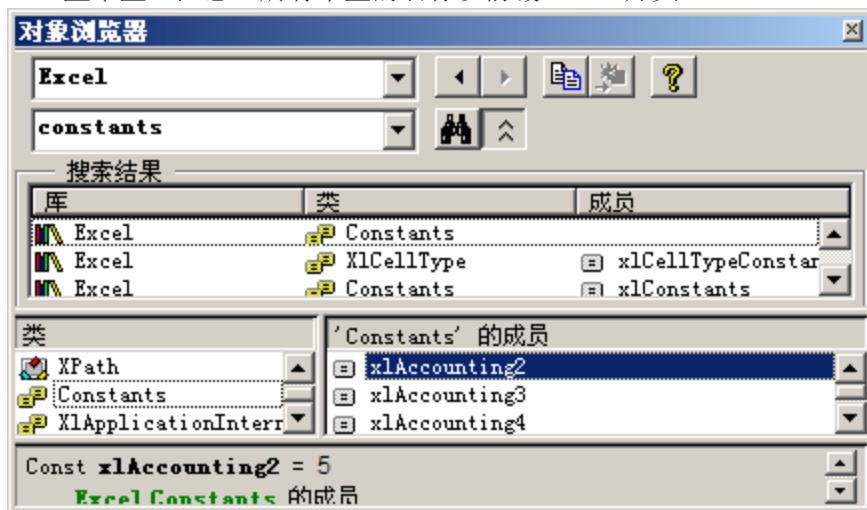


图3-3 使用对象浏览器查找内置常量

5. 要查找VBA常量，在工程/库文本框里输入VBA。注意，所有VBA的内置常量以前缀“vb”开头。学习内置常量的最好方法是使用宏录制器，我们来花上几分钟来录制最小化当前窗口的过程：

1. 在Excel窗口，选择“工具”－“宏”－“录制新宏”
2. 输入MiniWindow作为宏名，选择“当前工作簿”作为保存宏的地方，然后确定
3. 点击最小化按钮，确保你最小化了本文件的窗口，而不是Excel应用程序窗口
4. 点击“停止录制”按钮
5. 最大化该被最小化了的窗口
6. 切换到VB编辑器窗口，并且双击模块文件夹。代码显示如下：

```
Sub MiniWindow()  
    ActiveWindow.WindowState = xlMinimized  
End Sub
```

你有时可能会看到VBA过程使用数值，而不是内置常量名称，例如，常量xlMaximized的实际数值是-4137，常量xlMinimized的值是-4140，以及常量xlNormal的值为-4143（参见图3-4）

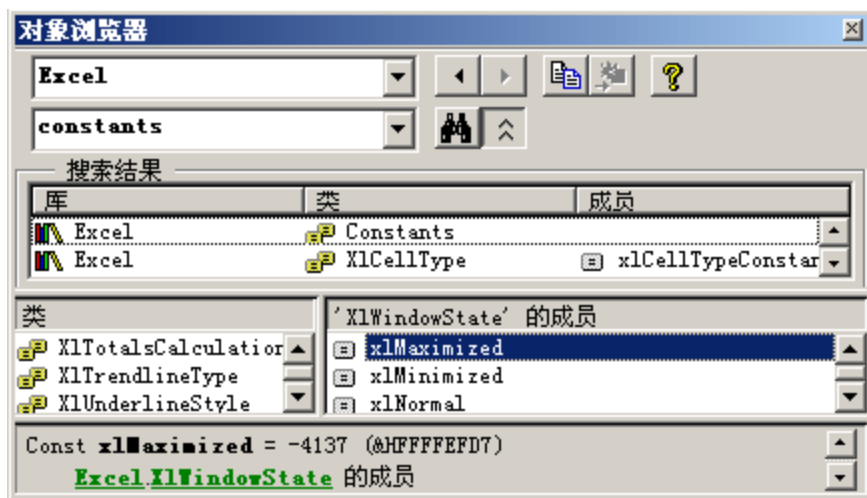


图3-4 你可以在对象浏览器里面选择常量名称然后下面的窗口里看到它的实际值

20 接下来……

本章介绍了几个VBA概念，包括数据类型，变量和常量。你学习了如何声明各种变量，也看到变量和常量之间的区别。既然你知道什么是变量，也知道如何使用它们，你就可以创建使用比前两章更有意义的方法操作数据的过程了。在下一章中，你将通过使用带有自变量和函数的过程来扩展你的VBA知识，另外，你还将学习函数，让你的VBA过程与用户进行交流。

第四章 VBA 过程：子程序和函数

作者: Julitta Korol 翻译: Tiger Chen Jan 16' 2005

在第二章中，你知道了过程是一组指令，它让你在程序运行的时候完成一些具体的任务。VBA有以下三种过程：

- **子程序过程**（子程序）执行一些有用的任务但是不返回任何值。它们以关键字Sub开头和关键字End Sub结束。子程序可以用宏录制器（第一章）录制或者在VB编辑器窗口里直接编写（见第二章和第三章）。在你第一章里已经学习了多种运行这种过程的方法。
- **函数过程**（函数）执行具体任务并返回值。它们以关键字Function开头和关键字End Function结束。在本章中，你将创建你的第一个函数过程。函数过程可以从子程序里执行，也可以从工作表里访问，就像Excel的内置函数一样。
- **属性过程**用于自定义对象。使用属性过程你可以设置和获取对象属性的值，或者设置对另外一个对象的引用。你将在第十一章中学习如何创建自定义对象和使用属性过程。

在本章中，你将学习如何创建和执行自定义函数，另外，你将发现变量（见第三章）如何用于传递数据给子程序和函数。在本章后面，你将对VBA中两种最有用的函数：MsgBox和InputBox进行比较彻底的了解。

1.关于函数过程

使用Excel几百种内置函数，你可以进行非常广泛的自动计算，然而，你总有要做个自定义计算的时候。使用VBA编程，你可以通过创建函数过程快速的完成这个特殊需求，你可以创建任何Excel没有提供的函数。

2.创建函数过程

象Excel函数一样，函数过程进行计算并返回数值。学习函数的最好方法就是自己创建一个。因此，我们开始吧。设置完一个新的VBA工程后，你将创建一个函数过程来加和两个数值。

1. 打开一个新Excel工作簿，并保存为Chap04.xls
2. 切换到VB编辑器窗口并且选择VBAProject (Chap04.xls)
3. 在属性窗口，将VBAProject改为MyFunctions
4. 在工程浏览器窗口选择MyFunctions (Chap04.xls)，然后选择“插入”－“模块”
5. 在属性窗口将“模块1”改为Sample1
6. 在工程浏览器窗口，点击Sample1并选择“插入”－“过程”（译者：需要激活右边的代码窗口）。添加过程对话框如图4－1所示

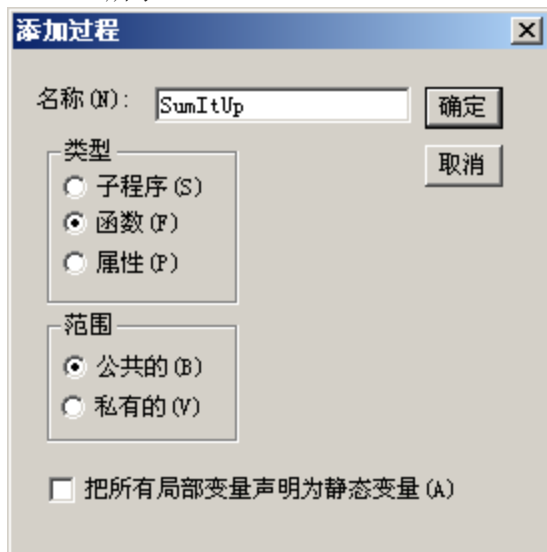


图4-1 你使用添加过程对话框时，VB自动创建你选择的过程类型

7. 在对话框里输入下列设置：
名称：SumItUp
类型：函数
范围：公共的
8. 点击确定退出添加过程对话框。VB输入了一个空函数过程如下：

```
Public Function SumItUp()
```

```
End Function
```

第一句声明函数过程名称，关键字Public表明这个函数可以在所有模块的所有过程里访问。关键字Public是可选的。注意，关键字Function后面是函数名称（SumItUp）和一对空括号。在括号里你可以列上计算中需要的数据项目。每个函数过程都以End Function语句结束。

技巧4-1 关于函数名称

函数名称应该点明该函数的作用，并且必须和变量的命名规则一致。

技巧4-2 设置VBA过程范围

在前几章你学习了变量的范围决定它可以在哪些模块和过程里使用，和变量一样，VBA过程也有范围。过程的范围决定其它模块里的过程是否可以调用该过程。所以的VBA过程默认为公共的，这意味着它可以被任何模块里的其它过程调用。因为过程默认为公共的，所以如果你愿意你可以忽略关键字Public。但是，如果你将Public关键字换成关键字Private，那么你的过程只能被同一模块里的其它过程调用，而不能被其它模块里的过程调用。

9. 将函数声明修改为这样：

```
Public Function SumItUp(m,n)
```

```
End Function
```

这个函数的目的是加和两个数值。不要将实际值输给函数。给该函数提供两个自变量以确保该函数具有灵活性。这样，你的自定义函数就能够将你提供的任何两个数值加和起来了。每个变量代表一个数值，你在运行该函数时要给每个变量提供数值。

技巧4-3 使用函数的理由

自定义VBA函数可以用于：

- 分析数据和进行计算
- 修正数据和汇报信息
- 基于提供的或计算的数据采取具体行动

10. 在Public Function和End Function之间输入下述语句：

```
SumItUp = m + n
```

这条语句意思是将储存于变量n上的数据加在储存于变量m的数值上，并且将结果返回给函数SumItUp。在等号后面输入该函数名称，再就是括号和需要加和的数值。在上面的语句中，设置函数名称等于m + n的和。完成的自定义函数过程如下：

```
Public Function SumItUp(m, n)
```

```
SumItUp = m + n
```

```
End Function
```

祝贺，你已经创建了你的第一个函数！然而，函数过程并没有什么用，除非你知道如何执行它。下一个段落将给你示范如何使你的函数工作。

3. 执行函数过程

在第一章里，你学习了很多中方法来运行子程序。和子程序不同，函数过程只能使用两种方法来执行。你可以用于工作表的公式里面，或者从另外一个过程里调用它。你在VBA里创建的函数过程，不能通过在Excel窗口选择“工具”－“宏”－“运行宏”访问；它们也不能在代码里面通过按F5键来运行。接下来的部分里，你将学习到执行函数的专门技术。

4. 从工作表里运行函数过程

自定义函数过程和内置函数一样，如果你不知道确切的函数名称或者它的自变量，那么你可以使用“插入函数”对话框来帮助你在工作表里输入需要的函数。

1. 切换到Excel窗口，并选择任何一个单元格
2. 点击函数工具栏上的“插入函数 (fx)”按钮（译者：或者选择“插入”－“函数”），Excel弹出插入函数对话框，上面显示了所选类别里所有函数，按字母顺序排列
3. 在类别下拉框里选择“全部”或者“用户定义”，然后滚动函数名称框，找到并选择本章中创建的函数SumItUp。当你选中这个函数名称时，在插入函数对话框的下部显示了该函数的语法：SumItUp(m, n)（译者：同时可以看到“没有帮助信息”。请参阅创建自定义函数的相关教材，如何给自定义函数提供帮助信息）

技巧4-4 私有函数用户是看不到的

使用关键字Private声明的函数不会出现在“插入函数”对话框上，私有函数不能用于公式里，它们只能从另一个VBA过程里调用。

技巧4-5 快速访问自定义函数

一旦你创建了一个公共的VBA函数，Excel就会将它加入到“插入函数”对话框的“用户定义”的类别里。通过选择这个类别，你可以快速地访问该自定义函数。

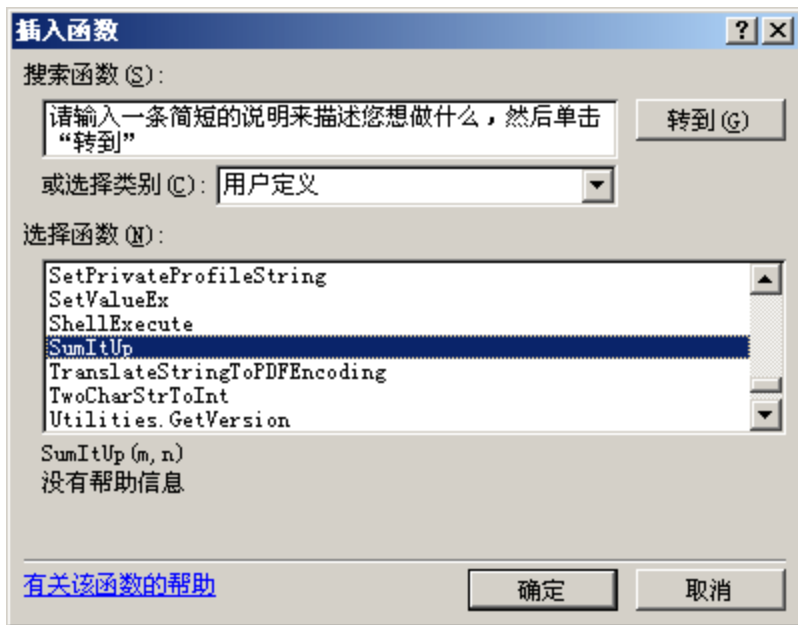


图4-2 VBA自定义函数出现在Excel内置函数同一清单上

4. 点击确定，开始写公式。“函数参数”对话框出现，如下图所示。对话框显示了函数名称和每个参数：m和n

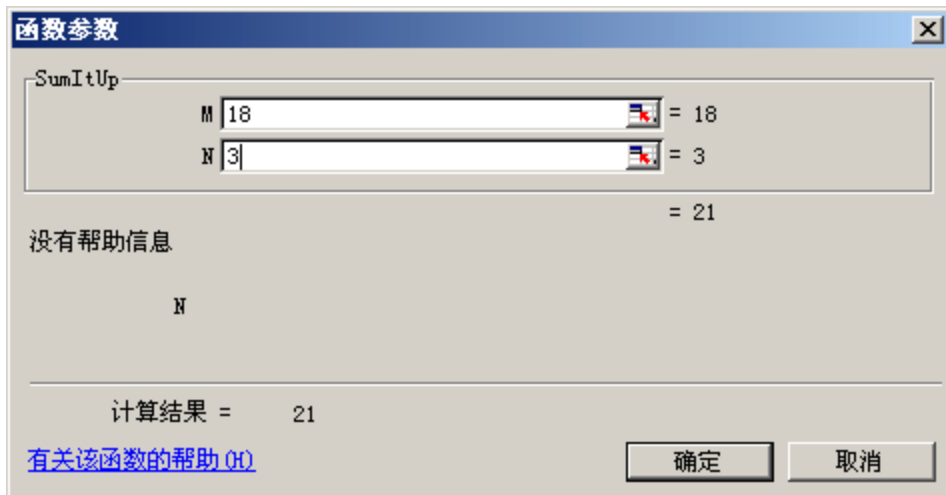


图4-3 公式的调色板功能有助于输入任何工作表函数，不论是内置或VBA编程的自定义函数

5. 如图4-3所示输入自变量数值，或者你任意输入数值。当你输入数值在参数文本框时，Excel显示你输入的数和当前的函数结果。因为两个参数（m和n）都是必须的，所以，如果你忽略了其中的某个时，函数会返回错误。
6. 点击确定退出函数参数对话框，Excel输入函数SumItUp在所选的单元格里，并且显示它的结果。要编辑这个公式的话，选择该显示公式结果的单元格，并且点击“插入函数”按钮，选择函数并点击确定以访问函数参数对话框。在函数参数m和n上输入不同的数值，并点击确定。也可以直接在该单元格上双击，修改函数参数值。

技巧4-6 确保你的自定义函数可用

只有在你储存该自定义函数的工作簿开启的时候，你的自定义VBA函数才可用，如果你关闭该工作簿，该函数便不再可用。要确保你的自定义函数每次在你使用Excel时都能用到，你可以做下述事情之一：

- 保存你的函数在个人宏工作簿
- 将含有你的自定义函数的工作簿保存在XLStart文件夹里
- 创建引用到含有该自定义函数的工作簿（请参见第二章，如何创建对另一个工程的引用）

5.从另外一个 VBA 过程里运行函数过程

正如前面提到的，你不能在VB窗口将光标放在代码里并且按F5来运行函数，也不能通过选择“运行”—“运行宏”来运行函数过程。要运行函数，你必须从另外一个过程里调用该函数。要执行自定义函数，编写一个VBA子程序并且在需要的时候调用该函数。下面的过程调用函数SumItUp并且将计算结果输出在立即窗口：

```
Sub RunSumItUp()  
    Dim m As Single, n As Single  
    m = 370000  
    n = 3459.77  
    Debug.Print SumItUp(m,n)  
    MsgBox "Open the Immediate window to see the result."  
End Sub
```

- 上面的子程序使用Dim语句声明变量m和n，它们用来给函数提供数据
- 接下来两行语句给变量赋值
- 再下来，VB调用函数SumItUp并且将储存于变量m和n的数值传递给它。当函数过程里SumItUp = m + n语句执行完后，VB返回到子过程RunSumItUp里面，并且使用Debug.Print语句将函数的结果输出在立即窗口
- MsgBox函数通知用户在哪里查看结果

依照下述步骤来试验上面的例子：

1. 在输入函数SumItUp的同一个模块里面输入过程RunSumItUp
2. 将光标放在该过程的任意地方，按下F5

技巧4—7 函数的快速测试

你编写自定义函数后，你可以在立即窗口快速的测试它。打开立即窗口，输入一个问号（?）在函数名称前，可以显示该函数的计算结果。记住，要在括号里输入函数的参数值。例如，输入：

? SumItUp(54, 367.24)

然后回车。你的函数使用参数m和n传递的数值进行计算，函数的结果显示在下一行：

421.24

6.传递参数

到目前为止，你已经创建了简单的可以执行具体任务的VBA过程，这些过程在它们运行前没有要求你提供额外的数据。然而，在现实生活中，过程（子程序和函数）经常需要参数。参数（自变量）是过程工作时需要的一个或多个数值。参数通常输入在括号之间，多个参数用逗号分割开来。使用Excel有一阵了，你已经知道Excel内置函数根据你提供的数据可能产生不同的结果。例如，如果单元格A4和A5分别含有数字5和10，加和公式=SUM(A4:A5)将会返回15，除非你更改单元格里面的数值。正如你可以传递数值给Excel内置函数，你也可以传递数值给自定义VBA过程。现在，我们来看看如何从子程序传递数值给函数SumItUp。这个自定义函数目的是计算一个人的姓和名的字母数目。

1. 在放置函数SumItUp的模块里输入下列子程序NumOfCharacters：

```
Sub NumOfCharacters()  
    Dim f As Integer  
    Dim l As Integer  
  
    f = Len(InputBox("Enter first name:"))  
    l = Len(InputBox("Enter last name:"))  
    MsgBox SumItUp(f,l)  
End Sub
```

2. 将光标放在过程NumOfCharacters的任意地方并按下F5。VB将显示信息输入框问你名字，这个信息框由下面的函数产生：InputBox("Enter first name:")
3. 输入任何名字，回车，VB接受你输入的文本并且将它作为一个参数传递给函数Len。函数Len计算提供的文本的字母数目。VB将函数Len的结果储存于变量f以供以后使用。这之后，VB显示下一个信息框，这次是问你的姓。
4. 输入任何姓，回车。VB将输入的姓传递给函数Len来获得姓的文本长度，然后数值储存于变量l。接下来发生什么呢？VB

遇到了函数MsgBox，这个函数告诉VB显示函数SumItUp的结果。然而，因为这个结果还没有准备好，VB很快就跳到函数SumItUp里，使用变量f和l储存的数值来做计算。在函数内部，VB用变量f的值取代参数m和变量l值取代参数n。一旦取代工作完成，VB将两个数值加和起来，并且将结果返回给函数SumItUp。函数SumItUp内部没有其它的任务了，因此，VB又马上返回子程序并且将函数SumItUp的结果作为一个参数提供给函数MsgBox。现在这个信息出现在屏幕上，显示了字母总数目。

5. 点击确定退出信息框，你可以多次运行过程NumOfCharacters，每次输入不同的姓名。

我们来看看另外一个使用变量传递参数的例子：

1. 在工程MyFunctions (Chap04. xls) 里添加一个新模块，并重命名为Sample2
2. 激活模块Sample2并且输入子程序EnterText：

```
Sub EnterText()  
    Dim m As String, n As String, r As String  
    m = InputBox("Enter your first name:")  
    n = InputBox("Enter your last name:")  
    r = JoinText(m, n)  
    MsgBox r  
End Sub
```

3. 输入下述函数过程：

```
Function JoinText(k, o)  
    JoinText = k + " " + o  
End Function
```

4. 运行过程EnterText

VB执行语句时，它收集用户输入的数据并且将这些数据储存在变量m和n上，然后传递这些数据到函数JoinText。VB将这些变量的值取代函数JoinText的参数，并且将结果赋到函数名称（JoinText）上。当VB返回过程EnterText时，函数的值储存于变量r。然后，函数MsgBox在信息框里显示变量r的内容。结果就是用户的完整姓名（姓和名用空格分开）。要从函数传递数值给子程序，需要将该值赋到函数名称，例如，下面的函数NumOfDays将值7传递到子程序DaysInAWeek：

```
Function NumOfDays()  
    NumOfDays = 7  
End Function  
Sub DaysInAWeek()  
    MsgBox "There are " & NumOfDays & " days in a week."  
End Sub
```

技巧4—8 函数过程不能做什么

函数过程不能进行任何动作，例如，它们不能在工作表里做插入，删除或设置数据格式操作，不能打开文件，或改变屏幕显示样式

7.明确参数类型

在前面的部分，你学习了函数根据它们自变量传递的值进行计算，当你声明函数时，你将参数名称列在一对括号里面。参数名称就像变量一样，每个参数名称在函数调用时，引用你提供的数值。当子程序调用函数过程时，它以变量形式传递参数。一旦函数做点什么，它就将结果赋给函数名称。注意，函数过程的名称当作变量来使用。

象变量一样，函数也有类型，函数的结果可以是字符串型，整型，长整型，等。要明确你函数的结果类型，只有在函数声明行后添加关键字As和你想要的类型即可。例如：

```
Function MultiplyIt(num1, num2) As Integer
```

如果你不明确数据类型，VB将把你的函数结果设置为默认类型（Variant数据类型）。当你明确你的函数结果的数据类型时，就象你明确变量的数据类型那样有一些好处，你的程序将更有效地使用内存，因此它运行得也更快些。

我们来看看这个例子，尽管其参数在主调过程声明的是单精度浮点型，但是函数仍然返回整型数据：

1. 在工程MyFunctions (Chap04. xls) 里添加一个新模块，重命名为Sample3
2. 激活模块Sample3并且输入子程序HowMuch，如下所示：

```
Sub HowMuch()
```

```

Dim num1 As Single
Dim num2 As Single
Dim result As Single
num1 = 45.33
num2 = 19.24
result = MultiplyIt(num1, num2)
MsgBox result

```

```
End Sub
```

3. 在子程序HowMuch下面输入下列函数过程：

```

Function MultiplyIt(num1,num2) As Integer
    MultiplyIt = num1 * num2
End Function

```

因为储存于变量num1和num2的数值不是整数，要确保它们相乘后的结果为整数，你就得将函数结果设置为整型。如果你不给函数MultiplyIt的结果设置数据类型，过程HowMuch将会将结果按变量result声明的数据类型显示，相乘的结果将是872.1492，而不是872。

你可以在每次运行该过程时，提供它们不同的数值，来使得该函数更有用，你可以使用InputBox函数来输入数据，而不拘泥于程序代码给定的数值。你自己可以依照前面章节中的子程序EnterText，花几分钟来修改过程HowMuch。

8.按地址和按值传递参数

在一些过程中，当你将参数作为变量传递时，VB可能突然改变该变量的数值。要确保该被调函数不改变传递的参数值，你应该在函数声明行在参数名称之前加上关键字ByVal。我们来看看这个例子：

1. 在工程MyFunctions (Chap04.xls)里添加一新模块，命名为Sample4
2. 激活模块Sample4并输入下列代码：

```

Sub ThreeNumbers()
    Dim num1 As Integer, num2 As Integer, num3 As Integer
    num1 = 10
    num2 = 20
    num3 = 30

    MsgBox MyAverage(num1, num2, num3)
    MsgBox num1
    MsgBox num2
    MsgBox num3
End Sub

```

```

Function MyAverage(ByVal num1, ByVal num2, ByVal num3)
    num1 = num1 + 1
    MyAverage = (num1 + num2 + num3) / 3
End Function

```

使用关键字ByVal在参数名称前，可以防止函数改变参数值。子程序ThreeNumbers给三个变量赋值，再调用函数MyAverage来计算，最后计算该三个变量的平均值。函数的参数就是变量num1，num2和num3。注意，所有变量的前面都有关键字ByVal。同时，注意，在计算均值之前，函数MyAverage改变了变量num1的值，在函数内部，变量num1等于11（10+1），因此，当函数将计算的均值传递给过程ThreeNumbers时，函数MsgBox显示的结果是20.3333333333333而不是期望的20，接下来的三个函数显示每个变量的内容，变量储存的内容和它们开始被赋的值一致——10，20，30。

如果你在函数声明行忽略了参数num1前面的关键字ByVal，结果会怎样呢？函数的结果仍然相同，但是函数MsgBox显示的变量num1的内容现在是11了。函数MyAverage不但返回了出乎意料的结果（20.3333333333333而非20），而且改变了储存在变量num1里的原始数值。要避免VB永久地改变提供给函数的数值，就得使用关键字ByVal。

因为每个要传递给函数过程（或子程序）的变量，都可能在接收时改变数值，所以知道如何来保护变量的原始数值是非常重要的。VB有两个关键字，提供或者否认改变变量内容的允许——ByRef和ByVal。VB默认地按地址（关键字ByRef）给函数过程（或子程序）传递信息，引用函数被调用时，函数参数明确的数据。因此，如果函数改变了参数值，原始的数值就被改变了。如果你在函数MyAverage声明参数num1的前面忽略了关键字ByVal时，你就会得到这种结果。如果你想要函数过程改变原始数值，你不必专门在参数前加关键字ByRef，因为，变量数值的传递默认就是ByRef。当你在参数名称前使用关键字ByVal时，VB按值传递参数，这意味着VB复制一份原始数据，然后将复制值传递给函数，如果函数改变了参数的数值的话，原始数据依然不会变——只有复制值变化。这就是为什么函数MyAverage改变了变量num1的数值，而它的原始值还保持不变。

9.使用可选的参数

有时候，你也许要给函数提供额外的参数，例如，你有一个计算每个人膳食的函数。然而，有时你不希望函数进行相同的计算。在参数名称前面加上关键字Optional可以指明该参数不是必须的。可选参数在必须的参数之后，列在参数清单的最后；可选参数总是Variant数据类型，这意味着你不能使用关键字As来明确可选参数的类型。在前面部分，你创建了一个计算三个数值的平均值的函数，假设，你有时只想要计算两个数的均值，你就可以将第三个参数设置为可选的。为了不破坏原来的函数MyAverage，我们来创建一个新的函数Avg，来计算两个或三个数字的均值：

1. 在工程MyFunctions (Chap04. xls)里添加一新模块，命名为Sample5
2. 激活模块Sample5并且输入下列函数：

```
Function Avg(num1, num2, Optional num3)
    Dim totalNums As Integer
    totalNums = 3
    If IsMissing(num3) Then
        num3 = 0
        totalNums = totalNums - 1
    End If
    Avg = (num1+num2+num3)/totalNums
End Function
```

3. 现在来从立即窗口调用该函数：

```
?Avg(2,3)
```

一旦你按下回车，VB就显示结果：2.5

```
?Avg(2,3,5)
```

这次结果为：3.33333333333333

你已经看到，函数Avg允许你计算两个或三个数字的平均值，你可以决定计算几个值的平均值。当你在立即窗口开始输入函数的参数时，VB将可选的参数显示在方括号里（译者：方括号里的参数表示是可选的）

我们花几分钟来分析一下函数Avg。该函数最多可以用三个参数；参数num1和num2是必须的，而第三个参数num3是可选的。注意，可选参数以关键字Optional开头，可选参数列在参数清单的后面。因为参数num1，num2和num3的类型都没有声明，VB对待它们为Variant。

函数内部，变量totalNums声明为整型，并且初始赋值为3。因为该函数必须能够处理两个或三个数字的平均值计算，有关很方便的内置函数IsMissing在检查参数的个数。如果第三个参数没有提供，函数IsMissing的值为0，同时变量totalNums的值被减去1，因此如果没有可选参数，totalNums为2。接下来的代码根据提供的数据计算平均值，并且将结果传递到函数名称。函数IsMissing用来检测可选参数是否提供，如果第三个参数没有提供，那么该函数返回逻辑值True，如果提供了第三个参数时，则返回False。在这里，函数IsMissing是和做一个判断的语句If...Then一起使用的（参见第五章有关该语句和VBA中其它判断语句信息）。如果没有参数num3（IsMissing），那么（Then）VB将num3设置为0，并且将变量totalNums减去1（totalNums = totalNums - 1）。你还能使用别的方法来运行函数Avg吗？使用你自己的方法在工作表里运行该函数，确保你使用两个和三个参数来运行该函数。

技巧4—10 测试函数过程

可以编写一个子程序调用自定义函数，并且显示它的结果，看它是否能得到设计结果。除此之外，该子过程还应该能显示参数的原始数据，这样，你才能很快知道参数数值是否改变了。如果该函数使用了可选参数，你也需要检查不使用可选参数时候的情况。

10.定位内置函数

VBA自带了很多内置函数，可以在在线帮助里很容易地找到这些函数。在VB编辑器窗口选择“帮助”—“Microsoft Visual Basic 帮助”可以访问所有VBA函数按字母顺序排序的清单，然后点击“函数”，

例如，以MsgBox或InputBox函数为例。一个程序好的功能之一就是要与用户互动，当你使用Excel时，你通过多种对话框与该应用程序交流，当你犯错的时候，对话框会弹出来，并且告诉你有错误。当你编写你自己的程序时，你也可以将出乎意料的错误或某个计算的结果通知用户，你可以使用函数MsgBox帮助你做这些。到目前为止，你只是看到了这个函数的一些简单应用，在下面的部分，你将了解如何控制你信息的外观；你也将学习如何使用函数InputBox从用户获得信息。在我们详细讨论这些函数之前，我们来看一个VBA函数，在你已经很熟悉变量和它们的类型的时候，它对你特别有用处。

VB有个VarType函数，它返回一个值变量类型的整数。图4—4例显示了函数VarType的语法和它返回的值。



图4—4 使用内置函数VarType，你可以判别变量的类型（译者：本截图与2002版本有区别）

现在，看看如何在立即窗口里使用这个函数：

1. 打开立即窗口
2. 输入下列给变量赋值的语句

```
age = 18
```

```
birthdate = #1/1/1981#
```

```
firstName = "John"
```

3. 现在询问VB每个变量的数据类型是什么：

```
?varType(age)
```

你按下回车时，VB返回2，如图4—4所示，数字2代表整数类型。

```
?varType(birthdate)
```

VB返回7代表日期。如果你在变量名称上犯了个小错误（比如说，你输入了birthday而不是birthdate），VB将返回0。

```
?varType(firstName)
```

VB告诉你变量firstName的数据是字符串（8）。

11.使用 MsgBox 函数

你目前使用的MsgBox函数局限于给用户用一个简单的，一个按钮的对话框显示信息。你点击确定按钮或者回车来关闭该信息框。要创建一个简单的信息框，只要在MsgBox函数名称后面带上一个用引号包括起来的文本就可以了。换句话说，要显示信息“过程已完成”，你应该准备下列语句：

```
MsgBox "过程已完成" '（注意，英文状态的引号）
```

你可以将它输入立即窗口，快速地测试上面的指令，当你输入完这条指令并且回车后，VB就显示如图4—5的信息框。



图4—5 将文本作为MsgBox函数的参数，来给用户显示信息

MsgBox函数允许你使用其它参数，使你可能决定可用的按钮数目，或者将默认的信息框的标题（Microsoft Excel）改为你自己的标题。也可以设置你自己的帮助主题。MsgBox的语法如下：

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

注意，MsgBox函数有五个参数，只有第一个，Prompt（提示），是必须的；这些列在方括号里面的参数都是可选的。当你在提示参数输入一个非常长的文本时，VB决定如何断句，使文本适合信息框大小。我们在立即窗口里来做些练习，看不同的文本格式技巧：

1. 在立即窗口输入以下指令，确保在一行里输入整个文本，回车

```
MsgBox "All done. Now open ""Chap04.xls"" and place an empty disk in the diskette drive. The following procedure will copy this file to the disk."
```

一旦回车，VB显示信息框，如图4—6



图4—6 如果你设置一下文本格式，长信息看上去将会更吸引人

如果你遇到编译错误，可以点击确定，然后确定文件名用双引号括起来——““Chap04.xls””。当你的信息文本特别长时，你可以使用VBA函数Chr将它分割为好几行。Chr函数需要你跟参数，这个参数是0到255之间的数字，它返回这个数字代表的字符。例如Chr(13)返回的是回车（这和按下回车键相同），以及Chr(10)返回换行字符（这在文本行之间添加空行很有用）。

2. 将上面的指令修改为下述方式：

```
MsgBox "All done." & Chr(13) & "Now open ""Chap04.xls"" and place" & Chr(13) & "an empty disk in the diskette drive." & Chr(13) & "The following procedure will copy this file to the disk."
```



图4—7 通过使用Chr(13)可以将长文本分割成几行

你必须将每段文本片断用引号括起来，内嵌在括号里面的文本（显示状态）需要再用一对括号来括起来，例如“Chap04.xls”。Chr(13)函数指明你希望开始新的一行的地方。字符串的连接字符(&)用来返回连接字符串的字符。在一行输入及其长的文本的时候，很容易失误。回想一下，VB有一个专门的线连续字符（下划线_）帮你将长VBA语句分割为几行，不幸的是，这个线连续符不能在立即窗口使用。

3. 在工程MyFunctions (Chap04.xls)里添加一个新模块并命名为Sample6

4. 激活模块Sample6并且输入如下所示的子程序MyMessage，确保在每个线连续符前面加个空格：

```
Sub MyMessage()  
    MsgBox "All done." & Chr(13) _  
    & "Now open ""Chap04.xls"" and place" & Chr(13) _  
    & "an empty disk in the diskette drive." & Chr(13) _  
    & "The following procedure will copy this file to the disk."  
End Sub
```

你运行过程MyMessage时，VB显示如图4—7一样的信息。正如你看到的，在几行输入的文本更具可读性，而且代码更容易维护。你可以在文本行之间添加一下空白行，来增加信息的可读性。使用Chr(13)或two Chr(10)函数就可以做到，如下列步骤所述。

5. 输入下面的MyMessage2过程：

```
Sub MyMessage2()  
    MsgBox "All done." & Chr(10) & Chr(10) _  
    & "Now open ""Chap04.xls"" and place" & Chr(13) _  
    & "an empty disk in the diskette drive." & Chr(13) & Chr(13) _  
    & "The following procedure will copy this file to the disk."  
End Sub
```

图4—8显示了MyMessage2过程产生的信息框。

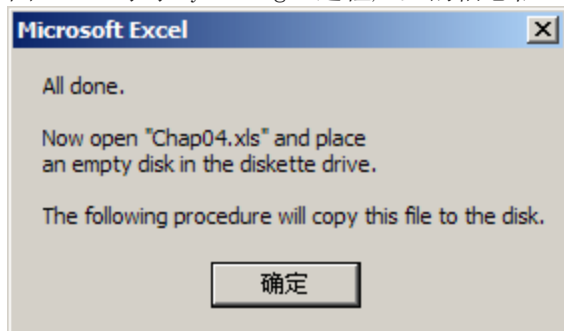


图4—8 你可以通过在文本行之间添加空行增加信息的可读性

既然你已经掌握了文本的格式技术，那么我们就来仔细地看看MsgBox函数的下一个参数吧。尽管按钮参数时可选的，但是它的使用还是很频繁的。这个按钮参数明确多少个按钮，并且是什么样的按钮你想要出现了信息框上，这个参数可以是个常量（参见表4—1），也可以是个数字。如果你忽略这个参数，结果辛苦只会会有一个“确定”按钮，正如你在前面的例子里看到的那样。

表4—1 MsgBox按钮参数的设置

常量	值	描述
按钮设置		
vbOKOnly	0	仅显示确定按钮，这是默认值
vbOKCancel	1	显示确定和取消按钮
vbAbortRetryIgnore	2	显示终止，重试和忽略按钮
vbYesNoCancel	3	显示是，否和取消按钮
vbYesNo	4	显示是和否按钮
vbRetryCancel	5	显示重试和取消按钮
图标设置		
vbCritical	16	显示重要信息图标

vbQuestion	32	显示问号图标
vbExclamation	48	显示警告信息图标
vbInformation	64	显示信息图标
默认按钮设置		
vbDefaultButton1	0	第一个按钮是缺省值
vbDefaultButton2	256	第二个按钮是缺省值
vbDefaultButton3	512	第三个按钮是缺省值
vbDefaultButton4	768	第四个按钮是缺省值
信息框形式		
vbApplicationModal	0	应用程序强制返回；应用程序一直被挂起，直到用户对消息框作出响应才继续工作。
vbSystemModal	4096	系统强制返回；全部应用程序都被挂起，直到用户对消息框作出响应才继续工作。
MsgBox显示的其它设置		
vbMsgBoxHelpButton	16384	将Help按钮添加到消息框
vbMsgBoxSetForeground	65536	指定消息框窗口作为前景窗口
vbMsgBoxRight	524288	文本为右对齐
vbMsgBoxRtlReading	1048576	指定文本应为在希伯来和阿拉伯语系统中的从右到左显示

你什么时候应该使用按钮参数呢？假设你要用户对一个问题回到“是”或“否”，你的信息框就需要两个按钮，当信息框有一个以上的按钮时，就需要将其中一个设置为缺省值，当用户回车的时候，这个默认的按钮就会自动地被选上。

因为，你可以显示各种各样的信息（重要，警告，信息），所以，你需要通过按钮参数设置图形代表（图标）来指明信息的重要性。

除了信息类型之外，按钮参数还可以设置是否用户必须先关闭该信息框才能切换到另外的应用程序。很多情况下，用户需要在对信息框的问题做出反应之前，切换到另外的程序或者进行另外的操作。如果这个信息框是应用程序模式（vbApplicationModal）的话，用户必须先关闭该信息框后才能继续使用你的应用程序。另一方面，如果你想要在用户对信息框响应之前，将所有应用程序挂起，那么你必须在按钮参数里加上系统强制返回设置（vbSystemModal）。按钮参数的设置分为五组：按钮设置，图标设置，默认按钮设置，信息框形式和其它的MsgBox显示设置（参见图4-1）。每组设置里面只能选一个加入按钮参数里面。你可以将每种需要的设置加和起来，来设置按钮参数，例如，要显示一个带两个按钮（“是”和“否”），问号图标以及将“否”按钮设置为缺省值的信息框，你可以在表4-1里查找相应的值并且加和起来，你应该得到292（4+32+256）。你可以在立即窗口里面输入下列代码，快速查看使用该计算的按钮参数的信息框：

```
MsgBox "Do you want to proceed?", 292
```

下面显示的就是信息框结果。当你直接使用加和起来的值作为参数时，你的程序可读性就不高了，因为没有参考索引表格供你检查292背后的意思。要改善你信息框函数的可读性，最好使用常量，而不要使用它们的值，例如，在立即窗口输入下列修改后的语句：

```
MsgBox "Do you want to proceed?", vbYesNo + vbQuestion + vbDefaultButton2
```

上面的语句得到如图4-9所示的相同结果。

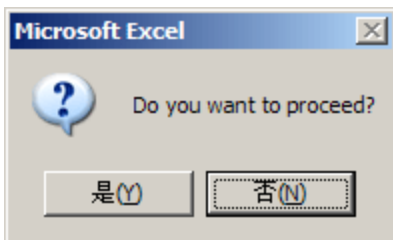


图4-9 你可以使用可选的按钮参数来确定信息框上的按钮个数

下面的例子示范如何在VB过程里使用按钮参数：

1. 在工程MyFunctions (Chap04.xls)里添加一新模块，并命名为Sample7
2. 激活Sample7模块，并且输入如下子程序MsgYesNo：

```
Sub MsgYesNo()
```

```
Dim question As String
Dim myButtons As Integer

question = "是否要打开一个新工作簿?"
myButtons = vbYesNo + vbQuestion + vbDefaultButton2
MsgBox question, myButtons
```

End Sub

在上面的子程序里，变量question储存了你的信息文本，而按钮参数的设置则储存于变量myButtons。除了使用常量名称之外，你还可以使用它们的值，例如下面的：

```
myButtons = 4 + 32 + 256
```

但是，明确了按钮常数的常量名称的话，你可以使你的程序对你自己以及将来可能要使用该程序的人来说更容易理解。变量question和myButtons用作MsgBox函数的参数。运行该程序后，你将看到如图4-9所示的结果。注意，现在按钮“否”是被选中的，它是该对话框的默认按钮，如果你按下回车，Excel将该信息框从屏幕上移除，因为MsgBox函数后面没有任何指令，所以，不会发生其它操作。将默认按钮换成vbDefaultButton1设置，可以更改默认按钮。

MsgBox函数的第三个参数是标题，虽然这也是个可选参数，但是它很方便，因为当你忽略它们（默认为Microsoft Excel）时，就不能给程序提供可视提示。你可以使用这个参数，将标题栏设置为你想要的任何文字。假设你要过程MsgYesNo显示标题为“新工作簿”，下面的过程MsgYesNo2示范如何使用标题参数：

```
Sub MsgYesNo2()
    Dim question As String
    Dim myButtons As Integer
    Dim myTitle As String

    question = "Do you want to open a new workbook?"
    myButtons = vbYesNo + vbQuestion + vbDefaultButton2
    myTitle = "New workbook"
    MsgBox question, myButtons, myTitle
End Sub
```

标题参数文本储存于变量myTitle。如果你没有明确标题参数的内容，VB将默认显示为“Microsoft Excel”。注意，参数是基于MsgBox函数决定的顺序列出的，如果你要按你自己的顺序列出这些参数的话，你就必须将参数名称一起写出，例如：

```
MsgBox title:=myTitle, prompt:=question, buttons:=myButtons
```

后面两个参数——帮助文件（helpfile）和上下文（context）——经常为那些对Windows环境下的帮助文件很熟悉的程序员使用。参数helpfile指明某个包含你要显示给用户的附加信息的具体帮助文件的名称，当你明确了这个参数后，Help按钮就会在信息框上显示出来。当你使用helpfile参数时，你同时也使用context参数。这个参数表明在帮助文件里你要显示的那个帮助主题。假设Help.hlp是你创建的帮助文件，55是你要使用的帮助主题，你可以按照如下指令来显示这些信息于信息框上：

```
MsgBox title:=mytitle, _
    prompt:=question _
    buttons:=mybuttons _
    helpFile:= "HelpX.hlp", _
    context:=55
```

上面只是一条VBA语句，使用连接符打断为好几行。

12.MsgBox 函数的运行值

当你显示只有一个按钮的信息框时，可以点击确定按钮或者回车键将信息框从屏幕上移除，然而，当信息框有两个或以上的按钮时，你的程序需要知道按的是哪个按钮。你可以将信息框结果储存在一个变量上来实现。表4-2 显示了MsgBox函数返回值。

表4-2 MsgBox函数返回值

选择的按钮	常数	值
-------	----	---

OK (确定)	VbOK	1
Cancel (取消)	vbCancel	2
Abort (终止)	vbAbort	3
Retry (重试)	vbRetry	4
Ignore (忽略)	vbIgnore	5
Yes (是)	vbYes	6
No (否)	vbNo	7

MsgYesNo3过程是MsgYesNo2过程修改后的版本，示范如何确定用户按下的是哪个按钮：

```
Sub MsgYesNo3()
    Dim question As String
    Dim myButtons As Integer
    Dim myTitle As String
    Dim myChoice As Integer

    question = "Do you want to open a new workbook?"
    myButtons = vbYesNo + vbQuestion + vbDefaultButton2
    myTitle = "New workbook"

    myChoice = MsgBox(question, myButtons, myTitle)
    MsgBox myChoice
End Sub
```

在上面的过程里，你将MsgBox函数的结果赋给变量myChoice。注意，现在，MsgBox函数的参数列在括号里面：

```
myChoice = MsgBox(question, myButtons, myTitle)
```

当你运行MsgYesNo3时，出现带有两个按钮的信息框，当你点击“是”时，MsgBox myChoice将显示数字6；当点击“否”则得到数字7。你将在第五章里面学习如果让程序根据按钮的选择进行不同的任务。

技巧4-11 MsgBox函数——使用还是不使用括号？

当你需要使用MsgBox函数返回的结果时，需要使用括号将该函数的参数包括起来。不使用括号，意味着你告诉VB你将忽略该函数的结果。当MsgBox函数包含两个或以上的按钮时，你很可能想要使用该函数的结果。

13.使用 InputBox 函数

InputBox函数显示一个信息提示用户输入数据，这个对话框有两个按钮——“确定”和“取消”，当你点击确定时，InputBox函数返回用户输入在信息框里的信息；当你点击取消时，函数则返回空字符串（" "）。InputBox函数的语法显示如下：

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

第一个参数，prompt，是你想要显示在对话框上的信息，你可以使用函数Chr(13)或Chr(10)将长文本打断为几行（参见本章中使用MsgBox函数的例子）。剩下所有的参数都是可选的。

第二个参数，title，让你改变对话框的默认标题，默认的标题是Microsoft Excel。

InputBox函数的第三个参数，default，让你在文本框里显示一个默认值，如果你忽略这个参数的话，显示的将是空白编辑框。接下来的两个参数，xpos和ypos，允许你设置该对话框在屏幕上出现的位置，如果你忽略这两个参数，对话框就会出现于当前窗口的中央，xpos参数决定对话框在屏幕上从左起的水平位置，忽略它时，对话框显示在水平中央，而ypos参数决定对话框在屏幕从上而下的竖直位置，忽略它，对话框就在竖直大约三分之一的位置。xpos和ypos都使用一个叫twips的专门单位衡量，1twip大约等于0.0007英寸。

最后两个参数，helpfile和context，和在本章前期讨论的MsgBox函数相应的参数使用方法一样。

现在你知道了InputBox参数的意义了，我们来看看这个函数的使用示例：

1. 在MyFunctions (Chap04.xls)工程里添加一个新模块，重命名为Sample8
2. 激活Sample8模块，并且输入下列子程序：


```
Sub Informant()
    InputBox prompt:="Enter your place of birth:" & Chr(13) _
        & "(e.g., Boston, Great Falls, etc.)"
End Sub
```

上面的过程显示一个带两个按钮的对话框，输入提示显示在两行里。象MsgBox函数一样，如果你想要使用用户输入的数据，那么你应该使用一个变量来储存该对话框结果。下面显示的子程序Informant2将InputBox函数的结果赋值给变量town：

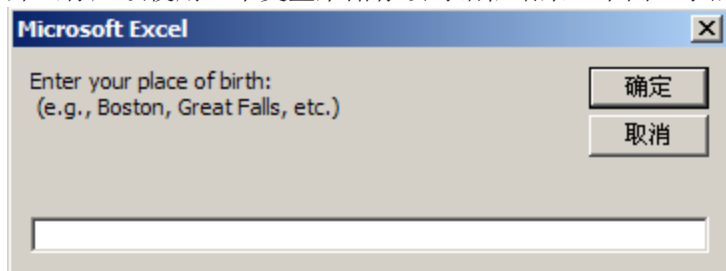


图4—10 Informant子程序产生的对话框

```
Sub Informant2()
    Dim myPrompt As String
    Dim town As String
    Const myTitle = "Enter data"

    myPrompt = "Enter your place of birth:" & Chr(13) _
        & "(e.g., Boston, Great Falls, etc.)"
    town = InputBox(myPrompt, myTitle)
    MsgBox "You were born in " & town & ". ", , "Your response"
End Sub
```

注意，这次，InputBox函数的参数列在了括号中间。如果你需要在稍后的程序中需要使用InputBox函数的结果，那么括号是必须的。Informant2子程序使用常数来确定显示在对话框标题上的文本。因为，这个值在过程执行过程中从始至终都是保持不变的，所有，可以将对话框的标题声明为一个常量，然而，如果你愿意，你也可以使用一变量。

当你运行使用了InputBox的过程时，使用该函数的对话框总是出现在屏幕的同一位置，你可以按前面解释的那样，提供xpos和ypos参数来改变对话框的位置。

3. 修改过程Informant2中的InputBox函数，让对话框显示在屏幕的左上角，例如：

```
town = InputBox(myPrompt, myTitle, , 1, 200)
```

注意，参数myTitle后面紧跟两个逗号，第二个逗号是忽略掉的默认值参数。下面两个参数决定对话框的水平和竖直位置。如果你忽略了参数myTitle后面的第二个逗号，VB将会使用数字1作为默认值。如果你使用了参数名称的话，（例如，prompt:=myPrompt, title:=myTitle, xpos:=1, ypos:=200），你就不必记住在每个忽略了参数的地方加逗号了。

如果你输入了一个数字，而不是一个城镇的名称，后果会怎样？因为，用户经常会在对话框里输入错误的数据，所以，你的程序必须验证用户输入的数据是否可以在将来的数据操作里使用。InputBox函数本身并没有提供验证数据的工具，要验证用户的输入，你必须使用其它的VBA指令，这将在下章里讲述。

14.数据类型转变

InputBox函数的结果总是字符串，如果用户输入的是个数字，用户输入的字符串值必须转换为数字值之后，才能用于你程序里的数学计算。VB转换数据类型轻而易举，不过，在早期的Excel版本里，这是不可能的。

1. 激活模块Sample8，并输入以下过程AddTwoNums：

```
Sub AddTwoNums()
    Dim myPrompt As String
    Dim valuel As String
    Const myTitle = "Enter data"
    Dim mySum As Single
```

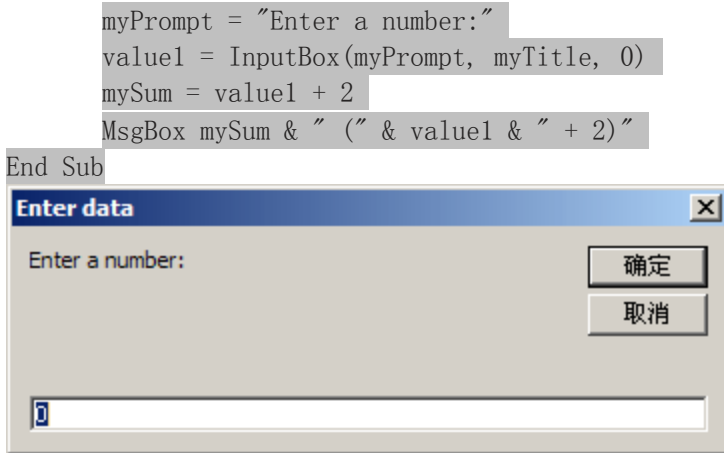


图4-11 要给用户提示数据的确切类型，你可以在北京框里提供一个默认值

上面的程序显示如图4-11所示的对话框。注意，这个对话框使用了两个专门的功能，InputBox函数的可选参数标题和默认值。该对话框显示了有常量myTitle确定的文本字符串作为标题，而不是默认的“Microsoft Excel”。在编辑框里面的0提示用户输入数字，而不能输入文本。一旦用户输入了数据并点击确定时，用户的输入就被赋值给了变量value1

```
value1 = InputBox(myPrompt, myTitle, 0)
```

变量value1的数据类型是字符串，你可以在上面指令的下面加上如下语句，快速地查看它的数据类型：

```
MsgBox varType(value1)
```

当VB运行上面的代码时，将显示信息—数字8，你可以在本章的图4-4里查看该数字代表字符串类型。

技巧4-12 定义常量

你可以将标题文本赋值到一个常量上，以确保某个VBA程序里所有的标题栏都显示相同的 文本。依照这个技巧，你可以在多次输入某标题文本时节省时间。

技巧4-13 避免类型不匹配错误

如果你试图在Excel的早期版本里（2000以前版本）运行AddTwoNums过程，当VB试图执行下列代码行时，你将得到类型不匹配的错误：

```
mysum = value1 + 2
```

使用内置函数CSng将储存于value1的字符串转换为一个单精度浮点类型的数字，可以避免类型不匹配错误，代码写成：

```
mysum = CSng(value1) + 2
```

下一行，mySum = value1 + 2，在用户输入的数据上加上2，并且将计算结果赋值给变量mySum。因为变量value1的数据类型是字符串，在使用它计算之前，VB在后台进行数据类型的转换，VB知道转换的需要。没有它，两种不兼容的数据类型（文本和数字）将会产生类型不匹配错误。程序最后是一个MsgBox函数，显示计算的结果已经给用户显示总数是如何组成的。

15.使用 InputBox 方法

除了InputBox函数之外，还有InputBox方法，如果你激活对象浏览器，然后搜索“inputbox”，VB将显示两个InputBox——一种为Excel库，另一种为VBA库（见图4-12）。InputBox方法在Excel库里面可用，它的语法和本章前面讲的InputBox函数的语法有轻微差别，它的语法为：

```
expression.InputBox(Prompt, [Title], [Default], [Left], [Top], [HelpFile], [HelpContextID], [Type])
```

所有方括号里面的参数都是可选的。Prompt（提示）参数是显示于对话框上的信息；Title是对话框的标题；而Default是对话框上文本框里的初始值。Left和Top参数是明确对话框在屏幕上的位置。这些参数的输入值的单位是Point（1/72英寸）。当用户点击帮助按钮时，参数HelpFile和HelpContextID明确帮助文件名称以及某个明确的帮助主题。InputBox方法的最后一个参数——Type（类型）明确返回的数据类型。如果你忽略这个参数，InputBox方法将会返回文本格式。类型参数的值列在表4-3里。

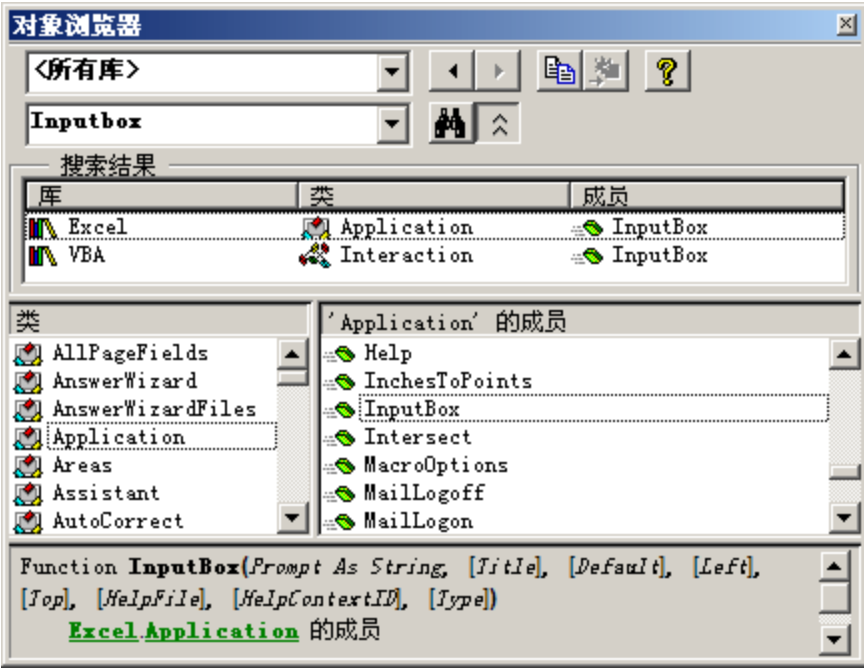


图4-12 别忘记使用对象浏览器来搜索VB函数和方法

表4-3 InputBox方法返回的数据类型

值	返回的数据类型
0	公式
1	数字
2	字符串（文本）
4	逻辑值（True或False）
8	单元格引用，作为一个Range对象
16	错误值，例如#N/A
64	数组

如果你使用3作为Type参数的话，用户将既可以输入一个数字也可以输入一个文本。这个值是将1（数字）和2（字符串）相加而得到的。InputBox方法很适合那些需要用户选择工作表单元格范围的VBA程序。

1. 关闭对象浏览器，如果你已经打开了话
2. 在模块Sample8里面，输入下列过程WhatRange:

```
Sub WhatRange()  
    Dim newRange As Range  
    Dim tellMe As String  
    tellMe = "Use the mouse to select a range:"  
    Set newRange = Application.InputBox(prompt:=tellMe, _  
        Title:="Range to format", _  
        Type:=8)  
    newRange.NumberFormat = "0.00"  
    newRange.Select  
End Sub
```

过程WhatRange开始于一对象变量的声明——newRange。试回想一下第三章，对象变量指向数据的地址。用户选择的单元格被赋值给对象变量newRange。注意变量名称前面的关键字Set:

```
Set newRange = Application.InputBox(prompt:=tellMe, _  
    Title:="Range to format", _  
    Type:=8)
```

类型参数（Type:=8）使用户能够选择任何单元格区域。当用户选中单元格区域时，下句指令：

```
newRange.NumberFormat = "0.00"
```

改变所选单元格的格式。最后一句选择用户加亮的区域。

3. 运行过程WhatRange。VB显示一个对话框，提示用户在工作表里选择一个单元格区域。

4. 使用鼠标选择你要的单元格，当鼠标在单元格上拖动时，VB就会将选择的区域引用到对话框的编辑框里面。

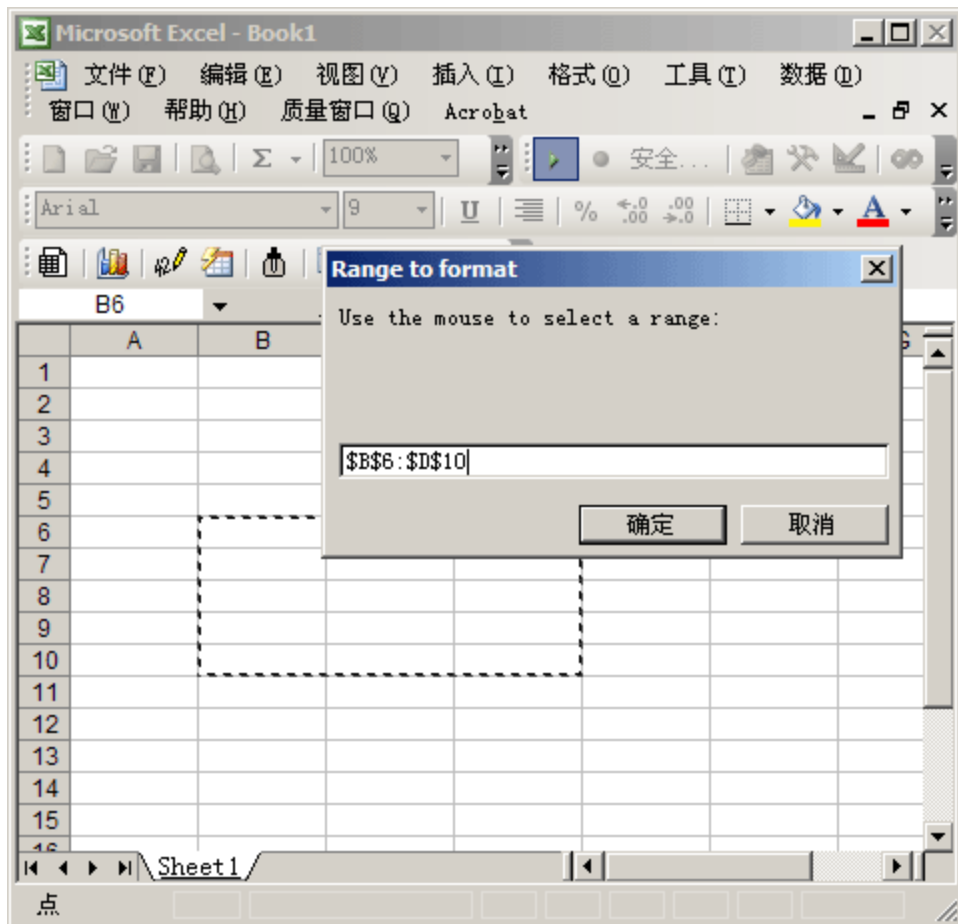


图4-13 使用Excel的InputBox方法，你可以从用户处获得区域地址

- 你选择了单元格后，点击对话框上的确定按钮，被选择的区域就已经设置好格式了。要检查是否按你的意思设置了，你可以在该区域的任意单元格里输入一个整数，这个数字应该显示为两位小数。
- 重新运行该过程，并且当出现对话框时，点击取消按钮。如果你在选择了一个单元格或者一个区域后点击确定按钮，过程WhatRange将工作正常。不幸地是，当你点击取消按钮或Esc按钮，VB将显示一错误信息——“要求对象”。当你点击错误对话框上的调试按钮，VB就会加亮导致错误的代码行。因为你不希望在取消对话框时选择任何单元格，所以你必须想个法子忽略VB显示的这个错误。使用一个专门的语句，On Error GoTo 标志，你就可以绕过错误的发生。该指令的语法如下：

On Error GoTo 标志

这个指令应该放在变量声明行的下面。标志可以是除了VB关键字之外的任何你想要的词语。如果错误发生时，VB就会直接跳到该特别的标志，如下面步骤8所示。

- 选择“运行”-“重新设置”以取消正在运行的程序。
- 将过程WhatRange修改为如下所示WhatRange2：

```
Sub WhatRange2()  
    Dim newRange As Range  
    Dim tellMe As String  
    On Error GoTo VeryEnd  
    tellMe = "Use the mouse to select a range:"
```

```

Set newRange = Application.InputBox(prompt:=tellMe, _
    Title:="Range to format", _
    Type:=8)
newRange.NumberFormat = "0.00"
newRange.Select
VeryEnd:
End Sub

```

9. 运行程序WhatRange2，一旦出现对话框时就点击取消按钮。注意，这次程序没有产生错误。当VB遭遇错误时，就会跳到位于程序结尾处的标志VeryEnd。位于错误和标志VeryEnd之间的语句被忽略了（原文不当：The statements placed between On Error Goto VeryEnd and the VeryEnd label are ignored）。你将在第十三章里面找到更多的诱捕VBA程序里错误的例子。

技巧4-14 子程序和函数：你应该使用哪个？

创建子程序的时候：

- 需要执行一些动作
- 需要获取用户信息
- 需要在屏幕上显示信息

创建函数的时候：

- 需要不只一次的做一些简单的计算
- 需要做复杂的计算
- 需要不只一次地调用相同的指令块
- 需要检查某些表达正确与否

16.使用主过程和子过程

当你大VBA程序得越来越大，要很好地维护这么多的代码行是很困难的。要让你的程序容易编写、理解和改变，你就应该使用井井有条的结构化程序。如何创建结构化程序？你只要简单地将大问题分成一些可以同时执行的小问题就行。在VBA中，你可以通过创建一个主过程和一个或多个子过程来实现它。因为主过程和子过程都是子程序，所以你都可以用关键字Sub将它们声明。主过程可以调用所需的子过程，并且将参数传递给它们。它也可以调用函数。下面的例子显示过程AboutUser。该过程要求用户姓和名，并且将姓和名从全名中分离出来。最后的语句显示用户的姓，随后是逗号和名。你再读下去，该过程将被分割成几个任务，以示范使用主过程，子过程和函数的概念。

```

Sub AboutUser()
    Dim fullName As String
    Dim firstName As String
    Dim lastName As String
    Dim space As Integer

    'get input from user 从用户获取信息
    fullName = InputBox("Enter first and last name:")

    'get first and last name strings 获得姓和名字符串
    space = InStr(fullName, " ")
    firstName = Left(fullName, space - 1)
    lastName = Right(fullName, Len(fullName) - space)

    'display last name, first name 显示姓和名
    MsgBox lastName & ", " & firstName
End Sub

```

过程AboutUser可以分割为一些细小的任务。第一个任务便是获取用户的全名；下一个任务则需要你将用户提供的信息分割为两个字符串：姓和名，这些任务可以交给不同的函数（例如：GetLast和GetFirst）；最后的任务是显示重新排列的姓名字符串信息。既然你已经知道了你应该注重于哪些任务，我们现在就来看看如何完成每个任务。

1. 在你当前的VBA工程里面添加一个模块，并重命名为Sample9
2. 在Sample9模块窗口里面输入下列过程AboutUserMaster:

```
Sub AboutUserMaster()  
    Dim first As String, last As String, full As String  
    Call GetUserName(full)  
    first = GetFirst(full)  
    last = GetLast(full)  
    Call DisplayLastFirst(first, last)  
End Sub
```

上面显示的主过程通过调用适当的子程序和函数来控制程序的主流程。该主过程以变量生命开始，第一条语句Call GetUserName (full)调用子过程GetUserName（见第三步）并且传递给一参数——变量full的内容。

因为变量在执行调用语句之前没有赋与任何值，所以它的值是一个空字符串（“ ”）。注意，子过程的名称在Call之后。尽管你在调用过程时并没有要求使用关键字Call，但是，你在调用一个需要参数的过程时必须使用它。参数列表必须包括在括号里面。

3. 输入下面的GetUserName子程序:

```
Sub GetUserName(fullName As String)  
    fullName = InputBox("Enter first and last name:")  
End Sub
```

过程GetUserName示范了两个非常重要的VB编程概念：如何传递参数给一子程序以及如何将值从子程序传递回给主调过程。在主过程（见第二步）中，你调用了过程GetUserName，并且将其作为一参数传递：变量full。该变量被参数fullName接收，该参数子过程GetUserName的Sub语句里声明了。因为在VB调用子过程GetUserName的时候，变量full包含一空字符串，参数fullName同样也接收了这个空字符串。当VB显示对话框并且获得用户的姓名时，这个姓名将赋给参数fullName。赋给参数的值被传递回给子过程执行后的匹配参数。因此，当VB返回主过程时，变量full就回包含用户的姓名。传递给子过程的自变量将被其参数接收。注意，参数名称（fullName）后面紧跟着数据类型的声明（As String）。虽然，参数的数据类型必须和相匹配的自变量的数据类型一致，但是，不同的名称还是可以使用给一个自变量和它相应的参数。

技巧4-15 自变量 (Arguments) 和参数 (Parameters)

- 自变量是传递给一个子过程的变量，常量或表达式
- 参数则只是接收值并传递给子过程的变量

4. 输入下述函数GetFirst:

```
Function GetFirst(fullName As String)  
    Dim space As Integer  
    space = InStr(fullName, " ")  
    GetFirst = Left(fullName, space - 1)  
End Function
```

主过程中的第二条语句（见第二步）first = GetFirst(full)，将变量full的值传递给函数GetFirst。函数的参数fullName接收到该值。要从用户提供的数据里分出姓和名，你就必须找到姓和名中间的空格。因此，该函数的开头是当地变量space的声明，下条语句则使用VBA内置函数InStr返回字符串fullName里空格（“ ”）的位置。然后将获得的数字赋值给变量space。最后，Left函数用来提取字符串fullName从左到某特定个数（space - 1）的字符。名的长度比储存在变量space的值少一个字符。函数的结果（用户的名）赋值给函数名。当VB返回主过程时，它就将结果放置于变量first。

5. 输入下列函数GetLast:

```
Function GetLast(fullName As String)  
    Dim space As Integer  
    space = InStr(fullName, " ")  
    GetLast = Right(fullName, Len(fullName) - space)  
End Function
```

主过程里面的第三条语句（见第二步）last = GetLast(full)，将变量full的值传递给函数GetLast。该函数的目的是提取用户提供的数据中的用户的姓。函数GetLast使用内置函数Len来计算字符串fullName的总字符数。函数Right提取字符串fullName从右边某个特定字符开始（Len(fullName) - space）的字符。然后，获得的字符串赋值给函数名称，一旦返回主

过程，它就储存于变量last。

6. 输入下述子过程DisplayLastFirst:

```
Sub DisplayLastFirst(firstName As String, lastName As String)
    MsgBox lastName & ", " & firstName
End Sub
```

主过程里面的第四条语句（见第二步）Call DisplayLastFirst(first, last)，调用子过程DisplayLastFirst并且将两个自变量：first和last。为了接收这些自变量，子过程DisplayLastFirst和两个相匹配的参数（firstName和lastName）一起被声明了。回想我们前面说过，不同的名称可以用在自变量和相应的参数上。然后子过程DisplayLastFirst显示用户的姓，逗号，和名。

技巧4-16 使用子过程的好处

- 维护多个子过程要比维护一个大过程要容易得多
- 由一个子过程执行的任务可以给好几个过程使用
- 每个子过程再放入主过程里之前就被单独测试
- 多个程序员可以负责各自的子过程，这些子过程再构建一个大的程序

17.接下来……

在本章里，你学习了子过程和函数之间的区别：子过程执行操作；函数返回数值。而且你可以通过录制或者输入来创建子过程，函数则不可以录制，因为它们可能需要参数，你必须手动输入。你看到了从工作表或者其他VB过程调用的函数实例。

你学习了如何给函数传递自变量，决定函数结果的数据类型。你在你的VBA关键字的系统里增加了ByVal，ByRef和Optional等关键字。你也看到如何将问题分割为更小更简单的任务，以使你的程序更容易理解。最后，你学习了子过程如何在参数的帮助下，将数值传递回到主调过程。

过了本章后，你应该能够创建适合你特定需要的你自己的自定义函数了。你应该可以通过使用MsgBox和InputBox函数轻松地 and 用户互动。第五张将介绍程序抉择，你将学习如何基于你提供的情形结果改变你的VBA过程的方向。

第五章 基于 VBA 做决定

作者: Julitta Korol 翻译: Tiger Chen Jan 16' 2005

我们每天要作成千上万个决定，有些决定是自发的，我们自动地作出了这些决定，而不需要停下来去想。其它的决定则需要我们事先两个或者两个以上的选择甚至计划好几个任务。VBA和其它的编程语言一样，提供了专门的语句允许你在自己的程序中包含抉择点。但是，什么是做决定呢？举例说某人问你这个问题：“你喜欢红色吗？”想过这个问题之后，你将回答“是”或者“不”。如果你不确定或者根本就不关心这个问题，你也许就回答“也许”或“可能”。在编程中，你必须决断，只有“是”或“否”的答案是允许的。在编程中，所有的决定都是基于提供的答案作出的。如果答案是肯定的，程序就会执行某段特定的指令；如果答案是否定的，程序则将执行另外一段指令或者干脆就不做任何操作。在本章，你将学习如何使用VBA条件语句来改变你的程序流向。条件语句通常称为“控制结构”，因为它们使你能够控制你的VBA过程的走向，跳过某些语句以及“分叉”到程序的另外一部份去了。

1.关系和逻辑运算符

你在你的VBA过程里面通过使用专门的控制结构里的条件表达式来做决定。条件表达式是使用关系运算符（见表5-1），逻辑运算符（见表5-2）或者两者结合的表达式。当VB在你程序里遇到条件表达式时，它将评估该表达式是对还是错。

表5-1 VBA中的关系运算符

运算符	描述
=	等于
<>	不等于

>	大于
<	小于
>=	大于等于
<=	小于等于

表5-2 VBA中的逻辑运算符

运算符	描述
AND	在采取某行动前，所有的条件都必须为真（TRUE）
OR	在采取某行动前，至少有一个条件为真（TRUE）
NOT	用来否定条件。如果该条件为真，NOT使它为假；如果条件为假，NOT使它变真

2.If...Then 语句

在VBA过程里面作决定的最简单的方法就是使用If...Then语句。假使你想要基于某个条件选择一个行动，那么你可以使用下述结构：

If 条件Then 语句

例如，要删除工作表里面的空行，首先得确认当前单元格为空白的，如果测试的结果为真则继续删除包含本单元格在内的整行：

```
If ActiveCell = "" Then Selection.EntireRow.Delete
```

如果当前单元格不是空白的，VB将忽略关键字Then后面的语句。

如果条件为真，你有时候可能想要执行好几个操作，虽然你可以在同一行加上其它的语句，通过冒号分隔它们，但是如果你使用多行If...Then语句，你的代码将更清晰，如下：

```
If 条件Then
```

```
    语句1
```

```
    语句2
```

```
    语句N
```

```
End If
```

例如，当当前单元格的数值大于50时执行一些操作，你可以编写如下指令：

```
If ActiveCell.Value >50 Then
    MsgBox "The exact value is " & ActiveCell.Value
    Debug.Print ActiveCell.Address & ": " & ActiveCell.Value
End If
```

在上面的例子中，如果当前单元格数值小于等于50的话，那么在关键字Then和End If之间的语句就不会执行。注意，If...Then语句必须以关键字End If结束。VB如何作决定呢？它评估在关键字If和Then中间找到的条件。我们来评估一下下面的条件：

```
ActiveCell.Value >50
```

1. 在一个空白工作表上选择任意一个单元格并输入50
2. 切换到VB编辑器窗口
3. 激活立即窗口
4. 输入下述语句，并且按下回车键

```
? ActiveCell.Value >50
```

回车后，VB写下测试结果——false。当测试结果为假时，VB将不会读代码中关键字Then之后的语句，它将直接跳过去读下行代码，但是，如果没有其它的代码行时，程序就将结束。

5. 现在，将运算符改为小于等于号，并且让VB评估下述条件：

```
? ActiveCell.Value <= 50
```

这次测试返回真（true），并且VB跳到关键字Then后面的语句上。

6. 关闭立即窗口。

既然你已经知道了VB如何评估条件的，我们就来在VBA过程里试试If...Then语句吧。

1. 打开一个新工作簿并保存为Chap05.xls

2. 切换到VB编辑器窗口，并且将VBA工程重命名为Decisions
3. 插入一新模块并重命名为IfThen
4. 在IfThen模块里，输入下述过程：

```
Sub SimpleIfThen()  
    Dim weeks As String  
    weeks = InputBox("How many weeks are in a year:", "Quiz")  
    If weeks <> 52 Then MsgBox "Try Again"  
End Sub
```

过程SimpleIfThen将用户的答案储存于一个名为weeks的变量上，然后，该变量的值将会和数值52进行比较。如果比较的结果为真（也就是说，变量weeks的值不等于52），VB就会显示信息“Try Again”

5. 运行过程SimpleIfThen并且输入一个除52之外的数字
6. 重新运行过程SimpleIfThen并输入数字52。当你输入正确的周数，VB将不会做任何事情，程序就直接结束了。当用户猜对时，最好也显示一信息。
7. 在关键字End Sub前的另外一行输入下述指令：

```
If weeks = 52 Then MsgBox "Congratulations!"
```

8. 再次运行过程SimpleIfThen并输入52。当你输入了正确的答案，VB不会执行语句MsgBox “Try Again.”。如果提供的条件测试结果为假时，过程的执行结果就是忽略关键字Then右边的语句。回想一下，VBA过程可以调用另外一个过程，我们来看看它是否可以调用本身

技巧5-1 If...Then语句的两种格式

If...Then语句有两种格式——单行和多行。这种短格式适合于可以写在一行里的语句，例如：

```
If secretCode <> 01W01 Then MsgBox "Access denied"
```

或者：

```
If secretCode = 01W01 Then alpha=True : beta = False
```

这里的secretCode，alpha和beta是变量名称。在第一个例子里，如果变量secretCode的值不等于01W01，那么VB显示信息“Access denied”。在第二个例子里，当变量secretCode值等于01W01时，VB就将变量alpha设置为真，变量beta为假。注意，执行的第二条语句用冒号与第一条语句分隔开来。

如果当条件为真需要执行很多语句或将要执行的语句及其长时，多行的If...Then语句会更清楚，如下例所示：

```
If ActiveSheet.Name = "Sheet1" Then  
    ActiveSheet.Move after:=Sheets _  
        (Worksheets.Count)  
End If
```

在这个例子中，VB将会检查当前工作表的名称。如果它是“Sheet1”，条件ActiveSheet.Name = “Sheet1”将为真，并且VB将继续执行关键字Then后面的代码行。结果，当前工作表将会被移动到工作簿的最后。

9. 按下面方法修改过程SimpleIfThen里的第一个If语句：

```
If weeks <> 52 Then MsgBox "Try Again" : SimpleIfThen
```

我们在原来的过程SimpleIfThen后面加上一个冒号和SimpleIfThen过程名称。如果用户输入了不正确的答案，他将看到一信息，并且他一旦点击确定按钮后，他将获得另外一次机会来提供正确的答案——输入框将再次出现。用户将能够不断地猜测答案，事实上，他不能适当地推出该过程，直到他提供了正确的答案。如果他点击了取消按钮，他将不得不去处理不友好的错误信息“类型不匹配”。在上章里，你看到了如何使用On Error GoTo标志语句来绕过错误，至少你在第十三章里学习更多的关于错误处理的知识之前，你可以那么做。现在，你可以这样来修改你的过程SimpleIfThen：

```
Sub SimpleIfThen()  
    Dim weeks As String  
    On Error GoTo VeryEnd  
    weeks = InputBox("How many weeks are in a year:", "Quiz")  
    If weeks <> 52 Then MsgBox "Try Again" : SimpleIfThen  
    If weeks = 52 Then MsgBox "Congratulations!"  
    VeryEnd:  
End Sub
```

10. 运行几遍过程SimpleIfThen，提供一些不正确的答案。你在程序里面加的错误捕捉指令使得用户可以推出猜测，而不必面对这可恶的错误信息。

3. 基于多于一个条件的决定

在上面段落里的过程SimpleIfThen里，If...Then语句仅评估一个条件，然而，这个语句可以使用一个以上的条件。你必须使用逻辑运算符AND和OR（参见本章前面的表5-2）来明确If...Then语句里的多个条件。这里是使用AND运算符的语法：

If 条件1 AND 条件2 Then 语句

在上面的语法例，条件1和条件2都必须为真，VB才会执行关键字Then右边的语句。例如：

```
If sales = 10000 AND salary < 45000 Then SlsCom = Sales * 0.07
```

在这个例子中，条件1 sales=10000，条件2 salary<45000。当AND使用在该条件表达式中时，两个条件都必须为真，VB才会计算销售佣金（SlsCom）。如果两个条件中只要有一个为假，或者都为假，VB将会忽略Then后面的语句。如果符合其中一个条件就足够好时，你就应该使用运算符OR。这里是语法：

If 条件1 OR 条件2 Then 语句

运算符OR更灵活一些，只要任何一个条件为真时，VB就会执行关键字Then后面的语句。我们来看看这个例子：

```
If dept = "S" OR dept = "M" Then bonus = 500
```

在上面的例子里，如果有个条件为真，VB就将给变量bonus赋值500。如果两个条件都为假，那么VB就会忽略该行剩余的代码。现在我们来看个完整的例子。假设如果你采购50套产品的话，就可以获得10%的折扣，单价为\$7.00。过程IfThenAnd示范运算符AND的使用。

1. Decisions (Chap05.xls)工程的IfThen模块里输入下述过程：

```
Sub IfThenAnd()  
    Dim price As Single  
    Dim units As Integer  
    Dim rebate As Single  
  
    Const strmsg1 = "To get a rebate you must buy an additional "  
    Const strmsg2 = "Price must equal $7.00"  
    units = Range("B1").Value  
    price = Range("B2").Value  
  
    If price = 7 AND units >= 50 Then  
        rebate = (price * units) * 0.1  
        Range("A4").Value = "The rebate is: $" & rebate  
    End If  
  
    If price = 7 AND units < 50 Then  
        Range("A4").Value = strmsg1 & 50 - units & " unit(s)."  
    End If  
  
    If price <> 7 AND units >= 50 Then  
        Range("A4").Value = strmsg2  
    End If  
    If price <> 7 AND units < 50 Then  
        Range("A4").Value = "You didn't meet the criteria."  
    End If  
End Sub
```

上面的过程IfThenAnd使用了四个If...Then语句来评估两个变量price和units的内容。在If...Then关键字之间的运算符AND使得测试多于一个的条件成为可能。使用了AND运算符时，所有条件都必须为真，VB才会去执行关键字Then和End之间的语句。因为过程的运行依赖于工作表单元格里输入的数据，所以从Excel窗口来运行它比较方便。

2. 切换到Excel应用窗口，并且选择“工具”-“宏”-“运行宏”

3. 在宏对话框里，选择IfThenAnd并点击“选项”按钮
4. 给你的宏设置快捷键：Ctrl+Shift+I，并且退出宏对话框
5. 在工作表里输入以下数据：

	A	B
1	Units	234
2	Price	7

6. 按下Ctrl+Shift+I运行过程IfThenAnd
7. 改变单元格B1和B2的值，以便你每次运行该过程时，不同的If...Then语句为真

技巧5-2 If指令块和缩进

要使If程序块更容易阅读和理解，可以使用缩进。对比下面的代码书写：

```
If condition Then
action1
End If
If condition Then
    action
End If
```

看看下面的代码，你可以轻易知道该程序块开始在哪里，结尾又在哪里。

4.The If...Then...Else 语句

现在，你知道当一个或多个条件为真或为假时如何显示信息或采取行动。然而，如果你的程序需要在条件为真时采取某个行动，而条件为假时采取另外一个行动，应该怎么办呢？你可以通过添加一个Else子句就可以根据测试的结果将你的过程引导到一个合适的语句。If...Then...Else语句有两种格式——单行和多行。单行的格式为：

```
If 条件 Then 语句1 Else 语句2
```

当条件为真时，执行关键字Then后面的语句，当条件为假时，则执行Else后面的语句。例如：

```
If Sales>5000 Then Bonus = Sales * 0.05 Else MsgBox "No Bonus"
```

如果储存在变量Sales的值大于5000的话，那么VB将使用下述公式：Sales*0.05来计算股红（bonus）。然而，如果变量Sales不大于5000的话，VB就会显示信息“No Bonus”。If...Then...Else语句应该用于决定执行两个操作中的哪一个。当你要执行多个语句时，你最好使用多行格式的If...Then...Else语句：

```
If 条件 Then
    如果条件为真时要执行的语句
Else
    如果条件为假时要执行的语句
End If
```

注意，多行的If...Then...Else语句以关键字End If结束。使用上面显示的缩进使得程序结构易于阅读。在下面的例子中，如果条件ActiveSheet.Name = “Sheet1”为真，VB就执行Then和Else之间的语句，并且忽略Else和End If之间的语句。当条件为假时，VB就忽略Then和Else之间的语句，并且执行Else和End If之间的代码。

```
If ActiveSheet.Name = "Sheet1" Then
    ActiveSheet.Name = "My Sheet" MsgBox "This sheet has been renamed."
Else
    MsgBox "This sheet name is not default."
End If
```

让我们来看看程序示例：

1. 在工程Decisions(Chap05.xls)里插入一个新模块
2. 重命名该模块为IfThenElse
3. 输入下列过程WhatTypeOfDay：

```
Sub WhatTypeOfDay()
    Dim response As String
    Dim question As String
```



```
Dim strmsg1 As String, strmsg2 As String
Dim myDate As Date
```

```
question = "Enter any date in the format mm/dd/yyyy:" _
    & Chr(13) & " (e.g., 11/22/1999)"
strmsg1 = "weekday"
strmsg2 = "weekend"
```

```
response = InputBox(question)
myDate = Weekday(CDate(response))
If myDate >= 2 AND myDate <= 6 Then
    MsgBox strmsg1
Else
    MsgBox strmsg2
End If
End Sub
```

上面的过程要求用户输入任意一个日期。用户提供的字符串通过一个内置函数CDate转变为Date数据类型，最好，函数Weekday将日期转变为一个指明该日期在一周里的天数（参见表5-3）。该整数储存于变量myDate里。条件测试用以检查变量myDate是否大于等于2以及小于等于6。如果测试结果为真，那么用户就被告知，该提供的数据是个工作日；否则，程序宣布这是个周末。

表5-3 内置函数Weekday返回的值

常数	值
vbSunday	1
vbMonday	2
vbTuesday	3
vbWednesday	4
vbThursday	5
vbFriday	6
vbSaturday	7

4. 从VB窗口运行该程序。运行几次，每次提供不同的日期。对照你的桌面或日历检查VB的答案是否正确。

技巧5-3 什么是结构化程序？

结构化编程要求所有的程序具有模块化的设计，并使用三种逻辑结构：顺序，决定和循环。顺序结构为一条接一条地执行语句；决定结构则是让你基于一些条件的测试来执行一些特定的语句；而只要某特定的条件为真，循环结构就重复地执行某条或某些语句。循环是下章的主题。在结构化编程里，其它一些逻辑语句，例如GoTo，是不允许的。结构化程序的代码容易跟踪——它从上到下平稳地走下来，没有任何跳跃到特定标志去的语句。下面就是一个结构化程序和非结构化程序的例子：

非结构化程序：

```
Sub GoToDemo()
Dim num, mystr
    num = 1
If num = 1 Then
    GoTo line1
Else
    GoTo Line2
Line1:
    mystr = "Number equals 1"
    GoTo LastLine
Line2:
```



```

    mystr = "Number equals 2"
LastLine:
    Debug.Print mystr
End sub

```

结构化程序:

```

Sub Structure()
    Dim num, mystr
    num = 1

    If num = 1 Then
        mystr = "Number equals 1"
        Debug.Print mystr
    Else
        mystr = "Number equals 2"
    End if
End Sub

```

当你在起草你的VBA程序，并且需要从一个程序的一行跳到另一行时，你可能会被使用GoTo语句所诱惑，不要跳跃。依赖于使用GoTo语句来你程序的路径将导致令人迷惑的代码，被称为意大利式面条代码。使用结构化编程，你可以轻易地到达你程序里的目的地。

这里是另外一个示范If...Then...Else语句的过程:

```

Sub EnterData()
    Dim cell As Object
    Dim strmsg As String

    On Error GoTo VeryEnd
    strmsg = "Select any cell:"
    Set cell = Application.InputBox(prompt:=strmsg, Type:=8) cell.Select
    If IsEmpty(ActiveCell) Then
        ActiveCell.Formula = InputBox("Enter text or number:")
    Else
        ActiveCell.Offset(1, 0).Select
    End If
    VeryEnd:
End Sub

```

上面的字程序EnterData提示用户选择任意单元格，然后单元格地址赋值于对象变量cell。If...Then...Else结构检查被选择的单元格是否为空。IsEmpty是个内置函数，用来决定某个变量是否已经被初始化了。如果该变量没有被初始化，那么IsEmpty函数返回true（真）。回想我们说过，当变量被赋予第一个值时，它就被初始化了。在本过程中，如果当前单元格为空时，VB将它当作一个零长度的字符串（""）。除了：

```
If IsEmpty(ActiveCell) Then
```

你还可以使用下述指令：

```
If ActiveCell.Value = "" Then
```

如果当前单元格为空，Then后面的语句就会被执行。该语句提示用户输入一个文本或数字，并且一旦数据被提供，该数据就会输入给当前单元格。如果当前单元格不为空，VB将跳到子句Else后面的指令。该指令将让VB选择同列里的下一个单元格。当你运行该过程，信息框提示你选择一个单元格，在工作表上，点击任何单元格。被选择的单元格地址就会出现在信息框的编辑框里面。点击确定退出信息框。VB检查被选择单元格的内容并且跳到你过程里的true或false指令块（true指令块在Then后面，而false指令块在Else后面）。

5.If...Then...ElseIf 语句

很多时候，你需要检查很多种情况，你可以使用子句ElseIf来将一些If条件结合在一起。使用If...Then...ElseIf语句，你可以比用前面章节中的If...Then...Else语句评估更多的条件。这里是If...Then...ElseIf语句的语法：

```
If 条件1 Then
    条件1为真时要执行的语句
ElseIf 条件2 Then
    条件2为真时要执行的语句
ElseIf 条件3 Then
    条件3为真时要执行的语句
ElseIf 条件N Then
    条件N为真时要执行的语句
Else
    所有条件都为假时要执行的语句
End If
```

Else子句是可选的；如果当所有条件为假时，没有要执行的操作，那么你就可以忽略它。

技巧5-4 ElseIf子句

你的程序里可以包括任何多个ElseIf子句和条件。ElseIf子句总是出现在Else子句之前的，只有当ElseIf子句的条件为真时，它的语句才会被执行。

我们来看看下述例子：

```
If ActiveCell.Value = 0 Then
    ActiveCell.Offset(0, 1).Value = "zero"
ElseIf ActiveCell.Value > 0 Then
    ActiveCell.Offset(0, 1).Value = "positive"
ElseIf ActiveCell.Value < 0 Then
    ActiveCell.Offset(0, 1).Value = "negative"
End If
```

该例检查当前单元格的值，并且在相邻的列贴上适当的“标签”（零，正和负）。注意，此时没有使用Else子句。如果第一种情况（ActiveCell.Value = 0）为假，VB将跳到下一个ElseIf语句，并且评估该条件（ActiveCell.Value > 0），如果该值不大于0，VB将跳到下个ElseIf并检查条件ActiveCell.Value < 0。

我们来看看If...Then...Else语句在一个完整的过程中如何工作：

1. 在当前工程里插入一新模块
2. 重命名模块为IfThenElseIf
3. 输入下列过程WhatValue：

```
Sub WhatValue()
    Range("A1").Select
    If ActiveCell.Value = 0 Then
        ActiveCell.Offset(0, 1).Value = "zero"
    ElseIf ActiveCell.Value > 0 Then
        ActiveCell.Offset(0, 1).Value = "positive"
    ElseIf ActiveCell.Value < 0 Then
        ActiveCell.Offset(0, 1).Value = "negative"
    'End If (原文错误，多一个End If)
End If
End Sub
```

因为你需要运行过程WhatValue好几次来测试各种条件，所以，我们给它设置个临时的快捷键。

4. 打开立即窗口，并且输入下列语句：

```
Application.OnKey "^+y", "WhatValue"
```

一旦按下回车键，VB就会运行OnKey方法将过程WhatValue赋予组合键Ctrl+Shift+Y。这个键盘快捷键只是临时的——当你重新启动Excel后它就不起作用了。你同样也可以用Excel界面-工具菜单-宏对话框里的选项来设置快捷键。

5. 切换到Excel界面，并激活Sheet1
6. 在单元格A1里输入0，并且按下Ctrl+Shift+Y。VB将调用过程WhatValue并在单元格B1厘米输入“zero”
7. 在单元格A1里输入任意大于0的数字，并按下Ctrl+Shift+Y，VB将再次调用WhatValue。VB评估第一种条件，因为该测试的结果为假，所以它跳到ElseIf语句。第二个条件为真，因此VB执行Then后面的语句，并且跳过下一条语句，直接到End If。因为End If后面并没有其它的语句了，该过程便结束了，单元格B1现在显示“positive”。
8. 在单元格A1里输入任意小于0的数字，并按下Ctrl+Shift+Y。这次，前面两个条件都返回假，因此VB继续检查第三个条件。因为这次的测试为真，VB就在单元格B1里贴上标签“negative”
9. 在单元格A1里输入任何文本，并按下Ctrl+Shift+Y，VB的反应是“positive”，然而，这不是个满意的答案。你也许希望VB通过显示“text”来区分开正数和文本。要使你的过程WhatValue更“聪明”些，你就需要学习如何通过使用嵌套的If...Then语句来作一些更复杂的决定。

6. 嵌套的 If...Then 语句

将一个If...Then语句或If...Then...Else语句放在另外一个If...Then语句或If...Then...Else语句里面，你可以在你的VBA过程里作出更复杂的决定。这种一个If语句里包含另一个If指令块的结构称为嵌套的If语句。

接下来的过程TestConditions世上节里的过程WhatValue的修正版，演示嵌套的If...Then语句是如何工作的：

```
Sub TestConditions()  
    Range("A1").Select  
    If IsEmpty(ActiveCell) Then  
        MsgBox "The cell is empty."  
    Else  
        If IsNumeric(ActiveCell.Value) Then  
            If ActiveCell.Value = 0 Then  
                ActiveCell.Offset(0, 1).Value = "zero"  
            ElseIf ActiveCell.Value > 0 Then  
                ActiveCell.Offset(0, 1).Value = "positive"  
            ElseIf ActiveCell.Value < 0 Then  
                ActiveCell.Offset(0, 1).Value = "negative"  
            End If  
        Else  
            ActiveCell.Offset(0, 1).Value = "text"  
        End If  
    End If  
End Sub
```

为了使过程TestConditions更容易理解，每个If...Then语句都显示为不同的格式，现在你可以清楚地看到过程使用了三个If...Then程序块。

第一个If块（粗体）检查当前单元格是否为空，如果为真，就会显示信息，然后VB将跳过Else部分找到相应的End If，该语句位于关键字End Sub之前。

如果当前单元格不为空，IsEmpty (ActiveCell)条件返回假，并且VB运行粗体Else下面的单下划线的If块。该单下划线的If...Then...Else语句就是嵌套在第一个If块（粗体）的。该语句检查当前单元格是否是个数字。注意，我们通过另一个内置函数IsNumeric来做这个。如果当前单元格的值不是一个数字，条件就为假，因此，VB跳到单下划线的Else处，并且在B1里输入“text”。然而，如果当前单元格包含个数字时，VB就会运行双下划线的If块，评估每种情况并作出相应的决定。

第一个If块（粗体）被称为外部If语句，这个外部语句包含两个内部的If语句（单下划线和双下划线）。

技巧5-5 嵌套语句

嵌套是指将一种控制结构放在另外一控制结构里面。你将在第六章里的循环结构里看到更多的嵌套的例子。

7.Select Case 语句

为了避免难以弄清的复杂的嵌套的If语句，你可以使用Select Case语句代替。它的语法为：

Select Case 测试表达式

Case 表达式1

 如果表达式1匹配测试表达式的语句

Case 表达式2

 如果表达式2匹配测试表达式的语句

Case 表达式N

 如果表达式N匹配测试表达式的语句

Case Else

 如果没有表达式匹配测试表达式要执行的语句

End Select

你在关键字Select Case和End Select之间放置任意多个条件以测试。子句Case Else是可选的，当你希望可能有条件表达式返回假时使用它。在Select Case语句里，VB将每个表达式和测试表达式相比较。

这里是Select Case语句背后的逻辑。当VB遇到Select Case子句，它记下测试表达式的值。然后它前进到下面的第一个Case子句，如果这个表达式的值和测试表达式的值匹配的话，VB就会执行语句直到遇到另外一个Case子句并且跳到End Select语句。然而，如果第一个Case子句后面的表达式测试结果和测试表达式不匹配时，VB就会检查每一个Case子句，直到它找到一个匹配的为止。如果没有一个Case子句后面的表达式匹配测试表达式的值的话，VB就会跳到Case Else子句并执行该语句直到遇到关键字End Select。注意，Case Else子句是可选的，如果你的程序里面没有使用Case Else并且没有一个Case子句的表达式和测试表达式相匹配，VB就会跳到End Select后面的语句，并且继续执行你的程序。

我们来一个使用Select Case语句的程序例子。在第四章里，你学习了MsgBox函数允许你显示带有一个或多个按钮的信息，你也学习了MsgBox函数的结果可以赋予一个变量。使用Select Case语句，你现在可以基于用户按下的按钮决定采取哪个行动。

1. 在当前工程里插入一新模块
2. 重命名新模块SelectCase.
3. 输入下述过程TestButtons:

```
Sub TestButtons()  
    Dim question As String  
    Dim bts As Integer  
    Dim myTitle As String  
    Dim myButton As Integer  
  
    question = "Do you want to open a new workbook?"  
    bts = vbYesNoCancel + vbQuestion + vbDefaultButton1  
    myTitle = "New Workbook"  
  
    myButton = MsgBox(prompt:=question, buttons:=bts, _ title:=myTitle)  
  
    Select Case myButton  
        Case 6  
            Workbooks.Add  
        Case 7  
            MsgBox "You can open a new book manually later."  
        Case Else  
            MsgBox "You pressed Cancel."  
    End Select  
  
End Sub
```

End Sub

过程TestButtons的第一部分显示一个带有三个按钮的信息框：是，否和取消。用户选择按钮的值赋予变量myButton。

如果用户点击“是”，那么变量myButton就会被赋值常量vbYes或它对应的值6；如果用户点击“否”，那么变量myButton则赋值为常量vbNo或它对应的值7；最后，如果点击了“取消”，变量myButton的内容就等于vbCancel或2。

Select Case语句对照储存在变量myButton里的值检查Case子句提供的值。当有匹配时，就会执行适当的Case语句。如果你使用常量，而不是按钮值，过程TestButtons同样会运行一致。

```
Select Case myButton
    Case vbYes
        Workbooks.Add
    Case vbNo
        MsgBox "You can open a new book manually later."
    Case Else
        MsgBox "You pressed Cancel."
End Select
```

你可以忽略Else子句，可以按下述方法修改一下Select Case语句：

```
Select Case myButton
    Case vbYes
        Workbooks.Add
    Case vbNo
        MsgBox "You can open a new book manually later."
    Case vbCancel
        MsgBox "You pressed Cancel."
End Select
```

4. 运行过程TestButtons三次，每次选择一个不同的按钮。

技巧5-6 通过Case Else捕捉错误

尽管在Select Case语句里使用Case Else不是强制的，使用它总是很好的，以防止万一测试有没有预料到的值。Case Else子句是个放置错误信息的好地方。

8.和 Case 子句一起使用 Is

有时候，作决定是基于测试表达式的条件，例如它是否大于，小于，等于或使用一些其它的关系运算符（参见表5-1）。关键字Is使你能够在Case子句里使用条件表达式。使用关键字Is的Select Case语句的语法如下：

```
Select Case 测试表达式
    Case Is 条件1
        如果条件1为真时执行的语句
    Case Is 条件2
        如果条件2为真时执行的语句
    Case Is 条件N
        如果条件N为真时执行的语句
End Select
```

例如，我们来比较几个数字：

```
Select Case myNumber
    Case Is <10
        MsgBox "The number is less than 10"
    Case 11
        MsgBox "You entered eleven."
    Case Is >=100
        MsgBox "The number is greater than or equal to 100."
    Case Else
        MsgBox "The number is between 12 and 99."
End Select
```

假设变量myNumber为120，那么第三个Case子句为真，并且只有Case Is >=100和Case Else之间的语句会被执行。

9.确定 Case 子句里数值的范围

在前面的例子里，你看到了在每个Case子句里使用一个简单表达式。然而，很多时候，你可能需要在Case子句里确定一个数值范围。可以通过关键字To用于表达式的数值之间来实现它，如下所示：

```
Select Case unitsSold
    Case 1 to 100
        Discount = 0.05
    Case Is <= 500
        Discount = 0.1
    Case 501 to 1000
        Discount = 0.15
    Case Is >1000
        Discount = 0.2
End Select
```

我们来分析一下上面的Select Case代码块，假设变量unitsSold当前值为99。VB将变量unitsSold的值与Case子句的条件表达式进行比较。第一和第三条Case子句示范如何通过使用关键字To在条件表达式里使用数值范围。因为unitsSold=99，第一个Case子句里的条件表达式为真，因此，VB将0.05赋给变量Discount。第二个Case子句如何呢？它也为真。尽管，很明显99小于等于500，VB不会执行相关的语句Discount=0.1。原因是，一旦VB找到了一个真条件的Case子句，它就不会去管其它的Case子句，它将跳过那些代码，继续执行End Select语句后面可能有的语句。

我们来练练使用Select Case语句，在函数过程里使用它。回想在第四章里，函数过程允许你将结果返回给一个子过程。假设该子过程必须根据销售的套数来显示一个折扣，你可以从用户那里获得销售套数，然后允许一个函数来确定需要的折扣：

1. 在模块SelectCase里输入下列子过程：

```
Sub DisplayDiscount()
    Dim unitsSold As Integer
    Dim myDiscount As Single
    unitsSold = InputBox("Enter the number of sold units:")
    myDiscount = GetDiscount(unitsSold)
    MsgBox myDiscount
End Sub
```

2. 输入下列函数过程：

```
Function GetDiscount(unitsSold As Integer)
    Select Case unitsSold
        Case 1 To 200
            GetDiscount = 0.05
        Case Is <=500
            GetDiscount = 0.1
        Case 501 To 1000
            GetDiscount = 0.15
        Case Is >1000
            GetDiscount = 0.2
    End Select
End Function
```

3. 将光标放在过程DisplayDiscount的任意地方并且按下F5来运行它。

过程DisplayDiscount将储存于变量unitsSold的值传递给函数GetDiscount。当VB遇到Select Case语句时，它检查第一个Case子句里的值是否合储存于unitsSold里面的值是否匹配。如果匹配，VB给函数名称赋值百分之五(0.05)，并且跳到关键字End Select。因为，在函数过程里面没有更多需要运行的语句，VB就返回主调过程——DisplayDiscount，在这里，它将函数的结果赋予变量myDiscount。最后的语句用信息框来显示获得的折扣。

10.在 Case 子句里确定多个表达式

你可以使用逗号明确单一Case子句里的多个表达式:

```
Select Case myMonth
    Case "January", "February", "March"
        Debug.Print myMonth & ": 1st Qtr."
    Case "April", "May", "June"
        Debug.Print myMonth & ": 2nd Qtr."
    Case "July", "August", "September"
        Debug.Print myMonth & ": 3rd Qtr."
    Case "October", "November", "December"
        Debug.Print myMonth & ": 4th Qtr."
End Select
```

技巧5-7 Case子句的多个条件

用来分隔开Case子句里面多个条件的逗号，和用于If语句里的运算符OR意义一样。只要这些条件有一个为真，Case子句就为真。

11.接下来...

