3. VBA 初步

3.1. VBA 简介

VBA 是 Visual Basic for Application 的缩写，是一种应用程序自动化语言。所谓应用程序自动化，是指通过程序或者脚本让应用程序，例如 Microsoft Excel、Word 自动化完成一些工作，例如在 Excel 里自动设置单元格的格式、给单元格充填某些内容、自动计算等。在前面的章节里，我们通过使用宏来设置 Excel 单元格的格式、填充单元格的内容，而使宏完成这些工作的正是 VBA。

VBA 的由来

在 20 世纪 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，可以使用 Excel 的宏语言使 Excel 自动化，使用 Word Basic 使 Microsoft Word 自动化，等等。因此，Microsoftt 决定开发一种应用程序共享的通用自动化语言 VBA，这就是 Visual Basic for Application（VBA）的由来。Visual Basic for Application 可以看做是非常流行的应用程序开发语言 Visual Basic 的一个子集，Visual Basic 其他子集还包括 VB Script 等。

VBA 具有 VB 语言的大多数特征和易用性，它最大特点就是将 Excel 作为开发平台来开发应用程序，可以应用 Excel 的所有已有功能，例如数据处理、图表绘制、数据库连接、内置函数等等。

本部分将对 VBA 及其开发环境 IDE（集成开发环境）、VBA 的基本语法、应用 VBA 自动化 Excel 做一简单介绍。文中会涉及到一些诸如对象、事件等部分读者可能不熟悉或不清楚的概念，对于此类问题可直接忽略之，因为在后面会有详细介绍。本部分也不是一个 VBA 的参考文档，只是其语法、特征的快速浏览和介绍。

学习完本部分的内容后，读者应该可以应用 Excel 和 VBA 进行一些简单的开发。

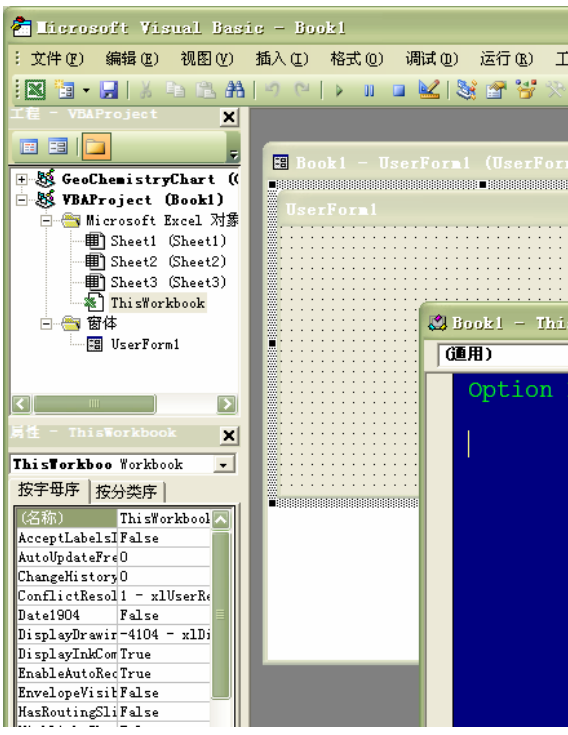
3.2. VBA 快速入门

3.2.1. VBA 开发环境

VBA 集成开发环境（IDE，Integrated Development Environment[3]的缩写）是进行 VBA 程序设计和代码编写的地方，同一版本的 Office 共享同一 IDE。VBA 代码和 Excel 文件是保存在一起的，可以通过点击“工具 — 宏 — Visual Basic 编辑器”打开 VBA 的 IDE 环境（图 3-1），进行程序设计和代码编写。

打开 VBA IDE 的方法：

- 通过“工具 — 宏 — VISUAL BASIC 编辑器”
- 通过快捷键“ALT + F11”
- 右键单击工具栏，选择“Visual Basic”，此工具栏有录制宏，打开 VBA IDE 等的快捷按钮：



[3] 翻译为中文即“集成开发环境”。

图 3-1 Visual Basic IDE 环境

图 3-1 为 Excel VBA 的 IDE 环境，对于所有使用同一版本 VBA 的应用程序，都共享相同的 IDE 环境。对于同一程序，例如 Excel，不管你打开几个 Excel 文件，但启动的 VBA 的 IDE 环境只有一个。缺省情况下，VBA IDE 环境上方为菜单和工具条（图 3-1），左侧上方窗口为工程资源管理器窗口，资源管理器窗口之下为属性窗口，右侧最大的窗口为代码窗口。

在资源管理器窗口可以看的所有打开和加载的 Excel 文件及其加载宏。每一个 Excel 文件，在 VBA 下，称为一个工程，如果你同时打开了多个 Excel 文件，则在 VBA IDE 下可以看到有多个工程存在。

每个 Excel 文件（工作薄）对应的 VBA 工程都有 4 类对象（图 3-2）。，包括：

- Microsoft Excel 对象
- 窗体
- 模块
- 类模块

Microsoft Excel 对象代表了 Excel 文件及其包括的工作薄和工作表等几个对象，包括所有的 Sheet 和一个 Workbook，分别表示文件（工作薄）中所有的工作表（包括图表），例如缺省情况下，Excel 文件包括 3 个 Sheet，在资源管理器窗口就包括 3 个 Sheet，名字分别是各 Sheet 的名字。ThisWorkbook 代表当前 Excel 文件。双击这些对象会打开代码窗口（图 3-1 右侧窗口），在此窗口中可输入相关的代码，响应工作薄或者文件的一些事件，例如文件的打开、关闭，工作薄的激活、内容修改、选择等（有关事件、Excel 对象模型见后）。

窗体对象代表了自定义对话框或界面，例如你编写了一个 VBA 求个人所得税的小程序，需要输入税率、收入等参数，那么就可以使用窗体设计一个对话框，来获取用户输入。

模块是自定义代码，包括我们录制的宏等 VBA 代码保存的地方。

类模块是以类或对象的方式编写的代码保存的地方。通过创建类模块，在 VBA 中也可以创建自己的类和对象，我们在“VBA 语言”一章会详细介绍 VBA 的面向对象编程的问题。