在本章介绍的条件语句，让你控制你的过程走向。通过测试条件的真假，你可以决定哪些语句需要执行，哪些要跳过。换句话说，不必从上到下，一行一行地运行你的过程，你可以只执行某些行，如果你犹豫应该使用哪种条件语句，这里是一些指南：

- 如果你只要提供一个条件，简单的If...Then语句是最好的选择
- 如果你要决定运行两个条件中的一个，那么使用If...Then...Else语句
- 如果你的程序需要两个或多个条件，那么使用If...Then...ElseIf或者Select Case语句
- 如果你的程序有很多条件，那么就使用Select Case语句。这个语句比If...Then...ElseIf语句更灵活并且更容易理解。

有些决定是需要重复的，例如，你可能需要在工作表里的每个单元格里或者一个工作簿里的每个表里重复同样的操作。下章将教你如何一次又一次地做同样的操作。

第六章 在 VBA 中重复操作

作者: Julitta Korol 翻译: Tiger Chen Feb 1' 2005

既然你已经学习了条件语句如何赋予你的VBA过程作决定的能力，是时候深入了。不是所有的决定都容易，有时候你将需要运行一些语句好几次才能达到某个条件。然而，另一方面，当你达到这个决定后，你可能需要一直运行某些语句，只要条件为真，或直到条件变为真。在编程中，重复地执行任务被称为循环。VBA有好些个循环结构，允许你多次重复一系列的语句。你将在本章里学习如何循环你的代码。

1.Do Loops: Do...While 和 Do...Until

VB有两种Do循环语句，只要或者直到某个条件为真，它们就会重复一系列的语句。只要条件为真，Do...While循环就允许你重复某个操作。这个循环的语法如下：

```
Do While 条件
    语句1
    语句2
    语句N
Loop
```

当VB遇到这个循环时，它首先条件的真假，如果条件为假，循环内部的语句就不会被执行，VB将继续执行关键字Loop后面的第一条语句。如果条件为真，循环里面的语句则会被一条一条地执行，直到遇到Loop语句。Loop语句告诉VB重复这个过程，只要Do While语句里的条件为真的话。

现在，我们来看看如何在Excel里面好好利用Do...While循环语句。在第四章里，你学习了如何根据一个单元格的内容来作决定。让我们再进一步，看看如何在一系列单元格上作同样的决定。该决定是给一系列中的非空单元格设置粗体格式。

1. 打开一个空工作簿，并且命名为Chap06.xls
2. 切换到VB编辑屏幕，并且将新工程改名为Repetition (Chap06.xls)
3. 在工程Repetition里插入一新模块，并重命名为DoLoops
4. 输入如下过程：

```
Sub ApplyBold()  
    Do While ActiveCell.Value <>""  
        ActiveCell.Font.Bold = True  
        ActiveCell.Offset(1, 0).Select  
    Loop  
End Sub
```

5. 在单元格A1:A7里输入任意数据（文本或数字）
6. 选择单元格A1
7. 选择“工具”-“宏”-“运行宏”。在宏对话框里，双击过程ApplyBold（或者选中该过程然后点击运行）

当运行过程ApplyBold时，VB首先评估Do While语句里的条件——ActiveCell.Value<>”，该条件意思是：只要当前单元格的值不是一个空字符串（” ”），就执行下列语句。因为你已经在单元格A1里输入了数据并且激活了该单元格（见第六步），第一个测试返回真，所以VB执行语句ActiveCell.Font.Bold = True，它的意思是给当前单元格设置粗体格式。接下来，VB选择了下一行的单元格（参见第二章里的Offset属性）。因为该语句之后就是关键字Loop，VB返回到Do While语句，并且再次检查条件。如果新选中的单元格（当前激活的单元格）不为空，那么VB就会重复循环内部的语句。该过程会继续，直到检查到单元格A8的内容为空，测试条件的结果为假，因此，VB就跳过循环内部的语句。并且在关键字Loop后面没有其它的语句了，所以该过程就结束了。

我们来看看另外一个Do...While循环的例子。是不是很想知道如何在Excel的状态栏里显示今天的日期和时间？这里有个例子，如何让它显示十秒钟：

```
Sub TenSeconds()  
    Dim stopme  
    stopme = Now + TimeValue("00:00:10")  
  
    Do While Now < stopme  
        Application.DisplayStatusBar = True  
        Application.StatusBar = Now  
    Loop  
    Application.StatusBar = False  
End Sub
```

在上面的程序里，只要函数Now返回的时间小于变量stopme的值，Do...While循环里的语句就会被执行。变量stopme储存值为当前时间加上十秒（参见在线帮助里的另外一个使用内置函数TimeValue的例子）。

语句Application.DisplayStatusBar告诉VB打开状态栏的显示，下条语句则是将当前日期和时间放在状态栏上。当显示时间时（只显示10秒）用户无法使用系统（光标变成了沙漏）。十秒钟后（也就是，当条件Now < stopme为真），VB跳出循环并且执行关键字Loop后面的语句，该语句将状态栏返回到默认信息“就绪”。

技巧6-1 什么是循环？

循环是一种导致一部分程序代码重复执行的编程结构。VBA提供了多种结构在你的过程里执行循环：Do...While, Do...Until, For...Next, For...Each, and While...Wend

Do...While循环还有另外一种语法，你可以在循环的底部测试条件，例如：

Do

```

语句1
语句2
语句N

```

Loop While 条件

当你在循环的底部测试条件时，意味着循环里面的语句至少运行了一次。看一下这个例子：

```

Sub SignIn()
    Dim secretCode As String

    Do secretCode = InputBox("Enter your secret code:")
        If secretCode = "sp1045" Then Exit Do
    Loop While secretCode <> "sp1045"
End Sub

```

注意，在条件被测试之时，VB至少已经执行了一次循环里的语句。除了将条件放在循环之后外，过程SignIn示范如何使用条件跳出循环。当Exit Do语句执行时，循环便立即停止。

技巧6-2 避免无限循环

如果你没有正确地设计你的循环，你将导致一无限循环——永无休止的循环。你将无法使用Esc键来停止该循环。在下面的过程里，因为用户忘了放置测试条件而导致了永无休止的循环：

```

Sub SayHello()
    Do
        MsgBox "Hello."
    Loop
End Sub

```

你必须按下Ctrl+Break键（译者：现在，有些电脑使用别的组合键来中断程序。例如我的手提电脑就是Fn+Break）才能终止该无限循环，当VB显示信息“代码执行被中断”时，点击结束以退出过程。

另外一种方便的循环Do...Until，也可以让你重复一条或多条语句，直到条件为真。换句话说，Do...Until语句是只要当某个条件为假的时候重复一块代码。这是它的语法：

```

Do Until 条件
    语句1
    语句2
    语句N

```

Loop

使用上面的语法，你可以将前面的过程ApplyBold重新写成下面的方式：

```

Sub ApplyBold2()
    Do Until IsEmpty(ActiveCell)
        ActiveCell.Font.Bold = True
        ActiveCell.Offset(1, 0).Select
    Loop
End Sub

```

该过程的第一条语句意思是执行下列语句，直到遇到第一个空单元格。结果上，如果当前单元格不为空，VB就执行循环内部的那两条语句。只要条件IsEmpty(ActiveCell)测试为假，这个过程就反复继续着。因为过程ApplyBold2在循环的前面就测试条件，如果第一个单元格就为空的话，循环内部的语句就不会运行。在下一段，你将有机会试验它。

和Do...While循环类似，Do...Until循环也有第二种语法让你在循环的底部测试条件：

```

Do
    语句1
    语句2
    语句N
Loop Until 条件

```

如果你想要程序至少执行一次，那么就将条件放置于Loop语句一行，无论条件的值是什么。我们来试验一下下面的例子，该例子将工作簿里的空工作表删除。

1. 在你前面创建的DoLoop模块里输入下面的过程DeleteBlankSheets:

```
Sub DeleteBlankSheets()  
    Dim myRange As Range  
    Dim shcount As Integer  
  
    shcount = Worksheets.Count  
    Do  
        Worksheets(shcount).Select  
        Set myRange = ActiveSheet.UsedRange  
  
        If myRange.Address = "$A$1" And _  
            Range("A1").Value = "" Then  
            Application.DisplayAlerts = False  
            Worksheets(shcount).Delete  
            Application.DisplayAlerts = True  
        End If  
        shcount = shcount - 1  
    Loop Until shcount = 1  
End Sub
```

2. 手动在当前工作簿里面插入一些工作表。在一个工作表里输入一些数据与单元格A1；另一个工作表的单元格B2和C10里输入一些数据；第三个工作表里不要输入任何数据。
3. 运行过程DeleteBlankSheets。当你运行该过程时，无论何时，只要两个条件都为真——熟悉UsedRange返回单元格A1并且A1为空，VB就会删除所选的工作表。属性UsedRange应用于对象Worksheet，包含工作表中的每个非空单元格以及他们之间的空单元格。例如，如果你在单元格B2和C10里输入里东西（译者：包括格式），使用了的区域为\$B\$2:\$C\$10。如果你后面又在A1里输入了数据，那么UsedRange将会是\$A\$1:\$C\$10。已使用区域是一个从左上角最远的地方到右下角最远的地方包围起来的区域。因为工作簿至少要保留一个工作表，所以代码执行到变量shcount等于1时就停止了。语句shcount = shcount-1确保变量shcount在循环里面的代码每执行一次就减少1。变量shcount的值在过程的开始处用下列语句：Worksheets.Count初始化了。注意，当删除工作表的时候，Excel通常会显示一个确认对话框，如果你不想看到这个确认提示框的话，就是要下列语句：
Application.DisplayAlerts = False
当你完成任务时，使用下列语句，再打开系统信息。
Application.DisplayAlerts = True

技巧6-3 计数器

计数器是个数字变量，用来追踪已进行的项目次数。上面的过程DeleteBlankSheets声明了变量shcount来追踪检查的工作表个数。计数器变量必须在程序的开始就被初始化（赋值），这可以确保你总能在开始使用之前知道计数器的确切值。计数器可以按照确定的值增加或减少。参加本章后面的使用计数器的For...Next循环。

2.观察过程执行

当你使用循环结构运行过程时，有时很难看到该过程会按预期地执行。有时，你很想观察程序慢慢地运行，这样你就能够检查该程序的逻辑。我们来看看VB如何让你一行接一行地执行程序。

1. 在单元格区域A1:A5里面输入任何数据
2. 选择单元格A1
3. 在Excel窗口，选择“工具”-“宏”-“运行宏”
4. 在宏对话框里，选择ApplyBold2并点击“单步执行”按钮。VB编辑屏幕将出现，过程的名称被黄色加亮（参加图6-1）。注意代码窗口左边的黄色箭头。

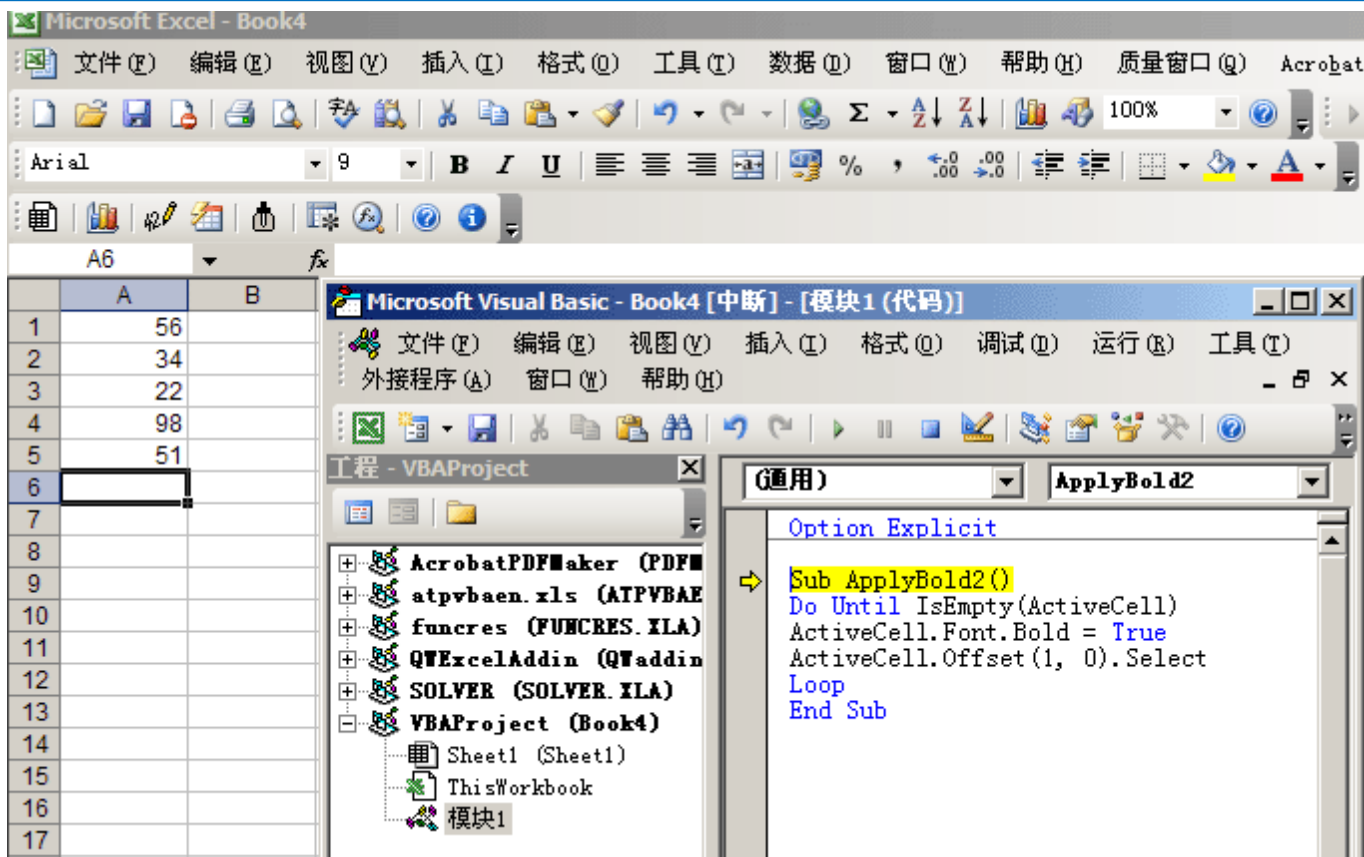


图6-1 观察程序一行接一行地执行

5. 使VB窗口缩小一些，可以点击VB标题栏的“还原”按钮缩小窗口
6. 按下F8，黄色加亮区将跳到Do Until IsEmpty(ActiveCell)行
7. 继续按F8，同时观察代码和工作表窗口

3.While...Wend 循环

While...Wend循环功能上和Do...While循环一样，它是从Microsoft Basic的早期版本遗留下来的并且VBA保留它也是为了支持兼容性。该循环以关键字While开始以关键字Wend结束。这是它的语法：

While 条件

语句1

语句2

语句N

Wend

条件在循环的上面就被测试，只要提供的条件为真，这些语句就会被执行。一旦条件为假，VB就将退出该循环。我们来看一个使用While...Wend循环结构的过程：

1. 在当前工程里插入一新模块，重命名为WhileLoop
2. 输入下述过程：

```
Sub ChangeRHeight()
    While ActiveCell <>""
        ActiveCell.RowHeight = 28
        ActiveCell.Offset(1, 0).Select
    Wend
End Sub
```

3. 在单元格区域B1:B4里输入一些数据
4. 选择单元格B1并且运行过程ChangeRHeight。当当前单元格不为空时，上面的过程ChangeRHeight将设置行高位28。

4.For...Next 循环

当你知道你需要重复运行多少次某段语句时，可以使用For...Next语句。它的语法如下：

```
For 计数器 = 开始 To 结束 [步长]
```

```
    语句1
```

```
    语句2
```

```
    语句N
```

```
Next [计数器]
```

括号里面的代码是可选的。计数器是个储存反复次数的数字型变量，开始是你期望的起始计数点，结束则表明循环应该执行多少次。

例如，你想要重复执行循环里的语句5次，使用下述For语句：

```
For counter = 1 To 5
```

```
    你的语句
```

```
Next
```

当VB遇到关键字Next时，它将回到循环的开始处，并且再次执行循环里面的代码，直到计数器到达结束值。一旦计数器的值大于关键字To后面的数值，VB就会跳出循环。因为计数器变量在每次执行循环后会自动地变化，它早晚会超出结束的值。每次VB执行循环里的语句后，变量计数器的值会默认地增加1，你可以使用Step语句来改变这个默认设置。例如，要使计数器每次增加3，就可以使用以下语句：

```
For counter = 1 To 5 Step 3
```

```
    你的代码
```

```
Next counter
```

当VB遇到上面的语句，它会执行循环里的语句两次。在第一次的循环里，counter等于1，第二次则等于4（3+1）。在执行两次循环后，counter就等于7（4+3），这导致VB退出该循环。

注意，步长（Step）是可选的。可选语句总是显示在方括号里面（参加本段开头部分的语法）。步长不需要明确，除非它不等于1。你可以在Step后面放个负值作为步长，那么VB每次遇到关键字Next后都会将计数器减小。

关键字Next后面的变量名称（counter）也是可选的，然而，好的编程习惯是要强制在关键字Next后面加上计数器。

如何将For...Next循环使用在Excel里面呢？假使你只想要在你的销售报告里面包括某几个特定月份的产品销售，当你从Access导入数据时，你同样也会将那些销售额为0的数据行一起导入。你如何快速取出数据为0的行呢？尽管，有很多种方法可以解决这个问题，但是，我们来看看如何使用For...Next循环来处理这个问题吧。

1. 在VB窗口，在当前工程里插入一个模块并且命名为ForNextLoop
2. 在ForNextLoop模块里输入下列过程：

```
Sub DeleteZeroRows()
```

```
    Dim totalR As Integer
```

```
    Dim r As Integer
```

```
    Range("A1").CurrentRegion.Select
```

```
    totalR = Selection.Rows.Count
```

```
    Range("B2").Select
```

```
    For r = 1 To totalR-1
```

```
        If ActiveCell = 0 Then
```

```
            Selection.EntireRow.Delete
```

```
            totalR = totalR - 1
```

```
        Else
```

```
            ActiveCell.Offset(1, 0).Select
```

```
        End If
```

```
    Next r
```

```
End Sub
```

3. 切换到Excel界面，并且准备下述表格：

	A	B
1	Product Name	Sales (in Pounds)
2	Apples	120
3	Pears	0
4	Bananas	100
5	Cherries	0
6	Blueberries	0
7	Strawberries	160

4. 运行过程DeleteZeroRows。

我们来一行接一行地检查一下过程DeleteZeroRows。开始两语句计算当前区域的总行数，并且将该值储存于变量totalR，接下来，VB选择单元格B2然后遇到关键字For。因为电子表格的第一行包含了列标题，所以要从总行数里减掉1（totalR-1）。VB将需要执行循环里面的指令6次。

嵌套在循环里面的条件语句（If...Then...Else）告诉VB根据当前活动单元格的值作出决定。如果该值为0，VB就删除当前行，并且将总行数减掉1。否则，条件为假，因此，VB将选择下一行的单元格。VB每完成一次循环，它就跳到关键字For来比较r的值和totalR-1的值。当过程结束后，销售表里就不会包含没有销售的产品了。

技巧6-4 成对语句

For和Next必须是成对的，如果有一个漏掉了，VB就将产生一个错误信息“For没有Next”

5.For Each...Next 循环

当你的过程需要在一个集合的所有对象或者一个数组的所有元素（数组将在第七章里涉及）之间循环时，应该使用For Each...Next循环。该循环不需要计数器变量，VB自己知道应该执行几次循环。我们拿工作表集合作个例子，要删除工作簿里面的工作表，你首先不得不要选择它，再选择“编辑”-“删除工作表”。如果要只留一个工作表在工作簿里面的话，你就不得使用同样的命令，次数取决于工作表的总数。因为每个工作表都是工作表集合里的一个对象，所以使用For Each...Next循环来加速删除工作表。该循环的形式是：

For Each 元素 In 组合

语句1

语句2

语句N

Next [元素]

在上面的语法中，元素一个数组或者集合的所有元素都将被赋予的变量，如果是数组的话，该变量必须为Variant数据类型；如果是集合的话，则必须是个对象数据类型。组合是集合的名称或者数组的名称。

现在，我们来使用For Each...Next循环删除工作表。

1. 在当前工程里插入新模块并且重命名为ForEachNextLoop
2. 在模块ForEachNextLoop里输入下列过程：

```
Sub RemoveSheets()  
    Dim mySheet As Worksheet  
    Application.DisplayAlerts = False  
    Workbooks.Add  
    Worksheets("Sheet2").Select  
  
    For Each mySheet In Worksheets  
        ActiveWindow.SelectedSheets.Delete  
    Next mySheet  
End Sub
```

3. 运行过程RemoveSheets。

VB将会打开一个新工作簿并且删除除Sheet1之外的所有工作表。注意，变量mySheet代表工作表集合里的所有对象。除了按通常的方法将对象变量声明为Object类型，你还可以将它声明为更具体的对象类型，这样会更好。在这个具体的例子里，你可

以使用下面的声明：

```
Dim mySheet As Worksheet
```

而不是：

```
Dim mySheet As Object
```

第一条指令Application.DisplayAlerts = False让Excel在过程运行的时候不要显示警告和信息。如果你忽略了它，Excel将会要你确认是否删除所选的工作表。接下来，过程打开一个新工作簿并且选择Sheet2。For Each…Next循环遍历每个工作表（从所选的Sheet2开始）并且删除它们。当过程结束的时候，该工作簿只剩一个工作表Sheet1了。

这里是另外一个检查某个工作表是否存在于一工作簿中：

```
Sub IsSuchSheet()
```

```
    Dim mySheet As Worksheet
```

```
    Dim counter As Integer
```

```
    counter = 0
```

```
    For Each mySheet In Worksheets
```

```
        If mySheet.name = "Sheet2" Then
```

```
            counter = counter + 1
```

```
        End If
```

```
    Next mySheet
```

```
    If counter = 1 Then
```

```
        MsgBox "This workbook contains Sheet2."
```

```
    Else
```

```
        MsgBox "Sheet2 was not found."
```

```
    End if
```

```
End Sub
```

7.提前跳出循环

有时候，你并不想等到循环自己结束，可能是用户输入了错误的的数据，过程遇到了错误或者可能是任务已经完成并且没有必要作更多的循环。你可以提前跳出循环，而不必等到条件正常结束。VB有两种Exit语句：

- Exit For语句用来提前退出For…Next或者For Each…Next循环

- Exit Do语句立即退出任何VBA Do 循环

下面的过程示范如何使用Exit For语句提前跳出For Each…Next循环：

1. 在当前模块里输入下列过程：

```
Sub EarlyExit()
```

```
    Dim myCell As Range
```

```
    For Each myCell in Range("A1:H10")
```

```
        If myCell.Value = "" Then
```

```
            myCell.Value = "empty"
```

```
        Else
```

```
            Exit For
```

```
        End If
```

```
    Next myCell
```

```
End Sub
```

EarlyExit过程检查特定区域A1:H10里每个单元格的内容，如果当前单元格为空，VB就会在当前单元格力输入文本“empty”。当VB遇到第一个非空单元格，它就会跳出循环。

2. 打开一个新工作簿并且在单元格区域A1:H10的任意单元格里输入数据

3. 运行过程EarlyExit

技巧6-5 退出过程

如果你想提前退出子过程，那么可以使用Exit Sub语句。如果该过程是一个函数的话，就使用Exit Function语句代替就行。

8. 循环嵌套

到目前为止，你已经在本章里尝试了很多种循环了，每种过程示范每个循环结构的使用。然而，在编成中，一循环总是放在另外一循环中的。VB允许你将不同类型的循环（For和Do循环）“嵌套”在同一个过程里。当你编写循环嵌套时，请确保每个内部的循环在外部循环里面已经完成。另外，每个循环都必须有其自己独特的计数器变量。如果使用循环嵌套，你可以更有效地执行特定的任务。下面显示的过程ColorLoop示范如何嵌套一个For...Next循环在另一个For...Next循环里面：

```
Sub ColorLoop()
    Dim myRow As Integer
    Dim myCol As Integer
    Dim myColor As Integer

    myColor = 0

    For myRow = 1 To 8
        For myCol = 1 To 7
            Cells(myRow, myCol).Select
            myColor = myColor + 1

            With Selection.Interior
                .ColorIndex = myColor
                .Pattern = xlSolid
            End With
        Next myCol
    Next myRow
End Sub
```

上面的过程ColorLoop使用了两个For...Next循环来改变工作表中前面八行和七列里的每个单元格的 颜色。当外部的循环在追踪行号的时候，内部的循环在做更多的事情，它首先确定当前的列号，基于当前的行号的列号选择适当的单元格，然后给所选的单元格设置颜色。

内部的For...Next循环给工作表的第一行的七个单元格（A1，B1，C1，D1，E1，F1和G1）设置不同的颜色。当变量myCol大于7时，VB跳回外部循环并且变量myRow增加1，再回到内部循环去设置下一行单元格的 颜色。当过程结束时，56个单元格（8*7）被设置了当前调色板上可用的所有颜色。第一个单元格，A1，被设置了黑色（颜色索引号为1），第二个单元格B1则被设置为白色了（颜色索引号为2）。每次单元格地址变化——Cells(myRow, myCol).Select——变量myColor的内容也会改变——myColor = myColor + 1

9. 接下来...

在本章里，你学习了如何在循环里重复一组代码。通过使用好几种类型的循环，你看到了每种循环稍稍不同地进行重复。你有了经验后，你将更容易地选择合适的控制结构来执行你的任务。

在本书的后续章节中，将会有更多的使用循环的例子。例如，在下章里，你将看到如何使用数组合嵌套的循环来创建一个VBA过程，该过程将帮你选择彩票号码。在下章里，你将学习如何处理大量的数据，而不会迷失在变量的海洋里。

第七章 利用 VBA 数组管理数据清单和表格

作者: Julitta Korol 翻译: Tiger Chen Feb 1' 2005

在前面的章节里，你在很多VBA过程里使用变量来储存特定的对象信息，属性或者数值。对于你想要处理的单个数值，你可

以声明变量，但是，对于一系列的数值呢？如果你不得不编写VBA过程来处理大量的数据，你就得声明足够的变量来处理所有的数据。你能想象将世界上所有国家的货币交换利率储存在你的程序的噩梦吗？要创建一个表格来储存这些必要的数据的话，你至少要给每个国家创建三个变量：国家名称，货币名称和交换比率。幸运的是，VB有方法来解决该问题。将相关的变量归为一类，你的VBA过程可以轻松处理大量的数据。在本章里，你将学习如何使用数组来操作数据清单和数据表。

1.了解数组

在VB里，数组一种特殊的变量，代表拥有相同数据类型（字符串，整型，货币，日期，等等）的一组相似的数值。两种最通常的数组是一维数组（清单）和二维数组（表格）。有时，一维数组被称为清单。一维数组或编号清单的例子有：购物清单，星期名称的清单或员工清单。清单里面的每个值都有一个索引。下面是一个含有六个成员的清单的图解：

项目（1）	
项目（2）	
项目（3）	
项目（4）	
项目（5）	
项目（6）	

注意，列代表一维的当前为空的数组。如果你想用数据填充这个数组，只要使用一个变量名称，附带括号编号就行，而不需要使用六个不同的标签。在上面的图解里，“项目”一变量名称，括号里的数字明确数组里的每个成员。

数组的所有成员都必须具有相同的数据类型，换句话说，一个数组不能同时储存字符串和整型数据。接下来的图解是一维数组的两个例子：第一个叫做cities的一维数组由文本组成（字符串数据类型——\$），第二个叫做lotto的一维数组则包含六个抽奖号码（整数数据类型——%）。

一维数组cities\$ （字符串数据类型）		一维数组lotto% （整数数据类型）	
Cities(1)	Baltimore	Lotto(1)	25
Cities(2)	Atlanta	Lotto(2)	4
Cities(3)	Boston	Lotto(3)	31
Cities(4)	Washington	Lotto(4)	22
Cities(5)	New York	Lotto(5)	11
Cities(6)	Trenton	Lotto(6)	5

正如你看到的，每个数组成员的内容和变量的数据类型是相匹配的。如果你想要在同一数组里面储存不同数据类型的数据，那么你必须将数据声明为Variant。

二维数组是由行和列代表的数据表。表中每个成员的位置是由它的行和列号码决定的。下面是一个空的二维数组的图解。

行号	1	2	3	列号
1	(1,1)	(1,2)	(1,3)	
2	(2,1)	(2,2)	(2,3)	
3	(3,1)	(3,2)	(3,3)	
4	(4,1)	(4,2)	(4,3)	
5	(5,1)	(5,2)	(5,3)	

注意，二维数组里的项目是如何有行和列索引指定的？在该图解里，数组里的第一个成员位于第一行和第一列里(1,1)，而最后一个成员则位于第五行和第三列里的(5,3)。下面，我们来给该数组填充一些数据。下面显示的二维数组储存了国家名称，它的货币名称以及和美元的汇率。

Japan (1,1)	Japanese Yen (1,2)	128.2 (1,3)
Mexico (2,1)	Mexican Peso (2,2)	9.423 (2,3)
Canada (3,1)	Canadian Dollar (3,2)	1.567 (3,3)
Norway (4,1)	Norwegian Krone (4,2)	8.351 (4,3)

Hungary (5,1)	Hungarian Forint (5,2)	266.7 (5,3)
------------------	---------------------------	----------------

尽管VBA数组最大可以拥有60维，但是，绝大多数人发现非常困难去想象超过三维的数组。三维的数组是一个具有相同行数和列数的表格的集合。在三维数组里的每个成员由下面三个数据决定：行号，列号和表格号。

技巧7-1 数组变量是什么？

数组是拥有共同名称的变量的集合。一个典型的变量只能储存一个数据，然而，一个数组变量却能够储存大量的变量。你可以使用变量名称和索引号来指向数组中某个确定的数据。

技巧7-2 下标变量

数组变量的括号里的数字成为下标，而每个单独的变量则称为下标变量或成员。例如，`cities(6)`是`cities`数组里的第六个下标变量（成员）。

2. 声明数组

因为数组也是变量，所以，你必须用声明其它变量的类似方法声明数组——使用`Dim`语句。当你声明一个数组时，你便设定了该数组储存数据所需要的内存空间。

我们来看看一个数组声明的例子：

```
Dim cities(6) As String
Dim daysOfWeek(7) As String
Dim lotto(6) As Integer
Dim exchange(5, 3) As Variant
```

注意，变量名称后面带有括号以及括号里有数字。一维数组要求括号里带一个数字，这个数字决定了这个数组能够储存的最大成员数。二维数组后面总是带有两个数字——第一个数字是行索引号，而第二个数字是列索引号。在上面的例子里，数组`exchange`最多可以储存15个数据($5 \times 3 = 15$)。

数组声明的最后一部份是定义数组将要储存数据的数据类型。数组可以储存下列任何一种数据类型：`Integer`, `Long`, `Single`, `Double`, `Variant`, `Currency`, `String`, `Boolean`, `Byte`, or `Date`。

当你声明了一个数组，VB会自动占据足够的内存空间，分配的内存空间取决于该数组的大小和数据类型。当你声明一个名叫`lotto`的带有6个成员的一维数组时，VB将留出12个字节——数组的每个成员各占2个字节（回想整型数据类型为2个字节，因此 $2 \times 6 = 12$ ）。数组越大，储存数据需要的内存空间就越大。因为数组会吃掉很多内存，并因此影响你电脑的运行，因此，建议你仅仅根据你可能使用的成员数来声明数组。

3. 数组的上界和下界

VBA默认将数组的第一个成员设置为0（译者：索引号），因此，数字1代表数组中的第二个成员，而数字2则代表第三个，等等。因为数字编号起始于0，所以，一维数组`cities(6)`包含从0到6的七个成员。如果你宁愿从1开始计数你数组里的成员，那么你可以使用`Option Base 1`语句来强制指定该数组的下界。该指令必须置于VBA模块任何`Sub`语句上面的声明部分。如果你不明确`Option Base 1`，那么VBA在使用数组是就会假定使用`Option Base 0`来从0开始编号你的数组成员。

你也可以让数组从除0或1之外的数字开始编号，要达到该目的，你在声明数组变量时就必须明确该数组的边界。数组的边界是指它最小和最大的索引号。我们来看看下面的例子：

```
Dim cities(3 To 6) As Integer
```

上面的语句声明了一个带有四个成员的一维数组。数组名称后面括号里的数字明确了数组的下界（3）和上界（6）。该数组的第一个成员编号为3，第二个为4，第三个为5，以及第四个为6。注意下界和上界之间的关键字`To`。

技巧7-3 数组范围

`Dim`语句明确的数组的下标区间就称为数组的范围，例如：`Dim mktgCodes(5 To 15)`

4. 在 VBA 过程里使用数组

你声明了数组后，就必须给该数组的每个成员赋值，这也经常成为“填充数组”。我们来尝试使用一维数组有规划地显示六个美国城市的清单：

1. 打开一个新工作簿，并保存为Chap07.xls
2. 切换到VB编辑器窗口，并重新命名VBA工程为Tables
3. 插入一新模块，重新命名为StaticArrays
4. 输入下列过程FavoriteCities:

```
'start indexing array elements at 1 从1开始给数组成员编号
Option Base 1
```

```
Sub FavoriteCities()
```

```
'now declare the array
Dim cities(6) As String
```

```
'assign the values to array elements
```

```
cities(1) = "Baltimore"
cities(2) = "Atlanta"
cities(3) = "Boston"
cities(4) = "Washington"
cities(5) = "New York"
cities(6) = "Trenton"
```

```
'display the list of cities
```

```
MsgBox cities(1) & Chr(13) & cities(2) & Chr(13) _
    & cities(3) & Chr(13) & cities(4) & Chr(13) _
    & cities(5) & Chr(13) & cities(6)
```

```
End Sub
```

在FavoriteCities过程开始之前，缺省的索引编号方式改变了，注意，Option Base 1语句是位于模块窗口Sub语句之上的。该语句告诉VB给数组的第一个成员赋值数字1，而不是缺省的0。

数组cities()声明为带六个成员的字符串类型变量。然后，给数组的每个成员都赋上了值。最后的语句使用Msgbox函数显示城市清单。当你运行该过程时，城市名称将会出现在分开的行上（参见图7-1）。你可以改变显示数据的顺序，改变索引号。

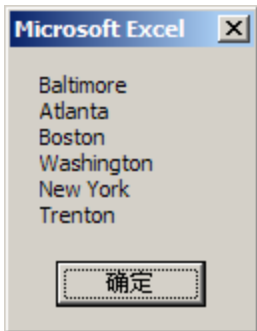


图7-1 你可以用Msgbox函数来显示一维数组的成员

5. 运行FavoriteCities过程并且检查结果
6. 修改FavoriteCities过程，让它逆序显示城市名称（从6到1）

技巧7-4 数组成员的初始值

在给数组成员赋值之前，该成员具有缺省值。数字变量的缺省值为0，而字符串变量的缺省值为空字符串。

5.数组和循环语句

现在要执行一些例如填充数组或显示数组成员的任务了，你在第六章里学过的好些个循环语句（参见For...Next和For Each ...Next循环）就变得非常方便了。现在是时候将你所学到的技巧结合起来使用了。如何重新编写FavoriteCities过程，让每个城市名称在不同的信息框里显示出来？

下面显示的过程FavoriteCities2将原来过程的最后部分取代为For Each...Next循环：


```

Sub FavoriteCities2()
'now declare the array
Dim cities(6) As String
Dim city As Variant

'assign the values to array elements
cities(1) = "Baltimore"
cities(2) = "Atlanta"
cities(3) = "Boston"
cities(4) = "Washington"
cities(5) = "New York"
cities(6) = "Trenton"

'display the list of cities in separate messages
For Each city in cities
    MsgBox city
Next
End Sub

```

注意For Each...Next循环使用的是Variant数据类型的变量city。回想在前面的章节里，For Each...Next让你在一个集合的所有对象间或者一个数组的所有的成员间循环，并且对每个对象或成员执行同样的操作。当你运行过程FavoriteCities2时，数组里有几个成员循环就会执行几次。

我们来看一下过程FavoriteCities的另一种变化。在第四章里，你练习了将参数作为变量传递给子过程和函数。过程FavoriteCities3示范了如何将数组的成员传递给另一个过程。

1. 在当前模块里，输入下述两个过程：

```

Sub FavoriteCities3()
'now declare the array
Dim cities(6) As String

'assign the values to array elements
cities(1) = "Baltimore"
cities(2) = "Atlanta"
cities(3) = "Boston"
cities(4) = "Washington"
cities(5) = "New York"
cities(6) = "Trenton"

'call another procedure and pass the array as argument
Hallo cities()
End Sub

Sub Hallo (cities() As String)
Dim counter As Integer
For counter = 1 to 6
    MsgBox "Hello " & cities(counter)
Next
End Sub

```

过程Hallo的声明里有一个数组类型的参数——cities()。

2. 运行过程FavoriteCities3。将一个子过程的数组成员传递给另一个子过程或者函数过程让你可以在许多过程里使用相同的数组，而不需要重复的程序代码。

技巧7-5 在过程之间传递数组

当一个数组在一个过程里被声明时，它是局部的，并且是不为其他过程所知的。然而，你可以将局部数组传递给其它的过程，通过在声明语句里，写上数组名称，并且后面紧跟一对空括号。例如，语句Hallo cities() 调用一个名叫Hallo的过程，并且将数组cities()传递给它。

这里有个例子，如何将你新学到的关于数组的知识和循环运用到现实生活中。如果你是个狂热的彩票玩家的话，当你厌倦了选择你的幸运号码，你可以让VB为你选择。下面的过程Lotto使用1到51的六个数字填充数组：

```
Sub Lotto()
    Const spins = 6
    Const minNum = 1
    Const maxNum = 51

    Dim t As Integer 'looping variable in outer loop 外部循环变量
    Dim i As Integer 'looping variable in inner loop 内部循环变量
    Dim myNumbers As String 'string to hold all picks 储存选号的字符串
    Dim lucky(spins) As String 'array to hold generated picks 储存产生的选号的数组
    myNumbers = ""

    For t = 1 To spins
        Randomize
        lucky(t) = Int((maxNum-minNum+1) * Rnd) + minNum

        'see if this number was picked before 检查本数字是否之前被选出来过
        For i = 1 To (t-1)
            If lucky(t)=lucky(i) Then
                lucky(t) = Int((maxNum-minNum+1) * Rnd) + minNum i = 0
            End If
        Next i

        MsgBox "Lucky number is " & t & lucky(t)
        myNumbers = myNumbers & " -" & lucky(t)
    Next t

    MsgBox "Lucky numbers are " & myNumbers
End Sub
```

Randomize 语句将随机数字发生器初始化。指令 `Int((maxNum-minNum+1) * Rnd + minNum)` 使用函数 Rnd 来产生一个在 minNum 和 maxNum 之间的随机数值。函数 Int 将随机数转变为一个整数。除了给 minNum 和 maxNum 赋予常量之外，你也可以使用函数 InputBox 从用户那里获得数据。

内部 For...Next 循环确保每个选出的数字是唯一的——它不能是之前选出的任何一个数字。如果你忽略了内部循环并且多次运行该过程，你很可能看到重复的号码。

6.使用二维数组

既然你已经知道了如何有规划地产生一个清单（一维数组），是时候仔细看一下如何使用数据表了。下面的过程产生一个二维数组，储存国家名称，货币名称和交换汇率。

```
Sub Exchange()
    Dim t As String
    Dim r As String
    Dim Ex(3, 3) As Variant
    t = Chr(9) 'tab
    r = Chr(13) 'Enter

    Ex(1, 1) = "Japan"
    Ex(1, 2) = "Yen"
    Ex(1, 3) = 128.2
    Ex(2, 1) = "Mexico"
    Ex(2, 2) = "Peso"
    Ex(2, 3) = 9.423
    Ex(3, 1) = "Canada"
    Ex(3, 2) = "Dollar"
```

```

Ex(3, 3) = 1.567
MsgBox "Country " & t & t & "Currency" & t & "per US$" _
& r & r _
& Ex(1, 1) & t & t & Ex(1, 2) & t & Ex(1, 3) & r _
& Ex(2, 1) & t & t & Ex(2, 2) & t & Ex(2, 3) & r _
& Ex(3, 1) & t & t & Ex(3, 2) & t & Ex(3, 3), , _
"Exchange"
End Sub

```

当你运行过程Exchange时，你将看到一个信息框，显示三列信息（见图7-2）

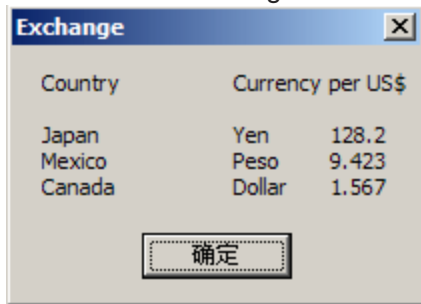


图7-2 显示在信息框上的文本是可以自定义格式的。

7.静态和动态数组

到目前为止，本章介绍的都是静态数组。静态数组是具有确定大小的数组。当你事先知道数组的大小时使用静态数组。静态数组的大小是在数组的声明语句里确定的，例如，语句`Dim Fruits(10) As String`声明了一个由10个成员组成的叫做Fruits的静态数组。

但是，万一你不肯定你的数组会包含多少个成员呢？如果你的过程由用户输入决定，每次程序执行时，用户提供的成员数可能会变化的。你如果确保你声明的数组不会浪费内存呢？

回想你声明了一个数组后，VBA会留出足够的内存来储存数组。如果你声明一个比你需要的更多成员的数组的话，你将浪费计算机资源。这个问题的解决方法是让你的数组变为动态的。动态数组是大小可以改变的数组。如果数组的大小每次都由程序运行而决定的话，就使用动态数组。

技巧7-6 固定大小的数组

静态数组包含固定成员个数。静态数组的成员个数在它被声明后就再也不能改变了。

要声明动态数组，那么不要在数组名称后面的括号里放置数字：

```
Dim Fruits( ) As String
```

动态数组通过在数组名称后面附带空括号来声明。在你使用动态数组于过程里之前，你必须使用`ReDim`语句来动态地设置数组的上界和下界。`ReDim`语句随着程序代码的执行重新设定数组大小，`ReDim`语句通知VB数组的新大小，这个语句可以在同一个过程里多次使用。现在，我们来看看如何使用动态数组。

1. 在当前工程里插入一个新模块并且重新命名为DynamicArrays
2. 输入下列过程DynArray:

```

Sub DynArray( )
    Dim counter As Integer

    'declare a dynamic array
    Dim myArray( ) As Integer

    'specify the initial size of the array
    Redim myArray(5)

    Worksheets.Add

    'populate myArray with values

```

```

For counter = 1 to 5
    myArray(counter) = counter + 1
    ActiveCell.Offset(counter-1, 0).Value = myArray(counter)
Next

```

```

'change the size of myArray to hold 10 elements
Redim Preserve myArray(10)

```

```

'add new values to myArray
For counter = 6 To 10
    myArray(counter) = counter * counter
    With ActiveCell.Offset(counter-1, 0)
        .Value = myArray(counter)
        .Font.Bold = True
    End with
Next counter
End Sub

```

3. 将你的Excel窗口和VB编辑器窗口并排显示

4. 逐步运行过程DynArray。你可以将鼠标置于代码中间，并且按下F8来执行逐条语句。程序DynArray的结果如下图所示。

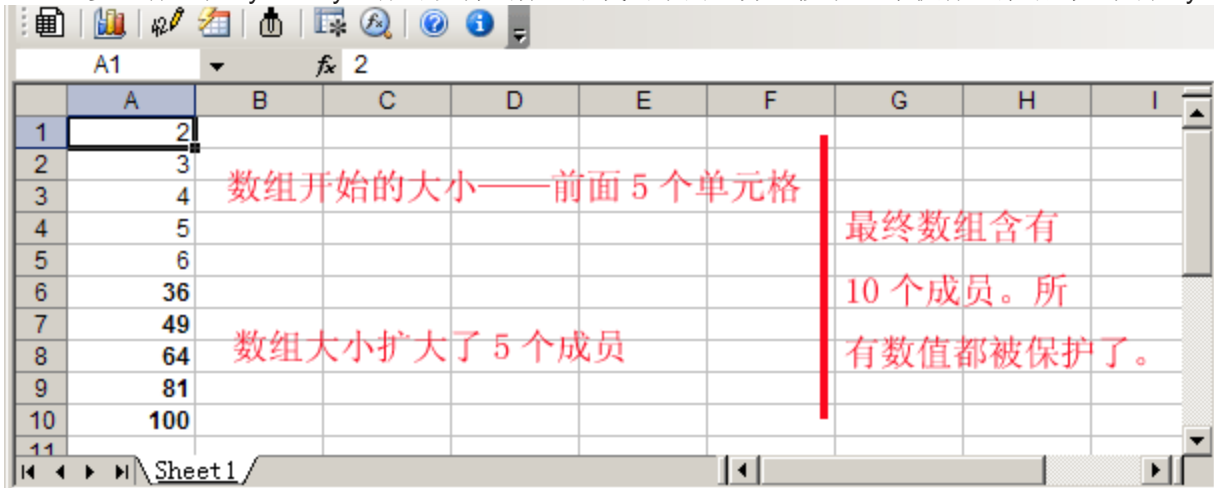


图7-3 显示10个数据的数组

在过程DynArray里，Dim myArray() As Integer语句声明了一个叫做myArray的动态数组。尽管该语句声明了数组，但是没有分配任何内存给该数组。第一条ReDim语句明确了myArray的开始大小并且占据了10个字节的内存让它储存5个成员，正如你所知，每个整型数据需要两个字节的内存。语句Workbooks.Add打开一新工作簿，然后For...Next循环用数据填充数组myArray并且将数组的成员写入工作表。在循环开始之前，变量counter等于1。循环里的第一条语句：

```
myArray(counter) = counter + 1
```

分配数值2给myArray的第一个成员。第二条语句：

```
ActiveCell.Offset(counter-1, 0).Value = myArray(counter)
```

将myArray成员的值输入到当前单元格里。当前单元格为A1。因为变量counter等于1，所以上面的语句就等于：

```
ActiveCell.Offset(1-1, 0).Value = myArray(1)
```

或者

```
ActiveCell.Offset(0,0).Value = myArray(1)
```

上面的语句在单元格A1里输入数据。循环里面的语句被执行5次。VB在合适的工作表单元格里马输入数据并且进行到下一语句：

```
ReDim Preserve myArray(10)
```

通常，当你改变一个数组的大小时，你将失去该数组原来的所有数据。语句ReDim将数组重新初始化。

然而，你可以将新成员加入到现存的数组里去，通过在语句ReDim后面带上关键字Preserve。换句话说，关键字Preserve保证重新改变大小的数组不会弄丢现有的数据。如果你忽略它，新数组将会是空的。

第二个For...Next循环给数组myArray的第六，第七，第八，第九和第十个成员赋值。这次，数组成员的数值是相乘的：counter * counter。VB使用粗体将数组其它的数值输入到合适的工作表的单元格里面。

技巧7-7 确定数组大小

在使用数组之前，必须在Dim或ReDim语句里确定数组的大小。这意味着你不可以给数组成员赋值，直到你使用Dim或者ReDim语句声明了该数组。

8.数组函数

你可以通过五个VBA内置函数来操作数组：Array, IsArray, Erase, LBound和UBound。接下来的章节将示范每个函数在VBA过程里的使用。

9.Array 函数

Array函数允许你在代码执行中间创建一个数组，而不必事先确定其大小。该函数总是返回一个Variant数组。使用函数Array你可以快速地将一系列数据放置在一个清单里面。下面的过程CarInfo创建了一个叫做auto的固定大小，一维的三个成员的数组。

1. 在当前工程里插入一新模块，重命名为Array_Function
2. 输入下列过程CarInfo:

```
Option Base 1
```

```
Sub CarInfo()
```

```
    Dim auto As Variant
```

```
    auto = Array("Ford", "Black", "1999")
```

```
    MsgBox auto(2) & " " & auto(1) & ", " & auto(3)
```

```
    auto(2) = "4-door"
```

```
    MsgBox auto(2) & " " & auto(1) & ", " & auto(3)
```

```
End Sub
```

另外一个例子，示范如何使用Array函数将列标输入到工作表里：

```
Sub ColumnHeads()
```

```
    Dim heading As Variant
```

```
    Dim cell As Range
```

```
    Dim i As Integer
```

```
    i = 1
```

```
    heading = Array("First Name", "Last Name", "Position", _  
    "Salary")
```

```
    Workbooks.Add
```

```
    For Each cell in Range("A1:D1")
```

```
        cell.Formula = heading(i)
```

```
        i = i+1
```

```
    Next
```

```
    Columns("A:D").Select
```

```
    Selection.Columns.AutoFit
```

```
    Range("A1").Select
```

```
End Sub
```

10.IsArray 函数

使用IsArray函数你可以测试某个变量是否数组。如果该变量是个数组，那么IsArray函数返回True，否则返回False。请看例子：

1. 在当前工程里插入模块，命名为IsArray_Function
2. 输入如下过程IsThisArray:

```
Sub IsThisArray()
    'declare a dynamic array 声明一动态数组
    Dim sheetNames() As String
    Dim totalSheets As Integer
    Dim counter As Integer
    'count the sheets in the current workbook 计数当前工作簿里的工作表数目
    totalSheets = ActiveWorkbook.Sheets.Count

    'specify the size of the array 明确数组大小
    ReDim sheetNames(1 To totalSheets)
    'enter and show the names of sheets 输入和显示工作表名称
    For counter = 1 to totalSheets
        sheetNames(counter) = ActiveWorkbook.Sheets(counter).Name
        MsgBox sheetNames(counter)
    Next counter
    'check if this is indeed an array 检查它是否确实为数组
    If IsArray(sheetNames) Then
        MsgBox "The sheetNames is an array."
    End If
End Sub
```

11.Erase 函数

当你要清除数组里的数据时，应该使用Erase函数。该函数删除静态或动态数组储存的所有数据，另外，对于动态数组，Erase函数将重新分配原来分配给该数组的所有内存。下面的例子教你如何删除数组cities里的数据。

1. 在当前工程里插入一新模块，重命名为Erase_Function
2. 输入如下过程FunCities:

```
' start indexing array elements at 1
Option Base 1

Sub FunCities()
    'declare the array
    Dim cities(1 to 5) As String

    'assign the values to array elements
    cities(1) = "Las Vegas"
    cities(2) = "Orlando"
    cities(3) = "Atlantic City"
    cities(4) = "New York"
    cities(5) = "San Francisco"

    'display the list of cities
    MsgBox cities(1) & Chr(13) & cities(2) & Chr(13) _
        & cities(3) & Chr(13) & cities(4) & Chr(13) _
        & cities(5)

    Erase cities

    'show all that was erased
    MsgBox cities(1) & Chr(13) & cities(2) & Chr(13) _
        & cities(3) & Chr(13) & cities(4) & Chr(13) _
        & cities(5)
End Sub
```


在函数Erase清除数组里的数据后，函数MsgBox就显示一个空信息框了。

12.LBound 函数和 UBound 函数

LBound函数和UBound函数分别返回表明数组的下界和上界的数字。

1. 在当前工程里插入模块，命名为L_and_UBound_Function
2. 输入如下代码FunCities2:

```
Sub FunCities2()
    'declare the array
    Dim cities(1 to 5) As String

    'assign the values to array elements
    cities(1) = "Las Vegas"
    cities(2) = "Orlando"
    cities(3) = "Atlantic City"
    cities(4) = "New York"
    cities(5) = "San Francisco"

    'display the list of cities
    MsgBox cities(1) & Chr(13) & cities(2) & Chr(13) _
        & cities(3) & Chr(13) & cities(4) & Chr(13) _
        & cities(5)

    'display the array bounds
    MsgBox "The lower bound: " & LBound(cities) & Chr(13) _
        & "The upper bound: " & UBound(cities)
End Sub
```

当你要确定一个二维数组的上下界时，你就必须明确维数：1表示第一维，2表示第二维。

在本章早先时候将的Exchange过程里的后面加上如下语句，可以确定该二维数组的上下界（将下列代码加入到关键字End Sub之前）：

```
MsgBox "The lower bound (first dimension) is " _
    & LBound(Ex, 1) & "."
MsgBox " The upper bound(first dimension) is " _
    & UBound(Ex, 1) & "."
MsgBox "The lower bound (second dimension) is " _
    & LBound(Ex, 2) & "."
MsgBox " The upper bound(second dimension) is " _
    & UBound(Ex, 2) & "."
```

13.数组中的错误

使用数组时，出错是很容易的。如果你试图给数组赋予比声明数组时更多的成员的话，VBA就会显示错误信息“下标越界”

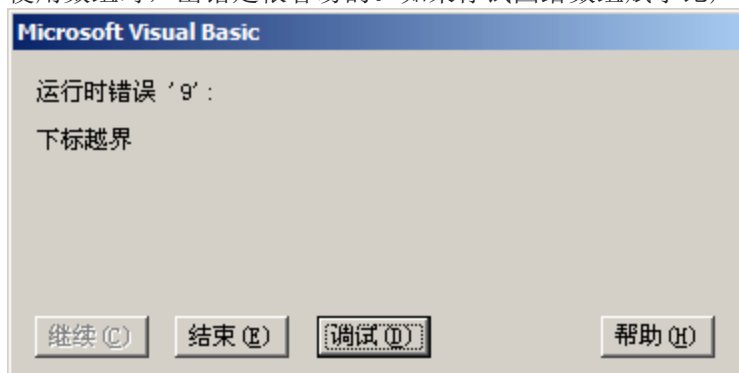


图7-4 该错误出现于试图访问并不存在的数组成员

假设你声明了一个包含6个成员的一维数组，而你却试图给第八个成员赋值，当你运行该过程时，VB无法找到第八个成员，所以显示错误信息。点击调试按钮，VB将导致错误的代码行（见图7-5）加亮。检查数组的声明语句，并且更改被加亮代码行括号里的索引号。

“下标越界”错误经常是由使用循环的过程引发的。下面的过程Zoo1就是这种情况的一个例子。在用户取消在输入框里输入数据之前，循环里的语句反复被执行。在执行该过程时，当变量 i 等于4的时候，VB无法在这个只有三个成员的数组里找到第四个成员，那么错误信息就出现了。修改后的过程Zoo2示范了前面章节里介绍的LBound和UBound函数如何能够避免试图访问不存在的数组成员的错误。

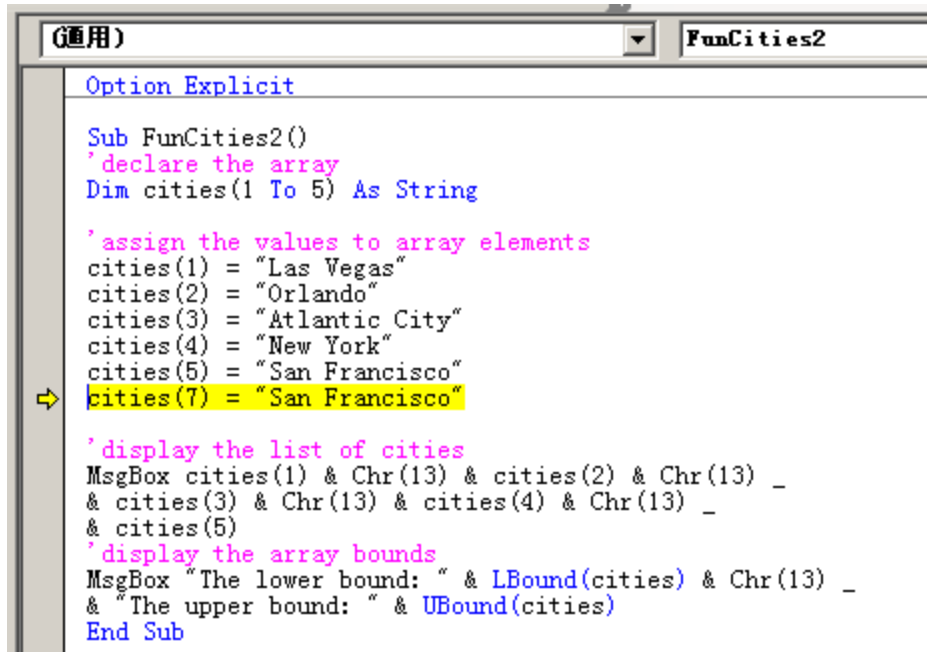


图7-5 当你点击错误信息的调试按钮，VB就会加亮引发错误的语句

1. 在当前工程里插入新模块，命名为Errors_In_Arrays
2. 输入下列过程Zoo1和Zoo2:

```
Sub Zoo1()
    'this procedure triggers an error "Subscript out of range" 本过程引发“下标越界”错误
    Dim zoo(3) As String
    Dim i As Integer
    Dim response As String

    i = 0
    Do
        i = i + 1
        response = InputBox("Enter a name of animal:")
        zoo(i) = response
    Loop until response = ""
End Sub

Sub Zoo2()
    'this procedure avoids the error "Subscript out of range" 本过程避免“下标越界”错误
    Dim zoo(3) As String
    Dim i As Integer
    Dim response As String

    i = 1
```

```

Do While i>=LBound(zoo) And i <=UBound(zoo)
    response = InputBox("Enter a name of animal:")
    If response = "" Then Exit Sub
    zoo(i) = response
    i = i + 1
Loop
For i = LBound(zoo) To UBound(zoo)
    MsgBox zoo(i)
Next
End Sub

```

另外一个使用数组时经常碰到的错误是类型不匹配。要避免这类错误，就要牢记一个数组的每个成员都必须具有相同的数据类型。如果你试图给数组成员赋予和数组声明的数据类型矛盾的数据的话，你就将在执行代码时收到“类型不匹配”的错误。要让一个数组出错不同类型的数据类型的话，你就得声明数组为Variant类型。

14. 数组作为参数

在第四章里面，你学习了数据可以在子过程或者函数过程之间作为必须或者可选参数传递。如果传递的参数不是过程执行一定要的话，那么这个参数名称就应该在前面加关键字Optional。然而，有些时候，你事先并不知道你要传递多少个参数。一个典型的例子就是加法。你可能想要将两个数字加和，或者，你也许要加和3个，10个，或者15个数字。使用关键字ParamArray，你就可以将一个包含任意个成员的数组传递给你的子过程和函数过程。下面的函数过程AddMultipleArgs将加和你所需要的任何多个数字。该函数以数组myNumbers的声明开始，注意关键字ParamArray的使用。该数组必须声明为Variant类型，并且它必须是函数过程的最后一个参数。

1. 在当前工程里插入一新模块，命名为ParameterArrays
2. 输入如下函数过程AddMultipleArgs：

```

Function AddMultipleArgs(ParamArray myNumbers() As Variant)
    Dim mySum As Single
    Dim myValue As Variant

    For each myValue in myNumbers
        mySum=mySum+myValue
    Next
    AddMultipleArgs = mySum
End Function

```

3. 激活立即窗口来试验上面的函数，在立即窗口里输入指令：

```
?AddMultipleArgs(1, 23.24, 3, 24, 8, 34)
```

当你按下回车键，VB就会返回上面参数的总和：93.24。你可以提供无限制的参数数目。注意，每个函数的参数之间要用逗号分开。

15. 接下来...

在本章里，你学习了通过创建数组，你可以编写需要大量变量的过程。通过例子，示范了如何声明和使用一维数组（清单）和二维数组（表）。你也学习了静态数组和动态数组之间的差别。本章的结尾介绍了五个VBA内置函数和关键字ParamArray，它们经常使用于数组。

现在你已经知道了可以使你等程序更智能的控制结构了：条件语句，循环和数组。在本书的后面，你将学习如何使用集合代替数组来操作更大量的数据。通过使用从第一到第七章学到的知识，你现在可以开始编写VBA过程让的任务自动化了，而这些在你开始学习本书之前似乎是不可能的事情。接下来的一章将讲述文件和文件夹的管理。

第八章 利用 VBA 操纵文件和文件夹

作者：Julitta Korol 翻译：Tiger Chen Mar 5' 2005

在工作过程中，你肯定访问、创建、复制和删除过成百上千的文件和文件夹。然而，你可能从未用程序执行过这些任务。所以，现在就是机会。本章侧重于专门处理文件和文件夹的VBA函数和指令。通过使用这些函数，你将能够：

- 获得当前文件夹的名称（CurDir函数）
- 更改文件和文件夹名称（Name函数）
- 检查某文件或文件夹是否存在于某硬盘上（Dir函数）
- 获取某文件最后修改的时间和日期（FileDateTime函数）
- 获取文件大小（FileLen函数）
- 检查和更改文件属性（GetAttr和SetAttr函数）
- 更改缺省文件夹或者硬盘（ChDir和ChDrive语句）
- 创建和删除文件夹（MkDir和Rmdir语句）
- 复制和删除文件或文件夹（FileCopy和Kill语句）

此外，本章也给你往三类文件写入或者读取数据的知识：连续的，随机的和二进制的文件。除了使用Excel应用软件界面之外，你将学习如何直接操作文件。在本章的最后，将给你介绍最新的操作文件和文件夹的方法，通过利用称为Windows Scripting Host (WSH)的工具来操作文件和文件夹。

操作文件和文件夹

本节将讨论多种操作文件和文件夹的函数。

1. 获取当前文件夹的名称（CurDir 函数）

当你使用文件时，经常会需要知道当前文件夹的名称，你使用CurDir函数轻易地获取该信息：

CurDir([drive])

Drive是一可选参数，如果你忽略它，VBA将使用当前驱动器（drive）。

CurDir函数返回一个文件路径作为Variant（变量）。如果要返回作为字符串（String）的路径的话，就得使用CurDir\$（这里的\$是字符串的类型声明字符）。让我们在立即窗口里做些练习，练习使用这些函数吧：

1. 打开一个新工作簿，并且切换到VB编辑器窗口
2. 激活立即窗口并敲入下述代码：

?CurDir

当你按下回车，VB就显示当前文件夹名称，例如：

C:

如果你有第二个硬盘（或者光驱）的话，你可以获取D盘上的当前文件夹，例如：

?CurDir("D:\")

如果你提供了一个并不存在的驱动字母的话，VB就将显示下述错误信息：“设备不可用”

3. 要储存当前驱动名称到变量myDrive，可以输入下述指令：

myDrive = Left(CurDir\$,1)

当你按下回车键时，VB将当前驱动器的字母储存到变量myDrive

敲入下述指令并回车，可以检查变量myDrive的内容：

?myDrive

你还可以将上面的指令改为如下：

myDrive = left(CurDir\$,1) & ":"

VB将返回驱动器字母，后面带有一个冒号。

2. 更改文件或文件夹名称（Name 函数）

使用函数Name可以重命名文件或者文件夹，例如：

Name old_pathname As new_pathname

Old_pathname是你想用重命名的文件或文件夹的名称和路径，New_pathname则明确文件或文件夹的新名称和位置。使用函数Name，你可以将一个文件从一个文件夹移动到另外一个文件夹，但是，你不可以移动文件夹。

请在立即窗口里试演该函数（用你文件的实际名称替换示例名称）。这里有些需要考虑的注意事项：

- 在New_pathname里的文件名称不要指向已经存在的文件

Name "C:\System.1st " As "C:\test.txt"

因为文件C:\test.txt已经存在于C盘，VB将显示错误信息：“文件已存在”，同样，如果你要重命名的文件不存在的话，就会出现“文件未找到”的错误信息。

- 如果New_pathname已经存在，并且和Old_pathname不同，函数Name必要时将文件移动到新文件夹并且更改它的名称。

Name "C:\System.1st " As "D:\test.txt"

因为文件test.txt在D盘的根目录下并不存在，VB将C:\System.1st移动到指定的驱动盘，然而，并不重命名该文件。（译者：本段与上面的内容似乎矛盾，而且未能试验成功，未知是原书失误与否。读者应仔细验证）

- 如果New_pathname和Old_pathname指向不同的目录，以及提供的文件名称相同，那么Name函数将指定的文件移到新地址，不用更改文件名。

Name "D:\test.txt " As "C:\DOS\test.txt"

上面的指令将test.txt移动到C盘下的DOS文件夹里。

技巧8-1 你不能重命名开启的文件

在重命名文件之前，你必须关闭该文件。文件名称里不能包含通配符“*”或者“?”。

3.检查文件或文件夹是否存在（Dir 函数）

Dir函数，返回文件或者文件夹名称，语法如下：

Dir[(pathname[, attributes])]

Dir函数的两个参数都是可选的，pathname是文件或文件夹名称，对于参数attributes，你可以下列常量或者数值之一：

表8-1 文件属性

常量	数值	属性名称
vbNormal	0	Normal 普通文件
vbHidden	2	Hidden 隐藏文件
vbSystem	4	System 系统文件
vbVolume	8	Volume label 卷标
vbDirectory	16	Directory or Folder 目录或文件夹

Dir函数常用来检查某个文件或文件夹是否存在，如果不存在，那么就返回空字符串（""）。我们到立即窗口里试验几个Dir函数的练习：

- 在立即窗口，输入下述指令：

```
?Dir("C:\", vbNormal)
```

你一旦按下回车键，VB就会返回该文件夹下的第一个文件名。普通文件（vbNormal）就是除隐藏，卷标，目录，文件夹或系统文件之外的任何文件。

要返回当前目录下的其它文件名称的话，就使用不带参数的Dir函数：

```
?Dir （并且回车）
```

- 在立即窗口里输入下列指令，并且在你回车时检查其结果：

```
mfile = Dir("C:\", vbHidden)
```

```
?mfile
```

```
mfile = Dir
```

```
?mfile
```

```
mfile = Dir
```

```
?mfile
```

- 在立即窗口输入下述指令：

```
If Dir("C:\stamp.bat") = "" Then Debug.Print "文件未找到。"
```

因为stamp.bat文件不在C盘上，所以VB就在立即窗口里写下文本信息“文件未找到。”

- 在立即窗口输入下述语句，可以检查某文件是否存在于某驱动盘上：

```
If Dir ("C:\Autoexec.bat") <> "" Then Debug.Print "该文件不在C盘上。"
```

函数Dir允许你在文件路径名中使用通配符——星号（*）代表多个字符，问号（?）代表单个字符：例如，要在WINDOWS文件夹中查找所有配置设置的文件，你可以查找所有的INI文件，如下：

```
?Dir("C:\WINNT\*.ini", vbNormal)
```

```
system.ini
```

```
?dir WIN.INI
?dir
WINFILE.INI
?dir control.ini
?dir EQUIP32.INI
?dir
sxpwin32.ini
```

下面显示的过程在立即窗口里写上了确定目录下的文件名称。函数LCase\$让文件名称显示为小写字母。

1. 打开一新工作簿，并保存为Chap08.xls
2. 切换到VB编辑器窗口并重命名VBA工程为FileMan
3. 插入新模块，重命名为DirFunction
4. 输入下述VBA过程：

```
Sub MyFiles()
Dim mfile As String
Dim mpath As String

mpath = InputBox("Enter pathname,e.g., C:\Excel")
If Right(mpath, 1) <> "\" Then mpath = mpath & "\"

mfile = Dir(mpath & "**.*")
If mfile <> "" Then Debug.Print "Files in the " & mpath _
    & "folder"
    Debug.Print LCase$(mfile)
    If mfile = "" Then
        MsgBox "No files found."
        Exit Sub
    End If
    Do While mfile <> ""
        mfile = Dir
        Debug.Print LCase$(mfile)
    Loop
End Sub
```

上面的过程myFiles向用户询问文件路径名。如果该路径结尾没有反斜杠，函数Right就会将反斜杠附加在路径名字符串上。接下来，VB在该确定的文件夹里搜索所有文件（*）。如果没有文件的话，就会有信息显示，如果文件存在，那么文件名就会被写入立即窗口。

5. 在同一个模块里输入另外一个过程：

```
Sub GetFiles()
Dim nfile As String
Dim nextRow As Integer 'next row index
nextRow = 1
With Worksheets("Sheet1").Range("A1")
nfile = Dir("C:\", vbNormal)
.Value = nfile
Do While nfile <> ""
nfile = Dir
.Offset(nextRow, 0).Value = nfile
nextRow = nextRow + 1
Loop
End With
End Sub
```

过程GetFiles获取C盘根目录下的所有文件名并且将每个文件名写入工作表。

4.获得文件修改的日期和时间（FileDateTime 函数）

如果你的过程需要知道某文件的最后修改的时间的话，可以使用函数FileDateTime：

FileDateTime(文件路径名)

文件路径名是个字符串，明确你要用的文件，并且需要包括驱动和文件夹的名称。该函数返回某文件的日期和时间印记。日期和时间的格式取决于视窗控制面板的原始设置。

我们在立即窗口里来练习使用该函数：

1. 在立即窗口里输入：

```
?FileDateTime("C:\config.sys")
```

回车后，VB返回下述格式的日期和时间

```
5/4/2001 10:52:00 AM
```

要分开获取日期和时间时，可以将函数FileDateTime作为函数DateValue或TimeValue的参数来使用。例如：

```
?DateValue(FileDateTime("C:\config.sys"))
```

```
?TimeValue(FileDateTime("C:\config.sys"))
```

2. 在立即窗口里将下述语句在一行输入：

```
If DateValue(FileDateTime("C:\config.sys"))< Date then Debug.Print "This file was not modified today."
```

Date函数返回当前系统日期，也是视窗控制面板的日期/时间对话框里设定的。

5. 获得文件大小（FileLen 函数）

如果你需要检查某文件是否能够存在某磁盘上，那么你应该按照下述方式使用FileLen函数：

FileLen(文件路径名)

FileLen函数一字节方式返回文件的大小。如果该文件已打开，那么VB将返回该文件最后一个保存时的大小。

假设你想要获取Windows目录下进行配置设置的所有文件的总大小：

1. 在当前工程里插入新模块，并重命名为FileLenFunction

2. 在代码窗口输入过程TotalBytesIni:

```
Sub TotalBytesIni()
```

```
    Dim iniFile As String
```

```
    Dim allBytes As Long
```

```
    iniFile = Dir("C:\WINDOWS\*.ini")
```

```
    allBytes = 0
```

```
    Do While iniFile <> ""
```

```
        allBytes = allBytes + FileLen("C:\WINDOWS\" & iniFile)
```

```
        iniFile = Dir
```

```
    Loop
```

```
    Debug.Print "Total bytes: " & allBytes
```

```
End Sub
```

6. 返回和设置文件属性（GetAttr 函数和 SetAttr 函数）

文件和文件夹具有类似“只读”，“隐藏”，“系统”和“档案”的特点。这些特点就是属性。可以使用GetAttr函数来获得文件或文件夹的属性。该函数的唯一参数就是文件或文件夹路径名：

GetAttr(文件路径名)

上面的函数返回一个整数，代表下面显示的常量中德一个或多个常量之和。

表8-2 文件和文件夹属性

常量	数值	属性名称
vbNormal	0	普通文件（没有设置其它属性）
VbReadOnly	1	不可修改的文件或文件夹
vbHidden	2	在普通设置下不可见的文件或文件夹
vbSystem	4	系统文件
vbDirectory	16	对象为一个目录
vbArchive	32	档案（在最后一次备份后，该文件被修改）

要知道某文件是否具有上述的属性，可以使用AND运算符来比较GetAttr函数的结果和常量数值。如果函数返回一个非零数值，那么该文件或文件夹具有和你测试的属性一样的属性。

C:\MsDos.sys的属性是什么呢？你可以在立即窗口里快速获得：

```
?getattr("C:\MsDos.sys") AND vbReadOnly
1
?getattr("C:\MsDos.sys") AND vbHidden
2
?getattr("C:\MsDos.sys") AND vbSystem
4
?getattr("C:\MsDos.sys") AND vbArchive
32
```

现在，我们来将这些信息一起放在一个过程里：

1. 插入新模块，并重命名为GetAttrFunction
2. 输入下述过程GetAttributes:

```
Sub GetAttributes()
    Dim attr As Integer
    Dim msg As String
    attr = GetAttr("C:\MSDOS.SYS")
    msg = ""

    If attr AND vbReadOnly Then msg = msg & "Read-Only (R)"
    If attr AND vbHidden Then msg = msg & Chr(10) & "Hidden (H)"
    If attr AND vbSystem Then msg = msg & Chr(10) & "System (S)"
    If attr AND vbArchive Then msg = msg & Chr(10) & "Archive (A)"
    MsgBox msg, "MSDOS.SYS"
End Sub
```

3. 当你运行上面的过程时，你将看到如图8-1的信息框



图8-1 使用GetAttr函数可以获得任何文件的属性

GetAttr函数的相反函数是SetAttr函数，它允许你设置一个文件或文件夹的属性。语法如下：

SetAttr 文件路径名, 属性

文件路径名确定你要设置的文件或文件夹，第二个参数，属性，是一个或多个你要设置的属性常量。参见表8-1本章前面介绍的常量清单。

假设你有一个叫做“C:\stamps.txt”的文件，并且要设置两个属性，“只读”和“隐藏”。在立即窗口里输入下述指令来设置文件属性（可以找个你硬盘上存在的文件来试验）：

```
SetAttr "C:\stamps.txt", vbReadOnly + vbHidden
```

技巧8-2 调用SetAttr语句

你不能给打开的文件设置属性，在使用SetAttr函数之前，你必须关闭该文件。

7.更改缺省文件夹或驱动器（ChDir 语句和 ChDrive 语句）

使用ChDir语句，你可以轻易更改缺省文件夹，例如：

ChDir Path

在上面的语句中，Path是新的缺省文件夹名称。Path可以包含驱动器名称。如果Path没有包括驱动名称，那么缺省文件夹将

会更改为当前驱动。当前驱动不变。

假设缺省文件夹为“C:\DOS”，语句：

```
ChDir "D:\MyFiles"
```

将缺省文件夹更改为“D:\MyFiles”，然而，当前驱动仍然是C盘。

要更改当前驱动的话，你就应该使用ChDrive语句，按如下格式：

```
ChDrive 驱动
```

“驱动”是你将要设置的新的缺省驱动名称。例如，在立即窗口里输入下述指令将缺省驱动设置为D驱或者E驱：

```
ChDrive "D"
```

或者

```
ChDrive "E"
```

如果你指向一个并不存在的驱动，你就会看到一个信息框“设备不可用”

8.创建和删除文件夹（Mkdir 语句和 Rmdir 语句）

依照下面的Mkdir语句语法，你可以创建一个新文件夹：

```
Mkdir Path
```

Path明确你要创建的新文件夹名称。如果你没有写驱动器的名称的话，VB就将在当前的驱动上创建新文件夹。现在，我们来看几个例子：

1. 在立即窗口里输入指令，在C盘上创建一个叫“Mail”的文件夹：

```
Mkdir "C:\Mail"
```

2. 将缺省文件夹更改为“C:\Mail”：

```
ChDir "C:\Mail"
```

3. 获取当前文件夹名称：

```
?CurDir
```

使用Rmdir函数来删除不需要的文件夹。该函数的语法如下：

```
Rmdir Path
```

Path明确你要删除的文件夹名称。Path可以包含驱动名称，如果你忽略了驱动名称，那么VB就会试图删除当前驱动下的相同名称的文件夹，如果存在的话；否则，VB将显示错误信息：“路径未找到”

4. 删除刚才创建的文件夹C:\Mail：

```
Rmdir "C:\Mail"
```

技巧8-3 Rmdir移除空文件夹

如果文件夹里有东西，你不可以删除它（使用Rmdir）。你应该先用Kill语句删除这些文件（在本章后面讨论）

9.复制文件（FileCopy 语句）

使用FileCopy语句，可以在文件夹之间复制文件：

FileCopy 来源，目的地

该语句的第一个参数是文件来源，明确你要复制的文件名称，该名称可以包含驱动名称。第二个参数是复制的目的地，可以包括驱动和文件夹的地址。两个参数都是必须的。假设你要将用户确定的一个文件复制到一个叫做“C:\Abort”的文件夹，下面的过程示范如何完成它：

```
Sub CopyToAbort()
```

```
    Dim folder As String
```

```
    Dim source As String
```

```
    Dim dest As String
```

```
    Dim msg1 As String
```

```
    Dim msg2 As String
```

```
    Dim p As Integer
```

```
    Dim s As Integer
```

```
    Dim i As Long
```

```
    On Error GoTo ErrorHandler
```

```

folder = "C:\Abort"
msg1 = "The selected file is already in this folder."
msg2 = "was copied to"
p = 1
i = 1

' get the name of the file from the user 从用户处获取文件名称
source = Application.GetOpenFilename
' don't do anything if cancelled 如果取消则不进行任何操作
If source = "False" Then Exit Sub
' get the total number of backslash characters "\" in the source 获取文件来源字符串中的反斜杠数
' variable's contents
Do Until p = 0
    p = InStr(i, source, "\", 1)
    If p = 0 Then Exit Do
    s = p
    i = p + 1
Loop
' create the destination file name 创建目的文件名称
dest = folder & Mid(source, s, Len(source))
' create a new folder with this name 创建同名文件夹
MkDir folder
' check if the specified file already exists in the 检查该文件是否在目的地已经存在
' destination folder
If Dir(dest) <> "" Then
    MsgBox msg1
Else
    ' copy the selected file to the C:\Abort folder 复制所选文件到文件夹“C:\Abort”
    FileCopy source, dest
    MsgBox source & " " & msg2 & " " & dest
End If
Exit Sub

```

```

ErrorHandler:
If Err = "75" Then
    Resume Next
End If
If Err = "70" Then
    MsgBox "You can't copy an open file."
    Exit Sub
End If
End Sub

```

过程CopyToAbort使用了Excel应用程序的方法GetOpenFilename从用户那里获取文件名称。该方法导致弹出内置的打开对话框。使用该对话框，你可以在任何驱动的任何文件夹里选择任何文件。如果用户取消了，VB就返回值“False”并且程序结束。如果用户选取了某个文件并且点击了打开，那么该选中的文件就会赋值到变量source。因为复制的目的，你只需要文件名称（而不需路径名），所以Do...Until循环用来找到最后一个反斜杠（“\”）在变量source里的位置。

接下来，VB给FileCopy语句的第二个参数准备了一个字母字符串，并且将其赋值到变量dest。该变量储存的字符串是目标文件夹（C:\Abort）和用户指定的文件名前面加反斜杠连接起来的。函数MkDir创建了一个叫C:\Abort的文件夹，如果它不存在于C盘上的话。如果这样的文件夹已经存在的话，那么VB就需要去处理错误75了。这个错误会被在程序后面的错误处理代码捕获。注意，错误处理器是一代码片断，它用ErrorHandler带冒号标志。

当VB遇到Resume Next语句时，就会继续执行过程里面导致错误的代码行下面的代码。这意味着语句MkDir folder不会被执行。在这之后，程序将检查被选择的文件是否已经存在于目的文件夹。如果文件在那，那么用户将收到储存于变量msg1里面的信息；如果文件不存在于目的文件夹并且该文件当前没有打开的话，VB就会将文件复制到指定的文件夹，并且用相应的信息通知用户。如果该文件被打开了，VB将遇到运行时间错误70，并且因此而运行ErrorHandler里面的相应指令。

1. 在一名为FileCopyStatement的信魔窟里输入过程CopyToAbort
2. 运行该程序几次，从不同的文件夹里选择文件
3. 试着复制该程序之前复制过的文件到文件夹C:\Abort
4. 打开某个文件，并且在其开着的情况下试图用过程CopyToAbort来复制它
5. 运行本章前面准备的过程MyFiles，在立即窗口里列出文件夹C:\Abort里面的内容

注意，不要删除文件夹C:\Abort和你复制的文件，你将在下一节里面使用一个叫RemoveMe的VBA过程来同时删除文件和文件夹。

10.删除文件（Kill 语句）

你已经从前面的章节里知道了不能删除含有文件的文件夹，要从文件夹里面删除文件的话，可以使用下面的Kill语句：

Kill 文件路径名

文件路径名明确一个或多个你要删除的文件的名称，随你意，也可以将驱动器和文件夹名称包括在里面。你可以在文件路径名参数里使用通配符（*或?）来确保快速删除文件。你不能删除开启的文件。

如果你是跟着前面的练习一步一步过来的，那么你的硬盘上应该有了文件夹C:\Abort和好几个文件了里面了。在下面的练习里，你将首先删除文件夹Abort里面的所有文件，然后再删除文件夹本身：

1. 在当前工程里插入新模块，并重命名为KillStatement
2. 在过程RemoveMe里输入代码，如下所示：

```
Sub RemoveMe()
    Dim folder As String
    Dim myFile As String

    'assign the name of folder to the folder variable
    'notice the ending backslash "\"
    folder = "C:\Abort\"
    myFile = Dir(folder, vbNormal)
    Do While myFile <> ""
        Kill folder & myFile
        myFile = Dir
    Loop
    Rmdir folder
End Sub
```

3. 运行过程RemoveMe，当程序运行结束，点击Windows文件浏览器看看该文件夹是否已经被删除了。

11.从文件读取和写入数据（Input/Output）

你已经从前面的章节里知道了如何使用VBA打开一个电子表格，例如指令：

```
Application.Workbooks.Open Filename:="C:\Excel\Report.xls"
```

打开位于文件夹C:\Excel里面的文件Report.xls。除了使用专门的应用程序打开文件之外，你如果也想要创建VBA过程能够打开其它类型的文件并使用它们的内容的话，你就应该学习一些关于被称为低级别的文件I/O（input/output）。接下来关于顺序，随机和二进制文件的章节将会带你直接接触你的数据。

12.文件访问类型

计算机使用的文件类型有三种：

- **顺序访问文件**是指按储存相同的顺序找回数据的文件。例如以CSV格式（逗号分割文本），TXT格式（以Tab键分割的文本）或者PRN格式（以空格分隔的文本）储存的文件。顺序文件访问经常用来写文本文件，例如错误日志，参数设定和报告。顺序文件有下列模式：Input, Output 和 Append。模式决定了文件打开后你如何使用它。
- **随机访问文件**是文本文件，它的数据以同等长度储存并在一个以逗号分割的区域了。随机访问文件只有一个模式——Random
- **二进制访问文件**是图形文件和其它非文本文件。二进制文件只能够在Binary模式下访问。

13.使用顺序文件

你的电脑硬盘上有成百上千的顺序文件。参数文件，错误日志，HTML文件以及所有类型的无格式文本文件都是顺序文件。这些文件以字母顺序在硬盘上储存。新文本行的开始以两个专门的字符表示，一个叫做carriage return（回车），另一个叫line feed（换行）。当你使用顺序文件是，你从文件的开头始，一个字符一个字符的向前移动，一行接一行，直到文件的结尾。顺序文件容易打开和操作，任何文本编辑器都可以。

技巧8-4 什么是顺序文件？

顺序文件就是访问它里面的记录时必须按它占据的顺序进行的文件，这意味着在你想访问第三个记录之前，你必须先访问第一个记录，接着是第二个记录。

技巧8-5 使用Open语句打开文件

当你使用顺序访问来打开一个文件时，该文件必须是已经存在的。

14.读取储存于顺序文件里的数据

我们来用一个已经在你电脑上的顺序文件并且在Excel VB编辑器窗口直接使用VBA来读取它的内容。要从一个文件读取数据，你就必须先使用Open语句打开该文件。这是它的语法：

Open pathname **For** mode [Access access][lock] **As** [#]filenumber [Len=reclength]

Open语句有三个必须的参数，它们是pathname, mode, 和 filenumber。上面的语法里，这三个参数前面都有用粗体显示的关键字。

- Pathname是你打开的文件名称
- Pathname可以包括驱动器和文件夹名称
- Mode是个决定文件如何打开的关键字。顺序文件可以以下列模式之一来打开：Input, Output 或Append。使用Input读文件，Output写文件，将覆盖任何存在的文件，以及Append来写入文件，同时加上任何已经存在的信息。
- Access是决定文件读写的关键字，Access可以是：Shared（共享），Lock Read（锁定读），Lock Write（锁定写）或Lock Read Write（锁定读写）。
- Lock决定了哪些文件的操作是允许其它过程进行的。例如，如果某文件是在网络环境下打开的，“锁定”决定了其他人如何访问它。下述锁定关键字是可以用的：Read, Write 或者 Read Write。
- Filenumber是从1到511的数字，该数字用来指向顺序操作中的文件。通过使用VB内置函数FreeFile，你可以获得一个唯一的文件号码。
- Open语句里的最后一个成员reclength明确顺序文件里总字符数，或者是随机文件里记录大小。

考虑一下前面的例子，为了读取数据，要打开C:\Autoexec.bat或者其它顺序文件，你应该使用下面的指令：

Open "C:\Autoexec.bat" For Input As #1

如果某文件已经打开输入了，那么从它读取数据。在打开一格顺序文件后，你就可以使用下面的语句读取它的内容：Line Input #或者 Input # 或者使用Input 函数。

15.逐行读取文件

使用下面的语句来逐行读取Autoexec.bat或者其它任何顺序文件里的内容：

Line Input #filenumber, variableName

#filenumber是用Open语句打开文件时使用的数字，variableName是个String或者Variant变量，用来储存读取的行。

Line Input #语句仅读取一开启顺序文件里的一行并且储存在一变量里。记住，Line Input # 语句一次读取顺序文件里的一个字符，直到它遇到回车字符（Chr(13)）或者回车-换行字符（Chr(13) & Chr(10)）。这些字符（回车，换行）在读取过程中返回的文本里是会忽略掉的。

接下来的过程ReadMe示范如何使用Open和Line Input #语句逐行读取Autoexec.bat文件的内容。试试用同样的方法来读取其它顺序文件。

1. 在当前工程里面插入新模块并重命名为SeqFiles

2. 输入下列过程ReadMe：

Sub ReadMe()

Dim rLine As String

Dim i As Integer ' line number


```

i = 1
Open "C:\Autoexec.bat" For Input As #1

' stay inside the loop until the end of file is reached

Do While Not EOF(1)
    Line Input #1, rLine

    MsgBox "Line " & i & " in Autoexec.bat reads: " _
        & Chr(13) & Chr(13) & rLine

    i = i + 1
Loop

MsgBox i & " lines were read."
Close #1
End Sub

```

3. 按下F8，逐句运行该过程

为了读取内容，过程ReadMe将文件Autoexec.bat在模式Input里作为文件号码1打开。Do...While循环告诉VB一直执行循环里面的语句，直到到达文件结尾。文件的结尾由函数EOF的结果决定。

EOF函数当下个要读取的字符已经过了文件结尾时，返回逻辑值True。注意，EOF要求一个参数——你要检查的打开了的文件号码，是前面Open语句使用的同一个数字。使用EOF函数来确保VB不会超出文件结尾处。

Line Input # 语句将每行内容储存于变量rLine里，然后，信息框显示行号和它的内容。之后如果函数EOF的结果还是为假（还未到达文件结尾处）的话，VBA给行计数器增加1，并且开始读取下一行。当函数EOF结果为真是，VB就会退出循环。在VBA结束前，还会再运行两条语句，显示读取行的总数，以及关闭该打开的文件。

16.从顺序文件中读取字符

假设你的程序需要检查文件Autoexec.bat里出现了多少个冒号，你可以使用函数Input来返回特定的字符数，而不必读取整行。接下来，If语句用来比较获取的字符和你寻找的字符。在写过程之前，我们来看看函数Input的语法：

Input(number, [#]filenumber)

Input函数的两个参数都是必须的，number明确你要读取的字符数，而filenumber是Open语句用来打开文件的同一个数字。

Input函数返回所有读取的字符，包括逗号，回车，文件结束字符，引号和前导空格。

1. 在SeqFile模块里输入下述过程Colons:

```

Sub Colons()
    Dim counter As Integer
    Dim char As String

    counter = 0
    Open "C:\Autoexec.bat" For Input As #1

    Do While Not EOF(1)
        char = Input(1, #1)
        If char = ":" Then

            counter = counter + 1
        End If
    Loop

    If counter <> 0 Then
        MsgBox "Characters found: " & counter
    Else
        MsgBox "The specified character has not been found."
    End If
    Close #1
End Sub

```

End Sub

2. 逐句执行该过程

3. 将冒号换成其它你想寻找的字符并且重新执行该程序。**Input**函数允许你返回顺序文件的任何字符。如果你使用VB函数**LOF**作为**Input**函数的第一个参数时，你将能够快速读取顺序文件里的内容，而不需要在整个文件上循环。**LOF**函数返回一个文件上的字节数。每个字节对应了文本文件里的一个字符。过程**ReadAll**将文件**System.ini**的内容读取到立即窗口里：

Sub ReadAll()

```
Dim all As String
Open "C:\WINNT\System.ini.bat" For Input As #1
all = Input(LOF(1), #1)
Debug.Print all
Close #1
```

End Sub

除了将文件内容打印到立即窗口之外，你还可以将其读取到一个文本框并且放置到工作表中去（见图8-2）：

Sub WriteToTextBox()

```
Dim mysheet As Worksheet
Set mysheet = ActiveWorkbook.Worksheets(1)
On Error GoTo CloseFile
Open "C:\WINNT\System.ini" For Input As #1
mysheet.Shapes(1).Select
Selection.Characters.Text = Input(LOF(1), #1)
CloseFile:
Close #1
```

End Sub

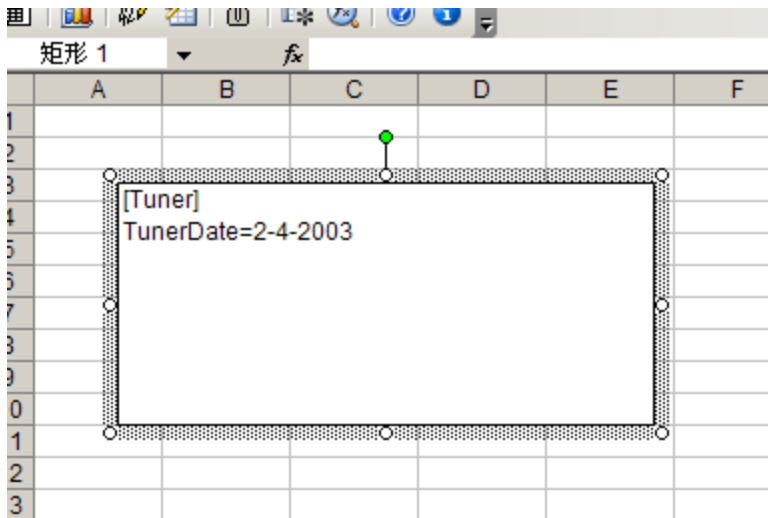


图8-2 文件TDate.ini（译者：原文为System.ini）的内容显示在工作表中的文本框里

在你运行上面的程序之前，你得在工作表里画一个文本框。注意，**On Error GoTo CloseFile**语句激活错误捕捉，如果错误在程序的执行过程中发生了，就会立即跳到**CloseFile**标签处，**Close #1**语句无论有无错误发生都会被执行。

17. 读取分隔文本文件

在某些文本文件中（文件通常保存为**CSV**、**TXT**或**PRN**格式）输入在每行的数据由逗号，**Tab**或者空格分隔。这种类型的文件用**Input #**语句可以比前面介绍的**Line Input #**语句读取更快些。**Input #**语句允许你从一个打开的文件中读取数据到好几个变量，该函数如下所示：

Input #filename, variablelist

Filename是用**Open**语句打开文件时的同一个号码，**variablelist**是一个以逗号分开的变量清单，用来储存读取的数据。你不能使用数组或对象变量，然而，你可以使用用户定义的变量（这种变量将在本章后面介绍）。

下面的例子是一个用逗号分隔数据的顺序文件：

Smith,John,15

Malloney,Joanne,28

Ikatama,Robert,15

要读取该格式的文本的话，你必须给每个数据明确一个变量：姓，名和年龄。

1. 打开一个新工作簿，并且输入下面的数据：

	A	B	C
1	Smith	John	15
2	Malloney	Joanne	28
3	Ikatama	Robert	15

2. 将文件保存为CSV格式在C:\Winners。Excel将显示信息告诉你，该格式文件不支持含有多个工作表的工作簿。点击确定，只保存当前工作表。

3. 输入下面的过程Winners:

```
Sub Winners()
    Dim lName As String, fName As String, age As Integer
    Open "C:\Winners.csv" For Input As #1
    Do While Not EOF(1)
        Input #1, lName, fName, age
        MsgBox lName & ", " & fName & ", " & age
    Loop
    Close #1
End Sub
```

4. 在运行过程Winners之前，你要确保该文件在指定的路径下，或者在程序里指定文件Winners.csv的正确位置。

上面的程序打开文件Winners.csv读取数据，并且建立了一个Do...While循环，在整个文件里运行直到文件的结尾。Input #1语句用来将每行的内容读取到三个变量：lName, fName和age，然后，一个信息框将这三个变量的内容显示出来。程序最后关闭文件Winners.csv。

18.往顺序文件里写数据

当你要往一个顺序文件里写入数据时，你应该以Append或者Output模式打开该文件。这些模式的解释如下：

- **Append** 允许在一个现存文件的结尾处添加数据。例如，如果你以Append模式打开Readme.txt文件，并且将文本“谢谢你阅读本文件”加到该文件，VB不会删除或者以任何方式改变该文件中已经存在的文本，但是，会在文件的结尾处加上新的文本。
- **Output** 当你以Output模式打开一个文件时，VB将会将文件里的现存的数据删除，而且，如果该文件并不存在的话，就会创建一个全新的文件。例如，如果你以Output模式打开文件Readme.txt，并且试图往里面写数据的话，那么以前储存在该文件里的文本就会被删除掉。如果你在写入数据之前没有备份该文件的话，那该失误的代价将会是非常大的。如果你想要用新数据取代整个内容的话，就应该以Output模式打开该已存在的文件。

这里有些例子，什么时候应该用Append，什么时候用Output:

- 要在文件C:\Readme.txt后面添加新文本，按下面以Append模式打开该文本：

```
Open "C:\Readme.txt" For Append As #1
```

- 要在一个叫C:\Result.txt的全新文件里输入一些文本，那么以Output模式打开该文件：

```
Open "C:\Result.txt" For Output As #1
```

- 要取代现存文件C:\Winners.csv的内容，首先将原始文件备份一份，然后将原始文件以Output模式打开：

```
FileCopy "C:\Winners.csv", "C:\Winners.old"
```

```
Open "C:\Winners.csv" For Output As #1
```

技巧8-6 不可同时读写

顺序文件必须分别打开来执行读和写的操作，你不可以同时执行这些操作。例如，在一个文件已经打开并且写入数据后，该文件必须先关闭，之后才能再打开来读取数据。

技巧8-7 顺序文件的优势和劣势

尽管顺序容易创建和使用，并且不回浪费空间，但是它们也有很多不好的地方。例如，要是不读一大部分文件内容的话，你是很难找到某个特定项目的。同时，文件的个别项目不容易改变或删除——你必须重新写入整个文件。在有，在技巧8-6例说的，顺序文件必须分开进行读和写的操作。

19.使用 Write # 和 Print # 语句

既然打开一文本文件来写入数据的两种方法（Append或Output）你都已经知道了，那么是时候学习Write #和Print #语句了，它们让你将数据发送到文件。当你使用Input #语句从一个顺序文件读取数据的时候，通常可以使用Write #语句往该文件写数据，如下所示：

Write #filename, [outputlist]

Filename明确你正使用的文件的号码，它是Write #语句的唯一必须的参数。Outputlist是你写入的文本。Outputlist可以是你写入的单个文本字符，也可以是包含数据的变量清单。如果你只明确了文件号码，VB就会在打开的文件里写入一个空行。我们准备一个文本文件，里面是三个人的名，姓，生日和兄弟姐妹的数目，演示数据是如何写入文件的：

1. 在当前模块里输入过程DataEntry：

```
Sub DataEntry()
    Dim lname As String
    Dim fname As String
    Dim birthdate As Date
    Dim s As Integer

    Open "C:\My Documents\Friends.txt" For Output As #1
    lname = "Smith"
    fname = "Gregory"
    birthdate = #1/2/63#
    s = 3
    Write #1, lname, fname, birthdate, s

    lname = "Conlin"
    fname = "Janice"
    birthdate = #5/12/48#
    s = 1
    Write #1, lname, fname, birthdate, s

    lname = "Kaufman"
    fname = "Steven"
    birthdate = #4/7/57#
    s = 0
    Write #1, lname, fname, birthdate, s
    Close #1
End Sub
```