关于各对象的具体含义和使用后面会详细介绍，这里只是做个最初的了解。在工程资源管理器窗口的右键菜单下，有添加用户窗体、模块、类模块的选项，也可以将已有的模

块移除、导入和导出。在工程资源管理器之下，也可以通过将一个工程中的模块用鼠标拖拽到另一个工程实现模块在工程之间的拷贝。

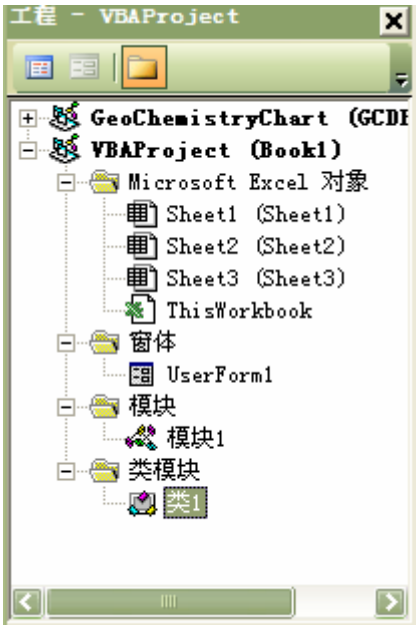


图 3-2 VBA 工程资源管理器窗口
包括 2 个 VBA 工程

建议随时更改 Excel VBA 工程的名称，其缺省名称为“VBAProject”，可以通过选中工程，在属性窗口更改为有意义的名称，或者在菜单的“工具 － VBAProject 属性”对话框中更改。

在 VBA 工程资源管理器之下是属性窗口（图 3-3），主要用于对象属性的交互式设计和定义，例如选中图 3-2 中的 VBAProject，在属性窗口即可更改其名称。属性窗口除了更改工程、各对象、模块的基本属性外，主要用途是用户窗体（自定义对话框）的交互式设计。图 3-3 显示的就是一个打开的窗体（UserForm）的属性窗口。

图 3-3 VBA 属性窗口

在 IDE 窗口的右侧，可以打开代码窗口。在资源管理器窗口中的每一个对象会对应一个代码窗口（用户窗体包括一个设计窗口和一个代码窗口）。可以通过在对象上双击、在右键菜单或资源管理器工具栏上选择查看代码（或对象）打开代码窗口。

对于 IDE 环境、菜单、工具栏的具体使用和说明，在后面的讲解中会逐步讲解说明。

单击“**视图 － 对象浏览器**”或工具栏上的“**对象浏览器**”按钮即可打开对象浏览器窗口（图 3-4），在此窗口内可查看当前工程及其引用对象的属性、方法和事件。对象浏览器对于熟悉和查看相应的 Excel 对象、引用对象（包括 COM 对象、其他 Excel 程序）所包含的类、属性、方法和事件非常有用，特别是在没有相应的帮助资料或者文档的情况下，对象浏览器是查看一个对象的内容的最有效的工具。

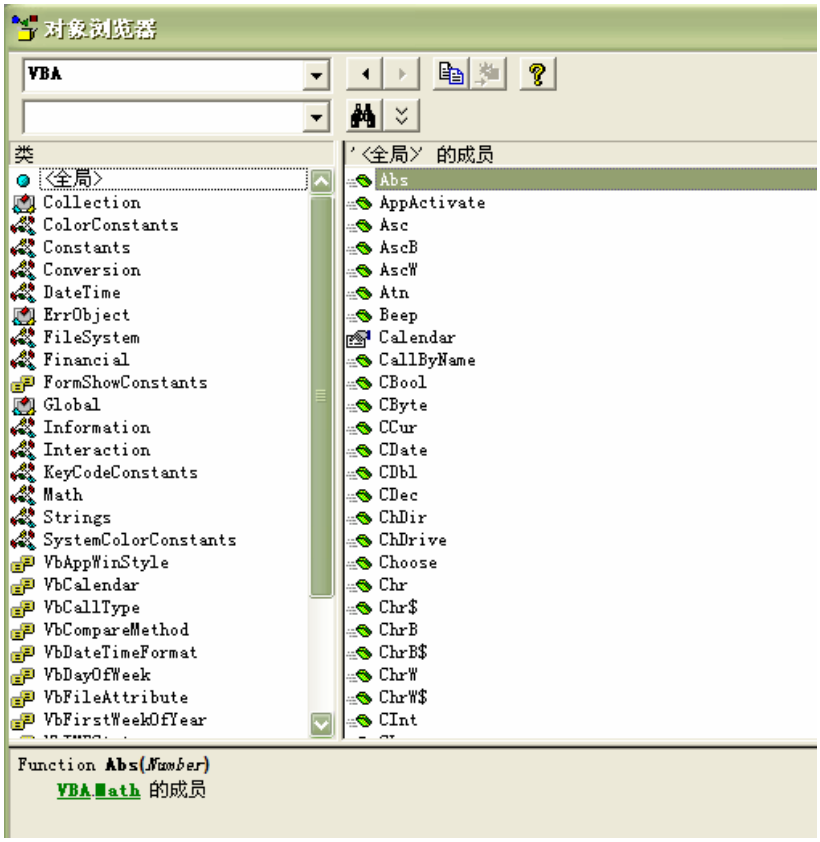


图 3-4 VBA IDE 环境的对象浏览器

上部可以选择对象，也可以输入一些查询条件；左侧为对象列表；右侧是对象的成员，包括属性、方法和事件；下部为成员或者对象的说明

3.2.2. 在 VBA IDE 下进行开发

熟悉了 VBA 的 IDE 环境后，我们来开发 VBA 之旅的第一个程序。新建一个 Excel 文件，通过菜单或键盘快捷键打开 VBA 集成开发环境，在 VBAProject 上单击右键，选择“插入 － 模块”。这样，系统将打开一个代码窗口，在窗口中输入以下代码[4]。

```
#001 Sub MyFirstVBAProgram()
#002     Dim strName As String
#003     Dim strHello As String
#004     strName = InputBox("请输入你的名字：")
#005     strHello = "你好，" & strName & "!"
#006     MsgBox strHello
#007 End Sub
```

将鼠标光标放置在这段代码之内，单击菜单“运行 － 运行子过程/用户窗体”，或者

[4] 代码内的“#003”为行号，在实际代码中不能输入，在此只为文中叙述方便，之后不再重复。

在工具栏单击运行按钮，则可运行这段代码。运行结果会显示一个对话框，输入一些内容后，会显示相应的问候语。同样，这段代码可以和宏一样，在 Excel 下选择并执行。

与其他程序设计语言不同，VBA 程序是事件驱动的，没有 Main 函数之类的入口的概念。如果在 IDE 环境下，鼠标光标不在任何过程内，单击工具栏或运行菜单的运行，会显示一个对话框，要你选择要运行的过程。

本质上，VBA 代码应该只是一些完成具体工作的集合，而通过界面元素或者 Excel 的事件驱动执行，你可以通过自定义按钮、菜单，并指定一个宏（VBA 过程，自定义界面也可以通过编程手段完成此类工作），通过单击此按钮即可调用相应的 VBA 代码，或者将调用绑定在 Excel 的某个事件下。