上面的过程打开文件C:\My Documents\Friends.txt来写入数据。因为该文件还不存在于你的硬盘上，所以VB就创建了一个全新的文件并写入三个记录。写入文件的数据储存在变量上。注意，这些字符串由双引号分隔，而生日则有井号包围起来了。当你使用Windows记事本打开文件Friends.txt是，你将看到下述输入：

```
"Smith","Gregory",#1963-01-02#,3
"Conlin","Janice",#1948-05-12#,1
"Kaufman","Steven",#1957-04-07#,0
```

注意，Write #语句自动在每个数据之间插入逗号并且将行结束字符(Chr(13) & Chr(10))放在每行文本的后面，以至于每行新记录都从新的行开始。在上面的例子里面，每行文本显示一条记录——每条记录以姓开始，以同胞数目结束。

如果你想要将数据显示在一列中，而不是用逗号分隔数据，那么就使用Print #语句。例如，如果将上面的程序DataEntry里的Write #语句用Print #语句代替的话，那么VB就会按下面的方式写入数据：

```
Smith      Gregory    1/2/63      3
Conlin     Janice     5/12/48    1
Kaufman    Steven     4/7/57      0
```

尽管Print #语句和Write #语句的语法一样，但是，Print #以一个准备打印的格式将数据写入顺序文件。清单里的变量可能用

分号或者空格分隔。要打印多个空格的话，你就应该使用Spc(n)指令，这里n是空格数。类似地，要将数据输入到第五列的话，你就应该使用指令Tab(5)。

我们来看一些格式例子：

- 使用带逗号的Write #语句，往文件里输入空行

```
Write #1,
```

- 在第五列输入文本“fruits”

```
Write #1, Tab(5); "fruits"
```

- 用五个空格分隔开单词“fruits”和“vegetables”

```
Write #1, "fruits"; Spc(5); "vegetables"
```

20.操作随机文件

当某文件包含结构数据时，就以随机模式打开它。以随机模式打开文件让你能够：

- 同时读写
- 快速访问某特别记录

在随机文件里，所有记录都是等长度的，并且每条记录都有相同数目的固定大小区域。记录或者区域的长度必须在文件写入数据之前就确定。如果写入某区域的字符串长度小于该区域的大小，那么VB就会自动在该字符串后面加空格来填充区域。如果写入的文本比区域长度长的话，超出的字符就不会被写入。

要知道如果操作随机文件，你现在就需要创建一个小数据库用作外语学习。该数据库将包含由两个区域组成的记录，储存英语词组和其外语等同语。

技巧8-8 随机文件是什么？

随机文件是储存的记录可以随机访问的文件，这意味着随机文件里的任何记录都可以读取，而不必读取它之前的每条记录。

21.创建用户定义的数据类型

除了第三章里介绍的内置数据类型之外，VB允许你在模块的上面使用Type...End Type语句定义一个非标准的数据类型。该非标准数据类型也经常成为用户自定义的数据类型。用户自定义数据类型可以包括各种数据类型（字符串，整型，日期等等）的内容。当你使用随机访问的文件是，你经常要创建一用户定义的变量，因为，该变量使你可以轻易地访问个别记录。

1. 在当前工程里插入新模块并且重命名为RandomFiles
2. 在模块上面，紧接着Option Explicit语句下面，输入下述类型定义：

```
Option Explicit
```

```
' define a user-defined type
```

```
Type Dictionary
```

```
    en As String * 16 ' English word up to 16 characters
```

```
    sp As String * 20 ' Spanish word up to 20 characters
```

```
End Type
```

用户定义的名Dictionary的类型包括两个声明为String（字符串）的项目，并且有特定的大小。成员en可以接受最多16个字符，第二个项目sp的大小不能超过20个字符。如果你将这两个成员的长度加起来，那么记录长度将为36（16+20）。如果模块已经有了Option Explicit语句的话，你就不必再输入它了。

3. 输入下面的过程EnglishToSpanish

```
Sub EnglishToSpanish()
```

```
    Dim d As Dictionary
```

```
    Dim RecNr As Long
```

```
    Dim choice As String
```

```
    Dim totalRec As Long
```

```
    RecNr = 1
```

```
    'open the file for random access 打开文件作随机访问
```

```
    Open "Translate.txt" For Random As #1 Len = Len(d)
```


Do

```
' get the English word 活动英文词语
choice = InputBox("Enter an English word", "ENGLISH")
d.en = choice
```

```
' exit the loop if cancelled 如取消则退出循环
```

```
If choice = "" Then Exit Do
choice = InputBox("Enter the Spanish equivalent for " & _
& d.en, "SPANISH EQUIVALENT " & d.en)
If choice = "" Then Exit Do
d.sp = choice
```

```
' write to the record 写入记录
```

```
Put #1, RecNr, d
' increase record counter 增加记录计数器
recNr = recNr + 1
```

```
Loop Until choice = "" 'ask for words until Cancel 询问词语直到取消
```

```
totalRec = LOF(1) / Len(d)
MsgBox "This file contains " & totalRec & " record(s)."
```

```
' close the file
Close #1
```

End Sub

过程EnglishToSpanish开始时，声明四个变量，变量d声明为用户定义的类型Dictionary。该类型在前面就用Type语句声明了（见第二步）。在给变量RecNr赋予了初始值之后，VB打开文件Translate.txt，并且将其作为文件编码1随机访问。指令Len(d)告诉VB每条记录的大小为36字符。接下来VB执行Do...Until循环里面的语句，直到你取消。循环里的第一条语句提示你输入一个英语单词，并且将其赋予变量choice，然后该值被传递给用户定义d的第一个成员（d.en）。

一旦你停止输入数据，VB就会退出Do循环，并且执行程序里的最后的语句计算和显示文件里的记录总数。最后一条语句将文件关闭。如果你输入了英文词语并点击确定，那么下个对话框就会提示你输入外语等同语。当然，如果你决定现在就退出的话，VB就会跳出循环并且继续剩下的语句。如果一切正常，你输入了外语翻译，那么VB就会将它赋予变量choice并且传递给用户自定义变量d的第二个成员（d.sp），接下来，VB使用下述语句将整条记录写入到文件里：

```
Put #1, recNr, d
```

写入第一条记录后，VB会给记录计数器增加1，然后重复循环里的语句。过程EnglishToSpanish允许你在你的字典里输入任意多条记录。当你退出提出词语时，过程使用LOF和Len函数来计算文件里的总记录数。VB在显示信息后关闭该文本文件（Translate.txt）。

创建随机文件仅仅是个开始，接下来，过程VocabularDrill示范如何使用一个开启的随机文件的记录。这里，你将学习让你快速找到你文件中适合的数据的语句。

技巧8-9 理解Type语句

Type命令允许你创建一个自定义组，包括混合的变量类型，称为“用户自定义数据类型”。Type语句通常用于随机文件，将信息作为区域储存为固定大小的记录。我们可以将随机文件用的区域使用Type语句集中成为一个用户自定义，而不必为每个区域都声明一个变量。例如，如下所示定义一个包含三个区域的记录：

```
Type MyRecord
country As String * 20
city As String * 14
rank As Integer
End Type
```

一旦定义了类型，你必须给使用这种类型的变量名称：

```
Dim myInfo As MyRecord
```

使用变量名称后面加上句号和内部变量可以访问内部变量（country, city, rank），例如，要明确城市，输入：

```
MyInfo.city = "Warsaw"
```

4. 在过程EnglishToSpanish下面输入下面显示的过程VocabularDrill，代码的介绍在其后面。


```

Sub VocabularyDrill()
    Dim d As Dictionary
    Dim totalRec As Long
    Dim recNr As Long
    Dim randomNr As Long
    Dim question As String
    Dim answer As String

    ' open a random access file 打开随机访问文件
    Open "Translate.txt" For Random As #1 Len = Len(d)
    ' print the total number of bytes in this file 打印本文件的总字节数
    Debug.Print "There are " & LOF(1) & " bytes in this file."
    ' find out and print out the total number of records 找到并且打印总记录数
    recNr = LOF(1) / Len(d)
    Debug.Print "Total number of records: " & recNr

    Do
        ' get a random record number 获取随机记录数
        randomNr = Int(recNr * Rnd) + 1
        Debug.Print randomNr
        ' find the random record 找到该随机记录
        Seek #1, randomNr
        ' read the record 读取记录
        Get #1, randomNr, d
        Debug.Print Trim(d.en); " "; Trim(d.sp)
        ' assign answer to a variable 将答案赋予变量
        answer = InputBox("What's the Spanish equivalent?", d.en)

        ' finish if cancelled 如取消则介绍
        If answer = "" Then Close #1: Exit Sub
        Debug.Print answer

        ' check if the answer is correct 检查答案是否正确
        If answer = Trim(d.sp) Then
            MsgBox "Congratulations!"
        Else
            MsgBox "Invalid Answer!!!"
        End If
        ' keep on asking questions, until Cancel is pressed 不断提问，直到按下取消

    Loop While answer <> ""
    ' close file 关闭文件
    Close #1
End Sub

```

声明变量之后，过程VocabularyDrill打开一个随机访问文件，并且告诉VB每个记录的长度Len = Len(d)，接下来，在立即窗口里打印文件的总字节数和总记录数。字节数是由语句LOF(1)返回的。

记录数是总字节数（LOF）除以一个记录的长度——Len(d)。接下来，VB执行循环里的语句直到按下Esc键或者取消按钮。循环里的第一条语句将函数Rnd的结果赋予变量randomNr。接下来的语句将这个数字写入立即窗口，指令Seek #1, randomNr在开启的文件中移动光标到变量randomNr明确的记录处，在下来的指令读取找到的记录内容。要在打开的随机访问文件中读取数据，你必须使用Get语句。指令：

Get #1, randomNr, d

告诉VB要读取的记录号码（randomNr）以及要读取数据的变量（d）。随机文件中的第一个记录在位置1，第二个记录在位置2，依次类推。忽略记录号码会导致VB读取下一个记录。然后，用户定义的类型字典的两个成员都被写入了立即窗口。函数Trim(d.en)和Trim(d.sp)将读取的记录前后可能含有的空格。接下来，VB显示信息，提示用户提供显示单词的外语等同语。该单词赋予变量answer，如果你按下Esc而不是点击确定的话，VB就会关闭文件并且接受程序，否则，VB将打印你的答案到立

即窗口，并且通知你，你的答案是否正确。当你要退出单词训练程序，随时可以按下Esc或者点击对话框的取消按钮。如果你决定继续并且点击了确定按钮，程序就会产生一个新随机号码，并且会找回一个英语单词并且问你相对应的西班牙语。

你可以修改该过程VocabularyDrill，因此你可以将每个不正确翻译的单词写到工作表。同样，你也许想要将文件Translate.txt里的所有记录到写到工作表里，这样你就总可以知道你的字典内容。你可以在本书带的CD里找到这两个程序。



图8-3 在记事本里打开随机文件的内容

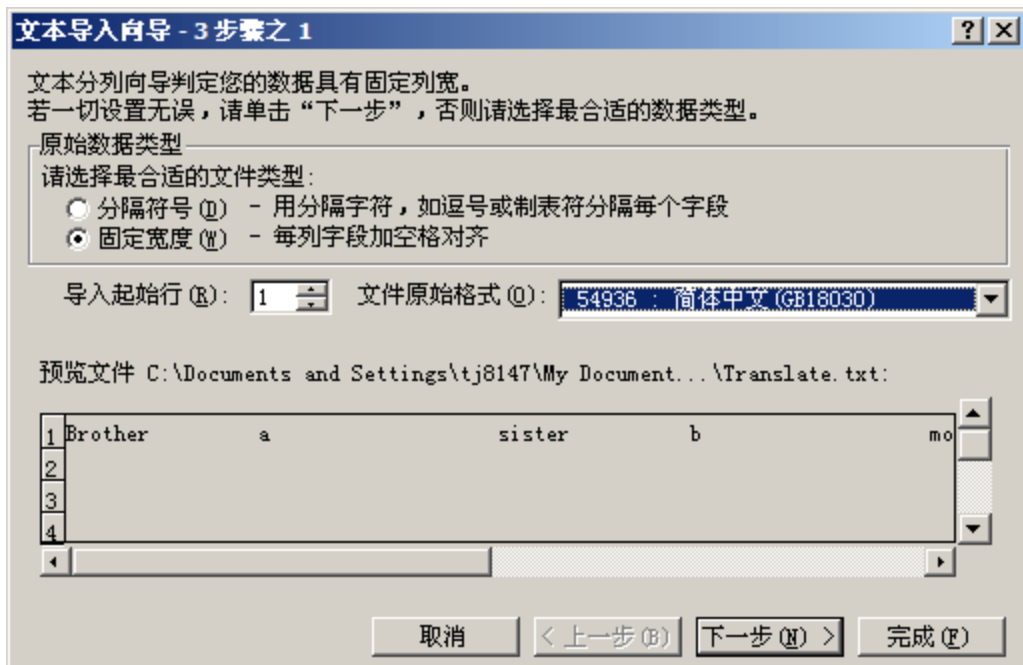


图8-4 试图用Microsoft Excel打开随机文件的内容。注意，Excel正确的认识了原始的数据类型——随机文件里的数据是固定宽度的。

技巧8-10 随机文件的优势与劣势

不想顺序文件，随机文件里的数据可以被很快地访问，而且，这些文件在往里面写信息和读信息期间不需要关闭文件。随机访问文件不必按顺序读写。因为它们的记录和区域都有固定的长度，不管储存的字符有多少，使用的字节数总是一样的，因此，如果有些区域是空的或者比声明的区域短时，就会浪费许多空间。

22.操作二进制文件

不象随机文件那样以固定长度储存数据，二进制文件是一些长度变化的记录的集合。例如，文件包含的第一个记录可以使10个字节，第二个记录可以只有5个字节，而第三个却可以使15个字节。这种储存数据的方法节省很多的硬盘空间，因为VB不需要在要储存的字符串后面加上多余的空格来确保它们有相同的长度（象往随机文件里写数据那样），在二进制文件里没有空间浪费。这就不奇怪二进制文件比前面所讲的两种文件占用的硬盘空间要少。正如随机文件，二进制文件也可以打开同时进行读和写的操作。然而，因为二进制文件里的记录是不同长度的，所以，这些文件的操作是更苦难一些的。要找回正确的数据的话，你就必须将每个区域和记录的大小信息储存起来。

你将使用下述四种语句来操作二进制文件：

- 使用Get语句来读取数据
- Put语句允许你往二进制文件输入新数据
- Loc语句返回所读的最后字节数（在随机文件里，Loc语句返回最后所读记录的数字）

- Seek语句将光标移动到文件中合适的位置。

为了快速掌握上面语句的使用，我们来打开立即窗口，并且将下面表格里左边的指令输入到立即窗口。本练习的目的是在一个叫做MyData.txt文件里输入你的姓和名，然后再找回你输入的数据。

立即窗口输入代码	解释
Open "MyData.txt" For Binary As #1	打开文件“MyData.txt”作为文件编号1来作二进制访问
MsgBox "Total bytes: " & LOF(1)	显示打开文件的字节数（该文件现在为空）
fname = "Julitta"	给变量fname赋值
ln = len(fname)	将储存于变量fname的字符串长度赋予变量ln
Put #1, , ln	将变量ln的内容放置在二进制文件的下一个字节
MsgBox "The last byte: " & LOC(1)	显示最后一个字节的位置
Put #1, , fname	在下一个位置放置变量fname的内容
lname = "Korol"	给变量lname赋值
ln = len(lname)	将储存于变量lname的字符串长度赋予变量ln
Put #1, , ln	将变量ln的数值输入到二进制文件的下一个字节
Put #1, , lname	在下一个位置放置变量lname的内容
MsgBox "The last byte: " & LOC(1)	显示最后一个字节的位置
Get #1, 1, entry1	读取第一个字节位置的数值并将其赋予变量entry1.
MsgBox entry1	显示变量entry1的内容
Get #1, , entry2	读取下一个位置的数值并将其赋予变量entry2.
MsgBox entry2	显示变量entry2的内容
Get #1, , entry3	读取下一个位置的数值并将其赋予变量entry3.
MsgBox entry3	显示变量entry3的内容
Get #1, , entry4	读取下一个位置的数值并将其赋予变量entry4.
MsgBox entry4	显示变量entry4的内容
Debug.Print entry1;entry2;entry3;entry4	将所有数据打印在立即窗口
7 Julitta 5 Korol	立即窗口里显示的上条指令的结果
Close #1	关闭文件

注意，上面的质量可以在CD里的过程EnterAndDisplay里找到。

当往二进制文件输入数据时，请遵循下述指导：

- 在往二进制文件写入字符串之前，将该字符串的长度赋予一个整型变量，通常可以使用下述代码块：

字符串长度 = Len(变量名称)

Put #1, , 字符串长度

Put #1, , 变量名称

- 当你从二进制文件读取数据时，先得读取该字符串长度，然后才是字符串内容。可以使用Get语句和String\$函数来实现：

Get #1, , 字符串长度

变量名称=String(字符串长度, "")

Get #1, , 变量名称

技巧8-11 二进制文件的优势与劣势

与顺序文件和随机文件相比，二进制文件是最小的，因为它使用变化长度的记录，可以保存硬盘空间。和随机访问文件一样，你可以同时读写二进制文件。二进制文件的一个最大的不好之处就是你必须要准确地知道你要找回或者要操作的数据是如何储存在文件里的。

23.操作文件和文件夹的时髦方法

在你的电脑上有一个被隐藏的宝贝叫做Windows Scripting Host（WSH视窗脚本主机），它允许你创建一些程序，可以控制视

窗操作系统和它的应用程序，以及从操作系统找回信息。WSH是一种ActiveX控件，可以在文件Wshom.ocx里找到。如果你正在使用Windows 95，98，NT5.0，2000，XP或者IE4，5或者6的话，该文件就会自动安装在Windows System32文件夹里面。

WSH是一种脚本语言。脚本语言是指可以自动运行的一套命令。可以使用Command Scripting Host (Cscript.exe)从命令提示，或者使用Windows Scripting Host (Wscript.exe)从视窗直接创建或者运行脚本。在本章接下来的本分，你将学习WSH如何和VBA结合工作的。

WSH有它自己的对象层次。使用CreatObject函数，你可以在VBA过程里引用WSH对象。在开始编写使用WSH的VBA过程之前，我们来看看你将能控制的一些对象。

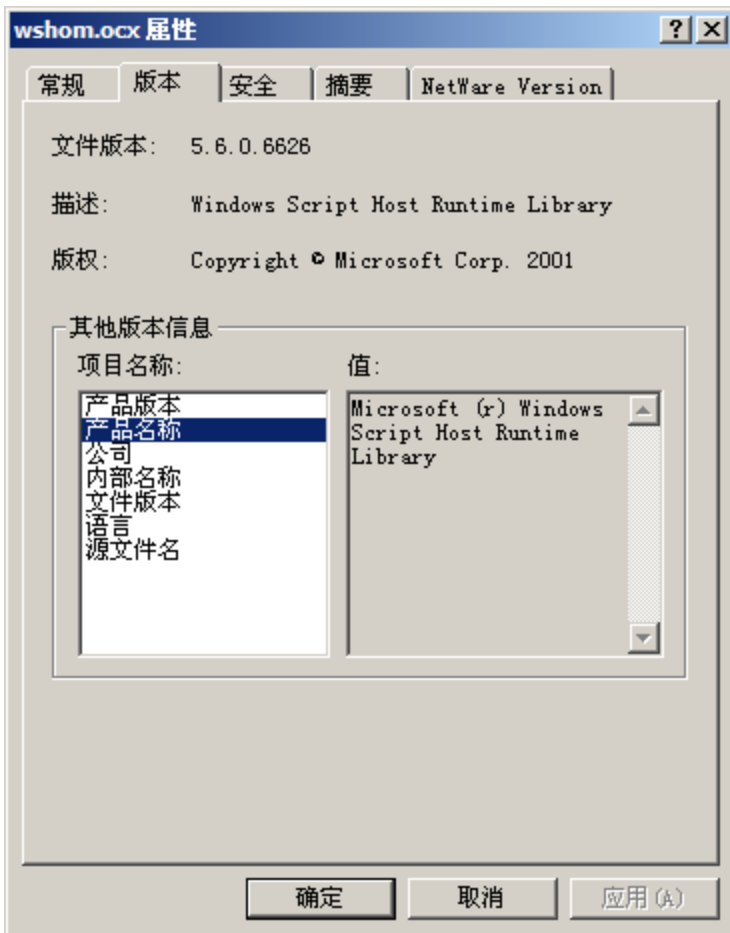


图8-5 WSH是一个ActiveX控件，用来创建一些进行简单或复杂操作的脚本，而这种工作在早前的MS-DOS操作系统里是有Writing batch文件（.bat）来完成的

1. 在VB编辑器窗口，选择“工具” - “引用”
2. 在引用对话框，找到并选择Microsoft Scripting Runtime

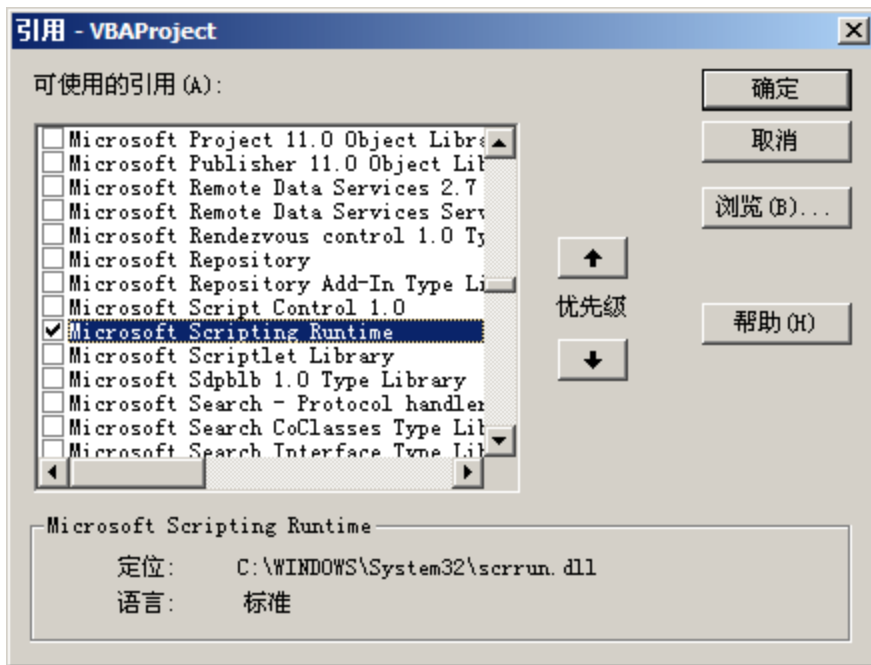


图8-6 对Microsoft Scripting Runtime创建引用

3. 现在，按下F2打开对象浏览器
4. 在“所有库”的下拉列表里选择“Scripting”。你将看到WSH库里面的部分对象列表。WSH让你轻易的获得一些问题的答案，例如“在哪个硬盘上我可以找到某个文件？”（GetDrive方法），“某文件的扩展名是什么？”（GetExtensionName方法），“该文件最后一次修改在什么时候？”（DateLastModified属性）以及“在给定的硬盘上存在某个文件夹或者文件吗？”（FolderExists和FileExists方法）。

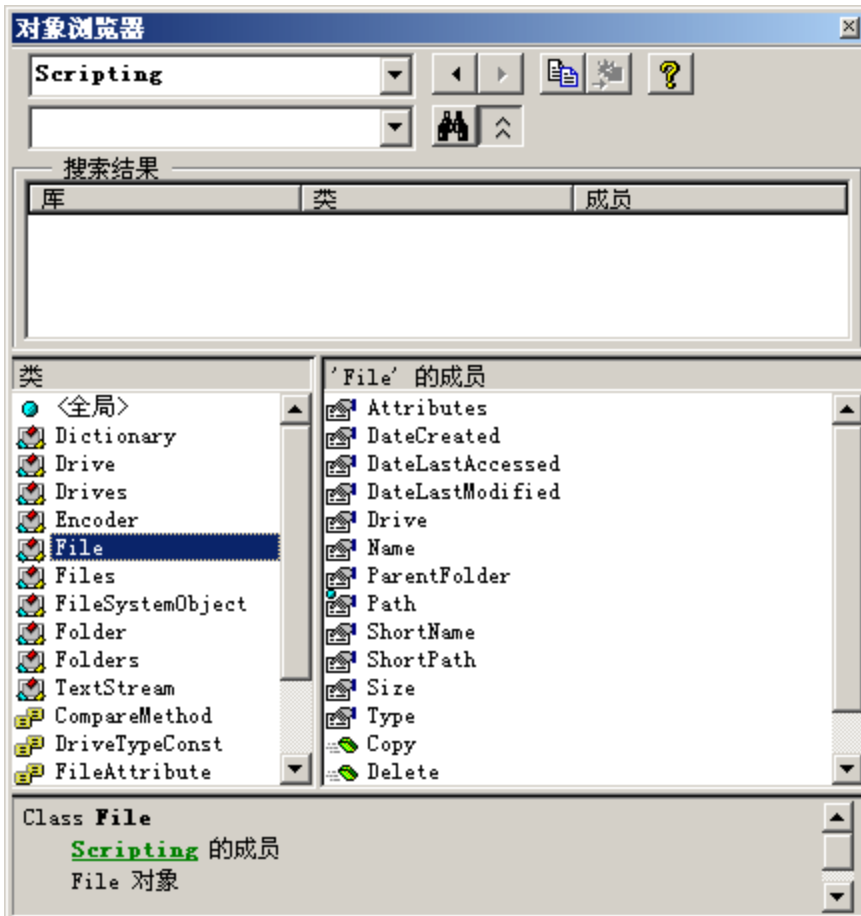


图8-7 创建对Microsoft Scripting Runtime的引用之后（见图8-6），对象浏览器显示了很多对象，让你使用硬盘，文件夹，文件和其它内容。

24.使用 WSH 获取文件信息

WSH有个对象叫做FileSystemObject，该对象有好几种方法来操纵文件系统。我们来看看你如何获取某个特定文件的信息：

1. 在当前工程里插入一新模块，并重命名为WSH
2. 在模块WSH里，输入下述过程FileInfo:

```
Sub FileInfo()
    Dim fs As Object
    Dim objFile As Object
    Dim strMsg As String

    Set fs = CreateObject("Scripting.FileSystemObject")
    Set objFile = fs.GetFile("C:\WINNT\System.ini") '译者：有可能是C:\Windows\System.ini
    strMsg = "File name: " & _
        objFile.Name & vbCrLf
    strMsg = strMsg & "Disk: " & _
        objFile.Drive & vbCrLf
    strMsg = strMsg & "Date Created:" & _
        objFile.DateCreated & vbCrLf
    strMsg = strMsg & "Date Modified:" & _
        objFile.DateLastModified & vbCrLf
    MsgBox strMsg, "File Information"
End Sub
```

上面显示的过程FileInfo使用VBA函数CreateObject来创建一个ActiveX对象（FileSystemObject），该对象提供访问电脑文件系统的途径


```
Dim fs As Object
```

```
Set fs = CreateObject("Scripting.FileSystemObject")
```

上面的代码声明了一个对象变量，名为fs。然后使用函数CreateObject创建一个ActiveX对象并且将该对象赋予对象变量。上面程序的第二行代码

```
Set objFile = fs.GetFile("C:\WINNT\System.ini"),
```

创建和返回对C:、WINNT的System.ini文件对象的引用，并且将其赋值给对象变量objFile。你可以读取File对象的很多属性，例如，objFile.Name语句返回该文件的完整文件名，语句objFile.Drive返回该文件所处的硬盘名称，语句objFile.DateCreated和objFile.DateLastModified分别返回该文件的创建日期和最后一次修改日期。这个程序可以容易地修改成返回文件类型，属性和它的父文件夹的名称。请试图使用下述指令你自己修改该过程：objFile.Type, objFile.Attributes, objFile.ParentFolder以及 objFile.Size。在对象浏览器里面点击File对象，看看关于文件，你还可以学习到别的什么。

25.FileSystemObjec 的方法和属性

FileSystemObjedt是个ActiveX控件，提供了到计算机文件系统的访问。该对象提供了很多种方法，表8-3里面列出了其中的一些。

方法	描述
FileExists	如果文件存在就返回True
	<pre>Sub FileExists() Dim fs As Object Dim strFile As String Set fs = CreateObject("Scripting.FileSystemObject") strFile = InputBox("Enter the full name of the file:") If fs.FileExists(strFile) Then MsgBox strFile & " was found." Else MsgBox "File does not exist." End If End Sub</pre>
GetFile	返回一对象File
GetFileName	返回带路径文件名
GetFileVersion	返回文件版本
CopyFile	复制文件
	<pre>Sub CopyFile() Dim fs As Object Dim strFile As String Dim strNewFile As String strFile = "C:\Hello.doc" strNewFile = "C:\Program Files\Hello.doc" Set fs = CreateObject("Scripting.FileSystemObject") fs.CopyFile strFile, strNewFile MsgBox "A copy of the specified file was created." Set fs = Nothing End Sub</pre>
MoveFile	移动文件
DeleteFile	删除文件
	<pre>Sub DeleteFile() Dim fs As FileSystemObject Set fs = New FileSystemObject fs.DeleteFile "C:\Program Files\Hello.doc" MsgBox "The requested file was deleted." End Sub</pre>
DriveExists	如给定硬盘存在则返回True

	<pre> Function DriveExists(disk) Dim fs As Object Dim strMsg As String Set fs = CreateObject("Scripting.FileSystemObject") If fs.DriveExists(disk) Then strMsg = "Drive " & UCase(disk) & " exists." Else strMsg = UCase(disk) & " was not found." End If DriveExists = strMsg ' run this function from the worksheet ' by entering in any cell the following: =DriveExists("E:\") End Function </pre>
GetDrive	返回对象Drive
	<pre> Sub DriveInfo() Dim fs, disk, infoStr, strDiskName strDiskName = InputBox("Enter the drive letter:", _ "Drive Name", "C:\") Set fs = CreateObject("Scripting.FileSystemObject") Set disk = fs.GetDrive(fs.GetDriveName(strDiskName)) infoStr = "Drive: " & UCase(strDiskName) & vbCrLf infoStr = infoStr & "Drive letter: " & _ UCase(disk.DriveLetter) & vbCrLf infoStr = infoStr & "Drive Type: " & disk.DriveType & vbCrLf infoStr = infoStr & "Drive File System: " & _ disk.FileSystem & vbCrLf infoStr = infoStr & "Drive Serial Number: " & _ disk.SerialNumber & vbCrLf infoStr = infoStr & "Total Size in Bytes: " & _ FormatNumber(disk.TotalSize / 1024, 0) & " Kb" & vbCrLf infoStr = infoStr & "Free Space on Drive: " & _ FormatNumber(disk.FreeSpace / 1024, 0) & " Kb" & vbCrLf MsgBox infoStr, vbInformation, "Drive Information" End Sub </pre>
GetDriveName	返回一个含有硬盘名称或者网络共享名称的字符串
	<pre> Function DriveName(disk) Dim fs As Object Dim strDiskName As String Set fs = CreateObject("Scripting.FileSystemObject") strDiskName = fs.GetDriveName(disk) DriveName = strDiskName ' run this function from the Immediate window ' by entering ?DriveName("D:\") End Function </pre>
FolderExists	如文件夹存在则返回True
	<pre> Sub DoesFolderExist() Dim fs As Object Set fs = CreateObject("Scripting.FileSystemObject") MsgBox fs.FolderExists("C:\Program Files") End Sub </pre>
GetFolder	返回对象Folder
	<pre> Sub FilesInFolder() Dim fs As Object Dim objFolder As Object Dim objFile As Object Set fs = CreateObject("Scripting.FileSystemObject") </pre>

	<pre> Set objFolder = fs.GetFolder("C:\") Workbooks.Add For Each objFile In objFolder.Files ActiveCell.Select Selection.Formula = objFile.Name ActiveCell.Offset(0, 1) _ .Range("A1").Select Selection.Formula = objFile.Type ActiveCell.Offset(1, -1) _ .Range("A1").Select Next Columns("A:B").Select Selection.Columns.AutoFit End Sub </pre>
GetSpecialFolder	<p>返回操作系统文件夹路径：</p> <p>0 — 视窗文件夹</p> <p>1 — 系统文件夹</p> <p>2 — 临时文件夹</p>
	<pre> Sub SpecialFolders() Dim fs As Object Dim strWindowsFolder As String Dim strSystemFolder As String Dim strTempFolder As String Set fs = CreateObject("Scripting.FileSystemObject") strWindowsFolder = fs.GetSpecialFolder(0) strSystemFolder = fs.GetSpecialFolder(1) strTempFolder = fs.GetSpecialFolder(2) MsgBox strWindowsFolder & vbCrLf _ & strSystemFolder & vbCrLf _ & strTempFolder, vbInformation + vbOKOnly, _ "Special Folders" End Sub </pre>
CreateFolder	创建文件夹
	<pre> Sub MakeNewFolder() Dim fs, objFolder Set fs = CreateObject("Scripting.FileSystemObject") Set objFolder = fs.CreateFolder("C:\TestFolder") MsgBox "A new folder named " & _ objFolder.Name & " was created." End Sub </pre>
CopyFolder	复制文件夹
	<pre> Sub MakeFolderCopy() Dim fs As FileSystemObject Set fs = New FileSystemObject If fs.FolderExists("C:\TestFolder") Then fs.CopyFolder "C:\TestFolder", "C:\FinalFolder" MsgBox "The folder was copied." End If End Sub </pre>
MoveFolder	移动文件夹
DeleteFolder	删除文件夹
	<pre> Sub RemoveFolder() Dim fs As FileSystemObject Set fs = New FileSystemObject If fs.FolderExists("C:\TestFolder") Then fs.DeleteFolder "C:\TestFolder" MsgBox "The folder was deleted." End If End Sub </pre>

CreateTextFile	创建文本文件
OpenTextFile	打开文本文件
	<pre> Sub ReadTextFile() Dim fs As Object Dim objFile As Object Dim strContent As String Dim strFileName As String strFileName = "C:\WINNT\System.ini" Set fs = CreateObject("Scripting.FileSystemObject") Set objFile = fs.OpenTextFile(strFileName) Do While Not objFile.AtEndOfStream strContent = strContent & objFile.ReadLine & vbCrLf Loop objFile.Close Set objFile = Nothing ActiveWorkbook.Sheets(3).Select Range("A1").Select Selection.Formula = strContent End Sub </pre>

对象FileSystemObject只有一个属性，叫做Drives，它返回对硬盘驱动集合的引用。使用该属性，你可以查看一个电脑硬盘的清单，如下所示：

```

Sub DrivesList()
Dim fs As Object
Dim colDrives As Object
Dim strDrive As String

Set fs = CreateObject("Scripting.FileSystemObject")
Set colDrives = fs.Drives
For Each Drive In colDrives
strDrive = "Drive " & Drive.DriveLetter & ": "
Debug.Print strDrive
Next
End Sub

```

26.对象 File 的属性

对象File允许你访问某特定文件的所有属性，下面的代码创建了对对象File的引用：

```

Set fs = CreateObject("Scripting.FileSystemObject")
Set objFile = fs.GetFile("C:\My Documents\myFile.doc")

```

在本章前面的过程FileInfo里你可以发现使用对象File的例子。

属性	描述
Attributes	返回文件属性（和本章前面介绍的VBA函数GetAttr比较一下）
DateCreated	文件创建日期
DateLastAccessed	最后访问日期
DateLastModified	最后修改日期
Drive	驱动器名称，后面带冒号
Name	文件名
ParentFolder	文件的父文件夹
Path	文件完整路径
Size	以字节表示的文件大小（与本章前面介绍的VBA函数FileLen比较一下）

Type	文件类型。这就是显示在Windows文件浏览器类型一列中的文本（例如，参数设置，应用程序，快捷方式）
------	--

27. 文件夹对象属性

对象Folder提供了对所有文件夹属性的访问。下面的代码行创建了对对象Folder的引用：

```
Set fs = CreateObject("Scripting.FileSystemObject")
Set objFolder = fs.GetFolder("C:\My Documents")
```

属性	描述
Attributes	文件夹属性
DateCreated	文件夹创建日期
Drive	包含该文件夹的驱动器（译者：原文为Name of Folder）
Files	文件夹中的文件集合
	<pre>Sub CountFilesInFolder() Dim fs, strFolder, objFolder, colFiles strFolder = InputBox("Enter the folder name:") If Not IsFolderEmpty(strFolder) Then Set fs = CreateObject("Scripting.FileSystemObject") Set objFolder = fs.GetFolder(strFolder) Set colFiles = objFolder.Files MsgBox "The number of files in the folder " & _ strFolder & "=" & colFiles.Count End If End Sub</pre> <p>上面的过程调用函数IsFolderEmpty，该函数将在本表中的大小属性里讨论。</p>
IsRootFolder	如果本文件夹为根文件夹则返回True
Name	文件夹名称
ParentFolder	该文件夹的父文件夹
Path	文件夹的完整路径
Size	文件夹大小，以字节表示
	<pre>Function IsFolderEmpty(myFolder) Dim fs, objFolder Set fs = CreateObject("Scripting.FileSystemObject") Set objFolder = fs.GetFolder(myFolder) IsFolderEmpty = (objFolder.Size = 0) End Function</pre>
SubFolders	该文件夹中的子文件夹集合
Type	文件夹类型（例如文件夹或回收站）

28. 驱动器对象属性

对象Drive提供了访问到电脑或服务上某驱动的属性。下述代码创建了对Drive对象的引用：

```
Set fs = CreateObject("Scripting.FileSystemObject")
Set objDrive = fs.GetDrive("C:\")
```

在下面的表格里，你将看到好几个使用Drive对象的程序示例。

属性	描述
AvailableSpace	可用空间，字节表示
FreeSpace	同AvailableSpace
DriveLetter	驱动器字母（没有冒号）

DriveType	驱动器类型： 0 — 未知 1 — 可移动 2 — 固定 3 — 网络 4 — CD驱动 5 — RAM
	<pre> Sub CDROM_DriveLetter() Const CDROM = 4 Dim fs, colDrives Set fs = CreateObject("Scripting.FileSystemObject") Set colDrives = fs.Drives For Each Drive In colDrives If Drive.DriveType = CDROM Then MsgBox "The CD-ROM Drive: " & _ Drive.DriveLetter End If Next End Sub </pre>
FileSystem	文件系统，例如，FAT, NTFS或CDFS
IsReady	如果合适的媒体（例如CD）插入了并且可以访问则返回True
	<pre> Function IsCDROMReady(strDriveLetter) Dim fs, objDrive Set fs = CreateObject("Scripting.FileSystemObject") Set objDrive = fs.GetDrive(strDriveLetter) IsCDROMReady = (objDrive.DriveType = 4) And _ objDrive.IsReady = True ' run this function from the Immediate window ' by entering: ?IsCDROMReady("D:") End Function </pre>
Path	根文件夹路径
SerialNumber	驱动器系列号
TotalSize	驱动器总容量，以字节表示

29.使用 WSH 创建文本文件

WSH提供了三种创建文本文件的方法：CreateTextFile，OpenTextFile和OpenAsTextStream。下面的表格里列出了每种方法和示例。

方法/语法	示例
CreateTextFile	<pre>object.CreateTextFile(filename[, overwrite[, unicode]])</pre> <p>Object是对象FileSystemObject或Folder的名称。 filename是明确要创建文件的字符串表达式。 Overwrite (可选的) 是个布尔值，表明你是否要覆盖已经存在的文件，如果可以覆盖那么该值为True，如果不能覆盖则为False，如果忽略，那么现存的文件将不会被覆盖。 Unicode (可选的) 是个布尔值，表明该文件创建为Unicode或者ASCII类型的文件。如果文件创建为Unicode类型那么该值为真，如果文件为ASCII的话那么该值为假。如果忽略掉，那么就会创建ASCII类型的文件。</p> <pre> Sub CreateFile_Method1() Dim fs, objFile </pre>

	<pre> Set fs = CreateObject("Scripting.FileSystemObject") Set objFile = fs.CreateTextFile("C:\Phones.txt", True) objFile.WriteLine ("Margaret Kubiak: 212-338-8778") objFile.WriteLine ("Robert Prochot: 202-988-2331") objFile.Close End Sub </pre> <p>上面的过程创建了一个文本文件来储存两个人的姓名和电话号码。因为在覆盖参数处是一个True布尔值，所以，如果C:\Phones.txt已经存在的话就会被覆盖。</p>
OpenTextFile	<pre> object.OpenTextFile(filename[, iomode[, create[, format]])] </pre> <p>Object是FileSystemObject对象的名称。 Filename是明确要打开的文件名称的字符串表达式。 Iomode (可选的) 是个布尔值，表示如果该文件不存在时是否创建一个新文件，如果可以创建新文件则为真，否则为假。如果忽略该参数，那么不会创建新文件。（译者，红色段落为原文翻译，应该是错误的） 参数iomode 可以是下述常数之一： ForReading (1) ForWriting (2) ForAppending (8) Create (可选的) 是个布尔值，表示如果该文件不存在时是否创建一个新文件，如果可以创建新文件则为真，否则为假。如果忽略该参数，那么不会创建新文件。 Format (可选的) 是用来表明打开文件类型的三种类型之一。如果忽略，文件就会打开为ASCII。 TristateTrue = 打开文件为ASCII。 TristateFalse = 打开文件为Unicode。 TristateUseDefault = 使用系统默认方式打开文件</p> <pre> Sub CreateFile_Method2() Dim fs, objFile Set fs = CreateObject("Scripting.FileSystemObject") Set objFile = fs.OpenTextFile("C:\Shopping.txt", _ ForWriting, True) objFile.WriteLine ("Bread") objFile.WriteLine ("Milk") objFile.WriteLine ("Strawberries") objFile.Close End Sub </pre>
OpenAsTextStream	<pre> object.OpenAsTextStream([iomode, [format]]) </pre> <p>Object是File对象名称。 Iomode (可选的) 表明读写模式，它可以是下述三个常数之一： ForReading (1) ForWriting (2) ForAppending (8) Format (可选的) 是用来表明打开文件类型的三种类型之一。如果忽略，文件就会打开为ASCII。 TristateTrue = 打开文件为ASCII。 TristateFalse = 打开文件为Unicode。 TristateUseDefault = 使用系统默认方式打开文件</p> <pre> Sub CreateFile_Method3() Dim fs, objFile, objText Set fs = CreateObject("Scripting.FileSystemObject") fs.CreateTextFile "New.txt" Set objFile = fs.GetFile("New.txt") Set objText = objFile.OpenAsTextStream(ForWriting, _ TristateUseDefault) objText.Write "Wedding Invitation" </pre>

```
objText.Close
Set objText = objFile.OpenAsTextStream(ForReading, _
TristateUseDefault)
MsgBox objText.ReadLine
objText.Close
End Sub
```

30.使用 WSH 进行其它操作

WSH使任何安装在你计算机上的自动化对象的操作成为可能。

除通过FileSystemObject访问文件系统之外，WSH也允许你进行其它的一些操作，例如，处理WSH和ActiveX对象，设定和去除打印机和远程驱动器，操纵注册表，创建视窗和互联网快捷方式以及访问Windows NT 活动地址服务。WSH对象模型由下述三种主要对象组成：WScript，WshShell和WshNetwork。本节示范如何利用WshShell对象来编写程序启动其它应用程序和创建快捷方式。

31.运行其它应用程序

在本书的下一章，你将学习多种从Excel里启动外部应用程序的方法。你可以加上即将在本节找到的三种方法。

假设你想要从VBA过程里启动Windows记事本，接下来的过程，你将看到使用WSH的对象WshShell来运行一个应用程序是多么容易。如果你要运行内置的计算器的话，只要将记事本应用程序名称改为Calc就可以了。

```
Sub RunNotepad()
    Dim WshShell As Object
    Set WshShell = CreateObject("WScript.Shell")
    WshShell.Run "Notepad"
    Set WshShell = Nothing
End Sub
```

上面的过程以声明和创建一个Wshshell对象开始：

```
Dim WshShell As Object
Set WshShell = CreateObject("WScript.Shell")
```

下一语句则使用Run方法来运行要求的应用程序：

```
WshShell.Run "Notepad"
```

使用相同的概念，很容易运行视窗的公用应用程序，例如计算器或浏览器：

```
WshShell.Run "Calc"
WshShell.Run "Explorer"
```

过程的最后一行消灭了对对象WshShell，因为，不再需要它了。

```
Set WshShell = Nothing
```

你可以启动你应用程序和某个特定的文件，而不是打开一个空的应用程序窗口，如下所示：

```
Sub OpenTxtFileInNotepad()
    Dim WshShell As Object
    Set WshShell = CreateObject("WScript.Shell")
    WshShell.Run "Notepad C:\Phones.txt"
    Set WshShell = Nothing
End Sub
```

试验下面的过程来打开MS-DOS窗口，并且将当前目录下的文件清单打印出来：

```
Sub RunDOSCommand()
    Dim WshShell As Object
    Set WshShell = CreateObject("WScript.Shell")
    WshShell.Run ("Command /c Dir >lpt1:")
End Sub
```

32.创建快捷方式

当你开始传播你的VBA应用程序的时候，用户可能会要求你自动在他们的桌面上放置一个你的软件的快捷方式。VBA自己没有提供创建快捷方式的方法。很幸运的是，你现在知道如何使用Wsh了，你可以使用它的对象Shell创建应用程序或者网页的快捷方式，不必要用户的干涉。对象WshShell使用了方法CreateShortcut，你可以按照下述方法：

Set myShortcut = WshShell.CreateShortcut(Pathname)

Pathname是明确快捷文件完整路径的字符串。所有的快捷方式文件都有扩展名.lnk，并且该扩展名必须包括在文件路径名里面。CreateShortcut方法返回快捷方式对象，下面的表格里列出了很多属性和一个方法。

方法/语法	示例
TargetPath	TargetPath属性是可执行文件的路径 WshShell.TargetPath = ActiveWorkbook.FullName
WindowStyle	WindowStyle属性明确快捷方式使用的窗口类型 1 – 普通窗口 3 – 最大化窗口 7 – 最小化窗口 WshShell.WindowStyle = 1
HotKey	HotKey属性是键盘快捷方式（例如，Alt+f, Shift+g, Ctrl+Shift+z, 等等） WshShell.Hotkey = "Ctrl+Alt+w"
IconLocation	IconLocation属性是快捷方式图标的位置。因为图标文件里通常不止一个图标，所以你应该提供图标文件的路径，并且后面标明图标在文件里的索引号。如果不明确的话，Windows 会使用缺省的图标。 WshShell.IconLocation = "notepad.exe, 0"
Description	Description属性包含一个描述快捷方式的字符串 WshShell.Description = "Wordware Web Site"
WorkingDirectory	WorkingDirectory属性明确快捷方式的工作目录 strWorkDir = WshShell.SpecialFolders("Desktop") WshShell.WorkingDirectory = strWorkDir
Save	这是对象Shortcut的唯一方法。在使用方法CreateShortcut创建一个快捷方式对象并且设置该快捷方式的属性后，必须使用Save方法将快捷方式对象保存到硬盘上。

创建快捷方式是个三步的过程：

1. 创建一个WshShortcut对象
2. 初始化它的属性
3. 用方法Save将它保存到硬盘

下面的例子创建一个WshShell对象和使用CreateShortcut方法创建两个快捷方式：一个到当前Excel工作簿的Windows快捷方式，和一个到Wordware Publishing网页的互联网快捷方式。两个快捷方式都放在用户的桌面上。该过程使用对象WshShell的SpecialFolders属性来返回到视窗桌面的路径。

Sub CreateShortcut()

```
' this script creates two desktop shortcuts
Dim WshShell As Object
Dim objShortcut As Object
```

```
Set WshShell = CreateObject("WScript.Shell")
```

```
' create an internet shortcut
```

```
Set objShortcut = WshShell.CreateShortcut(WshShell. _
    SpecialFolders("Desktop") & "\Wordware.url")
objShortcut.TargetPath = "http://www.wordware.com"
objShortcut.Save
```

```
' create a file shortcut
```

```
Set objShortcut = WshShell.CreateShortcut(WshShell. _
```

```
SpecialFolders("Desktop") & "\" & ActiveWorkbook.Name & ".lnk")
```

```
With objShortcut
    .TargetPath = ActiveWorkbook.FullName
    .WindowStyle = 7
    .Save
End With

Set objShortcut = Nothing
Set WshShell = Nothing
End Sub
```

技巧8-12 使用SpecialFolders属性

你可以使用SpecialFolders属性在你的机器上找到特殊文件夹的位置。下述特殊文件夹是可用的：AllUsersDesktop（所有用户桌面），AllUsersStartMenu（所有用户开始菜单），AllUsersPrograms（所有用户程序），AllUsersStartup（所有用户启动），Desktop（桌面），Favorites（收藏），Fonts（字体），MyDocuments（我的文档），NetHood（网络连接），PrintHood（打印机），Programs（程序），Recent（最近），SendTo（发送到），StartMenu（开始菜单），Startup（启动）和 Templates（模版）。如果请求的特殊文件夹不可用，那么SpecialFolders属性就会返回一个空字符串。

33.接下来……

在本章的课程里，你学习并且测试了让你操作文件系统的VBA函数和语句。你知道了如何读取和修改与文件和文件夹有关的信息，而且，知道了如何执行对顺序，随机和二进制文件的读和写的操作。你也学习了如何使用WSH来访问File SystemObject和进行其它操作，例如启动应用程序和使用对象WshShell创建Windows快捷方式。如果你对讨论的函数和语句更详细的东西感兴趣的话，那么就花些时间来浏览一下VB在线帮助吧。

接下来的一章将给你介绍更多的自动化任务。例如，你将学习如何使用VBA来控制其它应用程序。你将学习启动应用程序的多种方法，并且研究如何直接从Microsoft Excel里操纵其它应用程序。

第九章 利用 VBA 控制其它应用程序

你每天在办公室里或者家里在你的电脑上工作时，都要用到很多种应用程序。要从你的硬盘或者软盘上查找某个文件的话，你就要打开视窗浏览器。当你要设置系统时间或者更改屏幕外观的话，可以点击控制面板上的相应的图标。如果你的电脑上安装了微软办公软件套餐的话，就可以使用Word创建各种各样的文件，并且依靠Excel进行所有的计算。微软Access对于保存重要的数据表非常有用，而PowerPoint则有助于你使用声音和图片。最后，微软Outlook使你易于保存你的联系、时间和约会并且分享给他人。使用这些应用软件的时候，你经常要在它们之间切换，你可以使用键盘直接输入数据或者复制或移动数据。这些操作——打开应用程序以及在它们之间传输数据时不需要手动操作的。它们可以通过一些很有趣的VBA函数和指令来自动完成。在本章，你将学习多种从VBA过程里打开应用程序的方法，并且找到如何使用称为自动化的技术直接从微软Excel直接控制其它应用程序。

1.启动应用程序

启动一个应用程序的方法不止一个，实际上，你至少可以使用五种方法手动打开某个程序：通过“开始”|“程序”菜单，快捷键，“运行”命令，MS-DOS窗口，或者在视窗浏览器里双击可执行文件。

本节假设你对手动启动应用程序很熟悉，并且很想从Excel内部的VB编辑窗口试验其它启动应用程序的方法。

我们从最简单的开始吧——Shell函数。该函数使你从VBA过程里直接打开任意程序。假设你的过程必须打开视窗记事本，要打开记事本，你所有要做的就是关键字Sub和End Sub之间加上一条语句，或者更好的方法是在立即窗口里输入下述语句，并且按下回车键：

```
Shell "notepad.exe", vbMaximizedFocus
```

你将立即看到结果。

在上面的语句里，“notepad.exe”是你要打开的程序的名称。如果你担心程序找不到的话，那么该名称就应该包含完整的路径（启动器名称和文件夹名称）。注意，程序名称用双引号括起来了。Shell函数的第二个参数可以忽略。该参数明确窗口形式

（也就是当程序启动的时候，它如何显示在屏幕上的）。在上面的例子里，记事本将显示为最大化的窗口。如果没有明确窗口形式，那么程序就会被最小化（参见表9-1）。

窗口形式常数	值	窗口显示情况
vbHide	0	窗口被隐藏
vbNormalFocus	1	普通大小，并带焦点
vbMinimizedFocus（默认设置）	2	最小化，并带焦点（这是缺省设置）
vbMaximizedFocus	3	最大化，并带焦点
vbNormalNoFocus	4	普通大小，并失去焦点
vbMinimizedNoFocus	6	最小化，并失去焦点

如果Shell函数能够启动某个可执行文件，那么它就会返回一个叫做任务ID的号码。该号码是指示应用程序启动的唯一号码。如果Shell函数不成功的话（也就是说某应用程序不能打开），VB就会产生一错误。如果你要使用Shell函数启动的应用程序的话，就不要在Shell函数后面输入任何语句。Shell函数启动程序是不同时的，意思是说VB启动Shell函数指定的应用程序，并且，VB在启动程序后，立即就回到过程里面去继续剩余的指令（因此，你没有机会立即使用该应用程序）。你如果使用Shell函数来启动控制面板呢？

1. 打开一新工作簿，保存为Chap09.xls
2. 在VB编辑器窗口，插入新模块
3. 重新命名工程为WorkWApplets，模块名为ShellFunction
4. 输入下面显示的过程StartPanel:

```
Sub StartPanel()  
    Shell "Control.exe", vbNormalFocus  
End Sub
```

控制面板里面有很多图标，每个图标执行一个或者多个任务。众所周知，在每个图标后面都有一个程序的，当用户双击图标或者用箭头选择该图标然后按下Enter键，该程序都会被激活。作为一个规律，你总是可以通过查看某个图标的属相来检查什么文件名驱动某个图标。不幸的是，控制面板里面的图标的属性选择都被禁止了。然后，你可以通过创建一个到该图标的快捷键来查找控制面板里图标文件。例如，在你创建一个更改电脑原始设置的过程之前，我们来找出激活该图标的文件名称。

1. 从“开始”菜单里选择“设置”，然后选择“控制面板”（在Windows XP开始菜单里可以直接看到“控制面板”）
2. 在控制面板窗口里，右键单击“初始选项”图标，并且从快捷菜单中选择创建快捷键
3. 点击确定，将快捷键放在桌面上
4. 关闭控制面板窗口
5. 返回桌面，在初始选项的快捷键上单击右键，然后选择属性
6. 在属性窗口，点击快捷键页，然后点击更改图标按钮



图9-1 每个控制面板里的图标都有一个后缀名为.cpl的文件

7. 写下.cpl文件名称（Control Panel Library）或者动态链接库文件（.dll）并关闭该练习中开启的所有窗口

表9-2 激活控制面板图标的一些文件示例

控制面板图标	.cpl或者.dll文件
电话和调制解调器选项	TELEPHON.CPL或MODEM.CPL
添加/删除程序	APPWIZ.CPL
网络和拨号连接	NETCPL.CPL或NETSHELL.DLL
32-Bit ODBC	ODBCCP32.CPL
系统	SYSDM.CPL
邮件	MLCFG32.CPL
用户和密码	PASSWORD.CPL或NETPLWIZ.DLL
日期/时间	TIMEDATE.CPL
区域选项	INTL.CPL
Internet选项	INETCPL.CPL
声音和多媒体属性	MMSYS.CPL
显示	DESK.CPL
鼠标	MAIN.CPL

下面的国产ChangeSettings示范如何使用Shell函数来启动控制面板的初始设置图标。注意Shell函数的参数必须写在括号里，如果你后面需要在你的程序里使用它返回值的话。

1. 在当前模块里输入过程ChangeSettings，如下所示：

```
Sub ChangeSettings()  
    Dim nrTask  
    nrTask = Shell("Control.exe intl.cpl", vbMinimizedFocus)  
    Debug.Print nrTask  
End Sub
```

2. 运行几次过程ChangeSettings，每次从表9-2里列出的清单里提供一个不同的.cpl文件。你可能需要将程序改为：

```
Sub ChangeSettings2()  
    Dim nrTask  
    Dim iconFile As String
```



```
iconFile = InputBox("Enter the name of the CPL or DLL file:")
nrTask = Shell("Control.exe " & iconFile, vbMinimizedFocus)
Debug.Print nrTask
```

End Sub

如果你要启动的程序是微软应用程序，那么除了使用Shell函数外，你还可以很方便地使用VB的方法ActivateMicrosoftApp来实现。该方法在微软Excel应用程序的对象里是可用的，例如，要从立即窗口启动PowerPoint的话，你所有要做的事情就是输入下面的指令并且按下Enter：

Application.ActivateMicrosoftApp xlMicrosoftPowerPoint

注意ActivateMicrosoftApp方法要求一个常量来指定要启动的程序。如果PowerPoint没有打开的话，上面的过程就会打开PowerPoint，但是如果该程序已经打开的话，该指令不会再打开一个新的PowerPoint界面，只是简单的激活已经在运行的应用程序。你可以结合ActivateMicrosoftApp方法使用下列常量，常量的名称指名应用程序名称。

应用程序名称	常量
Access	xlMicrosoftAccess
FoxPro	xlMicrosoftFoxPro
Mail	xlMicrosoftMail
PowerPoint	xlMicrosoftPowerPoint
Project	xlMicrosoftProject
Schedule	xlMicrosoftSchedulePlus
Word	xlMicrosoftWord

2.在应用程序之间切换

因为用户可以同时在Windows环境下使用多个应用程序，所以你的VBA过程必须要知道如何在打开的程序之间切换。假设除了Excel之外，你还打开了另外两种应用程序：Word和Explorer。你可以按照下面的语法使用AppActivate语句来激活已经打开的程序：

AppActivate title [, wait]

只有标题参数是必须的，这是应用程序的名称，正如它显示在应用程序窗口的标题栏那样，或者它也可以是Shell函数返回的任务ID号码。注意，参数title要跟每个正运行的应用程序的标题字符串进行对比，如果没有精确的匹配，那么任何标题字符串里前面的字符和参数title一致的应用程序就会被激活。（译者：例如，你要激活Excel，那么title参数应该是“Microsoft Excel”，如果你写的是“Microsoft”，那么激活的就也可能是Word，PowerPoint……）。第二个参数wait是可选的，它是个布尔值（True或False），明确VB什么时候激活应用程序。如果在这里是False的话，该应用程序就立即会被激活，甚至被调应用程序并没有焦点。如果在wait参数处放置True的话，那么被调的应用程序就会等到它有了焦点，然后才会激活该应用程序。例如，要激活Word，你就得输入下列语句：

AppActivate "Microsoft Word"

注意，应用程序名称用双引号引用起来。你也可以使用Shell函数返回的数值作为语句AppActivate的参数：

```
' run Microsoft Word 运行Word应用程序
ReturnValue = Shell("C:\Microsoft Office\Office\Word.exe",1)
' activate Microsoft Word 激活Word
AppActivate ReturnValue
```

语句AppActivate用来在应用程序之间切换，所以要求这些程序已经在运行。该语句仅仅改变焦点，指定的应用程序变为当前活动的窗口。AppActivate语句不会启动任何应用程序，参见下一章节的过程FindCPLFiles，这也是使用该语句的一个例子。我们来练习一下最近介绍的几个VBA语句：

1. 通过在立即窗口里输入下列VBA语句来打开资源管理器：

Shell "Explorer"

按下回车键后，被请求的应用程序就被打开了，带有“我的文档”文件夹的图标就会出现在任务栏上。

2. 在立即窗口里输入下列代码：

AppActivate "My Documents"

按下回车键后，焦点就会移至我的文档窗口。

3.控制其它应用程序

既然你已经知道了如何使用VBA语句来启动一个程序，以及在应用程序之间切换，那么让我们看看一个应用程序是如何与另

外一个应用程序交流的。对于一个应用程序来说，要控制另一个应用程序的最简单的方式就是使用SendKeys语句。该语句允许你将许多的按键发送到活动应用程序窗口，你可以发送一个或组合键并且得到直接使用键盘的同样效果。SendKeys语句如下所示：

SendKeys string [, wait]

这个必须的参数string是你要发送到活动应用程序窗口的键或组合键，例如，使用下列指令来发送字母“f”键：

SendKeys "f"

要发送组合键Alt+f，使用：

SendKeys "%f"

百分号(%)是表示Alt键的字符串。要发送例如Shift+Tab的组合键的话，那么就要使用下面的语句：

SendKeys "+{TAB}"

加号(+)表示Shift键。要发送其它键或者其它组合键的话，请参见表9-3列出的相应字符串。

技巧9-1 SendKeys和其它应用程序

你只能发送按键到那些为微软视窗操作系统设置的应用程序。

技巧9-2 SendKeys和被保护的字符

有些字符在和SendKeys语句一起使用时具有特殊的意义，它们是：加号(+)，脱字符号(^)，符合(~)和括号()。要发送这些字符到另一个应用程序的话，就必须将它们用打括号{}括起来。要发送打括号时，则需要输入{}和{}。

SendKeys语句的第二个参数是可选的，wait是个逻辑值True或者False。如果是False（缺省），那么VB在发送按键后立即返回过程，如果为True，那么VB只有在发送的按键执行后才能返回到过程。

如果要发送一个表格9-3里面没有列出的字符的话，那么记住这些代码必须用引号括起来，例如：

SendKeys "{BACKSPACE}"

表9-3 SendKeys语句里使用的按键代码

键	代码	键	代码
空格键	{BACKSPACE}	滚动锁定	{SCROLLLOCK}
	{BS}	Tab	{TAB}
	{BKSP}	向上箭头	{UP}
Break键	{BREAK}	F1	{F1}
大写锁定键	{CAPSLOCK}	F2	{F2}
删除键	{DELETE}	F3	{F3}
	{DEL}	F4	{F4}
向下箭头	{DOWN}	F5	{F5}
End键	{END}	F6	{F6}
回车键	{ENTER}	F7	{F7}
	~	F8	{F8}
Esc键	{ESC}	F9	{F9}
帮助键	{HELP}	F10	{F10}
Home键	{HOME}	F11	{F11}
插入键	{INSERT}	F12	{F12}
	{INS}	F13	{F13}
向左箭头	{LEFT}	F14	{F14}
数字锁定键	{NUMLOCK}	F15	{F15}
向下翻页键	{PGDN}	F16	{F16}
向上翻页键	{PGUP}	Shift	+
屏幕打印键	{PRTSC}	Ctrl	^
向右箭头	{RIGHT}	Alt	%

技巧9-3 SendKeys语句对格敏感

当你使用SendKeys语句发送按键时，你一定要牢记区分字符的大小格。因此，要发送组合键Ctrl+d的话，你必须使用^d，而发送Ctrl+Shift+D的话，则必须使用字符串：^+d

在本章前期，你学习了.cpl文件启动多种控制面板的图标。你现在要创建的VBA过程目的是要定位你硬盘上所有扩展名为.cpl的文件。

1. 使用立即窗口来启动资源管理器：

Shell "Explorer."

“我的文档”图标将出现在屏幕下方的任务栏上。

2. 在当前工程里插入新模块并且重命名为SendKeysStatement
3. 输入过程FindCPLFiles，如下所示：

```
Sub FindCPLFiles()
    ' The keystrokes are for Windows 2000
    AppActivate "My Documents"
    ' activate the Search window 激活搜索窗口
    SendKeys "{F3}", True
    ' move the pointer to the Search for files 将光标移到搜索文件
    ' and folders named text box 和文件夹（名称在文本框里）
    SendKeys "%m", True
    ' type in the search string 输入要搜索的字符串
    SendKeys "*.cpl", True
    ' move to the Look in drop down box 焦点移到下拉框
    SendKeys "{Tab}{Tab}", True
    ' change to the root directory 更改根目录
    SendKeys "C:\", True
    ' execute the Search 执行搜索
    SendKeys "%s", True
End Sub
```

4. 切换到Excel应用程序窗口并且运行过程FindCPLFiles（使用Alt+F8打开宏对话框，选择过程名称，再点击运行）。

上面过程的第一条语句使用AppActivate语句（参见前面章节）来激活已经打开的应用程序，还记得你在立即窗口里使用Shell语句激活了资源管理器吗？剩余的语句发送一些必要的按键到活动应用程序。本过程的结果是扩展名为.cpl的控制面板文件的搜索结果列表。你也可以使用一个SendKeys语句来发送所有必须的按键（参见下面的例子），然而，一步一步发送按键更容易理解程序。

```
Sub FindCPLFiles2()
    AppActivate "My Documents"
    SendKeys "{F3}%m*.cpl{Tab}{Tab}C:\%s", True
End Sub
```

4.控制应用程序的其它方法

尽管你可以使用SendKeys语句来传递命令给其它应用程序，但是你还是必须要求助于其它方法来获得对该应用程序的充分控制。有两种标准方法可以供应用程序和另外一种应用程序交流。最新的方法，被称为自动控制，它允许你访问和操纵另一种应用程序的对象。你可以通过自动控制编写VBA过程，通过引用其它应用程序的对象、属性和方法来控制其它应用程序。在本章接下来的章节里，你将学习如何通过自动控制来控制其它应用程序。称为DDE（动态数据交换）的老数据交换技术是允许你在两个应用程序之间动态发送数据的协议，它通过创建一个特殊的通道来发送和结束信息。DDE非常慢，使用困难，只有当你需要与一个不支持自动控制的老应用程序交流时，才需要使用DDE。

5.了解自动控制

当和另外一个应用程序交流时，你可能需要更多的功能，而不只是激活它来发送按键。例如，你可能需要在该应用程序里创建和操纵对象，你可以在Excel电子表格力插入整个Word文档。因为Excel和Word都支持自动控制，所以，你可以在Excel里编写一个VBA过程在操作Word对象，比如文档或者段落。支持自动控制的应用程序称为自动控制服务器（Automation servers）或者自动控制对象（Automation objects）。

能够操作服务器对象的应用程序称为自动控制控件。有些应用程序只能是服务器或者控件，而其它的则既可以是服务器也可以是控件。Microsoft Office 2000和2002都可以作为自动控制服务器和控件。自动控制控件可以是安装在你电脑上的各种

ActiveX控件，你将在下一章里学习这些对象。

6. 了解链接和嵌入

在你学习如何使用自动控制从VBA过程控制其它应用程序之前，我们来看一看如何手动链接和插入对象。人们熟知的OLE，对象链接和嵌入，允许你创建组合文档。组合文档包含其它应用程序创建的对象。例如，如果你要在Excel里嵌入一个Word文档的话，Excel只要知道创建该对象需要用到的应用程序名称，以及该对象在屏幕上显示的方法。组合文档有链接或者对象嵌入产生。当你使用手动方法来嵌入对象时，你首先要在一个应用程序里复制它，再粘贴到另一个应用程序里。链接对象和嵌入对象的主要区别是对象储存和更新的方式。我们来试验一下：

1. 激活Word并打开任意一个文档
2. 选择和复制任意一段文本
3. 在Excel工作表里，使用下述四种方法之一将复制的文本进行粘贴：
 - 粘贴为文本（选择编辑|粘贴）。复制的文本就会出现在活动单元格（见图9-2，单元格A2）
 - 粘贴为嵌入对象（选择编辑|选择性粘贴，点击“粘贴选项”按钮，并且在清单里选择“Microsoft Word Document 对象”。）粘贴的文本将作为一个嵌入的对象（见图9-2，单元格A5）。该嵌入的对象成为了目的文件的一部分。因为该嵌入的对象没有和原始数据链接，所以该信息是静态的。当文件源中的数据改变时，该嵌入的对象不会被更新。如果要更改嵌入的数据，你就必须双击它，这样就会打开该对象在源程序里编辑它。当然，该源程序必须已经安装在你的电脑上了。当你嵌入对象时，所有的数据都会存储在目的文件里，这会导致文件大小显著增大。注意，当你嵌入一个对象后，Excel的编辑栏里将显示：
`=EMBED("Word.Document.8","")`
 - 粘贴为链接对象（选择编辑|选择性粘贴，点击“粘贴链接”选项，然后在列表里选择“Microsoft Word Document 对象”）。虽然目的文件显示了所有的数据，但是它仅仅储存了该数据的地址。当你双击该链接的对象时（见图9-2，单元格A9），原应用程序就会被启动。链接对象是一种动态的操作，这意味着当源文件里的数据改变时，链接的数据就会自动更新。因为目的文件只包含对象如何与源文件链接的信息，所以，对象链接并不会增加目的文件的大小。下面的公式是Excel用来链接对象的：
`=Word.Document.8|'C:\Documents and Settings\tj8147\My Documents\Tiger\VB\Excel2002_Programming\Chinese\Excel2002VBA_Ch9.doc'!!OLE_LINK2'`（译者：由于文件存储位置不同，本节的翻译可能和你的情况不一样，请注意分辨）
 - 粘贴为超链接（选择粘贴|超链接译者：应该为“编辑”|“粘贴为超链接”）粘贴的数据在工作表里显示为带下划线、有颜色的文本（见图9-2，单元格A11）。点击该超链接，你可以快速地激活该源文件。

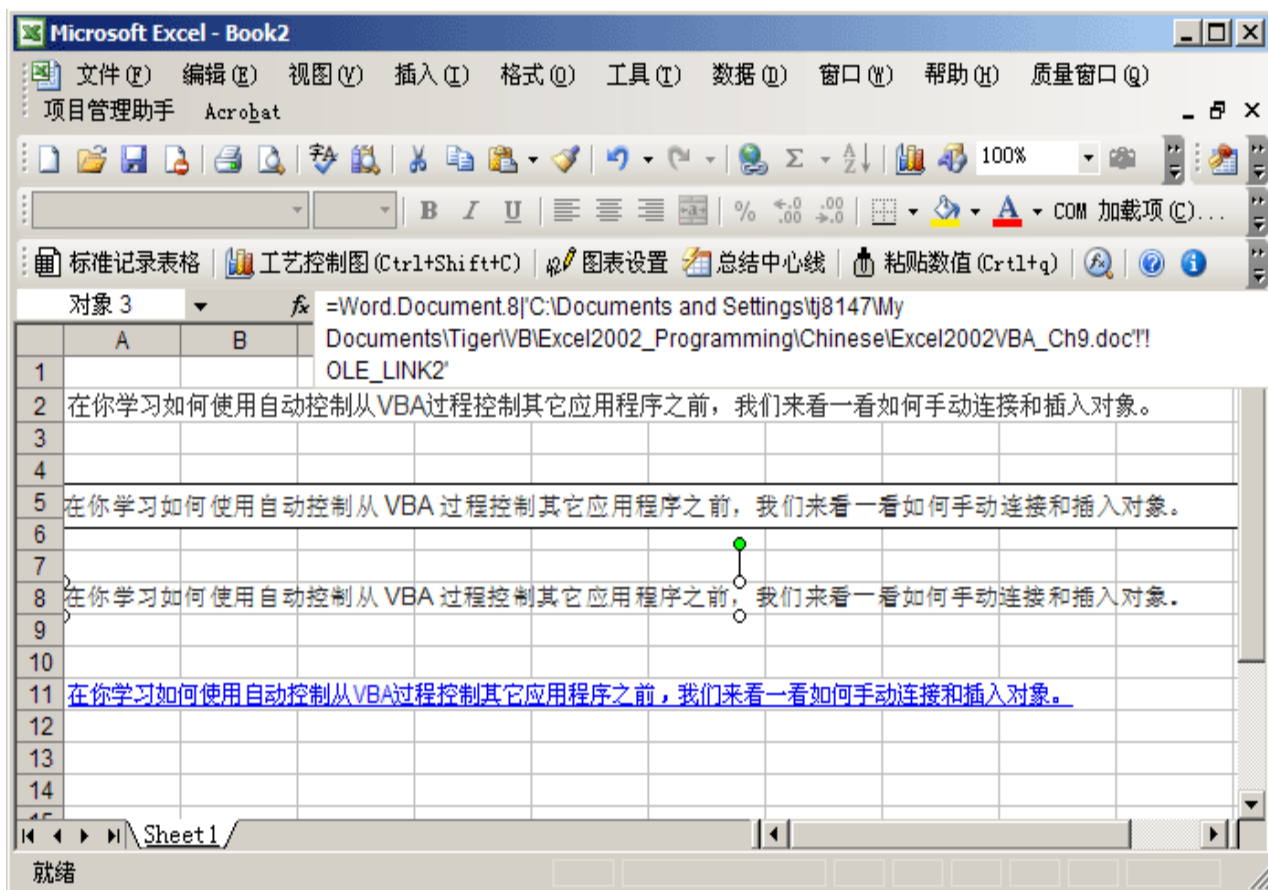


图9-2 示范链接和嵌入

7.使用 VBA 进行链接和嵌入

过程InsertLetter示范了如何使用程序在Excel嵌入一个Word文档。用你自己的文件名称代替引用“C:\Hello.doc”。过程InsertLetter使用AddOLEObject方法，该方法创建一个OLE对象，并且返回一个对该新OLE对象的Shape对象。在VB在线帮助里面，你可以找到AddOLEObject方法可用的其它参数。

1. 在当前工程里面插入一新模块，并重命名为OLE
2. 输入过程InsertLetter，如下所示：

```
Sub InsertLetter()  
    Workbooks.Add  
    ActiveSheet.Shapes.AddOLEObject FileName:="C:\Hello.doc"  
End Sub
```

上面的过程打开一个新工作簿，然后嵌入该指定的Word文档。要链接一个文档的话，你就必须明确另外一个参数Link，如下所示：

```
ActiveSheet.Shapes.AddOLEObject _  
    FileName:="C:\Hello.doc", Link:=True
```

技巧9-4 对象链接和嵌入

当你不得不做出决定是否使用嵌入还是链接对象时，只要有下列之一的条件，那么就使用嵌入：

- 你不在乎文档大小，或者你有足够的硬盘空间和内存来处理大文件
- 你再也不会其它复合文档里使用源文件或者源文本
- 你想要将该文档通过电子邮件或者磁盘发送给别人，并且确保他们能够顺利地读取数据。

（译者：本人也倾向于使用嵌入，因为链接经常会问你是否要更新链接，而且，很多人经常会忘记发送源文件给别人。）

8.COM 和自动控制

在自动控制后面的驱动力量是组件对象模型（COM），它决定了服务器应用程序创建对象的规则，也明确服务器和控制应用程序在使用这些对象时必须遵循的方法。COM标准包含作为自动控制界面（Automation interfaces）可用的函数集合。当服务器应用程序创建一个对象是，它会自动地制作一个和它一起可用的界面。该界面包括该对象可识别的属性、方法和事件。控制应用程序不需要为了控制该对象去了解它的内部结构，只需要知道如何操作服务器应用程序制作的对象界面。

9.了解绑定

对于控制应用程序与自动控制对象（服务器）来说，你必须将你的VBA过程中可用的对象和服务器的自动控制对象联系起来，这个过程就叫做绑定。这里有两种类型的绑定：后期绑定和早期绑定。你对绑定的选择对你的应用程序表现影响很大。

10.后期绑定

当你声明一个变量 **As Object** 或者**As Variant**时，VB使用的是后期绑定。后期绑定也叫运行绑定。简单地说，后期绑定意味着VB在设计时不会将你的对象变量和自动控制对象联系起来，而是要等到你实际运行该过程时才联系起来。因为**As Object**或者**As Variant**的声明在本质上是非常普通的，所以，VB在汇编时不能决定你变量指向的对象真正具有你的VBA过程使用的属性和方法。

下面的声明导致对指定对象的后期绑定：

```
Dim mydoc As Object
```

后期绑定的优势是所有的自动控制对象都知道如何使用。

后期绑定的劣势是对内置常量不支持。因为在设计时，VB并不知道你的对象指向的类型库，所以，你必须通过在应用程序文档里查询数值在你的代码里定义常量。同样，在运行时询问应用程序将放慢你程序的执行。

注意：后期绑定使得在另外一个应用程序的类型库里访问对象称为可能，而不需要首先建立对该对象库的引用。如果你不肯定你的用户是否在他们的机子上安装了指向的类型库，那么就使用后期绑定。

下面的过程示范如何使用后期绑定来打印Word文档。

```
Sub PrintWordDoc()
    Dim objWord As Object
    Set objWord = CreateObject("Word.Application")

    With objWord
        .Visible = True
        .Documents.Open "C:\Hello.doc"
        .Options.PrintBackground = False
        .ActiveDocument.PrintOut
    End With

    objWord.Documents.Close
    objWord.Quit
    Set objWord = Nothing
End Sub
```

技巧9-5 这是什么类型的绑定？

无论何时你使用常用的**Object**或**Variant**数据类型声明对象变量，请考虑后期绑定。后期绑定和早期绑定的主要区别是你如何声明你对象变量。

11.早期绑定

当你声明对象变量为明确的对象类型时，VB使用的是早期绑定。早期绑定也熟知为汇编绑定。这意味着VB在源代码转变为可执行代码时期，就将你的对象变量和自动控制对象联系起来了。常见的语法如下所示：

```
Dim objectVariable As Application.ObjectType
```

在上面的语法中，Application是应用程序的名称，正如它出现在对象浏览器里的工程库下拉清单里的样子（例如Word和Excel）。

ObjectType是对象类型（例如应用程序，文档，工作簿，工作表）。

下面的声明导致早期绑定：

```
Dim mydoc As Word.Document
```

```
Dim mydoc As Excel.Worksheet
```

早期绑定让你能够充分利用VB编辑器上可用的许多调试工具。例如你可以使用对象浏览器查找外部对象，属性和方法。VB的自动语法检测，自动列出成员以及自动显示快速信息（这些都在第二章里有讨论）可以让你在编写代码时更快，更少出错。另外，早期绑定允许你使用内置常量作为方法和属性设定的参数。因为这些常量在设计的时候在类型库里面就是可用的，所以你不需定义它们。这些非常方便的内置语法检测，智能特点和对内置常量的支持，在后期绑定里是不可用的。虽然使用早期绑定的VBA过程执行得更快一些，但是一些非常老的视窗应用程序只能使用后期绑定。

注意：为了使用早期绑定，你必须首先建立对对象库的引用（参见接下来的章节）。当你确定你的用户安装了引用的类型库时，就使用早期绑定。

12.建立到对象库的引用

如果你决定通过自动控制使用早期绑定连接到另外的应用程序的话，你首先就应该建立对包括你要操作对象的对象库的引用。依照下面的步骤创建对Microsoft Word 对象库的引用：

1. 激活VB编辑器窗口
2. 在工程浏览器里选择当前工程，并且选择“工具”|“引用”
3. 在引用对话框里，选择“可使用的引用”列表框里面的应用程序名称。例如，点击Microsoft Word 9.0 Object Library或者Microsoft Word 10.0 Object Library旁边的复选框（见图9-3）（译者：这里引用的是Microsoft Word 11.0 Object Library）。拉下可用引用列表框的滚动条定位该对象库，如果你已经安装了某个应用程序但是它的类型库在可用引用列表框里面没有出现的话，那么你可以点击“浏览”按钮。
4. 点击确定按钮关闭引用对话框。

引用对话框列出了VBA工程可用的引用名称。没有使用的引用按字母顺序列出，勾选上了的引用按优先顺序列出。例如，在Excel里，Microsoft Excel 10.0 Object Library比Microsoft Word 10.0或者9.0 Object Library具有更高的优先顺序。当一个过程引用一个对象时，VB从引用对话框里列出的库里按顺序搜索所有被引用的对象库。

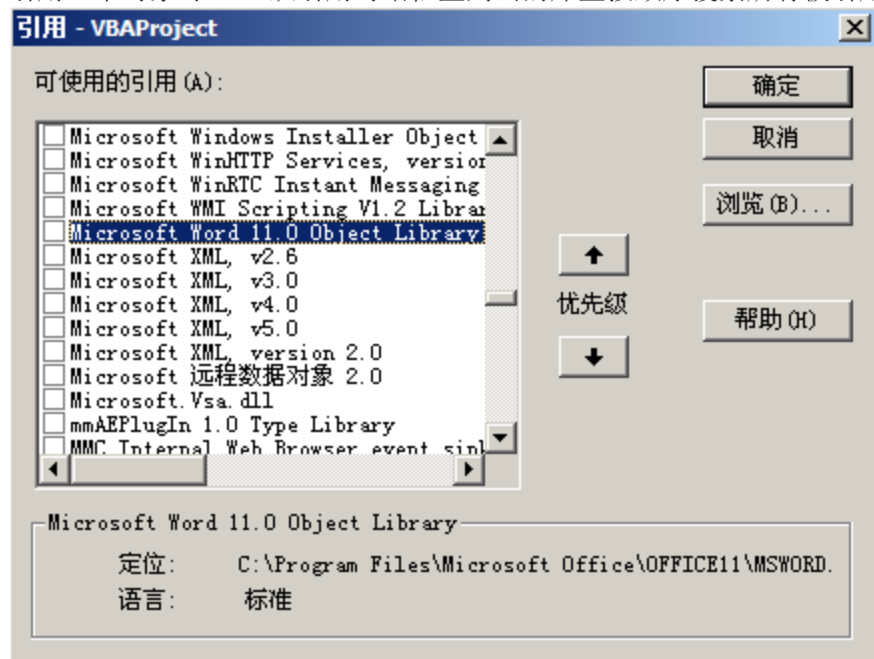


图9-3 为了要操作其它应用程序的对象，你应该建立对该需要的对象库的引用

在建立了对所要求对象库的引用后，你就可以在对象浏览器里浏览该对象的属性和方法。

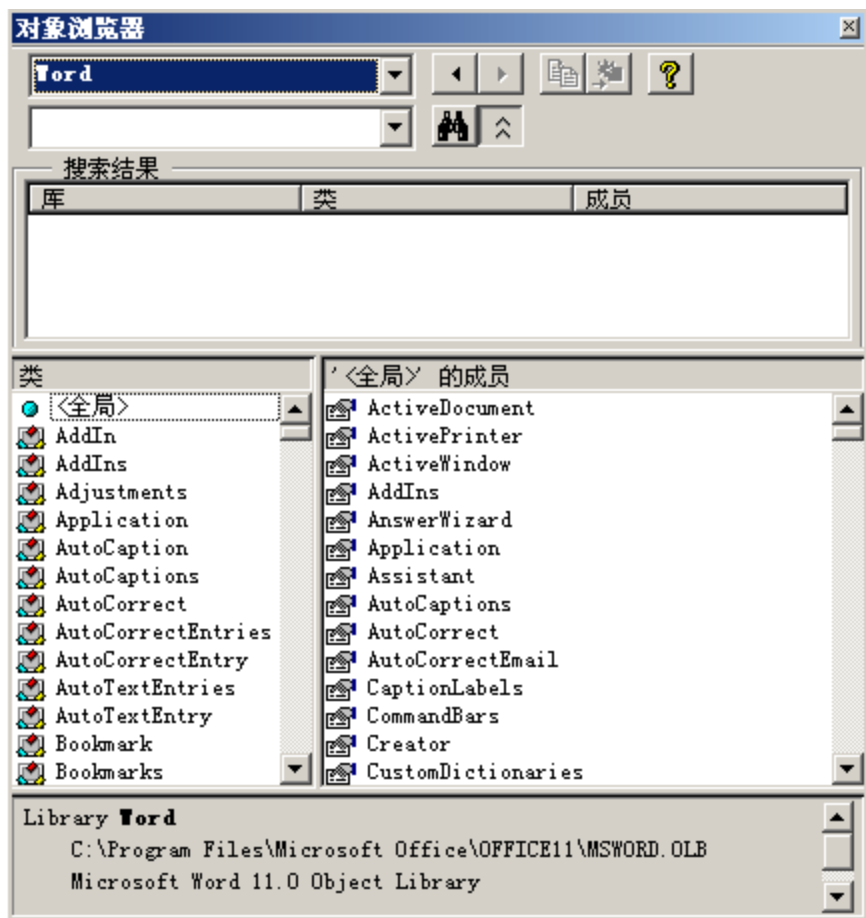


图9-4 建立了对Microsoft Word 11.0 Object Library的引用后（见图9-3），Microsoft Word的所有对象，属性和方法都可以从Excel VBA工程里访问了。

13.创建自动控制对象

依照下列步骤，在你的VBA过程里创建一个自动控制对象：

- 使用Dim...As Object或者Dim...As Application.ObjectType子句声明对象变量（参见前面章节里的后期和早期绑定使用主题）
- 如果你在使用早期绑定，那么使用引用对话框来建立对应用程序对象类型库的引用
- 如果自动控制对象不存在，那么使用CreateObject函数；如果自动控制对象已经存在的话，那么就使用GetObject函数来建立对该对象的引用
- 使用关键字Set将CreateObject或者GetObject函数返回的对象赋值到对象变量

14.使用 CreateObject 函数

按照下面的语法，使用CreateObject函数从VBA过程里创建对自动控制对象的引用：

CreateObject(class)

参数class是你想要引用的应用程序的名称。该名称包含早先讨论的对象类类型（参见早期绑定部分）。必须使用关键字Set将自动控制对象赋予对象变量，如下所示：

```
Set variable_name = CreateObject(class)
```

例如，使用自动控制对象来激活Word，你的VBA过程里需要包括下面的声明语句：

'early binding **早期绑定**

```
Dim wordAppl As Word.Document
```

```
Set wordAppl = CreateObject("Word.Application")
```

或者

'late binding 后期绑定

```
Dim wordAppl As Object
```

```
Set wordAppl = CreateObject("Word.Application")
```

通常，CreateObject函数创建该特定自动控制对象的新示例。然而，一些应用程序注册为一种叫做“单一示例”的应用程序了，这就是说你不能同时运行一个以上的示例。Microsoft Word和PowerPoint就是这种单一示例的应用程序，因此，如果Word或者PowerPoint已经在运行，那么CreateObject函数就会直接引用到在运行的示例去，而不会再创建一个新的示例。

15.使用自动控制创建一个新的 Word 文档

我们来看看你如何将前面章节学习到的关于绑定的知识应用到现实生活中的例子里。你也许有时需要从Excel直接通过程序打开一个Word文档，并且往里面写入数据，下面的例子使用了早期绑定。

1. 在当前工程里插入新模块，并重命名为Automation
2. 在工程浏览器里，选择当前工程，并且选择“工具”|“引用”
3. 如果可用引用列表里的Microsoft Word 9.0 Object Library或者Microsoft Word 10.0 Object Library没有被勾选，那么找到它们并勾选上，点击确定退出。
4. 输入下面过程WriteLetter：

```
Sub WriteLetter()
```

```
Dim wordAppl As Word.Application
```

```
Application.StatusBar = "Creating Word Application Object..."
```

```
Set wordAppl = CreateObject("Word.Application")
```

```
With wordAppl
```

```
.Visible = True
```

```
Application.StatusBar = "Creating a new document..."
```

```
.Documents.Add
```

```
.ActiveDocument.Paragraphs(1).Range.InsertBefore "Invitation"
```

```
Application.StatusBar = "Saving document..."
```

```
.ActiveDocument.SaveAs "C:\Invite.doc"
```

```
Application.StatusBar = "Exiting Word..."
```

```
.Quit
```

```
End With
```

```
Set wordAppl = Nothing
```

```
Application.StatusBar = False
```

```
End Sub
```

5. 切换到Excel应用程序窗口，并选择“工具”|“宏”|“宏”，找到过程WriteLetter并点击运行

过程WriteLetter开始时声明对象变量为特定的对象类型（Word.Application）。回想这种生命（早期报道）要求你建立对Word对象库的引用（本章的前面讲过）。CreateObject函数返回的自动控制对象赋值到一个叫做wordAppl的对象变量，因为由于子控制启动的应用程序不会出现在屏幕上，所以使用语句：

```
wordAppl.Visible = True
```

使启动的Word应用程序可见，这样你就可以观察VBA的工作情况。该过程里后面的语句打开一个新文档（Add方法），并且在第一段输入文本（InsertBefore方法），将文档保存到硬盘（SaveAs方法），以及关闭Word应用程序（Quit方法）。每条语句之前都有一条指令，将信息显示在Excel应用程序窗口下面的状态栏上。当Word应用程序关闭后，指令：

```
Set wordAppl = Nothing
```

清除对象变量，收回该对象占用的内存。语句：

```
Application.StatusBar = False
```

将状态栏上的信息恢复为默认的“就绪”。

前面提到过，Word是单一示例（single-instance）应用程序，这意味着你不能同时运行一个以上的Word示例，简单说，如果你没有启动Word，WriteLetter过程里面的CreateObject函数将会启动Word，否则，它将会使用当前活动的Word示例。

16.使用 GetObject 函数

如果你确定自动控制对象以及存在并且已经打开，那么就考虑使用GetObject函数，如下所示：

GetObject([pathname][, class])

GetObject函数有两个参数，它们都是可选的。使用第一个参数来明确你要打开的文件名称，应该提供完整的文件路径。如果你忽略该参数，那么不必明确参数class，指明要使用的对象类型，例如：

Excel.Application

Excel.Sheet

Excel.Chart

Excel.Range

Word.Application

Word.Document

PowerPoint.Application

在Invite.xls的基础上创建一个Excel对象，并且强制设置为Excel 5工作表，你可以使用下列声明：

'late binding 后期绑定

Dim excelObj As Object

Set excelObj = GetObject("C:\Invite.xls", Excel.Sheet.5)

要设定对象变量为某个特定的Word文档的话，你可以使用：

'early binding 早期绑定

Dim wordObj As Word.Application

Set wordObj = GetObject("C:\Invite.doc")

如果要访问一个正在运行的Office应用程序对象，那么可以将第一个参数空出：

Dim excelObj As Object

Set excelObj = GetObject(, "Excel.Application")

当你调用不带第一个参数的GetObject函数时，它就会返回一个对该应用程序示例的引用，如果该应用程序没有启动的话，就会产生错误。

17.打开存在的 Word 文档

下面的过程CenterText示范了GetObject函数的使用，访问Invite.doc文件。回想一下，该文件是在本章前面的过程WriteLetter里创建的。过程CenterText会将指定Word文档里的第一段居中。该过程使用了一个叫做DocExists的自定义函数来检查指定的文件是否存在。另外一个自定义函数（IsRunning）检查Word是否已经在运行。基于上述检查结果，使用CreateObject或者GetObject函数。如果出现错误，那么错误编号和错误描述将会显示出来。

Sub CenterText()

Dim wordDoc As Word.Document

Dim wordAppl As Word.Application

Dim mydoc As String

Dim myAppl As String

On Error GoTo ErrorHandler

mydoc = "C:\Invite.doc"

myAppl = "Word.Application"

'first find out whether the specified document exists 首先查明该文档是否存在

If Not DocExists(mydoc) Then

MsgBox mydoc & " does not exist." & Chr(13) & Chr(13) _

& "Run the WriteLetter procedure to create " & mydoc & "."