下面我们简单看一下上面这段代码的组成，代码第 1 行表示这是一个新的过程，名称为“MyFirstVBAProgram”，第 2、3 行定义了 2 个变量，其类型为字符串类型，第 4 行调用 InputBox 这个内置函数，并将返回值赋给 strName 这个变量，第 5 行将几个字符串组合成一个新的字符串，第 6 行调用 MsgBox 这个函数，显示一个对话框，第 7 行表示过程结束。VBA 程序由不同的模块组成，在模块内部，可以定义不同的变量、过程或函数，由此组成一个完整的程序。

代码窗口的设置

中文环境下 VBA IDE 代码窗口缺省的设置比较糟糕，字体为宋体，大小是 9 磅，使用不很方便，可以在“工具 － 选项”对话框下的“编辑器格式”页内设置代码窗口字体、颜色、背景。字体建议使用 Courier New，大小可以按照自己的分辨率设置标准以阅读清楚为宜。

在此模块内，再新建一段代码：

```
#001 Function MyAdd(varA, varB) As Variant
#002     MyAdd = varA + varB
#003 End Function
```

此段代码非常简单，只有 3 行，第 1 行表示这是一个函数，具有 2 个参数 varA,和 varB，函数与过程的差别在于函数有返回值，第 2 行将参数 varA,和 varB 的和赋给函数，代表其返回值。函数无法直接运行，必须从工作表或者其他程序调用，例如，我们可以写以下一段简单的程序调用此函数：

```
#001 Sub TestAdd()
```



```
#002     Dim a, b, c
#003     a = 12
#004     b = 34
#005     c = MyAdd(a, b)
#006     MsgBox c
#007 End Sub
```

其中第 5 行为函数 MyAdd 的调用，函数将返回值赋给 c。需要说明的是，VBA 中，调用过程可以使用 Call 语句，也可省略，调用过程时，其参数的括号可以省略，但调用函数必须有括号。

也可以直接在工作表内使用自定义的函数，例如在工作表中，我们可以和 Excel 内置函数一样使用自定义的函数（图 3-5），Excel 会负责参数传递，将返回值赋给相应的单元格，在引用参数改变时会自动重新计算，总之，与内置函数的使用没有什么不同。

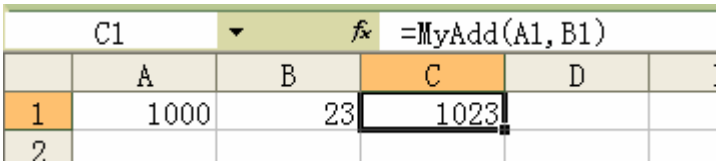


图 3-5 在工作表中使用自定义函数

以上通过 2 个 例子简单介绍了 VBA 编程的过程和概念，后面我们将正式进入 VBA 编程之旅，逐步讲解模块、函数与过程、数据类型、基本语法等概念。

VBA 程序的保存

当关闭 VBA IDE 的时候，不会提示保存用户所做的修改，当我们退出 Excel 保存其文件时，VBA 程序代码也随之保存，因为 VBA 代码是寄生于 Excel 或其他文档的，保存文档即保存了 VBA 代码。

对于一般的 VBA 程序，这点没有什么问题，但如果是在进行加载宏的开发，那么退出 IDE 环境时，一定要点击其文件菜单或工具栏的保存，保存所做修改，否则会丢失所做的任何修改。

3.2.3. 模块、过程和函数

VBA 代码必须存放在某个位置，这个地方就是**模块**。**模块**是作为一个单元保存在一起的 VBA 定义和过程的集合。VBA 中有两种基本类型的模块：标准模块和类模块。模块可以包括 2 类子程序：过程或者函数。

一般来说，简单的 VBA 程序设计，大部分工作集中在标准模块中（后面简称为模块）。当 Excel 在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。可以在工程资源管理器上单击右键来增加删除模块（图 3-2），选中模块后可以在属性窗口修改其名称（图 3-3）。

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 VBA 语句。一个工作簿的每个过程都有唯一的名字加以区分。过程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，所看到的就是过程。下面就是一个简单的修改选中文字的字体的过程。

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

将这个过程输入到 VBA 中的某个过程，返回 Excel，在某个单元格输入文字，选中他，然后选择“工具 — 宏 — 宏”，在宏对话框中选择“ChangeFont”，然后执行他，我们会看到选中的单元格文字格式被更改为“Arial”，16 点大小。

函数过程通常情况下称为**函数**，是会返回一个值的过程。函数和过程的差别是其定义方式不同，函数使用 VBA 的关键字 Function 定义，而过程使用关键字 Sub 定义。函数返回的值称为返回值，这个数值通常是计算的结果或是测试的结果。我们可以用 VBA 创建自定义函数，并且在工作表上使用所创建的函数。以下程序是一个计算价格的 10%为运费的简单例子。

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

这个函数使用一个参数（Price），过程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。函数计算运费，计算结果在函数中通过赋给函数名“Shipping”来返回给调用者。Price 可以是数字和单元格引用。这个函数可以被其他过程或者函数调用，也可以使用在电子表格中。例如图 3-5 和图 3-6 所示。

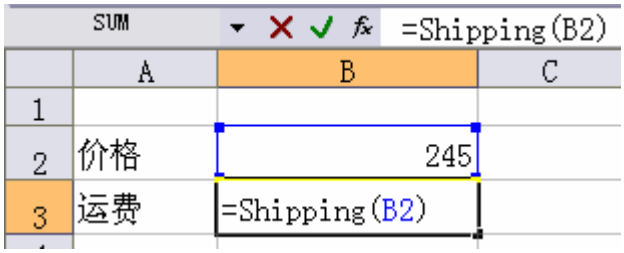


图 3-6 在工作表中使用函数 Shipping

3.2.4. 创建过程和函数

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不同的过程和函数。对于一个应用程序，可以使用一个模块，也可以使用多个模块。如果你的程序比较复杂，使用多个模块可以更好的组织代码。

下面我们从头开始创建一个只显示一个消息框的过程，来学习过程的创建，请按照以下步骤进行：

（1）打开一个新工作簿，进入 Visual Basic 编辑器，也即 VBA IDE；

（2）在`VBA IDE 的左面的“工程资源管理器”窗口 VBA 工程上单击鼠标右键，选择“**插入 － 模块**”，这样就将一个模块添加到应用程序中了；

（3）从菜单选择“**插入 － 过程**”，显示“添加过程”对话框；

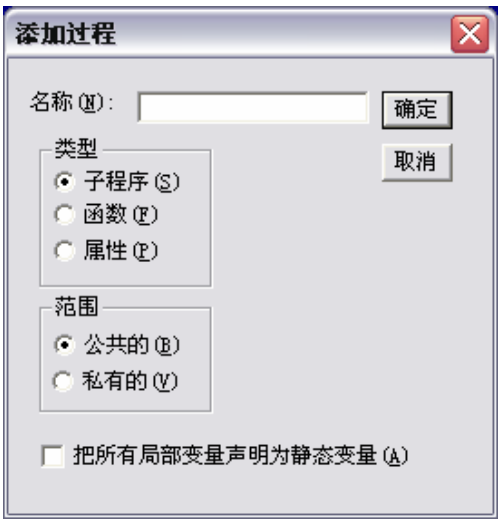


图 3-7 添加过程对话框

（4）输入“HelloMsg”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”按钮继续；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下代码：


```
Public Sub HelloMsg()

End Sub
```

（5）在过程中输入以下语句：

```
MsgBox "这是我的第一个过程"
```

在输入 MsgBox 后，会自动弹出一个有关这条命令的信息，称之为即时代码提示或自动列表技术。输入后的过程如下所示：

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

创建这个过程后，可以运行一下。运行一个过程有几种方法：将鼠标光标放置在这段代码之内，单击菜单“**运行 － 运行子过程/用户窗体**”，或者在工具栏单击运行按钮，则可运行这段代码。

运行后，此过程执行并显示一个消息框；单击消息框之中的“确定”按钮，关闭该消息框，程序运行结束。

图 3-8 过程运行结果

要保存新过程，需要保存过程所在的工作簿。可以在 VBA IDE 环境下选择“**文件 － 保存工作簿**”保存，也可以在 Excel 环境下保存。

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。

在过程或者函数中，可以调用其他的过程或者函数，例如：

```
#001  Function Add(a, b)
#002 ‘计算 a 和 b 的加法和
#003      Add = a + b
#004  End Function
#005
#006  Sub Display()
```



```
#007     Dim sum
#008     sum = Add(2, 5)
#009     MsgBox sum
#010  End Sub
```

函数 Add 计算 2 个数值的值，并返回他，过程 Display 定义了 sum 这个变量，调用了 Add 函数计算（8 行），最后调用 MsgBox 命令显示结果（9 行）。在第二行，我们看到有一句“计算 a 和 b 的加法和”的说明，称为注释，是对程序进行说明的语句，在执行过程中并不执行。注释可以更好的帮助我们阅读、修改代码，对于一段半个月前写的代码，你重新打开需要修改，经常会忘了程序的功能和流程，这时候，就需要注释帮助我们。VBA 中，注释的格式是在一句话之前使用单引号“’”来表示注释，注释可以位于程序的任意地方。

3.2.5. 变量

变量是用于保存数据的地方，每个变量都需要使用一个名字来表示，称为变量名。在程序设计中，我们把需要使用的数据，例如数值、字符串、Excel 的对象保存在变量中。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。我们回到前边学习过的一段代码，来说明为什么要使用变量。

```
#001 Sub MyFirstVBAProgram()
#002    Dim strName As String
#003    Dim strHello As String
#004    strName = InputBox("请输入你的名字：")
#005    strHello = "你好，" & strName & "!"
#006    MsgBox strHello
#007 End Sub
```

这段代码通过 InputBox 这个命令显示一个对话框，输入内容后，通过 MsgBox 显示相应的问候语。那么其中的 strName 和 strHello 就是变量，如果没有 strName，那么在 InputBox 中输入的名字就会丢失，因此需要创建一个变量，以保存程序运行过程中的一些中间结果。

上面的例子里，定义变量后面的“As String”定义了变量的数据类型。和 Excel 单元格里的数据一样，VBA 里的变量也需要有数据类型。表 3-1 列出了 VBA 支持的部分常用数据类型以及各种类型的变量能够存储的数值范围。

表 3-1 VBA 支持的部分数据类型

数据类型	范围
Byte	0 到 255
Boolean	True 或 False

Integer	-32,768 到 32,767
Long (长整型)	-2,147,483,648 到 2,147,483,647
Single (单精度浮点型)	负数时从 -3.402823E38 到 -1.401298E-45；正数时从 1.401298E-45 到 3.402823E38
Double (双精度浮点型)	负数时从 -1.79769313486231E308 到 -4.94065645841247E-324；正数时从 4.94065645841247E-324 到 1.79769313486232E308
Date	100 年 1 月 1 日 到 9999 年 12 月 31 日
String (变长)	0 到大约 20 亿
Variant (数字)	任何数字值，最大可达 Double 的范围
Variant (字符)	与变长 String 有相同的范围

以上数据类型是 VBA 支持的部分数据类型（不是全部），其中 Byte、Integer、Long 都用来表示不同大小的整数；Boolean 表示判断，即对（True）还是错（False）；Single 和 Double 表示不同大小范围的实数；Date 用来表示日期和时间；String 用来表示一个在发出；Variant 是一个特殊的数据类型，可以表示任意类型的数据，对于没有定义数据类型的变量，默认为 Variant 类型。在程序设计中，我们应该根据需要选择合适的数据类型。

以下我们将创建变量。创建变量可以使用 Dim 语句，创建变量通也称为“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。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，也不能包含这些字符：@ & $ #，名字最大长度为 255 个字符。这条语句中的数据类型部分可以是表 3-1 中的任何一种数据类型。

例如在上面的程序里，我们创建了 2 个 String 类型的变量，用来存储我们输入的名字：

```
Dim strName As String
Dim strHello As String
```

语句“strHello = "你好，" & strName & "!"”中，“&”表示连接 2 个字符串，我们也可以使用“+”来连接 2 个字符串，但推荐使用“&”，以避免和一般的加法运算混淆。

我们再看一个例子：

```
Sub Add2Number()
```



```
    Dim i As Integer
    Dim j As Integer
    Dim sum As Integer
    i = InputBox("请输入 i：")
    j = InputBox("请输入 j：")
    sum = i + j
    MsgBox "i+j=" & sum
End Sub
```

这个例子里我们定义了 3 个 Integer 类型的变量，使用 InputBox 给 i 和 j 赋值，然后通过计算得到结果 sum，最后使用 MsgBox 输出结果。

VBA 的整型变量（Byte，Integer，Long）和浮点类型变量（Single 和 Double）可以进行加、减、乘、除、乘方等数学运算，使用的算术运算符分别为：

1. ^ 运算符：求一个数字的某次方，如 A^B；
2. * 运算符：乘法运算；
3. / 运算符：除法运算；
4. \ 运算符：对两个数作除法并返回一个整数；
5. Mod 运算符：求两数的余数；
6. + 运算符：加法运算；
7. - 运算符：减法运算。