Exit Sub

End If

'now check if Word is running 现在检查Word是否正在运行

If Not IsRunning(myAppl) Then

MsgBox "Word is not running - will create a new instance of _

Word. "

Set wordAppl = CreateObject("Word.Application")

Set wordDoc = wordAppl.Documents.Open(mydoc)

Else

```
MsgBox "Word is running - will get the specified document. "
'bind the wordDoc variable to a specific Word document 将变量wordDoc绑定到特定的Word文档
Set wordDoc = GetObject(mydoc)
```

```
End If
```

```
'center the 1st paragraph horizontally on page 将第一段水平居中
```

```
With wordDoc.Paragraphs(1).Range
    .ParagraphFormat.Alignment = wdAlignParagraphCenter
```

```
End With
```

```
wordDoc.Application.Quit
```

```
SaveChanges:=True
```

```
Set wordDoc = Nothing
```

```
Set wordAppl = Nothing
```

```
MsgBox "The document " & mydoc & " was reformatted."
```

```
Exit Sub
```

```
ErrorHandler:
```

```
MsgBox Err.Description, vbCritical, "Error: " & Err.Number
```

```
End Sub
```

```
Function DocExists(ByVal mydoc As String) As Boolean
```

```
On Error Resume Next
```

```
If Dir(mydoc) < > "" Then
```

```
    DocExists = True
```

```
Else
```

```
    DocExists = False
```

```
End If
```

```
End Function
```

```
Function IsRunning(ByVal myAppl As String) As Boolean
```

```
Dim applRef As Object
```

```
On Error Resume Next
```

```
Set applRef = GetObject(, myAppl)
```

```
If Err.Number = 429 Then
```

```
    IsRunning = False
```

```
Else
```

```
    IsRunning = True
```

```
End If
```

```
'clear object variable 清除对象变量内容
```

```
Set applRef = Nothing
```

```
End Function
```

18.使用关键字 New

除了使用CreateObject函数来引用到其它的应用程序之外，你可以使用关键字New。关键字New告诉VB创建一个对象的新示例，返回到该示例的引用，以及将引用赋予该对象变量。例如，你可以按下面的方式使用关键字New：

```
Dim objWord As Word.Application
```

```
Set objWord = New Word.Application
```

```
Dim objAccess As Access.Application
```

```
Set objAccess = New Access.Application
```

使用关键字**New**声明的对象变量总是早期绑定的。使用关键字**New**比使用**CreateObject**函数更高效。你每次使用关键字**New**的时候，**VB**就会创建应用程序的一个新示例。如果该应用程序以及运行，你就不需要打开另外一个示例，你应该使用**GetObject**函数。关键字**New**也可以用来在声明对象变量的时候，同时创建一个新的对象示例，例如：

```
Dim objWord As New Word.Application
```

注意，当你使用关键字**New**在**Dim**语句里声明对象变量的时候，你就不需要使用**Set**语句了。然而，不建议使用这种创建对象变量的方法，因为当该对象变量真正被创建后，你就失去对它的控制了。在声明中使用关键字**New**会导致创建对象，即使它没有被使用到。因此，如果你想要控制创建的对象变量，那么总是使用下述语法声明你的对象变量吧：

```
Dim objWord As Word.Application
```

```
Set objWord = New Word.Application
```

Set语句可以进一步在你需要使用该对象的地方使用，接下来的章节将示范如何使用关键字**New**来创建**Microsoft Outlook**的新示例，并且编写你的联系地址到**Excel**工作表中。

19.使用自动控制访问 Microsoft Outlook

要从**Excel**直接访问**Outlook**的对象模型的话，首先就要建立对**Microsoft Outlook 10.0**或者**9.0 Object Library**的引用。下面的程序例子将在**Excel**工作表里插入你**Outlook**里面的联系信息。

```
Sub GetContacts()
```

```
    Dim objOut As Outlook.Application
```

```
    Dim objNspc As NameSpace
```

```
    Dim objItem As ContactItem
```

```
    Dim Headings As Variant
```

```
    Dim i As Integer ' array element 数组成员
```

```
    Dim r As Integer ' row index 行号
```

```
    r = 2
```

```
    Set objOut = New Outlook.Application
```

```
    Set objNspc = objOut.GetNamespace("MAPI")
```

```
    Headings = Array("Full Name", "Street", "City", _  
                    "State", "Zip Code", "E-Mail")
```

```
    Sheets(1).Activate
```

```
    For Each cell In Range("A1:F1")
```

```
        cell.FormulaR1C1 = Headings(i)
```

```
        i = i + 1
```

```
    Next
```

```
    For Each objItem In objNspc.DefaultFolder _
```

```
        (olFolderContacts).Items
```

```
        With ActiveSheet .Cells(r, 1).Value = objItem.FullName
```

```
            .Cells(r, 2).Value = objItem.BusinessAddress
```

```
            .Cells(r, 3).Value = objItem.BusinessAddressCity
```

```
            .Cells(r, 4).Value = objItem.BusinessAddressState
```

```
            .Cells(r, 5).Value = objItem.BusinessAddressPostalCode
```

```
            .Cells(r, 6).Value = objItem.Email1Address
```

```
        End With
```

```
        r = r + 1
```

```
    Next objItem
```

```
    Set objItem = Nothing
```

```
    Set objNspc = Nothing
```

```
    Set objOut = Nothing
```

```
End Sub
```

过程**GetContacts**开始声明一个叫做**objOut**的对象变量来存储到**Outlook**应用程序的引用，该变量定义为明确的对象类型（**Outlook.Application**），因此**VBA**使用早期绑定。

注意在该过程里，我们使用关键字**New**（在前面部分由讨论）来创建一个新的**Outlook**应用程序对象示例，返回引用到该示例，并且将该引用赋予声明的变量**objOut**。

为了访问**Outlook**里的联系项目，你也需要声明对象变量来引用**Outlook**的**NameSpace**和**ContactItem**。**NameSpace**对象代表了储存为**MAPI**（信息应用程序编程界面）的信息。**NameSpace**对象包含了文件夹（联系地址，日志，任务，等等），每个文件夹由一次有它们的项目。一个项目是**Outlook**的一个详细数据，例如邮件信息，或者联系地址。

使用**For...Each...Next**循环在工作表里写入列标题之后，过程使用另外一个**For...Each...Next**循环来遍历联系地址文件夹中的项目。**GetDefaultFolde**方法返回一个联系地址文件夹的对象变量，该方法有一个参数，该常量代表了你要访问的文件夹。当所有的联系地址都被写入**Excel**工作表后，该过程释放所有对象变量，将它们设定为**Nothing**。

注意，当你运行过程**GetContacts**时，你可能会看到一个警告信息，告诉你程序试图访问电子邮件地址，点击确定允许操作。

20.接下来……

在本章，你学习了如何从**VBA**程序里启动、激活和控制其它应用程序（**Word**和**Outlook**）。你学习了如何使用**SendKeys**方法发送按键到另一个应用程序。你也学习了如何手动和编程地添加链接和嵌入对象。最后，你使用自动控制从**Excel**里创建新的**Word**文档，以及后来访问该文档并设置一些格式。你也学习了如何从**Outlook**里获取联系地址并放置到**Excel**工作表中。你使用两个新函数**CreateObject**和**GetObject**扩展了你的**VBA**知识。你也学习了如何以及何时使用关键字**New**。请在第十五章里学习如何从**Excel**里控制**Microsoft Access**。

在下一章，你将学习如何通过自定义窗体从用户处收集更多的数据。

第十章 对话框和自定义窗体

在第四章，你学习了如何使用**Excel**内置的**InputBox**函数在**VBA**过程执行期间从用户处收集单一数据。但是，万一你的程序在运行时需要多个数据怎么办呢？用户也许希望一次就提供所有数据，或者从项目清单中作出所有合适的选择。如果你定程序必须收集数据的话，那么你可以：

- 使用内置对话框集合
- 创建一个自定义窗体

本章将教你如何从**VBA**过程里显示内置的对话框，以及从零开始设计你自己的自定义窗体。

Excel对话框

在开始创建自己的窗体之前，你应该花上一些时间学习如何利用**Excel**内置的对话框，这些内置对话框本来就是为我们准备的。我讲的不是手动选择适合的选项，而是从你自己的**VBA**过程里调用这些对话框。

Excel有一个特殊的内置对话框集合，它们用开头为**xlDialog**的常量表示，例如**xlDialogClear**，**xlDialogFont**，**xlDialogDefineName**和**xlDialogOptionsView**。这些内置对话框是**Excel**对象，属于**内置Dialogs**集合，每个**dialog**对象代表一个内置对话框。

表10-1 常用的内置对话框

对话框名称	常量
新建	xlDialogNew
打开	xlDialogOpen
另存为	xlDialogSaveAs
页面设置	xlDialogPageSetup
打印	xlDialogPrint
字体	xlDialogFont

按照下述格式使用**Show**方法来显示对话框：

`Application.Dialogs(常量).Show`

例如，下面的语句显示字体对话框：

`Application.Dialogs(xlDialogFont).Show`

如果你在对象浏览器里面选择**Excel**库后，再输入**xlDialog**搜索的话，那些代表**Excel**内置对话框的常量清单就会显示在对象浏览器里面了（参见图10-1）

1. 打开一个新工作簿并且保存为Chap10.xls
2. 切换到VB编辑器窗口
3. 打开立即窗口
4. 输入下述语句并查看结果:

```
Application.Dialogs(xlDialogClear).Show
Application.Dialogs(xlDialogFont).Show
Application.Dialogs(xlDialogFontProperties).Show
Application.Dialogs(xlDialogDefineName).Show
Application.Dialogs(xlDialogOptionsView).Show
```

最后一句指令显示“选项”对话框的“视图”。显示内置对话框后，你可以选择合适的选项，然后Excel就会将当前被选择的单元格，区域或者整个工作表设置相应的格式。

尽管你不能更改内置对话框的外观和行为，但是当你从你的VBA过程显示内置对话框的时候，你可以决定它的初始设置。如果你不更改初始设置，那么VBA将显示对话框和其缺省设置。

假设你要显示清除对话框，并且所有按钮都被选择上。通常Excel显示对话框的时候，内容选项按钮是被选择上的。在立即窗口里输入下列语句：

```
Application.Dialogs(xlDialogClear).Show 1
```

你可以在Show方法后面加上一系列的参数，在清除对话框里，“全部”选项按钮出现在四个选项按钮组的最开头。Excel通常将可用的选项进行编号，因此，“全部”=1，“格式”=2，“内容”=3，以及“批注”=4。在线帮助可以搜索到内置对话框的参数列表（参见图10-3）

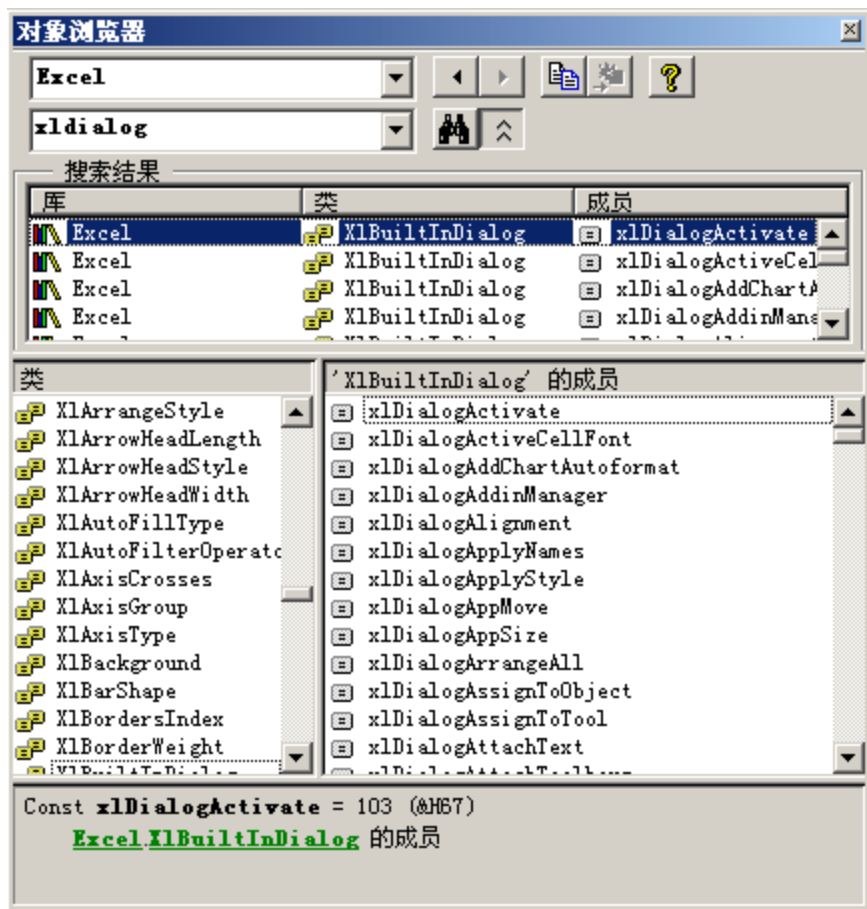


图10-1 前缀为“xlDialog”的常量识别Excel内置对话框

在立即窗口里输入下面的语句，可以显示字体对话框，并且当前选择为“Arial”字体和14字号：

```
Application.Dialogs(xlDialogFont).Show "Arial", 14
```

如果只要明确字号的话，那么可以在第一个参数的位置放置一个逗号就行：

```
Application.Dialogs(xlDialogFont).Show , 8
```

下面的指令显示“定义名称”对话框，并且在工作簿中的“名称”文本框中输入“John”，“引用位置”里引用到单元格A1：

```
Application.Dialogs(xlDialogDefineName).Show "John", "=$A$1"
```

如果你点击确定Show方法就返回True，点击取消则为False。

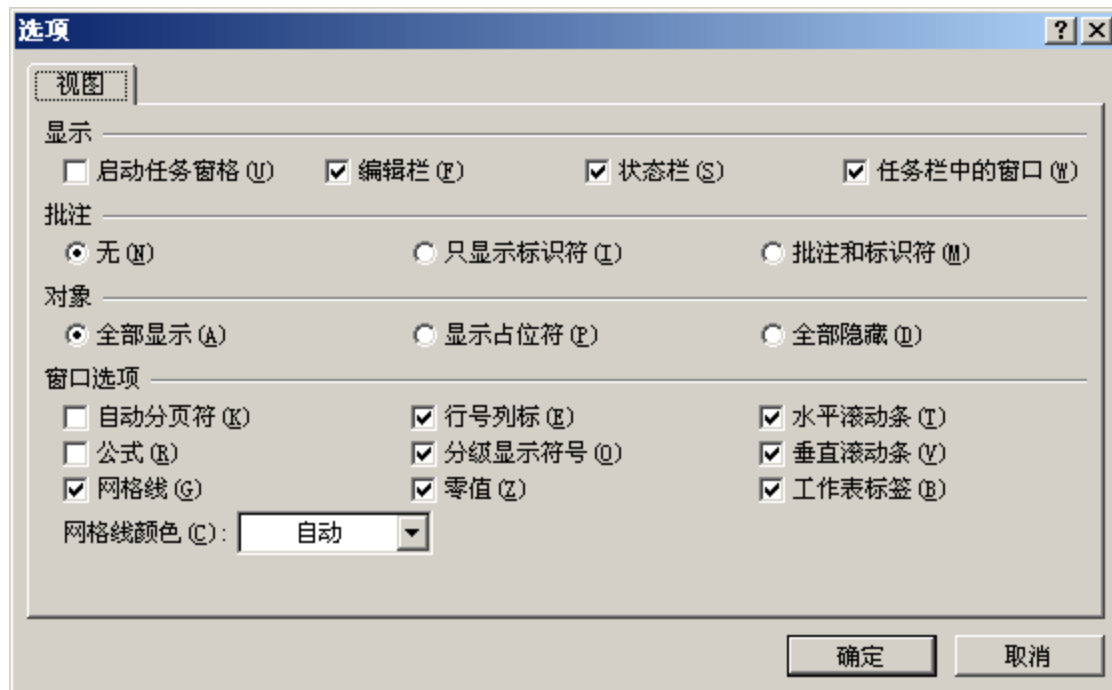


图10-2 以常量xlDialogOptionsView代表的“选项”对话框“视图”的可用设置

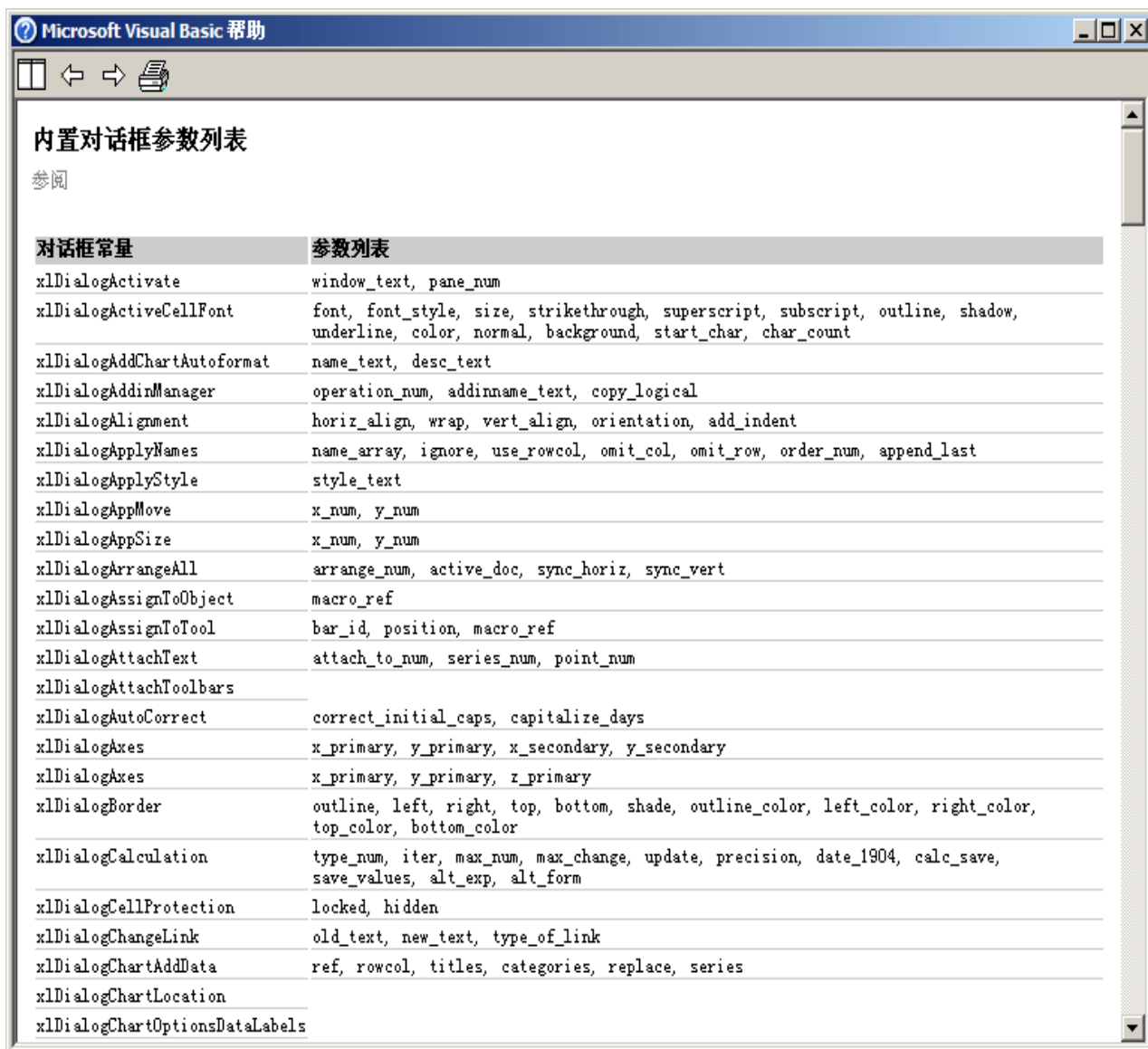


图10-3 Excel内置对话框参数列表

1. 文件打开和另存为对话框

OfficeXP中一个新而功能强大的对象是FileDialog。该对象允许你从你的VBA过程里显示文件打开和文件另存为对话框。因为FileDialog对象是Microsoft Office 10.0 Object Library的一部分，所以它在所有的Office XP应用程序里都是可用的。在前期的Excel版本中，程序员使用了两种特殊的方法来显示文件打开和文件另存对话框，这些方法（GetOpenFilename和GetSaveAsFilename）将在本章后面解释。要在你的VBA过程里面使用新的FileDialog对象来显示文件打开对话框的话，你可以输入下列语句：

```
Application.FileDialog(msoFileDialogOpen).Show
```

要显示文件另存对话框的话，则使用下面的语句：

```
Application.FileDialog(msoFileDialogSaveAs).Show
```

现在，我们在立即窗口里输入上面的语句来看看文件打开和文件另存对话框。

除了文件打开和文件另存为对话框之外，FileDialog对象也能够显示“浏览”对话框，列出文件和文件夹（参见图10-4），或者文件夹（图10-5）：

```
‘ browse the list of files and folders 浏览文件和文件夹清单
```

```
Application.FileDialog(msoFileDialogFilePicker).Show
```

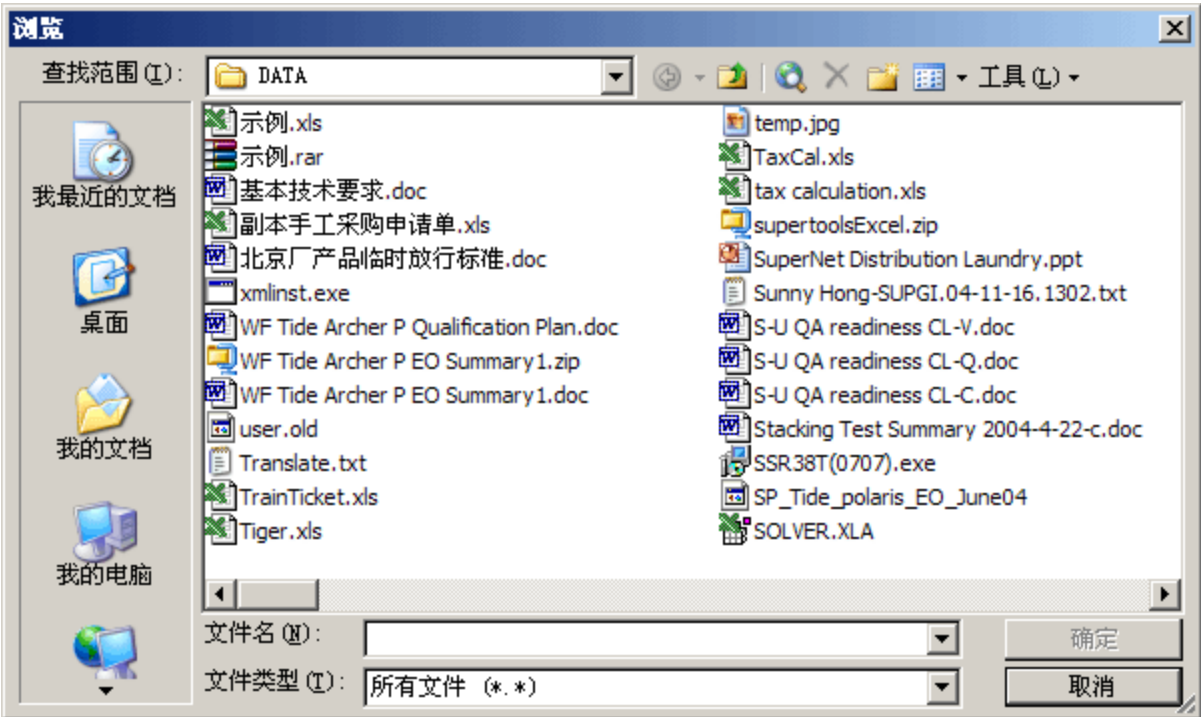


图10-4 文件采集对话框允许用户选择一个或多个文件，该对话框显示文件和文件夹列表，并且标题显示为“浏览”

‘ browse the list of folders [浏览文件夹清单](#)
Application.FileDialog(msoFileDialogFolderPicker).Show

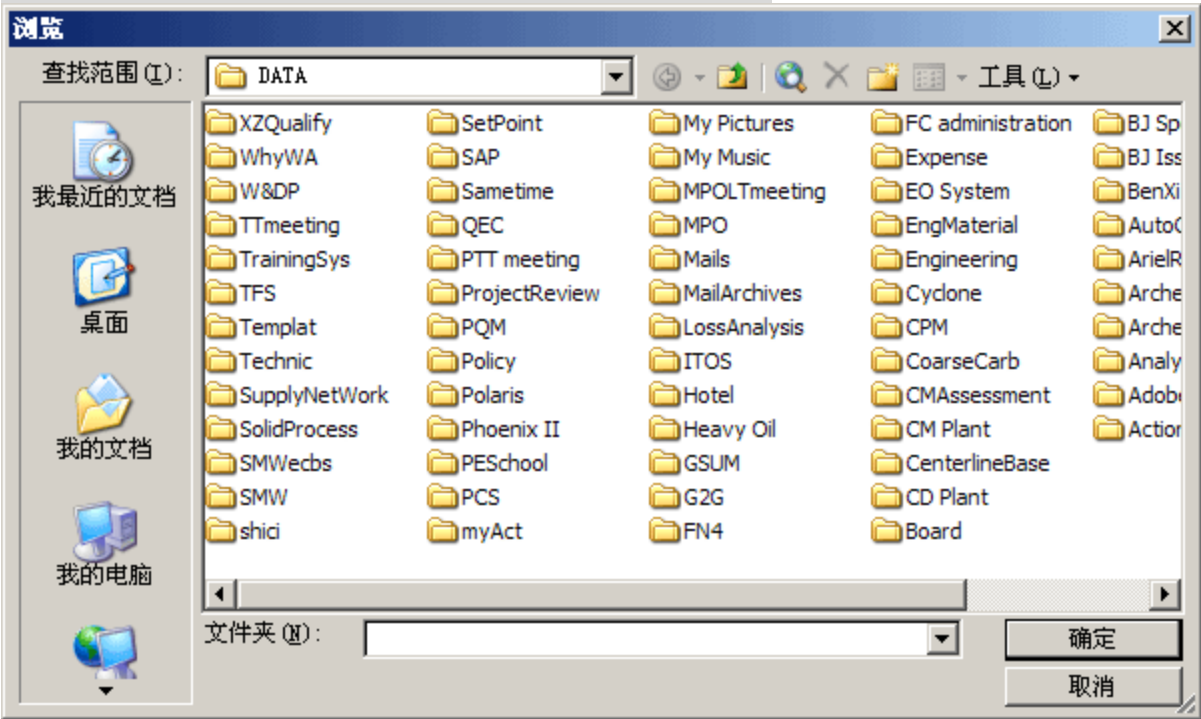


图10-5 文件夹采集对话框允许用户选择一个路径，该对话框显示目录列表，并且标题显示为“浏览”

文件对话框使用的常量列在下面的表格里，前缀“mso”表明这些常量都是Microsoft Office 对象模型里一部分。

msoFileDialog常量	值
msoFileDialogOpen	1

msoFileDialogSaveAs	2
msoFileDialogFilePicker	3
msoFileDialogFolderPicker	4

可以使用FileDialog的Filters属性来控制显示文件的类型。如果你打开文件打开对话框下面的“文件类型”下拉列表框时，你将看到许多可选择文件过滤器。那里有24种预先设置好的文件过滤器，你也可以在该清单里添加你自己的过滤器。在立即窗口里输入下述语句，我们就可以得到缺省的文件过滤器数目了：

```
set f = Application.FileDialog(msoFileDialogOpen).Filters
?f.count
```

FileDialog对象的过滤器储存在FileDialogFilters集合里面。我们来创建一个简单的过程，将缺省的文件过滤器返回到Excel工作表：

1. 在当前VBA工程里插入一个新模块，并且重命名为DialogBoxes
2. 在DialogBoxes代码窗口里输入下面显示的ListFilters过程：

```
Sub ListFilters()
    Dim fdfs As FileDialogFilters
    Dim filt As FileDialogFilter
    Dim c As Integer

    Set fdfs = Application.FileDialog(msoFileDialogOpen).Filters
    Sheets(3).Cells(1, 1).Select
    Selection.Formula = "List of Default Filters"

    With fdfs
        c = .Count
        For Each filt In fdfs
            Selection.Offset(1, 0).Formula = filt.Description & _
                ": " & filt.Extensions
            Selection.Offset(1, 0).Select
        Next
        MsgBox c & " filters were written to Sheet3."
    End With
End Sub
```

该过程声明了两个对象变量，变量fdfs返回对FileDialog对象里的FileDialogFilters集合的引用，而对象变量filt则储存对对象FileDialogFilter的引用。FileDialogFilters集合的Count属性返回文件过滤器的总数。之后，过程遍历过滤器集合，并且找到每个过滤器的描述和扩展名。

使用FileDialogFilters集合的Add方法，你可以轻易地将你自己的过滤器添加到缺省的过滤器中去。下面修改后代工程ListFilters2示范了如何将临时文件 (*.tmp) 过滤器添加到过滤器清单中去。该过程里的最后语句将打开文件打开对话框，因此你自己可以检查自定义的过滤器是否已经被添加到了文件类型下拉列表框里。

```
Sub ListFilters2()
    Dim fdfs As FileDialogFilters
    Dim filt As FileDialogFilter
    Dim c As Integer

    Set fdfs = Application.FileDialog(msoFileDialogOpen).Filters
    Sheets(3).Cells(1, 1).Select
    Selection.Formula = "List of Default Filters"

    With fdfs
        c = .Count
        For Each filt In fdfs
```



```

        Selection.Offset(1, 0).Formula = filt.Description & _
            ": " & filt.Extensions
        Selection.Offset(1, 0).Select
    Next

```

```

    MsgBox c & " filters were written to Sheet3."
    .Add "Temporary Files", "*.tmp", 1
    c = .Count
    MsgBox "There are now " & c & " filters." & vbCrLf _
        & "Check for yourself."
    Application.FileDialog(msoFileDialogOpen).Show
End With

```

```
End Sub
```

你可以使用FileDialogFilters集合的Clear方法清除所有预设的过滤器。修改一下上面的过程，在添加自定义的临时文件 (*.tmp) 过滤器之前，清除内置的过滤器。

当你从文件打开对话框里选择一个文件时，该被选择的文件名称和路径就会被放置在FileDialogSelectedItems集合里。使用SelectedItems属性可以返回FileDialogSelectedItems集合。通过设定FileDialog对象的AllowMultiSelect属性为True，用户就可以同时按下Shift键或者Ctrl键和文件名称，选择一个或多个文件。

接下来的过程示范了如何使用上面提及的属性，该过程打开一个新的工作簿并且插入一个列表框控件。允许用户选择一个以上的文件，然后被选择的文件将加入到该列表框控件里，并且加亮第一个文件名。

```

Sub ListSelectedFiles()
    Dim fd As FileDialog
    Dim myFile As Variant
    Dim lbox As Object

    Set fd = Application.FileDialog(msoFileDialogOpen)

    With fd
        .AllowMultiSelect = True
        If .Show Then
            Workbooks.Add

            Set lbox = Worksheets(1).Shapes. _
                AddFormControl(xlListBox, _
                    Left:=20, Top:=60, Height:=40, Width:=300)
            lbox.ControlFormat.MultiSelect = xlNone

            For Each myFile In .SelectedItems
                lbox.ControlFormat.AddItem myFile
            Next

            Range("B4").Formula = _
                "You've selected the following " & _
                lbox.ControlFormat.ListCount & " files:"
            lbox.ControlFormat.ListIndex = 1
        End If
    End With
End Sub

```

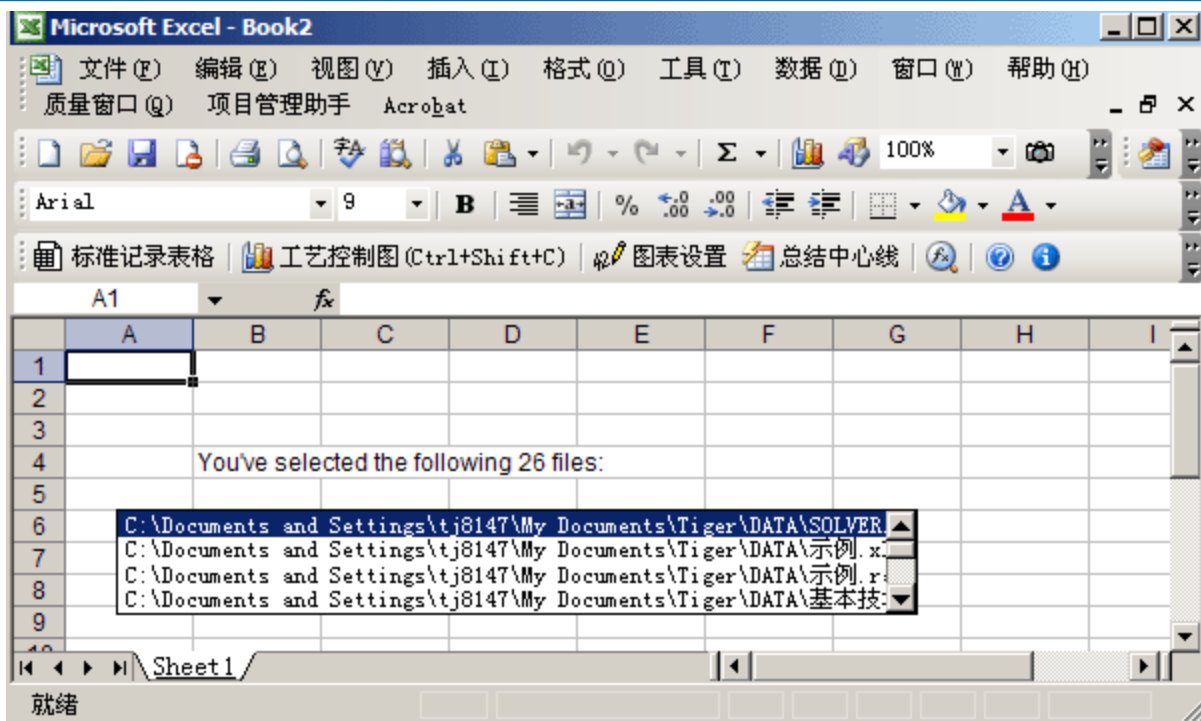


图10-6 使用过程ListSelectedFiles（见上面）将用户选择的文件添加到工作表中列表框控件中去

注意，Show方法不会将用户所选的文件打开，它仅仅显示文件打开对话框。当用户点击“打开”按钮时，文件名称通过SelectedItems属性从SelectedItems集合里获得。如果你希望用户点击“打开”按钮时立即执行文件的打开操作的话，你就应该使用FileDialog对象的Execute方法。下面的过程示范了如何立即打开用户选择的文件：

```
Sub OpenRightAway()
    Dim fd As FileDialog
    Dim myFile As Variant

    Set fd = Application.FileDialog(msoFileDialogOpen)

    With fd
        .AllowMultiSelect = True
        If .Show Then
            For Each myFile In .SelectedItems
                .Execute
            Next
        End If
    End With
End Sub
```

2.GetOpenFilename 和 GetSaveAsFilename 方法

从多年以前开始，Excel就给程序员们提供了两种方便的VBA方法来显示文件另存为和文件打开对话框：GetOpenFilename和GetSaveAsFilename。这些方法只有在Excel里可用，并且在Excel2002里面如果需要向后兼容的话仍然可用。GetOpenFilename方法显示“打开”对话框，在那里你可以选择要打开的文件名称，第二个方法（GetSaveAsFilename）则显示另存为对话框。

1. 在立即窗口输入下面的指令：

```
Application.GetOpenFilename
Application.GetSaveAsFilename
```

```
Application.GetSaveAsFilename ("Plan2.xls")
```

GetOpenFilename方法从用户处获得文件名称，而不必实际打开某特定的文件。该方法有四个可选的参数，经常使用的是第一和第三个参数，显示入下表：

GetOpenFilename参数	描述
fileFilter	该参数决定了对话框的文件类型（译者：原文为Save as type，有误）下拉框了的内容。例如，要在文件类型下拉框里显示“Excel Files(*.xls)”的话，你就应该输入下列文本作为fileFilter：“Excel Files(*.xls), *.xls”（译者：“Excel Files, *.xls”也一样。）过滤器的前面部分（逗号前）决定文件类型下拉框要显示的文本，第二部分（逗号后）明确你要显示的那种类型的文件。确保你按照表格里的例子试验一下。
title	这是对话框的标题，如果忽略，对话框将显示标题为“打开”

在立即窗口里输入下列语句（确保在一行输入），来看看这些参数是如何使用的：

```
Application.GetOpenFilename("Excel Files(*.xls), *.xls"), "Highlight the File"
```

GetOpenFilename方法返回所选的或者指定的文件名称，该名称之后可以在你的VBA过程里用来打开该文件，例如：

```
yourFile = Application.GetOpenFilename  
?yourFile  
C:\EXCEL\Mark.xls  
Workbooks.Open Filename:=yourFile
```

在上面的例子里，文件名称被赋予变量yourFile，接下来的两条输入为询问文件名称（?yourFile）和显示该名称（C:\EXCEL\Mark.xls）。第四条语句打开变量yourFile明确的文件。如果你通过点击Esc键或者对话框上的取消按钮来取消对话框的话，那么GetOpenFilename方法就会返回False。

GetSaveAsFilename方法返回文件名和路径，然而，它不会自动地保存该特定的文件。输入下述指令提供文件名称：

```
Application.GetSaveAsFilename ("Plan2.xls")
```

如果你忽略文件名称的话，Excel就会显示当前活动文件的名称。当你使用GetSaveAsFilename方法时，你可以明确文件过滤器和对话框自定义标题：

```
yourFile = Application.GetSaveAsFilename("Plan2.xls", "Excel Files(*.xls), *.xls", "Name your file")  
要显示另存为对话框的话，就要将GetSaveAsFilename方法的结果赋予一个变量，如上所示。
```

3.创建窗体

尽管内置的对话框很方便使用，但是它并不能满足你所有的VBA应用程序的要求。除了将对话框显示在屏幕上和初始化它的设置之外，你不能控制对话框的外观，你不能决定增加哪个按钮，删除哪个按钮，而哪个又移动，同样，你不能改变内置对话框的大小。如果你想用提供一个自定义的界面的话，那么你的唯一办法就是创建一个用户窗体。

用户窗体看上去就像一个自定义对话框，你可以在上面添加各种各样的控件，给这些控件设置属性以及编写对窗体反应的VBA过程和控制事件。窗体是单独的对象，你可以在VB编辑器菜单里选择“插入”|“用户窗体”来添加窗体。

窗体可以在不同的应用程序之间分享使用，例如，你可以在Word或者任何其它使用VB编辑器的应用程序里面，重新使用Excel里设计的窗体。

按照下述步骤创建自定义窗体：

- 切换到VB编辑器窗口
- 选择“插入”|“用户窗体”

一个叫做窗体的文件夹显示在工程浏览器窗口，该文件夹包含一个空白用户窗体。工作区域自动显示窗体和带有添加控件的工具箱。

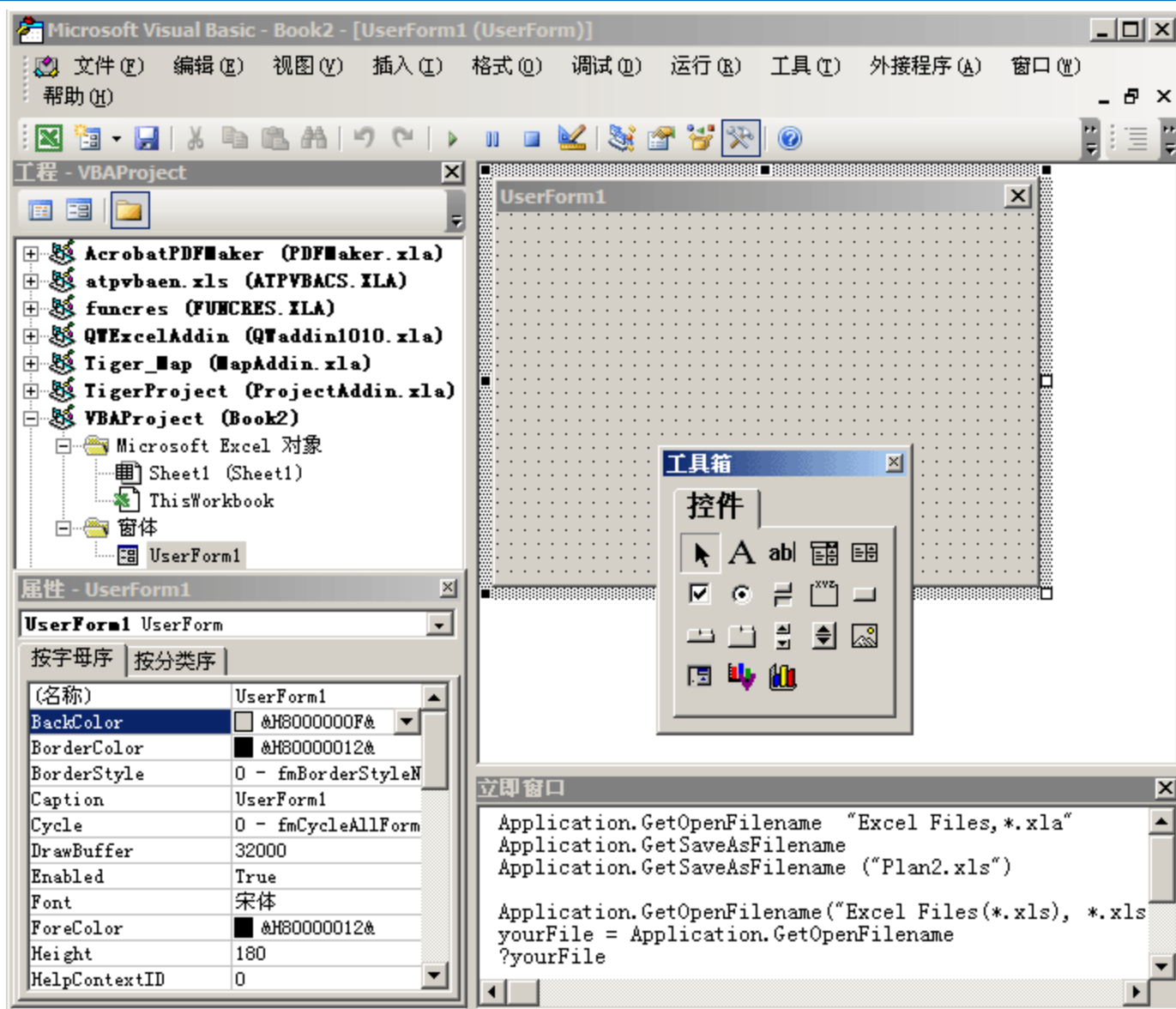


图10-7 通过选择插入菜单里面的用户窗体可以添加一个新的窗体到活动VBA工程

属性窗口（见图10-8）显示很多种属性，你可以根据你的需要进行设置。该窗体属性安排成七个类别：外观，行为，字体，杂项，图片，位置和滚动。要按类别显示窗体属性的话，那么可以点击属性窗口上的“按分类序”。要查找某特定属性的信息的话，可以点击该属性名称并且按下F1键，在线帮助将启动该属性描述主题。在你的VBA工程里添加新窗体后，你应该通过设置“名称”（Name）属性给该窗体设置一个独特的名称，除了名称之外，每个窗体还应该包含一个标题，你可以使用Caption属性给你的窗体设置标题。

技巧10-1 在VBA应用程序之间共享窗体

所有使用VB编辑器的VBA应用程序在创建自定义窗体时共享一些功能特点，你可以通过从文件中导出和导入，或者通过拖曳窗体对象到另一个工程来共享你的窗体。要导入或者导出窗体文件，可以选择“文件”|“导入”或者“文件”|“导出”。在你导出某个窗体文件之前，确保你在工程浏览器窗口里选中了该窗体。在将窗体拖曳到一个不同的VBA应用程序之前，你得安排好VBE窗口，确保你能同时看到两个应用程序的工程浏览器窗口，拖住工程浏览器里的窗体名称到另外一个工程浏览器。

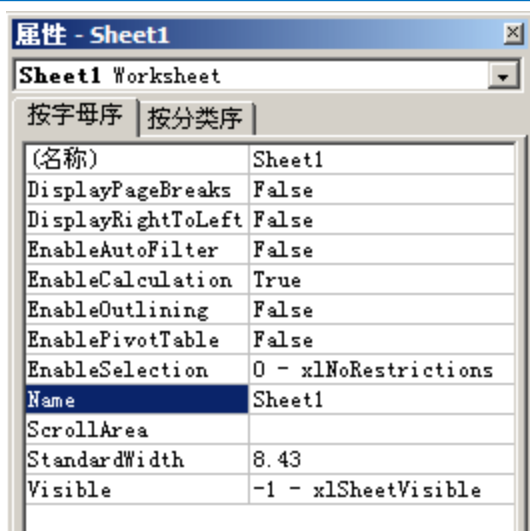


图10-8 使用属性窗口，你可以很容易地更改你自定义窗体的外观，行为和其它特点

4.创建用户窗体的工具

当你设计一个窗体时，你可能要插入一些合适的控件来使它有用。工具箱包含了标准VB所有你可以添加到窗体上的控件按钮，它也可以包含已经安装在你电脑上的额外的控件。工具箱可用的控件被称为ActiveX控件，这些控件能够对特定的用户行为例如点击该控件或改变它的制作出反应。你将在本章剩余的部分学习如何使用工具箱控件。

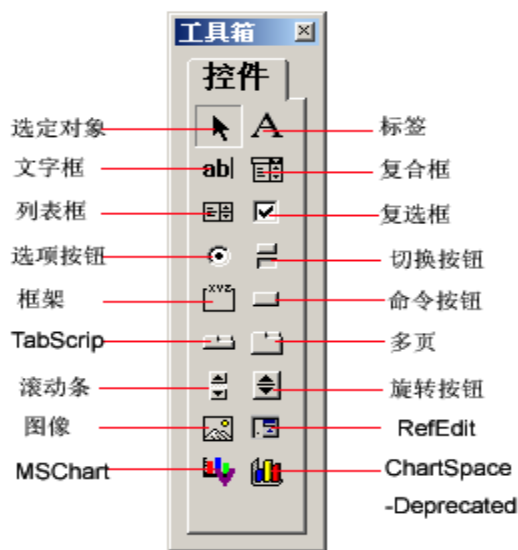


图10-9 工具箱显示了可以添加到你的自定义窗体的控件（译者：在每台电脑上的工具箱上显示控件可能不一样，可以通过“附加控件”在工具箱上添加额外的控件按钮）

Microsoft Office套装提供了额外可以放置在工具箱上以快速访问的ActiveX控件，如果你的电脑上还安装了其它包含ActiveX应用程序的话，那么你也可以将它们放置在工具箱上。依照下述步骤在工具箱上添加其它的ActiveX控件：

1. 在工具箱上（非控件区域）单击右键并且点击“新建页”，并且选择重命名
2. 在“题注”框里面输入新名称“额外控件”，在“控件提示文字”框里面输入“额外的ActiveX控件”，点击确定返回到工具箱
3. 在新页的任意地方单击右键，并从快捷菜单中选择“附加控件”。如果该选项不可用的话，那么确定你在该页区域单击右键，而不是点击“额外控件”本身
4. 当附加控件对话框出现时，点击你要添加的每个控件前面的选项按钮。图10-10显示了加亮的日历控件。当你点击确定时，

这个控件就会出现在工具箱的当前页上。



图10-10 你在工具箱上附加额外安装在你电脑上的ActiveX控件

在接下来的几段中将描述标准的VB控件。

5. 标签

标签允许你在窗体上添加文本，标签控件经常用来添加字幕，标题，抬头和解释。你可以使用标签给那些不带Caption属性的控件（例如文字框，列表框，滚动条和旋转按钮）加上标题。你可以给标签定义快捷键，例如，在添加完标签控件和设置它Accelerator属性后，通过按下Alt键和某个特定字母键，你就可以立即激活该控件（译者：激活标签控件后面的控件，例如文字框，列表框等。这是一个非常有用的工具！）。要给某个已经存在的控件添加一个标题或者一个键盘快捷键的话，你就应该添加一个标签控件，并且在它属性窗口的Accelerator属性里输入一个字母。下一步选择“视图”|“Tab键顺序”，并且确保该标签名称出现在你要使用快捷键激活的控件名称之前。你将在本章后面学习如何使用Tab键顺序对话框（参见图10-14）

6. 文字框

文字框是最流行的窗体控件，因为它们可以用来显示或者从用户处获取数据。你可以在文字框里输入文本、数字、单元格引用或者公式。通过更改MultiLine属性设置，你可以在文字框里输入一行以上的文本。当你设置了WordWrap属性时，文本行可以自动换行。此外，如果你将EnterKeyBehavior属性设置为True，同时MultiLine属性也为真的话，那么你将能够在文字框里面通过按下回车键开始新的一行。另外一个属性EnterFieldBehavior，决定当用户选择文本区域时是否选择文本（译者：应该说激活该控件，例如说使用Tab键来激活文字框。），设置该属性为0（fmEnterFieldBehaviorSelectAll）的话，将会选择该区域的所有文本，设置该属性为1（fmEnterFieldBehaviorRecallSelection）的话，将仅仅选择用户上次激活该区域时选择的文本。如果你想要限制用户在文字框输入字符数的话，那么你可以设置MaxLength属性的字符确切数目。

7. 框架

框架允许你可视地组织和逻辑地聚合窗体上面的各种控件。当你使用框架控件包围选项按钮时，VB将这些按钮视为相互排斥的，你同时只能选择其中的一个按钮，因此，如果用户选择了可用的选项按钮之一i，那么其它的选项按钮就不能再被选择。在本章的后面，你将找到使用两个框架的“信息调查”窗体示例。其中一个将“硬件”和“软件”选项按钮组织成一个逻辑化的集合，而第二个框架则将和电脑类型相关的复选框集中到一起（参见图10-11）。

8. 选项按钮

选项按钮让你在许多相互排斥的选项中选择一个，选项按钮通常以两个或者多个按钮组织在一起并且包围在一个框架之内。在任何时刻，只能选择一个选项按钮。当你选择一个新选项按钮，之前被选择的选项按钮就会自动取消选定。要激活或者失活一个选项按钮的话，只要将它的Value属性设置为True或者False。True意味着该选项被激活，False则意味着该选项失活。

9. 复选框

复选框用来将具体的选项打开或者关闭，不同于选项按钮只让你同时选择一个选项，用户可以同时选上一个或者多个复选框。如果复选框被选择，那么它的Value属性就设置为True，如果复选框没有被选择，那么它的Value属性则设置为False。

10. 切换按钮

切换按钮看上去像命令按钮，而作用与选项按钮类似。当你点击切换按钮时，该按钮会保持凹下去，如果再点击一次的话，按钮则恢复正常。按下的切换按钮的值属性为True。

11. 列表框

除了可以使用文字框提示用户输入明确的数据之外，有时候提供一个可供选择的清单可能会更好，列表框排除了输入错误数据的可能性。列表框的内容可以在工作表中敲入，也可以直接从VBA过程里使用AddItem方法添加。RowSource属性指定显示在列表框里面的数据源，例如引用\$A\$1:\$B\$8，列表框里将显示这些区域里的内容。

当你设置ColumnCount属性时，列表框可以显示一行或者多列数据。另外一个属性，ColumnHeads，可以设置为True，显示列表框的列标题。用户也没有限制只选择一个选项，如果过程需要选择两个或者多个列表项目的话，你也可以设置MultiSelect属性为True。

12. 复合框

复合框是一个结合文字框和列表框在一起的控件，该控件经常用来节省窗体上的空间。当用户点击复合框右边的下拉箭头时，它会打开显示一系列可供选择的项目。如果里面没有一个适用的选项的话，你可以将MatchRequired属性设置为False，这样就允许用户直接输入一个新的数据。ListRows属性决定了下拉清单出现时，显示的项目数。Style属性决定了复合框的类型。使用0（fmStyleDropDownCombo）可以允许用户从清单里选择一项或者在文字框里输入一新项。如果要限制用户的选择只能在该复合框的可用清单里的话，那么就将Style属性设置为2（fmStyleDropDownList）。

13. 滚动条

你可以在窗体上放置水平的或者竖直的滚动条。尽管滚动条通常使用于定位窗口，但是它也可以在你窗体上来输入一些预设范围的数值。滚动条的当前值由Value属性设置或者返回。滚动条的Max属性让你设置它的最大值，而Min属性则决定了它的最小值。LargeChange属性决定了当用户点击滚动条内部时Value属性的改变值。同样，当使用滚动条编程时，不要忘记设置SmallChange属性，它决定当你点击滚动条的箭头时Value属性如何改变。

14. 旋转按钮

旋转按钮作用类似于滚动条。你可以点击旋转按钮来增加或者减小某个数值。旋转按钮经常和文字框一起使用，因为这样用户就可以使用旋转按钮在文字框里敲入精确的值或者选择数值。和文字框一起使用旋转按钮的技术将会在本章后面讨论到。

15. 图像

图像控件让你可以在窗体上显示图像，该控件支持下列文件格式：*.bmp，*.cur，*.gif，*.ico，*.jpg和*.wmf。就像工具箱里的其它控件一样，图像控件也有许多属性可以设定。例如，你可以使用PictureSizeMode属性控制图片的外观，该属性有三

种设置：0（fmPictureSizeModeClip将不在图片框里面的部分截除），1（fmPictureSizeModeStretch水平或竖直拉伸图片，使之正好适合图片框）和3（fmPictureSizeModeZoom按比例放大图片）。

16.多页控件

多页控件可以在窗体顶部显示一系列的页面（参见图10-17）。每小页作为单独的页面使用。使用多页控件，你可以设置包含两页或多页的窗体，你可以在每页上放置不同的控件。当一个窗体包含很多的数据时，它的可读性便降低了。点击窗体页要比在一个使用滚动条的长窗体上移动要轻松的多。默认上，每个多页控件在窗体上显示两页，通过快捷菜单可以添加新页，也可以在VBA过程里使用Add方法添加新页。本章中的第二个实践练习时犯了如何使用该控件来追踪学生的考试分数。

17.TabStrip 控件

虽然TabStrip和多页控件看上去非常相似，但是它们有各自不同的作用。TabStrip控件（参见图10-17）允许你使用相同的控件来显示多套相同的数据。假设窗体显示学生的考试，每个学生必须通过相同科目的考试。每个科目可以放在一个单独的页上（tab），每页包含相同的控件来收集得分和考试日期。当你激活任何页的时候，你将看到相同的控件，只有这些控件里的数据变化。参见本章的第二个实践练习，看看如何使用TabStrip控件。

18.RefEdit 控件

RefEdit控件是专门在Excel里面创建的窗体控件，它允许你在工作表里选择一个单元格或者单元格区域并且传递到你的VBA过程。看一下一些Excel内置对话框，你就可以看到这个控件是如何工作的，例如，“数据”菜单里的“合并计算”对话框就有一个标为“引用位置”的RefEdit控件，让你选定想要进行合并计算的数据区域。点击RefEdit右边的按钮，可以在选择单元格区域的时候暂时隐藏对话框。本章的第二个实践练习使用RefEdit控件给列表框添加学生姓名。

19.在窗体上放置控件

当你创建自定义窗体的时候，你将工具箱里可用的很多控件（参见图10-9）放置在一个空窗体上。选择什么样的控件取决于控件需要储存的数据类型，以及你窗体的功能。当你使用窗体时，工具箱总是可见的，你可以在屏幕上移动它，改变它的大小，或者当你将所有需要的控件放在窗体上了并且你要做的只是设置它们的属性，你也可以关闭它。临时被移除的工具箱也可以通过选择“视图”|“工具箱”重新显示。

工具箱的使用是很容易的，要在窗体上添加新控件的话，可以先点击工具箱上面的控件图标，然后在窗体上点击一下或者画一个框。在窗体上点击一下（不画框）将会在窗体上放置一个缺省大小的控件。每个控件的标准设置可以在它的属性窗口里查找到。例如，标准的文字框大小为18X72磅（参见文字框的Height和Width属性）。在窗体上放置控件后，“选定对象”按钮（用箭头代表）成为工具箱上的活动控件。如果你双击工具箱上的控件时，你可以随你需要画上很多这个控件。例如，要快速在窗体上放置三个文字框，可以双击文字框控件，然后在窗体上点击三次。点击工具箱上的选定对象按钮，可以失活所选的控件。

技巧10-2 设置网格选项

当你在窗体上拖曳控件时，VB将调节控件以使得它和窗体的网格对齐。通过使用“选项”对话框你可以按你的喜好设置窗体的网格。要访问网格选项的话，可以选择“工具”|“选项”，然后点击选项对话框的“通用”页，窗体网格设置区域允许你关闭网格、调整网格大小，以及决定是否需要控件和网格对齐。

20.应用程序示例 1：信息调查

既然你已经通读了创建用户窗体的理论知识，并且了解了工具箱上不同控件之间的区别，你已经可以来做一些实践练习了。你可能也知道，理解一个复杂特征的最好方式就是将它应用到一个实际生活的项目中。在这部分，你将给合作者创建一个自定义窗体，他要求你将给工作表输入调查数据的单调过程简单化。使用该窗体时（参见图10-11），你将有机会体验许多控件和它们的属性，另外，你也将学习如何将数据从你的窗体转移到工作表（参见图10-12）

在本章结束的时候，你将拥有创建自定义窗体的必要技能，适应你VBA应用程序独特的要求。

1. 在工程浏览窗口，选中工程VBAProject (Chap10. xls) 并且在属性窗口将工程名称改为CustomForms
2. 选择“插入”|“用户窗口”，添加一个空白窗体
3. 在属性窗口，双击Name属性并输入InfoSurvey，将窗体的缺省名称（UserForm1）更改掉。你将在VBA过程里使用给名称引用到该用户窗体
4. 双击Caption属性，并输入窗体新标题：Info Survey。该名称将出现在窗体的标题栏上
5. 双击BackColor属性，点击“调色板”并且给窗体底色选上一种颜色

图10-11 Info Survey窗体允许用户通过作一些适当的选择就可以快速地数据数据

	Description	Hardware	Software	IBM Compatible	Notebook/Laptop	Macintosh	Where Used	Percent (%) Used	Male	Female
3	Network	*		*			work	0	*	
4	Monitor	*			*		school	100	*	
5	Joystick	*		*			home	55		
6	Desktop Publishing	*			*		school	45		*
7	Accounting Programs	*	*	*			work/home/school	75		

图10-12 每次使用窗体Survey后，用户的选择就会写入到工作表里面

21.在窗体上添加按钮、选项框和其它控件

给自定义窗体设置完初始属性（Name和Caption）后，我们继续来给窗体放置需要的控件吧。这里是一步一步的指导如何准备如图10-11显示的窗体。

1. 更改窗体大小

当你在工程里插入的缺省窗体太小，不够放置你VBA程序要求的控件时，你可以按照下述方法之一来更改它的大小：

- a. 使用鼠标调整大小
 - 点击窗体的空白部分，窗体周围便会出现好几个选用符
 - 将鼠标放在窗体右边中间的选用符上，并且将其向右拖曳至你想要的位置，释放鼠标
 - 将鼠标放在窗体下边中间的选用符上，并且将其向下拖曳至你想要的位置，释放鼠标
- b. 通过属性窗口调整窗体大小：

每个新建的窗体缺省大小为180 X 240。窗体尺寸单位是磅。一磅等于1/72英寸。输入窗体两个属性：Height和Width的新数值，可以改变窗体的大小。

- 点击窗体的标题栏（显示“Info Survey”的地方）
- 在属性窗口，双击属性Height并且输入值252.75，更改Width属性为405.75

为了避免重复工作，总是在添加需要的控件之前调整窗体的大小。

2. 添加框架

- 点击工具箱上的框架控件，这时鼠标光标变成了一个十字架并且跟随着被选择控件的标志
- 指向窗体的左上角，然后点击并拖曳鼠标画出一个矩形。当你释放鼠标后，你将看到一个标题为“Frame1”的小矩形。当该框架被选择上后，它旁边就会出现一些选用符，并且属性窗口的标题栏便会显示“属性-Frame1”
- 在属性窗口，双击Caption属性并将默认的标题Frame1改为“Main Interest”

3. 添加选项按钮

- 点击工具箱上的选项按钮，将鼠标移动到你刚才在窗体上添加的框架“Main Interest”内部，点击并且向右拖曳鼠标，直到看到一个带有标签“OptionButton1”的矩形
 - 在属性窗口将该选项按钮的Caption属性改为“Hardware”
 - 使用相同的技术，在“Main Interest”框架里添加另外一个选项按钮并且将Caption属性改为Software
- 无论何时当用户必须从一组相互排斥的选项中选择一个时就要使用选项按钮，如果用户必须选择多余一个的选项的话，就要使用复选框。

4. 添加列表框

- 点击工具箱上的列表框控件，这时鼠标光标变成了一个十字架并且跟随着被选择控件的标志。在Main Interest框架下面点击并且向下向右拖曳鼠标画出一个列表框，当你释放鼠标后，将看到一个白的矩形。

图10-11显示了添加了各种硬件的列表框。在本实践工程的后面，你将学习如何在该列表框里面显示合适的项目。

5. 添加带选项按钮的框架

- 按照第二步在列表框下面插入一个框架并且按照第四步将框架的Caption属性改为Gender。在该框架里面添加两个选项按钮，并且将第一个按钮的Caption属性改为Male，第二个为Female。参见图10-11

技巧10-3 操作和移动控件

如果你想要复制控件的话，那么就选择该控件（被选择的控件将会有选用符在其周围），按住Ctrl键，将光标置于控件中央然后按下鼠标左键，拖曳光标到你需要的位置，然后释放鼠标，更改按钮的Caption属性。

在选择和移动一组控件的话，点击工具箱上面的“选定对象”工具，并且在你需要移动的一组控件周围画一个矩形框，当你释放鼠标后，所有控件都将被选择上。（你也可以通过按住Shift键，并点击每个要选择的控件来选择一个以上的控件——不要只看，动手试试）要移动被选择的控件组到窗体上的另外位置的话，可以点击选择区域并拖曳鼠标到预期位置。

6. 添加带复选框的框架

- 点击工具箱上的框架控件，并且在Main Interest框架右边画一个矩形
- 将其Caption属性改为Computer Type
- 点击工具箱上的复选框按钮，并且在刚添加的框架内部点击一下，框架里应该出现CheckBox1控件
- 更改CheckBox1的Caption属性为IBM/Compatible

- 在Computer Type框架里面再放置两个复选框，使用Caption属性将这两个复选框标题设置为：Notebook/Laptop 和Macintosh。最后的结果应该和图10-11一致。

不像选项按钮那样相互排斥，复选框允许用户同时激活一个或者多个选项。复选框在特定时候可以是选定的，未选定的或者不可用的。不可用的复选框会变灰并且不能被选定。选定的复选框的标题前面有一个x号，具有焦点的复选框的标题周围有虚线包围。

技巧10-4 复选框还是选项按钮

同时只能选择一个选项的时候使用选项按钮，而复选框让用户选择任意多个适合的选项。

7. 添加标签和复合框

- 点击工具箱上的标签控件
- 点击Computer Type框架下面的空白地方，Label1控件应该就会出现在那里
- 将Label1的Caption属性改为Where Used
- 点击工具箱上的复合框控件
- 点击标签Where Used下面的空白地方并且拖曳鼠标画出一个长方形，释放鼠标。

只有当你点击控件右边的向下箭头时，复合框才会显示可用的选项清单。复合框有时也被称为下来列表并且用来屏幕上的宝贵空间。尽管用户一时只能看见清单中的一项，但是通常点击向下箭头可以很快地改变当前选择。

8. 添加标签、文字框和旋转按钮

- 点击工具箱上的标签控件
- 在Where Used复合框下面的空白地方点击一下，将会出现一个标签控件，更改其Caption属性为Percent (%) Used
- 点击工具箱上的文字框控件
- 在Percent (%) Used标签右边点击一下放置一个默认大小的文字框
- 点击工具箱上的旋转控件，然后点击文字框控件的右侧，出现一个默认大小的旋转按钮。最后的样子如图10-11所示

旋转按钮有两个箭头，用来在一个给定范围内增加或者减少数值。最大值由Max属性的设置决定，而最小值则由Min属性设定。旋转按钮和滚动条有相同的属性，但是也有两个不同之处：旋转按钮没有滚动条，而且它少一个LargeChange属性。文字框通常放置在旋转按钮的旁边，这样允许用户直接在文字框里面输入数据，或者使用旋转按钮控制数值。如果旋转按钮必须和文字框一起使用的话，那么你定VBA过程必须确保文字框里输入的数据和旋转按钮保持同步。在本练习里，你将使用旋转按钮来显示所选硬件或者软件的兴趣百分比。

9. 添加命令按钮

- 双击工具箱上的命令按钮控件。回想一下双击工具箱上的控件时，表明你想要添加一个以上的所选控件
- 点击窗体的右上角，这将导致出现CommandButton1
- 点击CommandButton1的下面，出现CommandButton2
- 将CommandButton1的Caption属性为OK，CommandButton2为Cancel。

多数自定义窗体都有两个命令按钮，确定和取消，使用户能够接受窗体上输入的数据，或者离开窗体。在本练习里，OK按钮将输入在窗体上的数据转移到工作表里。无论何时当他完成数据输入的时候，用户也可以点击取消按钮。你将在本章后面编写合适的VBA过程，使按钮对用户的操作有反应。

10. 添加图像控件

- 点击工具箱上的图像按钮
- 在Cancel按钮下面用鼠标点击，并且拖曳鼠标，画一个长方形，释放鼠标，最后的样子如图10-11所示。窗体将根据选择的是Hardware或者Software选项按钮，显示不同的图片。图像将用VBA程序上载。

11. 检查窗体外观

- 点击窗体标题栏，或者点击窗体上的任意空白区域，选定窗体
- 按下F5键或者选择“运行”|“运行子过程/用户窗体”来显示用户将看到的窗体
- VB将切换到Excel窗口的当前活动工作表，并且显示你设计的自定义窗体。如果你忘了选择窗体，就会出现宏对话框（译者：按下F5的时候），关闭宏对话框，然后重复前面两步
- 点击窗体右上角的关闭按钮(x)来关闭该窗体，返回到VB编辑器窗口。回想一下，我们放置在窗体上的OK和Cancel按钮都还没有任何功能，它们需要VBA过程让它们工作。

在窗体上添加完控件后，你可以使用鼠标或者格式菜单命令来调节控件的对齐和空间。

技巧10-5 使用用户窗体工具栏

用户窗体工具栏包含很多有用的使用窗体的快捷键，例如使控件大小一样，水平或者数值居中，控件边缘对齐，以及组合或取消组合控件。选择“视图”|“工具栏”|“用户窗体”。

22.更改控件名称

在开始编写程序控制窗体之前，你应该给每个放置在窗体上的控件分配自己的名称。尽管VB自动给每个控件分配一个缺省名称，但是要在一个过程里引用多个几乎具有一样的名称的同类对象的话，这些名称是很难区分的，例如：OptionButton1，OptionButton2，等等。给窗体上的控件分配有意义的名称可以让你引用这些控件的VBA过程可读性增加。

给控件命名：

- 点击窗体上合适的控件
- 双击属性窗口的Name属性

注意：在更改Name属性之前，你得确保属性窗口的标题栏显示正确的控件类型。例如，要给一个框架控件重命名的话，就要先点击窗体上的框架控件。当属性窗口显示“属性-Frame1”时，双击Name属性，并且在默认名称加亮区域输入新的名称。

不要将控件的名称和标题（Caption）混淆。例如，在Info Survey窗体上，框架的缺省名称是Frame1，但是该控件的标题是Main Interest。控件的标题可以通过设置其Caption属性来更改。控件的标题允许用户明确控件的目的，以及建议预期的数据类型，然而控件的名称是用在VBA代码里使事情发生的。

1. 给窗体Info Survey上的控件分配名称，如下表所示：（译者：你可以按照自己的方式命名控件，但是一个好的方法是按表中的样式命名控件。例如第一个选项按钮，optHard，opt表示该控件类型为OptionButton，而Hard则表明该控件的作用是Hardware选项。）

对象类型	Name属性
第一个选项按钮	optHard
第二个选项按钮	optSoft
列表框	lboxSystems
第三个选项按钮	optMale
第四个选项按钮	optFemale
第一个复选框	chkIBM
第二个复选框	chkNote
第三个复选框	chkMac
复合框	cboxWhereUsed
文字框	txtPercent
旋转按钮	spPercent
第一个命令按钮	butOK
第二个命令按钮	butCancel
图像	picImage

23.设置其它控件属性

放置在Info Survey窗体上的控件也是对象，这些对象中的每一个都有它们自己的属性和方法。在前面的部分，你更改了所有对象的Name属性，这些后面将在VBA过程里引用。控件的属性可以在自定义窗体的设计时候设定，也可以在运行的时候设置（也就是说，在VBA过程执行的时候）。

我们现在来给所选控件设置一些属性。点击窗体上的控件，定位属性窗口上希望设置的属性，并且在属性名称右边输入新值。例如，设置控件lboxSystems的ControlTipText属性，点击窗体Info Survey上的列表框并且在属性窗口里找到ControlTipText属性，在属性窗口的右边一列里输入你要显示的文本，当用户将鼠标移动到该列表框上面时就会显示该文本：Select only one item

1. 更改对象属性，如下所示：

对象名称	属性	更改为
------	----	-----

lboxSystems	ControlTipText	Select only one item
spPercent	Max	100
spPercent	Min	0
OK button	Accelerator	O
Cancel button	Accelerator	C
picImage	PictureSizeMode	0 - fmPictureSizeModeClip

Accelerator属性指明对象名称中的哪一个字母可以用在键盘快捷组合家里来激活该控件。该特定的字母在该对象的标题里将显示为下划线。例如，显示该窗体后，你将可以使用组合键Alt+O快速选择OK按钮。Info Survey窗体对象的其它属性将VBA过程里直接设置。

24.准备工作表以储存窗体数据

用户在窗体上选择了合适的选项并点击OK按钮之后，这些被选择的数据将会转移到一个工作表中。然而，在此之前，你得准备一个工作表来接收这些数据，并且给用户一个易于操作的界面来启动你的窗体。按照下述步骤来准备你的工作表：

1. 激活Excel窗口
2. 双击工作簿Chap10.xls的Sheet1页，并且输入新名称：Info Survey
3. 输入列标，如图10-13所示
4. 选择K列和第一行，并且将它们的底色改为你喜欢的颜色（使用“格式”工具栏上的“填充底色”按钮）。
从工作表里启动自定义窗体的最简单方法是点击一个按钮，接下来的步骤带领你在工作表Info Survey里添加按钮Survey
5. 选择“视图”|“工具栏”，并选择“窗体”
6. 点击窗体工具栏上的按钮控件，在单元格K2里放置一个按钮。当出现指定宏对话框时，在“宏名”框里面输入DoSurvey，并且点击确定。稍后你将编写该过程。
7. 当你返回工作表时，该被指定宏DoSurvey的按钮（按钮1）应该仍然被选中了，给它输入一个新名称：Survey。如果该按钮没有被选定的话，那么就在其上单击右键来选中它，选择快捷菜单上的“编辑文字”并输入Survey作为其新名称。要退出编辑模式，可以点击按钮之外的任何地方。
8. 保存你对Chap10.xls做的改变。

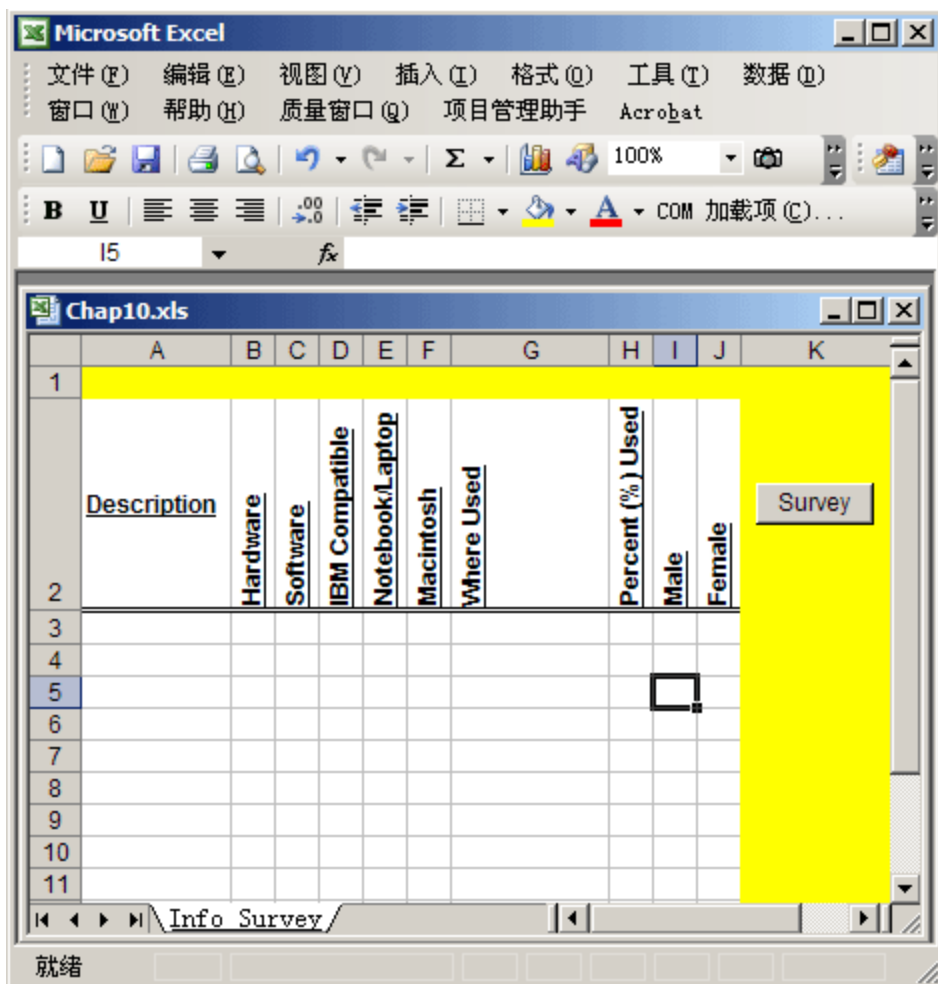


图10-13 Survey按钮将会启动Info Survey窗体。当用户点击窗体上的OK按钮时，窗体数据将会放置到该工作表里面

25.显示自定义窗体

每个用户窗体都有Show方法，让你可以给用户显示该窗体。在下面的例子里，你将准备DoSurvey过程。回想前面部分，你已经将该过程指定给了Info Survey工作表的Survey按钮。

1. 在VB编辑器窗口，选择工程CustomForms (Chap10. xls)，并且选择“插入”|“模块”
2. 在属性窗口，将新模块的名称改为ShowSurvey
3. 输入下述显示自定义窗体的过程

```
Sub DoSurvey()  
    InfoSurvey.Show  
End Sub
```

注意，Show方法前面是窗体对象的名称，正如出现在窗体文件夹里面的那样（InfoSurvey）

4. 保存Chap10. xls的变化
5. 切换到Excel窗口，并点击Survey按钮，窗体Info Survey将出现。

注意，如果你点击Survey按钮后出现错误信息的话，那么你可以没有按前面步骤6那样给该按钮指定需要的宏。要更正该错误，可以点击确定，单击右键于Survey按钮，并选择快捷菜单上的“指定宏”，然后点击指定宏对话框里的DoSurvey宏，并点击确定退出。现在，你可以点击Survey按钮显示窗体了。

6. 通过点击窗体右上角的关闭按钮（x），关闭Info Survey窗体

26.设置 Tab 顺序

用户可以使用鼠标或者Tab键在窗体上移动，因为许多用户倾向于使用键盘在窗体上移动，所以决定窗体上控件激活的顺序是

很重要的。下列步骤示范设置Info Survey窗体上控件的Tab顺序：

1. 在工程浏览器窗口里的窗体文件夹里，双击InfoSurvey窗体
2. 选择“视图”|“Tab键顺序”。Tab键顺序对话框出现了，该对话框按控件添加的顺序显示窗体InfoSurvey上的所有控件名称。对话框的右边有一些按钮，允许你向上或者向下移动所选的控件。要移动某个控件的话，可以点击其名称并且点击上移或者下移按钮，直到你想要的位置。
3. 按照图10-14显示的那样重新安排Info Survey窗体上的控件
4. 点击确定，退出Tab键顺序对话框
5. 返回Excel界面，并且点击按钮Survey
6. 按Tab键向前移动，按Shift+Tab向后移动
7. 关闭InfoSurvey窗体。如果你想要更改控件激活的顺序的话，那么重新打开Tab键顺序对话框，并作适当的更改。

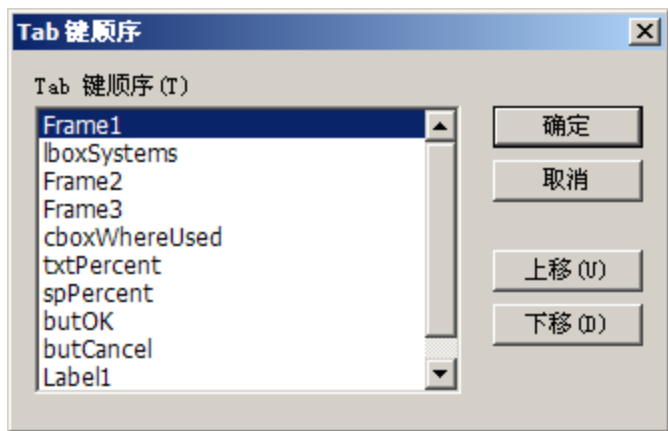


图10-14 Tab键顺序对话框让你觉得按Tab键时控件激活的顺序

27. 了解窗体和控件事件

除了属性和方法之外，每个窗体和控件都有一套预先设计好的事件。事件是指一类操作，例如点击鼠标、按键、从清单里选择一项或者改变列表框里可用的清单或项目。事件可以由用户或者系统引发。编写事件过程，可以明确窗体或控件如何对该事件作出反应。

当你设计一个自定义窗体的时候，你应该预想和规划运行（当窗体使用的时候）时能够发生的一些事件。最常见的事件是点击事件。每当点击一个命令按钮的时候，就会引发某个事件过程对该按钮的点击事件作出反应。窗体本身可以对20多种事件作出反应，包括Click（点击）、Db1Click（双击）、Activate（激活）、Initialize（初始化）和Resize（重置大小）。

表10-2列出了各种窗体控件可以识别的事件。如果某个控件不能识别某个事件，那么表格相应单元格便显示“N”，否则为空白。花几分钟熟悉一下事件名称吧，例如，看看表格里的AddControl事件，一眼就可以看出，该事件只对三个对象可用：框架、多页和窗体本身。

Event name	UserForm	Label	TextBox	ComboBox	ListBox	CheckBox	OptionButton	ToggleButton	Frame	CommandButton	TabStrip	MultiPage	ScrollBar	SpinButton	Image	RefEdit
Activate		N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
AddControl		N	N	N	N	N	N	N		N	N		N	N	N	N
AfterUpdate	N	N							N	N	N	N			N	
BeforeDragOver																
BeforeDropOrPaste																
BeforeUpdate	N	N							N	N	N	N			N	
Change	N	N							N	N					N	
Click			N										N	N		N
DblClick													N	N		
Deactivate		N		N	N	N	N	N	N	N	N	N	N	N	N	N
DropButtonClick	N	N			N	N	N	N	N	N	N	N	N	N	N	
Enter	N	N													N	
Error																
Initialize		N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
Exit	N	N													N	
KeyDown		N													N	
KeyPress		N													N	
KeyUp		N													N	
Layout		N	N	N	N	N	N	N		N	N		N	N	N	N
MouseDown													N	N		
MouseMove													N	N		
MouseUp													N	N		
QueryClose		N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
RemoveControl		N	N	N	N	N	N	N		N	N		N	N	N	N
Resize		N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
Scroll		N	N	N	N	N	N	N		N	N			N	N	N
SpinDown	N	N	N	N	N	N	N	N	N	N	N	N	N		N	N
SpinUp	N	N	N	N	N	N	N	N	N	N	N	N	N		N	N
Terminate		N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

表10-2 窗体和控件事件

你创建的每个窗体都有一个窗体模块用来储存VBA事件过程。你可以通过以下几种方式来进入窗体模块，编写事件过程或者找到某个控件可识别的事件：

- 双击某个控件
- 在控件上单击右键，并选择快捷菜单上的查看代码
- 点击工程浏览窗口的查看代码按钮
- 双击用户窗体上任何未用的区域

执行以上任意操作都会导致打开窗体的代码窗口。图10-15显示了通过双击窗体上的命令按钮而激活的代码窗口。注意，Microsoft Visual Basic标题栏显示的标题：Chap10.xls - [UserForm1(代码)]。窗体模块包含一个“通用”部分，也包含放置于窗体上面的每个控件的部分。通用部分用来声明窗体变量或者常量。

你可以通过点击右上角复合框右边的向下箭头来进入预期的部分。该复合框被称为过程框，它显示左手边复合框显示控件可以识别的所有事件过程。已经编写了过程的事件显示为粗体。

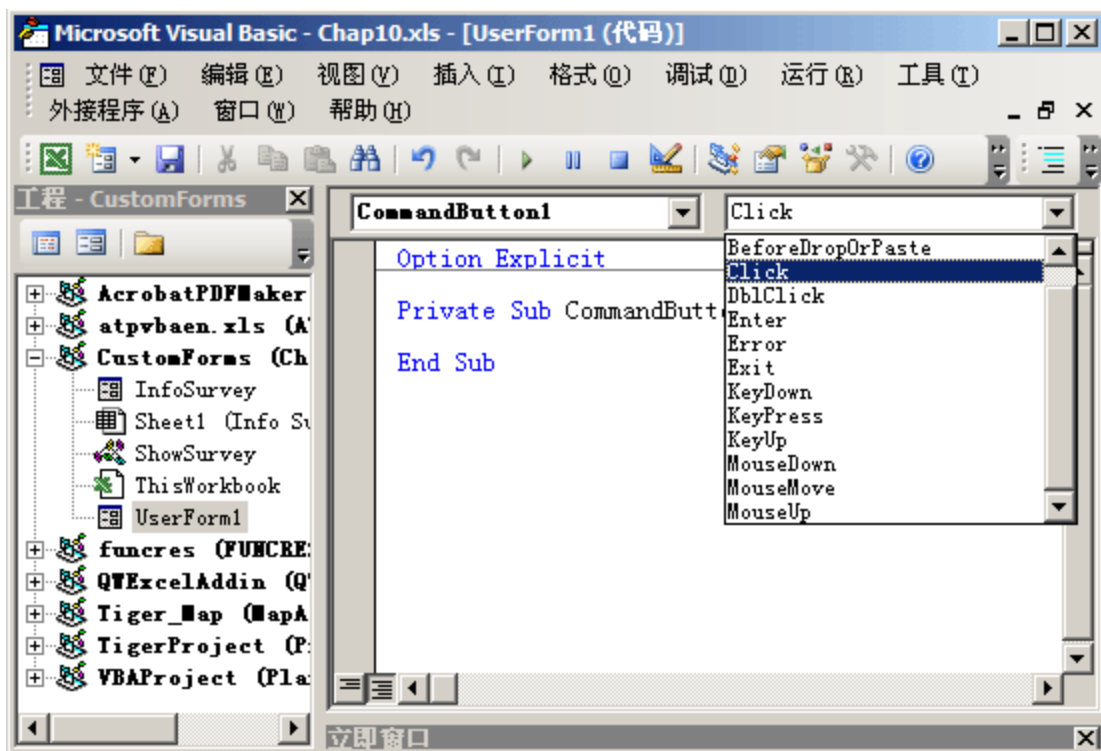


图10-15 过程框列出了命令按钮控件可用的事件过程

28.编写 VBA 过程对窗体和控件事件反应

在用户能够使用自定义窗体完成特殊任务之前，你通常必须编写一些VBA过程。正如前面提及的，VB编辑器创建的每个窗体都有一个模块以储存该窗体使用的过程。

在显示自定义窗体之前，你可以需要设置控件的初始值。编写一个Initialize事件过程，可以给控件设置初始值，或者说默认值，每次显示窗体的时候控件都会拥有这些值。

Initialize事件发生在窗体启动时，但是在它显示在屏幕之前。假设你想要Info Survey窗体显示以下初始设置：

1. 在Main Interest框架里，选择了Hardware按钮
2. 下面的列表框包含了对应Hardware选项按钮的内容
3. Computer Type复选框里没有一个复选框是被选中的
4. 标签Where Used下面复合框显示第一条可用的项目，并且用户不能给该复合框添加项目
5. 在旋转按钮旁边的文字框显示初始值（0）
6. 图像控件显示与Hardware或Software选项按钮相关的图片

29.编写过程来初始化窗体

1. 在工程浏览器窗口，双击InfoSurvey窗体
2. 双击窗体背景，打开活动窗体的代码窗口。

当你双击窗体或控件的时候，代码窗口会自动打开并且该被点击的窗体或者控件的Click事件就会出现以编辑。

在过程定义（图10-15）时，VB自动在关键字Sub之前添加Private。私有过程只能从当前窗体模块里调用，换句话说，在当前工程的其它模块里的过程不能调用该私有过程。在代码窗口的顶端，有两个复合框，左边的复合框显示所有的窗体对象；右边的复合框则显示所选控件能够识别的所有事件。

3. 点击过程框的向下箭头，并且选择Initialize事件，VB将显示UserForm_Initialize过程在代码窗口：

```
Private Sub UserForm_Initialize()  
End Sub
```

4. 在关键字Private Sub和End Sub之间输入窗体的初始设置，完整的UserForm_Initialize过程如下所示：

```
Private Sub UserForm_Initialize()
```

```

' select the Hardware option 选择Hardware选项
optHard.Value = True

' turn off the Software option and all the check boxes 关闭Software选项和所有复选框
optSoft.Value = False
chkIBM.Value = False
chkNote.Value = False
chkMac.Value = False

' display a zero in the text box 文字框显示0
txtPercent.Value = 0

' call the procedure to populate the list box with
' hardware options 调用过程用硬件选项来填充列表框
Call ListHardware

' populate the combo box 添加复合框项目
With Me.cboWhereUsed
    .AddItem "home"
    .AddItem "work"
    .AddItem "school"
    .AddItem "work/home"
    .AddItem "home/school"
    .AddItem "work/home/school"
End With

' select the first element in the combo box 选择复合框第一个项目
Me.cboWhereUsed.ListIndex = 0

' select the first element in the list box 选择列表框的第一个项目
Me.lboxSystems.ListIndex = 0

' load a picture file for the Hardware option 上载Hardware选项的图片文件
Me.picImage.Picture = LoadPicture("C:\cd.bmp")
End Sub

```

你可以使用关键字Me代替窗体的实际名称来简化事件过程代码，例如，除了使用语句：

```
InfoSurvey.cboWhereUsed.ListIndex = 0
```

之外，你也可以使用下面的语句，节约打字时间：

```
Me.cboWhereUsed.ListIndex = 0
```

特别是当窗体名称很长时，这种技术很有用。注意，列表框的第一个成员的索引号为0，因此，如果想要选择列表里的第二个项目的话，你就必须设置其ListIndex属性为1。

该过程结束时给图像控件上载图片，请确保该指定的图片文件可以在指定的文件夹里找到。如果你没有该文件，那么输入你想显示的图片文件的完整路径。

过程UserForm_Initialize调用外部过程（ListHardware）用硬件成员来填充它的列表框控件。

5. 激活ShowSurvey模块并且输入过程ListHardware，如下所示：

```

Sub ListHardware()
    With InfoSurvey.lboxSystems
        .AddItem "CD-ROM Drive"
        .AddItem "Printer"
        .AddItem "Fax"
    End With
End Sub

```



```
.AddItem "Network"
.AddItem "Joystick"
.AddItem "Sound Card"
.AddItem "Graphics Card"
.AddItem "Modem"
.AddItem "Monitor"
.AddItem "Mouse"
.AddItem "Zip Drive"
.AddItem "Scanner"
```

```
End With
```

```
End Sub
```

既然你已经准备好了UserForm_Initialize过程和ListHardware过程，那么你可以运行窗体查看结果了。

6. 在工作表Info Survey，使用Survey按钮来启动窗体，窗体显示后，用户选择合适的选项或者点击Cancel按钮。当用户点击Software选项按钮时，下面的列表框应该显示不同的项目，同时，图像控件应该上载一个不同的图片。下一节将解释如何编写这些事件。

30.编写过程填充列表框控件

在前面的部分，你准备了ListHardware过程用Hardware成员来填充lboxSystems列表框，你可以使用同样的方法将Software项目上载到该列表框。

1. 激活ShowSurvey模块，并输入过程ListSoftware代码，如下所示：

```
Sub ListSoftware()
    With InfoSurvey.lboxSystems
        .AddItem "Spreadsheets"
        .AddItem "Databases"
        .AddItem "CAD Systems"
        .AddItem "Word Processing"
        .AddItem "Finance Programs"
        .AddItem "Games"
        .AddItem "Accounting Programs"
        .AddItem "Desktop Publishing"
        .AddItem "Imaging Software"
        .AddItem "Personal Information Managers"
    End With
End Sub
```

31.编写过程控制选项按钮

1. 激活窗体InfoSurvey，并双击位于Main Interest框架里的Software选项按钮
2. 当代码窗口出现optSoft_Click过程构架的时候，选中该代码并按下Delete键
3. 点击右上角的复合框向下箭头，并且选择Change事件过程，VB将会自动为你输入optSoft_Change过程的开头和结尾
4. 输入optSoft_Change过程代码，显示如下：

```
Private Sub optSoft_Change()
    Me.lboxSystems.Clear
    Call ListSoftware
    Me.lboxSystems.ListIndex = 0
    Me.picImage.Picture = LoadPicture("C:\Books.bmp")
End Sub
```

过程optSoft_Change开始时，使用Clear方法将列表框lboxSystems的当前成员列表清除，下一条语句调用ListSoftware过程用软件成员来填充列表框，换句话说，当用户点击Software按钮时，程序将硬件成员清除然后添加软件成员。如果你

在添加新成员之前，不删除列表框的话，这些新成员就会附加在当前清单后面。语句Me.lboxSystems.ListIndex = 0选择列表里的第一条成员。该过程里的最后一条语句为图像控件上载图片文件，请确保换上你电脑里有效图片的完整路径。因为用户在选择Software按钮之后，还可能要重新选择Hardware按钮，所以，你必须给optHard选项按钮创建一个类似的Change事件过程。

5. 在optSoft_Change过程下面，输入下述过程optHard_Change:

```
Private Sub optHard_Change()  
    Me.lboxSystems.Clear  
    Call ListHardware  
    Me.lboxSystems.ListIndex = 0  
    Me.picImage.Picture = LoadPicture("C:\cd.bmp")  
End Sub
```

6. 点击工作表Info Survey里的按钮Survey启动窗体来查看结果。当你点击Software选项按钮时，你应该看到列表框里的软件成员，同时，图像控件也应该显示了相应的图片。点击Hardware选项按钮后，列表框应该显示适当的硬件成员，同时，图像控件应该显示一张不同的图片。
7. 点击窗体右上角的关闭按钮，关闭该窗体

32.编写过程同步文字框和旋转按钮

Info Survey窗体在旋转按钮前面有个文字框，用户可以直接在文字框里输入或者使用旋转按钮，在文字框里表明Hardware或者Software成员使用的百分率。文字框的初始值为0，假设用户输入了10，现在要用旋转按钮将它增加到15。要激活这个动作的话，那么该文字框和旋转按钮必须得同步。每个对象需要有一个单独的Change事件过程。

1. 在旋转按钮上单击右键，并选择快捷菜单上的查看代码
2. 输入过程spPercent_Change过程，如下所示：

```
Private Sub spPercent_Change()  
    txtPercent.Value = spPercent.Value  
End Sub
```

使用旋转按钮将会导致文字框数值增加或减少。

3. 输入下述过程txtPercent_Change:

```
Private Sub txtPercent_Change()  
    Dim entry As String  
  
    On Error Resume Next  
  
    entry = Me.txtPercent.Value  
    If entry > 100 Then  
        entry = 0  
        Me.txtPercent.Value = entry  
    End If  
  
    spPercent.Value = txtPercent.Value  
End Sub
```

过程txtPercent_Change确保只有从0到100之间的数值可以输入在文字框里，该过程使用了On Error Resume Next语句来忽略错误数据的输入。如果用户输入了一个非数字数据（或者大于100的数字），VB将会将文字框的值重新设置为0。每次点击旋转按钮，文字框里的数字将会增加或者减少1。

33.编写过程关闭用户窗体

显示窗体之后，用户可能需要通过按Esc键或者点击Cancel按钮来取消窗体，可以准备一个使用Hide方法的简单过程，让窗体从屏幕上消失。

1. 双击Cancel按钮，并且输入下述过程cmdCancel_Click:

```
Private Sub butCancel_Click()
    Me.Hide
End Sub
```

方法Hide将对象隐藏，但是不从内存里清除它。这样，当用户看不到窗体的时候，你的VBA过程仍然可以在屏幕后面使用该窗体的对象和属性。使用Unload方法可以将窗体从屏幕上卸退并且从内存里清除：

```
Unload Me
```

当窗体卸退后，所有相关的内存都会被收回，用户不能再和窗体相互交流了，窗体的对象也不能为你的VBA过程访问了，除非使用Load语句再将窗体放置于内存里。

34. 转移窗体数据到工作表

当用户点击OK按钮，窗体的选择应该写入工作表中，用户可以通过点击Cancel按钮随时退出窗体。

1. 双击OK按钮，并输入过程cmdOK_Click，如下所示：

```
Private Sub butOK_Click()
    Me.Hide
    r = Application.CountA(Range("A:A"))
    Range("A1").Offset(r + 1, 0) = Me.lboxSystems.Value
    If Me.optHard.Value = True Then
        Range("A1").Offset(r + 1, 1) = "*"
    End If

    If Me.optSoft.Value = True Then
        Range("A1").Offset(r + 1, 2) = "*"
    End If

    If Me.chkIBM.Value = True Then
        Range("A1").Offset(r + 1, 3) = "*"
    End If

    If Me.chkNote.Value = True Then
        Range("A1").Offset(r + 1, 4) = "*"
    End If

    If Me.chkMac.Value = True Then
        Range("A1").Offset(r + 1, 5) = "*"
    End If

    Range("A1").Offset(r + 1, 6) = Me.cboWhereUsed.Value
    Range("A1").Offset(r + 1, 7) = Me.txtPercent.Value

    If Me.optMale.Value = True Then
        Range("A1").Offset(r + 1, 8) = "*"
    End If

    If Me.optFemale.Value = True Then
        Range("A1").Offset(r + 1, 9) = "*"
    End If
    Unload Me
End Sub
```

过程butOK_Click以隐藏用户窗体开始。语句：

```
r = Application.CountA(Range("A:A"))
```

使用了VB函数CountA来计算A列里含有数据的单元格数目，函数的结果被赋予变量r。下一句：

```
Range("A1").Offset(r + 1, 0) = Me.lboxSystems.Value
```

将列表框选择的项目输入到A列最后一个使用了的单元格的下面一个单元格（r+1）。接下来，是好几条条件语句。第一条告诉VB，当Hardware选项按钮被选中时，在B列适当的单元格里输入一个星号。B列位于A列的右边一列，因此，在Offset方法第二个参数的位置为1。第二条If语句，当用户选择Software选项按钮的话，就在C列输入星号。类似的指令记录复选框的实际数值。在G列，将会输入Where Used复合框里所选的项目。H列显示Percent (%) Used 文字框里的数字，而I列和J列则分别显示提高调查人员的性别。

35.使用 Info Survey 应用程序

现在，你的应用程序已经准备好做最后测试了。

1. 切换到Excel界面，Info Survey工作表，并且点击Survey按钮
2. 当窗体出现时，选择适当的选项，并点击OK
3. 激活窗体几次，每次选择不同的选项
4. 保存Chap10.xls变化

36.应用程序示例 2：学生和考试

近年来，许多Windows应用软件越来越依赖于那些将各种控件组合在一起的对话框。“选项”对话框就是很好的例子，使用Tab页，你可以给一个对话框里的很多控件提高设置。用Tab页组合的对话框非常容易和方便使用。当给更高级的VBA应用程序设计自定义窗体时，你可以利用工具箱里可用的两种特殊的Tab控件：多页控件和TabStrip控件。应用程序示例2，本章剩下的主题，使用这些和其它的高级控件（例如RefEdit和Calendar）来追踪学生和他们的考试分数。

37.使用多页和 TabStrip 控件

图10-16所示的自定义窗体中间的对象是一个多页控件，它由两页组成。第一页含有文字框和复合框来收集学生信息，例如社会保险号码（SSN: Social Security Number），名和姓，学习年限和专业。窗体上部有一组选项按钮，让你明确学生状况：New或者Active。如果该学生的数据还没有输入到工作表中，那么该学生的状况就是New。当窗体第一次启动时，New选项按钮就会自动被选上。当你想要回顾或者更新已有学生的数据时，点击Active选项按钮将会显示RefEdit控件，用来使用学生姓名填充列表框控件，如同工作表中的一样。

1. 在当前工程里插入一个新用户窗体，并重命名为Students
2. 将窗体的Caption属性改为Students and Exams
3. 点击工具箱上的多页控件，然后点击窗体的左上方并拖曳鼠标到窗体右下角
4. 右键单击Page1页，并且选择快捷菜单上的“重命名”，在“题注”文字框里面输入Students，并在“加速键”区域输入S。用同样的方法，重命名第二页Exams，以及输入加速键X
5. 点击Students页，使用工具箱上的控件，在Students页上添加如图10-16所示的所有控件。按照下述指南：
 1. 框架Status包含两个选项按钮，设置它们的Caption属性为New和Active，再将它们的Name属性设置为optNew和optActive
 2. 使用标签标示的文字框，标签的Caption属性分别为SSN, Last Name和First Name。更改文字框的Name属性为txtSSN, txtLast和txtFirst
 3. 标示复合框的标签Caption属性设置为Year和Major。复合框的Name属性设置为cboxYear和cboxMajor。要让用户只能选择一个选项的话，可以给每个复合框的下述属性：MatchRequired为True，MatchEntry为1-fmMatchEntryComplete
 4. 在RefEdit控件上面放置标签Name Range。设置RefEdit控件的Name属性为refNames
 5. 设置列表框的Name属性为lboxStudents，设置ColumnCount属性为2，来显示两列数据
 6. 设置命令按钮的Caption属性为OK和Cancel，并且设置它们的Name属性为cmdOK和cmdCancel

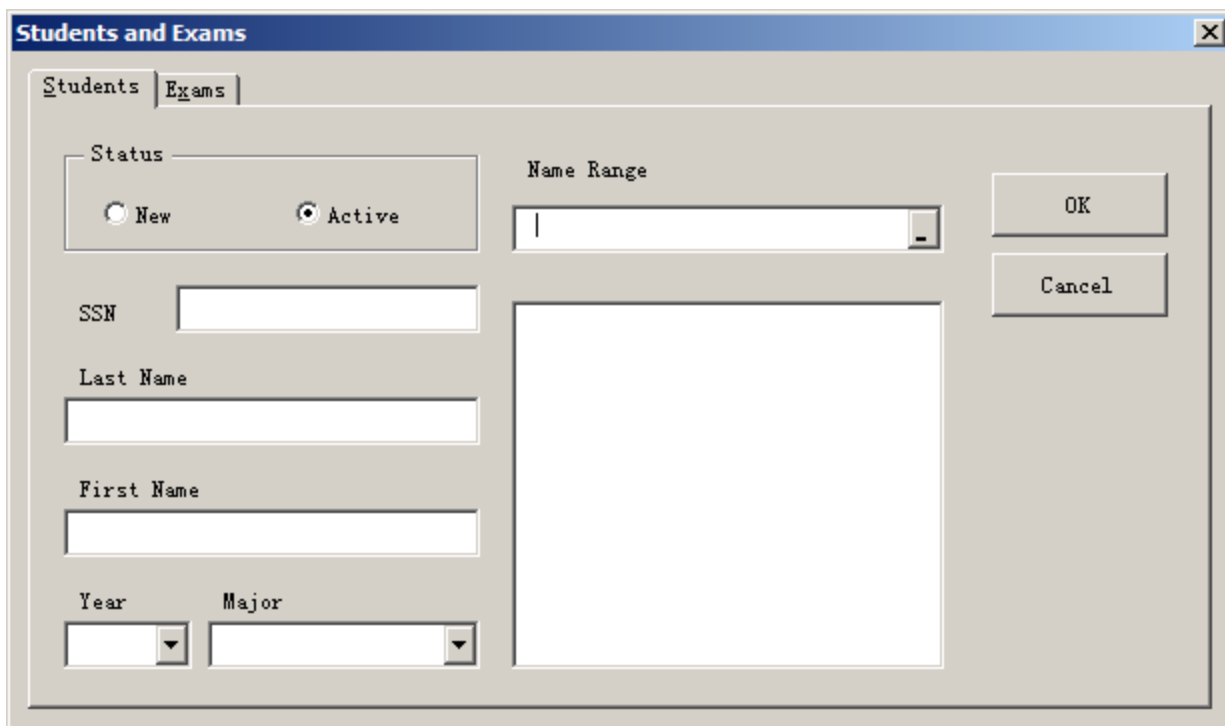


图10-16 多页控件可以包含两页或者多页，每页显示不同的控件

多页控件的第二页（图10-17）用来记录和显示与考试相关的信息。该页包含两个对象。标签控件临时标题为“Last, First”，在运行的时候，该标签将显示前一页选定学生的姓名。该页上的第二个控件是TabStrip控件，它包含了四个tab页，是四门功课考试。尽管在本练习中，页显示在控件的上部，但是，VB同样也允许你将tab的方位设置在底部，左边或者右边。当你从一页翻到另一页时，你可以看到这些相同的控件，只是每个控件显示的数据改变了。

1. 点击多页控件上的Exams页
2. 点击工具箱上的标签控件，并点击Exams页的左上角，拖曳鼠标画一个足够放下人姓名的长方形
3. 在属性窗口，设置该标签的Name属性为lblWho，Caption属性为“Last, First”，设置Font属性为Arial，Bold和14磅
4. 点击工具箱上的TabStrip控件，在紧接标签控件的下面点击，并拖曳鼠标直到达到你想要的大小和形状（参见图10-17）。默认地，TabStrip控件将会显示两页：Tab1和Tab2。要添加新的tab页，首先要点击TabStrip控件内部以选择该对象，再次点击，直到TabStrip控件周围的虚框显示为粗体（或者看到页头有虚线包围），单击右键，并选择快捷菜单上的新建页，VB就会添加Tab3，使用相同的方法添加Tab4。在每页上单击右键，并且选择快捷菜单上的重命名，参见图10-17，给每页设置名称。注意，每页的名称有一个带下划线，它允许用户从键盘访问该页，例如按下Alt+E将激活English页。
5. 点击English页，并开始画下述控件：
 - 如果工具箱上只有题为控件的一页的话，那么回到本章的前面标题为“创建用户窗体的工具”的部分，找到如何添加Calendar控件到工具箱上。然后将Calendar控件放置到TabStrip里，如图10-17所示。设置Calendar控件的ShowDateSelectors属性为True。
 - 添加Enter/Change Grade标签和一复合框，设置复合框的Name属性为cboxGrade
 - 添加两个标签，Caption属性设置为Grade和Date，（另外两个标签的）Name属性设置为lblGrade和lblDate。更改这两个标签的SpecialEffect属性为3-fmSpecialEffectEtched

技巧10-6 设置TabStrip控件

设置TabStrip控件的最好方法是先添加它本身，然后放置其它的控件在里面。然而，如果已经有了控件在那里，那么你可以在它们上面画TabStrip控件，然后使用“移至底层”命令，将TabStrip控件放到Z-顺序的底层。

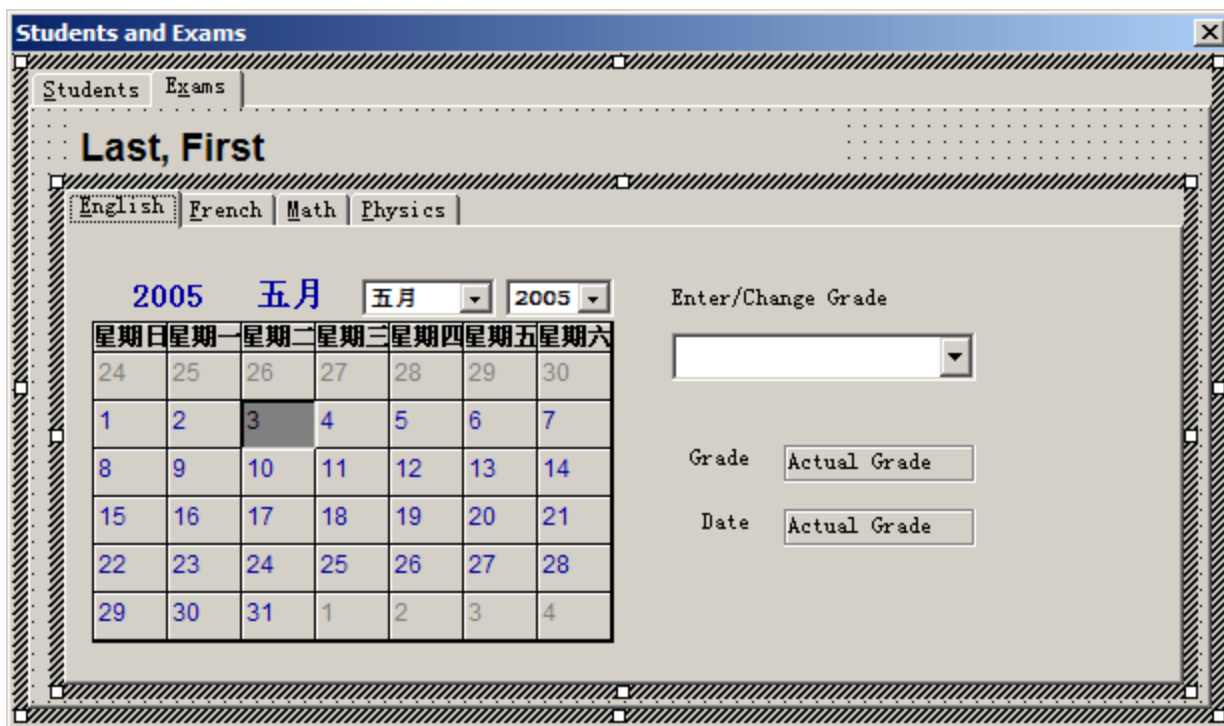


图10-17 外面的框架包含一个多页控件，而里面的框架则包含TabStrip控件

Students and Exams自定义窗体允许你输入新数据，或者显示数据，就像在工作表里一样。图10-18显示了本窗体使用的工作表。

1. 准备如图10-18所示的工作表
2. 在工作表里添加按钮Display Form，并且给他指定宏DoStudents
3. 在VB编辑器屏幕，在当前工程里添加新模块，并设置模块的Name属性为InfoStudents
4. 在InfoStudents模块里输入下述过程DoStudents：

```
Sub DoStudents()  
    Students.Show  
End Sub
```