例如：

```
k = i + j
k = i * j
j = 2 + i ^ 3
```

VBA 的算术运算符的优先级和数学中一样，例如“2 + i ^ 3”会先计算 i 的 3 次方，然后再进行加法。

前面说过，如果在 Dim 语句中不提供数据类型，变量将被定义为 Variant 类型，这时，这个变量可以根据程序的上下文存储任意的数据类型。因为 VBA 中默认的数据类型是 Variant。使用 Variant 类型的优势是简单，这也正是 VBA 容易学习的原因之一。我们在后面的学习中会再返回到这个话题，一般来说，我们在模块、过程和函数内定义变量的时候，都应该指定其数据类型；但如果读者对变量类型感到迷惑，那么可以使用如下方式来定义变量，而不指定数据类型：

```
Dim Var
```

对于函数的参数和返回值，可以使用以下方法来指定参数，例如：


```
Function Add(a As Double, b As Double) As Double
    Add = a + b
End Function
```

这个函数第一行的 a 后面的“As Double”，指定 a 的数据类型为浮点型，b 后面的类似，最后的“As Double”指定了函数返回值的数据类型。

3.2.6. 程序流程

VBA 程序是由语句组成的，语句的定义为可表达一种动作、声明或定义的程序单元。一般来说，每个语句应该占一行，表示要进行的操作，例如进行计算、输入输出等等。

程序设计语言的语句的组织形式可以划分为 3 种：顺序语句、条件语句和循环语句。顺序语句、条件语句和循环语句可以互相组合嵌套，从而完成复杂的程序。

（1）顺序语句，即按照语句的顺序来执行的，例如下面的例子：

```
Sub Sequence()
    Dim i, j
    Dim sum
    i = 1
    j = 3
    sum = i + j
    MsgBox sum
End Sub
```

程序首先对 i 和 j 赋值，然后计算其结果，最后使用 MsgBox 显示结果。程序执行的过程是按照语句的顺序来进行的。

（2）条件语句，即根据某种条件判断来决定执行的顺序，而不是顺序执行，我们留在后面详细介绍。

（3）循环语句，即根据某些条件，对某条或某段语句循环反复执行数词，称为循环语句，例如有时候需要对某些值进行求和，如果这些数值有成千上万条，那么就需要通过循环语句来累加。我们后面会详细介绍。

3.2.7. 条件语句

我们先看一个例子：

```
#001    If  T < 2 Then
#002        Target = 200
#003    ElseIf  T >= 2 And T < 3 Then
```



```
#004       Target = 300
#005    Else
#006       Target = 500
#007    End If
```

这是个完整的 If 逻辑判断式，意思是说，假如 If 后的判断条件成立的话，就执行第 2 行语句，否则假如 ElseIf 后的判断条件成立的话，就执行第 4 行语句，Else 的意思是说，假如以上条件都不成立的话，就执行第 6 行语句。

条件判断是在程序设计中经常会碰到的，例如根据用户的输入来选择不同的操作，或者根据某段程序计算结果的不同而采取不同的操作。一般来说，判断语句是类似这样的形式：**如果**一件事情成立，**那么**我们应该这样做，**否则**我们那样做。例如，如果下雨了，那么出门就要带把伞，否则就不用带。实际生活中的如果、那么转换为 VBA 程序设计语言就是 If、Then、ElseIf、Else 这些关键词。

下面我们通过例子来说明判断语句的使用方法，例如下面整个例子：

```
#001 Sub IfTest()
#002    If Application.ActiveCell = 23 Then
#003       MsgBox "单元格是 23"
#004    Else
#005       MsgBox "单元格不是 23"
#006    End If
#007 End Sub
```

这段代码判断当前活动单元格的值是不是 23，然后弹出一个对话框来反馈结果。第 2 行的 Application.ActiveCell 是 Excel 对象，表示当前活动单元格，我们下一节将会介绍相关的内容。这句话判断这个值是否 23，最后的 Then 是 VBA 关键字，之后的语句在上面的判断成立的情况下执行；如果判断结果为假，则执行 Else 之后的语句。最后要以 End If 来结束判断语句。

这样，我们可以看到判断语句的标准形式是以下形式：

```
#001 If 条件 1 Then
#002     ......
#003 ElseIf 条件 2 Then
#004     ......
#005 Else
#006     ......
#007 End If
```

在进行条件判断时需要使用比较运算符来进行判断，常用的比较运算符有：

- =，判断左右两边的值是否是否相等；

- <，判断左侧的值是否小于右侧的值；
- >，判断左侧的值是否大于右侧的值；
- <=，判断左侧的值是否小于等于右侧的值；
- >=，判断左侧的值是否大于等于右侧的值；

在进行条件判断时，经常需要判断多个条件，例如判断某个值是否在 10 和 20 之间（10 < x < 20），那么我们就需要使用逻辑运算符。例如下面的例子，判断一个数值是否在 2 和 15 之间，我们不能这样做：

```
If 2 < temp < 15 Then
   ......
End If
```

以上写法在数学上是这样的，但在程序设计语言里，是不正确的，而应该写作：

```
If temp > 2 And temp < 15 Then
   ......
End If
```

这里，我们使用了 And 这个关键字，这是一个 VBA 逻辑运算符，表示如果左边和右边的语句都是正确的话，那么这个语句就返回 True。这是没有学习过程序设计语言的用户经常会犯的一个错误。

所以，判断语句中当需要进行多个条件判断时，会使用到以下逻辑运算符。

- And 运算符：通过“result = expression1 And expression2”使用，如果两个表达式的值都是 True，则 result 是 True。如果其中一个表达式的值是 False，则 result 是 False。
- Not 运算符：通过“result = Not expression”使用，如果 expression 是 True，则返回 False，否则返回 True。
- Or 运算符：使用同 And 运算符，如果两个表达式的值有一个是 True，则 result 是 True。如果两个表达式的值都是 False，则 result 是 False。

我们来看几个简单的使用逻辑运算符的例子：

```
If x = 1 And y > 5 Then
   MsgBox "x=1 和 y>5 都成立！"
Endif
```

这段代码使用了 And 逻辑运算符，只有 x 等于 1 和 y 大于 5 都成立的时候下，才会弹出 MsgBox 对话框。下面的例子使用了 Or 运算符，所以只要 x 等于 1 和 y 大于 5 这 2 个条件有一个满足，结果就是 True，就会显示 MsgBox 对话框：


```
If x = 1 Or y > 5 Then
    MsgBox " x=1 和 y>5 至少有一个成立！"
End If
```

如果读者对逻辑运算符还有什么疑惑的话，可以输入这 2 个例子到 VBA 代码编辑器中，改改条件，自己试试就会明白了。

我们接下来看一个比较复杂的例子：

```
Function Degree(f As Double) As String
    If f < 60 Then
        Degree = "不及格"
    ElseIf f >= 60 < 70 Then
        Degree = "及格"
    ElseIf f >= 70 And f < 85 Then
        Degree = "良好"
    ElseIf f >= 85 And f <= 100 Then
        Degree = "优秀"
    Else
        Degree = "错误的成绩"
    End If
End Function
```