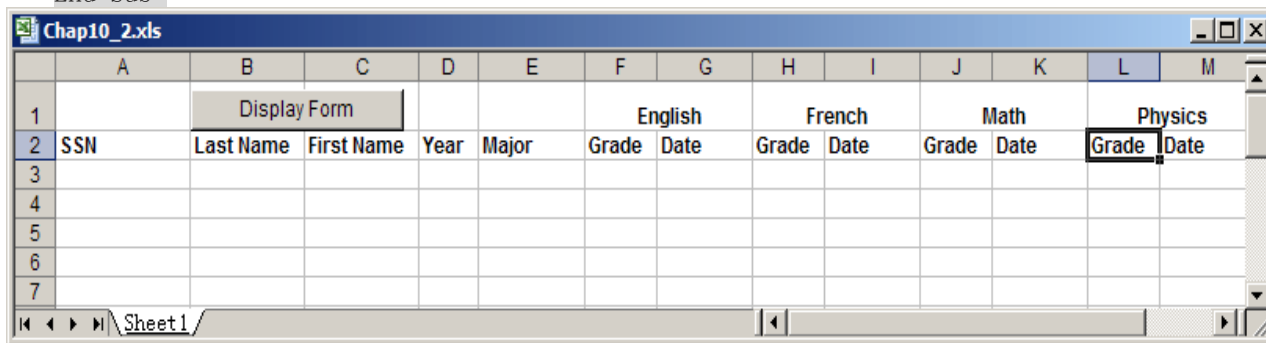


图10-18 Students and Exams应用程序的协助工作表

5. 返回到工作表，并点击按钮Display Form，测试DoStudents过程
6. 点击窗体右上角的关闭按钮，关闭窗体

38.给窗体 Students and Exams 自定义窗体编写 VBA 过程

自定义窗体Students and Exams包含很多VBA过程，显示在下面。这些过程的代码必须输入在窗体模块里面，双击窗体背景，激活窗体模块。

1. 从代码窗口左上角的复合框里选择“(通用)”。右边的过程选择复合框应该显示“(声明)”，输入下述变量声明：

2. 输入过程UserForm_Initialize代码，初始化窗体

```

'Declarations
Dim r As Integer
Dim nr As Integer
Dim indexPlus As Integer
Dim YesNo As Integer

Private Sub UserForm_Initialize()
    'select first page of the MultiPage control 选择多页控件的第一页
    'page numbering begins from zero (0) 页码编号从0开始
    Me.MultiPage1.Value = 0

    'choose the New option button 选择New选项按钮
    optNew.Value = True

    'hide three controls on startup 启动时先隐藏三个控件
    lblNames.Visible = False '前面应该将标签Name Range命名为lblNames（原文为lblLast）
    refNames.Visible = False
    lboxStudents.Visible = False

    'populate the Year combo box 填充Year复合框
    With Me.cboxYear
        .AddItem "1"
        .AddItem "2"
        .AddItem "3"
        .AddItem "4"
    End With

    'populate the Major combo box 填充Major复合框
    With Me.cboxMajor
        .AddItem "English"
        .AddItem "Chemistry"
        .AddItem "Mathematics"
        .AddItem "Linguistics"
        .AddItem "Computer Science"
    End With

    'populate a combo box with grades 填充得分复合框
    With Me.cboxGrade
        .AddItem "A"
        .AddItem "B"
        .AddItem "C"
        .AddItem "D"
        .AddItem "F"
    End With

    'display date in the lblDate label control 在lblDate标签里显示日期
    Me.lblDate.Caption = Me.Calendar1.Value

    'activate the first tab in the TabStrip control 激活TabStrip控件的第一页
    Me.TabStrip1.Value = 0

```

```
' activate the SSN text box 激活SSN文字框
Me.txtSSN.SetFocus
```

```
End Sub
```

3. 输入两个过程来控制选项按钮 (optNew_Click和optActive_Click)

```
Private Sub optNew_Click()
```

```
    lblNames.Visible = False
```

```
    refNames.Visible = False
```

```
    lboxStudents.Visible = False
```

```
    Me.MultiPage1(1).Enabled = False
```

```
    If lboxStudents.RowSource < > "" Then
```

```
        Me.txtSSN.Text = ""
```

```
        Me.txtLast.Text = ""
```

```
        Me.txtFirst.Text = ""
```

```
        Me.cboYear.Text = ""
```

```
        Me.cboMajor.Text = ""
```

```
        Me.txtSSN.SetFocus
```

```
    End If
```

```
    Me.txtSSN.SetFocus
```

```
End Sub
```

```
Private Sub optActive_Click()
```

```
    lblNames.Visible = True
```

```
    refNames.Visible = True
```

```
    refNames.SetFocus
```

```
    If lboxStudents.RowSource < > "" Then
```

```
        lboxStudents.Visible = True
```

```
        Call lboxStudents_Change
```

```
    End If
```

```
End Sub
```

4. 输入过程lboxStudents_Change和refNames_Change的代码，这些控制Students页上面的RefEdit和列表框控件：

```
Private Sub lboxStudents_Change()
```

```
    indexPlus = lboxStudents.ListIndex + 3
```

```
    With ActiveWorkbook.Worksheets("Sheet2")
```

```
        Me.txtSSN.Text = Range("A" & indexPlus).Value
```

```
        Me.txtLast.Text = Range("B" & indexPlus).Value
```

```
        Me.txtFirst.Text = Range("C" & indexPlus).Value
```

```
        Me.cboYear.Text = Range("D" & indexPlus).Value
```

```
        Me.cboMajor.Text = Range("E" & indexPlus).Value
```

```
        Call TabStrip1_Change
```

```
        Me.MultiPage1(1).Enabled = True
```

```
    End With
```

```
End Sub
```

```
Private Sub refNames_Change()
```

```
    lboxStudents.RowSource = refNames.Value
```

```
    lboxStudents.ListIndex = 0
```

```
lboxStudents.Visible = True
Call lboxStudents_Change
```

```
End Sub
```

5. 输入代码来控制命令按钮OK (cmdOK_Click) 和Cancel (cmdCancel_Click):

```
Private Sub cmdOK_Click()
    If Me.optNew.Value = True Then
        Me.Hide
        ActiveWorkbook.Sheets("Sheet2").Select
        r = ActiveSheet.UsedRange.Rows.Count
        nr = r + 1
        Range("A" & nr).Value = Me.txtSSN.Text
        Range("B" & nr).Value = Me.txtLast.Text
        Range("C" & nr).Value = Me.txtFirst.Text
        Range("D" & nr).Value = Me.cboYear.Text
        Range("E" & nr).Value = Me.cboMajor.Text
        Me.txtSSN.Text = ""
        Me.txtLast.Text = ""
        Me.txtFirst.Text = ""
        Me.cboYear.Text = ""
        Me.cboMajor.Text = ""
        Me.txtSSN.SetFocus
```

```
' redisplay the form
Me.Show
```

```
Else
    MsgBox "This control is currently unavailable."
```

```
End If
```

```
End Sub
```

```
Private Sub cmdCancel_Click()
    Unload Me
    Set Students = Nothing
```

```
End Sub
```

6. 输入过程cboxGrade_Click来控制位于Exams页上的Grade复合框:

```
Private Sub cboxGrade_Click()
    YesNo = MsgBox("Enter the grade in the worksheet?", _
        vbYesNo, "Modify Grade")
```

```
If YesNo = 6 Then
    Me.lblGrade.Caption = cboxGrade.Value
    Select Case TabStrip1.Value
        Case 0
            Range("F" & indexPlus).Value = Me.lblGrade.Caption
        Case 1
            Range("H" & indexPlus).Value = Me.lblGrade.Caption
        Case 2
            Range("J" & indexPlus).Value = Me.lblGrade.Caption
        Case 3
            Range("L" & indexPlus).Value = Me.lblGrade.Caption
    End Select
```

```
cboxGrade.Value = ""
```

```
End If
```

```
End Sub
```

7. 输入过程Calendar1_Click, 如下所示:

```
Private Sub Calendar1_Click()
```

```
YesNo = MsgBox("Enter the date in the worksheet?", vbYesNo, _  
"Modify Date")
```

```
If YesNo = 6 Then
```

```
Me.lblDate.Caption = Calendar1.Value
```

```
Select Case TabStrip1.Value
```

```
Case 0
```

```
Range("G" & indexPlus).Value = Me.lblDate.Caption
```

```
Case 1
```

```
Range("I" & indexPlus).Value = Me.lblDate.Caption
```

```
Case 2
```

```
Range("K" & indexPlus).Value = Me.lblDate.Caption
```

```
Case 3
```

```
Range("M" & indexPlus).Value = Me.lblDate.Caption
```

```
End Select
```

```
End If
```

```
End Sub
```

8. 输入过程TabStrip1_Change和MultiPage1_Change, 如下所示:

```
Private Sub TabStrip1_Change()
```

```
indexPlus = lboxStudents.ListIndex + 3
```

```
With ActiveWorkbook.Worksheets("Sheet2")
```

```
Select Case TabStrip1.Value
```

```
Case 0
```

```
' English
```

```
Me.lblGrade.Caption = Range("F" & indexPlus).Value
```

```
Me.lblDate.Caption = Range("G" & indexPlus).Value
```

```
Case 1
```

```
' French
```

```
Me.lblGrade.Caption = Range("H" & indexPlus).Value
```

```
Me.lblDate.Caption = Range("I" & indexPlus).Value
```

```
Case 2
```

```
' Math
```

```
Me.lblGrade.Caption = Range("J" & indexPlus).Value
```

```
Me.lblDate.Caption = Range("K" & indexPlus).Value
```

```
Case 3
```

```
' Physics
```

```
Me.lblGrade.Caption = Range("L" & indexPlus).Value
```

```
Me.lblDate.Caption = Range("M" & indexPlus).Value
```

```
End Select
```

```
End With
```

```
End Sub
```

```
Private Sub MultiPage1_Change()
```

```
Me.lblWho.Caption = Me.txtLast.Value & ", " & _
```

```
& Me.txtFirst.Value
```

```
Call TabStrip1_Change
```

End Sub

39.使用自定义窗体 Students and Exams

既然你已经准备好有必须的VBA过程了，我们来看看该窗体是如何对用户操作反应的。

1. 切换到Excel窗口，并激活Sheet1
2. 点击按钮Display Form

点击Display Form按钮运行过程DoStudents，该过程显示自定义窗体Students and Exams。在窗体出现在屏幕上之前，VB执行过程UserForm_Initialize里面的每条语句。结果显示为图10-19。

图10-19 工作表里的按钮Display Form让你快速访问自定义窗体Students and Exams来查看或者输入数据。当窗体上载时，只有符合选定的选项按钮的控件才会显示。

窗体显示后，你就可以输入新学生并点击OK将学生的数据转移到工作表。当你点击OK按钮时，就会执行cmdOK_Click过程。注意，你不能输入新学生参加的考试，因为多页控件的第二页（Exams）这时不可用。一旦新学生的数据被写入工作表，该窗体就会重新显示，你可以继续输入数据，或者你可以点击Cancel，将窗体从屏幕上清除。当你点击Cancel按钮时，就会执行过程cmdCancel_Click。

3. 使用自定义窗体Students and Exams输入两位新学生的数据。任何时候，你都可以点击Active选项按钮，从已有学生中上传数据。当你点击Active选项按钮时，标签Name Range和RefEdit控件就变为可见的了。
4. 点击Active选项按钮，然后点击RefEdit控件的减号按钮
5. 选择工作表里的姓名区域，如图10-20所示。

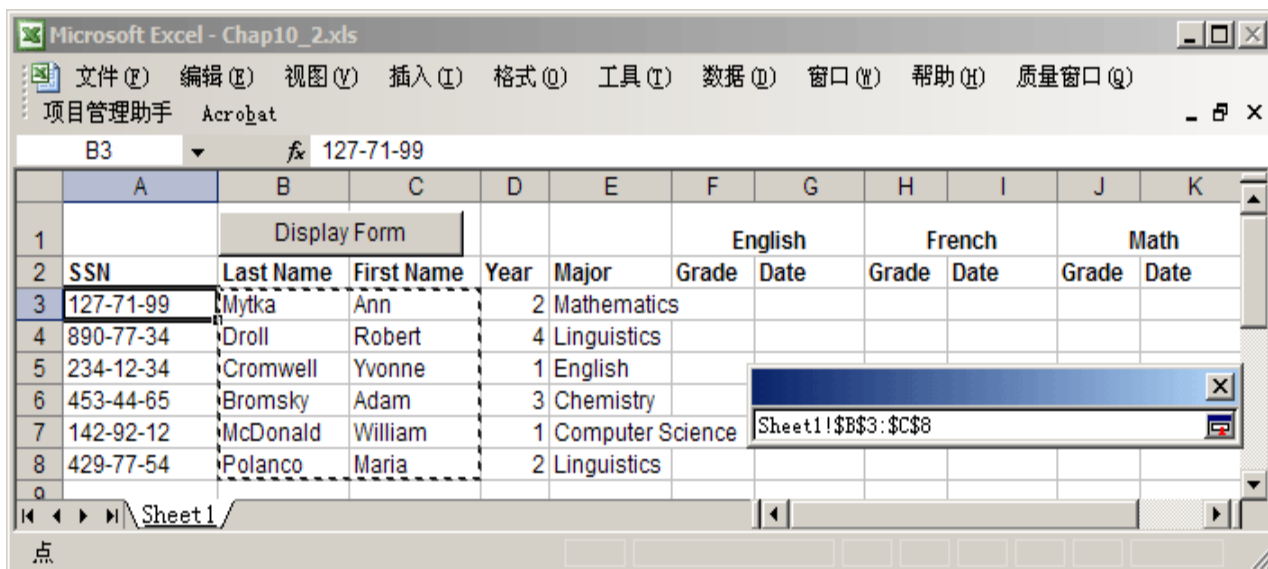


图10-20 使用RefEdit控件，你可以选择包含你想要使用数据的单元格区域

使用RefEdit控件，你可以选择工作表里的单元格区域，在本练习里，选择了包含学生姓和名的单元格。选择有效数据很重要，开始点击Last Name列标下面的单元格（B3）并且向下向右拖曳鼠标以至包括学生的名

注意，当你使用RefEdit控件时，窗体临时被隐藏。你所选择的单元格出现在RefEdit控件上。点击RefEdit控件的减号按钮返回窗体。当你返回窗体时，过程refNames_Change正在运行，该过程使用单元格区域地址，将学生姓名填充到列表框控件里。该过程的最后一条语句调用lboxStudents_Change过程，确保窗体的文字框和复合框与列表框里所选学生姓名同步。列表框显示已有学生的姓名，被选学生的数据显示在左边的文字框和复合框里面（参见图10-21）

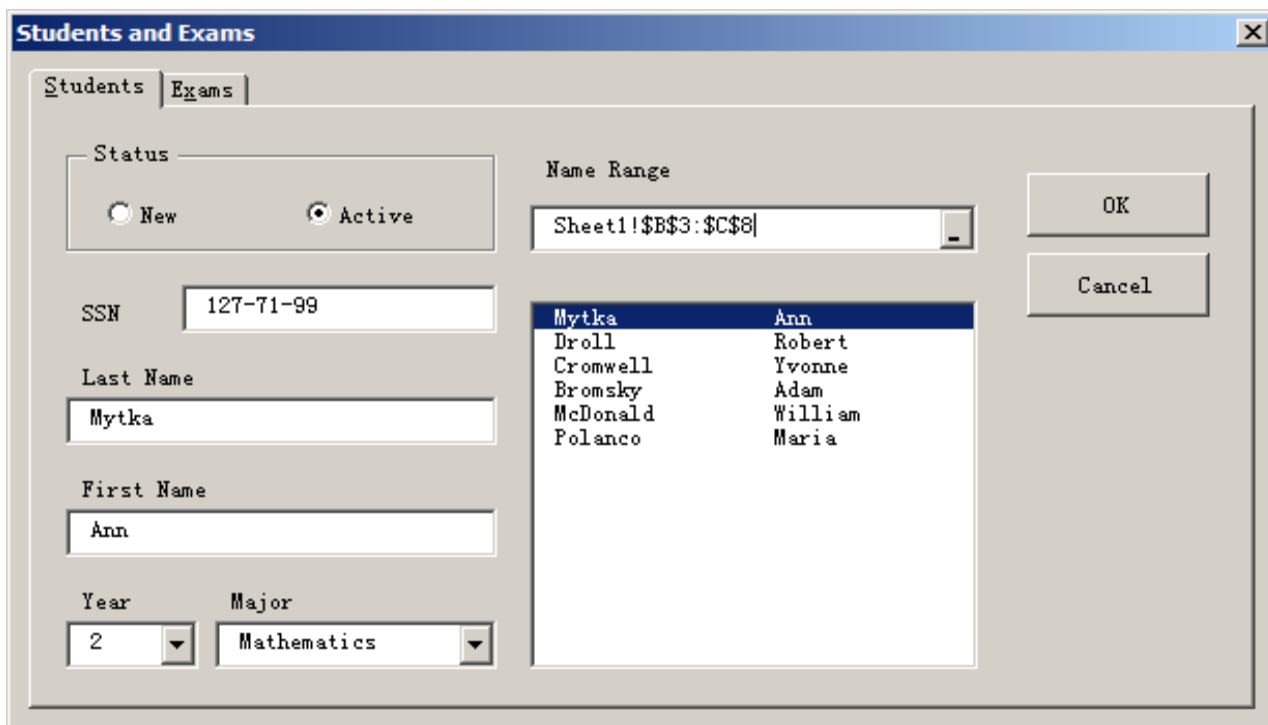


图10-21 窗体上的列表框是通过RefEdit控件将储存在工作表里的数据填充的

6. 点击列表框里的任意姓名，并查看该学生的数据

7. 点击列表框里的任意姓名，然后点击Exams页。

Exams也显示了被选学生（参见图10-22）的姓名。TabStrip控件显示考试科目，如果被选学生参加过任何考试，当你点击适当科目页时，考试的日期和分数就会显示出来。

图10-22 Exams页显示被选学生和科目的考试日期和分数

你可以通过提供的复合框和日历控件，输入或者更改学生的分数和考试日期。VB将问你是否修改数据（回顾Calendar1_Click和cboxGrade_Click过程的VBA代码）在你以确定回应对话框后，所选的数据或者分数就会写入到工作表中相应的列（参见图10-23）

TabStrip1_Change过程确保你点击科目页的时候，VB会从适当的工作表单元格显示考试的分数和考试日期。MultiPage1_Change过程则确保当你点击Exams页时，lblWho标签显示当前列表框里选项学生的姓和名。

	A	B	C	D	E	F	G	H	I	J	K	L	M
1		Display Form				English		French		Math		Physics	
2	SSN	Last Name	First Name	Year	Major	Grade	Date	Grade	Date	Grade	Date	Grade	Date
3	127-71-99	Mytka	Ann	2	Mathematics	A	2006-5-1	C	2015-5-1			B	2011-5-1
4	890-77-34	Droll	Robert	4	Linguistics								
5	234-12-34	Cromwell	Yvonne	1	English			C	2006-5-1			B	2004-5-1
6	453-44-65	Bromsky	Adam	3	Chemistry								
7	142-92-12	McDonald	William	1	Computer S	B	2006-5-1			A	2002-5-1		
8	429-77-54	Polanco	Maria	2	Linguistics								

图10-23 工作表F到M列里的数据是通过用户窗体Students and Exams上Exams页输入的

40.接下来……

既然你到了这个相对比较长的章节的结尾，那么你已经有了必要的技巧来设计有用的窗体。我们来简单总结一下你在本章学习的内容。内置对话框可以从你自己的VBA过程里显示，对于需要用户输入的自定义VBA应用程序，就需要创建一个自定义窗体。通过设置tab键顺序，确保用户可以窗体上按逻辑顺序移动。在窗体模块里面编写VBA过程，让窗体对用户操作作出反应。通过使用属性窗口或者编写UserForm_Initialize过程，给控件设置初始值。确保有过程将数据转移到工作表。在下章里，当你开发集合和自定义对象主题的时候，将获得更多于自定义窗体实用的经验。

第十一章 自定义集合和类模块

在第九章，你学习了如果通过使用自动控制（Automation）控制另一个应用程序的对象，回想一下，在创建了对Microsoft Word

10.0 Object Library的引用之后，你就能够控制Word应用程序，调用其对象、属性和方法。你也学习了如何使用自动控制从Microsoft Outlook获取联系地址。有个好消息，那就是，你不必局限于使用Excel的内置对象或者其它应用程序的对象，VBA允许你创建你自己的对象和对象集合，以及它们完整的方法和属性。在本章，你将学习如何使用集合，包括如何声明自定义集合对象。你也将学习如何使用类模块来创建用户定义的对象。

在跳入这些理论和实用的实例之前，我们来过一下一些本章将用到的术语吧：

集合 (Collection) —— 一个包含一组相关对象的对象

类 (Class) —— 对象的定义，包含其名称、属性、方法和事件。类充作一种对象模版，在允许的时候，由此创建对象示例。

示例 (Instance) —— 术语类的一种特定对象，称为类的示例。当你创建一个示例的时候，你也就创建了一个新对象，它拥有类定义的属性和方法。

类模块 (Class Module) —— 包含类定义的模块，包括它的属性和方法定义

模块 —— 模块含有Sub（子过程）和Function（函数）过程，可为其它VBA过程使用，并且和任何对象没有特别的关系。

窗体模块 —— 包含给由用户窗体或者其控件引发的事件过程使用的VBA代码。窗体模块是一种类模块。

事件 —— 一种可以为对象识别的对象，例如鼠标点击或者按键，你可以为其定义应对操作。事件可以由用户操作，或者VBA语句或者系统引发。

事件过程 —— 一个可以自动执行的过程，是对用户引发的事件或者系统引发的程序代码的反应。

1.使用集合

一组相类似的对象成为集合。例如，在Excel里，所有打开了的工作簿属于Workbooks集合，而某个具体工作簿里面的所有工作表都是Worksheets集合里面的成员。在Word里，所有打开的文档都属于Documents集合，一个文档里的每个段落都是Paragraphs集合的成员。集合是包含其它对象的对象。无论你想要使用什么集合，你都可以做下述事情：

- 使用索引值可以引用集合里的特定对象，例如，要指向Worksheets集合里的第二个对象的话，那么使用下述语句的任意一条：

```
Worksheets(2).Select
```

或者

```
Worksheets("Sheet2").Select
```

- 使用Count属性可以知道集合里的成员数目，例如，当你在立即窗口里输入语句：

```
?Worksheets.Count
```

VBA将会返回当前工作簿里的工作表总数

- 使用Add方法可以在集合里插入新的项目，例如，当你在立即窗口里输入语句：

```
Worksheets.Add
```

VBA将会在当前工作簿里面插入一个新的工作表，这时，Worksheets集合里多了一个成员。

- 使用For Each...Next循环可以遍历集合里的每个对象。假设你打开了一个工作簿，包含五个工作表，它们的名称为：“Daily wages”，“Weekly wages”，“Bonuses”，“Yearly salary”和“Monthly wages”。使用下述过程将名称里包含“wages”的工作表删除：

```
Sub DeleteSheets()  
    Dim ws As Worksheet  
    Application.DisplayAlerts = False  
    For Each ws In Worksheets  
        If InStr(ws.Name, "wages") Then  
            ws.Delete  
        End If  
    Next  
End Sub
```

当你编写你自己的VBA过程时，你可能会碰到这样一种情况，那就是，没有方便的内置集合来处理你的任务。解决办法就是创建自定义集合。从第七章，你就已经知道如何通过动态或静态数组来实用多个数据。因为，集合有允许你添加、移动和计算其成员的内置属性和方法，所以使用集合比使用数组容易得多。

2. 声明自定义集合

要创建一个用户定义的集合的话，你应该先声明一个Collection类型的对象变量，该变量在Dim语句里和关键字New一起声明，如下所示：

```
Dim 集合名称 As New Collection
```

3. 给自定义集合添加对象

在声明Collection对象后，你就可以使用Add方法往集合里插入新成员了。用来组成该集合的对象不必需要是同样的数据类型。Add方法如下所示：

```
object.Add item, key, before, after
```

你只需要明确对象和成员，object是集合名称，它是使用在Collection对象声明中的相同名称；item是你添加到集合里的对象。尽管其它的参数是可选的，但是它们也很有用。集合里的成员自动从1开始分配号码，了解这个很重要。然而，它们也可以给分配一个独特的键。除了通过索引号（1，2，3等等）访问某个特定的成员之外，你也可以在集合添加对象的时候给该对象分配一个键。例如，如果你创建一个自定义工作表集合，那么你应该使用工作表名称作为键；要鉴别学生或者员工集合里的单个人员的话，你就可以使用社会保险号码（SSN）作为他们的键。

如果你想要确定对象在集合里面的位置时，那么你就应该使用before或者after参数（不要同时使用它们）。参数before在此之前添加新对象的对象，而参数after是一个对象，在它之后添加新的对象。

接下来过程GetComments声明了一个叫做colNotes的自定义集合，该过程提示输入作者姓名，然后在活动工作簿里遍历所有的工作表，以找到该作者的批注。只有某个特定作者输入的批注会加入自定义集合。过程给第一个批注分配一个键，然后将剩余的批注添加到集合里，每次都把它们置于最后添加的批注之前（注意before参数的使用）。如果集合至少有一个批注，那么过程会在一个信息框显示由参数key确定的批注内容。注意，参数key如何用来引用集合里的成员。然后，过程将集合里的所有批注打印到立即窗口。文本函数（Mid和Len）用来仅获取批注内容，而排除作者姓名。接着，返回工作簿里的批注数目和自定义集合里的批注数目到Count属性。在试验过程GetComments之前，我们先得按照以下步骤创建一个工作簿：

1. 打开一个新工作簿并保存为Chap11.xls
2. 在工作表Sheet1的任意单元格上单击右键，并从快捷菜单上选择插入批注，随意输入一些文本。在批注框之外的任意地方点击一下，退出批注编辑模式。使用相同的技巧在工作表Sheet2再插入两个批注，给每个批注输入不同的文本。在当前工作簿里插入新工作表（Sheet4），并插入一个批注。现在你应该在三个工作表里一共有四个批注。
3. 选择“工具”|“选项”并且点击“常规”页，用户名文本框里面应该显示的是你的名字，删除你自己的名字，并输入Joan Smith，并点击确定。现在，在工作表Sheet2和Sheet4的任意地方各输入一个批注。这些批注应该会自动地印上Joan Smith的名字。当你输入完批注后，回到选项对话框，并将常规页上的用户名改回你自己的名字。
4. 切换到VB编辑器，并将VBA工程重命名为ObjColClass
5. 在当前工程里添加一新模块，并重命名为MyCollection
6. 输入过程GetComments，显示如下：

```
Sub GetComments()
    Dim sht As Worksheet
    Dim colNotes As New Collection
    Dim myNote As Comment
    Dim I As Integer
    Dim t As Integer
    Dim fullName As String

    fullName = InputBox("Enter author's full name:")
    For Each sht In ThisWorkbook.Worksheets
        sht.Select
        I = ActiveSheet.Comments.Count
        For Each myNote In ActiveSheet.Comments
            If myNote.Author = fullName Then
                MsgBox myNote.Text
                If colNotes.Count = 0 Then
```

```

        colNotes.Add Item:=myNote, key:="first"
    Else
        colNotes.Add Item:=myNote, Before:=1
    End If
End If
Next
t = t + 1
Next
If colNotes.Count <> 0 Then MsgBox colNotes("first").Text
MsgBox "Total comments in workbook: " & t & Chr(13) & _
    "Total comments in collection:" & colNotes.Count
Debug.Print "Comments by " & fullName
For Each myNote In colNotes
    Debug.Print Mid(myNote.Text, Len(myNote.Author) + 2, _
        Len(myNote.Text))
Next
End Sub

```

7. 运行过程GetComments并且查看结果

4.从自定义集合移出对象

从自定义集合里移出成员和添加成员一样简单。使用Remove方法可以移出对象，如下所示：

object.Remove item

object是自定义集合的名称，含有你要移出的对象；item是你要从集合移出的对象。

我们来修改你在前面部分准备的过程GetComments，示范从集合里移出成员。在该过程结尾，我们将当前集合colNotes里成员的内容一个一个地显示出来，并且询问用户，使用应该将该成员从该集合移出。

1. 将下面的声明加入到过程GetComments的声明部分：

```

Dim response
Dim myId As Integer

```

第一条语句声明了一个叫做response的变量，你将使用该变量来储存Msgbox函数的结果。第二条语句声明变量myId来储存集合对象的索引号。

2. 定位过程GetComments的下述语句：

```

For Each myNote In colNotes

```

在上面的语句之后，添加下述语句：

```

myId = 1

```

3. 定位到过程GetComments的下述语句：

```

Debug.Print Mid(myNote.Text, Len(myNote.Author) + 2, _
    Len(myNote.Text))

```

在该语句后，输入下面的代码块：

```

response = MsgBox("Remove this comment?" & Chr(13) _
    & Chr(13) & myNote.Text, vbYesNo + vbQuestion)
If response = 6 Then
    colNotes.Remove Index:=myId
Else
    myId = myId + 1
End If

```

4. 在该过程输入下述语句：

```

Debug.Print "The following comments remain in the collection:"
For Each myNote in colNotes
    Debug.Print Mid(myNote.Text, Len(myNote.Author) + 2, _

```

```
Len(myNote.Text))
```

```
Next
```

5. 运行过程GetComments，并且将显示在信息框里批注之一删除。

修改后的过程GetComments2可以在附带的CD里找到。该过程将指定的批注从自定义集合里清除，但是不会从工作表里删除批注。

技巧11-1 给集合重新编号

当移出对象后，集合回自动重新编号，因此，你可以使用1作为Index参数，来删除自定义集合里的所有对象，如下所示：

```
Do While myCollection.Count >0
    myCollection.Remove Index:=1
Loop
```

插入：模块还是类模块？

在VB编辑器的插入菜单里有两个模块命令：模块和类模块。这些在本章的开头有定义。到目前为止，你已经使用过标准模块来创建Sub和Function过程。你将第一次在本章使用类模块来创建自定义对象并且定义它的属性和方法。

5.创建自定义对象

创建一个新的，非标准的VBA对象，需要在你的工程里插入类模块并且在模块里添加代码。然而，在做此之前，你需要对类是什么要有一个基本的了解。

如果你回头看一下本章的开头，那么你就会知道类从某种意义上说是对象模板。一个经常使用的类比就是将类比作一个曲奇剪切机。就像曲奇剪切机决定某个曲奇做成什么样一样，类的定义决定某个对象应该是什么样以及如何做。在实际使用一个对象类之前，你必须先创建一个类的新示例。对象示例就是这些曲奇。每个示例都有其类定义的特点（属性和方法），正如你可以使用相同的区旗剪切机制作许多曲奇一样，你可以创建多个类的示例。你可以独立改变相同类中的任意示例中的类的每个示例的属性。

类模块允许你定义你自己的自定义类，连同自定义属性和方法。回忆一下，属性是定义对象某个特点的品质，例如外形、位置、颜色、标题等等。方法是对象可以执行的操作。通过在类模块里面编写属性过程，你可以给你的自定义对象创建属性。对象的方法也可以在类模块里面通过编写子过程或者函数过程来创建。在类模块里创建完你的对象后，你可以象使用其它的内置对象一样使用它。你也可以将对象类导出VBA工程给其它能使用VBA的应用程序使用。

6.创建类

本章的剩余部分将示范创建和使用一个叫做CEmployee的自定义对象的过程。该对象将代表一个员工，该CEmployee对象将具有属性例如：Id, FirstName, LastName和Salary。它也有一个方法，可以修改当前薪水。

1. 在工程浏览器窗口选择ObjColClass (Chap11.xls)，并选择“插入”|“类模块”
2. 选择工程浏览器里的类模块，并使用属性窗口将其重命名为CEmployee。

技巧11-2 给类模块命名

每次创建新类模块的时候，请给个它一个有意义的名称。将类模块命名为你想要在你的VBA过程使用的名称。你给你的类选择的名称应该易于理解并且明确对象类要代表的东西。作为一个规则，对象类名称一般前面有一个大写字母“C”。

7.变量声明

在添加和重命名类模块后，下一步就是声明变量，用来存储你要存储在该对象里数据。你想要存储与对象的每个数据都得分配一个变量。类的变量被称为数据成员，并且使用关键字Private声明。该关键字确保这些变量只在该类模块里面可用。使用关键字Private代替常用的Dim语句，隐藏该数据成员并且避免应用程序的其它部分引用它。只有定义这些变量的类模块中的过程才可以修改这些变量的值。

因为这些变量的名称也充作属性名称，所以要给你的对象数据成员使用有意义的名称。传统上使用m_作为变量名称的前缀，以表明它们是类的数据成员。

1. 在类模块CEmployee的顶端输入下述声明：

```
Option Explicit
' declarations
Private m_LastName As String
Private m_FirstName As String
Private m_Salary As Currency
Private m_Id As String
```

注意，每个数据成员变量的名称都以前缀“m_”

8. 定义类的属性

使用关键字Private声明变量保证变量不能从该对象以外直接访问，意思是说，该类模块以外的VBA过程不能设置或者读取存储与这些变量里的数据。要让你VBA应用程序的其它部分也能够设置或者读取员工数据的话，你就必须在CEmployee类模块里面添加特殊的属性过程。这里有三种属性过程：

- Property Let——该过程允许应用程序的其它部分设置属性值。
- Property Get——该过程允许应用程序的其它部分获得或读取属性值。
- Property Set——当设置对对象引用时，该过程用来代替Property Let。

属性过程在一个对象属性需要设置或者读取的时候被执行。Property Get过程可以和Property Let过程拥有一样的名称。你应该给对象的每个可能被VBA应用程序其它部分访问的属性创建属性过程。

三种属性语句中最容易理解的是Property Get过程，我们通过仔细看一下Property Get LastName过程来检查一下属性过程的语法。

属性过程包含下属部分：

- **过程声明行**明确属性名称和数据类型：

```
Property Get LastName ( ) As String
```

LastName是属性名称，As String决定该属性返回值的数据类型

- **赋值语句**，类似于我们在函数过程里使用的：

```
LastName = m_LastName
```

LastName是属性名称，而m_LastName是数据成员变量，它存储着你想要获取或者设置的属性值。m_LastName变量应该在类模块的顶端使用关键字Private定义好了。

如果获取的数据是一个计算的结果，那么你可以包含适当的VBA语句：

```
Property Get Royalty()
    Royalty = (Sales * Percent)-Advance
End Property
```

- End Property关键字表明属性过程结束

技巧11-3 从属性过程立即退出

正如关键字Exit Sub和Exit Function允许你提前退出子过程或者函数过程一样，Exit Property关键字同样也给了立即退出属性过程的方法。程序执行将会依照Property Get，Property Let或者Property Set过程语句继续下去（译者，原文不可理解，可能有误）

9. 创建 Property Get 过程

CEmployee类对象有四个属性需要给当前VBA工程里的其它模块的VBA程序使用，使用对象CEmployee时，你肯定想要获得员工的ID，姓和名以及当前薪水的信息。

1. 在类模块CEmployee里输入下述Property Get过程，紧接着声明部分输入：

```
Property Get Id( ) As String
    Id = m_Id
End Property
```

```
Property Get LastName( ) As String
```



```

        LastName = m_LastName
End Property

Property Get FirstName() As String
    FirstName = m_FirstName
End Property

Property Get Salary() As Currency
    Salary = m_Salary
End Property

```

每种员工必须的信息要求一个单独的Property Get过程，上面的每个Property Get过程返回当前属性的值。注意，Property Get过程和函数过程非常类似。就像函数过程，Property Get过程包含一个赋值语句。回忆一下第四章的内容，要从函数过程返回值的的话，你就必须将该值赋予该函数名称。

10.创建 Property Let 过程

除了使用Property Get过程来获取存储在数据成员（私有变量）里的数据之外，你必须准备相应的Property Let过程，以允许其它过程在必要的时候可以更改这些变量的值。然而，如果存储在私有变量的某个数据就是只读的话，那么你就不必准备Property Let过程了。假设你不希望用户更改员工ID，你只要不给它写Property Let过程，就可以使它为只读的了。因此类模块CEmployee将只有三个属性（LastName，First- Name和Salary），每个属性需要一个单独的Property Let过程。

1. 在类模块CEmployee里输入下述Property Let过程：

```

Property Let LastName(L As String)
    m_LastName = L
End Property

Property Let FirstName(F As String)
    m_FirstName = F
End Property

Property Let Salary(ByVal dollar As Currency)
    m_Salary = dollar
End Property

```

Property Let过程至少需要一个参数来明确你想要赋予该属性的数值，该参数可以按值传递（参见Property Let Salary过程里的关键字ByVal）或者按引用传递（ByRef是默认的）。如果你需要重新熟悉一下这些关键字的意思的话，那么参加第四章里的“按值或者按引用传递参数”部分。传递给Property Let过程的参数数据类型必须和同名称的Property Get 或者Set过程返回值的数据类型一致。注意，Property Let过程拥有前面部分准备的Property Get过程相同的名称。忽略了ID属性的Property Let过程，你将创建一个只读的ID属性，只能读取，不能设置。

技巧11-4 定义属性过程的范围

你可以将关键字Public，Private或Static放在属性过程名称的前面来确定它的范围。例如：
要创建一个可以从所有模块的过程里访问的Property Get过程的话，那么依照下属语句格式：

```
Public Property Get FirstName() As String
```

要让Property Get过程只能从声明它的模块中其它过程访问的话，那么使用下述语句格式：

```
Private Property Get FirstName() As String
```

在过程调用中，要保护Property Get过程的当地变量的话，那么可以使用下述语句格式：

```
Static Property Get FirstName() As String
```

如果没有明确使用Public或者Private，熟悉过程默认上就会使公共的，同样，如果没有使用关键字Static，那么当地变量在过程的调用中就不会被保护。

11.创建类方法

除了属性之外，对象通常还一个或者多个方法。方法是该对象可以执行的操作。方法允许你操作存储与类对象里的数据。方法可以使用自过程或者函数过程创建。要让方法在类模块之外可用的话，那么需要在自过程或函数过程定义前带关键字Public。你在本章创建的对象CEmployee有一个方法，让你计算新薪水。假设员工的薪水可以按照一个确定的百分比或值来增加或者减少。

1. 在类模块里输入下述函数过程CalcNewSalary:

```
Public Function CalcNewSalary(choice As Integer, _
    curSalary As Currency, amount As Long) As Currency
    Select Case choice
        Case 1
            ' by percent
            CalcNewSalary = curSalary + ((curSalary + amount)/100)
        Case 2
            ' by amount
            CalcNewSalary = curSalary + amount
    End Select
End Function
```

函数CalcNewSalary在类模块里使用关键字Public一起定义，作为类CEmployee的一个方法。要计算一个新的薪水的话，类模块之外的VBA过程必须提供三个参数：choice，CurSalary和amount。参数choice明确计算类型，假设你想要按5个百分点或者5美元增加员工的薪水，选择1将按5个百分点增加薪水，而选择2将当前的薪水基础上增加5美元。参数CurSalary是员工的当前薪水数字，而amount决定了薪水改变量。

技巧11-5 关于类方法

- 只有在类模块之外可以访问的方法应该声明为Public，所有其它的应该为Private
- 方法执行一些类包含的数据的操作
- 如果方法需要返回数值的话，那么就编写一个函数过程，否则创建一个子过程

12.创建类的示例

在类里输入完所有必须的Property Get，Property Let，自过程和函数过程后，你可以创建一个类的信示例了，称为对象。在你能够创建对象之前，对象变量必须在一个标准模块里声明好，以存储对该对象的引用。如果该类模块的名称为CEmployee的话，该类的新示例就可以使用下述语句创建：

```
Dim emp As New CEmployee
```

变量emp代表从类CEmployee的一个对象的引用。当你使用关键字New声明对象变量的时候，VBA就会创建该对象并分配内存给它；然而，该对象并没有获得示例，直到你在你的程序代码里通过赋值给它的属性或者执行它的方法创建对该对象的引用。你同样可以通过声明一个对象变量为类定义的数据类型来创建该对象的一个示例。例如：

```
Dim emp As CEmployee
Set emp = New CEmployee
```

如果你不使用关键字New在Dim语句里（如上所示），那么VBA就不会份哦内存给你的自定义对象，直到你的程序真正需要它。

13.类模块里的事件过程

事件基本上是一个对象可以识别的操作。自定义类只可识别两种事件：Initialize和Terminate。这两个事件分别在该类的示例创建和消灭的时候引发。

Initialize事件在从类创建对象的时候产生（参见前面部分关于“创建类的示例”）。在类CEmployee例子里面，Initialize事件在你代码里第一次使用变量emp的时候也会引发。因为Initialize事件里的语句将是该对象第一个要执行的，在任何属性被赋值之前，也在任何方法被执行之前，所以Initialize事件是一个执行类创建的对象初始化最好的地方。回忆一下，在类模块CEmployee里，ID是只读的，你可以使用Initialize事件给m_Id变量赋予一个单独的五位数。

1. 在类模块CEmployee里输入下述过程Class_Initialize:

```
Private Sub Class_Initialize()  
    Randomize  
    m_Id = Int((99999 - 10000) * Rnd + 10000)  
End Sub
```

Class_Initialize过程给变量m_Id赋予一个独特的五位数，初始化对象CEmployee。使用下述公式，可以产生一个介于起始值10000和结束值99999之间的随机数：

```
=Int((结束值 - 起始值)*Rnd + 起始值)
```

Class_Initialize过程也使用了Randomize语句来初始化随机数发生器。可以搜索在线帮助，获得更多关于使用Rnd和Int函数，以及Randomize语句的信息。

Terminate事件发生在释放该对象的引用时。这是一个执行任何必要的清理任务的好地方。Class_Terminate过程使用下述语法：

```
Private Sub Class_Terminate()  
    [你的清理代码]  
End Sub
```

使用下述语法，将对象变量从对象上释放出来：

```
Set objectVariable = Nothing
```

当你设置对象变量为Nothing的时候，Terminate事件就发生了，届时，任何位于该事件里的代码就会被执行。

14.创建用户界面

如果你跳过了前面的章节的话，那么你可能得返回去，因为，执行你的自定义对象CEmployee需要你设计一个自定义窗体。

1. 选中当前VBA工程，并且选择“插入” | “用户窗体”
2. 按照图11-1所示准备好该窗体：

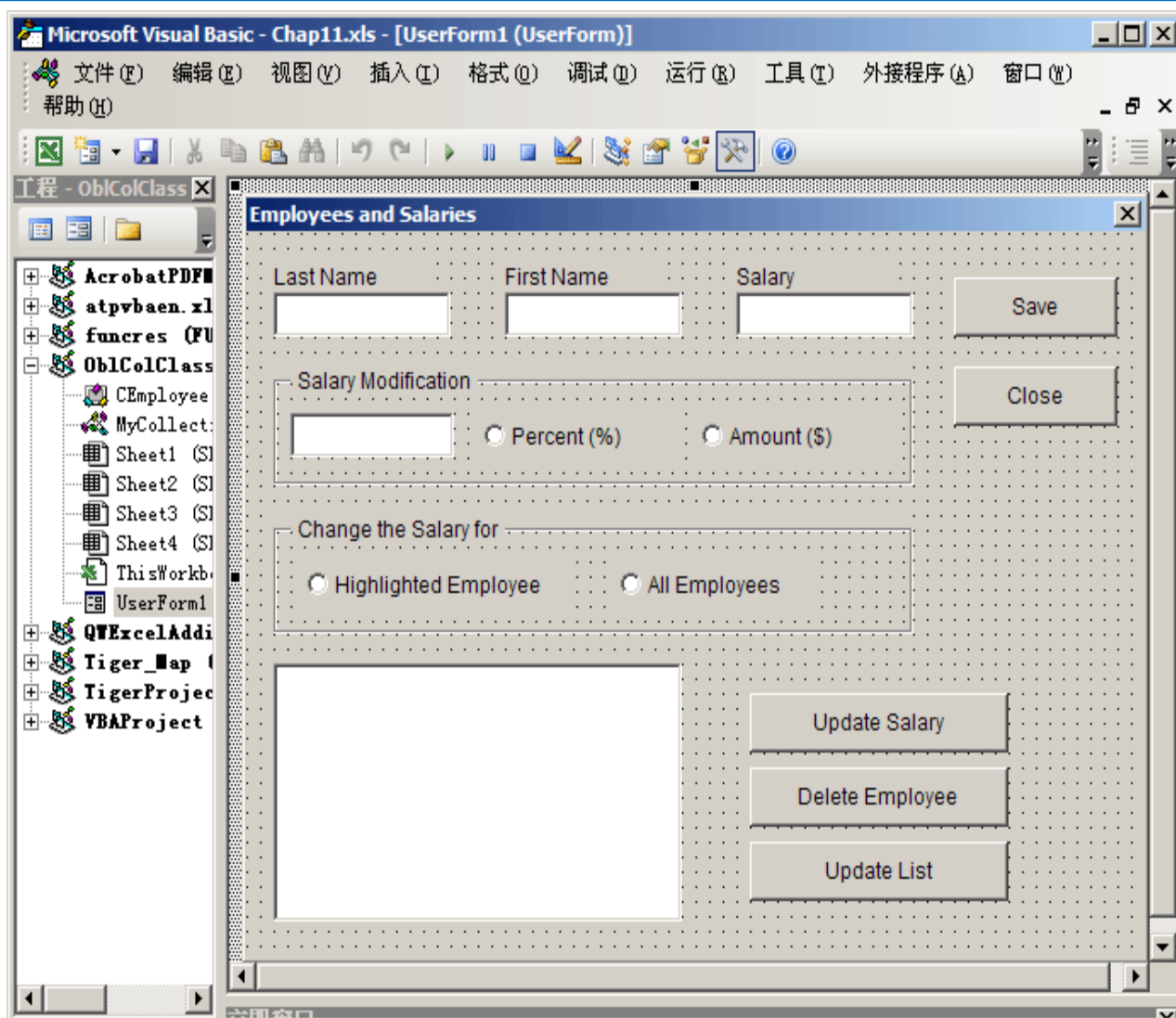


图11-1 本窗体示范了自定义对象CEmployee的使用

3. 给窗体和它的控件设置下述属性：

对象	属性	设置
窗体	Name	Salaries
	Caption	Employees and Salaries
标签1	Caption	Last Name
标签Last Name 下面的文字框	Name	txtLastName
标签2	Caption	First Name
标签First Name 下面的文字框	Name	txtFirstName
标签3	Caption	Salary
标签Salary 下面的文字框	Name	txtSalary
框架1	Caption	Salary Modification
框架Salary Modification下面的文字框	Name	txtRaise
选项按钮1	Name	optPercent
	Caption	Percent (%)
选项按钮2	Name	optAmount
	Caption	Amount (\$)
框架2	Caption	Change the Salary for

选项按钮3	Name	optHighlighted
	Caption	Highlighted Employee
选项按钮4	Name	optAll
	Caption	All Employees
列表框	Name	lboxPeople
	Height	91.45
	Width	180.75
命令按钮1	Name	cmdSave
	Caption	Save
命令按钮2	Name	cmdClose
	Caption	Close
命令按钮3	Name	cmdUpdate
	Caption	Update Salary
命令按钮4	Name	cmdDelete
	Caption	Delete Employee
命令按钮5	Name	cmdEmployeeList
	Caption	Update List

4. 准备一个数据输入工作表，如图11-2所示：

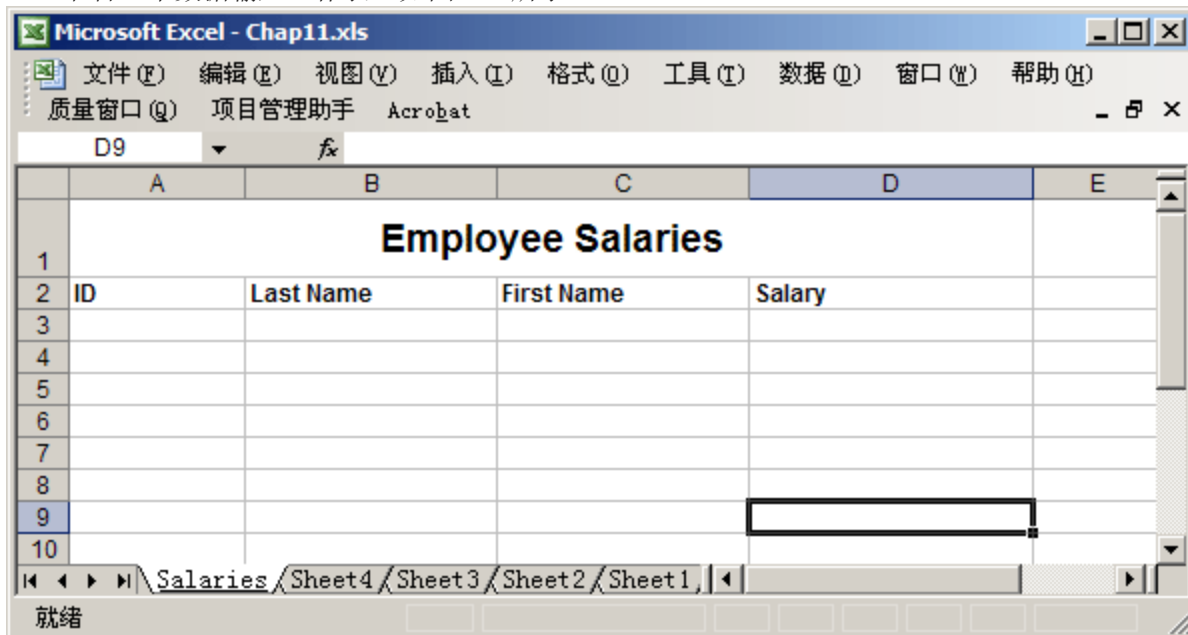


图11-2 在窗体Employees and Salaries上输入的数据将会转移到该工作表

5. 切换到VB编辑器窗口，双击窗体背景以激活窗体模块
6. 在窗体模块代码窗口上部输入下述声明：

```
Option Explicit
Dim emp As New CEmployee
Dim CEmployees As New Collection
Dim index As Integer
Dim ws As Worksheet
Dim extract As String
Dim cell As Range
Dim lastRow As Integer
Dim empLoc As Integer
Dim startRow As Integer
```

```
Dim endRow As Integer
Dim choice As Integer
Dim amount As Long
```

第一条语句声明变量emp为类CEmployee的一个新示例，第二条语句声明了一个自定义集合，集合CEmployees将会用来存储员工数据。这里声明的其它变量将会用于窗体上各种控件的VBA过程里面。

7. 输入下述UserForm_Initialize过程来激活或者禁止窗体上的控件：

```
Private Sub UserForm_Initialize()
    txtLastName.SetFocus
    cmdEmployeeList.Visible = False
    lboxPeople.Enabled = False
    Frame1.Enabled = False
    txtRaise.Value = ""
    optPercent.Value = False
    optAmount.Value = False
    txtRaise.Enabled = False
    optPercent.Enabled = False
    optAmount.Enabled = False
    Frame2.Enabled = False
    optHighlighted.Enabled = False
    optAll.Enabled = False
    cmdUpdate.Enabled = False
    cmdDelete.Enabled = False
End Sub
```

当窗体启动时，UserForm_Initialize过程内部的语句只会激活需要的控件（见图11-3）。

图11-3 当窗体第一次启动时，UserForm_Initialize过程禁用某些控件

8. 输入下述过程cmdSave_Click将输入在窗体的数据转移到工作表：


```

Private Sub cmdSave_Click()
    If txtLastName.Value = "" Or txtFirstName.Value = "" Or _
        txtSalary.Value = "" Then
        MsgBox "Enter Last Name, First Name and Salary."
        txtLastName.SetFocus
        Exit Sub
    End If
    If Not IsNumeric(txtSalary) Then
        MsgBox "You must enter a value for the Salary."
        txtSalary.SetFocus
        Exit Sub
    End If
    If txtSalary < 0 Then
        MsgBox "Salary cannot be a negative number."
        Exit Sub
    End If
    Worksheets("Salaries").Select
    index = ActiveSheet.UsedRange.Rows.Count + 1
    lboxPeople.Enabled = True

    'set and enter data into the CEmployees collection
    With emp
        Cells(index, 1).Formula = emp.Id
        .LastName = txtLastName
        Cells(index, 2).Formula = emp.LastName
        .FirstName = txtFirstName
        Cells(index, 3).Formula = emp.FirstName
        .Salary = CCur(txtSalary)
        If .Salary = 0 Then Exit Sub
        Cells(index, 4).Formula = emp.Salary
        CEmployees.Add emp
    End With

    'delete data from text boxes
    txtLastName = ""
    txtFirstName = ""
    txtSalary = ""

    'enable hidden controls
    cmdEmployeeList.Value = True
    cmdEmployeeList.Visible = True
    cmdUpdate.Enabled = True
    cmdDelete.Enabled = True
    Frame1.Enabled = True
    txtRaise.Enabled = True
    optPercent.Enabled = True
    optAmount.Enabled = True
    Frame2.Enabled = True
    optHighlighted.Enabled = True
    optAll.Enabled = True

```

```
txtLastName.SetFocus
```

```
End Sub
```

cmdSave_Click过程以验证用户的姓、名和薪水文字框开始，如果用户输入了正确的数据，VBA将当前工作表里的第一空白行赋予变量Index。下一条语句激活窗体的列表框控件。

当程序到达With emp结构时，类CEmployee的一个新示例便产生了。属性LastName，FirstName和Salary的设置基于相应文字框里输入的数据，而ID属性则是由Class_Initialize事件过程里的随机数语句产生的数字设置的。VBA每次看到对对象emp的引用时，它就会调用位于类模块里的适当Property Let过程。

本章的最后部分示范如何一步一步地跟踪过程的运行，准确地查看什么时候运行属性过程。设置完对象的属性值后，VBA将员工数据转移到工作表里。With emp结构里面的最后一条语句将用户定义的对象emp添加到一个叫做CEmployees的自定义集合。

接着，VB将窗体文字框里的输入清除并且激活开始在UserForm_Initialize过程里关闭的命令按钮。注意，本代码块的第一条指令：cmdEmployeeList.Value=True，该语句导致自动执行cmdEmployeeList_Click过程，该过程附加于按钮UpdateList（顺便说一下，这是唯一用户从未见到的控件）。该过程的代码如下所示。

9. 输入cmdEmployeeList_Click过程，如下所示：

```
Private Sub cmdEmployeeList_Click()
```

```
lboxPeople.Clear
```

```
For Each emp In CEmployees
```

```
lboxPeople.AddItem emp.Id & ", " & _
```

```
emp.LastName & ", " & emp.FirstName & ", $" & _
```

```
Format(emp.Salary, "0.00")
```

```
Next emp
```

```
End Sub
```

cmdEmployeeList_Click过程附加在命令按钮UpdateList之上，该按钮由cmdSave_Click过程控制，并且导致新的员工数据添加到列表框控件里。cmdEmployeeList_Click过程以清除列表框的内容开始，然后用自定义集合CEmployees的成员来填充列表框。

图11-4 列表框控件显示员工数据，正如在自定义集合输入的一样

10. 输入下述过程cmdClose_Click:

```
Private Sub cmdClose_Click()
    Unload Me
End Sub
```

cmdClose_Click过程让你将用户窗体从屏幕上清除, 并结束使用员工的自定义集合。当你再次运行窗体, 你输入的员工将会成为新集合CEmployees的成员。

11. 输入下述过程cmdDelete_Click:

```
Private Sub cmdDelete_Click()
    ' make sure that an employee is highlighted in the
    ' list control
    If lboxPeople.ListIndex > -1 Then
        MsgBox "Selected item number: " & lboxPeople.ListIndex
        extract = CEmployees.Item(lboxPeople.ListIndex + 1).Id
        MsgBox extract
        Call FindId
        MsgBox empLoc
        Range("A" & empLoc).Delete (3)
        MsgBox "There are " & CEmployees.Count & _
            " items in the CEmployees collection. "
        CEmployees.Remove lboxPeople.ListIndex + 1

        MsgBox "The CEmployees collection has now " & _
            CEmployees.Count & " items."
        cmdEmployeeList.Value = True
        If CEmployees.Count = 0 Then
            Call UserForm_Initialize
        End If
    Else
        MsgBox "Click the item you want to remove."
    End If
End Sub
```

过程cmdDelete_Click让你从自定义集合CEmployees里面清除员工。要删除员工的话, 你必须点击列表框的适当成员。当你点击一个列表成员 (和按钮Delete Employee), cmdEmployeeList_Click过程就自动执行。该过程确保更新列表框的内容。员工将同时从集合和列表框里删除。如果列表框里面只有一个员工, 那么VBA将调用过程UserForm_Initialize在清除最后一个员工后将某些控件失活。cmdDelete_Click过程里有好几个MsgBox语句, 让你在做决定的时候检查列表框控件的内容。除了从自定义集合里删除员工之外, 过程cmdDelete_Click也必须从工作表的相应行删除员工信息。使用函数FindId可以很方便地在工作表里找到员工数据 (该过程的代码见下面的第12步)。该函数将要删除的行号返回到过程cmdDelete_Click。

12. 输入下述函数过程:

```
Private Function FindId()
    Set ws = ActiveWorkbook.Sheets("Salaries")
    startRow = ActiveSheet.UsedRange.Rows.Count + _
        1 - CEmployees.Count
    endRow = ActiveSheet.UsedRange.Rows.Count
    For Each cell In ws.Range(Cells(startRow, 1), _
        Cells(endRow, 1))
        If cell.Value = extract Then
            empLoc = cell.Row
            FindId = empLoc
        End If
    Next cell
    Exit Function
End Function
```

```
End If
```

```
Next
```

```
End Function
```

函数过程FindId将含有窗体列表框中当前选择的员工数据的行号返回到主调过程。工作表中的数据搜索基于变量extract的内容，它存储员工的唯一号码。员工ID的搜索被限制与工作表的第一列，并从集合的第一个成员放置的位置开始搜索，这样使搜索更快一些。你不要在工作表整个使用的区域里搜索。想想如果你不只一次地使用了窗体，但是，自定义集合的内容不会包含前面输入的员工。

13. 输入下述过程cmdUpdate_Click:

```
Private Sub cmdUpdate_Click()
```

```
    If optHighlighted = False And optAll = False Then
```

```
        MsgBox "Click the 'Highlighted Employee' or " & " 'All Employees' option button."
```

```
    Exit Sub
```

```
End If
```

```
    If Not IsNumeric(txtRaise) Then
```

```
        MsgBox "This field requires a number."
```

```
        txtRaise.SetFocus
```

```
    Exit Sub
```

```
End If
```

```
    If optHighlighted = True And _
```

```
        lboxPeople.ListIndex = -1 Then
```

```
        MsgBox "Click the name of the employee."
```

```
    Exit Sub
```

```
End If
```

```
    If lboxPeople.ListIndex <> -1 And _
```

```
        optHighlighted = True And _
```

```
        optAmount.Value = True And _
```

```
        txtRaise.Value <> "" Then
```

```
        extract = CEmployees.Item(lboxPeople.ListIndex + 1).Id
```

```
        MsgBox extract
```

```
        Call FindId
```

```
        MsgBox empLoc
```

```
        choice = 2
```

```
        amount = txtRaise
```

```
        CEmployees.Item(lboxPeople.ListIndex + 1).Salary = _
```

```
            emp.CalcNewSalary(choice, _
```

```
            CEmployees.Item(lboxPeople.ListIndex + 1).Salary, amount)
```

```
        Range("D" & empLoc).Formula = CEmployees._
```

```
            Item(lboxPeople.ListIndex + 1).Salary
```

```
        cmdEmployeeList.Value = True
```

```
    ElseIf lboxPeople.ListIndex <> -1 And _
```

```
        optHighlighted = True And _
```

```
        optPercent.Value = True And _
```

```
        txtRaise.Value <> "" Then
```

```
        extract = CEmployees.Item(lboxPeople.ListIndex + 1).Id
```

```
        MsgBox extract
```

```
        Call FindId
```

```
        MsgBox empLoc
```

```
        CEmployees.Item(lboxPeople.ListIndex + 1).Salary = _
```

```

        CEmployees.Item(lboxPeople.ListIndex + 1).Salary = _
        (CEmployees.Item(lboxPeople.ListIndex + 1).Salary * _
        txtRaise / 100)
Range("D" & empLoc).Formula = CEmployees. _
    Item(lboxPeople.ListIndex + 1).Salary
cmdEmployeeList.Value = True
ElseIf optAll = True And _
    optPercent.Value = True And _
    txtRaise.Value <> "" Then

    For Each emp In CEmployees
        emp.Salary = emp.Salary + ((emp.Salary * txtRaise) _
            / 100)
        extract = emp.Id
        MsgBox extract
        Call FindId
        MsgBox empLoc
        Range("D" & empLoc).Formula = emp.Salary
    Next

    emp cmdEmployeeList.Value = True
ElseIf optAll = True And _
    optAmount.Value = True And _
    txtRaise.Value <> "" Then

    For Each emp In CEmployees
        emp.Salary = emp.Salary + txtRaise
        extract = emp.Id
        MsgBox extract
        Call FindId
        MsgBox empLoc
        Range("D" & empLoc).Formula = emp.Salary
    Next emp

    cmdEmployeeList.Value = True
Else
    MsgBox "Enter data or select an option."
End If
End Sub

```

有了过程cmdUpdate_Click,你就可以使用确定的百分比或者特定量来修改薪水。可以为选定的员工或者列表框和集合里面列出的所有员工更新薪水。cmdUpdate_Click过程核实用户是否选择了适当的选项按钮,然后在文字框里输入增加数字。取决于你选择了哪个选项按钮,给某个员工或者所有员工更新的薪水量可以是按照百分比,也可以是按照某个固定的量。薪水的更改也会反映在工作表里。图11-15显示James Nolan的薪水,以按百分之十增加了该薪水。在文字框里面输入负数,你可以按照特定的百分比或者量减少薪水。

图11-5 可以按照特定的百分比或者固定量增加或者减少员工的薪水

14. 选择“插入”|“模块”在当前工程里插入一个标准模块，重命名该模块为WorkAndPay，输入下述过程来显示窗体Employees and Salaries:

```
Sub ClassDemo( )
    Salaries.Show
End Sub
```

15. 运行过程ClassDemo，使用自定义类。

你可以点击窗体的背景并且按下F5，运行窗体Salaries，或者你也可以在工作表里放置一个按钮，并将过程ClassDemo指定给它（参见第十章，如何将按钮放在工作表里）。

15.观察 VBA 过程的执行

为了帮助你理解代码运行时会发生什么，以及你的自定义对象如何工作，我们来逐步运行过程cmdSave_Click。本练习也可以说是第十三章里将包括的调试技巧的简单介绍。

1. 在工程浏览窗口，选择Salaries窗体，并点击窗口上面的查看代码按钮
2. 出现Salaries代码窗口时，从代码窗口左上角的复合框里选择过程cmdSave
3. 在下述代码行的左边框上点击一下，设置断点：

```
If txtLastName.Value = "" Or txtFirstName.Value = "" Or _
txtSalary.Value = "" Then
```

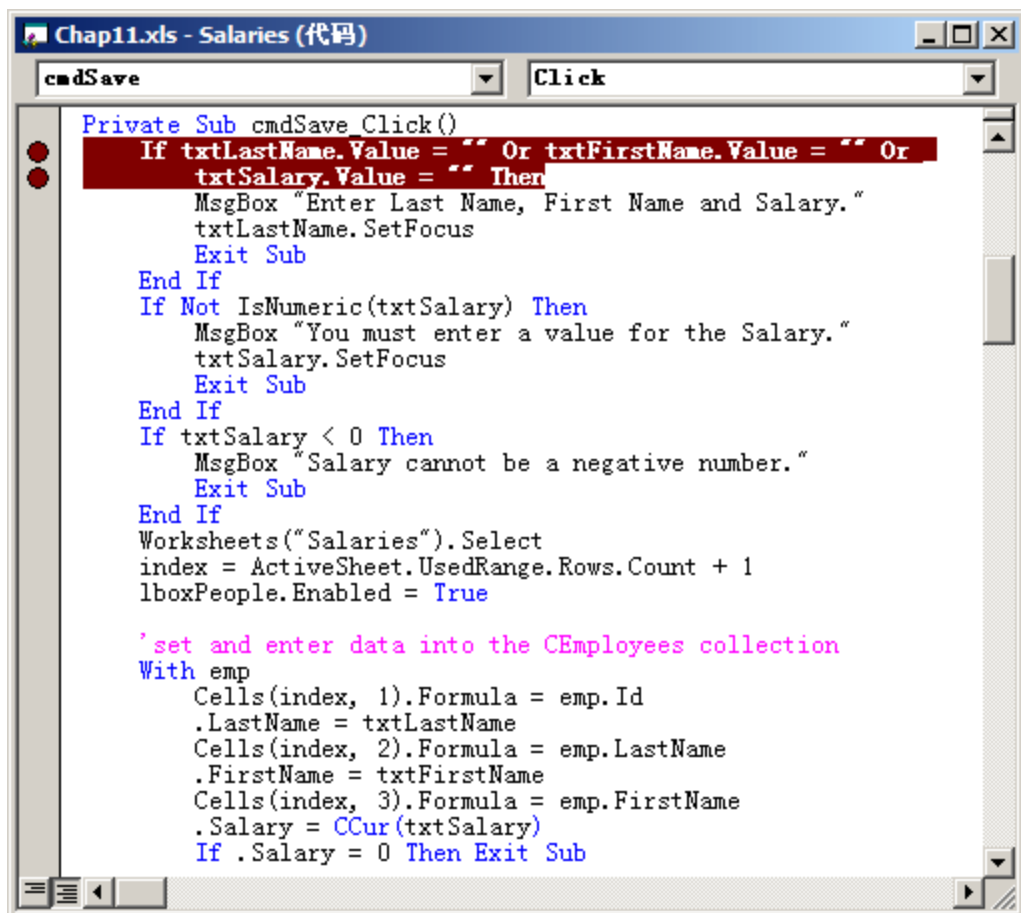



图11-6 边框上的红色圆圈代表断点。当VBA遇到带断点的语句时，它就会自动切换到代码窗口并且显示该白字红底的文本。

4. 在工程浏览窗口，选中模块WorkAndPay并点击查看代码按钮
5. 将光标放在过程ClassDemo里的任意位置，并且按下F5，或者选择“运行”|“运行子过程/用户窗体”
6. 当窗体出现时，在Last Name, First Name和Salary文字框里输入数据，然后点击窗体的按钮Save。现在VB将切换到代码窗口，因为它碰到了过程cmdSave_Click第一行的断点。

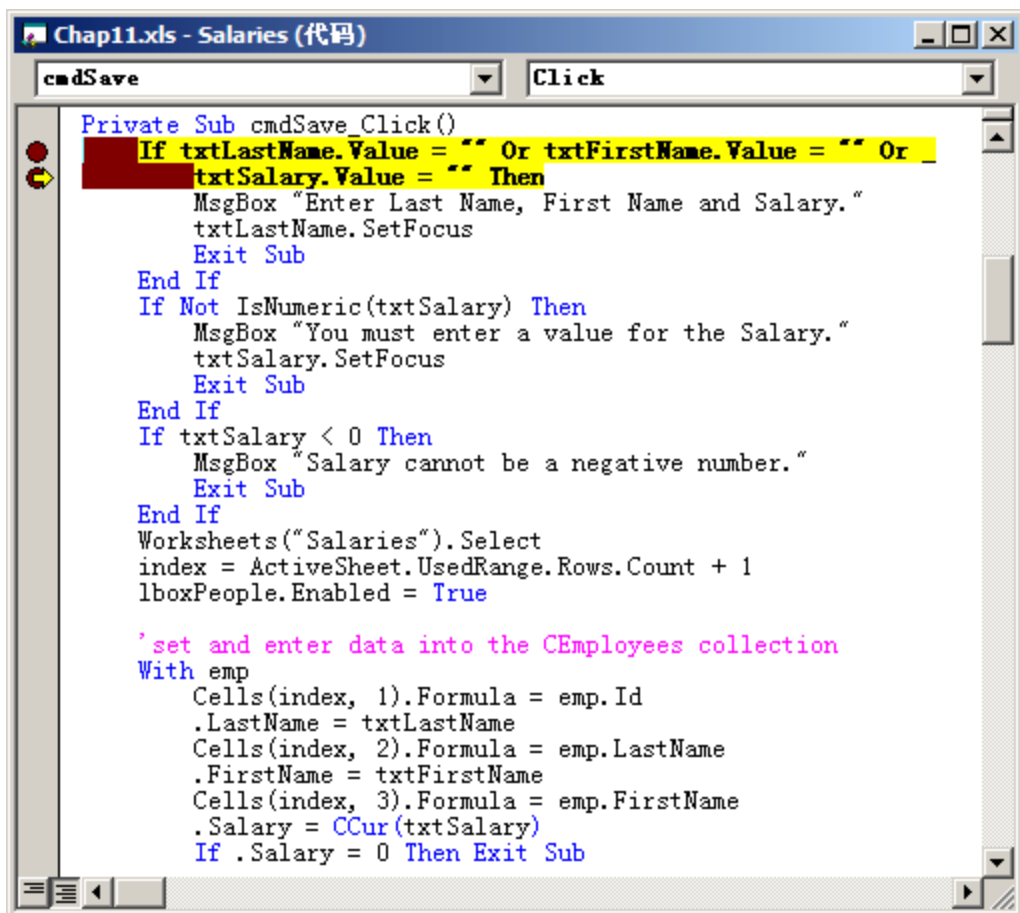


图11-7 当VB运行过程遇到断点时，它会切换到代码窗口，并在中断过程的语句的左边边框上显示一个黄色箭头

7. 通过按F8逐句运行代码，VB运行当前语句，并且自动向前移动到下一句然后停止执行。当前语句的边框上显示了黄色箭头，并且为黄色底色。不断地按F8逐句执行该过程。当VB遇到With emp语句时，它会切换到过程Class_Initialize。

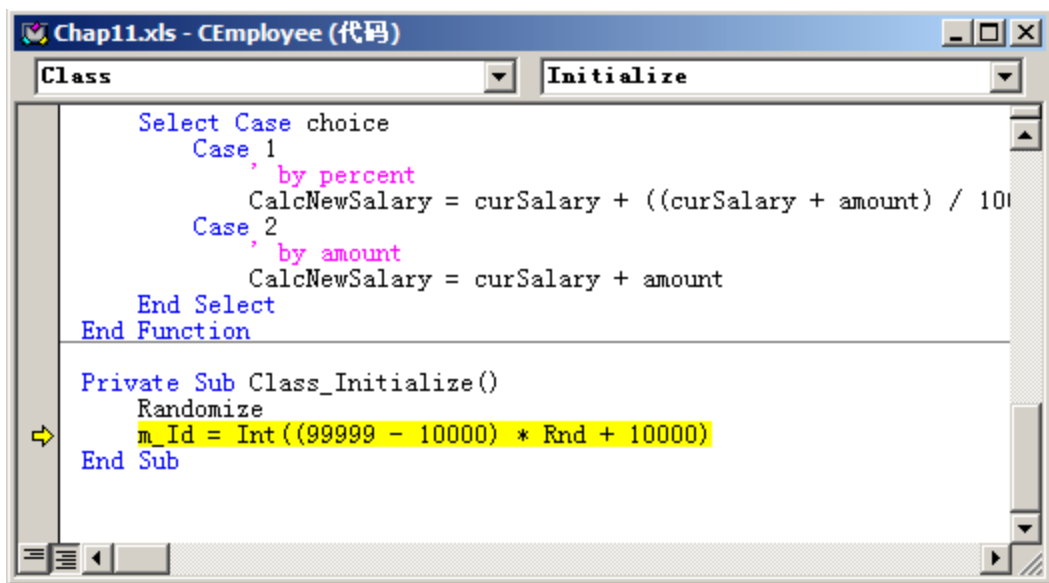


图11-8 当VB遇到对对象emp的引用的时候，它就会出去执行过程Class_Initialize。在它执行完该过程里的语句之后，VBA会返回到过程cmdSave_Click里面。

当VB遇到语句Cells(Index, 1).Formula = emp.ID时，它就会出去执行类模块CEmployee里的过程Property Get Id。

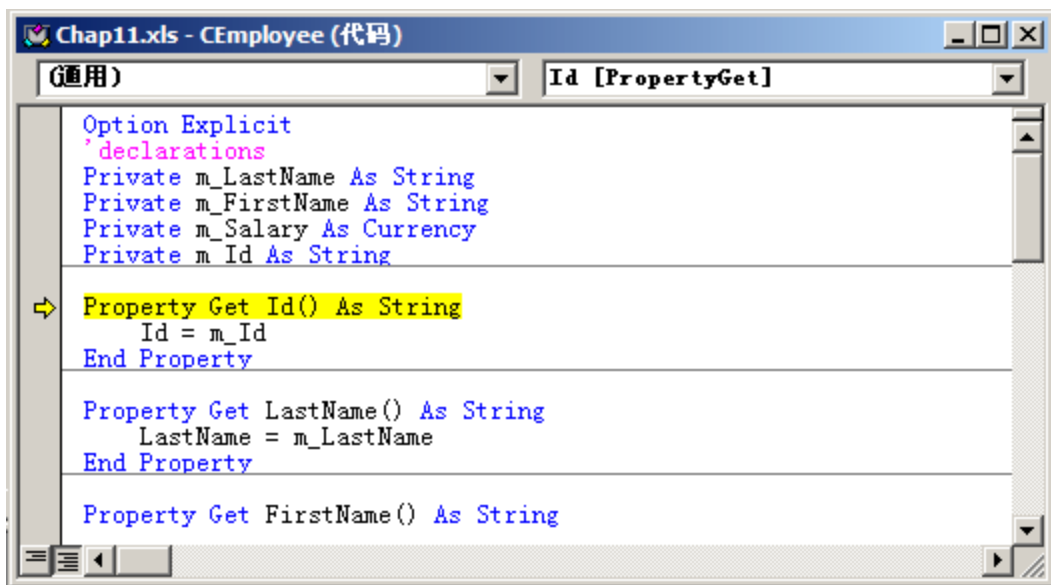


图11-9 对自定义对象属性的读取要通过过程Property Get来实现。

8. 使用F8键，追踪过程cmdSave_Click的执行，直到结束。

当VBA遇到过程的结束（End Sub）时，黄色加亮将会关闭。这时候，点击屏幕下面的视窗任务栏上的Excel按钮返回到当前窗体。输入一个新员工的数据，然后点击按钮Save。当VB显示代码窗口时，选择“调试”|“清除所有断点”。现在按F5运行完剩余的代码。

技巧11-6 VBA调试工具

VB提供了许多调试工具，帮助你分析你的应用程序如何操作，以及找到你过程里的错误源。参见第十三章这些工具的使用。

16.接下来……

在本章里，你学习了在VBA过程里如何创建和使用你自己的对象和集合。你使用了类模块来创建一个用户定义（自定义）对象。你已经看到了如何使用Property Get和Property Let过程定义你的自定义对象的属性。你也学习了如何给你的自定义对象编写方法，再有，你看到了如何通过创建一个自定义窗体使得类模块为用户可用。最后，你学习了如何通过逐句执行代码来分析你的VBA程序。在下一章，你将学习如何通过自定义菜单和工具条让你的VBA程序为终端用户所用。

第十二章 使用 VBA 创建自定义菜单和工具栏

注意，使用中文版的用户需要将相应的代码作一定的修改。下面列出了中文版的工作表菜单栏标题
文件(&F) 编辑(&E) 视图(&V) 插入(&I) 格式(&O) 工具(&T) 数据(&D) 窗口(&W) 帮助(&H)

用户肯定期望在任何Windows应用程序里有一个方便的方法来选择命令和选项，因此，当你某个工作表自动化编写完VBA程序时，你还应该花些时间添加一些功能，让你的应用程序更加使用方便快捷。用户界面最为期望的功能就是自定义菜单和工具栏，当你的VBA程序包含很多个过程时这尤为重要。简单地在内置菜单或者自定义工具栏上创建一个控件，就可以对某个命令提供一个快捷访问。本章将教你如何编程使用菜单和工具栏。

1.工具栏

术语“工具栏”既指工具栏又指菜单栏，工具栏给用户提供了对应用程序命令的快捷方便的访问。可以通过自定义对话框（参见图12-1）轻松才创建和修改工具栏。访问该对话框的方法之一是选择“工具”|“自定义”，你也可以选择“视图”|“工具栏”|“自定义”，或者在工具栏的任意位置单击右键，然后从快捷菜单上选择“自定义”。工具栏可以包括按钮，菜单，或者两者都有。菜单栏位于应用程序窗口的顶部（紧挨着它的标题栏），它是一种特殊的工具栏。除了命令之外，菜单栏也可以包

含图片，允许用户很快将命令和工具栏上相应的按钮联系起来。例如，“文件”菜单里的“新建”和“打开”，在该命令的左边显示了图片，这些相同的图片也可以在Excel“常用”工具栏里面找到。

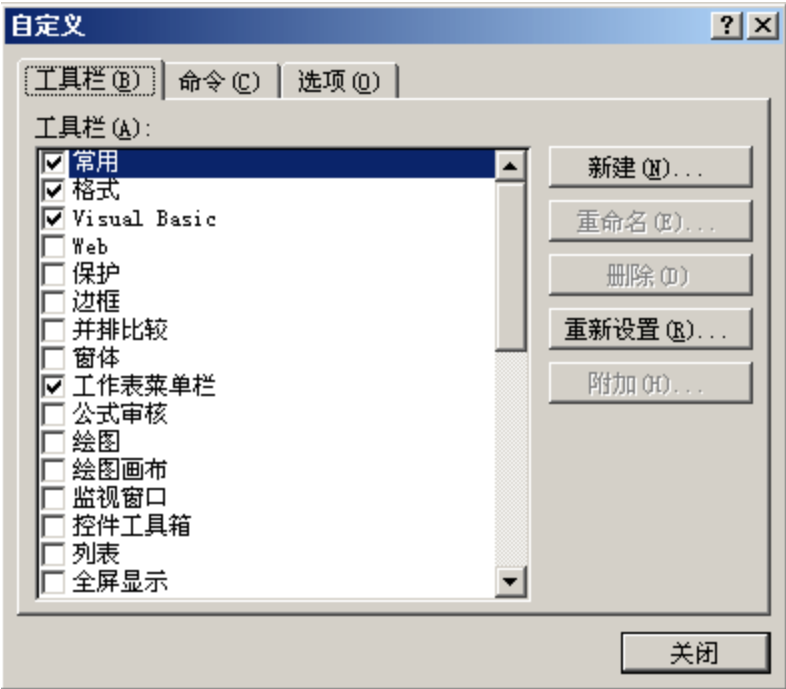


图12-1 使用自定义对话框，可以手动自定义菜单和工具栏

自定义对话框包含三页：工具栏，命令和选项。使用工具栏页，你可以创建一个新工具栏、更改现有工具栏的名称、清除工具栏或者重新设置工具栏。命令页允许你将新的命令拖曳到活动菜单里或者任何可见的工具栏。点击某个类别后，你可以看到它里面可用的命令清单。选项页让你通过设置图标大小，显示关于工具栏的屏幕提示以及选择动画，来设置你个性化的菜单和工具栏。如果你需要重新看一下如何通过对话框操作菜单和工具栏的话，可以看看在线帮助。本章侧重于VBA语句和过程，以获取对应用程序的菜单和工具栏的完全控制。

使用对象CommandBar

CommandBars是对象集合，代表当前应用程序里的所有工具栏。该集合里的每个对象称为CommandBar。术语“CommandBar”用来代表菜单栏、快捷菜单或者工具栏。因为CommandBar对象可以代表各种工具（工具栏，菜单栏，快捷菜单），所以该对象有个专门的属性Type，可以用来返回工具栏的特定类型，如表12-1所示。

表12-1 集合CommandBars里的CommandBar对象类型

对象类型	索引	常量
工具栏	0	msoBarTypeNormal
菜单栏	1	msoBarTypeMenuBar
快捷菜单	2	msoBarTypePopup

- 1. 打开一新工作簿并保存为Chap12.xls
- 2. 切换到VB编辑器屏幕
- 3. 选择当前VBA工程Chap12.xls，并重命名为CustomTools
- 4. 添加一个新模块
- 5. 输入过程MyToolBars，如下所示：

```
Sub MyToolBars()  
    Dim bar As CommandBar  
    Dim r As Integer  
  
    r = 1  
    ActiveSheet.Range("A1").Formula = "List of Toolbars"
```

```

For Each bar In CommandBars
    If bar.Type = msoBarTypeNormal Then
        With Worksheets("Sheet1").Range("A1")
            .Offset(r, 0) = bar.Name
            .Offset(r, 1) = bar.Index
        End With
        r = r + 1
    End If
Next
Set bar = Nothing
End Sub

```

上面的过程在集合CommandBars里面搜索工具，并且只选择Type属性为msoBarTypeNormal的工具。如果集合CommandBars里面的某个成员是工具栏的话，那么VB就会将它的名称输入到活动工作表的第一列，B列将保存该对象的索引号。

修改上面的过程，让它输入集合CommandBars里所有对象（工具栏，菜单栏，快捷菜单）的名称到工作表中去，使用表12-1作为参考。

可以使用工具栏的名称或者索引号来引用CommandBars集合里的某个特定的工具栏。

1. 在立即窗口里输入下述语句：

```
?CommandBars(1).Name
```

当你按下回车键后，VB就会返回CommandBars集合里的第一个成员的名称。

2. 在立即窗口输入下述语句：

```
?CommandBars("Circular Reference").Type
```

VB返回0，这是工具栏的索引号码（参见表12-1）

3. 要计算CommandBars集合里可用工具的总数，可以使用Count属性。在立即窗口里输入下述语句：

```
?CommandBars.Count
```

2.创建自定义工具栏

要创建自定义工具栏、菜单栏或者快捷菜单，可以使用CommandBars对象的Add方法。

假设你想要创建一个叫做“Budget Plans”的新工具栏，你要调用的Add方法如下所示：

```
CommandBars.Add(Name, Position, MenuBar, Temporary)
```

可选参数Name是你想要分配给你的新命令条的名称，如果你不明确该名称，VB会分配一个普通名称，例如“自定义 1”。

Position参数决定新命令条将出现在屏幕的哪里（参见表12-2）。

表12-2 CommandBar对象的位置常量

位置常量	索引	描述
msoBarLeft	0	命令栏位于应用程序窗口的左边
msoBarRight	2	命令栏位于应用程序窗口的右边
msoBarTop	1	命令栏位于应用程序窗口的上面
msoBarBottom	3	命令栏位于应用程序窗口的底部
msoBarFloating	4	命令栏浮在屏幕上
msoBarPopup	5	快捷菜单
msoBarMenuBar	6	命令栏取代系统菜单（仅用于Macintosh）

参数MenuBar是个逻辑值（True或False），它决定新命令条是否取代活动菜单条。如果你想要取代活动菜单条的话，就输入True，否则使用False。参数Temporary是逻辑值（True或False），决定何时删除命令条。使用True，当Excel程序关闭的时候命令条就会自动删除。使用False的话，当你退出该程序的时候，该工具栏不会被删除。

你可以在立即窗口里试验创建工具栏。

1. 在立即窗口里面输入下述语句，注意要将完整的语句书写在一行：

```
set newToolbar = CommandBars.Add("Budget Plans", msoBarRight, False, True)
```

当你按下回车键时，VB就会在集合CommandBars里面添加一个新的工具栏，具体名称为“Budget Plans”。切换到Excel应用软件窗口并且选择“视图”|“工具栏”，你可以看到Excel显示的一列可用工具栏清单，包括你刚才创建的那个（参见图12-2）。

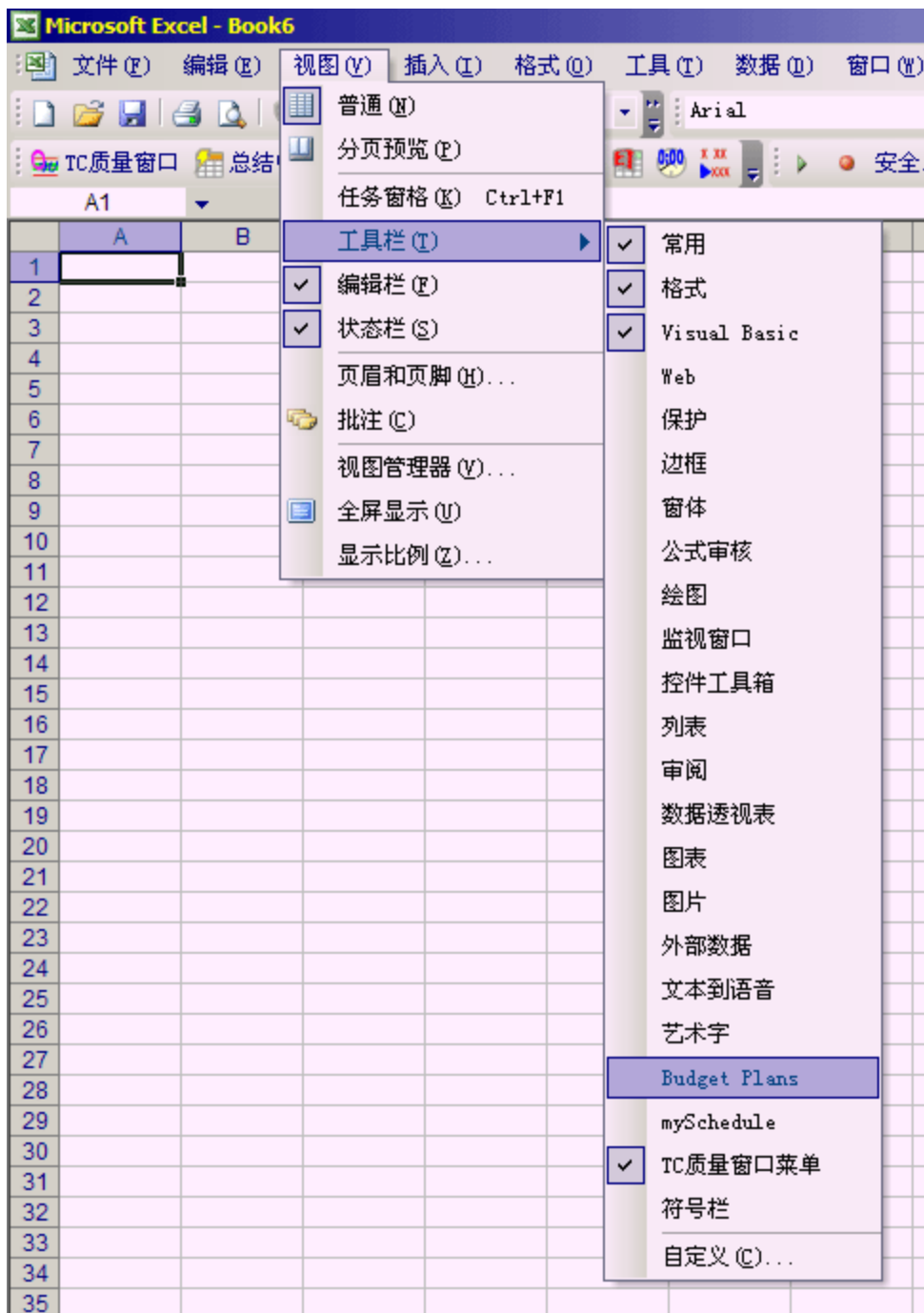


图12-2 你在Excel内置工具栏清单里面添加了一个自定义工具栏

2. 切换回VB编辑器窗口，并在立即窗口里输入下述语句：

```
CommandBars("Budget Plans").Visible = True
```

切换到Excel应用程序窗口查看该工具栏，工具栏“Budget Plans”出现在垂直滚动条的右边。还记得你创建该工具栏的时候吗？你使用的是常量msoBarRight来决定其位置。

3. 现在关闭Excel应用软件，重新打开它并且查看工具栏Budget Plans是否依然出现在应用软件窗口的右边。因为你在Add方法最后一个参数的位置使用了逻辑值True，工具栏Budget Plans应该已经不在。

有个好主意，在你试图创建新工具栏之前，需要检查某个特定名称的工具栏是否已经存在于集合CommandBars里了。下述过程将创建工具栏“Budget Plans”，倘若不存在具有该相同名称的工具栏的话。在工程CustomTools (Chap12.xls)的代码窗口里面输入该过程，并且运行两次。第二次执行该过程的时候，你将看到一信息，提醒你已经有了这样一个工具栏。


```
Sub MakeToolBar()  
    Dim bar As CommandBar  
    Dim flagExists As Boolean  
  
    flagExists = False  
  
    For Each bar In CommandBars  
        If bar.Name = "Budget Plans" Then  
            flagExists = True  
            MsgBox "The toolbar with this name already exists."  
            Exit For  
        End If  
    Next bar  
  
    If Not flagExists Then  
        Set bar = CommandBars.Add("Budget Plans", _  
            msoBarBottom, False, True)  
        CommandBars("Budget Plans").Visible = True  
    End If  
    Set bar = Nothing  
End Sub
```

3.删除自定义工具栏

如果你创建了工具栏但不想保留它，那么你可以去掉它而不用关闭Excel应用软件，只要使用Delete方法就可以了。例如要删除工具栏“Budget Plans”，你可以在立即窗口里输入下述语句：

```
CommandBars("Budget Plans").Delete
```

注意，你不能删除内置工具栏。

4.使用 CommandBar 的属性

对象CommandBar有许多属性，你在立即窗口里面使用它们中的一些。

1. 使用立即窗口来创建一个叫“My Reports”的工具栏：

```
set myBar= CommandBars.Add("My Reports", msoBarBottom, False)
```
2. 使用下述语句来确定某个工具栏是否是内置的：

```
?CommandBars("My Reports").BuiltIn
```
3. 输入下述语句可以确定新工具栏在CommandBars集合里的索引号：

```
?CommandBars("My Reports").Index
```

当设置属性Visible为True时，该工具栏将显示在屏幕上；而设置属性Visible为False时则可以隐藏该工具栏。

5.使用 CommandBar 控件

一个空的工具栏并不能做什么，要让你的工具栏有用的话，你就需要将想要地控件放置在上面并且给它们指定适当的VBA过程。有三种类型的命令条控件，如下表所示：

表12-3：可以放置在工具栏上的控件类型

对象名称	描述
CommandBarButton	该对象代表工具栏按钮和菜单选项。当你点击按钮或者选择菜单选项时，就会执行相应的VBA过程
CommandBarPopup	该对象代表弹出控件，点击时显示一菜单或者子菜单

CommandBarComboBox 该对象代表文本框、列表框或者下拉列表框（例如，格式工具栏上的字体和字号控件，或者常用工具栏上的缩放控件）

CommandBar对象的一个重要属性是Controls属性，该属性返回某特定工具栏上所有控件的集合。

1. 在立即窗口里敲入下述语句：
`?CommandBars(1).Controls.Count`
当你按下回车键时，VB就会返回工作表系统菜单条上所有可用控件的总数。
2. 输入下述语句来返回工作表系统菜单条上第一个控件的名称：
`?CommandBars(1).Controls(1).Caption`
VB返回第一个控件的名称：**&File**。字母F前面的字符&表明该菜单选项可以通过键盘按下Alt+F来执行。
3. 输入下面的语句来执行一特定选项：
`CommandBars(1).Controls(1).Execute`
方法Execute激活该特定的控件，文件菜单应该被打开了。
4. 在当前工程代码窗口里输入过程ControlList，来将活动菜单条上所有控件的名称写入立即窗口：

```
Sub ControlList()  
    Dim bar As CommandBar  
    Dim ctrl As CommandBarControl  
  
    Set bar = CommandBars(1)  
    Debug.Print bar.Name & ": " & bar.Controls.Count  
  
    For Each ctrl In bar.Controls  
        Debug.Print ctrl.Caption  
    Next  
End Sub
```
5. 运行上述过程后，查看立即窗口，你将看到下述清单：

```
Worksheet Menu Bar: 10  
&File  
&Edit  
&View  
&Insert  
F&ormat  
&Tools  
&Data  
A&ction  
&Window  
&Help
```

为CommandBar添加控件

要运行期望的VBA过程的话，那么你可以添加一个内置的或者自定义控件到内置工具栏。如果你更愿意，你也可以添加控件到自定义工具栏。无论你是添加内置控件还是自定义控件，到内置工具栏或者到自定义工具栏，总是要使用Add方法，语法如下：

`CommandBar.Controls.Add(Type, Id, Parameter, Before, Temporary)`

CommandBar是你要添加控件的那个对象。

Type是一常量，决定你添加的自定义控件的类型，你可以从下述类型中选择一个：

msoControlButton	1
msoControlPopup	10
msoControlEdit	2
msoControlDropDown	3
msoControlComboBox	4

Id是个整数，指定你想要添加的内置控件编号。

Parameter用来给VB过程发送信息，或者储存关于该控件的信息。

Before参数是新控件添加在之前的那个控件的索引号，如果忽略，那么VB将在该命令条的结尾处添加控件。

Temporart参数是个逻辑值(True或False)，决定控件什么时候被删除。设置该参数为True的话，将导致应用软件关闭时，该控件将自动被删除。

1. 在代码窗口里输入过程AddBarAndControls，如下所示：

```
Sub AddBarAndControls( )
    With Application.CommandBars.Add("Test", , False, True)
        .Visible = True
        .Position = msoBarBottom
        With .Controls.Add(msoControlButton)
            .Caption = "List of Controls"
            .FaceId = 4
            .OnAction = "ControlList"
        End With
    End With
End Sub
```

该过程创建了一个名为**Test**的新工具栏，并将它放在应用软件窗口的底部。接下来，**Add**方法在其上放置一个名为**List of Controls**的按钮，并用打印机图标以识别。当用户点击该按钮时，之前已准备好的过程**ControlList**就会被执行。

6.理解和使用控件属性

放置在工具栏上的控件有许多属性可供你读取或者设置。要知道某个控件是内置的或者自定义的话，那么你可以使用属性**BuiltIn**，如果返回值为True，那么该控件就是内置控件；所有用户定义的控件将返回值False。如果**Enabled**属性的值为True，那么该控件就是活动的，并且可以对鼠标点击作出反应。非活动控件的**Enabled**属性被设置为False了。不用说，所有控件拥有属性**Caption**，可以用来确定或者设置控件标题。

以**CommandBarComboBox**对象为代表的组合类型的控件具有一些特殊的属性，例如**DropDownLines**，**DropDownWidth**，**List**，**ListCount**，**ListIndex**以及**Text**。参见表12-4对这些属性的解释。

表12-4 对象**CommandBarComboBox**选取的属性

属性	描述
DropDownLines	返回或者设置当用户点击组合框下拉箭头时，显示的项目数量
DropDownWidth	返回或者设置组合框控件的宽度，以像素为单位
List(Index)	返回或者设置由 Index 指定的列表项目值（列表里的第一个项目索引号为0）
ListCount	返回列表清单里总项目数
ListIndex	返回或者设置清单里选定项
Text	返回或者设置出现在组合框控件部分——文字框显示的文本

1. 在代码窗口输入过程MyCombo，如下所示：

```
Sub MyCombo()
    Dim cbo As CommandBarControl
    Set cbo = CommandBars(4).Controls.Add(Type:=4, Before:=1)

    With cbo
        .AddItem Text:="Row", Index:=1
        .AddItem Text:="Column", Index:=2
        .Caption = "Insert Row/Column"
        .DropDownLines = 2
        .DropDownWidth = 80
    End With
End Sub
```

过程**MyCombo**创建了一个组合框（**Type:=4**表明**msoControlComboBox**）并将其放置在内置“格式”工具栏（该工具栏是**CommandBars**集合里的第四个**CommandBar**对象）的最前面。接下来，有两个项目被添加到组合框控件。该过程也设置了组合框标题以及组合框控件宽度。

2. 切换到Excel窗口检查格式工具栏里的第一个控件
3. 返回到VB编辑器窗口
4. 在立即窗口里输入下述语句，从格式工具栏里删除由MyCombo过程创建的组合框控件：

```
CommandBars(4).Controls(1).Delete
```

当你按下回车键后，VB就将格式工具栏里的第一个控件删除了。

由于有放置在其上的图像，这些出现在工具栏上的按钮都很好辨认。如果工具栏上的控件是个CommandBarButton对象，属性FacelId将返回或者设置按钮上图标ID编号。大多数情况下，图标的ID编号和控件的ID属性是相同的。使用CopyFace方法可以将图标图片复制到Windows的剪切板上。

接下来的过程Images将出现在标准工具栏上的按钮列出到电子表格上。除了按钮名称，该清单同时也显示它的图标。因为不能复制当前禁用的图标图像（参见标准工具栏上的“撤销”和“恢复”按钮），所以当VB试图复制按钮的图标至剪切版时，就会遇到错误。过程Images利用On Error GoTo ErrorHandler语句捕获该错误。这样一来，当VB遇到错误时，就会跳到ErrorHandler:标志并且执行该标志下面的指令。最后一条语句Resume Next会让VB回到刚才导致该错误的下面一条语句，并且该过程会继续直到标准工具栏上所有的按钮都被检查一遍了。你将在下一章学习更多有关错误捕捉的知识。

	A	B	C	D	E	F
1	Image	Index	Name	FacelId		
2		2520	新建(&N)	2520		
3		23	打开	23		
4		3	保存(&S)	3		
5		9004	权限(无限制访问)	9004		
6		3738	邮件收件人(&M)	3738		
7		2521	打印 (Acrobat Distiller)	2521		
8		109	打印预览(&V)	109		
9		2	拼写检查(&S)...	2		
10		7343	信息检索(&R)...	7343		
11		21	剪切(&T)	21		
12		19	复制(&C)	19		
13		6002	粘贴(&P)			
14		108	格式刷(&F)	108		
15		128	撤销 (&U)			
16		129	恢复(&R)			
17		9071	墨迹注释(&A)	9071		
18		1576	超链接(&I)...	1576		
19		226	自动求和(&A)			
20		210	升序排序(&A)	210		
21		211	降序排序(&C)	211		
22		436	图表向导(&C)	436		
23		204	绘图(&D)	204		
24		1733	显示比例(&Z):			
25		984	Microsoft Excel 帮助(&H)	984		
26						
27						

图12-3 标准工具栏上图标列表。你可以修改过程Images来列出任何工具栏上的完整按钮和它们的图标。

```
Sub Images()
    Dim i As Integer
    Dim total As Integer
    Dim buttonId As Integer
    Dim buttonName As String
    Dim myControl As CommandBarControl
    Dim bar As CommandBar
```

```
On Error GoTo ErrorHandler
```

```
Workbooks.Add
```

```

Range("A1").Select

With ActiveCell
    .Value = "Image"
    .Offset(0, 1) = "Index"
    .Offset(0, 2) = "Name"
    .Offset(0, 3) = "FaceId"
End With

Set bar = CommandBars(3)
total = bar.Controls.Count

With bar
    For i = 1 To total
        buttonName = .Controls(i).Caption
        buttonId = .Controls(i).ID
        Set myControl = CommandBars.FindControl(ID:=buttonId)
        myControl.CopyFace

        ' error could occur here
        ActiveCell.Offset(1, 0).Select
        ActiveSheet.Paste

        With ActiveCell
            .Offset(0, 1).Value = buttonId
            .Offset(0, 2).Value = buttonName
            .Offset(0, 3).Value = myControl.FaceId
        End With
    Next i

    Columns("C:C").EntireColumn.AutoFit
Exit Sub

ErrorHandler:
Set myControl = CommandBars(3).Controls.Add
With myControl
    .FaceId = buttonId
    .CopyFace
    .Delete (False)
End With
Resume Next
End With
End Sub

```

7.控件方法

控件拥有很多相关的方法，这些方法允许你进行一些操作，例如移动、复制和删除控件。假设你想复制格式工具栏上的“粗体”按钮到标准工具栏上：

1. 在立即窗口里输入下述三条语句：

```

set myBar = CommandBars(3)
set myControl = CommandBars(4).Controls(3)
myControl.Copy Bar:=myBar, Before:=1

```
2. 切换到Excel应用程序窗口，你应该能看到标准工具栏“新建”按钮的左边有了“粗体”按钮。
3. 切换到VB编辑器屏幕，并在立即窗口里输入下述语句从标准工具栏上删除该粗体按钮：

```

CommandBars(3).Controls(1).Delete

```

将Copy方法改成Move方法的话，你就可以将粗体按钮从格式工具栏移动到标准工具栏，你可以自己试验一下。使用方法Reset，你可以将工具栏恢复为缺省设置。当你练习完移动和复制按钮后，请在立即窗口里输入下述语句：

```
CommandBars(3).Reset
```

```
CommandBars(4).Reset
```

如果某个控件是个组合框（CommandBarComboBox）的话，那么你就可以使用AddItem方法给它的下拉清单添加新项目。如果要从该清单删除项目的话，就可以使用RemoveItem方法。我们花上几分钟来在立即窗口里练习这些方法吧。

1. 激活你之前准备的过程MyCombo，运行该过程在格式工具栏上放置一个自定义组合框控件
2. 在立即窗口里输入下述语句：

```
set myBar = CommandBars(4)
```

```
set myControl = CommandBars(4).Controls(1)
```

```
myControl.RemoveItem(1)
```

```
myControl.AddItem "Cells", 1
```

3. 切换到Excel窗口，并且查看格式工具栏上该自定义组合框控件里可用的项目
4. 返回到VB编辑器窗口，在立即窗口里面输入下述语句并且回车，以重新设定格式工具栏：

```
CommandBars(4).Reset
```

8.使用菜单

就像工具栏一样，菜单也是CommandBar对象。有两类菜单：内置菜单和快捷菜单。内置菜单出现在应用程序窗口的顶端，紧接着在标题栏之下。在Excel 2002里，有两种内置菜单：工作表菜单和图表菜单。

如果当前活动的是工作表时，那么出现的就是工作表菜单（见图12-4），并列出几个主要的菜单。每个主菜单组都和某些特定的任务相关联，可以在工作表或者工作簿上执行。例如，格式菜单包含各种选项允许将各种各样的格式应用到工作表。有些菜单选项组合了一些更详细的选项在子菜单里（见图12-5）

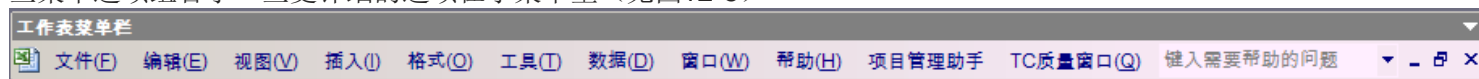


图12-4 Excel应用软件的工作表菜单栏

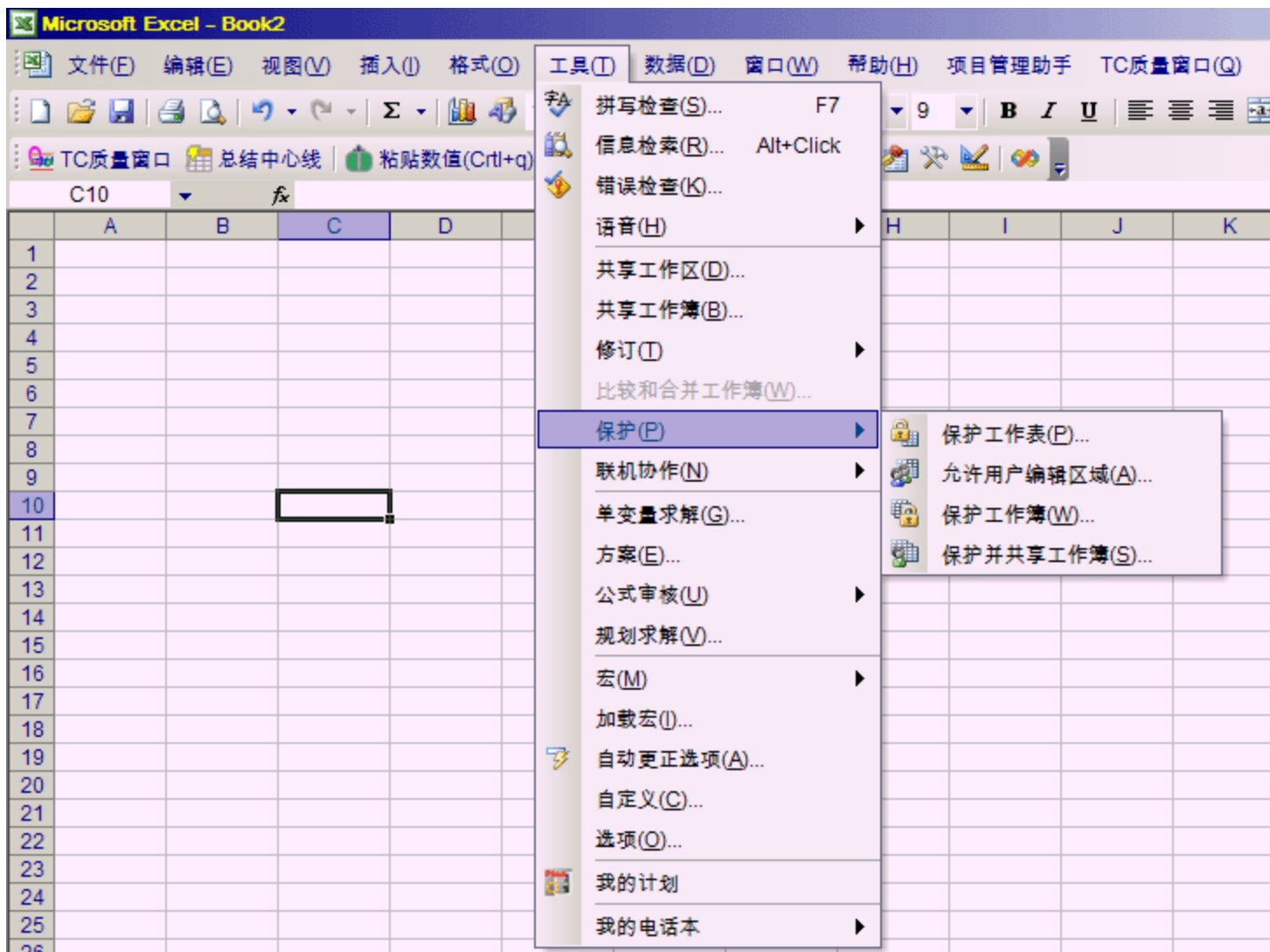


图12-5 选择右边带三角的菜单选项将展开一个带有更多选项的子菜单

当用户正在使用图表页或者选择了内嵌于工作表里的图表时，工作表菜单栏就会被图表菜单栏取代（见图12-6）。同一时间应用程序窗口上只能显示一个菜单栏。



图12-6 Excel内置的图表菜单栏

当你在某个对象上单击右键或者按下Shift+F10时，就会出现快捷菜单。Excel 2002 有50个以上的快捷菜单。快捷菜单包含一些经常使用的命令，例如，当你在工作表的任何单元格单击右键时，单元格快捷菜单就会出现（见图12-7）。当你在工作表标签上单击右键时，就会出现工作表标签的标准菜单（见12-8）。

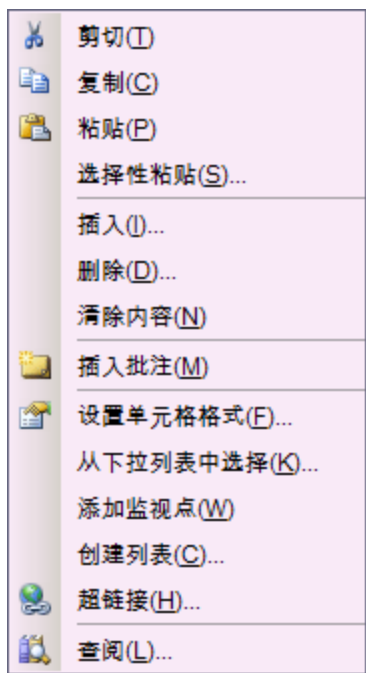


图12-7 当右键单击任何单元格时出现该快捷菜单

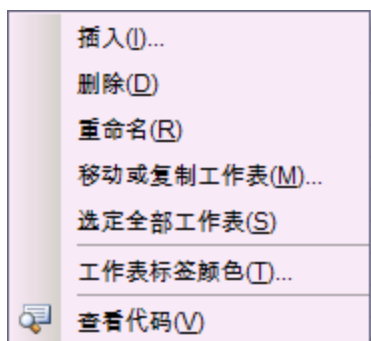


图12-8 当右键单击工作表标签时出现该快捷菜单

菜单栏和工具栏一样，用同样的对象`CommandBar`代表。使用对象`Control`指向菜单，菜单选项，子菜单或快捷菜单。`Control`对象的类型由适当的常量决定。使用`msoControlPopup`指向菜单，`msoControlButton`常量指向菜单选项，而`msoBarPopup`常量则指向快捷菜单。你将在下一章节里学习如何使用这些常量。

9. 菜单编程

使用VBA，你可以进行一些操作，例如创建新菜单栏，添加新菜单到内置菜单栏，激活内置或者自定义菜单栏，删除用户定义菜单栏，重设内置菜单，判断某个菜单栏时内置的或者自定义的，等等。

1. 在立即窗口里输入下述语句，返回当前活动菜单栏地名称：

```
?CommandBars.ActiveMenuBar.Name
```

当你按下回车，VB就会返回活动菜单栏的名称：工作表菜单栏。

菜单栏上的每个菜单都有一个标题，可以通过属性`Caption`和`Id`返回或者设置。

2. 在本章的模块里面输入下述过程，来返回内置菜单栏上格式菜单的ID：

```
Sub Return_ID()  
    Dim myControl As Object  
  
    Set myControl = CommandBars("Worksheet menu bar").Controls("Format")  
    Debug.Print myControl.Caption & " Id is " & myControl.Id  
End Sub
```

如果你想让上面的过程更灵活一些的话，可以按照下述Set语句，让用户可以返回工作表菜单栏上其它菜单的ID：

```
Set myControl = CommandBars("Worksheet menu bar").Controls (InputBox("Enter the menu name (Example: Format:")))
```

3. 运行过程Return_Id，然后切换到立即窗口查看其结果

4. 在立即窗口里输入下述语句，可以创建一个名为Other的自定义菜单，并将它放置在内置工作表菜单栏上：

```
CommandBars("Worksheet menu bar").Controls.Add(Type:=msoControlPopup, before:=10).Caption = "&Other"
```

当你按下回车键并切换回Excel应用程序窗口时，工作表菜单栏将会在“帮助”菜单前显示你的自定义菜单。如果你没有将上面的语句输入在一行的话，它就不会起作用。

现在“Other”菜单是空的，下面一步时示范如何添加菜单命令。

5. 在立即窗口里输入下述语句，给自定义菜单里添加自定义命令（选项）：

```
CommandBars("Worksheet menu bar").Controls("Other").Controls.Add(Type:=msoControlButton, before:=1) _  
.Caption = "Gridlines"
```

当你按下回车键并切换到Excel应用程序菜单，然后选择Other，你将看到命令Gridlines。如果你没有将上面的语句输入在一行的话，它就不会起作用。

下一步将要求你给你的自定义菜单选项指定一个适当的VBA过程，当用户选择该菜单时就会执行。

6. 在当前工程代码窗口里输入下述过程打开或者关闭网格线的显示：

```
Sub GridOnOff()  
ActiveWindow.DisplayGridlines = Not ActiveWindow.DisplayGridlines  
End Sub
```

7. 在立即窗口里输入下面的代码，将过程GridOnOff指定给你的自定义菜单选项：

```
CommandBars("Worksheet menu bar").Controls("Other").Controls("Gridlines").OnAction = "GridOnOff"
```

当你按下回车，VB就会将过程GridOnOff指定给Gridlines菜单项。如果你没有将上面的语句输入在一行的话，它就不会起作用。

当你切换到Excel应用程序窗口并且选择“Other”|“Gridlines”的时候，如果网格线显示被关闭时，VB就会打开网格线显示，反之亦然。

将菜单项的属性Enabled设置为False可以临时禁用该菜单项。一个被禁用的菜单项其名称将变为灰色，并且点击它时不会有任何反应。

8. 在立即窗口里输入下述语句在同一行，可以禁用Other菜单里的Gridlines命令：

```
CommandBars("Worksheet menu bar").Controls("Other").Controls("Gridlines").Enabled = False
```

当你按下回车，VB就会将Gridlines菜单项禁用。如果你没有将上面的语句输入在一行的话，它就不会起作用。当你切换到Excel应用程序窗口并且选择Other时，Gridlines选项不再可用。

9. 要激活Other菜单里的Gridlines命令的话，可以在立即窗口里将逻辑值False取代为True：

```
CommandBars("Worksheet menu bar").Controls("Other").Controls("Gridlines").Enabled = True
```

注意，内置菜单上的每个具体选项都是和它们相似的命令组织在一起的，组和组之间用一条横线分割开（见图12-9）。使用方法BeginGroup可以在菜单项之间添加这样一条线。

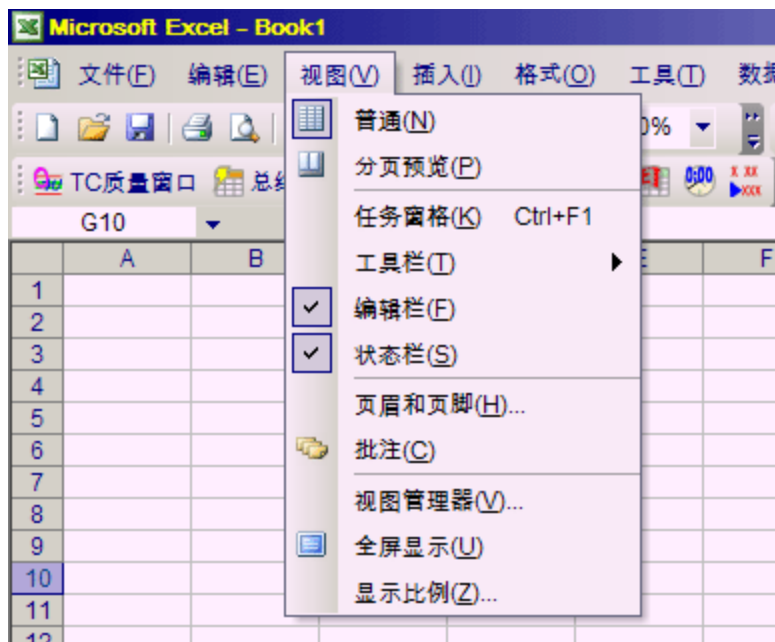


图12-9 每个菜单用横线分成好几个部分

10. 在立即窗口里输入下述语句，可以在“窗口”菜单的“隐藏”命令上面添加一条横线：

```
CommandBars("Worksheet menu bar").Controls("Window").Controls("Hide").BeginGroup = True
```

当你按下回车，VB就会在窗口菜单的隐藏命令上面添加一条横线。如果你没有将上面的语句输入在一行的话，它就不会起作用。

当你切换到Excel应用程序窗口并且选择窗口，你就会看到隐藏和取消隐藏命令包含在两条横线里了。上面的那条就是你自己添加的。

当某个菜单项被选中时，该选项左边就会出现一个复选记号，例如在图12-9里的视图菜单里，编辑栏和状态栏左边的复选记号表明这些选项当前是有效的。

11. 要显示你的自定义Other菜单里的Gridlines选项被选中的话，可以修改过程GridOnOff，如下所示：

```
Sub GridOnOff()  
    Dim Other As Object  
  
    Set Other = CommandBars("Worksheet menu bar").Controls("Other")  
  
    ActiveWindow.DisplayGridlines = Not ActiveWindow.DisplayGridlines  
  
    If ActiveWindow.DisplayGridlines = True Then  
        Other.Controls("Gridlines").State = msoButtonDown  
    Else  
        Other.Controls("Gridlines").State = msoButtonUp  
    End If  
End Sub
```

运行该过程，然后切换到Excel窗口，并且选择Other | Gridlines。如果活动工作表的网格线时显示的话，那么它们现在就会被关闭。再次选择Other | Gridlines。

12. 在立即窗口里输入下述语句，以删除内置工作表菜单栏里的自定义菜单：

```
CommandBars("Worksheet menu bar").Controls("Other").Delete
```

当你删除某个自定义菜单的时候，里面所有的菜单项都会自动被删除。自定义菜单以及其选项一旦被删除，你就不能恢复它们。

10.创建子菜单

菜单项在其名称右边包含一个黑色三角的，都会显示一个子菜单，包括一些额外的命令。假设你想在工具菜单里添加一个子

菜单。

1. 在立即窗口里输入下述语句，给工具菜单添加一个子菜单：

```
CommandBars("Worksheet menu bar").Controls("Tools").Controls.Add(Type:=msoControlPopup, Before:=1) _  
.Caption = "My Submenu"
```

当你按下回车键，上面的指令会在工具菜单（工作表菜单栏）的上面放置一个自定义子菜单，叫做My Submenu。如果你没有将上面的语句输入在一行的话，它就不会起作用。

2. 在立即窗口里输入下述一行指令，可以在子菜单里添加自定义命令：

```
CommandBars("Worksheet menu bar").Controls("Tools").Controls("My Submenu").Controls _  
.Add(Type:=msoControlButton, Before:=1).Caption = "Option 1"
```

当你按下回车键，上面的指令会在工具菜单里的My Submenu里添加命令Option 1。如果你没有将上面的语句输入在一行的话，它就不会起作用。你可以使用相同的技术给你的子菜单里添加更多的菜单项。

下面的过程将在内置菜单格式里添加自定义子菜单Colors，并且在里面放置四个选项：Red, Green, Blue和Black。使用这些选项，你可以更改所选工作表单元格或者单元格区域里的文本颜色。接下来的过程将应用适当的颜色格式。

```
Sub Colors()  
    Dim myMenu As Object  
    Dim mySubMenu As Object  
  
    Set myMenu = CommandBars("Worksheet menu bar").Controls("Format")  
  
    With myMenu  
        .Controls.Add(Type:=msoControlPopup, Before:=2).Caption = "Colors"  
    End With  
  
    Set mySubMenu = myMenu.Controls("Colors")  
  
    With mySubMenu  
        .Controls.Add(Type:=msoControlButton).Caption = "Red"  
        .Controls.Add(Type:=msoControlButton).Caption = "Green"  
        .Controls.Add(Type:=msoControlButton).Caption = "Blue"  
        .Controls.Add(Type:=msoControlButton).Caption = "Black"  
        .Controls("Red").OnAction = "ColorRed"  
        .Controls("Green").OnAction = "ColorGreen"  
        .Controls("Blue").OnAction = "ColorBlue"  
        .Controls("Black").OnAction = "ColorBlack"  
    End With  
End Sub  
  
Sub ColorRed()  
    ActiveCell.Font.Color = RGB(255, 0, 0)  
End Sub  
  
Sub ColorGreen()  
    ActiveCell.Font.Color = RGB(0, 255, 0)  
End Sub  
  
Sub ColorBlue()  
    ActiveCell.Font.Color = RGB(0, 0, 255)  
End Sub  
  
Sub ColorBlack()  
    ActiveCell.Font.Color = RGB(0, 0, 0)  
End Sub
```

11.修改内置快捷菜单

Excel提供了60来个快捷菜单，带有不同的经常用到的菜单项。当你在Excel应用程序窗口的某个对象上单击右键时，快捷菜单就会出现。通过使用VBA，你可以返回快捷菜单的准确编号，还有它们的名称。

1. 在当前工程的模块里输入过程ShortcutMenus，如下所示：

```
Sub ShortcutMenus()  
    Dim myBar As CommandBar  
    Dim counter As Integer  
  
    For Each myBar In CommandBars  
        If myBar.Type = msoBarTypePopup Then  
            counter = counter + 1  
            Debug.Print counter & ": " & myBar.Name  
        End If  
    Next  
End Sub
```

注意，要使用常量msoBarTypePopup来确定CommandBars集合里的快捷菜单类型。使用常量msoBarTypeMenuBar，可以返回内置菜单的名称。

当你运行过程ShortcutMenus后，快捷菜单的名称就会打印在立即窗口里，这里也列出来了。

Excel 2002 的内置快捷菜单：	Excel 2003 的内置快捷菜单：
1: Query and Pivot	1: Query and Pivot
2: PivotChart Menu	2: PivotChart Menu
3: Workbook tabs	3: Workbook tabs
4: Cell	4: Cell
5: Column	5: Column
6: Row	6: Row
7: Cell	7: Cell
8: Column	8: Column
9: Row	9: Row
10: Ply	10: Ply
11: XLM Cell	11: XLM Cell
12: Document	12: Document
13: Desktop	13: Desktop
14: Nondefault Drag and Drop	14: Nondefault Drag and Drop
15: AutoFill	15: AutoFill
16: Button	16: Button
17: Dialog	17: Dialog
18: Series	18: Series
19: Plot Area	19: Plot Area
20: Floor and Walls	20: Floor and Walls
21: Trendline	21: Trendline
22: Chart	22: Chart
23: Format Data Series	23: Format Data Series
24: Format Axis	24: Format Axis
25: Format Legend Entry	25: Format Legend Entry
26: Formula Bar	26: Formula Bar
27: PivotTable Context Menu	27: PivotTable Context Menu
28: Query	28: Query
29: Query Layout	29: Query Layout
30: AutoCalculate	30: AutoCalculate
31: Object/Plot	31: Object/Plot

32: Title Bar (Charting)	32: Title Bar (Charting)
33: Layout	33: Layout
34: Pivot Chart Popup	34: Pivot Chart Popup
35: Phonetic Information	35: Phonetic Information
36: Auto Sum	36: Auto Sum
37: Paste Special Dropdown	37: Paste Special Dropdown
38: Find Format	38: Find Format
39: Replace Format	39: Replace Format
40: Shapes	40: Shapes
41: Inactive Chart	41: Inactive Chart
42: Excel Control	42: Excel Control
43: Curve	43: Curve
44: Curve Node	44: Curve Node
45: Curve Segment	45: Curve Segment
46: Pictures Context Menu	46: Pictures Context Menu
47: OLE Object	47: OLE Object
48: ActiveX Control	48: ActiveX Control
49: WordArt Context Menu	49: WordArt Context Menu
50: Rotate Mode	50: Rotate Mode
51: Connector	51: Connector
52: Script Anchor Popup	52: Script Anchor Popup
53: Canvas Popup	53: Canvas Popup
54: Organization Chart Popup	54: Organization Chart Popup
55: Diagram	55: Diagram
56: Add Command	56: Layout
57: Built-in Menus	57: Select
58: System	58: List Range Popup
59: Layout	59: List Range Layout Popup
60: Select	60: XML Range Popup
	61: List Range Layout Popup
	62: Built-in Menus

现在，你已经知道里Excel快捷菜单的准确名称了，你可以轻易地添加其它经常用到的命令到这些菜单中的任意菜单中去。尽管从工具栏上点击打印图标或者选择文件|打印来打印工作表都是很容易的事情，但是你可能还是想将该打印命令添加到快捷菜单中去，当你在工作表标签上单击右键时就会出现在该快捷菜单上。我们来看看如何添加该选项到Ply菜单上去。

2. 输入如下所示地过程AddToPlyMenu:

```
Sub AddToPlyMenu()
    With Application.CommandBars("Ply")
        .Reset
        .Controls.Add(Type:=msoControlButton, Before:=2).Caption = _
            "Print..."
        .Controls("Print...").OnAction = "PrintSheet"
    End With
End Sub
```

上面所用地Reset方法避免当你多次运行该过程时，将同样的选项放置到该快捷菜单上。

3. 运行过程AddToPlyMenu，然后返回到代码窗口，并且输入下述过程，当你从该快捷菜单上选择Print选项时，就会执行该过程：

```
Sub PrintSheet()
    Application.Dialogs(xlDialogPrint).Show
End Sub
```

4. 切换到Excel应用程序窗口，并且在任何工作表标签上单击右键，选择Print选项，你应该可以看到当你使用其它内置工具打印时看到的相同的对话框。



图12-10 自定义选项可以添加到内置快捷菜单上（参见Print选项在过程AddToPlyMenu里被添加上了）

12.创建快捷菜单

1. 在当前VBA工程的代码窗口里输入过程Create_ShortMenu，如下所示：

```
Sub Create_ShortMenu()
    Dim sm As Object
    Set sm = Application.CommandBars.Add("Information", msoBarPopup)

    With sm
        .Controls.Add(Type:=msoControlButton).Caption = "Operating System"

        With .Controls("Operating System")
            .FaceId = 1954
            .OnAction = "OpSystem"
        End With

        .Controls.Add(Type:=msoControlButton).Caption = "Total Memory"

        With .Controls("Total Memory")
            .FaceId = 1977
            .OnAction = "TotalMemory"
        End With

        .Controls.Add(Type:=msoControlButton).Caption = "Used Memory"

        With .Controls("Used Memory")
            .FaceId = 2081
            .OnAction = "UsedMemory"
        End With

        .Controls.Add(Type:=msoControlButton).Caption = "Free Memory"

        With .Controls("Free Memory")
            .FaceId = 2153
            .OnAction = "FreeMemory"
        End With
    End With
End Sub
```

上面的过程创建了一个名为**Information**的自定义快捷菜单，并给它添加了四个命令。注意，每个命令都指定了一个图标。当你从该快捷菜单选择一命令，步骤2里面的相应过程就会被执行。

2. 输入下面为**Create_ShortMenu**过程调用的过程：

```
Sub FreeMemory( )
    MsgBox Application.MemoryFree & " bytes", , "Free Memory"
End Sub
```

```
Sub OpSystem( )
    MsgBox Application.OperatingSystem, , "Operating System"
End Sub
```

```
Sub TotalMemory( )
    MsgBox Application.MemoryTotal, , "Total Memory"
End Sub
```

```
Sub UsedMemory( )
    MsgBox Application.MemoryUsed, , "Used Memory"
End Sub
```

要将名为**Information**的自定义快捷键显示在屏幕上的话，你可以使用方法**ShowPopup**，如步骤3所示。

3. 在立即窗口里输入下述语句：

```
CommandBars("Information").ShowPopup 0, 0
```

对象**CommandBar**的方法**ShowPopup**接受两个可选参数（**x**, **y**），决定快捷菜单在屏幕上的位置。在上面的例子里，快捷菜单**Information**将出现在屏幕的左上角。

假设你正在设计一个自定义窗体，并想要当用户右键单击一个命令按钮时显示一个快捷菜单：

1. 从VB编辑器菜单上，选择插入-用户窗体
2. 使用工具箱上的命令控件，在空白用户窗体的任意位置放置一个按钮
3. 通过点击工程浏览器窗口的查看代码按钮，切换到该窗体的代码窗口
4. 在**UserForm1**代码窗口里输入下述过程：

```
Private Sub CommandButton1_MouseDown(ByVal Button _
    As Integer, _
    ByVal Shift As Integer, _
    ByVal X As Single, _
    ByVal Y As Single)
    If Button = 2 Then
        Call Show_ShortMenu
    Else
        MsgBox "You must right-click this button."
    End If
End Sub
```

当用户右键点击窗体上按钮时，该过程就会调用过程**Show_ShortMenu**。点击鼠标按钮时VB会有两个事件过程响应。当你点击鼠标按钮时，VB会执行**MouseDown**事件过程，当你释放鼠标按键，**MouseUp**事件则会发生。

MouseDown和**MouseUp**事件过程要求下述参数：

- 参数**object**确定对象，在该例中，是窗体上面的命令按钮名称
- 参数**Button**是整型数据，确定按下的是哪个鼠标按键

Button参数值	意义
1	鼠标左键
2	鼠标右键
3	鼠标中键

- 参数**Shift**确定当事件发生时，用户是否按住了**Shift**, **Crel**或者**Alt**键。

Shift参数值	意义
1	Shift键
2	Ctrl键

3	Shift和Ctrl键
4	Alt键
5	Alt和Shift键
6	Alt和Ctrl键
7	Alt, Shift和Ctrl键

5. 在当前工程模块里输入过程Show_ShortMenu代码:

```
Sub Show_ShortMenu()
    Dim shortMenu As Object
    Set shortMenu = Application.CommandBars("Information")
    With shortMenu
        .ShowPopup
    End With
End Sub
```

注意, 本过程使用的方法ShowPopup没有使用决定快捷菜单屏幕显示位置的可选参数, 因此, 菜单将出现在鼠标点击的位置 (见图12-11)。

6. 要删除名为Information的快捷菜单的话, 请在代码窗口输入并且运行下述过程Delete_ShortMenu:

```
Sub Delete_ShortMenu()
    Application.CommandBars("Information").Delete
End Sub
```

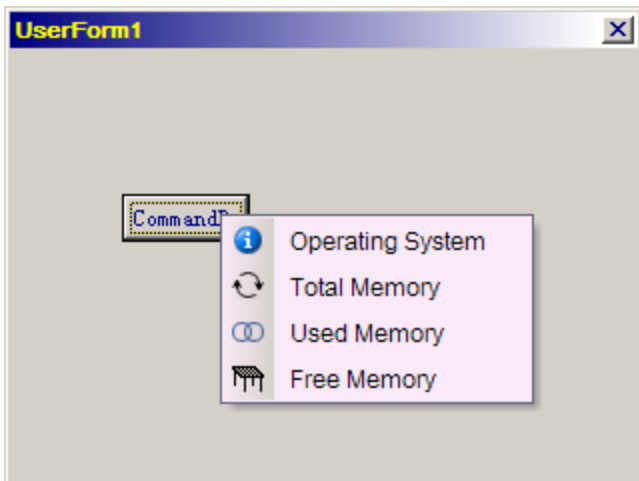


图12-11 当你右键点击一个对象时出现的自定义快捷菜单

13.接下来……

在本章, 你学习了如何使用VBA修改内置菜单和工具栏, 如何创建和显示你自己的工具栏, 菜单和快捷菜单。当使用菜单和工具栏时, 你使用了对象CommandBar的各种属性和方法。你学习了三种类型的对象CommandBar: Normal, MenuBar和Popup。使用立即窗口, 你试验了示范如何创建自己的工具栏和控件。

下一章将带你进入错误捕捉和调试, 换句话说, 你将学习当你的程序工作不正确时, 该做些什么。

第十三章 调试 VBA 过程和处理错误

错误要悄悄混入你的VBA过程很容易, 事实是, 无论你多么仔细, 你所有的VBA过程第一次就能全部运行正确, 这是极其少见的事。总有一些事情你错过了或者没有想到过。从第二章起, 你就知道有三种类型的VBA错误: 语法错误, 逻辑错误和运行时间错误。本章将介绍许多内置工具, 你会发现它们在你的过程代码的分析和定位错误源的过程中是很有用的。

1.测试 VBA 过程

迄今为止，在本书中，你已经创建和执行了很多过程和函数例子。因为这些程序中的大多数都很短，所以找错误并不是非常困难。然而，当你编写更长更复杂的过程时，查找错误源就更缓慢和费时了。幸运的是，VBA编辑器提供了一套方便的工具，让你追踪你VBA问题的过程更简单，更快捷，有更少的挫折。程序缺陷是电脑程序中的错误，而调试则是定位和解决这些错误的过程。调试让你找到你的程序为什么不按预期工作的原因。你可以通过步入程序代码或者检查变量值来达到目的。

使用下述指南进行你的VBA程序调试：

- 如果你想要分析你的过程，通过按F8或者选择调试-逐语句，逐语句地执行你的代码
- 如果你怀疑程序的某个地方有错误发生，那么可以使用断点
- 如果你想监测程序中某个变量或者表达式的值，那么可以添加一个监视表达式
- 如果你讨厌在冗长的程序代码中拉动滚动条到你感兴趣的部分去，那么你可以设置一个书签，快速跳到需要的地方。

每条指南在本章中都有实用的例子进行示范。

2. 终止过程

你知道如何终止VB过程吗？如果你想到了按Esc键，那么你对了。如果你在运行程序，并且突然按下Esc键，那么VB就会中断程序的运行，并显示如图13-1显示的信息。然而除了Esc键，这个在很多情况下都很有力而且可靠的方法，VBA还提供了很多其它的方法来中断你的过程，进入所谓的中断模式：

- 按Ctrl+Break
- 设置一个或多个断点
- 插入Stop语句
- 添加监视表达式

当你的程序执行被临时停止时，断点便发生了。VB会从过程的执行中记住所有变量和语句的值，当用户从工具栏点击运行宏（或者“运行”菜单上的相同名称），或者点击对话框（图13-1）上面的继续按钮，可以恢复。



图13-1 在你运行程序时，如果你按下Esc键或者Ctrl+Break键，就会出现该信息

图13-1显示的错误对话框通知你该过程已被中断，下述按钮可用：

继续	点击该按钮可以恢复代码执行。如果遇到错误该按钮将变灰
结束	如果你这次不想排除故障则点击该按钮，VBA将终止代码执行
调试	点击该按钮进入中断模式。代码窗口将出现，并且VBA会加亮过程执行时停止处的代码行。你可以检查，调试，中断或者逐句执行代码。 注意，当VBA工程被保护时，该按钮变灰
帮助	点击该按钮查看在线帮助，解释该错误信息的导致原因

技巧13-1 防止用户干预

通过将下述语句加入到过程代码中去，你可以防止用户中断你的程序：

```
Application.EnableCancelKey = xlDisabled
```

当用户在程序运行时按下Esc或者Ctrl+Break时，不会发生任何情况。应用程序对象的属性EnableCancelKey禁用了这些键。

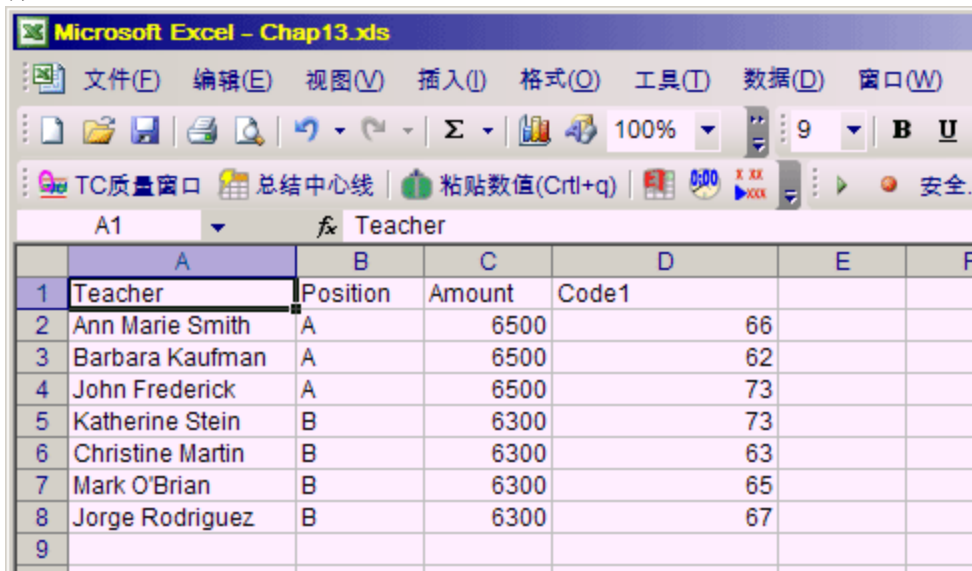
3.使用断点

如果你多少知道点你程序代码有问题，那么你应该在那里（给定的行）暂停代码执行。要设置断点，简单点就是当光标位于目标代码行时按下F9。当过程运行中VBA到达此处时，立即会显示代码窗口。这时，你可以通过按F8或者选择调试-逐语句来一行一行地运行代码。

我们来看看下面一个方案，看看它是如何工作的。假设在过程ChangeCode的执行中下面的代码行会出问题：

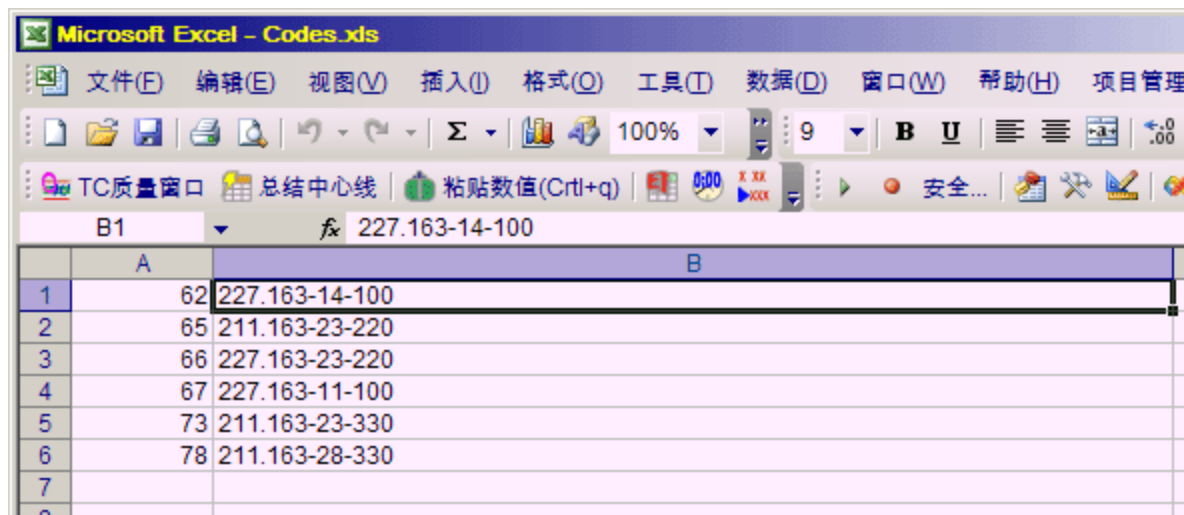
ActiveCell.Formula = "=VLookup(RC[1],Codes.xls!R1C1:R6C2,2)"

1. 准备好如图13-2和13-3所示的电子表格。保存图13-2所示的数据为Chap13.xls，图13-3所示的数据为Codes.xls。关闭文件Codes.xls。



	A	B	C	D	E	F
1	Teacher	Position	Amount	Code1		
2	Ann Marie Smith	A	6500	66		
3	Barbara Kaufman	A	6500	62		
4	John Frederick	A	6500	73		
5	Katherine Stein	B	6300	73		
6	Christine Martin	B	6300	63		
7	Mark O'Brian	B	6300	65		
8	Jorge Rodriguez	B	6300	67		
9						

图13-2 本文件中输入在D列的编号将会在过程ChangeCode中被显示于图13-3中的编号所代替



	A	B
1	62	227.163-14-100
2	65	211.163-23-220
3	66	227.163-23-220
4	67	227.163-11-100
5	73	211.163-23-330
6	78	211.163-28-330
7		
8		

图13-3 过程ChangeCode使用该编号表作查找目的

2. 激活文件Chap13.xls，切换到VB编辑器窗口
3. 使用属性窗口重新命名VBAProject(Chap13.xls)为Debugging
4. 插入模块到Debugging (Chap13.xls)工程，并且更改Name属性为Breaks
5. 输入过程代码ChangeCode，如下所示：

```
Sub ChangeCode()
    Workbooks.Open FileName:="C:\Codes.xls" '将此处文件路径改为你的真实路径
    Windows("Chap13.xls").Activate
```



```

Columns("D:D").Select
Selection.Insert Shift:=xlToRight
Range("D1").Select
ActiveCell.Formula = "Code"
Columns("D:D").Select
Selection.SpecialCells(xlBlanks).Select
ActiveCell.Formula = "=VLookup(RC[1],Codes.xls!R1C1:R6C2,2)"
Selection.FillDown

With Columns("D:D")
    .EntireColumn.AutoFit
    .Select
End With

Selection.Copy
Selection.PasteSpecial Paste:=xlValues
Rows("1:1").Select

With Selection
    .HorizontalAlignment = xlCenter
    .VerticalAlignment = xlBottom
    .Orientation = xlHorizontal
End With

Workbooks("Codes.xls").Close
End Sub

```

6. 在过程ChangeCode里，点击下述语句行的任意地方：
`ActiveCell.Formula = "=VLookup(RC[1],Codes.xls!R1C1:R6C2,2)"`
7. 按下F9（或者选择调试-切换断点）来在光标所在处设置一个断点。设置断点的另外一种方法是点击你要暂停程序的代码左边的页边。同时，带断点的代码行显示为白字紫红底色（见图13-4）。断点的颜色可以在选项对话框（工具菜单）的编辑器格式页上更改。

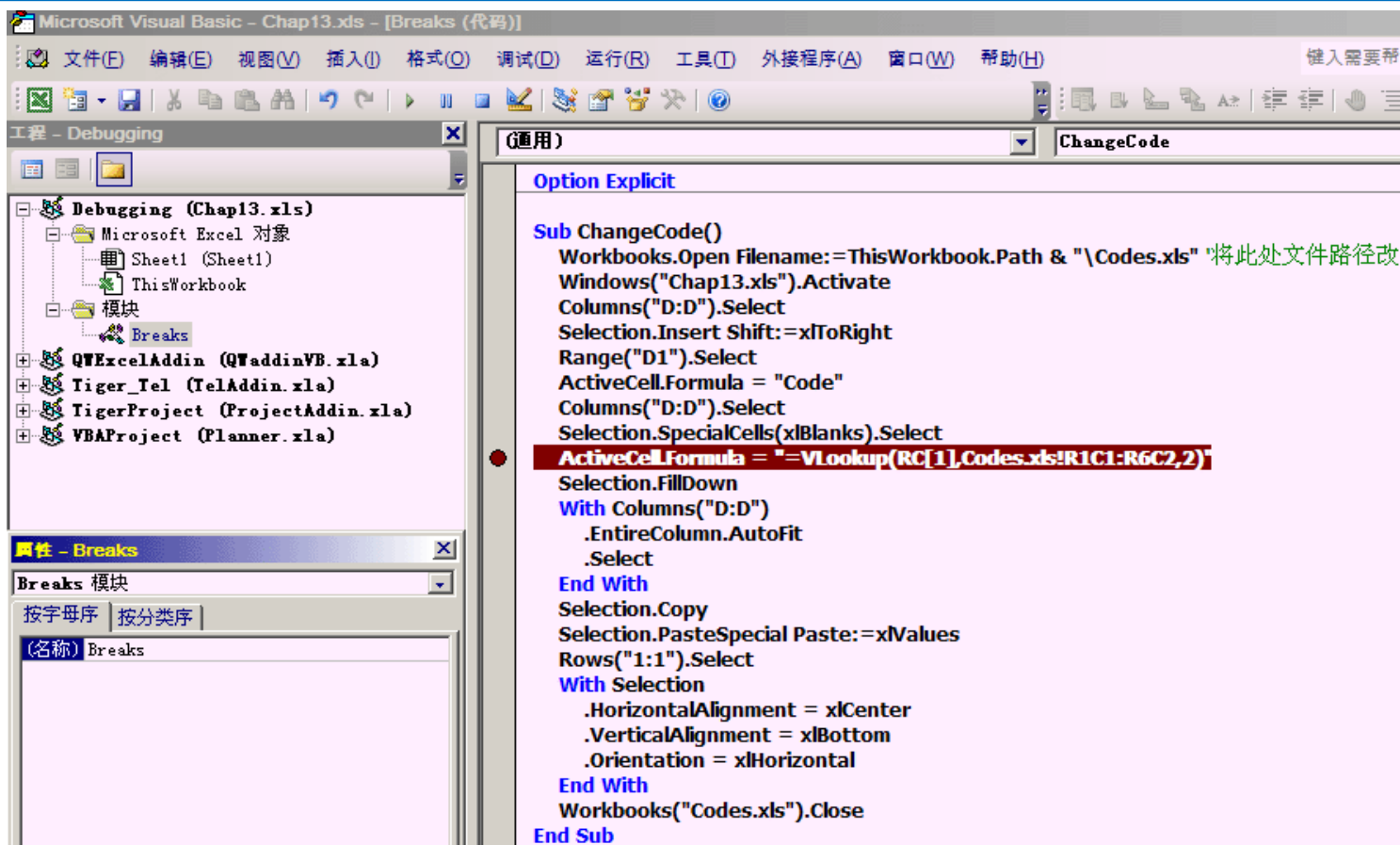


图13-4 设置了断点的代码行显示了选项对话框里编辑器格式设定的颜色

8. 运行过程ChangeCode。当你运行该过程时，VB将执行所有的语句，直到它遇到该断点。一旦遇到断点，代码便暂停了，并且屏幕显示代码窗口（见图13-5）。VB在该语句左边的页边上显示一个黄色箭头。同时，该语句出现在一个黄色底色的框里面。错误和框表明当前语句将要被执行。如果当前语句也包含一个断点，那么页边将重叠显示它们（圆圈和箭头）。

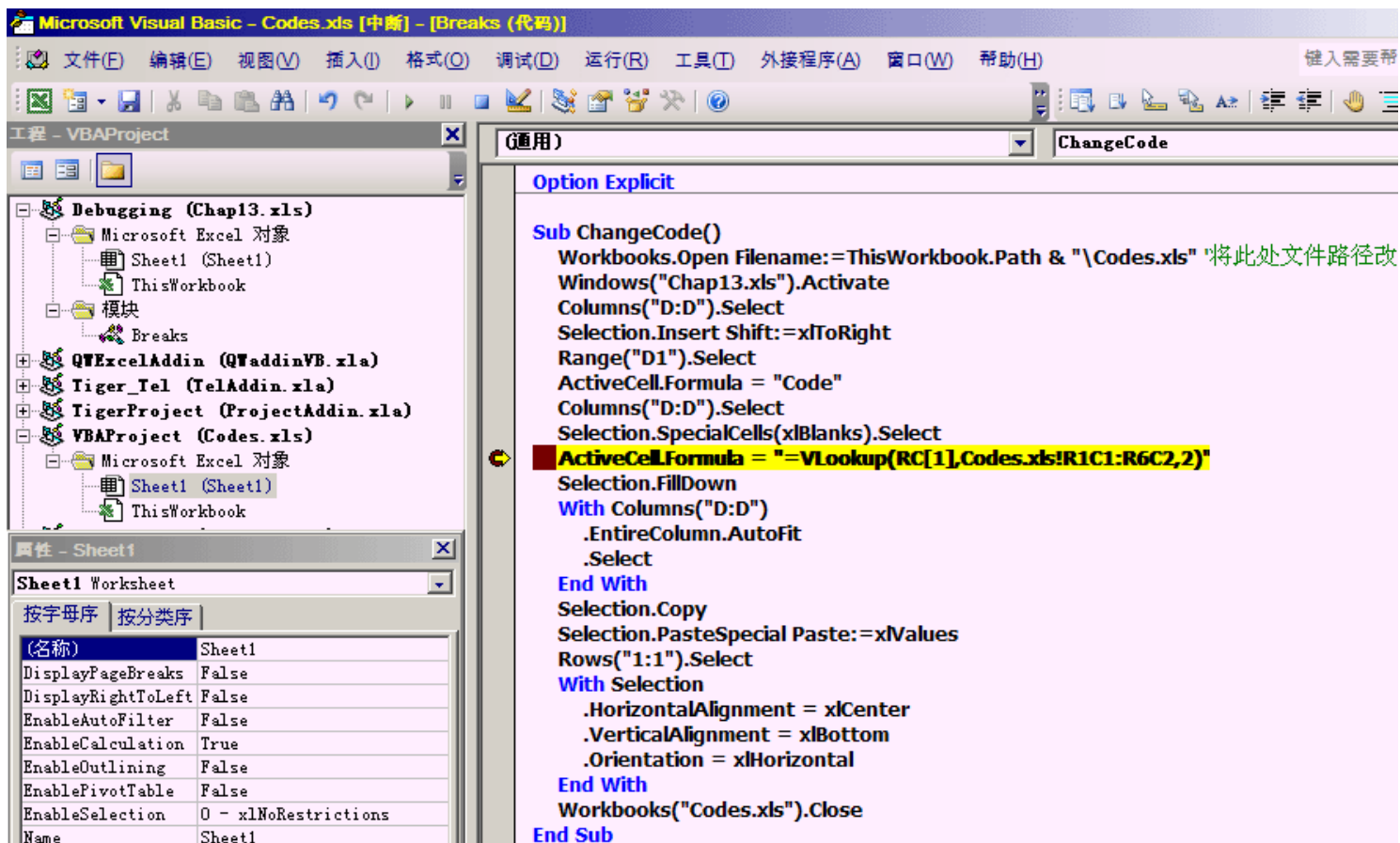


图13-5 当VB遇到断点时，就会显示代码窗口并且指定当前语句

9. 按F8，或者选择调试-逐语句
10. 重复几次步骤9的指令
11. 按F5（或者选择运行宏）来继续运行该过程，不必逐语句运行该过程。当你运行完该过程后，VB不会自动删除断点。注意带有VLookup函数的代码行还是加亮的。
在本例中，你只设置了一个断点。VB允许你在一个过程里设置任意多个断点。这样，你就可以随心所欲地暂停和继续你过程的执行了。你可以分析你的程序代码、检查执行暂停时变量的值。你也可以通过在立即窗口里敲入语句进行各种各样的测试。
12. 通过选择调试-清除所有断点，或者按下Ctrl+Shift+F9，可以清除断点。
所有断点都被清除了。如果你在某个过程里面设置了多个断点，并且想要只清除其中的一个或几个，那么可以点击你想要清除断点的代码行并且按下F9（或者选择调试-切换断点）。当断点不再需要时，你应该清除它们。当你关闭文件时，所有断点将自动被清除。

技巧13-2 什么时候使用断点

如果你怀疑你的过程根本就没有执行过某段代码块的话，那么设置断点。

4.在中断模式下使用立即窗口

一旦程序执行中断，当代码窗口出现时，你可以激活立即窗口并且输入VBA指令，例如，来查明哪一个单元格是当前活动的或者活动工作表名称是什么。你也可以使用立即窗口来更改变量的内容，以便改正可能导致错误的值。到现在，你应该已经是使用立即窗口的专家了。图13-6显示了该暂停的过程ChangeCode，和在中断模式下，向VB问问题的立即窗口。

在中断模式下，你可以很快地查明代码窗口里变量的内容。只要在运行的过程中简单地将光标移动到变量上，就可以知道该变量的内容了。例如图13-7里所示的VarValue过程，断点设置在第二次出现的Workbooks.Add语句上。

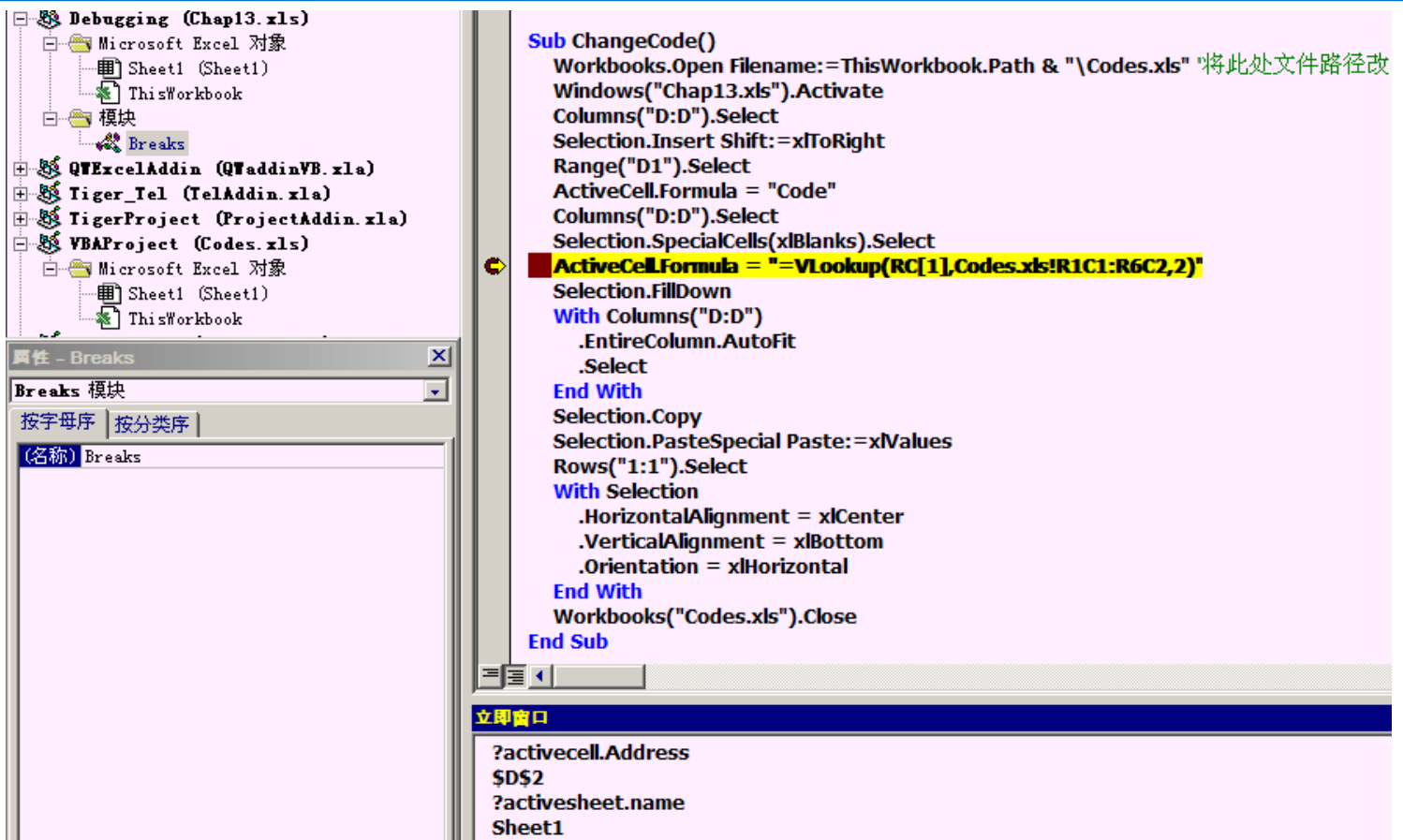


图13-6 当代码执行被暂停时，通过输入适当的语句到立即窗口里，你就可以找到很多问题的答案

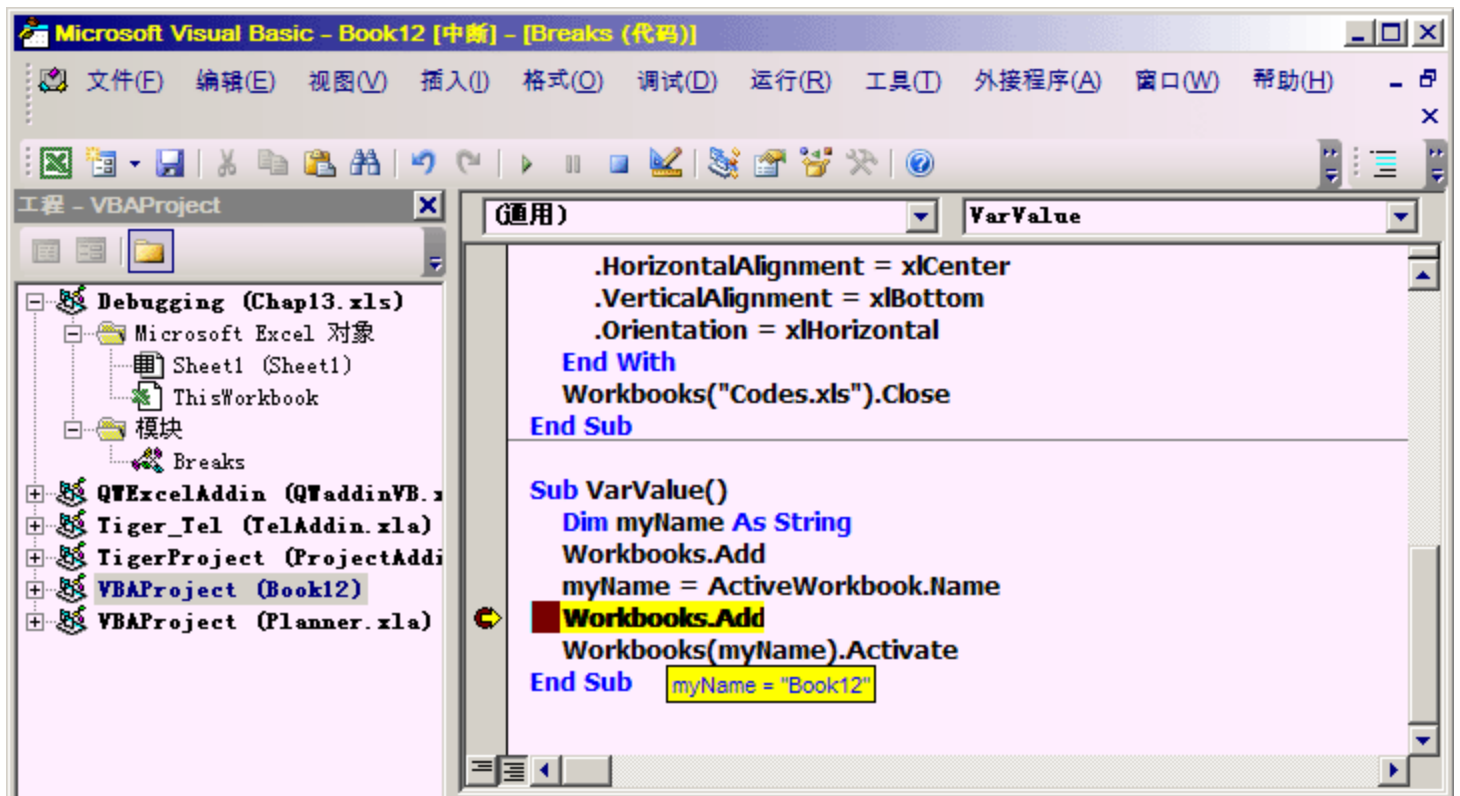


图13-7 在中断模式下，你可以将鼠标指向某个变量，查明该变量的值

当VB遇到该语句，代码窗口（中断模式）就会出现。因为VB已经执行了那条语句，将当前活动工作簿名称储存在变量myName（译者：原文为myBook），所以你将鼠标指向该变量名称时，就可以查明该变量的值。该变量名称及其当前值出现在文字框里面。要同时显示过程里使用的多个变量的值的话，你就应该使用本地窗口，本章的后面将会讨论到。

5.使用 Stop 语句

有时候，你不能马上测试你的程序，如果你设置了断点，然后关闭该文件，那么Excel就会清除你的断点，而下次当你准备测试程序时，你将不得不再次设置你的断点。如果你需要推迟测试工作，那么你可以使用不同的方式，简单的在你需要暂停程序的地方插入一个Stop语句。图13-8显示了For...Next循环之前的Stop语句。当VB遇到Stop语句，它就会暂停过程StopExample的执行，屏幕会显示中断模式下的代码窗口。尽管Stop语句和设置断点具有完全一样的效果，但是，它有一个弱点——所有Stop语句都留在程序里，直到你一个清除它们。当你不再需要暂停程序时，你必须定位并且清除所有的Stop语句。

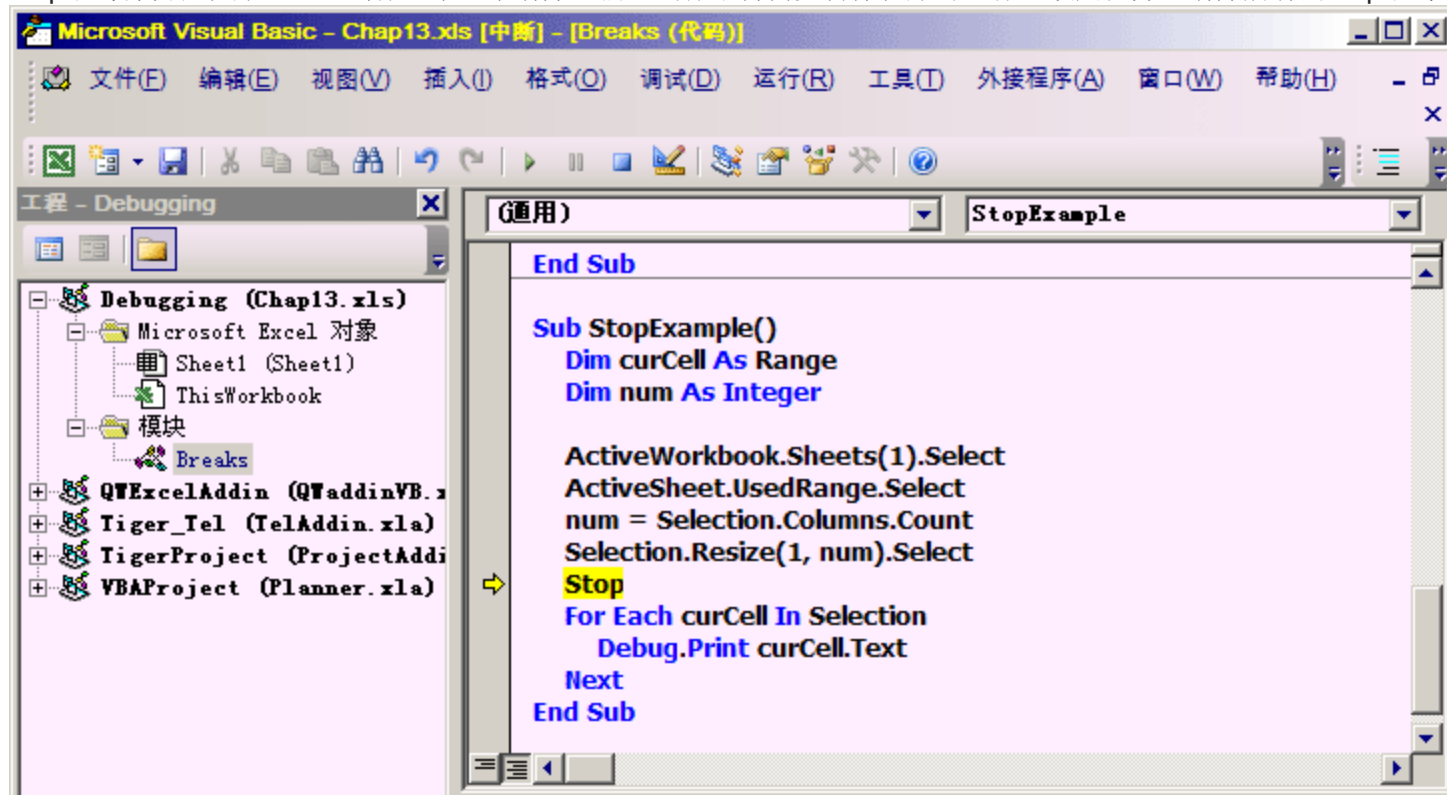


图13-8 你可以在VBA过程代码的任何地方插入Stop语句，当程序到达Stop语句时就会暂停，并且出现代码窗口，加亮该行

技巧13-3 在中断模式下在代码窗口内部工作

在中断模式下，你可以改变代码、添加新语句、每次执行一行语句、跳过代码行、设置下一条语句、使用立即窗口、以及更多。当VB处于中断模式时，调试菜单上所有的选项都可用了。你可以通过按下Esc，Ctrl+Break，或者通过设置断点，进入中断模式。

当你在中断模式下时，如果你更改某些代码，VBA将会提示你重新设置工程，显示如下错误信息：“该操作将重新设置工程，继续吗？”你可以点击确定，终止程序执行并继续编辑你的代码，或者点击取消，删除新变化并从中断点继续运行代码。要查看该错误的话，可以将程序进入中断模式，然后更改变量的声明，当你按下F5恢复代码执行的时候，VBA就会提示你重新设置你的工程。

6.添加监视表达式

程序中的许多错误是由变量获得未预期的值导致的。如果某个过程使用了一个变量，在不同的地方有不同的值，你可能想要停止程序查看该变量的当前值。VB提供了一个特别的监视窗口，允许你在过程运行时密切注视变量或者表达式。

进行下述操作，给你的过程添加监视表达式：

1. 在代码窗口，选择你想要监视的变量
2. 选择调试-添加监视

屏幕上会显示添加监视对话框，如图13-9所示。

添加监视对话框包含三部分，描述在下表中：

表达式	显示你的过程中加亮变量的名称。如果你打开添加监视对话框时没有选择变量名称，那么需要输入你想要监视的变量名称到表达式文字框里
上下文	在该节，你应该指明包含该变量的过程名称和该过程所在的模块名称
监视类型	明确如何监视该变量。如果你选择“监视表达式”选项按钮，那么你将在中断模式下能够在监视窗口里查看该变量的值。当你选择“当监视值为真时中断”选项按钮，那么当该变量值为真（非零）时VB将自动停止过程。最后一个选项按钮，“当监视值改变时中断”，每当该变量或表达式的值改变时，过程就会停止

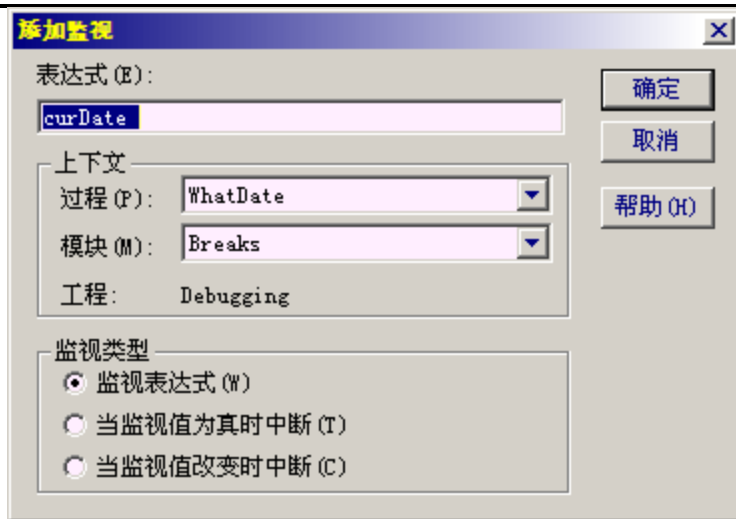


图13-9 添加监视对话框允许你定义在VBA过程运行时监视的情况

你可以在运行过程之前或者在过程执行中断之后添加监视表达式。

断点和监视表达式之间的区别是断点总是将过程中断在某个特定的位置，而监视表达式则是当特定情况（监视值为真中断或者监视值改变时中断）时中断过程。当你不确定变量在哪儿改变时，监视是极其有用的。你可以简单地添加一个监视断点在某个变量上并正常运行过程，而不必在这么多行代码里逐语句来找到变量在那里获取该特定的值。我们来看看这是如何实现的。

1. 准备如图13-10所示的过程

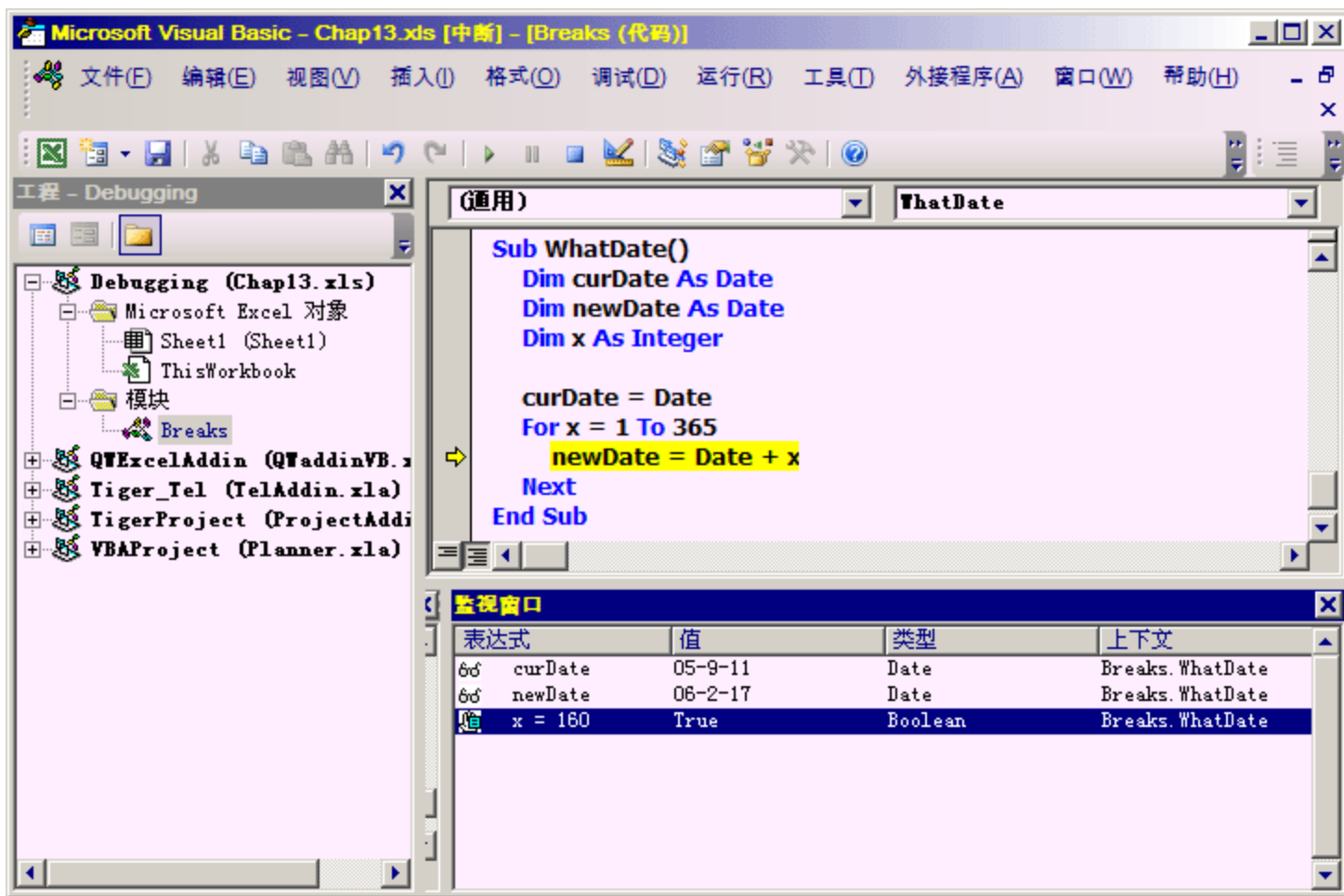


图13-10 使用监视窗口

过程WhatDate使用For...Next循环来计算将来x天后的日期。如果你运行该过程，你不会得到任何结果，除非你在过程里插入下述指令：

```
MsgBox "In " & x & " days, it will be " & NewDate
```

然而，这次，你不想一天一天地显示每一个日期。假如你想要当变量x等于160（译者：翻译时恰好用到160这个数，原文是211）的时候停止该程序，换句话说，你想要知道现在160天后是哪一天。要得到结果的话，你可以插入下述语句到过程里：

```
If x = 160 Then MsgBox "In " & x & " days it will be " & NewDate
```

假设你想要不输入任何新语句来得到结果，你如何做呢？如果你添加了监视表达式的话，当满足特定条件时，VB就会停止For...Next循环，然后，你就可以查看想要的变量值。

1. 选择调试-添加监视
2. 在表达式文字框里输入下述表达式：x=160
3. 在上下文部分，从过程下拉列表里选择WhatDate，从模块下拉列表里选择Breaks
4. 在监视类型部分，选择“当监视值为真时中断”选项按钮
5. 点击确定，关闭添加监视对话框，现在，你已经添加了你的第一个监视表达式
6. 在代码窗口，将光标放在变量curDate内部的任意地方
7. 选择调试-添加监视，并点击确定，设置缺省的监视类型
8. 在代码窗口，将光标放在变量newDate内部的任意地方
9. 选择调试-添加监视，并点击确定，设置缺省的监视类型

做完上面的步骤后，过程WhatDate包含了下述三个监视：

```
x = 160 当监视值为真时中断
```

```
curDate 监视表达式
```

```
newDate 监视表达式
```

10. 将光标放在过程WhatDate代码的任意地方，并且按下F5，VB在x=160的时候停下来了（见图13-10）

注意，变量x在监视窗口的值和你在添加监视对话框里指定的值一样。另外，监视窗口显示了变量curDate和newDate的值。该过程处于中断模式。你可以按F5继续或者你可以问另一个问题：277天后是哪一天？下一步将示范如何做。

11. 选择调试-编辑监视，然后输入下述表达式：x=227。你可以通过双击监视窗口里的表达式，快速显示编辑监视对话框
12. 点击确定，关闭编辑监视对话框。注意，现在监视窗口显示表达式的新值，现在x为False。
13. 按F5，当x值为227时过程再次停止。curDate的值相同，但是变量newDate现在有了一个新的值——现在之后277天的日期。你可以再次改变表达式的值，或者结束该过程。
14. 按F5以完成该过程。当你的过程正在运行，并且监视表达式有值，监视窗口就会显示该监视表达式的值，如果你在过程运行结束后打开监视窗口的话，那么你将看到<溢出上下文>，而不是变量值了。换句话说，当监视表达式溢出上下文时，它没有值。

7.清除监视表达式

在监视窗口里，点击你要清除的表达式并且按下Delete。清除你先前定义的所有监视表达式。

8.使用快速监视

如果你想查看一个表达式的值，但是你还没有定义监视表达式，那么你可以使用快速监视（见图13-11）

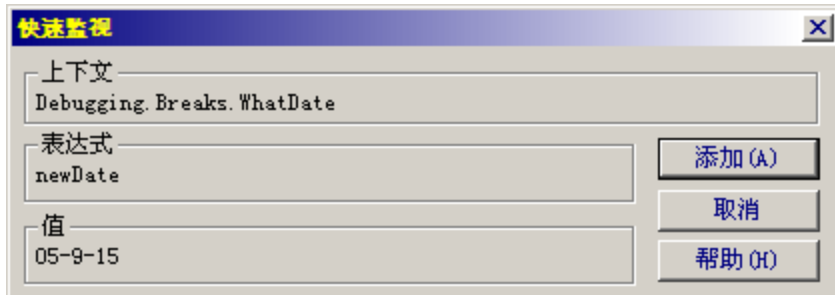


图13-11 快速监视对话框显示VBA过程里所选表达式的值

可以通过下述方法获取快速监视对话框：

- 在中断模式下，将光标放在你要监视的变量名称或者表达式内部
- 选择调试|快速监视，或者按下Shift+F9

快速监视对话框上面有个添加按钮，允许你在监视窗口里添加表达式。

确保过程WhatDate里不含有任何监视表达式，参见前面的章节有关如何从监视窗口清除监视表达式的内容。现在我们通过例子来看看如何利用快速监视。

1. 在过程WhatDate里，将插入点（光标）放在变量x处
2. 选择调试|添加监视
3. 输入下述表达式：
x = 50
4. 选择当监视值为真时中断，并点击确定
5. 运行过程WhatDate
当x等于50时VB将中断过程的执行，注意，监视窗口里没有变量newDate和curDate。想要查看这些变量的值的话，那么你可以将光标放在代码窗口里相应变量名称上，或者，你也可以调用快速监视窗口。
6. 在代码窗口里，将鼠标光标放在变量newDate上并且按下Shift+F9。快速监视窗口就会显示该表达式名称和其当前值
7. 点击取消返回代码窗口
8. 在代码窗口，将鼠标光标放在变量curDate上并且按下Shift+F9。现在，快速监视窗口就会显示变量curDate的值了
9. 点击取消返回代码窗口
10. 按下F5继续运行该过程

9.使用本地窗口和调用堆栈对话框

如果在VBA过程的执行过程中，你想密切注视所有声明的变量和它们的当前值，那么确保你在运行该过程前选择视图|本地窗口。当在中断模式下时，VB就会显示一系列的变量和它们相应的数值在本地窗口里（参见图13-12）

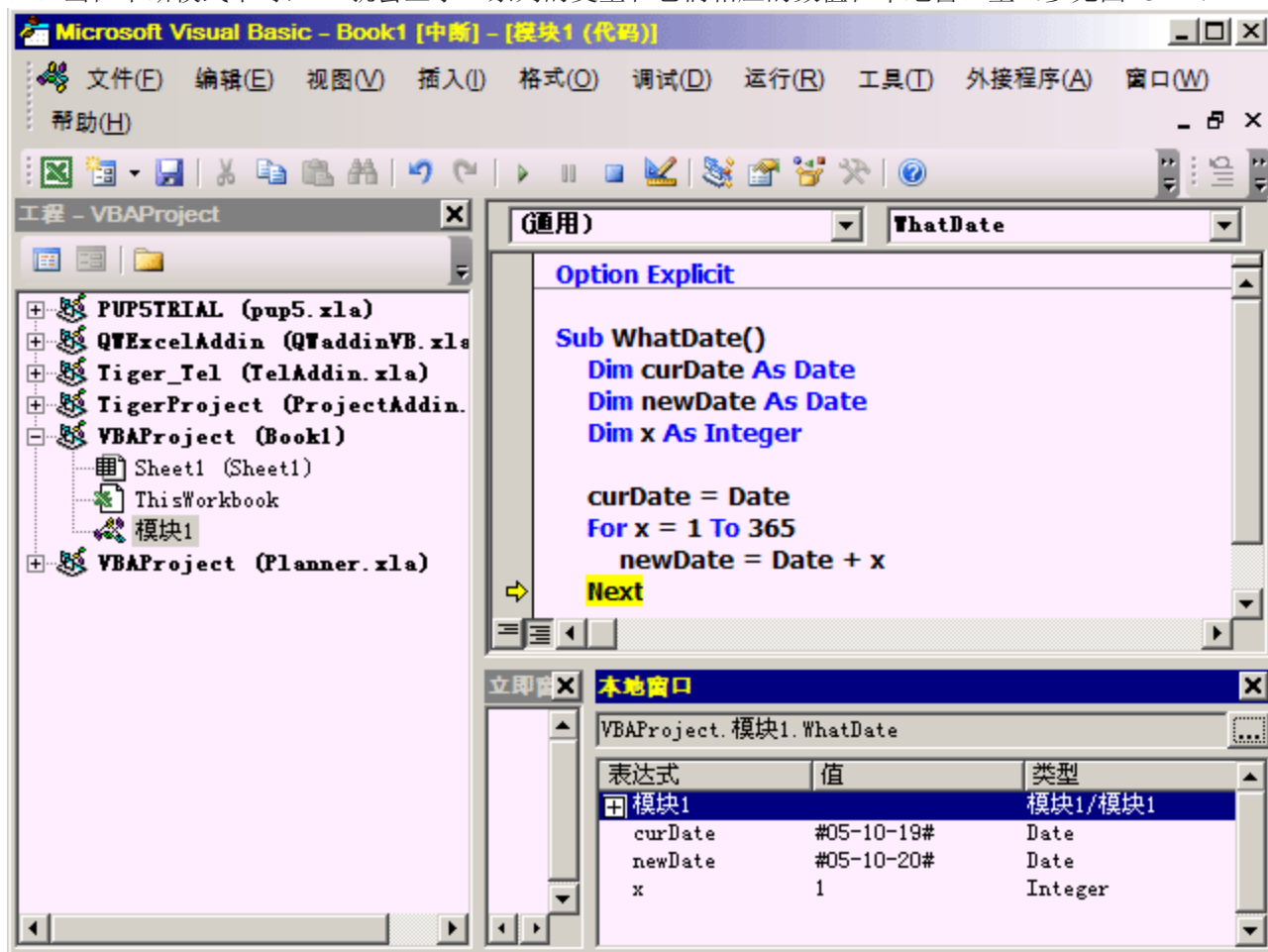


图13-12 本地窗口显示当前VBA过程里所有声明的变量和它们当前值

本地窗口包含三列，表达式列显示声明在当前过程里的变量名称。第一行显示前面带加号的模块名称，当你点击该加号，你就可以查看是否有变量声明在模块级。类模块将显示系统变量Me。在本地窗口，全局变量和被其它工程使用的变量不会显示出来。

第二列显示变量的当前值，在本列，你可以更改变量的值，只要点击它并输入新的值。更改了数值后，按下回车键以记录该变化。你也可以在更改数值后，按Tab键，Shift+Tab键或者向上或向下箭头，或者也可以点击本地窗口的其它任意地方。第三列显示每个声明了的变量的类型。

想要观察本地窗口里的变量值的话，请跟着做：

1. 选择视图|本地窗口
2. 点击过程WhatDate里的任意地方，并按F8，你将过程置于中断过程了。本地窗口显示了当前模块的名称，以及当地变量和它们的初始值
3. 按几下F8键，密切关注本地窗口
4. 按F5键继续运行该过程

本地窗口也包含一个带三个点的按钮，该按钮将打开调用堆栈对话框（参见图13-13），它显示所有活动调用过程的清单。活动的调用过程是指已经开始但是还没有完成的过程。你也可以通过选择视图|调用堆栈，该选项只有在中断模式下才是可用的。

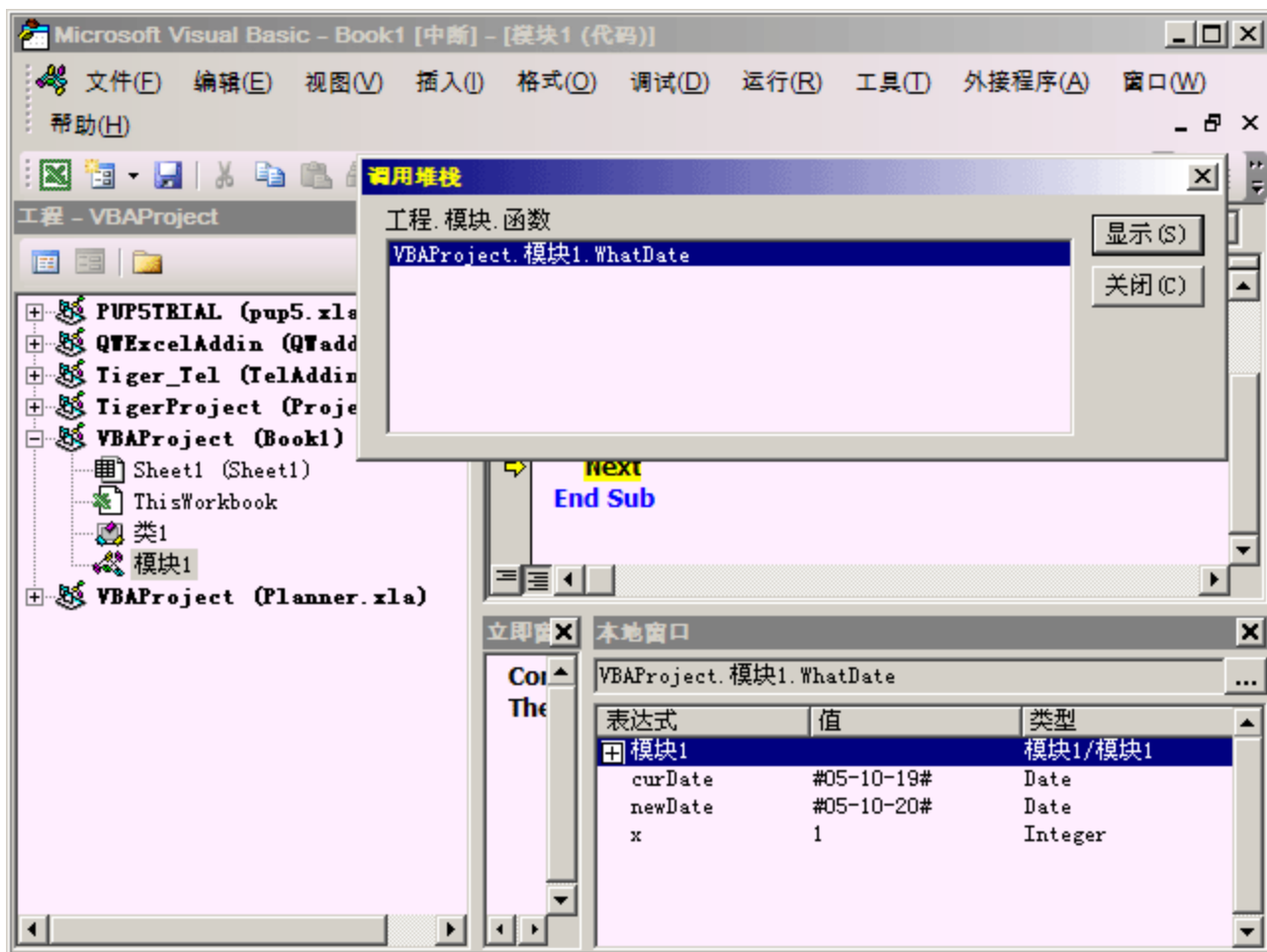


图13-13 调用堆栈对话框显示了开始但未完成的过程列表

调用堆栈对话框特别是在追踪嵌套的程序时有用。回想一下，嵌套过程是一个被另一个过程调用的过程。如果一个过程调用另一个过程，该被调用的过程名称就会自动添加到调用堆栈对话框里的调用列表。当VB执行完该被调过程后，该过程名称就会从调用堆栈对话框里自动清除。你可以使用调用堆栈对话框上的显示按钮，显示调用下一个过程的语句。

10. 逐句运行 VBA 过程

逐句运行代码意思是每次只运行一条语句，这样，可以允许你检查遇到的每一个过程里的每一条语句。想要从头开始逐句运行过程的话，可以将插入点置于过程代码的任意地方，并且选择调试|逐语句，或者按下F8。调试菜单包含好几个选项供你在逐步模式下执行（参见图13-14）

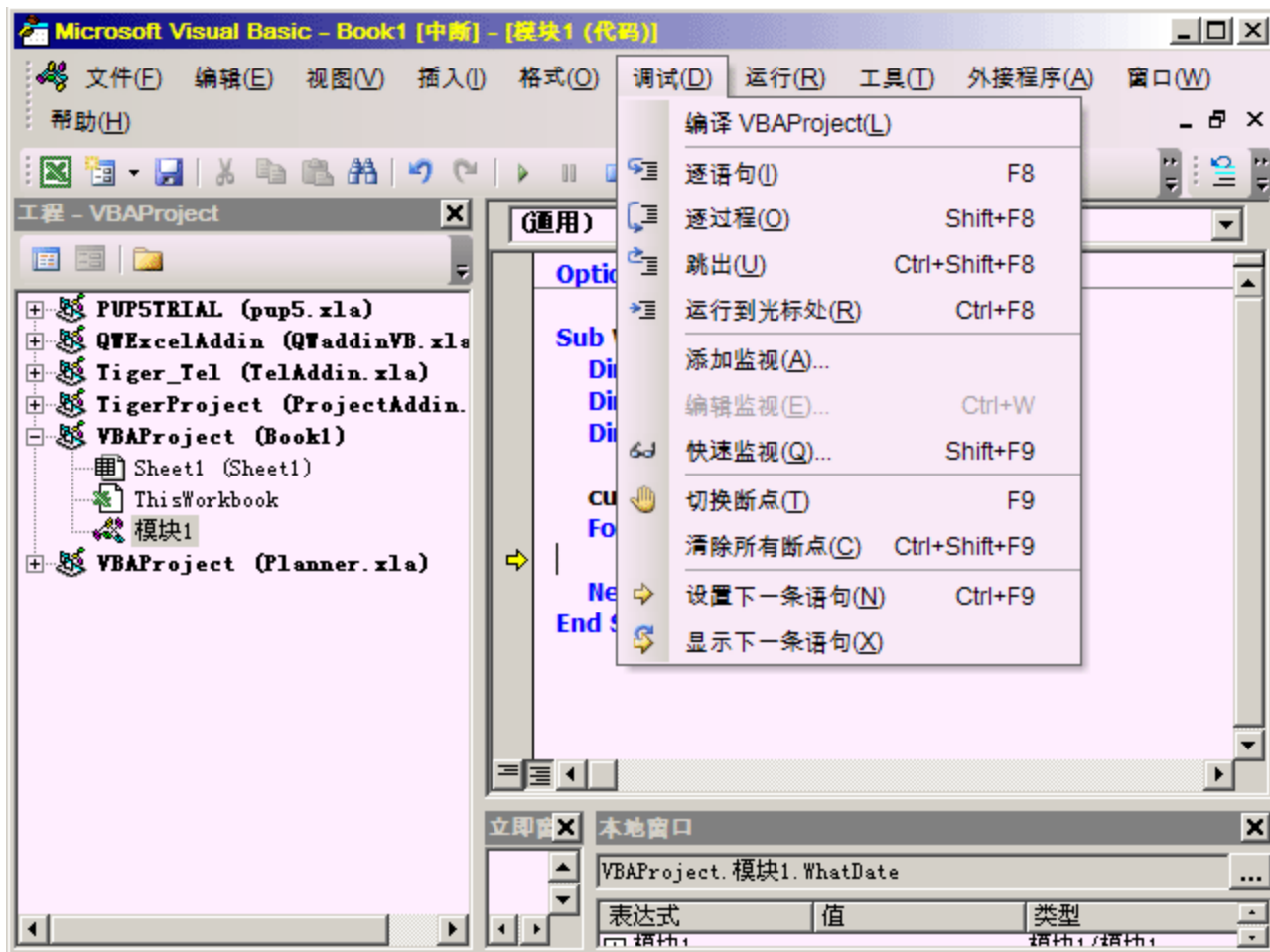


图13-14 调试菜单提供了许多命令以逐步VBA过程

当你每次运行一条语句时，VB将会执行每条语句，直到它碰到关键字End Sub。如果你不希望VB逐句运行的话，那么你可以随时按下F5来运行过程中剩余的代码，而不必逐步运行。

11.逐句运行过程

1. 将插入点置于你想要追踪的过程中代码的任意地方
2. 按下F8或者选择调试|逐语句。VB就会执行当前语句，并且自动跳到下一句并中断执行。在中断模式下，你可以激活立即窗口，监视窗口或者本地窗口，查看某特定语句中变量和表达式的值。以及，如果你正在逐语句执行的过程调用了其它过程，那么你也可以激活调用堆栈窗口来查看当前哪些过程是活动的
3. 再次按下F8，执行被选中的语句。执行完该语句后，VB会选中下一条语句，并且该过程将再次中断
4. 按下F8继续逐语句执行该过程，或者按F5无停止的执行完剩余的代码。你也可以选择运行|重新设置来终止该过程的执行，而不执行剩下的语句

当你逐过程运行某个过程（Shift+F8）时，VB将一次执行一个过程，好像里面只有一条语句一样。如果某过程调用了其它过程，并且你并不像逐语句执行这些过程，因为你已经测试过了，或者因为你只想侧重于尚未被调试的新代码，那么该选项特别有用。

12.逐过程执行过程

假设过程MyProcedure的当前语句调用过程SpecialMsg。如果你选择调试|逐过程（Shift+F8），而非调试|逐语句（F8），那么VB就会快速地执行过程SpecialMsg里面的所有语句，并且选择主调过程MyProcedure里的下一条语句。在过程SpecialMsg的

执行期间，VB将继续显示当前过程于代码窗口。

1. 在当前模块里输入下述过程：

```
Sub MyProcedure()
    Dim myName As String

    Workbooks.Add
    myName = ActiveWorkbook.Name
    ' choose Step Over to avoid stepping through the
    ' lines of code in the called procedure - SpecialMsg
    SpecialMsg myName
    Workbooks(myName).Close
End Sub

Sub SpecialMsg(n As String)
    If n = "Book2" Then
        MsgBox "You must change the name."
    End If
End Sub
```

2. 在下面语句处添加一个断点：

```
SpecialMsg myName
```

3. 将插入点置于过程MyProcedure的代码中，并按下F5运行它。VB到达断点时将中断执行
4. 按下Shift+F8，或者选择调试|逐过程。VB将会快速的运行过程SpecialMsg并且跳到紧挨着调用过程SpecialMsg的语句下面的那条语句
5. 按下F5无间断地完成过程的运行

当你不要分析被调过程中具体语句的话，逐过程执行是非常有用的。

调试菜单上的另外一个命令，跳出（Ctrl+Shift+F8），当你步入了某个过程，然后决定不继续逐步执行它，那么就可以使用该命令。当你喧杂该选项时，VB就会一步执行完该过程里的剩余语句，然后继续去激活主调过程中的下一条语句。

在逐步运行过程期间，你可以在逐语句，逐过程和跳出选项之间切换。选择哪个取决于这时你想要分析哪个代码片断。

调试菜单中运行到光标处（Ctrl+F8）命令让你运行过程，直到碰到你选中的行。如果你想要在运行一个大循环之前停止，或者想要跳过一被调过程时，该命令非常有用。

假设你想要执行过程MyProcedure到调用过程SpecialMsg的行。

1. 点击语句SpecialMsg myName内部
2. 选择调试|运行到光标处。当到达特定行时，VB将停止执行
3. 按下Shift+F8以逐过程地运行过程SpecialMsg
4. 按下F5无间断地执行完剩下的代码

13.设置下一条语句

有时，你也许想要重新运行过程中前面的几行代码，或者想要跳过一段将导致麻烦的代码。每遇到这种情况，你可以使用调试菜单里的设置下一条语句选项。当你中断过程的执行时，你可以随意恢复任何语句。VB将会跳过所选语句和中断处语句之间的语句。假设在过程MyProcedure（参见前面部分的代码）中，你已经在调用过程SpecialMsg的语句处设置了断点。要跳过程SpecialMsg的执行，你可以将光标置于语句Workbooks(myName)内，关闭并按下Ctrl+F9（或者选择调试|设置下一条语句）。除非你中断了过程的执行，否则不能使用设置下一条语句选项。

技巧13-4 跳过代码行

尽管跳过代码行在你的过程调试中非常有用，但是你得非常小心。当你使用下一条语句选项时，你告诉VB这是你想要执行的下一条语句。中间的所有代码行将被忽略，这意味着期间有很多你本预期要发生的事情并没有发生，这将可能导致意想不到的错误。

14.显示下一条语句

如果你不肯定过程的执行会从哪里继续，那么你可以选择调试|显示下一条语句，这样VB就会将光标放置到下次将运行的代码行。当你正在看别的过程，不知道下面会执行哪条代码的时候，该命令尤其有用。显示下一条语句选项仅在中断模式下可用。

15.终止和重新设置 VBA 过程

任何时候在逐步过程代码时，你可以：

- 按下F5无间断地执行剩余指令
- 选择运行|重新设置来终止过程，而不执行剩下的语句

当你重新设置过程时，所有变量将丢失它们的当前值。数字型变量恢复为其初始值0，变化长度的字符串变量初始化为0长度字符串（""），而固定长度的字符串用ASCII码0代表的字符或者Chr(0)填充。Variant型变量初始化为Empty，对象变量则设置为Nothing。

16.了解和使用条件编译

当你第一次运行某个过程时，VB会把你使用的VBA语句转变为计算机能够理解的机器码。该过程被称为编译。你也可以选择调试|编译（当前VBA工程名称），在你运行该过程之前执行整个VBA工程的编译。

使用条件编译，你可以告诉VB在编译或者运行时包括或者忽略某些代码块。取决于你设置的条件，你的过程可能会表现不同。例如，条件编译用来编译一个将会运行于不同平台（Windows 或者Macintosh，Win16或者Win32）上的应用软件。条件编译对于本地化使用于不同语言的应用软件也是很有用的。在条件编译时排除的程序代码将从最终文件中忽略掉，因此，它对文件大小或程序功效没有影响。

要激活条件编译的话，你应该使用叫做指示的特殊表达式。首先，你需要使用#Const指示声明一个布尔值（True或者False）常量，接下来，你在#If...Then...#Else指示中核实该常量。你需要进行条件编译的代码部分必须包括在这些指示中。注意，关键字If和Else前面都带有一个数字符号（#）。

如果一部分代码将要运行，那么该条件常量必须设置为真（-1），否则为假（0）。在模块的声明部分声明条件常量，例如：

```
#Const User = True
```

声明名为User的条件常量。

在接下来的过程中，当名叫verPolish条件常量为True时，数据就显示为波兰语。过程WhatDate调用函数DayOfWeek，它基于提供的日前返回星期名称。要用英语编译该程序的话，你所要做的全部就是将该条件常量改为False，然后VB就会跳到#Else指示后面的指令块去。

1. 在当前VBA工程插入一个新模块，并重命名为Conditional
2. 输入下述过程和函数：

```
' declare a conditional compiler constant
#Const verPolish = True
```

```
Sub WhatDay()
    Dim dayNr As Integer

    #If verPolish = True Then
        dayNr = WeekDay(InputBox("Wpisz date, np. 01/01/2000"))
        MsgBox "To bedzie " & DayOfWeek(dayNr) & "."
    #Else
        WeekdayName
    #End If
End Sub
```

```
Function DayOfWeek(dayNr As Integer) As String
    DayOfWeek = Choose(dayNr, "niedziela", "poniedzialek", "wtorek", _
        "sroda", "czwartek", "piatek", "sobota")
```

End Function

```
Function WeekdayName() As String
    Select Case WeekDay(InputBox("Enter date, e.g. 01/01/2000"))
        Case 1
            WeekdayName = "Sunday"
        Case 2
            WeekdayName = "Monday"
        Case 3
            WeekdayName = "Tuesday"
        Case 4
            WeekdayName = "Wednesday"
        Case 5
            WeekdayName = "Thursday"
        Case 6
            WeekdayName = "Friday"
        Case 7
            WeekdayName = "Saturday"
    End Select
    MsgBox "It will be " & WeekdayName & "."
End Function
```