这个例子是一个函数，这个函数有一个参数 f 代表了成绩，输入一个成绩后，他根据成绩的范围，然后返回成绩的描述，例如是不及格还是良好等等。

条件语句可以嵌套，以判断复杂的条件，例如：

```
#001  Function DegreePlus(f As Double, Subject As String) _
#002      As String
#003
#004      If Subject = "语文" Then
#005          If f < 60 Then
#006              Degree = "语文 不及格"
#007          ElseIf f >= 60 And f <= 100 Then
#008              Degree = "语文 及格"
#009          Else
#010              Degree = "语文 错误的成绩"
#011          End If
#012      ElseIf Subject = "数学" Then
#013          If f < 60 Then
#014              Degree = "数学 不及格"
#015          ElseIf f >= 60 And f <= 100 Then
#016              Degree = "数学 及格"
#017          Else
#018              Degree = "数学 错误的成绩"
```



```
#019         End If
#020     Else
#021         Degree = "错误的科目"
#022     End If
#023  End Function
```

这段代码是一个嵌套条件语句的例子。嵌套循环的意思是在条件语句内部可以嵌套条件语句。代码的外层判断判断成绩的科目：

```
#001    If Subject = "语文" Then
#002        ......
#003    ElseIf Subject = "数学" Then
#004        ......
#005    Else
#006        Degree = "错误的科目"
#007    End If
```

然后在判断条件语句之内，又添加了新的条件语句，在科目为“语文”的情况下，第 5 行到第 11 行为一个条件语句，判断语文科目的成绩；同样在科目为“数学”的情况下，第 13 到 19 行的条件语句判断科目数学的成绩。

3.2.8. 循环语句

下面来学习循环语句，我们还是从一个实例开始：

```
Sub NoLoop()
    Dim sum As Long
    sum = sum + 1
    sum = sum + 2
    sum = sum + 3
    sum = sum + 4
    sum = sum + 5
    sum = sum + 6
    sum = sum + 7
    sum = sum + 8
    MsgBox "sum = " & sum
End Sub
```

这段代码计算从 1 加到 8 的值，最后使用 MsgBox 输出结果。在不使用循环语句的前提下，这样的程序对于数据不多的情况，还可以实现，如果数据是从 1 到 1000 的值，我们不可以写 1000 条语句来计算这样的结果，这时就需要使用循环语句。

For..Next 循环

For…Next 循环是 VBA 循环语句的一种，以上代码可以使用 For…Next 循环改写为以下形式：

```
#001 Sub Loop1()
#002    Dim i As Long
#003    Dim sum As Long
#004    For i = 1 To 8
#005        sum = sum + i
#006    Next i
#007    MsgBox "sum = " & sum
#008 End Sub
```

这样，如果我们要从 1 加到 1000，我们只需要把第 4 行改为：

```
For i = 1 To 1000
```

这样，代码就可以从 1 加到 1000，这就是循环语句。循环语句首先指定循环的范围，需要一个数据类型为整型的循环变量（例如 Integer，Long），这个变量每次递增 1 或者其他指定的值，直到这个值超出了循环指定的循环范围之外。以上代码使用了循环变量 i，我们也可以使用任意变量。指定循环范围使用“最小值 To 最大值”的形式；之后是循环的内容，可以是一句或者一段代码；最后使用“Next [循环变量]”这样的语句结束循环。

我们还可以使用 Step 参数指定循环变量递增或者递减的步子（量），例如我们需要计算“1, 3, 5, … , 999”的和，我们可以使用如下的代码：

```
Sub Loop2()
    Dim i As Long
    Dim sum As Long
    For i = 1 To 999 Step 2
        sum = sum + i
    Next i
    MsgBox "sum = " & sum
End Sub
```

这里使用了“Step 2”这样的语句，意思是循环变量 i 递增的步子是一次增加 2，这样从 1 开始，每次增加 2，直到 i 小于等于 999，结果就是“1, 3, 5, … , 999”的和。当 Step 为 1 的时候，可以省略 Step 语句。Step 可以是正数，也可以是负数，负数表示递减。例如下面的代码：

```
Sub Loop3()
    Dim i As Long
    For i = 10 To 1 Step -1
        MsgBox i
```



```
    Next i
End Sub
```

输出的结果依次是：10，9，8，7，6，5，4，3，2 和 1。

Excel VBA 开发中经常使用 For…Next 循环来操作某个范围之内的工作表数据，例如读取 C 列第 2 行到第 25 行的数据，或者写入第四行第 4 列到第 50 列的数据。

For…Next 循环也可以嵌套使用，即在循环内部在增加一个或数个循环，如果我们需要编写一个查找工作表的程序，那么就可以使用第一个循环查找不同的行，然后使用第二个循环查找每一行中不同的列。例如，我们在当前工作表第 1 到第 20 行的第 1 到第 5 列中查找某个值的代码可以这样编写：

```
#001  Sub FindInSheet()
#002      Dim var
#003      Dim i As Long, j As Long
#004
#005      var = 3
#006
#007      For i = 1 To 20
#008          For j = 1 To 4
#009              If ActiveSheet.Cells(i, j).Value _
#010                  = var Then
#011                  MsgBox "找到了：" & i & " , " & j
#012              End If
#013          Next j
#014      Next i
#015  End Sub
```

在第 1 到第 20 行的第 1 到第 5 列的某个单元格输入 3，然后运行这段程序，当程序查找到这个单元格，会弹出对话框，显示行和列的值。这段代码第 7 行到第 14 行定义了第一个循环，在其内部定义了第二个循环（8 到 13 行），然后通过一个条件语句判断当前单元格的值和要查找的值是否相同。其中的“ActiveSheet.Cells(i, j).Value”为 Excel 对象，ActiveSheet 表示当前的数据表，Cells(i, j)表示数据表的第 i 行第 j 列的单元格，Value 表示单元格的值。

Do Until 循环

Do Until 循环是 VBA 的另一种循环语句，他执行循环的语句直到满足循环条件。我们写看一个例子：

```
Sub Loop4()
```



```
    Dim x As Long
    x = 0
    Do Until x = 100
       x = x + 1
    Loop
    MsgBox x
End Sub
```

这段代码将累加 x 直到 x 等于 100，可以看出，Do Until 循环与 For…Next 循环最大的区别就是 Do Until 循环不需要知道循环要循环多少次，而可以指定一个条件，当满足此条件时，循环结束。