3. 运行过程WhatDay。因为条件常量（verPolish）在模块顶端已被设置为True了，所以，VB将运行波兰版过程WhatDay。它用波兰语询问用户输入日期并且将结果显示为波兰语。要运行代码的英语版的话，需要将常量verPolish设置为False，然后重新运行过程

除了在模块顶部声明条件编译常量之外，你也可以选择工具|（VBAProject）属性（参见图13-15）。当你使用该属性窗口，在条件编译参数文本框里输入下述内容，以激活过程WhatDay的英语版本：

verPolish = 0

如果还有更多的条件编译常量的话，每个常量之间必须用冒号分割开。

4. 注释掉模块上部的#Const verPolish指示，并且在如图13-15所示地属性对话框里输入条件编译常量。然后运行过程WhatDay，看看Else部分是如何执行给说英语的用户的。

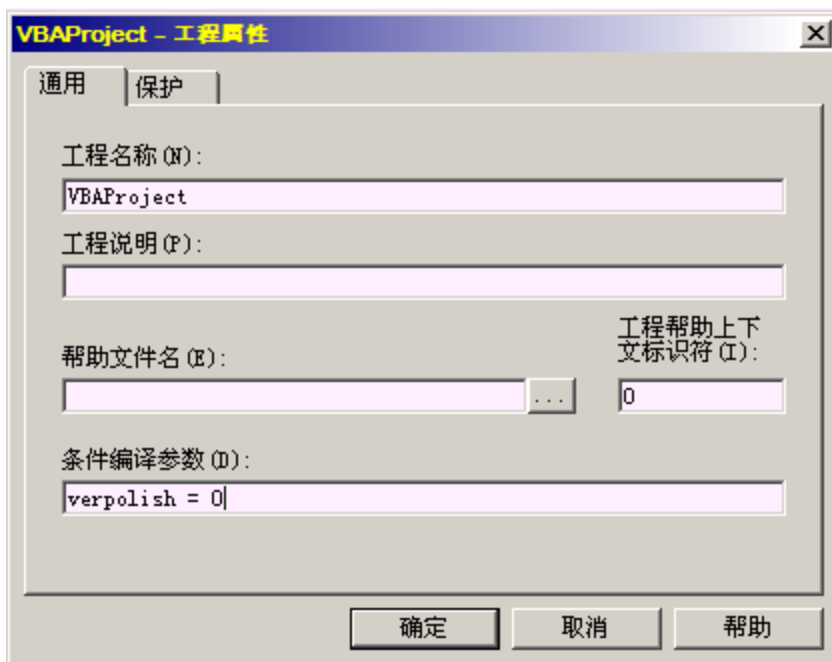


图13-15 条件编译常量可以在模块上部也可以在属性窗口声明，但是，不能同时在两个地方声明

17. 操纵书签

在分析和回顾你的VBA程序的过程中，你经常会发现你自己跳进了某代码区域。使用内置的书签功能，你可以轻易地标示你需要浏览的地方。设置书签：

1. 点击你想要定义为书签的语句的任意地方
2. 选择编辑|书签|切换书签（或者点击编辑工具栏上的切换书签按钮——参见图13-16）。VB将在语句左边的边界上放置一个蓝色的圆角矩形。

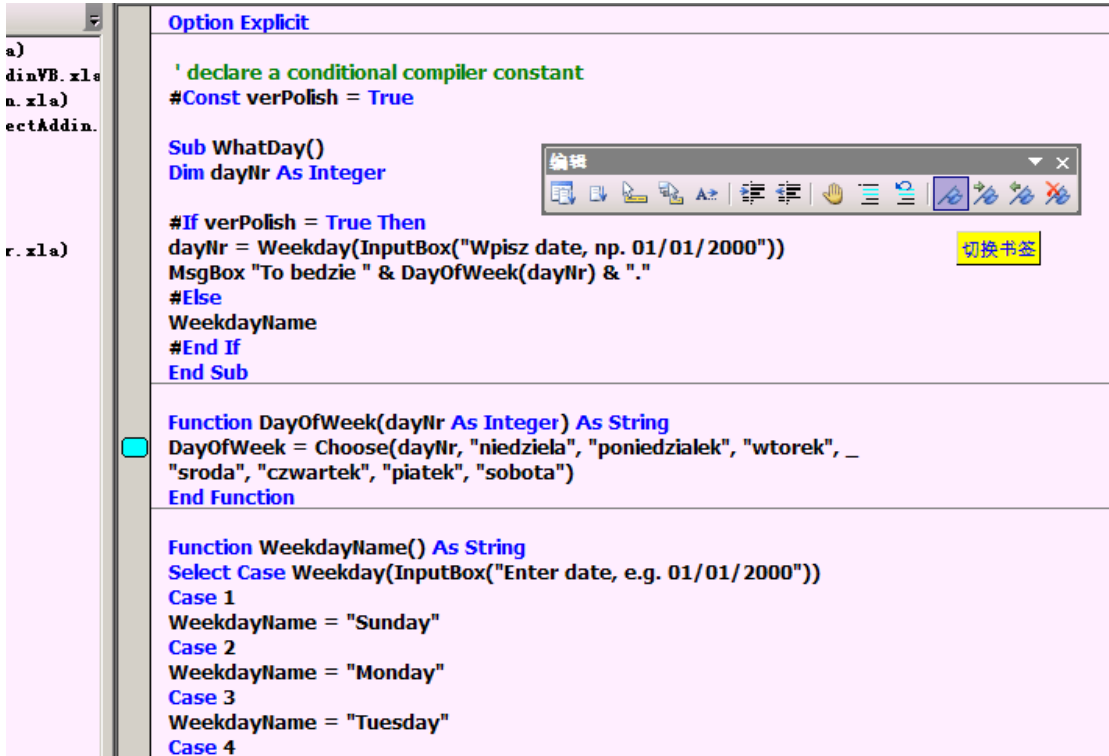


图13-16 你可以使用书签在经常要用的部分之间切换

你一旦设置了两个或以上的书签，就可以通过选择编辑|下一书签，或者简单地点击编辑工具栏上的下一书签按钮，在标志的代码处切换。你也可以在代码窗口的任意地方单击右键（译者：没有验证该快捷菜单），然后选择快捷菜单上的下一书签。要到前面的书签那里，则选择上一书签。

你随时可以通过选择编辑|书签|清除所有书签，或者点击编辑工具栏上的清楚按钮来清除所有书签。要清除单个书签的话，那么只要点击书签的任意地方然后选择编辑|书签|切换书签，或者点击编辑工具栏上的切换书签按钮。

18. 捕捉错误

没有人第一次就编写没有错误的程序。当你创建VBA过程，你必须决定你的程序如何应对错误。许多意想不到错误在运行时发生，例如，你的过程可能要试图给一个工作簿一个已经打开的工作簿的名称。运行时间错误经常不是被程序员发现，而是被试图做一些程序员没有预测到的事情的用户发现。如果程序运行时错误发生了，那么VB将显示一个错误信息，并且程序终止。大多情况下VB显示的错误信息对用户来说很隐秘。你通过在你的VBA过程里加入错误处理代码，预防用户经常看到运行时间错误。这样，当VB碰到错误，它就会显示一个更友好更好理解的错误信息，可能指导用户如何去改正错误，而不是简单的显示一个缺省的错误信息。

如何在你的VBA过程里实行错误处理呢？第一步，要将On Error语句放到你的程序里。该语句告诉VBA当运行时发生错误应该做什么，换句话说，VBA使用On Error 语句来激活错误处理程序以捕捉运行时间错误。取决于你的程序类型，你可以通过以下任何方式推出错误陷阱：Exit Sub, Exit Function, Exit Property, End Sub, End Function或者End Property。你应该给每个过程写一个错误处理程序。

On Error语句可以按下述方式之一使用：

On Error GoTo 标签	明确一个标签，当错误发生时跳到该标签。该标签标示错误处理程序的开始。错误处理是在你的应用软件中用来捕捉错误并作出响应的程序。该标签必须和On Error语句出现在同一过程里面
On Error Resume Next	当运行时间错误发生时，VB将忽略该导致错误的代码行，不显示错误信息，但是从下一行开始继续运行程序
On Error GoTo 0	关闭程序里的错误捕捉。当VBA运行该语句后，错误会被发现，但是没有错误陷阱在程序里

技巧13-5 是错误(Error)还是失误(Mistake)

在编程中，错误与失误并非相同的事情。失误，比如错误拼写，漏掉语句，放错地方的引号或逗号，或者给变量赋予了不匹配的值，通过适当的调试失误是可以从程序中清除的。尽管你的程序没有任何失误，但是，这并不意味着不会发生错误。错误是指一个事件或者操作没有按预期工作。例如，如果你的VBA程序需要访问硬盘上某个具体的文件，但是某人将该文件删除了或者移到别的地方去了，不管什么你总会得到一个错误。错误阻止程序完成具体任务。

下面显示的过程Archive使用了错误处理程序（见过程的下部）。该过程使用内置的方法SaveCopyAs，将当前工作簿保存复件到一文件，而不修改该已打开的工作簿在内存中的情况。

1. 在当前工程里插入一个新模块，并重命名为Traps
2. 输入过程Archive，如下所示：

```
Sub Archive()
    Dim folderName As String
    Dim DriveA As String
    Dim BackupName As String
    Dim Response As Integer

    Application.DisplayAlerts = False

    On Error GoTo DiskProblem

    folderName = ActiveWorkbook.Path
    If folderName = "" Then
        MsgBox "You can't copy this file. " & Chr(13) & "This file has not been saved.", _
            vbInformation, "File Archive"
    Else
        With ActiveWorkbook
            If Not .Saved Then .Save

            DriveA = "A:"

            MsgBox "Place a diskette in drive " & DriveA & " and click OK.", , "Copying to " & DriveA

            BackupName = DriveA & .Name
            .SaveCopyAs Filename:=BackupName

            MsgBox .Name & " was copied to a disk in drive " & DriveA, , "End of Archiving"
        End With
    End If

    GoTo ProcEnd

```

DiskProblem:

```

Response = MsgBox("There is no disk in drive A " & Chr(13) _
    & "or disk in drive " & DriveA & " is not formatted ", _
    vbRetryCancel, "Check Disk Drive")
If Response = 4 Then
    Resume 0
Else
    Exit Sub
End If
ProcEnd:
Application.DisplayAlerts = True
End Sub

```

声明完变量后，过程Archive的语句Application.DisplayAlerts = False 确保VB在运行时，不会显示自己的警告和信息。下一条语句，On Error GoTo DiskProblem，明确了一个标签，当发生错误时跳过去。保存活动工作簿的路径名称存储在变量folderName上。

当VB找不到该工作簿路径时，就会假设该文件没有保存并且显示相应的信息。接下来，VB跳到End If之后的语句处，并执行指令 GoTo ProcEnd，指向仅在End Sub之前的ProcEnd标签。注意，标签带有一个冒号。VB执行语句Application.DisplayAlerts = True，恢复系统的内置警告和信息。因为没有语句了，所以，过程结束。

如果活动工作簿的路径不是空字符串的话，那么VB就会检查该工作簿最近的更改是否已保存。如果没有，VBA使用语句If Not .Saved Then .Save来保存活动工作簿。Saved是工作簿对象的VBA属性。接下来，VB将软盘驱动名称"A:"存储到变量DriveA并且显示信息提示用户插入软盘。然后软盘名称和活动工作簿名称合并在一起，并且存储到一个叫BackupName的变量上。

你应该知道，当往软盘拷文件的时候，所有的事情都可能出错。例如，软驱可能是空的，或者软盘未格式化或已经满了。当VB检测到一错误时，它就会跳到以标签DiskProblem开始的代码行去，并且会显示相应的信息。如果用户点击了信息框上的重试按钮(值为4)，那么VB就执行语句Resume 0，该语句就会将VB送到导致错误的语句那里 (.SaveCopyAs FileName: = BackupName)，然后VB会再次执行它。如果用户点击取消按钮的话，VBA就会执行语句Exit Sub，过程结束。

如果软驱里面的A盘没有问题，那么VBA就会复制活动工作簿到该软盘，并且信息框会通知用户复制操作已成功。

3. 运行几次过程Archive，每次响应不同的选项，确保尽可能多的可能性。使用你在本章学习的多种调试技术。

技巧13-6 程序测试

你对你编写的代码负责，这意味着你在发布你的程序给其他人测试之前，你自己先测试它。毕竟，你了解它应该如何工作。有些程序员认为测试他们自己的代码是一种降格的事情，特别是当他们在有一个专门测试部门的组织中工作的时候。不要犯这种错误。程序员级别的测试过程是非常重要的，如同编写代码本身一样。在你自己测试完过程后，你应该给用户们去测试。用户会为你的问题，如：程序能产生预期的结果吗？用起来容易并且有趣吗？符合标准习惯吗？再有，将你的整个应用软件交给某个懂得一些使用该种应用软件的人，请他使用并试图打破它。

我们来看看另外一个程序例子，下面显示的过程OpenToRead示范了Resume Next和Error语句的使用，以及Err对象。

```

Sub OpenToRead()
    Dim myFile As String
    Dim myChar As String
    Dim myText As String
    Dim FileExists As Boolean

    FileExists = True

    On Error GoTo ErrorHandler

    myFile = InputBox("Enter the name of file you want to open:")

    Open myFile For Input As #1
    If FileExists Then
        Do While Not EOF(1) ' loop until the end of file 遍历文件
            myChar = Input(1, #1) ' get one character 获取一个字符
        Loop
    End If
End Sub

```



```

    myText = myText + myChar ' store in the variable myText 存储至变量myText
Loop

Debug.Print myText ' print to the Immediate window 打印到立即窗口
' Close the file -commenting out this instruction will cause
' error 52. 关闭文件 - 注释掉该指令 (Close #1) 会导致错误52

Close #1
End If

Exit Sub

ErrorHandler:
FileExists = False
Select Case Err.Number
Case 71
    MsgBox "The diskette drive is empty."
Case 53
    MsgBox "This file can't be found on the specified drive."
Case 75
    Exit Sub
Case Else
    MsgBox "Error " & Err.Number & " : " & Error(Err.Number)
    Exit Sub
End Select

Resume Next
End Sub

```

过程OpenToRead的目的是一字节一字节地读取用户提供的文本文件内容（操作文件在第八章里）。当用户输入了一个文件名，各种各样的错误可能发生。例如，文件名可能是错误的，或者用户可能试图从软盘上打开文件，而这时软驱里并没有软盘，或者试图打开一个已经打开了的文件。

要捕捉这些错误，过程OpenToRead结尾处的错误处理程序使用了Err对象的Number（原文为Name）属性。Err对象包含有关运行时间错误的信息。如果程序运行时错误发生了，Err.Number语句就会返回错误编号。

如果错误71，53或者75发生了，VB就会显示写在Select...Case代码块里的友好信息并且进行到语句Resume Next，它会将VB发送到导致错误的代码行下面的一行。如果是其它（意想不到）的错误发生了，那么VB就会返回错误编号（Err.Number）和错误描述（Error(Err.Number)）

在过程的开始处，变量FileExists被设置为真，这样，如果该程序没遇到错误的话，所有在If FileExists Then代码块里的指令就会被执行。然而，如果VBA遇到了错误，那么变量FileExists的值就会被设置为假（参见标签ErrorHandler下面的错误处理程序的第一行语句）。这样，VB在试图读取文件时就不会产生错误，导致打开错误。如果你注释掉语句Close #1的话，那么VB在下次试图打开同一文件时，就会遭遇错误。

注意ErrorHandler之前的语句Exit Sub。将Exit Sub语句放在错误处理程序的上面，你不会希望如果没有错误发生的时候还执行该错误处理程序。

我们来进行下述练习，测试过程OpenToRead并更好理解错误捕捉：

1. 用记事本准备一个名叫C:\Vacation.txt文本文件，输入任何文本
2. 逐语句执行过程OpenToRead四次，每次提供下述之一的信息：
 - C:\Vacation.txt文件名称
 - 不存在C:盘上的文件名
 - A:盘上的任意文件，但是软驱是空的
 - 注释掉语句Close #1，并且输入文件名C:\Vacation.txt

技巧13-7 错误：制造错误—测试错误处理程序

你可以故意制造一些错误来测试你程序里的错误陷阱：

- 通过使用下述语法设置内置错误：Error error_number。例如，要显示当除数为0时发生的错误的话，可以在立即窗口里输入：

Error 11

当你按下回车键后，VB就会显示错误信息：

运行时间错误“11”

除数为零

- 要检查产生错误的意义的话，可以使用语法：**Error(error_number)**。例如，想要知道编号为7的错误是什么意思，可以在立即窗口里输入下述指令：

?Error(7)

当你回车后，VB会返回该错误描述：

内存溢出

17.接下来……

在本章，你学习了如何测试你的VBA过程，以确保他们按计划进行。你使用断点和监视逐步程序来调试它。你学习了如何在中断模式下使用立即窗口。你知道了本地窗口如何能帮你检测变量值，以及调用堆栈对话框如何能在你复杂的程序里帮你追踪你在哪里。你已经学习了在编译时确定哪些需要包括哪些需要排除。最后，你也学习了如何使用错误处理程序捕捉错误。通过使用内置的调试工具，你可以快速指出程序的问题所在。请试着多花一些时间来熟悉这些工具，掌握了调试艺术，可以节省你许多时间并避免错误。

通过完成一到十三章，你已经获得了扎扎实实的VBA工作知识，很可能，你应该开始自己的Excel自动化工程了。本章结束了你使用Excel 2002 VBA的中级级别，VBA提供了许多更高级的功能，这些在本书的剩余章节将挖掘出来。

第十四章 微软 Excel 2002 中的事件编程

你如何使当用户点击工作表单元格时出现的内置快捷菜单失活？你如何在工作簿打开或者关闭之前显示一个自定义信息？你如何验证输入在单元格或者单元格区域里的数据？要想对Excel获得彻底控制的话，你必须学习如何响应事件。学习如何进行事件编程将让你贯彻你自己的功能性到Excel应用软件里去。你需要学习有关该主题的第一件事情就是，什么是事件。这里有个简单的定义：

事件是发生的事情 无需说，对象发生的事件是Excel的一部分，然而，一旦你学习了Excel中的事件知识，你将发现更容易去理解发生在Word或者其它任何微软办公软件的对象事件。事件是由对象认可的行动。既然你知道了什么是事件，那么你需要知道事件是可以被一个应用软件用户（例如你自己），另一个程序或者系统本身引发的。因此，你如何能够引发事件呢？假设你右键单击一个工作表单元格，该具体操作将显示一个内置的工作表单元格快捷菜单，允许你快速的访问和工作表单元格相关的频繁使用的命令。但是，万一在某种情况下该内置响应不对呢？你可能想要完全不接受工作表的右键单击，或者当用户右键单击任何单元格时，单元格快捷菜单上出现一个自定义菜单。有个好消息，就是你可以使用VBA来编写代码对事件进行反应。

Excel提供了许多事件供你响应，下述对象可以响应事件：

- 工作表
- 图表
- 透视表
- 工作簿
- 应用软件 通过编写时间过程，你可以决定当事件发生时发生什么。

1.事件过程介绍

事件过程 作为一种特殊的VBA过程 用来对特定的事件作出反应 该过程包含处理具体事件的VBA代码 有些事件只需要简单的一行代码，然而，其它的可能更复杂。事件过程拥有名称，按下述方式创建：

对象名称_事件名称() 在事件名称后面的括号里，你可以放置需要发送到过程里的参数。程序员不能更改事件过程名称。在你编写事件过程对Excel事件作出反应之前，你需要知道：

- 想要响应的具体对象和事件的名称
响应事件的对象在代码窗口的过程下拉清单里显示了一系列事件（见图14-1）。同样，你也可以使用对象浏览器找到事件名称（见图14-2）。
- 你需要放置代码的地方 有些事件在标准模块里，其它的在类模块里。然而，工作簿，图表和工作表事件对任何打开的工作

表或者工作簿可用。要 给一个内嵌的图表，透视表或者应用软件对象创建事件过程的话，那么你必须首先使用关键字 With Events在类模块里创建 一个新对象

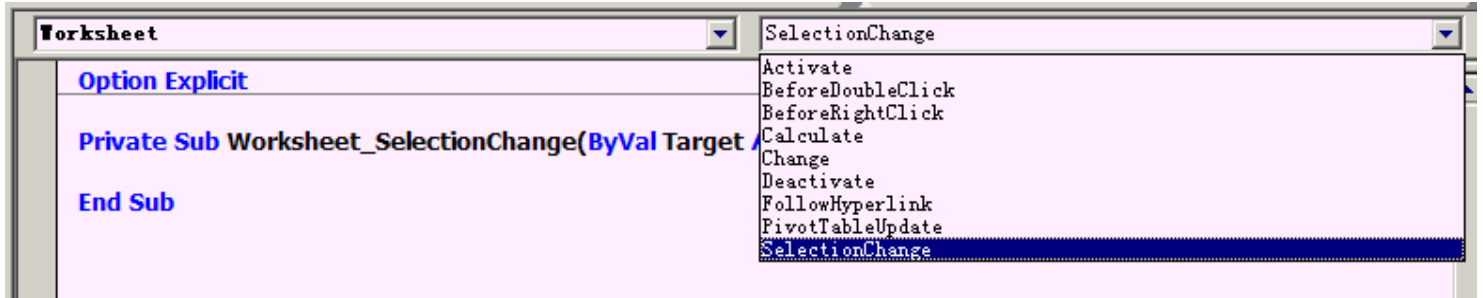


图14-1 你可以在代码窗口找到事件名称

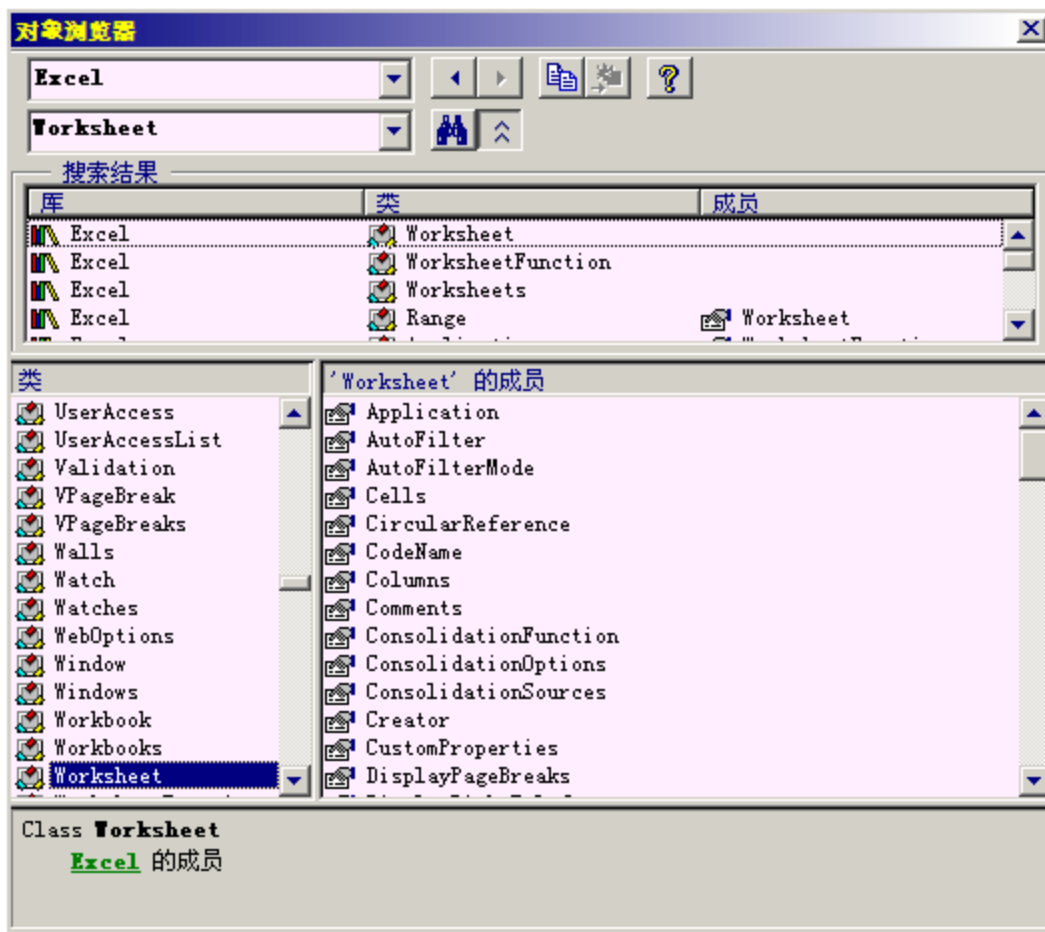


图14-2 你可以在对象浏览器里找到事件名称

2. 激活和失活事件

你可以使用应用软件对象的EnableEvents属性来激活或者失活事件。如果你编写了VBA过程但是不希望有个具体事件发生，那么就将EnableEvents属性设置为False。例如，为了避免在运行过程EnterData（参见下面代码）引发Workbook_BeforeClose事件，那么在调用Workbook对象Close方法之前，设置EnableEvents属性为False。在你的程序结束之前，你得将EnableEvents属性设置回True，确保事件被激活了。

1. 打开一新工作簿并另存为DisableEvents.xls
2. 切换到VB编辑器屏幕，双击工程浏览器窗口的ThisWorkbook，并且在出现的代码窗口里输入Workbook_BeforeSave事件过程

```
Private Sub Workbook_BeforeSave(ByVal SaveAsUI As Boolean, _
Cancel As Boolean)
If MsgBox("Would you like to copy " & vbCrLf _
& "this worksheet to " & vbCrLf _
& "a new workbook?", vbYesNo) = vbYes Then
    Sheets(ActiveSheet.Name).Copy
End If
End Sub
```

3. 选择插入|模块添加一个标准模块激活VBA工程，并且输入下面显示的过程：

```
Sub EnterData()
    With ActiveSheet.Range("A1:B1")
        .Font.Color = vbRed
        .Value = 15
    End With

    Application.EnableEvents = False
    ActiveWorkbook.Save
```

```
Application.EnableEvents = True  
End Sub
```

4. 切换到Excel应用软件窗口，并且选择文件|保存。这时将引发Workbook_BeforeSave事件。点击是响应该信息框。Excel会 打开一个新工作簿，复制当前的工作表内容。

5. 激活DisableEvents工作簿，并且选择工具|宏|宏。在对话框上，点击EnterData然后运行。注意，当你运行EnterData过程时，你没有被提示在保存之前复制工作表。这表明Workbook_BeforeSave事件没有运行。

3.事件次序

事件发生以相应具体的动作并且按预先设计的次序发生。下面的表格示范了打开新工作簿、往工作簿里添加新工作表以及关闭工作簿时事件的顺序。

动作	对象	事件顺序
打开一个新工作簿	Workbook	NewWorkbook, WindowDeactivate, WorkbookDeactivate, WorkbookActivate, WindowActivate
往工作簿添加新工作表	Workbook	WorkbookNewSheet, SheetDeactivate, SheetActivate
关闭工作簿	Workbook	WorkbookBeforeClose, WindowDeactivate, WorkbookDeactivate, WorkbookActivate, WindowActivate

4.工作表事件

工作表对象相应例如工作表激活和失活事件、计算工作表事件、工作表更改事件和双击或右键单击事件。本节讨论一些工作表对象能够响应的事件。

事件名称	激活
事件描述	示例1
当用户激活工作表时，发生该事件	<pre>Dim shtName As String 'declared at the top 在模块上部声明 'of the module Private Sub Worksheet_Activate() shtName = ActiveSheet.Name Range("B2").Select End Sub</pre>
示例程序中，每次该工作表被激活时，选择单元格B2	

示例1— 试验在VB编辑器窗口激活工程浏览器窗口并打开Excel对象文件夹双击Sheet2(Sheet2)并且输入示例程序到Sheet2代码窗口。接下来，切换到Excel窗口并激活Sheet2。注意，当Sheet2被激活时，被选择的总是单元格B2。

事件名称	失活
事件描述	示例2
当用户激活不同的工作表时，发生该事件	<pre>Private Sub Worksheet_Deactivate() MsgBox "You deactivated " & _ shtName & "." & vbCrLf & _ "You switched to " & _ ActiveSheet.Name & "." End Sub</pre>
示例程序中，当Sheet2失活时，显示一个信息框	

示例2— 试验：在VB编辑器窗口，激活工程浏览器窗口并且打开Excel对象文件夹，双击Sheet2 (Sheet2)，然后输入示例程序。接下来，切换到Excel应用程序窗口并且激活Sheet2。你在示例1里创建的Worksheet_Activate过程将会运行，Excel会选中单元格B2并且将工作表名称存储于你在Sheet2的代码模块上面声明的全局变量shtName里面。现在，点击当前工作簿里的其它工作表，

注意，Excel将显示你离开的工作表名称和你切换到的工作表名称。

事件名称	选择变化
事件描述	示例3
当用户选择工作表单元格时，发生该事件	<pre>Private Sub Worksheet_SelectionChange(ByVal Target As Excel.Range) '可省略Excel，下同。 On Error Resume Next Set myRange = Intersect(Range("A1:A10"), Target) If Not myRange Is Nothing Then MsgBox "Data entry or edits are not permitted." End If End Sub</pre>

示例程序中，当用户选择任何myRange中的单元格或区域时，显示一个信息框

示例3 – 试验：在VB编辑器窗口，激活工程浏览器窗口并且打开Excel对象文件夹，双击Sheet3 (Sheet3)，并且在Sheet3代码窗口里输入该示例程序。然后切换到Excel窗口并激活Sheet3。点击给定区域A1:A10中的任何单元格。注意，Excel将显示一个信息。

事件名称	变化
事件描述	示例4
当用户更改单元格内容时，引发该事件	<pre>Private Sub Worksheet_Change(ByVal Target As Excel.Range) Application.EnableEvents = False Target = UCase(Target) Columns(Target.Column).AutoFit Application.EnableEvents = True End Sub</pre>

示例程序中，你输入的内容会自动变为大写，并且该列列宽将自动适应内容文本长短

示例4— 试验：在VB编辑器窗口，激活工程浏览器窗口并且打开Excel对象文件夹，双击Sheet1 (Sheet1)，并且在Sheet1代码窗口里输入该示例程序。然后切换到Excel窗口并激活Sheet1。在任意单元格里输入任何文本，注意，你一旦按下回车键，Excel 就会将该文本变为大写，然后该列自动适应文本长度。

事件名称	计算
事件描述	示例5
当用户重新计算工作表时，引发该事件	<pre>Private Sub Worksheet_Calculate() MsgBox "The worksheet was recalculated." End Sub</pre>

一旦工作表重新计算，示例程序就会弹出一信息

示例5— 试验：在当前工作簿里添加一个新工作表，练习假定Excel会在你的工作簿里放置Sheet4。往Sheet4的单元格A2里输入1，B2里输入2。在单元格C2里输入下述公式：= A2 + B2。在VB编辑器窗口，激活工程浏览器窗口并且打开Excel对象文件夹，双击Sheet4 (Sheet4)，并且输入如上所示Worksheet_Calculate事件程序。切换到Excel窗口并激活Sheet4，在B2单元格里输入任何数字。注意，当退出编辑模式后，Worksheet_Calculate事件就被引发了，你也就看到了一个自定义的信息。

事件名称	双击前
事件描述	示例6
当用户双击工作表单元格时，引发该事件	<pre>Private Sub Worksheet_BeforeDoubleClick(ByVal Target As Range, Cancel As Boolean) If Target.Address = Range("\$C\$9") Then MsgBox "No double-clicking, please." Cancel = True Else MsgBox "You may edit this cell." End If End Sub</pre>

示例程序不允许双击时单元格内部直接编辑C9

示例6— 试验：在VB编辑器窗口，激活工程浏览器窗口并且打开Excel对象文件夹，双击Sheet2 (Sheet2)，并且在Sheet2代码窗口里输入该示例程序。然后切换到Excel窗口并激活Sheet2。当你双击单元格C9时，该事件过程取消了内置的Excel行为，用户不允许进行单元格内部直接编辑。然而，用户可以绕过该限制，点击编辑栏或者按下F2键。当你编写一些事件程序来禁止访问某些功能时，请编写一些额外的代码以禁止一些工作区。（如隐藏编辑栏，隐藏工作表标签等）

事件名称	右键单击前
事件描述	示例7
当用户右键单击工作表单元格时，引发该事件	

```
Private Sub Worksheet_BeforeRightClick(ByVal Target As Range, Cancel As Boolean)
    With Application.CommandBars("Cell")
        .Reset
        If Target.Rows.Count > 1 Or Target.Columns.Count > 1 Then
            With .Controls.Add(Type:=msoControlButton,
                                before:=1,
                                temporary:=True)
                .Caption = "Print..."
                .OnAction = "PrintMe"
            End With
        End If
    End With
End Sub
```


```

End With
End Sub

Sub PrintMe()
    Application.Dialogs(xlDialogPrint).Show arg12:=1
End Sub

```

示例程序中，当用户选择区域不止一行一列时，就会在单元格快捷菜单上添加一个Print选项

示例7—试验：在VB编辑器窗口，激活工程浏览器窗口并且打开Excel对象文件夹，双击Sheet2 (Sheet2)，并且在Sheet2代码窗口里输入该示例程序。在当前工程里添加一个新模块，并且输入PrintMe过程，如上所示。当用户从快捷菜单上选择Print选项时，该过程就会被Worksheet_BeforeRightClick事件调用。注意，对话框的Show方法后面带了个名为arg12:=1的参数。该参数让打印对话框显示时，勾选“所选区域”（通常是“所选工作表”）。在适当的模块里输入完两个过程后，切换到Excel窗口并激活Sheet2。在任何单个单元格上单击右键，注意，这时出现的快捷菜单为缺省选项。现在，我们重新选择，这次包括多个单元格，并且在所选区域上单击右键，你将看到Print...选项出现在第一位置。点击Print选项，注意，打印对话框显示的是所选区域，而不是缺省所选工作表。

注意：参见第十章中使用快捷菜单的更多信息，还有，参见图10-3，如何定位Excel的内置参数列表。

事件名称	跟踪链接
事件描述	示例8
当用户点击Excel工作表中的链接时，引发该事件	<pre> Private Sub Worksheet_FollowHyperlink(ByVal Target As Hyperlink) Target.AddToFavorites End Sub </pre>
示例程序将用户点击的链接添加到IE的收藏夹里	

示例8—试验：在VB编辑器窗口，激活工程浏览器窗口并且打开Excel对象文件夹，双击Sheet1 (Sheet1)，并且在Sheet1代码窗口里输入该示例程序。切换到Excel窗口，并在任何单元格里输入一网址，比如www.wordware.com然后回车。现在，点击该链接激活该网址，当IE窗口出现后，打开收藏菜单，就会发现Wordware网址已经被添加到该菜单里了。

事件名称	更新数据透视表
事件描述	示例9
工作表中的数据透视表被更新后就会发生该事件。它是Excel 2002的新事件。参数Target明确所选地数据透视表。pivTbl是个变量，指向类模块里使用 WithEvents 关键字声明的工作表的对象	<pre> Private Sub pivTbl_PivotTableUpdate(ByVal Target As PivotTable) MsgBox Target.Name & " report has been updated." & vbCrLf & "The PivotReport is located in cells " & Target.DataBodyRange.Address End Sub </pre>
示例程序显示一信息，描述被更新的数据透视表的名称和发生在工作表上报告的单元格区域地址。	

示例9—试验：打开位于本书附带光盘上的文件PivotReport_2.xls，点击数据透视表上的任意单元格，并且点击数据透视表工具栏上的更新数据按钮。图14-3和14-4示范了如何创建PivotTableUpdate事件句柄。

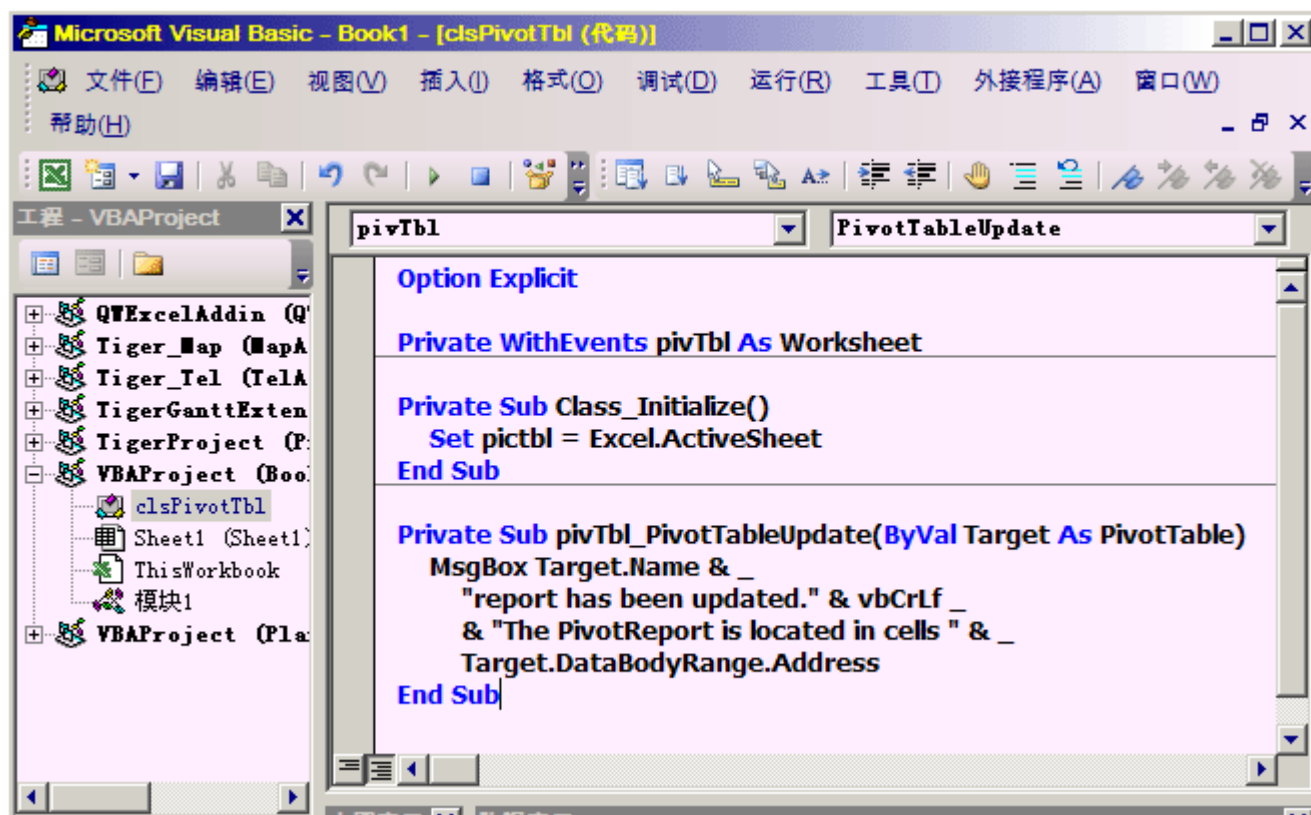


图14-3 你必须使用类模块来捕捉PivotTableUpdate事件。该类模块可以是任何有效的模块名称。对象变量名称pivTbl,也可以是任何有效的变量名称

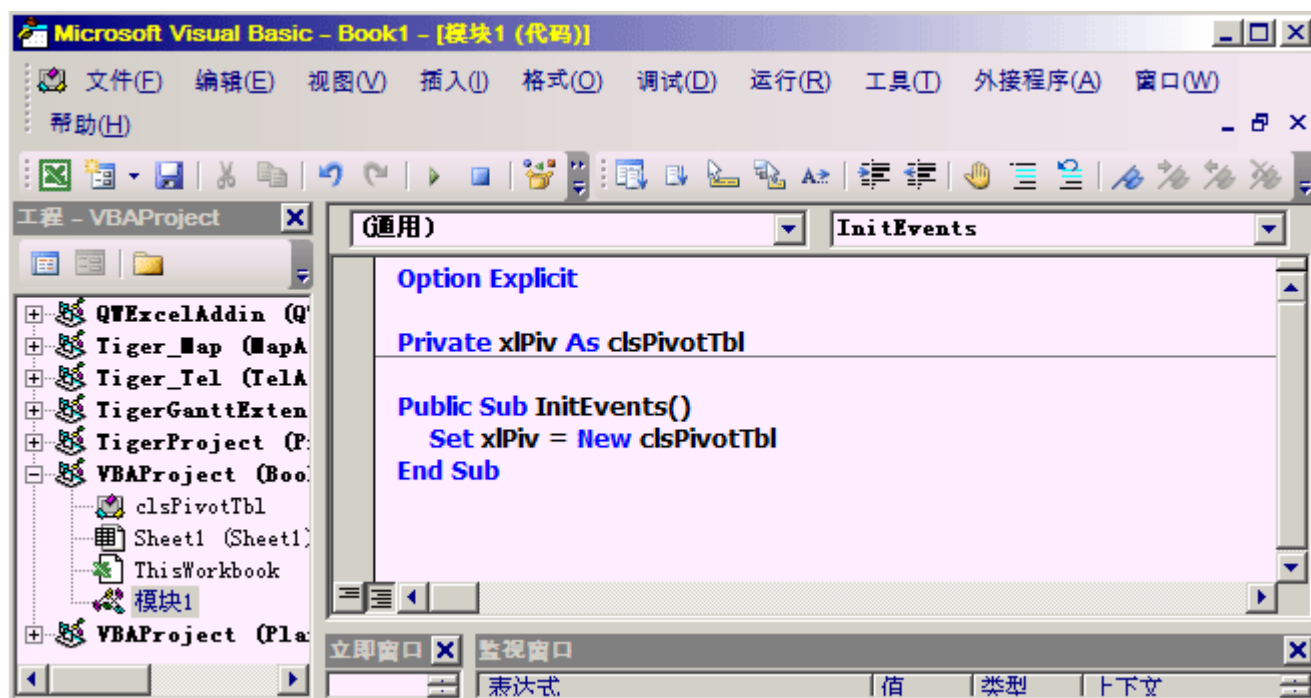


图14-4 你在能够捕捉PivotTableUpdate事件之前,必须在标准模块里设立类模块的示例,并且将对象Worksheet赋予新对象的属性pivTbl。

5.工作簿事件

当用户执行一些操作,如打开,激活,打印,保存以及关闭工作簿时,发生工作簿对象事件。工作簿事件不会创建在标准VBA模

块里，想要编写响应工作簿事件的代码的话，那么需要双击VB编辑器里的工程浏览器中的**ThisWorkbook**。在出现的代码窗口，打开对象下拉清单并且选择**Workbook**对象。在过程下拉清单里选择你想要地事件。被选择的事件过程将出现在代码窗口。例如：

```
Private Sub Workbook_Open()  
    在此放置你的事件处理代码
```

End Sub

本节描述工作簿一些可用的事件。

事件名称	激活
事件描述	示例10
当用户激活该工作簿时，引发该事件。当用户从其它应用程序切换到Excel工作簿时，不会引发该事件。	<pre>Private Sub Workbook_Activate() MsgBox "This workbook contains " & _ ThisWorkbook.Sheets.Count & "sheets." End Sub</pre>

当用户激活包含Workbook_Activate事件过程的工作簿时，示例程序显示该工作簿含有的工作表数目。

示例10 – 试验：在VB编辑器窗口，激活工程浏览器窗口并且打开Excel对象文件夹，双击ThisWorkbook并且在其代码窗口里输入示例过程。然后，切换到Excel窗口并且打开一个新工作簿。切换到你输入了Workbook_Activate过程的工作簿，这时，Excel将显示该工作簿里的工作表总数。

事件名称	失活
事件描述	示例11
当用户激活Excel的其它工作簿时，引发该事件。当用户切换到其它应用软件时，不会发生该事件。	<pre>Private Sub Workbook_Deactivate() For Each cell In _ ActiveSheet.UsedRange If Not IsEmpty(cell) Then Debug.Print cell.Address & _ ":" & cell.Value End If Next End Sub</pre>

当用户激活其它工作簿时，示例程序将在立即窗口里打印当前工作簿里有输入的单元格地址和数值。

示例11 – 试验：在VB编辑器窗口，激活工程浏览器窗口并且打开Excel对象文件夹。双击ThisWorkbook，然后输入示例程序于代码窗口。切换到Excel窗口，并且在活动工作表里输入一些东西。然后，激活某个不同的工作簿。该动作将激发Workbook_Deactivate事件过程。切换到VB编辑器窗口，并打开立即窗口察看什么单元格输入有了报告。

事件名称	打开
事件描述	示例12
当用户打开工作簿时引发该事件。	<pre>Private Sub Workbook_Open() ActiveSheet.Range("A1").Value = Format _ (Now(), "mm/dd/yyyy") End Sub</pre>

当工作簿被打开时，示例过程在单元格A1里放置当前日期。

示例12 – 试验：打开一个新工作簿，在VB编辑器窗口，激活工程浏览器窗口，并且打开Excel对象文件夹。双击ThisWorkbook，并输入示例过程。保存并且关闭该工作簿。当你再次打开该工作簿时，当前日期就会被放置在当前活动工作表地单元格A1里。

事件名称 保存前

事件描述 示例13

该事件发生在工作簿被保存之前。

参数SaveAsUI是只读的，指向SaveAs对话框。如果该工作簿并没有被保存过，那么SaveAsUI参数的值就是True，否则为False。


```
Private Sub
Workbook_BeforeSave(ByVal _
SaveAsUI As Boolean, Cancel As
Boolean)
If SaveAsUI = True And _
ThisWorkbook.Path =
vbNullString Then
MsgBox "This document has
not yet " _
& "been saved." & vbCrLf _
& "The Save As dialog box
will be displayed."
ElseIf SaveAsUI = True
Then MsgBox "You are not
allowed to use " _
& "the SaveAs option. "
Cancel = True
End If
End Sub
```


如果该工作簿没有被保存过，那么示例程序将显示另存为对话框。如果该文件没有被保存过，那么该工作簿的路径名将为字符串**NULL**（**vbNullString**）。过程不允许用户将该工作簿保存为一个不同的名称；另存为操作将通过设置参数**Cancel**为**True**而中断。用户将需要选择保存选项来保存该工作簿。

示例13 – 试验：打开一个新工作簿，在VB编辑器窗口，激活工程浏览器窗口并打开**Excel**对象文件夹。双击**ThisWorkbook**并且 在该代码窗口里输入示例程序。点击**“If SaveAsUI...”** 语句旁边的页边以放置一个断点。切换到**Excel**窗口，并且在任意单元格 输入一些数据。点击工具栏上保存按钮。Workbook_BeforeSave事件过程将会被激活。执行**If SaveAsUI**后面的语句。在另存为 对话框里输入**SaveEvent.xls**作为该工作簿的名称。在保存（并命名）该工作簿之后，对该工作簿作一些更改，然后选择文件|另 存为。这次**Elseif**子句将会被执行，并且你不会被允许通过使用**SaveAs**选项来保存该工作簿。

事件名称	打印前
事件描述	示例8
该工作簿发生工作簿打印之前，以及打印对话框出现之前	<pre>Private Sub Workbook_BeforePrint(Cancel _ As Boolean) Dim response As Integer response = MsgBox("Do you want to " & vbCrLf & _ "print the workbook's full name in the footer?", _ vbYesNo) If response = vbYes Then ActiveSheet.PageSetup.LeftFooter = _ ThisWorkbook.FullName Else ActiveSheet.PageSetup.LeftFooter = "" End If End Sub</pre>

在打印之前，如果用户点击了信息框的“是”，那么示例程序就会将工作簿的完整名称放入文件的脚注里

示例14 – 试验：打开一个新工作簿，在VB编辑器窗口，激活工程浏览器窗口并打开**Excel**对象文件夹。双击**ThisWorkbook**并且 在该代码窗口里输入示例程序。然后，切换到**Excel**窗口，并激活任意工作表。在任意单元格里输入你想要输入的任意内容。当 你按下工具栏上的打印预览按钮时，**Excel**将会询问你是否想要将工作簿名称和路径放入到脚注里。

事件名称	关闭前
事件描述	示例8
该事件发生在工作簿关闭之前，并且在系统询问用户是否保存变化之前	<pre>Private Sub Workbook_BeforeClose(Cancel _ As Boolean) If MsgBox("Do you want to change " & vbCrLf _ & " workbook properties before closing?", _ vbYesNo) = vbYes Then Application.Dialogs(xlDialogProperties).Show End If End Sub</pre>

如果用户对信息框响应“是”时，示例程序将显示属性对话框

示例15 – 试验：在VB编辑器窗口，激活工程浏览器窗口并打开**Excel**对象文件夹。双击**ThisWorkbook**并且在该代码窗口里输入 示例程序。然后，切换到**Excel**窗口，并关闭包含 **BeforeClose**事件程序的工作簿。在关闭之前，你将看到一个信息框，询问你是 否要更改文件属性。在察看和修改工作簿属性之后，该过程关闭该工作簿。如果有些改变你还没有保存，那么你还有机会去保存 该工作簿，取消该更改或者干脆终止该关闭操作。

事件名称	安装加载宏
事件描述	示例8
当用户安装该工作簿为一个加载宏时，引发该事件	<pre>Private Sub Workbook_AddinInstall() MsgBox "To create a calendar, " & vbCrLf _ & "enter CalendarMaker in the " & vbCrLf _ & "Macros dialog box." End Sub</pre>

请参考下述详细指导来引发Workbook_AddinInstall事件过程

示例16－ 试验：

1. 打开一个新工作簿
2. 切换到VB编辑器，激活工程浏览器窗口，并打开Excel对象文件夹

- 3. 双击 **ThisWorkbook** 并且输入示例16和示例17程序在 **ThisWorkbook** 代码窗口
- 4. 在当前VBA工程插入一个新模块，并且输入过程 **CalendarMaker**，显示于示例17之后
- 5. 切换到 **Excel** 窗口 并选择 **文件|属性** 在属性对话框里输入下述内容 标题 **Calendar Maker** 备注 **Create a monthly calendar in an Excel spreadsheet**。当你加亮该加载宏名称时，上面的信息将出现在加载宏对话框里面
- 6. 点击确定以退出属性对话框
- 7. 选择 **文件|另存为** 并且将该工作簿保存为 **Calendar.xls**
- 8. 现在，选择 **文件|另存为**，将 **Calendar.xls** 保存为一个加载宏。从另存为类型下拉清单，选择 **Microsoft Excel 加载宏**。输入一个名称（**CalendarMaker.xla**）然后点击保存
- 9. 关闭 **Calendar.xls** 工作簿
- 10. 打开一个新工作簿
- 11. 选择 **工具|加载宏** 使用浏览按钮将 **CalendarMaker** 加入到加载宏清单里面 勾选列表框里的 **CalendarMaker** 加载宏 当你点击确定后，**Workbook_AddInInstall** 过程就会被引发。点击信息框上的确定
- 12. 想要创建一个日历的话，可以选择 **工具|宏|宏**，在宏名称文本框里输入 **CalendarMaker** 然后点击运行 你将会被询问年和月，输入年和月（例如，**Nov2005**）并点击确定，将出现一日历页，如图14-5所示

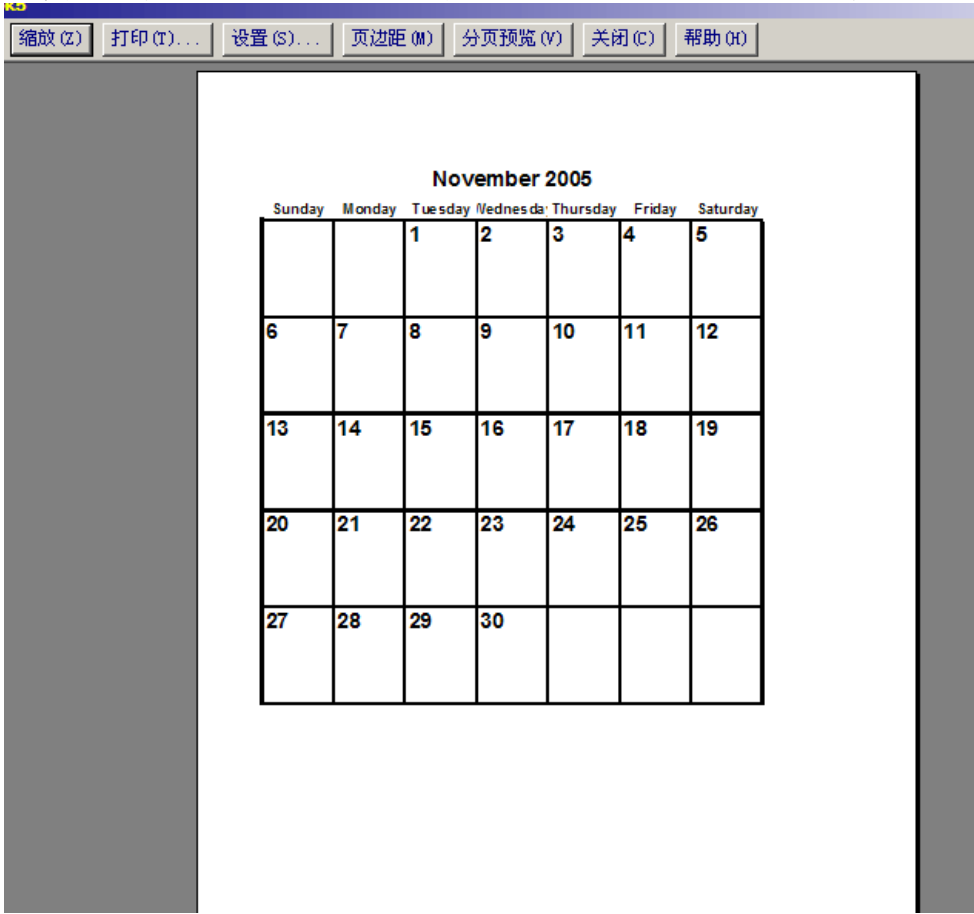


图14-5 由过程CalendarMaker创建的月历

- 13. 想要引发示例17中示范的AddInUninstall事件过程的话，可以选择 **工具|加载宏**，并清除 **CalendarMaker** 加载宏前面的勾选标志就可以了。

事件名称	卸载加载宏
事件描述	示例17
当用户卸载一个加载宏时，引发该事件	<pre>Private Sub Workbook_AddInUninstall() MsgBox "The CalendarMaker " & vbCrLf _ & "add-in was unloaded." End Sub</pre>

一旦该工作簿作为一个加载宏被卸载后，示例程序将显示一个信息

示例17 – 试验：参见示例16

下述过程 **CalendarMaker** 可在 **CodeLibrarian** 加载宏里获得 这里用来示范工作簿对象的安装加载宏和卸载加载宏事件的使用参见示例16和17）。

```

Sub CalendarMaker()
    Dim MyInput As String
    Dim StartDay As Date
    Dim DayOfWeek As Integer, CurYear As Integer, CurMonth As Integer
    Dim FinalDay As Long
    Dim cell As Range
    Dim RowCell As Integer, ColCell As Integer
    Dim x As Integer
    '以上变量声明为译者加上的，以确保程序正常运行

    ' protect sheet if had previous calendar to prevent error. 保护工作表，以避免之前日历带来的错误（原文误为Unprotect）
    ActiveSheet.Protect DrawingObjects:=False, Contents:=False, _
        Scenarios:=False

    ' Prevent screen flashing while drawing calendar. 制作日历时防止屏幕闪烁
    Application.ScreenUpdating = False

    ' Set up error trapping. 设置错误陷阱
    On Error GoTo MyErrorTrap

    ' Clear area a1:g14 including any previous calendar. 清除a1:g14单元格区域的内容
    Range("a1:g14").Clear

    ' Use InputBox to get desired month and year and set variable
    ' MyInput. 使用输入框从用户获得年月，并赋予变量MyInput
    MyInput = InputBox("Type in Month and year for Calendar ")

    ' Allow user to end macro with Cancel in InputBox. 允许用户点击取消以终止宏
    If MyInput = "" Then Exit Sub

    ' Get the date value of the beginning of input month. 获取输入年月的日期序号
    StartDay = DateValue(MyInput)

    ' Check if valid date but not the first of the month 检查是否是有效日期，但不是当月第一天
    ' -- if so, reset StartDay to first day of month. 如不是，则设置StartDay为当月第一天
    If Day(StartDay) <> 1 Then
        StartDay = DateValue(Month(StartDay) & "/1/" & _
            Year(StartDay))
    End If

    ' Prepare cell for Month and Year as fully spelled out. 准备年月的单元格，年月为完整拼写
    Range("a1").NumberFormat = "mmmm yyyy"

    ' Center the Month and Year label across a1:g1 with appropriate
    ' size, height and bolding. 将年月于A1:G1区域跨列居中显示，并设置字号，粗体何行高
    With Range("a1:g1")
        .HorizontalAlignment = xlCenterAcrossSelection
        .VerticalAlignment = xlCenter
        .Font.Size = 18
        .Font.Bold = True
        .RowHeight = 35
    End With

    ' Prepare a2:g2 for day of week labels with centering, size,
    ' height and bolding. 设置A2:G2区域为星期标志，设置为居中，大小，行高和粗体
    With Range("a2:g2")
        .ColumnWidth = 11
        .VerticalAlignment = xlCenter

```



```
.HorizontalAlignment = xlCenter  
.VerticalAlignment = xlCenter  
.Orientation = xlHorizontal
```

```

.Font.Size = 12
.Font.Bold = True
.RowHeight = 20
End With

```

```

' Put days of week in a2:g2. 输入星期
Range("a2") = "Sunday"
Range("b2") = "Monday"
Range("c2") = "Tuesday"
Range("d2") = "Wednesday"
Range("e2") = "Thursday"
Range("f2") = "Friday"
Range("g2") = "Saturday"

```

```

' Prepare a3:g7 for dates with left/top alignment, size, height
' and bolding. 设置A3:G7区域为左对齐和上对齐, 大小, 行高和粗体
With Range("a3:g8")
    .HorizontalAlignment = xlLeft
    .VerticalAlignment = xlTop
    .Font.Size = 18
    .Font.Bold = True
    .RowHeight = 21
End With

```

```

' Put input month and year fully spelling out into "a1". 在单元格A1里输入年月
Range("a1").Value = Application.Text(MyInput, "mmmm yyyy")

```

```

' Set variable and get which day of the week the month starts. 设置变量, 并获取该月第一天的星期
DayOfWeek = Weekday(StartDate)

```

```

' Set variables to identify the year and month as separate
' variables. 设置变量分别获取年和月
CurYear = Year(StartDate)
CurMonth = Month(StartDate)

```

```

' Set variable and calculate the first day of the next month. 设置变量并计算下个月地第一天
FinalDay = DateSerial(CurYear, CurMonth + 1, 1)

```

```

' Place a "1" in cell position of the first day of the chosen
' month based on DayOfWeek. 基于星期序号, 在选定月份第一天的位置放置 "1"
Select Case DayOfWeek

```

```

    Case 1
        Range("a3").Value = 1
    Case 2
        Range("b3").Value = 1
    Case 3
        Range("c3").Value = 1
    Case 4
        Range("d3").Value = 1
    Case 5
        Range("e3").Value = 1
    Case 6
        Range("f3").Value = 1
    Case 7
        Range("g3").Value = 1
End Select

```

```

' Loop through range a3:g8 incrementing each cell after the "1"
' cell. 在单元格区域A3:G8里面循环, 在1之后, 每个单元格增加1

```

```
For Each cell In Range("a3:g8")  
    RowCell = cell.Row  
    ColCell = cell.Column
```

```

' Do if "1" is in first column. 如果1在第一列
If cell.Column = 1 And cell.Row = 3 Then

' Do if current cell is not in 1st column. 如果当前单元格并非第一列
Elseif cell.Column <> 1 Then
    If cell.Offset(0, -1).Value >= 1 Then
        cell.Value = cell.Offset(0, -1).Value + 1

        ' Stop when the last day of the month has been
        ' entered. 当遇到当月的最后一天时，停止
        If cell.Value > (FinalDay - StartDay) Then
            cell.Value = ""

            ' Exit loop when calendar has correct number of
            ' days shown. 当日历显示了正确的数字时，退出循环
            Exit For
        End If
    End If
End If

' Do only if current cell is not in Row 3 and is in Column 1. 仅当当前单元格不在第三行，而在第一列时
Elseif cell.Row > 3 And cell.Column = 1 Then
    cell.Value = cell.Offset(-1, 6).Value + 1

    ' Stop when the last day of the month has been entered. 当遇到当月的最后一天时，停止
    If cell.Value > (FinalDay - StartDay) Then
        cell.Value = ""

        ' Exit loop when calendar has correct number of days
        ' shown. 当日历显示了正确的数字时，退出循环
        Exit For
    End If
End If
Next

' Create Entry cells, format them centered, wrap text, and border
' around days. 创建输入单元格，居中，自动换行，并在每日周围设置边框
For x = 0 To 5
    Range("A4").Offset(x * 2, 0).EntireRow.Insert
    With Range("A4:G4").Offset(x * 2, 0)
        .RowHeight = 65
        .HorizontalAlignment = xlCenter
        .VerticalAlignment = xlTop
        .WrapText = True
        .Font.Size = 10
        .Font.Bold = False

        ' Unlock these cells to be able to enter text later after
        ' sheet is protected. 解开这些区域的锁定，以供将来输入文本
        .Locked = False
    End With

    ' Put border around the block of dates. 在每日周围设置边框
    With Range("A3").Offset(x * 2, 0).Resize(2, _
        7).Borders(xlLeft)
        .Weight = xlThick
        .ColorIndex = xlAutomatic
    End With
    With Range("A3").Offset(x * 2, 0).Resize(2, _
        7).Borders(xlRight)

```

```
.Weight = xlThick  
.ColorIndex = xlAutomatic
```

End With

Range("A3").Offset(x * 2, 0).Resize(2, 7).BorderAround _
Weight:=xlThick, ColorIndex:=xlAutomatic

Next

If Range("A13").Value = "" Then Range("A13").Offset(0, 0) _
.Resize(2, 8).EntireRow.Delete

' Turn off gridlines. 关闭网格线
ActiveWindow.DisplayGridlines = False

' Protect sheet to prevent overwriting the dates. 保护工作表，防止覆盖日期
ActiveSheet.Protect DrawingObjects:=True, Contents:=True, _
Scenarios:=True

' Resize window to show all of calendar (may have to be adjusted
' for video configuration). 设置窗口大小以显示完整日历
ActiveWindow.WindowState = xlMaximized
ActiveWindow.ScrollRow = 1

' Allow screen to redraw with calendar showing. 允许屏幕重绘日历显示
Application.ScreenUpdating = True

' Prevent going to error trap unless error found by exiting Sub
' here. 放置没有错误时也允许错误陷阱
Exit Sub

' Error causes msgbox to indicate the problem, provides new input box, 错误导致信息框指明问题，提供新的输入框
' and resumes at the line that caused the error. 并恢复至导致错误的代码行

MyErrorTrap:

MsgBox "You may not have entered your Month and Year correctly." _
& Chr(13) & "Spell the Month correctly" _
& " (or use 3 letter abbreviation)" _
& Chr(13) & "and 4 digits for the Year"
MyInput = InputBox("Type in Month and year for Calendar")
If MyInput = "" Then Exit Sub
Resume

End Sub

事件名称	新建工作表
事件描述 当用户新建一个工作表后，引发该事件	<p>示例18</p> <pre>Private Sub Workbook_NewSheet(ByVal Sh As Object) If MsgBox("Do you want to place " & vbCrLf _ & "the new sheet at the beginning " & vbCrLf _ & "of the workbook?", vbYesNo) = vbYes Then Sh.Move before:=ThisWorkbook.Sheets(1) Else Sh.Move After:=ThisWorkbook.Sheets(_ ThisWorkbook.Sheets.Count) MsgBox Sh.Name & _ " is now the last sheet in the workbook." End If End Sub</pre>

当用户在弹出的信息框上点击是的话，示例程序就会将新建的工作表置于工作簿的开始，否则在结尾

示例18 – 试验：打开一个新工作簿，在VB编辑器窗口，激活工程浏览器窗口并且打开Excel对象文件夹。双击ThisWorkbook并且

输入示例程序于**ThisWorkbook**代码窗口 切换到**Excel**窗口 并且在工作表标签的任意地方单击右键 从快捷菜单上选择插入， 选择你想要插入的工作表类型，并点击确定。**Excel**将会问你将新建的工作表放置到什么地方。

下面是一个Excel工作表可以响应的事件列表。

事件名称	描述
SheetActivate	当用户激活当前工作簿里的任何工作表发生该事件 SheetActivate事件同样也可以发生在应用软件级别，当你激活任意打开的工作簿里的任意工作表时，都会激活该事件
SheetDeactivate	当用户激活工作簿里一个不同的工作表时，引发该事件
SheetSelectionChange	当用户选择不同单元格时，引发该事件。该事件发生在该工作簿里的每个工作表上
SheetChange	本事件发生在用户更改单元格内容时
SheetCalculate	本事件发生在用户重新计算工作表的时候
SheetBeforeDoubleClick	当用户双击工作表上的单元格时，引发该事件
SheetBeforeRightClick	当用户在工作表单元格上单击右键时，引发该事件

事件名称	激活窗口
事件描述	示例19
当用户激活任何显示该工作簿的窗口时，引发该事件	<pre>Private Sub Workbook_WindowActivate(ByVal Wn As Window) Wn.GridlineColor = vbYellow End Sub</pre>
当用户激活包含Workbook_WindowActivate程序的工作簿时，示例程序将更改工作表的网格线颜色为黄色	

示例19 – 试验：在VB编辑器窗口，激活工程浏览器窗口并打开Excel对象文件夹。双击ThisWorkbook，并输入示例程序。切换到Excel窗口，打开一个全新的工作簿。使用窗口菜单将Excel工作簿设置为竖直并排排列。当你激活含有示例程序的工作簿时，网格线应该会变为黄色了。

事件名称	窗口失活
事件描述	示例20
当用户将焦点从该工作簿窗口移走时，引发该事件	<pre>Private Sub Workbook_WindowDeactivate(ByVal Wn As Window) Wn.GridlineColor = vbRed End Sub</pre>
当用户从包含Workbook_WindowActivate过程代码的工作簿切换到另外一个工作簿时，示例程序将其网格线设置为红色	

示例20 – 试验：在VB编辑器窗口，激活工程浏览器窗口并打开Excel文件夹，双击ThisWorkbook并输入示例程序。切换到Excel窗口，打开一个全新的工作簿。使用窗口菜单将所有Excel工作簿设置为竖直排列。当你失活包含Workbook_WindowDeactivate事件代码的工作簿并切换到空工作簿时，被失活地工作表里的网格线就会被变为红色。

事件名称	窗口调整大小
事件描述	示例21
当用户打开窗口，调整窗口大小，最大化或者最小化窗口时，引发该事件	<pre>Private Sub Workbook_WindowResize(ByVal Wn As Window) If Wn.WindowState <> xlMaximized Then Wn.Left = 0 Wn.Top = 0 End If End Sub</pre>

当用户调整窗口大小时，示例程序将工作簿窗口移至屏幕的左上角

示例21 – 试验：在VB编辑器窗口，激活工程浏览器窗口并且打开Excel对象文件夹。双击ThisWorkbook并输入示例程序。切换到Excel窗口并且点击还原按钮（工作簿，非Excel应用软件）。通过拖曳窗口内部边框改变当前活动窗口大小时，一旦你完成大小调整操作，该工作簿窗口会自动跳到屏幕的左上角（Excel应用软件窗口的左上角）。

下述表格描述了Excel 2002里增加的工作簿事件。

事件名称	描述
PivotTableOpenConnection	当数据透视表报告打开对其数据源的连接时发生该事件。该事件要求你在类模块里面使用WithEvents关键字声明一个Application或者

	Workbook类型对象。(参见“图表事件”和“应用程序对象识别的事件”部分有关该关键字使用的示例)
PivotTableCloseConnection	当数据透视表报告关闭对其数据源的连接后发生该事件。该事件要求你在类模块里面使用WithEvents关键字声明一个Application或者Workbook类型对象。(参见“图表事件”和“应用程序对象识别的事件”部分有关该关键字使用的示例)
SheetPivotTableUpdate 需要下述两个参数： Sh – 被选择的工作表 Target – 被选择的数据透视表报告	当数据透视表报告被更新后发生该事件。该事件要求你在类模块里使用关键字WithEvents声明一个Application或者Workbook的type对象(参见示例 9，需要使用关键字WithEvents设置事件处理的相关信息)。该事件处理可以在附带CD里的PivotReport.xls里找到。 <pre>Private Sub App_SheetPivotTableUpdate(_ ByVal Sh As Object, ByVal Target As PivotTable) MsgBox "Pivot Table has been updated." End Sub</pre>

6.图表事件

众所周知，你可以创建一个内嵌在工作表里的图表，也可以创建一个单独的图表工作表。在本节，你将学习如何控制图表事件，不管你决定了将你的图表放在哪里。在你试验图表事件之前，做以下几件事情：

1. 打开一个新Excel工作簿，并且保存为ChartEvents.xls
2. 输入示例数据，如图14-6所示
3. 选择单元格区域A1:D4，并点击工具栏上的图表按钮
4. 准备一个柱型图，如图14-6所示，并将其内嵌于工作表内
5. 使用相同的数据，创建一个折线图于一个单独的图表工作表（参见图14-7）
6. 将图表工作表名称改为Sales Analysis Chart。

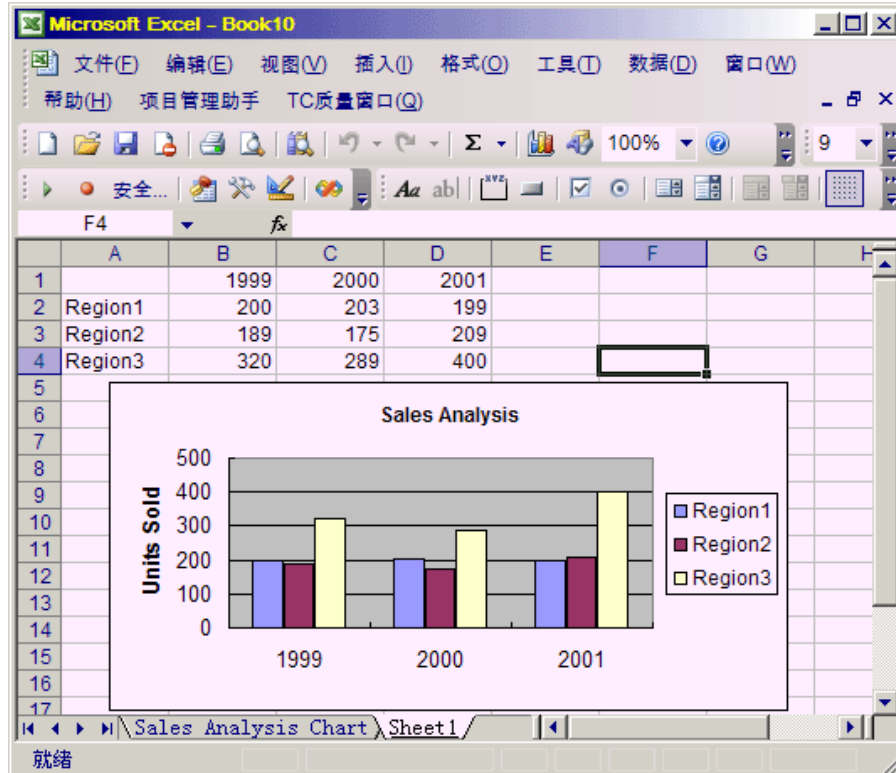


图14-6 内嵌于工作表的柱型图

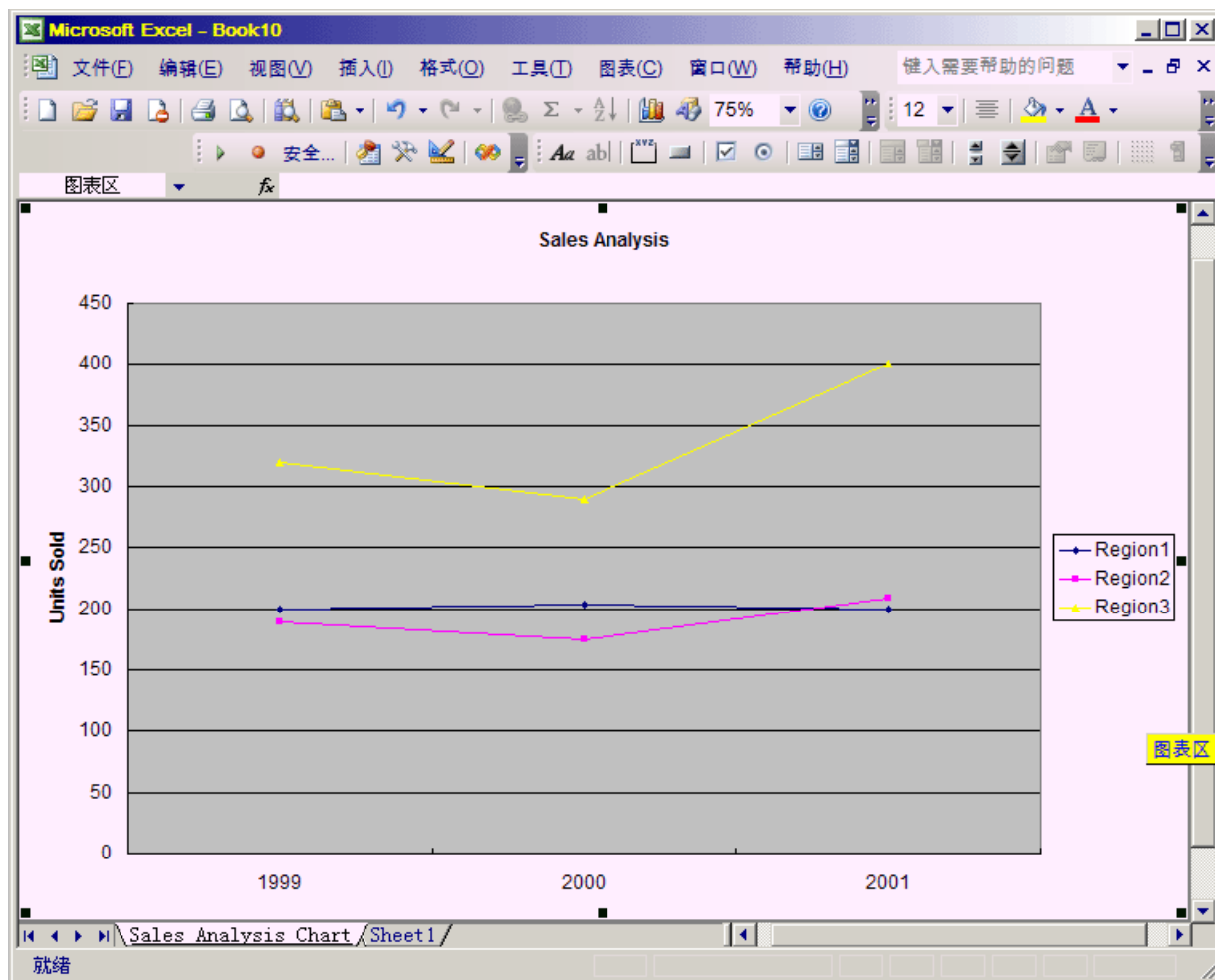


图14-7 置于图表工作表里的折线图

下面的表格列出了图表对象的事件。在该表格里示范的示例程序应该在你刚才放置在图表工作表里的折线图上试验（参见图14-7）。内嵌于工作表里的图表的事件需要特别设置，本章稍候将讲解。

1. 在VB编辑器窗口，激活工程浏览器窗口并打开Excel对象文件夹
2. 双击图表对象Chart1 (Sales Analysis Chart)
3. 在代码窗口，输入事件程序，如下表所示
4. 激活图表工作表，并执行一些将引发事件过程的操作。例如，点击图表标题后，Chart_MouseDown和Chart_Select事件就会被引发。

事件名称	描述
Activate	当用户激活图表工作表时引发该事件 <pre>Private Sub Chart_Activate() MsgBox "You've activated the chart sheet." End Sub</pre>
Deactivate	当用户离开该图表时引发该事件（进入其它工作表） <pre>Private Sub Chart_Deactivate() MsgBox "It looks like you want to leave the chart sheet." End Sub</pre>
Select	当用户选择了某个图表成员时引发该事件 <pre>Private Sub Chart_Select(ByVal ElementID As Long, _ ByVal Arg1 As Long, ByVal Arg2 As Long) If Arg1 <> 0 And Arg2 <> 0 Then</pre>

```
MsgBox ElementID & ", " & Arg1 & ", " & Arg2  
End If  
If ElementID = 4 Then  
MsgBox "You've selected the chart title."
```



```

ElseIf ElementID = 24 Then
    MsgBox "You've selected the chart legend."
ElseIf ElementID = 12 Then
    MsgBox "You've selected the legend key."
ElseIf ElementID = 13 Then
    MsgBox "You've selected the legend entry."
End If
End Sub

```

ElementId返回一个代表所选图表成员的常数。参数Arg1和Arg2用于某些相关的图表成员上。例如，图表坐标轴（ElementId=21），可以明确是主坐标轴（Arg1=0）或者次坐标轴（Arg1=1），而坐标轴的类型则由参数Arg2确定，它可以是下述三种之一：

0 – 分类轴，1 – 数值轴，或者3 – 系列轴

SeriesChange	当用户更改图表数据点时，引发该事件。图表对象应该在类模块里使用关键字WithEvents进行声明。
Calculate	<p>当用户选择新数据或者改变图表数据时，引发该事件。</p> <pre> Private Sub Chart_Calculate() MsgBox "The data in your spreadsheet has " & vbCrLf _ & "changed. Your chart has been updated." End Sub </pre>
Resize	当用户改变图表大小时，引发该事件。该图表对象应该在类模块里面使用WithEvents关键字来声明
DragOver	当用户拖曳数据到该图表时，引发该事件。该图表对象应该在类模块里面使用WithEvents关键字来声明
DragPlot	当用户拖曳单元格区域到图表时，引发该事件。该图表对象应该在类模块里面使用WithEvents关键字来声明
BeforeDoubleClick	当用户双击图表时，引发该事件
BeforeRightClick	<p>当用户右键单击图表时，引发该事件</p> <pre> Private Sub Chart_BeforeRightClick(Cancel As Boolean) Cancel = True End Sub </pre>
当你将参数Cancel设置为True后，MouseDown	<p>你就不能访问图表区域快捷菜单了</p> <p>当光标在图表之上并按下鼠标时，引发该事件</p> <pre> Private Sub Chart_MouseDown(ByVal Button As Long, _ ByVal Shift As Long, ByVal x As Long, ByVal y As _ Long) If Button = 1 Then MsgBox "You pressed the left mouse button. " ElseIf Button = 2 Then MsgBox "You pressed the right mouse button. " Else MsgBox "You pressed the middle mouse button. " End If End Sub </pre>
<p>按键参数决定哪个鼠标键被按下（MouseDown事件），或者释放（MouseUp事件）了：</p> <p>1 – 鼠标左键，2 – 鼠标右键，4 – 鼠标中键</p> <p>Shift参数明确Shift键，Ctrl键和Alt键的状态：</p> <p>1- 选择了Shift键，2 – 选择了Ctrl键，以及4 – 选择了Alt键</p> <p>参数x, y 分别明确鼠标指针坐标</p>	
MouseMove	当鼠标指针坐标在图表之上变化时，引发该事件
MouseUp	当鼠标按键在图表之上释放时，引发该事件

7.内嵌图表事件

想要捕捉工作表内嵌图表的事件的话，那么你首先要在类模块里使用关键字**WithEvents**创建一个新的对象。我们按照下述步骤来看看如何实现它：

1. 激活VB编辑器窗口
2. 在工程浏览器里，选择VBAProject(CharEvents.xls)
3. 选择插入|类模块
4. 在类模块文件夹里，你将看到一个名叫Class1的模块
5. 在属性窗口，将Class1重命名为clsChart
6. 在类模块的代码窗口里，声明一个对象变量，它将代表产生事件的图表对象
Public WithEvents xlChart As Excel.Chart 关键字Public将使对象变量xlChart可用于当前VBA工程里的所有模块。使用WithEvents关键字声明一个对象变量，将暴露 该被定义对象类型的所有事件。输入上述声明之后，对象变量xlChart就会添加到代码窗口左上角地对象下拉列表中去，而 与该对象变量相对应的事件就会出现在代码窗口右上角的过程下拉列表里面。
7. 打开对象下拉列表框并选择变量xlChart名称，现在代码窗口将出现xlChart_Activate过程的构架：

```
Private Sub xlChart_Activate()
```



```
End Sub
```
8. 在该事件过程里添加你的代码。在本练习中，我们将添加一条语句，显示一个信息框。添加完语句后，VBA过程应该像这样：

```
Private Sub xlChart_Activate()
    MsgBox "You've activated a chart embedded in " & _
        ActiveSheet.Name
End Sub
```

 输入完事件过程后，你仍然需要通知VB，你计划使用它。
9. 在工程浏览器窗口，双击ThisWorkbook对象，并且输入下述语句来创建名为clsChart的新示例：
Dim myChart As New clsChart 上面显示的指令声明一个名为myChart的对象变量，该变量将指向位于类模块clsChart里的对象xlChart 关键字New告诉VB 去创建特定对象的新示例。
10. 在ThisWorkbook代码窗口输入下述过程，以初始化对象变量myChart:

```
Sub InitializeChart()
    'connect the class module and its objects with the Chart object
    Set myChart.xlChart = _
        Worksheets(1).ChartObjects(1).Chart
End Sub
```
11. 运行InitializeChart过程。运行该过程之后，输入在类模块里的事件过程就会被相应的事件引发
12. 激活Excel窗口，并且点击内嵌图表。这次，你在第七步输入的xlChart_Activate事件将会被引发
13. 现在，你可以在类模块里给内嵌图表输入其它的事件过程了。

8.可为应用软件对象识别的事件

如果你想要你的事件过程无论在哪个当前活动的Excel工作簿上都能执行的话，那么你需要创建应用软件对象的事件过程。应用软件对象的事件过程具有全局性。这意味着只要Excel应用软件是开启的，那么该过程代码就会响应某个事件被执行。Application对象的事件列在接下来的表格中。和内嵌图表类似，Application对象的事件过程要求你在类模块里使用关键字 WithEvents创建一个新的对象。表格中示范的事件例程序应该在类模块里面输入。如何操作呢？在VB编辑器窗口选择插入|类 模块，在属性窗口重命名类模块为clsApplication。在clsApplication代码窗口声明一个对象变量来代表Application对象，如下所示：

Public WithEvents App As Application 在声明语句下面，输入下述事件过程，如表格所示：App_NewWorkbook，App_WorkbookOpen，App_WorkbookBeforeSave， App_WorkbookBeforeClose，App_Sheet- SelectionChange和App_WindowActivate事件过程。你输入完这些过程在类模块里 之后，插入一个标准模块到当前VBA工程里（选择插入|模块）。在标准模块里，创建一个类clsApplication的新示例，并且将位 于类模块clsApplication里的对象和代表Application对象的对象变量App连接起来，如下所示：

```
Dim DoThis As New clsApplication
```

```
Public Sub InitializeAppEvents ()
```

```
    Set DoThis.App = Application
```

End Sub 现在将鼠标光标置于过程InitializeAppEvents里并且按下F5键运行它。运行过程InitializeAppEvents的结果是类模块的对象App 将会指向Excel应用软件。从现在开始，当某个具体事件发生时，你已经输入在类模块里的事件过程就会被执行。如果你不想相 应Application对象产生的事件的话，你可以通过在一个标准模块里输入下述过程（并运行它）：

```
Public Sub CancelAppEvents()
```

```
    Set DoThis.App = Nothing
```

```
End Sub
```

当你设置对象变量为 **Nothing** 的时候，你实际上释放了内存，并且断开了对象变量和该变量指向的对象之间的连接。当你运行过程 **CancelAppEvents** 后，写在类模块里面的事件过程在某个事件发生就不会自动执行。注意：所有在本表格里示范的事件过程，都需要你在类模块里使用关键字 **WithEvents** 声明个对象变量。

事件名称	描述
NewWorkbook	<p>当用户创建一个新工作簿时引发该事件</p> <pre>Private Sub App_NewWorkbook(ByVal _ Wb As Workbook) Application.DisplayAlerts = False If Wb.Sheets.Count = 3 Then Sheets(Array(2, 3)).Delete End If Application.DisplayAlerts = True End Sub</pre>
WorkbookOpen	<p>该事件发生于用户打开一个工作簿</p> <pre>Private Sub App_WorkbookOpen(ByVal Wb As Workbook) If Wb.FileFormat = xlCSV Then If MsgBox("Do you want to save " & vbCrLf _ & " this file as an Excel workbook?", vbYesNo, _ "Original file format: " _ & "Comma delimited file") = vbYes Then Wb.SaveAs FileFormat:=xlWorkbookNormal End If End If End Sub</pre>
WorkbookActivate	当用户将焦点移到一个开启的工作簿时引发该事件
WorkbookDeactivate	当用户将焦点从一个开启的工作簿移开时引发该事件
WorkbookNewSheet	当用户在一个打开的工作簿上新建一个工作表时引发该事件
WorkbookBeforeSave	<p>该事件发生在以大开工作簿被保护之前</p> <pre>Private Sub App_WorkbookBeforeSave(ByVal _ Wb As Workbook, _ ByVal SaveAsUI As Boolean, _ Cancel As Boolean) If Wb.Path <> vbNullString Then ActiveWindow.Caption = Wb.FullName & _ " [Last Saved: " & Time & "]" End If End Sub</pre>
WorkbookBeforePrint	<p>该事件发生在以大开工作簿被打印之前</p> <pre>Private Sub App_WorkbookBeforePrint(ByVal _ Wb As Workbook, Cancel As Boolean) Wb.PrintOut Copies:=2 End Sub</pre>
WorkbookBeforeClose	<p>该事件发生于关闭工作簿之前</p> <pre>Private Sub App_WorkbookBeforeClose(ByVal _ Wb As Workbook, Cancel As Boolean) Dim r As Integer Sheets.Add r = 1 For Each p In Wb.BuiltinDocumentProperties On Error GoTo ErrorHandler Cells(r, 1).Value = p.Name & " = " & _ ActiveWorkbook. _ BuiltinDocumentProperties _</pre>

```
        .Item(p.Name).Value  
        r = r + 1  
    Next
```

	<pre> Exit Sub ErrorHandler: Cells(r, 1).Value = p.Name Resume Next End Sub </pre>
WorkbookAddInInstall	该事件发生于用户安装加载宏之时
WorkbookAddInUninstall	该事件发生于用户卸载加载宏之时
SheetActivate	当用户激活开启工作簿中某个工作表时引发该事件
SheetDeactivate	当用户离开某个工作表时引发该事件
SheetSelectionChange	<p>当用户改变选择工作表上的区域时引发该事件</p> <pre> Private Sub App_SheetSelectionChange(_ ByVal Sh As Object, ByVal Target As Range) If Selection.Count > 1 Or _ (Selection.Count < 2 And _ IsEmpty(Target.Value)) Then Application.StatusBar = Target.Address Else Application.StatusBar = Target.Address & _ "(" & Target.Value & ")" End If End Sub </pre>
SheetChange	当用户在一个打开的工作簿里改变单元格里内容时引发该事件
SheetCalculate	当用户重新计算某个开启的工作簿里的工作表时引发该事件
SheetBeforeDoubleClick	当用户双击工作表时引发该事件
SheetBeforeRightClick	当用户右键单击工作表单元格时引发该事件
WindowActivate	<p>当用户激活一个打开的窗口时引发该事件</p> <pre> Private Sub App_WindowActivate(ByVal _ Wb As Workbook, ByVal Wn As Window) Wn.DisplayFormulas = True End Sub </pre>
WindowDeactivate	当用户将焦点从开启的窗口移走时引发该事件
WindowResize	当用户调整开启的窗口的大小时引发该事件
WorkbookPivotTableClose- Connection (Excel 2002的新事件)	在数据透视表报告连接被断开后，引发该事件
WorkbookPivotTableOpen- Connection (Excel 2002的新事件)	在数据透视表报告连接被打开后引发该事件

9. 查询表时间

查询表是Excel工作表里代表从外部数据源得来的数据，例如SQL服务器数据库，Access数据库，网页，或者文本文件。查询表用对象QueryTable代表。Excel为QueryTable对象提供了两种事件：BeforeRefresh和AfterRefresh。想要试验一下本章后面示范的这些示例过程的话，那么请执行下面的这些操作。本示例假设你的机器上安装了Access以及其例子Northwind数据库。

1. 在Excel应用软件窗口，选择数据|导入外部数据，并且选择新建数据库查询以创建一个新数据库查询
2. 在数据源对话框里，选择新数据源，并点击确定
3. 在创建新数据源对话框里，输入SampleDb作为数据源名称
4. 在创建新数据源对话框上，从第二步旁边的下拉列表里，选择Microsoft Access driver (*.mdb)
5. 点击连接按钮
6. 在ODBC Microsoft Access安装对话框上点击选择按钮
7. 在选择数据库对话框上，找到文件Northwind.mdb。该文件通常可以在C:\Program Files\Microsoft Office\Office\Samples文件夹找到（译者用的是Office 2003，没有该文件，有一个类似的Nwind.mdb文件。大家可以在电脑上查找一下）

8. 选择该文件并且点击确定以关闭该选择数据库对话框
9. 在点击确定退出ODBC Microsoft Access安装对话框
10. 在创建新数据源对话框的第四步，在下拉列表框里选择Categories
11. 点击确定以关闭创建新数据源对话框

12. 在选择数据源对话框上，数据源名称 **SampleDb** 现在应该被加亮了，点击确定
 13. 在查询向导 – 选择列对话框里，点击 > 按钮，将 **Categories** 表中所有的区域移到查询框的列中去
 14. 点击下一步，直到你看到查询向导 – 完成对话框
 15. 在查询向导 – 完成对话框上，确保将数据返回到 **Microsoft Excel** 选项按钮是被勾选上的，并且点击完成
 16. 在导入数据对话框，当前电子表格单元格是被选中的，点击单元格 **A1** 并点击确定关闭对话框。完成上述步骤后，**Northwind** 数据库里 **Category** 表中的数据应该被放置在当前工作表里面了。重新获得数据是得花费好些步骤的。在下章，你将学习如何编程创建查询表。想要给查询表对象编写事件过程的话，你就必须创建一个类模块并且使用 **WithEvents** 关键字声明一个 **QueryTable** 对象。
1. 插入类模块到当前 VBA 工程并重命名为 **clsQryTbl**
 2. 在 **clsQryTbl** 代码窗口，输入下述语句：
`Public WithEvents qrytbl As QueryTable`
 当你使用 **WithEvents** 关键字声明完新对象 **qrytbl** 后，它就会出现在类模块的对象下拉列表中
 3. 在 **clsQryTbl** 代码窗口，输入两个事件过程，如下面的表格所示：**QryTbl_BeforeRefresh** 和 **QryTbl_AfterRefresh**。在你能够引发这些事件过程之前，你必须将你在类模块里声明的对象和某个特定的 **QueryTable** 对象连接起来
 4. 插入一个标准模块，并输入下述代码：
`Public Sub Auto_Open()
 ' connect the class module and its objects with the Query object
 Set sampleQry.qrytbl = ActiveSheet.QueryTables(1)
End Sub`
 上面的程序创建了一个 **QueryTable** 类 (**clsQryTbl**) 的新示例，并且将它和活动工作表里的第一个查询表连接起来。当你打开该工作簿时，**Auto_Open** 过程会自动执行。因此你不必手动运行它，以确保当数据被刷新时，查询事件将会被引发。
 5. 运行第四步输入 **Auto_Open** 过程，在你运行完该初始化过程后，你在类模块里声明的对象就会指向特定的查询表对象
 6. 在你放置从 **Access** 里导入的 **Category** 的工作表里，更改某个类别。选择查询表中的任意单元格，并且点击外部数据工具栏 上的刷新数据 或者选择数据 | 刷新数据。这次，事件过程 **qrytbl_BeforeRefresh** 将会被引发了，你将看到一个自定义信息框。如果你点击是，该数据将会被数据库里存在的数据刷新掉，你更改过的数据将会被覆盖掉。

事件名称	描述
BeforeRefresh	<p>该事件发生在查询表被刷新数据之前</p> <pre>Private Sub qryTbl_BeforeRefresh(Cancel As Boolean) Response = MsgBox("Are you sure you " & " want to refresh now?", vbYesNoCancel) If Response = vbNo Then Cancel = True End Sub</pre>
AfterRefresh	<p>该事件发生在查询完成或者被取消。如果查询成功完成则参数 Success 为 True。</p> <pre>Private Sub qryTbl_AfterRefresh(ByVal Success As Boolean) If Success Then MsgBox "The data has been refreshed." Else MsgBox "The query failed."</pre>

```
End If
End Sub
```

10.接下来……

在本章中，你获得了便利的事件经验和Excel的事件编程，这是无价的技术，不管你是否计划给他人创建电子表格应用软件，还是简单地将你的日常任务自动化。Excel提供了许多你可以响应的事件。通过编写事件过程，你可以更改对象对事件的响应方式。你的事件过程可以简单为一条语句，仅仅显示一自定义信息；也可以复杂到包括一些判断语句和其它允许你改变你的程序流的编程结构。当某个事件发生时，VB将会直接运行适当的事件过程，而不是按标准的内置方式进行响应。你已经学习了一些编写在标准模块里的事件过程（工作簿，工作表，图表工作表），然而，其它的（内嵌图表，应用软件，查询 表）则需要你在类模块里面使用WithEvents关键字创建一个新对象。你也学习了你可以使用EnableEvents属性激活或者禁止事件。

在下章，你将学习如何使用Excel VB环境下的VBA过程来使用Access数据库。

第十五章 在 Excel 里使用 Access

在第九章里面，你已经学习了从Excel里通过自动控制（用于允许一个应用程序控制另外一个应用程序的对象）来操纵Word和 Outlook 本章将给演示如何编程从Excel里使用Access使用下述方法获取Access数据到电子表格里面 Automation DAO (Data Access Objects)以及ADO (ActiveX Data Objects)。在你学习如何使用Excel VBA在Access数据库里执行各种任务以及获取和存储数据于Access数据库之前，我们来粗略地介绍一下，Microsoft Access用来编程对其对象访问的数据访问方法。

1.对象库

Access数据库包含各种类型的对象，储存在不同的对象库里面，用来使用VBA语言显示、存储或者管理数据。在本章，你将涉猎下面列出的几个库里的对象、属性和方法。

Access 10.0对象库提供了用来显示数据和在Access 2002应用软件上使用的对象。该库储存于MSACC10.OLB文件里，并且可以自阿C:\Program Files\Microsoft Office\Office文件夹里面找到。在引用对话框上设置了对该库的引用之后（将在下节涉及），你将能够在对象浏览器里面访问该库的对象、属性和方法（参见图15-1）。

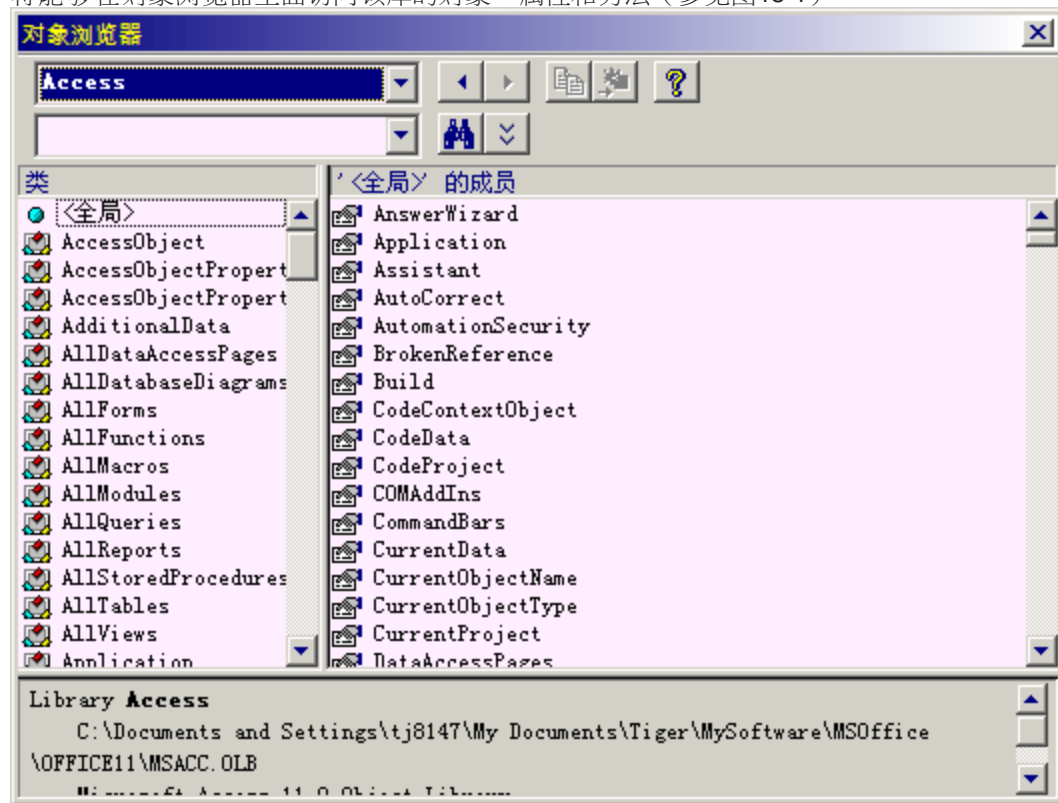


图15-1 Access库（译者：截图为Office 2003。Access库文件为MSACC.OLB）

Access DAO 3.6对象库提供了数据访问对象DAO 让你决定你的数据库的结构和使用VBA操作数据该库储存在DAO360.DLL文件里 并且可以在C:\Program Files\Common Files\Microsoft Shared\DAO文件夹里找到 在引用对话框上设置了对该库的引用之后（将在下节涉及），你将能够在对象浏览器里面访问该库的对象、属性和方法（参见图15-2）。

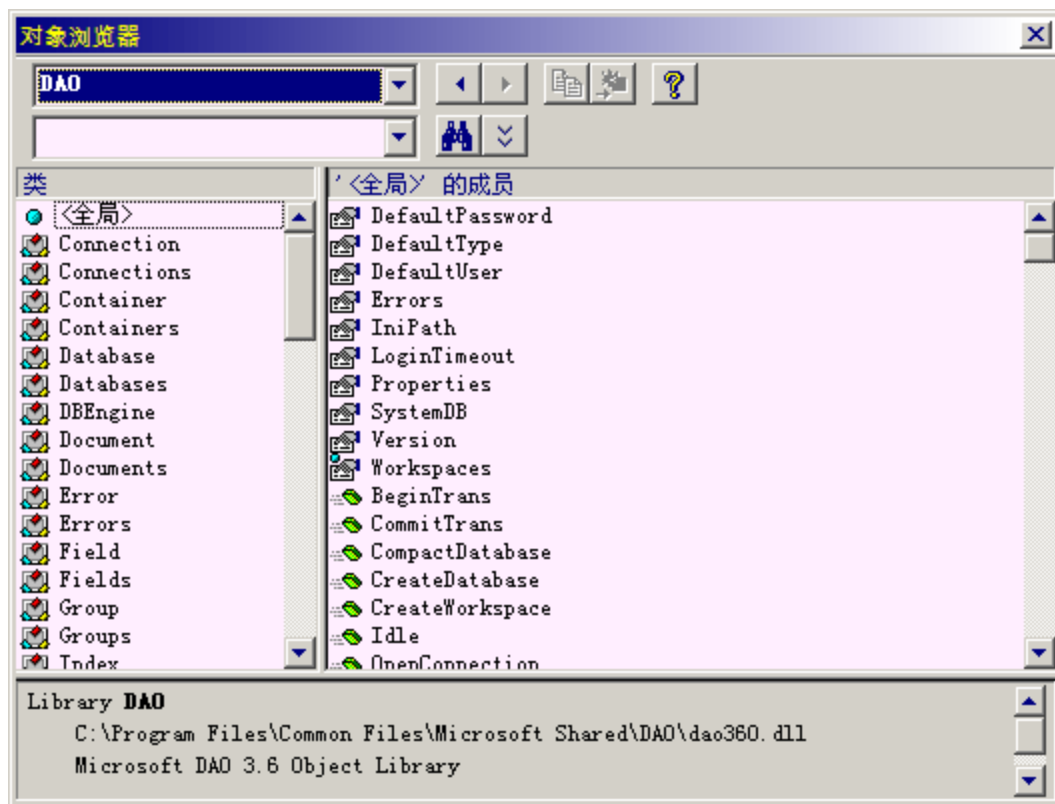


图15-2 DAO库

Microsoft ActiveX Data Objects 2.5 (ADO) 提供了控件数据对象（ADO）并且允许你使用OLE DB供应者访问和操作数据。ADO使得在Access数据库里对数据源创建链接，读取，插入，修改和删除数据成为可能。该库储存于MSADO15.DLL里面，并可以在C:\Program Files\Common Files\system\ado文件夹里找到。在引用对话框上设置了对该库的引用之后，你将能够在对象浏览器里面访问该库的对象、属性和方法（参见图15-3）。

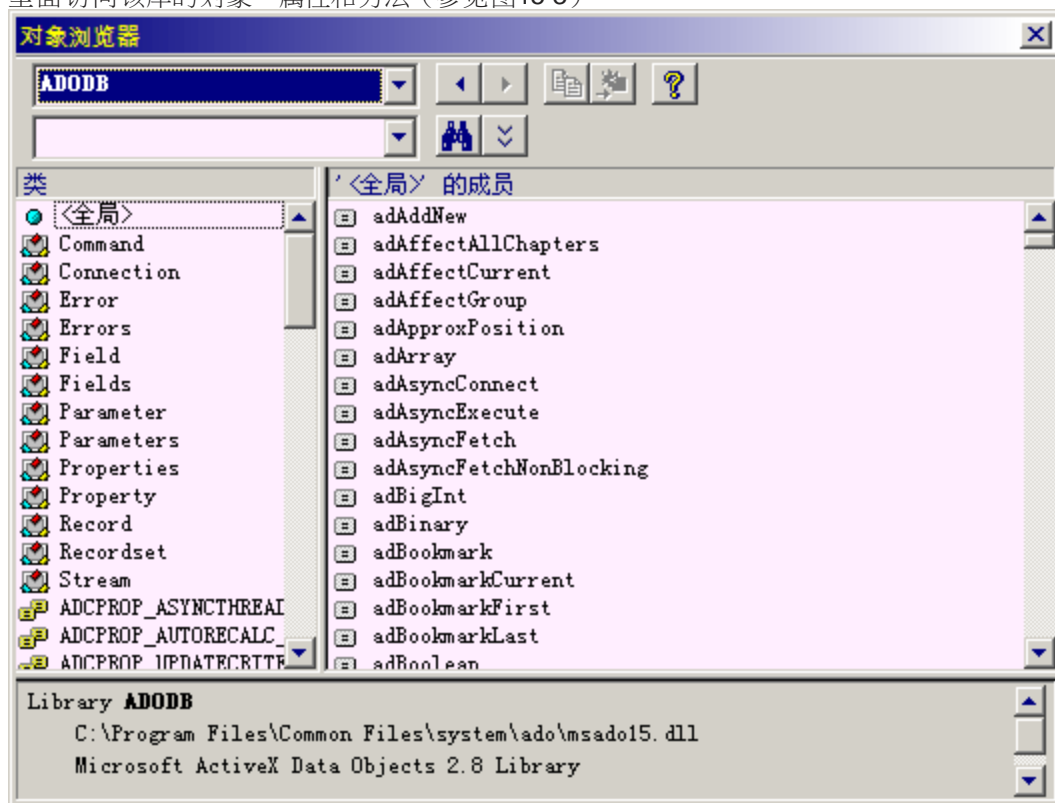


图15-3 ADODB库

Microsoft ADO Ext. 2.5 for DDL (动态数据链接) and Security (安全) (ADOX) 储存让你定义数据库结构和安全的对象。例如，你可以定义表格，索引和关系，以及创建和修改用户和用户组帐户。

该库储存在MSADOX.DLL里并且可以在C:\Program Files\Common Files\System\ado文件夹里找到。在引用对话框上设置了对该库的引用之后，你将能够在对象浏览器里面访问该库的对象、属性和方法（参见图15-4）。

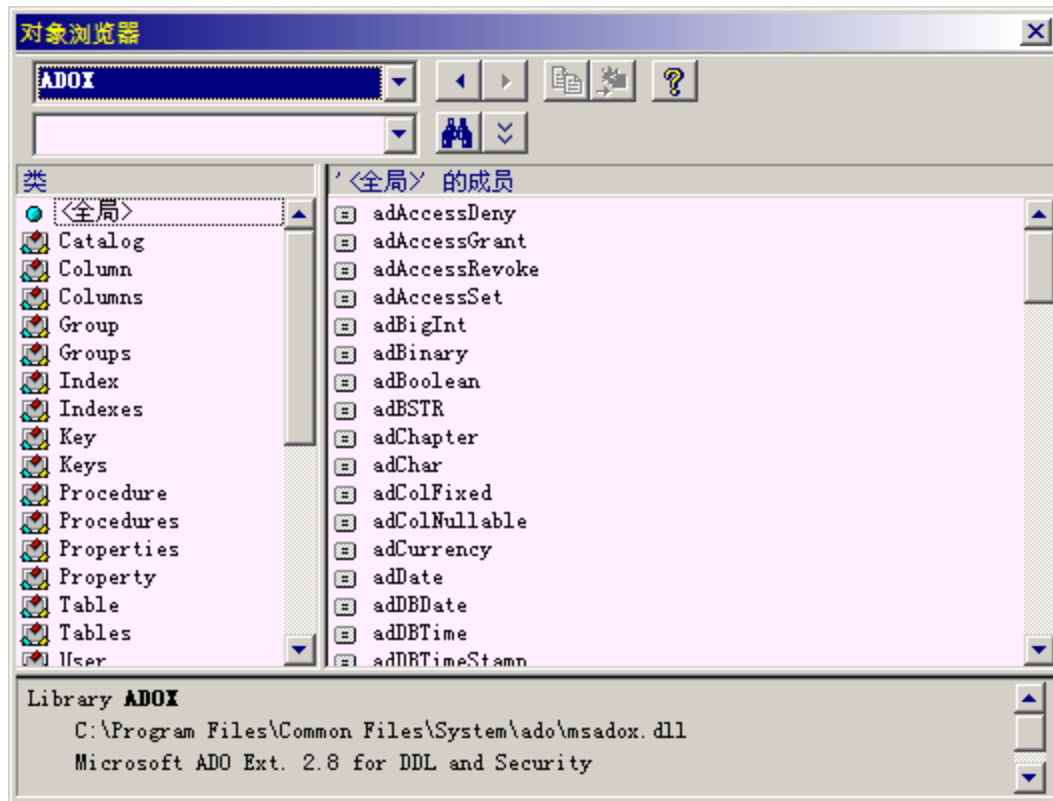


图15-4 ADOX库

Microsoft Jet and Replication Objects 2.6 库(JRO)包含用于对象库复制的对象。该库储存在MSJRO.DLL里并在C:\Program Files\Common Files\System\ado文件夹里可以找到。在引用对话框上设置了对该库的引用之后，你将能够在对象浏览器里面访问该库的对象、属性和方法（参见图15-5）。

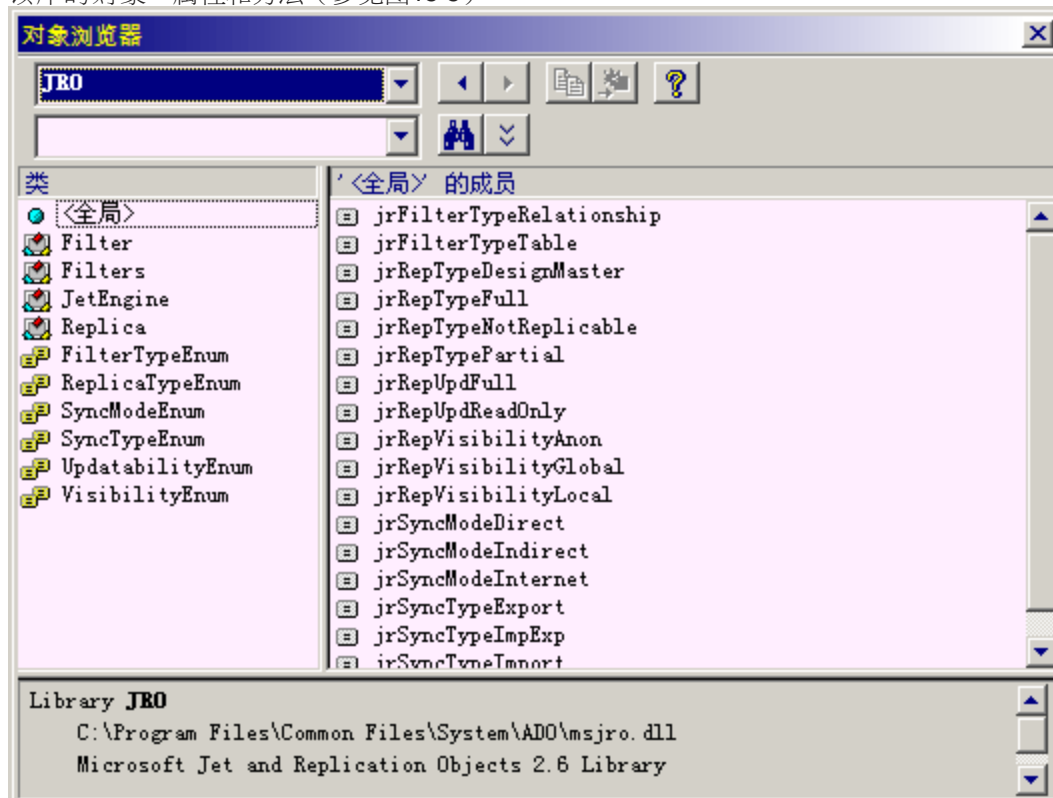


图15-5 JRO库

VBA对象库提供了很多VBA对象，函数和方法供你访问文件系统，操作日期和时间函数，进行数学和财务计算，与用户互动，转换数据和读取文本文件。该库储存在VBE6.DLL文件里，位于C:\Program Files\Common Files\Microsoft Shared\VBA\VBA6文件 夹里。当你安装Microsoft Excel 2002时，就会自动设置对该库的引用。该库在Office 2002所有的应用软件中共享(参见图15-6)。

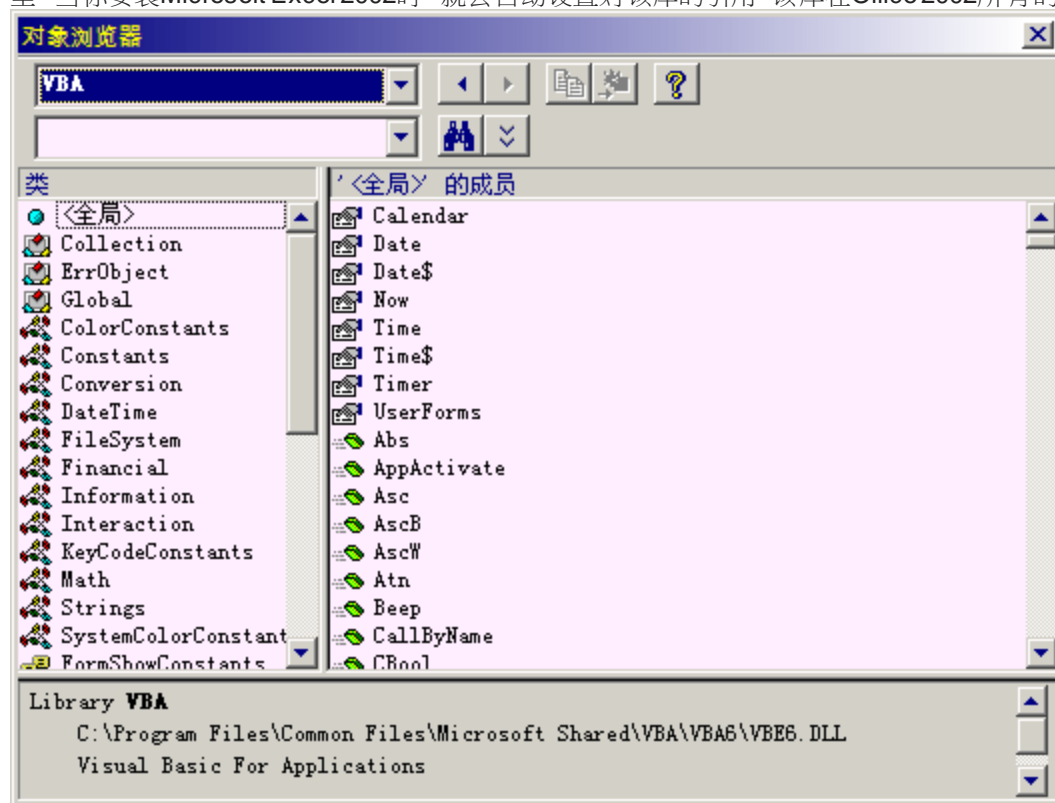


图15-6 VBA库

2. 建立对对象库的引用

要操作Access 2002里的对象的话，首先就得创建对Microsoft Access 10.0对象库的引用。

1. 在VB编辑器窗口，选择工具|引用以打开引用对话框。该对话框列出了你电脑上所有可用的类型库
2. 在清单上找到Microsoft Access 10.0 Object Library，并勾选上它
3. 关闭引用对话框 一旦创建了对Access类型库的引用，你就可以使用对象浏览器来查看该应用软件的对象，属性和方法了（参见前面的图15-1）。使用引用对话框创建对其它将在本章练习中访问对象库的引用你可以在前面部分里找到库清单你可以忽略对Microsoft Jet and Replication Objects 2.6 Library (JRO)的引用，因为这里不会用到它。如果你对数据库复制有兴趣的话，那么可以找到很多有关Access编程的书涉及该主题，包括本人的书Learn Microsoft Access 2000 Programming by Example（Wordware Publishing (ISBN 1-55622-770-1)）。

技巧15-1 创建对Access对象库引用的好处

当你设置对Access对象库的引用时，你将获得下述好处：

- 你可以在对象浏览器里面查看Access的对象，属性和方法
- 你可以在VBA过程里直接运行Access函数
- 你可以声明该应用软件类型的对象变量，而不必声明普通的Object类型。声明变量为Dim objAccess As Access.Application（早期捆绑）要比将其声明为Dim objAccess As Object（后期捆绑）要快。
- 你可以在你的VBA代码里使用Access内置常量
- 你的VBA过程将运行得更快一些

3. 链接到 Access

本章中的示例使用了多种链接到Access的方法，本节将详细讨论每种链接方法。你可以使用下述三种方法之一来建立对Access的链接：

- „ Automation
- „ Data Access Objects (DAO)
- „ ActiveX Data Objects (ADO)

要访问数据库里的数据的话，你就得打开它。如何打开某个具体的数据库，很大程度上取决于你使用了哪种数据库的链接方法。

4.使用 Automation 链接到 Access 数据库

当你通过Automation从Excel（或者其它应用软件）里使用Access时，你需要采取下述步骤：

1. 设置对Microsoft Access 10.0 Object Library的引用（参见本章前面的“建立对对象库的引用”）
2. 声明一个对象变量，代表Access应用软件对象

```
Dim objAccess As Access.Application
```

在该声明中，objAccess为变量名称，而Access.Application用提供该对象的VB对象库的名称来限定该变量。

3. 返回引用到应用软件对象上，并且将该引用到该对象变量。使用CreateObject函数，GetObject函数或者关键字New来返回 应用软件对象的引用。使用Set语句将应用赋值到对象变量。

```
Dim objAccess As Object
```

```
Set objAccess = CreateObject("Access.Application.10")
```

当对象还没有实例时，可以使用CreateObject函数返回一个引用到应用软件对象。如果Access已经运行了的话，那么新的 实例就已创建了并且确定的对象也已创建了。

```
Dim objAccess As Object
```

```
Set objAccess = GetObject(, "Access.Application.10")
```

或者

```
Set objAccess = GetObject("C:\Program Files\ _
```

```
& "Microsoft Office\Office\Samples\Northwind.mdb")
```

使用GetObject函数返回引用到应用软件对象，以使用Access的当前

实例，或者开启Access并打开一个文件（更多信息参见技巧15-2）

```
Dim objAccess As New Access.Application
```

上面的语句使用关键字New，声明了一个对象变量，返回引用到应用软件对象，并且将该引用赋予对象变量，一步完成。你也可以使用两步法来声明一个对象变量，这样，对该对象的控制会更多些：

```
Dim objAccess As Access.Application
```

```
Set objAccess = New Access.Application
```

„ 当使用关键字New声明对象变量时，Access应用软件并不会打开，直到你开始在VBA代码里真正使用该对象变量。

„ 当使用关键字New声明应用软件对象变量时，Access的一个新实例就会自动创建，你不需要使用CreateObject函数

„ 使用关键字New创建应用软件对象的新实例比使用CreateObject函数要快 因为电脑上可能安装了多个版本的Access，所以需要在函数GetObject或者CreateObject里包括该版本号。下面列出最后四个 Access版本：

Microsoft Access 2003 Access.Application.11 （译者加）

Microsoft Access 2002 Access.Application.10

Microsoft Access 2000 Access.Application.9

Microsoft Access 97 Access.Application.8

Microsoft Access 95 Access.Application.7 一旦使用第三步列出的方法之一创建了应用软件类的新实例，你就可以通过

OpenCurrentDatabase或者NewCurrentDatabase方 法的帮助来打开一个数据库或者创建一个新数据库。你可以使用

CloseCurrentDatabase方法关闭你在程序里打开的Access数据 库。

技巧15-2 GetObject函数的参数

GetObject函数的第一个参数 – Pathname – 是可选的。当你想要使用某个特定文件里的对象是要用到它。第二个参数 – Class – 是必需的，它明确哪个应用软件创建该对象，以及该对象的类型。当第一个参数为可选的而第二个为必需的时，你就必须在第一个参数位置放置一个逗号，如下所示：

```
Dim objAccess As Object
```

```
Set objAccess = GetObject(, Access.Application.10")
```

因为GetObject函数的第一个参数（Pathname）被忽略了，所以，就返回对Access应用软件类的现存的实例的引用。

```
Dim objAccess As Object
```

```
Set objAccess = GetObject("C:\Program Files\ & "Microsoft Office\Office\Samples\Northwind.mdb")
```

如果GetObject函数的第一个参数是个数据库文件的名称，那么就会使用该具体数据库，激活或者创建Access应用软件类的新实例。

既然你知道了如何创建代表应用软件的对象变量 那么我们就来看看一个从Excel VBA过程里直接打开Access数据库的程序示例吧。下页显示的过程AccessViaAutomation将打开一个Access文件Northwind数据库。该过程将使用Access自动控制服务器的当前实例，如果它可用的话。如果Access没有运行，运行时间错误将发生，并且该对象变量将被设置为Nothing。你可以通过在程序里放置On Error Resume Next 语句捕捉该错误。因此，如果Access没有运行，新的Access实例就会被打开。本例子使用关键字New来启动Access的新实例。正如前面所述，除了使用关键字创建新对象实例之外，你也可以使用CreateObject()函数来启动

自动控制服务器的新实例，如下所示：

```
Set objAccess = GetObject(, "Access.Application.10")
```

```
If objAccess Is Nothing Then
```

```
    Set objAccess = CreateObject("Access.Application.10")
```

End If 当你使用自动控制启动Access时，你将在任务栏上看到Access图标。Access应用软件对象的Visible属性被设置为False。想要恢复该应用软件窗口的话，就得将其Visible属性设置为True。在使用时，对象要消耗内存和系统资源。要释放这些资源的话，那么每次使用完它的时候总应该关闭该对象。下面示范的程序例子首先使用CloseCurrentDatabase方法关闭Northwind数据库。接着使用Quit方法关闭Access应用程序对象。在关闭对象后，你也应该将对象变量设置为关键字Nothing以释放该变量使用的内存资源。你可以将对象变量声明为模块级，而不是过程级变量，以避免Access实例被关闭。在这样的环境下，对数据库的链接就会保持，直到你关闭该自动控制的控制源（Excel）或者在VBA代码里使用Quit方法关闭它。

```
Sub AccessViaAutomation()
```

```
    Dim objAccess As Access.Application
```

```
    Dim strPath As String
```

```
On Error Resume Next
```

```
Set objAccess = GetObject(, "Access.Application.9")
```

```
If objAccess Is Nothing Then
```

```
    ' Get a reference to the Access Application object
```

```
    Set objAccess = New Access.Application
```

```
End If
```

```
strPath = "C:\Program Files\Microsoft Office\" & _  
    & "Office\Samples\northwind.mdb"
```

```
' Open the Northwind database
```

```
With objAccess
```

```
    .OpenCurrentDatabase strPath
```

```
    If MsgBox("Do you want to make the Access " & vbCrLf & _  
        & "Application visible?", vbYesNo, _  
        "Display Access") = vbYes Then
```

```
        .Visible = True
```

```
        MsgBox "Notice the Access Application icon " & _  
            & "now appears on the Windows taskbar."
```

```
    End If
```

```
' Close the database and quit Access
```

```
    .CloseCurrentDatabase
```

```
    .Quit
```

```
End With
```

```
Set objAccess = Nothing
```

```
End Sub
```

使用F8键逐语句运行上面的过程。

技巧15-3 打开被保护了的Access数据库

如果该Access数据库用密码保护了，那么会提示用户输入正确的密码。你必须使用Data Access Objects (DAO)或者ActiveX Data Access (ADO)来编程打开密码保护的Access数据库。下面的例子使用了Access对象的DBEngine属性来确定该数据库的密码。要让该过程工作的话，你就必须创建对Microsoft DAO 3.6对象库的引用，如本章开头所述。

```
Sub OpenSecuredDB()
```

```
    Static objAccess As Access.Application
```

```
    Dim db As DAO.Database
```

```
    Dim strDb As String
```

```
    strDb = "C:\Program Files\Microsoft Office\" & "Office\Samples\" & _  
        "Northwind.mdb"
```

```
    Set objAccess = New Access.Application
```

```
    Set db = objAccess.DBEngine.OpenDatabase(Name:=strDb, Options:=False, _
```


ReadOnly:=False, Connect:=";PWD=test")

```

With objAccess
    .Visible = True
    .OpenCurrentDatabase strDb
End With
db.Close
Set db = Nothing
End Sub

```

5.使用 DAO 链接到 Access 数据库

要使用数据访问对象(DAO)连接到Access数据库的话,你就必须首先在引用对话框里创建对Microsoft Data Access Objects 3.6 Library的引用。下面示范的程序例子DAOOpenJetDatabase使用DBEngine对象的OpenDatabase方法来打开Northwind数据库,并且通知用户该数据库已被打开。DBEngine对象让你初始化称为Microsoft Jet Engine的标准数据库引擎并打开一个数据库文件(.mdb)。过程使用Close方法关闭数据库文件。

```

Sub DAOOpenJetDatabase()
    Dim db As DAO.Database
    Set db = DBEngine.OpenDatabase _
        ("C:\Program Files\Microsoft Office\Office\Samples\Northwind.mdb")
    MsgBox "Northwind database has been opened."
    db.Close
    MsgBox "Northwind database has been closed."
End Sub

```

6.使用 ADO 链接到 Access 数据库

最新的,最建议的建立对Access数据库链接的方法是使用ActiveX 数据对象(ADO)。你必须先设置对微软ActiveX数据对象2.5库或者更高版本的引用。示例程序ADOOpenJetDatabase使用Connection对象链接到Northwind数据库。该对象通过Open方法打开。注意,Open方法需要一个包含数据提供者名称(本例中为Microsoft.Jet.OLEDB.4.0)和数据源名称(本例中为要打开的数据库文件完整名称)的链接字符串参数:

```

con.Open _
    "Provider=Microsoft.Jet.OLEDB.4.0;" _
    & "Data Source=C:\Program Files\Microsoft Office\" _
    & "Office\Samples\NorthWind.mdb;"

```

在建立对Northwind数据库的链接之后,你可以使用Recordset对象来访问其数据。

Recordset对象用来在记录级操作数据。Recordset对象由记录(行)和字段(列)组成。要获得一套记录,你就得使用Open方法打开Recordset。该方法需要明确的信息,例如Recordset记录源:

```

rst.Open "SELECT * FROM Customers " & _
    "WHERE City = 'London'", con, _
    adOpenForwardOnly, adLockReadOnly

```

记录源可以是返回记录的数据库表,或查询或SQL语句。在明确记录源后,你还需要表明对数据库(con)和两个常数,一个定义指针类型(adOpenForwardOnly),另一个为锁定类型(adLockReadOnly)。常数adOpenForwardOnly告诉VBA创建只能向前翻的Recordset。第二个常数adLockReadOnly明确在编辑时记录上的锁定类型。该记录为只读,意味着你不能改变该数据。过程的下一部分使用For...Each...Next循环遍历Recordset并将第一条记录的内容打印到立即窗口:

```

For Each fld In rst.Fields
    Debug.Print fld.Name & "=" & fld.Value & vbCr
Next

```

在获取第一条记录的数据后,过程使用了Close方法关闭Recordset和对Access数据库的链接:

```

rst.Close
con.Close

```

ADOOpenJetDatabase过程如下:

```

Sub ADOOpenJetDatabase()
    Dim con As New ADODB.Connection
    Dim rst As New ADODB.Recordset
    Dim fld As ADODB.Field

```

```
' Connect with the database
con.Open _
    "Provider=Microsoft.Jet.OLEDB.4.0;" _
    & "Data Source=C:\Program Files\Microsoft Office\" _
    & "Office\Samples\NorthWind.mdb;"
```

```

' Open Recordset based on the SQL statement
rst.Open "SELECT * FROM Customers " & _
"WHERE City = 'London'", con, _
adOpenForwardOnly, adLockReadOnly

' Print the values for the fields in
' the first record in the debug window
For Each fld In rst.Fields
    Debug.Print fld.Name & "=" & fld.Value & vbCr
Next

' Close the Recordset and connection with Access
rst.Close
con.Close

' Destroy object variables to reclaim the resources
Set rst = Nothing
Set con = Nothing
End Sub

```

7.从 Excel 执行 Access 任务

从Excel链接到Access后，你就可以执行Access应用软件的不同任务。本节示范如何使用VBA代码来：

- „ 创建新Access数据库
- „ 打开现存在的数据库表
- „ 创建全新的数据库表
- „ 打开数据库报表
- „ 运行Access函数

8.创建新 Access 数据库

如果你想要通过编程将Excel数据传送到一个新的Access数据库里面，那么你需要使用VBA代码创建一数据库。下面的过程示例示范了如何使用DAO来建立和Access的链接Workspace对象的CreateDatabase方法创建一个名为ExcelDump.mdb的新数据库于C盘根目录下。然后，Database对象的CreateTableDef方法用来创建一个名为tblStates的表。在表能够添加到数据库之前，必须先创建一个字段并附在该表上。该过程创建了一个文本字段（dbText），每个分别可以储存2，25和25个字符。每个字段创建后，使用Append方法将这些字段添加到TableDef对象的Fields集合里。字段一旦创建并添加到表之后，表本身就会使用Append方法被添加到数据库。因为名为“C:\ExcelDump.mdb”的数据库可能已经存在于该目录下，该过程包括了一个错误处理程序，将删除现有文件，以确保数据库创建过程继续。因为其它错误也可能发生，Else子句包括了显示错误描述的信息，并允许退出过程

```

Sub NewDB_DAO()
    Dim db As DAO.Database
    Dim tbl As DAO.TableDef
    Dim strDb As String
    Dim strTbl As String

    On Error GoTo Error_CreateDb_DAO

    strDb = "C:\ExcelDump.mdb"
    strTbl = "tblStates"

    ' Create a new database named ExcelDump
    Set db = CreateDatabase(strDb, dbLangGeneral)

    ' Create a new table named tblStates
    Set tbl = db.CreateTableDef(strTbl)

    ' Create fields and append them to the Fields collection

```

```
With tbl
    .Fields.Append .CreateField("StateId", dbText, 2)
    .Fields.Append .CreateField("StateName", dbText, 25)
    .Fields.Append .CreateField("StateCapital", dbText, 25)
```

End With

```
' Append the table object to the TableDefs
db.TableDefs.Append tbl
```

```
' Close the database
```

```
db.Close
```

```
Set db = Nothing
```

```
MsgBox "There is a new database on your hard disk. " & vbCrLf _
    & "This database file contains a table " & strDb & vbCrLf _
    & "named " & strTbl & "." & vbCrLf _
    & "Before you activate this database, close the Excel application."
```

```
Exit_CreateDb_DAO:
```

```
Exit Sub
```

```
Error_CreateDb_DAO:
```

```
If Err.Number = 3204 Then
```

```
' Delete the database file if it already exists
```

```
Kill "C:\Exceldump.mdb"
```

```
Resume
```

```
Else
```

```
MsgBox Err.Number & ": " & Err.Description
```

```
Resume Exit_CreateDb_DAO
```

```
End If
```

```
End Sub
```

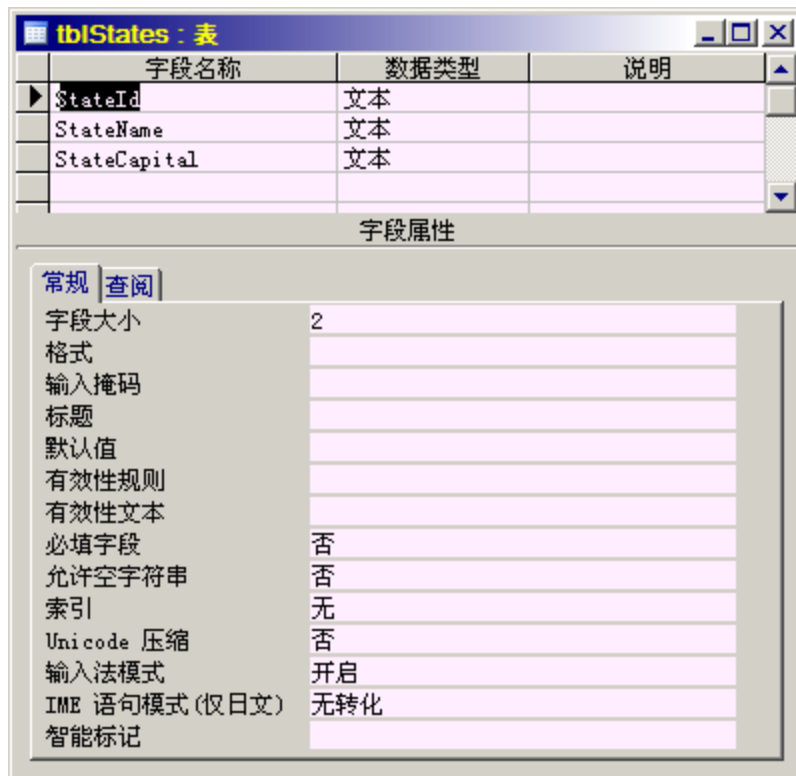


图15-7 Excel VBA过程创建的Access数据库表

9.打开 Access 窗体

你可以从Excel里打开Access窗体。你也可以创建新窗体。下述例子使用自动控制链接到Access。一旦链接建立后，就使用OpenCurrentDatabase方法来打开例子Northwind数据库。接着，使用DoCmd对象的OpenForm方法打开Customers窗体。该窗体

被打开为普通视图 (`acNormal`)。如果要窗体在设计视图里打开的话,那么可以使用 `acDesign` 常数代替。`DoCmd` 对象的 `Restore` 方法确保该窗体显示在屏幕上而不是最小化。`Access` 应用软件对象 (`objAccess`) 的 `Visible` 属性必须设置为 `True`, 以确保窗体可见。注意, `Access` 应用软件的对象变量 (`objAccess`) 在模块上面声明。为了让该过程运行正确,你必须建立对 `Access`

对象库的引用。图15-8显示了被打开的Customers窗体。

```
' declare at the top of the module
Dim objAccess As Access.Application

Sub DisplayAccessForm()
    Dim strDb As String
    Dim strFrm As String
    strDb = "C:\Program Files\Microsoft Office\" _
        & "Office\Samples\Northwind.mdb"
    strFrm = "Customers"
    Set objAccess = New Access.Application

    With objAccess
        .OpenCurrentDatabase(strDb)
        .DoCmd.OpenForm strFrm, acNormal
        .DoCmd.Restore
        .Visible = True
    End With
End Sub
```

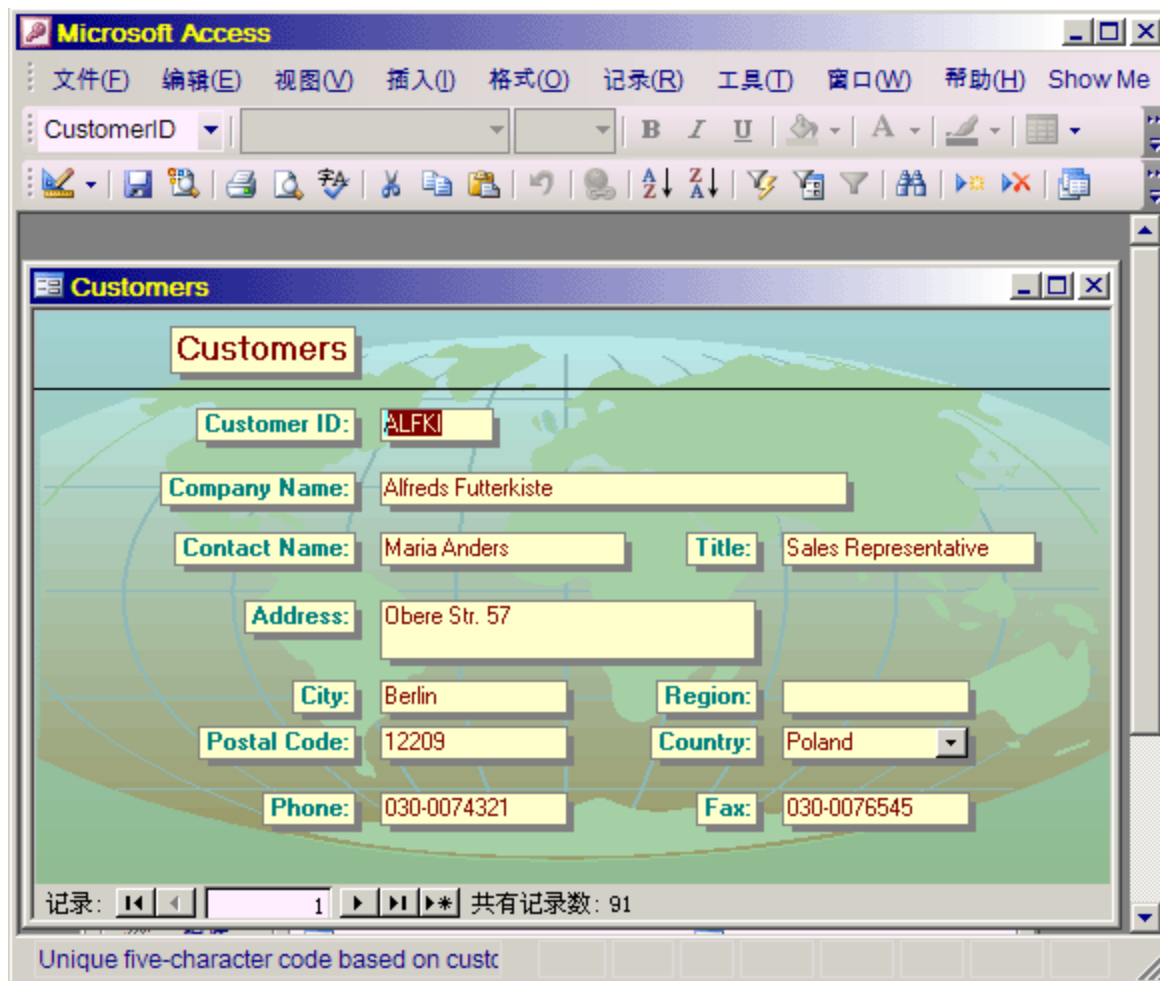


图15-8 可以用Excel VBA过程打开的Access窗体

如果你还想在编程中再进一步的话，那么从Excel VBA过程里创建一个全新的Access窗体，如下所示：

```
' declare at the top of the module
Dim obAccess As Access.Application （译者：原文为myAccess）
Sub CreateAccessForm()
    Dim myForm As Form
```

```
Dim myDb As String  
Dim myCtrl As Control  
Dim strFrmName As String
```

```

On Error GoTo Error_CreateForm
myDb = "C:\Program Files\Microsoft Office\" _
    & "Office\Samples\Northwind.mdb"
strFrmName = "frmCustomForm"
Set obAccess = New Access.Application
obAccess.OpenCurrentDatabase myDb

```

```

Set myForm = obAccess.CreateForm
myForm.Caption = "Form created by Excel"
myForm.RecordSource = "Employees"
obAccess.DoCmd.Save , strFrmName

```

```

' Create a label and text box on the form

```

```

Set myCtrl = CreateControl(FormName:=strFrmName, _
    ControlType:=acLabel, _
    Left:=1000, Top:=1000)

```

```

myCtrl.Caption = "Last Name:"

```

```

myCtrl.SizeToFit

```

```

Set myCtrl = CreateControl(FormName:=strFrmName, _
    ControlType:=acTextBox, _
    Parent="", _
    ColumnName="LastName", _
    Left:=2200, Top:=1000)

```

```

With obAccess

```

```

    With .DoCmd

```

```

        .Save , strFrmName

```

```

        .Close acForm, strFrmName

```

```

    End With

```

```

    .CloseCurrentDatabase

```

```

    .Quit

```

```

End With

```

```

Set obAccess = Nothing

```

```

MsgBox "In the Northwind database there is now " & vbCrLf _
    & "a new form named " & strFrmName & "." & vbCrLf _
    & "Close Excel prior to opening the Northwind " & vbCrLf _
    & "database to view this form."

```

```

ErrorHandler:

```

```

    Exit Sub

```

```

Error_CreateForm:

```

```

    MsgBox Err & " : " & Err.Description

```

```

    Resume ErrorHandler

```

```

End Sub

```

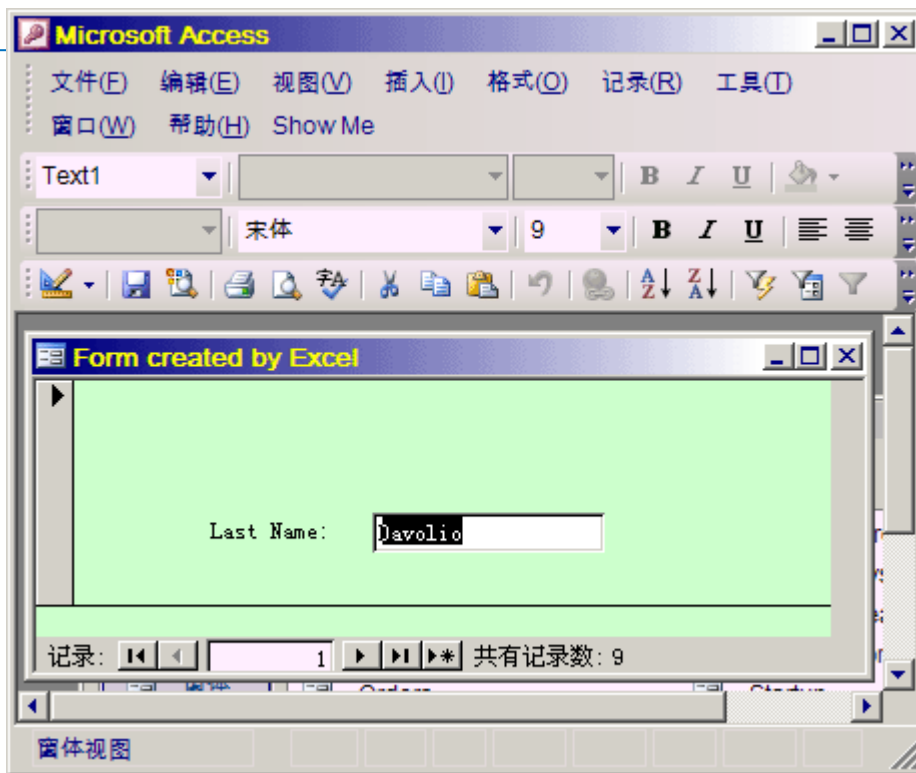


图15-9 Access窗体可以由Excel VBA过程创建（参见上面的CreateAccessForm过程代码）

10.打开 Access 报表

你可以从Excel里打开Access报表。下述过程示范了如何直接从Excel里显示已经存在的Access报表。

```
' declare at the top of the module
Dim objAccess As Access.Application
Sub DisplayAccessReport()
    Dim strDb As String
    Dim strRpt As String
    strDb = "C:\Program Files\Microsoft Office\" _
        & "Office\Samples\Northwind.mdb"
    strRpt = "Products by Category"
    Set objAccess = New Access.Application
    With objAccess
        .OpenCurrentDatabase (strDb)
        .DoCmd.OpenReport strRpt, acViewPreview
        .DoCmd.Maximize
        .Visible = True
    End With
End Sub
```

下面的过程更通用，因为它允许你在任意Access数据库里显示任意Access报表。注意，该过程需要两个字符串参数：Access数据库名称和报表名称。

```
Sub DisplayAccessReport2(strDb As String, strRpt As String)
    Set objAccess = New Access.Application
    With objAccess
        .OpenCurrentDatabase (strDb)
        .DoCmd.OpenReport strRpt, acViewPreview
        .DoCmd.Maximize
        .Visible = True
    End With
End Sub
```

你可以从立即窗口或者从如下所示的一个子过程里运行DisplayAccessReport2：

„ 从立即窗口运行DisplayAccessReport2过程

在立即窗口里在一行输入下述语句：

```
Call DisplayAccessReport2("C:\Program Files\Microsoft Office\Office\Samples\Northwind.mdb", "Sales Totals by Amount")
```


从一个子过程运行DisplayAccessReport2过程：

```
' Enter the following procedure in the Code window
Sub ShowReport()
    Dim strDb As String
    Dim strRpt As String
    strDb = InputBox("Enter the name of the database (full path): ")
    strRpt = InputBox("Enter the name of the report:")
    Call DisplayAccessReport2(strDb, strRpt)
End Sub
```

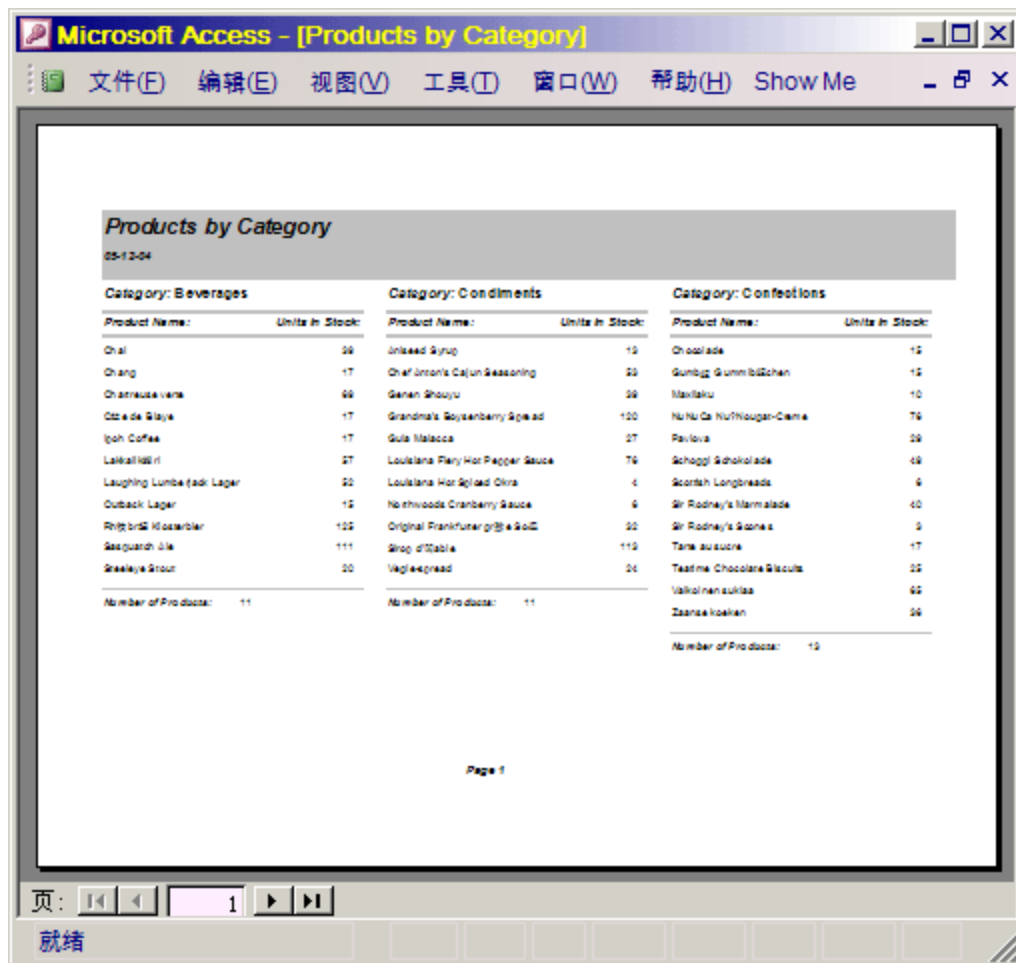


图15-10 Access报表可以在Excel VBA过程里打开

11.运行 Access 查询

接下来的两个程序例子将示范如何从Excel VBA过程里运行Access查询。在Access用户界面最常用的查询类型是选择查询和参数查询。两个示例程序都使用Range对象的CopyFromRecordset方法将查询到的数据放置到Excel工作表。和数据库的链接是通过ADO建立的。

ADOX对象库(参见本章前面的图15-4)让你访问数据库结构,安全和储存在数据库里面的过程。该库中最上面的对象是Catalog对象,代表整个数据库。该对象包含一些数据库成员,例如,表,字段,索引,视图和储存的过程。使用Catalog对象的Create方法,你可以创建一个新的数据库,例如:

```
Dim cat As ADOX.Catalog
Set cat = New ADOX.Catalog
cat.Create "Provider=Microsoft.Jet.OLEDB.4.0;" &
```

"Data Source=C:\ExcelDump2.mdb;" 上面的例子示范如何使用ActiveX数据对象创建新的数据库。回想一下,在本章前面你使用DAO创建了一个叫做NewDB_DA的新数据库。示例过程RunAccessQuery,首先创建一个可以指向该Catalog对象的对象

变量cat。接着，Catalog对象的属性 ActiveConnection定义对数据库创建链接的方法：

```
Set cat = New ADOX.Catalog  
cat.ActiveConnection = "Provider=Microsoft.Jet.OLEDB.4.0;" & _  
    "Data Source=" & dbPath
```

ADODB对象库里的Command对象（参见本章前面的图15-3）明确你为了从数据源获取数据而想要执行的命令。我们的过程尝试访问数据库某特定查询。

Set cmd = cat.Views(strQryName).Command Views集合，ADOX对象库的一部分，包含某特定目录的所有View对象。视图是筛选后的一组记录，或者由其它表或者视图创建的虚拟表。获得对数据库里需要的查询的访问后，你就可以按下述方式运行查询：

Set rst = cmd.Execute Command对象的Execute方法允许你激活某个特定的查询，SQL语句，或者储存的过程。然后，返回的一组记录会通过Set关键字被赋予对象变量Recordset。创建该组记录后，这些记录就会通过使用方法CopyFromRecordset放置到Excel工作表中。

12.运行选择查询

```
Sub RunAccessQuery(strQryName As String)
    ' prior to running this procedure you must set up
    ' references to the required object libraries
    Dim cat As ADOX.Catalog
    Dim cmd As ADODB.Command
    Dim rst As ADODB.Recordset
    Dim i As Integer
    Dim dbPath As String

    dbPath = "C:\Program Files\Microsoft Office\" & _
        & "Office\Samples\Northwind.mdb"

    Set cat = New ADOX.Catalog
    cat.ActiveConnection = _
        "Provider=Microsoft.Jet.OLEDB.4.0;" & _
        "Data Source=" & dbPath
    Set cmd = cat.Views(strQryName).Command
    Set rst = cmd.Execute
    Sheets(2).Select
    For i = 0 To rst.Fields.Count - 1
        Cells(1, i + 1).Value = rst.Fields(i).Name
    Next

    With ActiveSheet
        .Range("A2").CopyFromRecordset rst
        .Range(Cells(1, 1), _
            Cells(1, rst.Fields.Count)).Font.Bold = True
        .Range("A1").Select
    End With

    Selection.CurrentRegion.Columns.AutoFit
    rst.Close
    Set cmd = Nothing
    Set cat = Nothing
End Sub
```

想要运行上述过程的话，可以在立即窗口里面输入下述语句并回车：

```
RunAccessQuery("Current Product List")
```

	A	B
7	2	Chang
8	39	Chartreuse verte
9	4	Chef Anton's Cajun Seasoning
10	48	Chocolade
11	38	Coeur de Blaye
12	58	Escargots de Bourgogne
13	52	Filo Mix
14	71	Flemish
15	33	Geitost
16	15	Genen Shouyu
17	56	Gnocchi di nonna Alice
18	31	Gorgonzola Telino
19	6	Grandma's Boysenberry Spread

图15-11 从Excel VBA过程运行Access查询的结果被放在了一个工作表里了

(译者，从立即窗口运行可能遇到编译错误，如果这样的话，可以从一个子程序里调用该过程。)

13.运行参数查询

你可以运行Access参数查询并将其结果放置于Excel电子表格里面 例如 过程 RunAccessParamQuery通过Access数据库的参数查询运行Employee Sales by Country，并且取得7/1/96和7/30/96区间内的记录。Employee Sales by Country查询要求两个参数：开始和结束时间。

应该使用Command对象的Parameters集合定义这些参数：

```
cmd.Parameters("[Beginning Date]") = StartDate
```

```
cmd.Parameters("[Ending Date]") = EndDate
```

设置参数后，该查询就可以使用下述语句执行了：

Set rst = cmd.Execute 该查询返回的记录会被赋予对象变量Recordset并使用CopyFromRecordset方法复制到工作表（参见本章后面更多相关使用信息）。

```
Sub RunAccessParamQuery()
```

```
    ' prior to running this procedure you must set up
```

```
    ' references to the required object libraries
```

```
    Dim cat As ADOX.Catalog
```

```
    Dim cmd As ADODB.Command
```

```
    Dim rst As ADODB.Recordset
```

```
    Dim i As Integer
```

```
    Dim dbPath As String
```

```
    Dim StartDate As String
```

```
    Dim EndDate As String
```

```
    dbPath = "C:\Program Files\Microsoft Office\" & _
```

```
        & "Office\Samples\Northwind.mdb"
```

```
    StartDate = "7/1/96"
```

```
    EndDate = "7/31/96"
```

```
    Set cat = New ADOX.Catalog
```

```
    cat.ActiveConnection = "Provider=Microsoft.Jet.OLEDB.4.0; & _" & _
```

```
        "Data Source=" & dbPath
```

```
    Set cmd = cat.Procedures("Employee Sales by Country").Command
```

```
    cmd.Parameters("[Beginning Date]") = StartDate
```

```
    cmd.Parameters("[Ending Date]") = EndDate
```

```
    Set rst = cmd.Execute
```

```
Sheets(1).Select  
For i = 0 To rst.Fields.Count - 1  
    Cells(1, i + 1).Value = rst.Fields(i).Name  
Next
```

```

With ActiveSheet
    .Range("A2").CopyFromRecordset rst
    .Range(Cells(1, 1), Cells(1, rst.Fields.Count)).Font. _
        Bold = True
    .Range("A1").Select
End With

Selection.CurrentRegion.Columns.AutoFit
rst.Close
Set cmd = Nothing
Set cat = Nothing
End Sub

```

（译者：本人运行上述过程不成功，将Parameters参数改为cmd.Parameters(0) = StartDate，cmd.Parameters(1) = EndDate才运行成功。另外过程给定的日期范围查询结果为空，可以更改查询日期范围。译者使用示例文件Nwind.mdb。自此以下，因删除非法软件，译者使用Excel 2003英文版，Access 2002英文版）

14.调用 Access 函数

你可以通过自动控制从Excel里直接运行Access内置函数。下面的过程调用EuroConvert函数将1000西班牙比塞塔转变为欧元。EuroConvery函数使用欧盟确定的固定汇率。

```

Sub RunAccessFunction()
    Dim objAccess As Object

    On Error Resume Next
    Set objAccess = GetObject(, "Access.Application")

    'if no instance of Access is open, create a new one
    If objAccess Is Nothing Then
        Set objAccess = CreateObject("Access.Application")
    End If

    MsgBox "You will get " & _
        objAccess.EuroConvert(1000, "ESP", "EUR") & _
        " euro dollars. "

    Set objAccess = Nothing
End Sub

```

15.获取 Access 数据到 Excel 工作表

有很多种方法获取外部数据到Excel。本节给你示范下述不同的技巧将Access数据导入Excel工作表：

- 使用GetRows方法
- 使用CopyFromRecordset方法
- 使用TransferSpreadsheet方法
- 使用OpenDatabase方法
- 创建一个文本文件
- 创建一个查询表

16.使用 GetRows 方法获取数据

你可以使用GetRows方法将Access数据放置于Excel工作表。该方法返回一个二维的数组。第一个下标是一个代表字段的数字，而第二个下标则是代表记录的数字。记录和字段从0开始。你通过在VBA过程里使用DAO返回数据到Excel工作表。下述示例过程示范了如何运行Northwind数据库里的Invoices查询，并记录返回到Excel工作表。为了确保该过程工作正确，你必须首先建立对Microsoft Access 3.6 Object Library的引用。参考本章前面的创建对对象库的引用。

打开Access数据库后，GetData_withDAO2过程示范使用下述语句运行Invoices查询：


```
Set qdf = db.QueryDefs("Invoices")
```

Microsoft Access 3.6 对象库里的 **QueryDefs** 对象代表一选择或者行动查询。选择查询从一个或者多个表或者查询里返回数据，然而，行动查询允许你修改数据（使用行动查询你可以添加，修改或者删除记录）

执行查询后，过程将查询返回的记录通过OpenRecordset方法放置到对象变量Recordset上，如下所示：

Set rst = qdf.OpenRecordset 接下来，通过RecordCount方法获取记录数目并且放置于变量countR上。注意，为了获得正确的记录数目，记录指针必须通过 使用MoveLast方法移动到Recordset里的最后一条记录。

rst.MoveLast

countR = rst.RecordCount 接着，过程提示用户输入要返回到工作表的记录数目。你可以点击输入对话框上的取消按钮就此取消，或者输入记录数目获取数据。如果你输入的数字大于该记录数目，过程将获取全部记录。在获取记录之前，你必须使用方法MoveFirst将记录指针移动到 第一条记录。如果你忘了做这个，那么记录指针会停留在最后一条记录上，并且将只能获取一条记录。然后，该过程继续执行， 激活Get Records工作表和清除当前范围内容。首先，通过使用Recordset对象的GetRows方法，记录将返回到一个二维数组的 Variant类型变量。接着，过程在数组的两维中循环将记录放置到工作表中，从单元格A2开始。这一切完成后，另一个循环将在 工作表第一行里放置字段名称，并且将每列设置为自动适应列宽，以正确现实数据。

Sub GetData_withDAO2()

```
Dim db As DAO.Database
Dim qdf As DAO.QueryDef
Dim rst As DAO.Recordset
Dim recArray As Variant
Dim i As Integer
Dim j As Integer
Dim strPath As String
Dim a As Variant
Dim countR As Long
Dim strShtName As String
```

```
strPath = "C:\Program Files\Microsoft Office\" & _
    & "Office\Samples\northwind.mdb"
strShtName = "Returned records"
Set db = OpenDatabase(strPath)
Set qdf = db.QueryDefs("Invoices")
Set rst = qdf.OpenRecordset
rst.MoveLast
countR = rst.RecordCount
```

```
a = InputBox("This recordset contains " & _
    countR & " records." & vbCrLf _
    & "Enter number of records to return: ", _
    "Get Number of Records")
```

```
If a = "" Or a = 0 Then Exit Sub
```

```
If a > countR Then
    a = countR
    MsgBox "The number you entered is too large." & vbCrLf _
        & "All records will be returned."
```

```
End If
```

```
Workbooks.Add
ActiveWorkbook.Worksheets(1).Name = strShtName
rst.MoveFirst
```

```
With Worksheets(strShtName).Range("A1")
    .CurrentRegion.Clear
    recArray = rst.GetRows(a)
    For i = 0 To UBound(recArray, 2)
        For j = 0 To UBound(recArray, 1)
            .Offset(i + 1, j) = recArray(j, i)
        Next j
    Next i
```

```
For j = 0 To rst.Fields.Count - 1
    .Offset(0, j) = rst.Fields(j).Name
```

.Offset(0, j).EntireColumn.AutoFit

```

Next j
End With
db.Close
End Sub

```

17.使用 CopyFromRecordset 方法获取数据

想要将整个Recordset导入工作表的话，你可以使用Range对象的CopyFromRecordset方法。该方法可以使用三个参数：Data，MaxRows和MaxColumns。只有第一个参数Data是必须的。该参数可以是Recordset对象。可选参数MaxRows和MaxColumns 允许你明确应该返回的记录数目（MaxRows）和字段数目（MaxColumns）。如果你忽略MaxRows参数，那么所有返回的记录 将会复制到工作表；如果你忽略MaxColumns参数，那么所有的字段将会被获取。下面示范的过程GetProducts使用ADO对象建立对Northwind数据库的链接。为了让该过程工作正常，你必须先建立对Microsoft ActiveX Data Objects 2.6 Library的引用（参见本章前面有关创建对象库引用的指导）。

```

Sub GetProducts()
    Dim conn As New ADODB.Connection
    Dim rst As ADODB.Recordset
    Dim strPath As String

    strPath = "C:\Program Files\Microsoft Office\" _
        & "Office\Samples\Northwind.mdb"

    conn.Open "Provider=Microsoft.Jet.OLEDB.4.0;" _
        & "Data Source=" & strPath & ";"

    conn.CursorLocation = adUseClient

    ' Create a Recordset from all the records
    ' in the Products table
    Set rst = conn.Execute(CommandText:="Products", _
        Options:=adCmdTable)

    ' begin with the first record rst.MoveFirst
    ' transfer the data to Excel
    ' get the names of fields first

    With Worksheets("Sheet3").Range("A1")
        .CurrentRegion.Clear
        For j = 0 To rst.Fields.Count - 1
            .Offset(0, j) = rst.Fields(j).Name
        Next j

        .Offset(1, 0).CopyFromRecordset rst
        .CurrentRegion.Columns.AutoFit
    End With
    rst.Close
    conn.Close
End Sub

```

上述过程从Northwind数据库的Products表中复制所有的记录到Excel工作表。如果你想要复制某一些记录的话，那么你可以使用 MaxRows参数，如下所示：

.Offset(1, 0).CopyFromRecordset rst, 5 该语句告诉VB仅复制5条记录。该Offset方法导致输入到电子表格里的记录从电子表格当前行的第二行开始。想要仅将两个表字段的所有记录发送到工作表的话，可以使用下述语句：

.Offset(1, 0).CopyFromRecordset rst, , 2

该语句告诉VB从开始两列复制所有数据。在rst和数字2之间的逗号是个占位符，给被忽略的MaxRows参数。

18.使用 TransferSpreadsheet 方法获取数据

可能使用TransferSpreadsheet方法在当前Access数据库(.mdb)或者Access项目(.adp)和电子表格之间导入或者导出数据。你也可以将Excel电子表格里的数据链接到当前Access数据库。对于链接的电子表格，当Access仍然允许从Excel程序里完全访问时，你可以使用Access来查看和编辑电子表格数据。在VB里执行TransferSpreadsheet操作的TransferSpreadsheet方法语法如下：

DoCmd.TransferSpreadsheet [transfertype][, spreadsheettype], _
tablename, filename [, hasfieldnames][, range] 参数transfertype可以是以下常数之一：acImport（缺省设置），acExport或者acLink。这些常数定义数据是否是导入，导出或者 链接到数据库。

参数spreadsheettype可能是下述常数之一：

```
0      acSpreadsheetTypeExcel3 (default setting)
6      acSpreadsheetTypeExcel4
5      acSpreadsheetTypeExcel5
5      acSpreadsheetTypeExcel7
8      acSpreadsheetTypeExcel8
8      acSpreadsheetTypeExcel9
2      acSpreadsheetTypeLotusWK1
3      acSpreadsheetTypeLotusWK3
7      acSpreadsheetTypeLotusWK4
```

不难猜到，spreadsheettype参数明确电子表格名称和版本号。tablename参数是个字符串表达式，明确你想要往里面导入电子表格数据，或者从里面导出电子表格数据，或者将电子表格数据 链接到的Access表的名称。除了表名称之外，你也需要明确你想要导出数据到电子表格的选择查询名称。hasfieldnames参数是个逻辑值True(-1)或者False(0)。True表明工作表第一行包含字段名称；False则表示第一行包含普通数据。缺省设置为False（第一行里没有字段名称）。参数range是个字符串表达式，明确工作表中的单元格区域或者区域名称。该参数仅用于导入。如果你忽略range参数的话，那么整个电子表格将会被导入。如果你想要导出的话，就将该参数空在那里，除非你需要明确该工作表名称。下面示范的ExportData示例程序使用TransferSpreadsheet方法从Northwind数据库里的Shippers表中导出数据到Shippers.xls电子表格中。注意，该过程使用了自动控制来建立对Access的链接。建立链接后，使用OpenCurrentDatabase方法打开Northwind 数据库。运行完ExportData过程后，请打开C:\Shippers.xls文件查看获取的数据。

```
' declare at the top of the module
Dim objAccess As Access.Application
Sub ExportData()
    Set objAccess = CreateObject("Access.Application")
    objAccess.OpenCurrentDatabase filepath:= _
        "C:\Program Files\Microsoft Office\Office\" _
        & "Samples\Northwind.mdb"

    objAccess.DoCmd.TransferSpreadsheet _
        TransferType:=acExport, _
        SpreadsheetType:=acSpreadsheetTypeExcel9, _
        TableName:="Shippers", _
        Filename:="C:\Shippers.xls", _
        HasFieldNames:=True, _
        Range:="Sheet1"
    objAccess.Quit
    Set objAccess = Nothing
End Sub
```

（译者：原文为acSpreadsheetTypeExcel10运行失败）

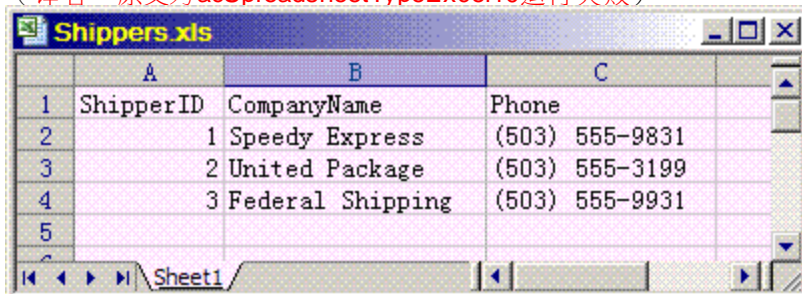


图15-12 使用TransferSpreadsheet方法可以将Access表里的数据导出到Excel电子表格里

19.使用 OpenDatabase 方法

Excel 2002提供了一个操纵数据库的新方法，OpenDatabase方法，应用于Workbooks集合，是将数据库数据导入Excel电子表

格最容易的方法。该方法要求你明确你想要打开的数据库文件名称。下面的示例过程打开位于 `C:\Program Files\Microsoft Office\Office10\Samples` 文件夹里的 `Northwind` 数据库。当你运行该过程，`Excel` 显示一个对话框，列出了该数据库里的所有表和 查询（参见图15-13）。从列表里选择后，就会打开一个全新的工作簿，显示被选上的表或者查询里的数据。


```
Sub OpenAccessDatabase()
    Workbooks.OpenDatabase _
        Filename:="C:\Program Files\Microsoft Office\" _
        & "Office10\Samples\Northwind.mdb"
```

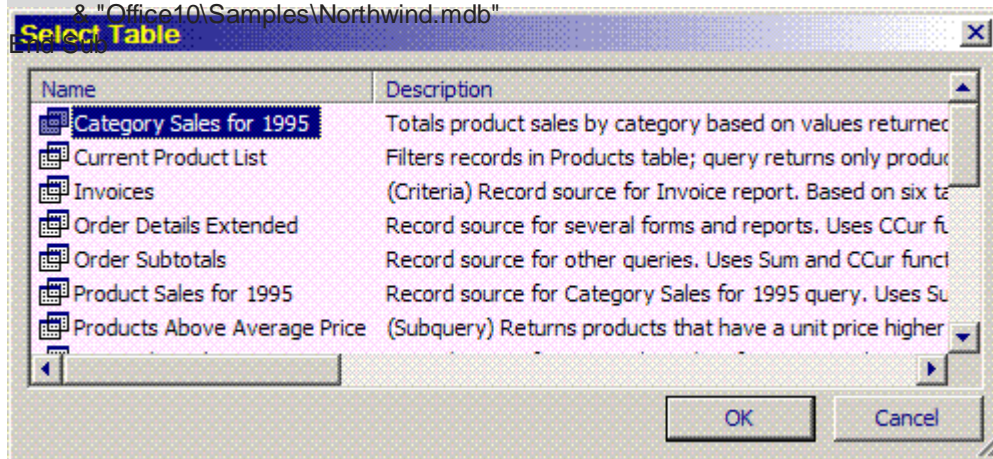


图15-13 使用带一个参数（数据库文件名称）的OpenDatabase方法允许从一个列表框里选择一个表或者查询

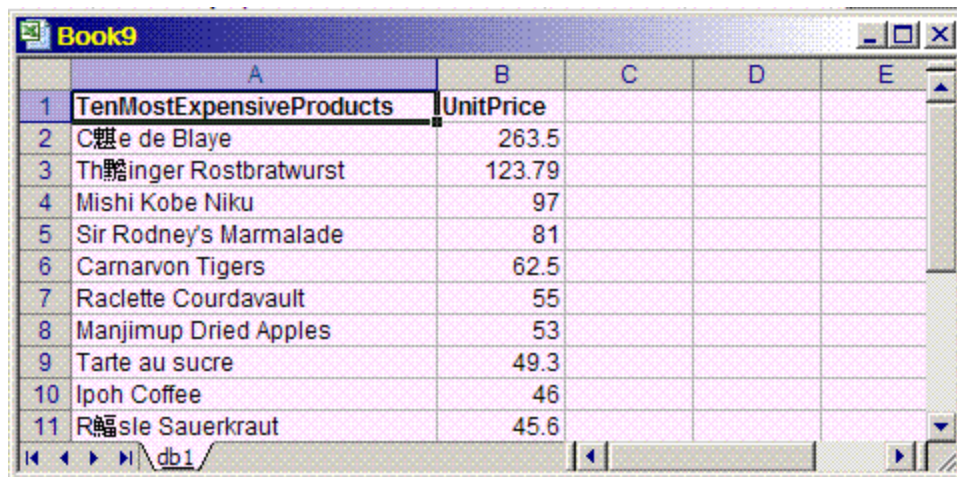


图15-14 使用Excel 2002里新增的OpenDatabase方法可以轻易地将储存在表或者查询里的数据库数据导入Excel工作簿

OpenDatabase方法有四个可选参数，可供你进一步限定你要获取的数据：

OpenDatabase方法的可选参数	数据类型	描述
CommandText	Variant	SQL查询字符串。参见使用该参数的示例
CommandType	Variant	查询的命令类型。可用的命令类型有：Deault，SQL
和表 BackgroundQuery	Variant	查询的背景。可以是以下常数之一：PivotCache
或者QueryTable ImportDataAs	Variant	明确查询的格式。使用
xlQueryTable报告创建一个查询表，或者		

xlPivotTableReport来创建一个数据透视表

接下来的示例过程示范了如何使用带可选参数的OpenDatabase方法。该过程从获取的客户记录创建了一个数据透视表。当你运行该过程时，Excel就会基于提供的查询的字符串显示一个可用字段的列表。你可以拖曳一个或者多个字段到该透视表中，以创建数据透视报告。图15-15显示了按国家分类的CustomerId字段。

```
Sub CountCustomersByCountry()
    Workbooks.OpenDatabase _
        Filename:="C:\Program Files\Microsoft Office\" _
        & "Office10\Samples\Northwind.mdb", _
        CommandText:="Select * from Customers", _
        BackgroundQuery:=PivotTable, _
        ImportDataAs:=xlPivotTableReport
```

End Sub

（译者：Excel 2003+Access 2002运行该过程有问题：BackgroundQuery:= PivotTable。此处有矛盾，上面的参数解释说该参数为PivotCache或者QueryTable，而这里却是PivotTable。这三个参数均导致错误。搜索对象浏览器说该参数为布尔类型。译者将该参数改为-1，1，2，10，True，False等运行，结果没有区别。）

	A	B	C
1			
2			
3	Count of CustomerID		
4	Country	CustomerID	Total
5	Argentina	CACTU	1
6		OCEAN	1
7		RANCH	1
8	Argentina Total		3
9	Austria	ERNSH	1
10		PICCO	1
11	Austria Total		2
12	Belgium	MAISD	1
13		SUPRD	1
14	Belgium Total		2
15	Brazil	COMMI	1
16		FAMIA	1
17		GOURL	1
18		HANAR	1
19		QUEDE	1
20		QUEEN	1

图15-15 使用OpenDatabase方法的可选参数，你可以明确获取数据库数据到一个特定的格式，例如数据透视报告或者查询表报告

20.从 Access 数据创建文本文件

你可以使用Excel的VBA过程从Access数据创建一个以逗号或者tab分开的文本文件。文本文件对于传输大量数据到电子表格特别有用。下面的示例程序示范了如何从一个ADO recordset创建一个tab分开的文本文件。为了确保该过程运行正确，你必须创建对Microsoft ActiveX Data Objects 2.6 Library的引用。参考第八章中操作文本文件的详细信息。运行该过程后，请在Excel里打开C:\ProductsOver50.txt

```
Sub CreateTextFile()
    Dim strPath As String
    Dim conn As New ADODB.Connection
    Dim rst As ADODB.Recordset
    Dim strData As String
    Dim strHeader As String
    Dim strSQL As String

    strPath = "C:\Program Files\Microsoft Office\" & "Office\Samples\Northwind.mdb"
    conn.Open "Provider=Microsoft.Jet.OLEDB.4.0;" & "Data Source=" & strPath & ";"
    conn.CursorLocation = adUseClient
    strSQL = "SELECT * FROM Products WHERE UnitPrice > 50"

    Set rst = conn.Execute(CommandText:=strSQL, Options:=adCmdText)
    'save the recordset as a tab-delimited file
    strData = rst.GetString(StringFormat:=adClipString, _
        ColumnDelimiter:=vbTab, _
        RowDelimiter:=vbCr, _
        nullExpr:=vbNullString)

    Open "C:\ProductsOver50.txt" For Output As #1
    For Each f In rst.Fields
```

```
        strHeader = strHeader + f.Name & vbTab  
Next  
Print #1, strHeader  
Print #1, strData
```

```
Close #1
```

```
End Sub
```

（译者：如果为强制要求声明对象则还需声明变量f。）在第八章中，你学习了如何使用FileSystemObject操作文本文件。下面的过程演示了如何使用该对象来创建一个名为 ProductsOver100.txt的文本文件：

```
Sub CreateTextFile2()
    Dim strPath As String
    Dim conn As New ADODB.Connection
    Dim rst As ADODB.Recordset
    Dim strData As String
    Dim strHeader As String
    Dim strSQL As String
    Dim fso As Object
    Dim myFile As Object

    Set fso = CreateObject("Scripting.FileSystemObject")
    Set myFile = fso.CreateTextFile("C:\ProductsOver100.txt", True)

    strPath = "C:\Program Files\Microsoft Office\" & _
        & "Office\Samples\Northwind.mdb"

    conn.Open "Provider=Microsoft.Jet.OLEDB.4.0;" & _
        & "Data Source=" & strPath & ";"

    conn.CursorLocation = adUseClient
    strSQL = "SELECT * FROM Products WHERE UnitPrice > 100"

    Set rst = conn.Execute(CommandText:=strSQL, Options:=adCmdText)

    'save the recordset as a tab-delimited file
    strData = rst.GetString(StringFormat:=adClipString, _
        ColumnDelimiter:=vbTab, _
        RowDelimiter:=vbCr, _
        nullExpr:=vbNullString)

    For Each f In rst.Fields
        strHeader = strHeader + f.Name & vbTab
    Next

    With myFile
        .WriteLine strHeader
        .WriteLine strData
        .Close
    End With
End Sub
```

（译者：如果为强制要求声明对象则还需声明变量f。）

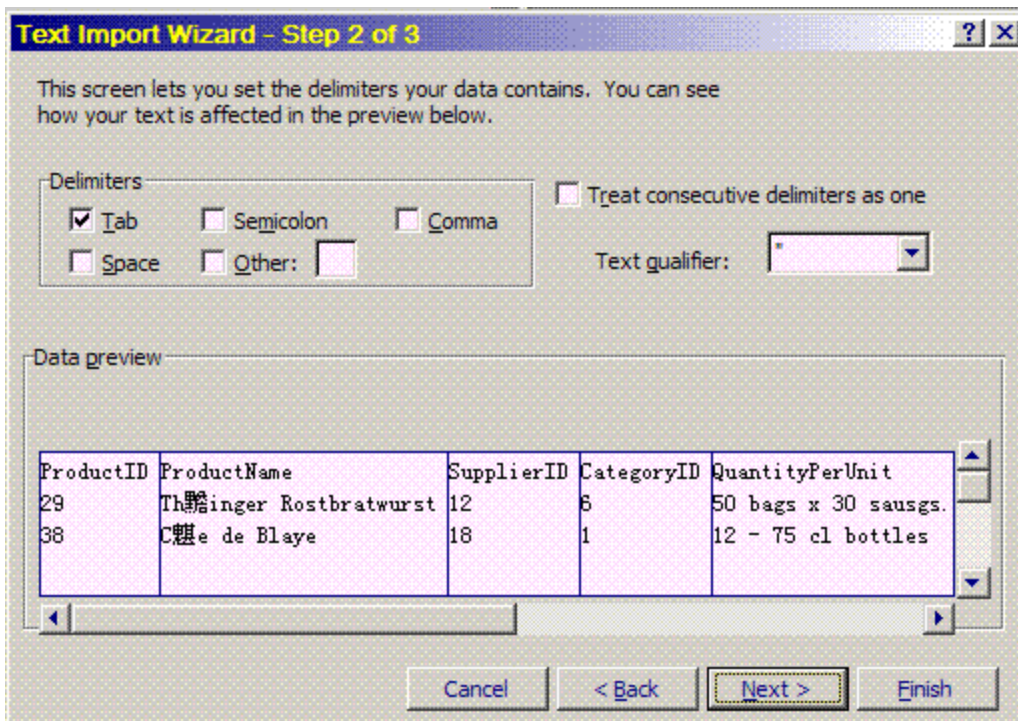


图 15-16 因为文本文件可以轻易地在Excel打开，所以你可以使用它在Access和Excel之间传输数据

21.从 Access 数据创建查询表

如果你想要在Excel使用来自外部的数据源而且你知道你将使用的数据会经常改变那么你需要创建一个查询表 查询表是Excel 工作表里的特殊表，它链接到外部数据源，例如Access数据库，SQL服务器，网页或者文本文件。用户可以轻易地刷新查询表 来获取最更新的信息。Excel提供了专门的菜单选项来获取外部数据：只要选择“数据”|“导入外部数据”，并选择“新数据库 查询”。通过查询外部数据库，你可以带来一些正好适合你要求得数据。例如，不必将所有的产品信息都带入你的电子表格来回 顾，你只要在获取数据之前明确数据必须达到的条件就行。因此，你可以只获取单价大于20美金的产品，而不是从Access导入 所有的产品。在VBA里，你可以使用QueryTable对象访问外部数据。每个QueryTable代表从外部数据源例如SQL服务器或者Access数据库创 建的工作表表格。要编程创建一个查询的话，你可以使用QueryTables集合对象的Add方法。该方法要求三个参数。本章结尾处 的示例过程使用下述语句在活动工作表上创建一个查询表：

Set myQryTable = ActiveSheet.QueryTables.Add(strConn, Dest, strSQL) strConn是为第一个参数——Connection提供数值的变量。它是必须的参数，为Variant数据类型，明确查询表数据源。 Dest是为第二个参数——Destination提供数值的变量。这是个必须的参数，为Range数据类型，明确在哪个单元格放置查询表。 strSQL是为第三个参数——SQL提供数值的变量。这是个必须的参数，为字符串数据类型，定义要从查询返回的数据。当你使用Add方法创建查询时，该查询不会运行，直到你调用Refresh方法。该方法接受一个参数——BackgroundQuery。这是一个Variant数据类型的可选参数，允许你决定是否在建立了对数据库的链接以及查询被提交（True）后将控制返回给过程，或者在查询已经运行并且所有数据已经获取到工作表里了（False）才将控制返回给过程。接下来的过程CreateQueryTable仅仅从Northwind数据库获取产品单价大于20的Products表中的产品。注意，该过程仅在所有相关记录都被获取之后，控制才交回给过程。方法RefreshStyle决定数据如何插入工作表。下述常数可供使用：

- xlOverwriteCells – 现存的单元格会被新数据覆盖
- xlInsertDeleteCells – 插入或者删除单元格以容纳新数据
- xlInsertEntireRows – 插入整行以容纳新数据

Sub CreateQueryTable()

Dim myQryTable As Object

Dim myDb As String

Dim strConn As String

Dim Dest As Range

Dim strSQL As String

myDb = "C:\Program Files\Microsoft Office\Office\"

& "Samples\Northwind.mdb"

strConn = "OLEDB;Provider=Microsoft.Jet.OLEDB.4.0;"

& "Data Source=" & myDb & ";"

```

Set Dest = Worksheets(1).Range("A1")
strSQL = "SELECT * FROM Products WHERE UnitPrice>20"
Set myQryTable = ActiveSheet.QueryTables.Add(strConn, _
    Dest, strSQL)
With myQryTable
    .RefreshStyle = xlInsertEntireRows
    .Refresh False
End With
End Sub

```

ProductID	ProductName	SupplierID	CategoryID	QuantityPerUnit	UnitPrice	UnitsInStock
4	Chef Anton's Cajun Seasoning	2	2	48 - 6 oz jars	22	53
5	Chef Anton's Gumbo Mix	2	2	36 boxes	21.35	0
6	Grandma's Boysenberry Spread	3	2	12 - 8 oz jars	25	120
7	Uncle Bob's Organic Dried Pears	3	7	12 - 1 lb pkgs.	30	15
8	Northwoods Cranberry Sauce	3	2	12 - 12 oz jars	40	6
9	Mishi Kobe Niku	4	6	18 - 500 g pkgs.	97	29
10	Ikura	4	8	12 - 200 ml jars	31	31
11	Queso Cabrales	5	4	1 kg pkg.	21	22
12	Queso Manchego La Pastora	5	4	10 - 500 g pkgs.	38	86
14	Tofu	6	7	40 - 100 g pkgs.	23.25	35
17	Alice Mutton	7	6	20 - 1 kg tins	39	0
18	Carnarvon Tigers	7	8	16 kg pkg.	62.5	42
20	Sir Rodney's Marmalade	8	3	30 gift boxes	81	40

图15-17 使用QueryTable对象可以在Excel里分析来自外部数据源例如Access数据库的数据

22.在 Excel 里使用 Access 数据

使用上面讨论过的方法之一从Access数据库获取数据之后，你可以使用许多Excel内置的工具来分析该数据。基于获取的信息来创建一些图表经常是很有用的。

23.用 Access 数据创建内嵌图表

使用VBA，你可以轻松的基于从Access数据库获取的数据创建图表。下面显示的ChartData过程使用从Access Northwind数据里获取的数据创建了一个内嵌图表。该图表由Charts集合的Add方法创建。图表的数据源由Range对象提供。CurrentRegion方法返回单元格A1周围的所有非空单元格。过程的剩余部分则通过设置图表的各种属性来设置图表格式。图表代码部分录制在一个分开的宏里，然后移到该VBA过程里并作一些修改以设置一些图表属性。

```

Sub ChartData()
    Dim db As DAO.Database
    Dim qd As DAO.QueryDef
    Dim rs As DAO.Recordset
    Dim mySheet As Worksheet
    Dim recArray As Variant
    Dim i As Integer
    Dim j As Integer
    Dim pathDb As String
    Dim qdName As String

    pathDb = "C:\Program Files\Microsoft Office\" & "Office\Samples\northwind.mdb"
    qdName = "Category Sales for 1997"
    Set db = OpenDatabase(pathDb)

    Set qd = db.QueryDefs(qdName)
    Set rs = qd.OpenRecordset

```

```
Set mySheet = Worksheets("Sheet2")
```

```
With mySheet.Range("A1")  
    .CurrentRegion.Clear
```

```

recArray = rs.GetRows(rs.RecordCount)
For i = 0 To UBound(recArray, 2)
    For j = 0 To UBound(recArray, 1)
        .Offset(i + 1, j) = recArray(j, i)
    Next j
Next i

For j = 0 To rs.Fields.Count - 1
    .Offset(0, j) = rs.Fields(j).Name
    .Offset(0, j).EntireColumn.AutoFit
Next j
End With
mySheet.Activate
Charts.Add
ActiveChart.ChartType = xl3DColumnClustered
ActiveChart.SetSourceData _
    Source:=mySheet.Cells(1, 1).CurrentRegion, PlotBy:=xlRows
ActiveChart.Location Where:=xlLocationAsObject, Name:=mySheet.Name

With ActiveChart
    .HasTitle = True
    .ChartTitle.Characters.Text = qdName
    .Axes(xlCategory).HasTitle = True
    .Axes(xlCategory).AxisTitle.Characters.Text = ""
    .Axes(xlSeries).HasTitle = False
    .Axes(xlValue).HasTitle = True
    .Axes(xlValue).AxisTitle.Characters.Text = mySheet.Range("B1") _
        & "($)"
    .Axes(xlValue).AxisTitle.Orientation = xlUpward
End With
db.Close
End Sub

```

过程ChartData的运行结果显示在图15-18上。

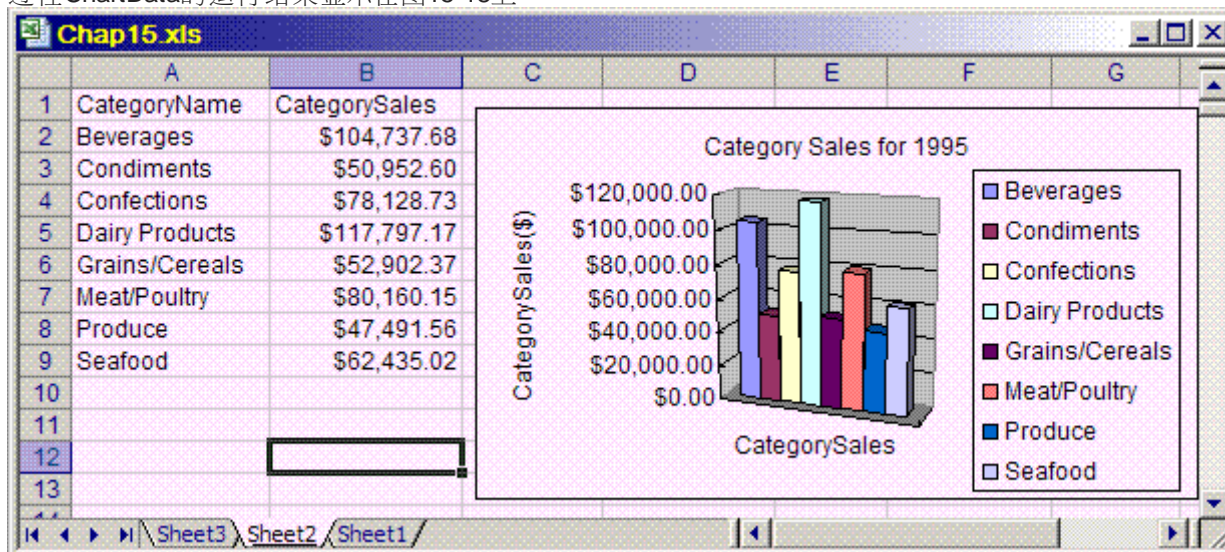


图15-18 你可以使用VBA编程基于从Access表，查询或者SQL语句获取的数据创建内嵌图表

24.传输 Excel 电子表格到 Access 数据库

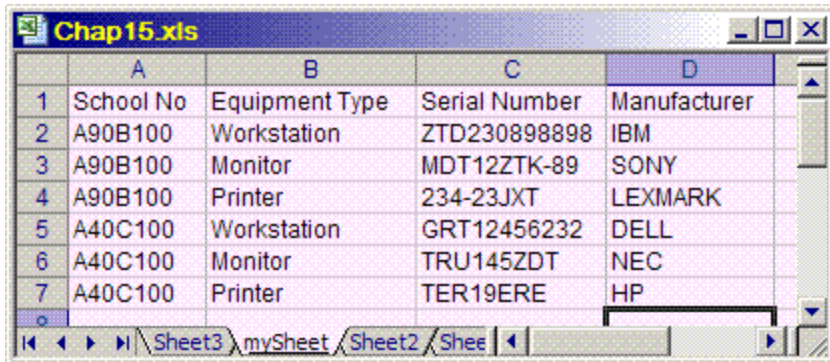
世界上许多大的数据库都是从电子表格开始的。当你需要从你的电子表格创建一个数据库应用软件时，你可以求助于缓慢呆滞的手动方法来传输数据。你也可以使用你新学的VBA编程技巧自动将你的电子表格变为数据库表。一旦在数据库格式了，你的Excel数据就可以使用在高级的公司范围的报告上使用，或者作为一个独立的应用软件使用（不必多言，后面的要求就是你要拥有数据库应用软件的设计技巧）。本章剩余的部分将示范如何将Excel电子表格链接和导入到Access数据库。在你移到Excel数据到Access

之前，你应该尽可能地整理好数据，这样，传输操作就会顺利。请牢记，电子表格里你要传输的每一行都会变为表中的一个记录，而每一列都会作为表的字段。因为这个原因，你计划传输到Access的电子表格的第一行应该包含字段名称。你要传输的数据列中间应该没有空白。换句话说，你的数据应该是连续的。如果你要传输的数据有大量的列，那么应该先打印出你的数

据并检查确保以后没有意外。如果你的数据的第一列为字段名称，那么建议你使用内置的Transpose函数先将数据转置，以确保数据是从上到下，而不是从左到右。让你数据顺利导入的关键是使你的电子表格尽可能象数据库表。

25.将 Excel 电子表格链接到 Access 数据库

你可以使用TransferSpreadsheet方法将Excel电子表格链接到Access数据库（参见本章“使用TransferSpreadsheet方法获取数据”部分有关该方法的使用详情）。下面的示例过程将图15-19显示的电子表格链接到Northwind数据库。在使用OpenCurrentDatabase方法打开Access数据库之后，该过程从Chap15.xls电子表格文件的mySheet工作表中位于单元格(A1:D7)区域里，使用Access DoCmd对象的TransferSpreadsheet方法创建了一个名为ExcelSheet的链接表。注意，DoCmd语句中的参数-1表明电子表格的第一行包含列标题。接着，该过程打开该链接的表为编辑模式，因此用户可以添加或者修改数据。如果在添加一条或者多条记录后，你要更改回到Excel，那么你将注意到在链接的Access表里作的改变将在Excel里立即可用。



	A	B	C	D
1	School No	Equipment Type	Serial Number	Manufacturer
2	A90B100	Workstation	ZTD230898898	IBM
3	A90B100	Monitor	MDT12ZTK-89	SONY
4	A90B100	Printer	234-23JXT	LEXMARK
5	A40C100	Workstation	GRT12456232	DELL
6	A40C100	Monitor	TRU145ZDT	NEC
7	A40C100	Printer	TER19ERE	HP

图 15-19 VBA过程LinkExcel_ToAccess将电子表格链接到Access的Northwind数据库

```
Sub LinkExcel_ToAccess()
    Dim objAccess As Access.Application
    Dim strName As String

    strName = "Linked_ExcelSheet"
    Set objAccess = New Access.Application
    With objAccess
        .OpenCurrentDatabase "C:\Program Files\Microsoft Office\" _
            & "Office\Samples\Northwind.mdb"
        .DoCmd.TransferSpreadsheet acLink, acSpreadsheetTypeExcel9, _
            strName, _
            "C:\Chap15.xls", _
            -1, "mySheet!A1:D7"
        .DoCmd.OpenTable strName, acViewNormal, acEdit
    End With
End Sub
```

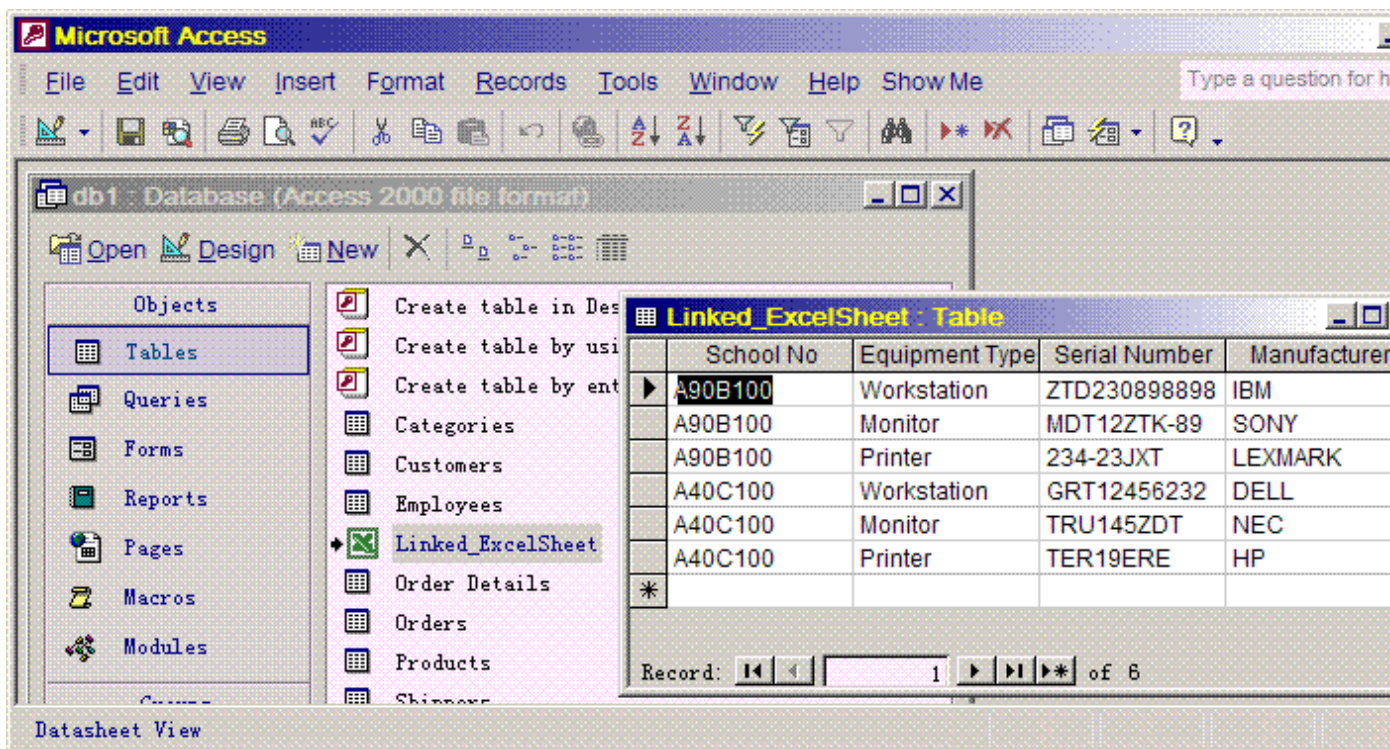



图15-20 Excel电子表格可以被链接到Access数据库

26.将 Excel 电子表格导入 Access 数据库

在前面的部分，你学习了如何将Excel电子表格链接到Access数据库。导入你的电子表格也是一样简单。你甚至可以使用用于链接的相同的VBA过程，作一些细微的改变：简单地将常数acLink改为acImport，就完工了。下述过程将图15-19里显示的电子表格（见前面部分）导入到Northwind数据库。

```
Sub ImportExcel_ToAccess()
    Dim objAccess As Access.Application
    Dim strName As String

    strName = "Imported_ExcelSheet"
    Set objAccess = New Access.Application
    With objAccess
        .OpenCurrentDatabase "C:\Program Files\Microsoft Office\" _
            & "Office\Samples\Northwind.mdb"
        .DoCmd.TransferSpreadsheet acImport, acSpreadsheetTypeExcel9, _
            strName, _
            "C:\Chap15.xls", _
            -1, "mySheet!A1:D7"
        .DoCmd.OpenTable strName, acViewNormal, acEdit
    End With
End Sub
```

27.放置 Excel 数据到 Access 表中

除了链接或者内嵌你的Excel电子表格，你万一想要从头开始创建一个Access表并且装载电子表格里面的数据呢？使用一些你已经在本书里获得的编程技巧，你可以轻松完成该任务。我们来看看一个VBA过程，基于图15-19显示的Excel电子表格，动态地创建一个Access表。注意，该过程使用ADO和Microsoft Jet.OLEDB.4.0提供者链接到Access数据库。创建链接之后，该过程使用ADOX对象库里的Catalog和Table对象创建一个新Access表。接着，对应电子表格列名称的字段被添加到该表中。注意，每个文本字段明确了它可以接受的最大字符数。如果电子表格单元格的长度大于该字段大小，那么错误处理程序将显示Access内置的信息，提示该错误并终止过程。该过程的最后一个任务是数据传输操作。要执行该任务，该过程打开了Access表的Recordset对象，因为你需要添加记录到该表，所以该过程使用了一个adOpenKeyset指针类型。现在该表已打开，该过程使用For...Next循环在Excel数据行中循环，将在每个单元格中找到的信息放置到相应的表字段中。注意，使用Recordset对象的AddNew方法往Access表里添加新的记录。从每行单

元格复制完数据之后，该过程使用Recordset对象的Update方法来保存该表记录。

```
Sub AccessTbl_From_ExcelData()  
    Dim conn As ADODB.Connection  
    Dim cat As ADOX.Catalog  
    Dim myTbl As ADOX.Table  
    Dim rstAccess As ADODB.Recordset  
    Dim rowCount As Integer  
    Dim i As Integer  
  
    On Error GoTo ErrorHandler  
  
    ' connect to Access using ADO  
    Set conn = New ADODB.Connection  
    conn.Open "Provider = Microsoft.Jet.OLEDB.4.0;" & _  
        "Data Source = C:\Program Files\Microsoft Office\Office\Samples\Northwind.mdb;"  
  
    ' create an empty Access table  
    Set cat = New Catalog  
    cat.ActiveConnection = conn  
    Set myTbl = New ADOX.Table  
    myTbl.Name = "TableFromExcel"  
    cat.Tables.Append myTbl  
  
    ' add fields (columns) to the table  
    With myTbl.Columns  
        .Append "School No", adVarChar, 7  
        .Append "Equipment Type", adVarChar, 15  
        .Append "Serial Number", adVarChar, 15  
        .Append "Manufacturer", adVarChar, 20  
    End With  
  
    Set cat = Nothing  
    MsgBox "The table structure was created."  
  
    ' open a recordset based on the newly created  
    ' Access table  
    Set rstAccess = New ADODB.Recordset  
    With rstAccess  
        .ActiveConnection = conn  
        .CursorType = adOpenKeyset  
        .LockType = adLockOptimistic  
        .Open myTbl.Name  
    End With  
  
    ' now transfer data from Excel spreadsheet range  
    With Worksheets("mySheet")  
        rowCount = Range("A2:D7").Rows.Count  
        For i = 2 To rowCount + 1  
            With rstAccess .AddNew ' add a new record to an Access table  
                .Fields("School No") = Cells(i, 1).Text  
                .Fields("Equipment Type") = Cells(i, 2).Value  
                .Fields("Serial Number") = Cells(i, 3).Value  
                .Fields("Manufacturer") = Cells(i, 4).Value  
                .Update ' update the table record  
            End With  
        Next i  
    End With  
  
    ' close the Recordset and Connection object and remove them  
    ' from memory  
    rstAccess.Close  
    conn.Close  
    Set rstAccess = Nothing  
    Set conn = Nothing
```

```
AccessTbl_From_ExcelDataExit:
```

```
Exit Sub
```

```
ErrorHandler:
```

```
MsgBox Err.Number & ": " & Err.Description
```

```
Resume AccessTbl_From_ExcelDataExit
```

```
End Sub
```

28. 接下来……

本章提供了多个例子将Excel数据放入Access，以及从Access获取数据到工作表。你学习了如何从Excel VBA过程中 控制Access应用软件，执行一些任务，例如打开Access窗体和报告，创建新窗体，运行选择和参数查询，以及调用Access内置 函数。另外，本章示范了一些创建文本文件，查询表，和图表的技术。你也学习了如何使用链接，导入和动态Access表，将Excel 数据放置到Access数据库里。在下章中，你将学习如何使用Excel创建，察看和分析因特网数据。