Exit 语句

在某些情况下，可能需要提前结束循环。例如我们在工作表中查找某个数值或者字符串，当查找到此数值或字符串就可以提前结束循环，而不必等到循环全部结束。例如下面这个例子：

```
#001 Sub ExitLoop()
#002    Dim i As Long
#003    For i = 1 To 100
#004       If i = 50 Then
#005          Exit For
#006       End If
#007    Next i
#008    MsgBox "i = " & i
#009 End Sub
```

这段代码的循环内使用一个条件语句判断 i 的值，如果等于 50，则使用 Exit 语句退出循环，最后使用 MsgBox 输出 i 的值，结果是 50。对于 Do 循环，则使用“Exit Do”语句提前结束循环。

对于嵌套循环，Exit 语句会退出他所在的那层循环，而不会退出其他循环，例如：

```
#001 Sub ExitLoop1()
#002    Dim i As Long
#003    Dim j As Long
#004
#005    For j = 1 To 5
#006       For i = 1 To 100
#007          If i = 50 Then
#008             Exit For
#009          End If
```



```
#010        Next i
#011
#012        MsgBox "j = " & j & ", i = " & i
#013
#014        If j = 4 Then
#015            Exit For
#016        End If
#017    Next j
#018    MsgBox "j = " & j & ", i = " & i
#019 End Sub
```

这段代码是一个嵌套循环的例子，在内部循环，如果 i 等于 50，则会退出内层循环（7-9 行）；然后输出当前的 i 和 j 的值，程序继续外层循环；当外层循环的 j 等于 4，则退出外层循环（14-16 行），输出 i 和 j 的当前的值（分别为 50 和 4）。

3.2.9. 数组

有时候我们需要定义一系列的变量来表示实际中的数值，例如成绩统计中，我们需要记录 30 个人的成绩，是否需要定义 30 个变量？有时候会是 300 个，而定义 300 个变量肯定不现实，因此就需要定义数组。数组是任何编程语言都有的一种数据类型，他表示一系列具有相同的数据类型的变量，但可以只使用一个变量来表示，对于其中的某个具体的值，我们使用索引来表示。

我们使用和定义变量一致的语法来定义数组，例如：

```
Dim Months(11) As String
```

其中 Months 是数组名称，11 表示数组的最大的索引，因为 VBA 缺省的索引是从 0 开始，因此“Months(11)”有 12 个元素，分别是 Months(0)、Months(1)、Months(2)到 Months(11)，一共 12 个元素。后面的 As 语句表示了数组的数据类型。

数组的元素的索引可以是常量，例如 1、2 这样的数字来表示，也可以使用 i 这样的整型变量来表示，例如 Months(i)、Months(j)这样的形式。这样，我们就可以在循环语句中比较方便的处理数组的每个元素。

例如我们决定使用一个数组来存储一个班的成绩，并计算平均成绩，可以这样做：

```
#001 Sub ArrayTest()
#002    Dim i As Long
#003    Dim Score(29) As Double
#004
#005    For i = 0 To 29 Step 1
```



```
#006        Score(i) = Rnd() * 100
#007    Next
#008
#009    Dim sum As Double
#010    Dim average As Double
#011
#012    For i = 0 To 29 Step 1
#013        sum = sum + Score(i)
#014    Next i
#015    average = sum / 30
#016    MsgBox "平均成绩是 " & average
#017 End Sub
```

代码的第 3 行定义了一个数组 Score(29)用来存储成绩，数组有 0 到 29 一共 30 个元素。第 5 到第 7 行使用了“Score(i) = Rnd() * 100”这样的语句给不同的成绩赋值，其中 Rnd 是 VBA 的函数，返回一个 0 到 1 之间的平均值，然后乘以 100 来模拟实际的成绩。在实际的程序中，可以从 Excel 读入成绩，也可以从其他文本文件读入成绩，我们后面会介绍相应的方法。第 9 到 10 行定义了 2 个变量，用来存储总成绩和平均成绩。第 12 到 14 行循环所有成绩来求成绩的和，“sum = sum + Score(i)”每次将 Score 的一个元素的值增加到 sum，最后计算平均值并输出。

3.2.10. 善用工具及其他

VBA 集成开发环境，提供了很多便利的工具可以帮助或辅助我们写出好的程序，其中的方便之处必须亲自使用才可以体会，因此，在学习 VBA 的时候，一定要善用工具[5]。下面举一些笔者认为比较有用的方面，以及编程过程中应该遵循的一些基本原则。

代码编辑器

VBA 的代码编辑器提供了几项非常有用的功能，如代码大小写自动切换，代码自动格式化，即时代码提示。

代码大小写自动切换，即如果我们定义了一个变量“myName”，后面在输入这个变量时，不管我们输入的是“myname”还是“Myname”，只要是相同的名字，VBA 的代码编辑器会自动把这个变量改为“myName”。代码自动大小写切换可以帮助我们发现拼写错误，

[5] 不仅是 VBA 开发，所有的程序开发工作中都应该善用工具。:）

如果我们所有的过程和变量都是按照首字母大写的规则定义的，那么输入这些过程或者变量时，可以全部小写，如果编辑器自动将其首字母改成了大写，那么说明拼写没有错误。

代码即时提示可以使我们不必记忆太多的东西，例如输入对象后会自动列出其属性、方法等内容；输入方法函数后会提示参数信息。

代码自动格式化可以使我们在书写代码时不必过于关心格式化的问题，如等号前后加空格之类。

要求变量声明

VBA 缺省可以不声明变量，在第一次使用的时候自动声明。但建议在开发中使用“要求变量声明”，以减少错误。可以通过“工具 — 选项”对话框，在“编辑器”页，选中“要求变量声明”（图 3-9），则在使用变量前，都必须先通过 Dim 语句声明。

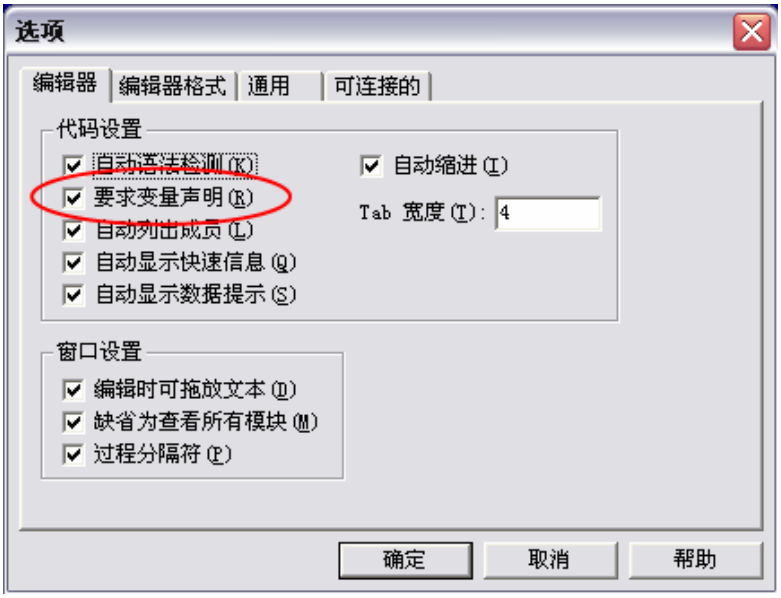


图 3-9 在 VBA IDE 下设置“要求变量声明”

帮助

在实际的编程过程中，一定要善于利用在线帮助，VBA 的在线帮助包含了大量对编程有用的参考信息，任何人都不可能记得住所有的函数、对象的用法和程序语言的语法，所以一定要利用好帮助。帮助的使用可以在任何关键字上按 F1 键查找相关内容，也可以通过帮助目录浏览，或者通过查找输入关键字查找相关内容。缺省安装 Office 或 Excel 不会安装 VBA 的帮助，需要定制安装选择安装 VBA 帮助才可以使用 VBA 的帮助文档。

对象浏览器

VBA 的对象浏览器可以浏览一个对象的属性、方法和事件，通过对象浏览器，我们可以获得一个对象的整体概念，特别对于某些没有提供帮助的对象或第三方对象，对象浏览器更有帮助文档的作用。

使用常量

VBA 中使用了很多常量，可以在编程中直接使用其代表的数字，也可以使用定义好的常量，例如 MsgBox 中的一些参数（按钮参数 vbYesNo 等）。使用常量一者可以获得好的可读性，二者也容易记忆。关于 VBA 以及 Excel 的常量，可以随时通过帮助文档查阅。

代码格式

对于代码格式，一定要养成缩进的习惯，在过程之后，循环语句、判断语句之内，如本书的例子样子，缩进 4 个字符，便于阅读。代码中，在一个逻辑或者操作完成之后，应该空一行，以表示其逻辑关系，在过程与过程之间，也应该空一行。

在代码书写中，如果一句代码过长，应该使用接行字符（“-”）将其分为几行，而不是书写为一行，一般来说，代码的长度不要超过 80 个字符为宜，这样阅读方便，不需要横向拉动滚动条，也不容易出错。例如以下打开文件语句使用了 3 个接行符：

```
Workbooks.OpenText Filename:=strFilename, _
    Origin:=936, StartRow:=1, DataType:=xlFixedWidth, _
    FieldInfo:=Array(Array(0, 1), Array(37, 1), _
    Array(52,1),Array(64,1)), TrailingMinusNumbers:=True
```

注释

VBA 中，使用单引号“’”表示注释，编写程序时，一定要养成注释的习惯。注释不是所有代码都要注释；一般来说，对一个模块、过程、函数，要大概说明其功能，参数；对于一个过程，如果涉及较复杂的算法，要说明其使用的算法或流程。在过程和函数中，对关键代码，说明其操作的目的、算法或流程。

例如以下代码，是一个实际的 VBA 程序的片断，这段代码首先注释了模块的功能；然后定义了一个常量，并对其做了注释；对过程“InsertTernaryAxeChart”也做了相应的注释。


```
Option Explicit

'*******************************************************
'三角坐标投图绘制模块
'   1．根据传入的定义文件，生成并调用对话框，获取数据
'   2．插入图形对象
'   ... ...
'*******************************************************

Private Const THreeEvolution As Double = 1.732051  '根号 3

Public Sub InsertTernaryAxeChart(ChartIniTool As CIniTools)
'三角坐标图解绘制模块主调程序
'参数：ChartIniTool（CIniTools），ini 文件解析对象
'流程：见模块说明

   Dim DiagramTernaryDefine As New CDiagramTernaryDefine
   Dim objChart As Chart
   ... ...
```

3.3. 应用 VBA 操作 Excel

前面介绍了 VBA 的基本语法和使用，但要在 Excel 下使用 VBA 进行开发，还要了解如何使用 VBA 来操作 Excel。我们前面已经或多或少涉及到了 Excel 对象，而 VBA 操作 Excel 正是通过 Excel 对象。

Excel 通过一系列的对象来暴露他的属性和方法，我们把这些对象称为 Excel 对象模型。Excel 对象模型包括了近 200 个对象，这些对象可以代表 Excel 的工作表、工作薄、单元格、图片、格式，以及 Excel 程序本身等等。每个对象都有数量不等的方法和属性，用来操作 Excel。在 VBA 中，我们正是通过操作这些对象来操作 Excel。这里我们只对 Excel 中最常用的对象做一个简单介绍，后面还有“Excel 对象模型”一章会对 Excel 对象做比较深入的介绍。

对于 Excel 对象模型的学习，有下面几个比较好的方法：

（1）查看 Excel VBA 的帮助，查看 Excel 对象模型。选择帮助，打开“Microsoft Excel Visual Basic 参考”，选择“Microsoft Excel 对象模型”。帮助文档有 Excel 对象模型的层次结构和关系，是掌握 Excel 对象模型整体概念的较好方法。

（2）使用 VBA IDE 的对象浏览器，上节已经介绍过了。使用对象浏览器，选择 Excel 对象（图 3-10），可以查看 Excel 的所有对象及其属性、方法，并有简短的介绍。

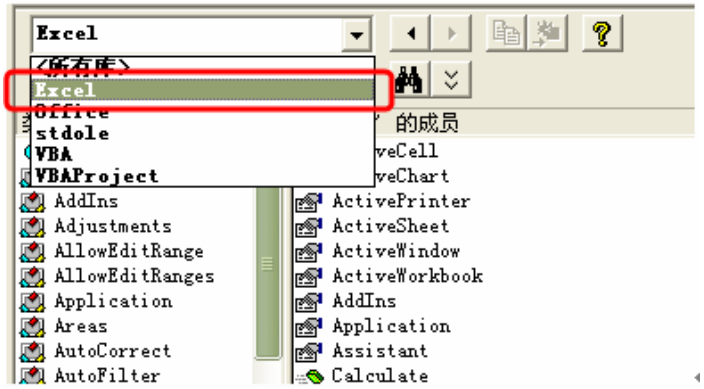


图 3-10 在对象浏览器里选择 Excel，查看其对象模型

（3）使用宏录制器，录制宏，然后查看 VBA 代码。对于很多操作，我们知道如何使用 Excel 完成，但不知道如何使用 VBA 来操作其 Excel 对象，那么就可以采用这种方法。

以上方法通过配合使用，应该会有比较好的学习效果。

Application 对象

Application 对象代表了 Excel 程序，通过 Excel 的 Application 对象，可以操作当前激活的工作薄、工作表或者活动单元格。

下面的例子使用 ActiveSheet 属性来操作当前活动的工作表，将活动工作表的第一行第二列的单元格设置为当前时间：

```
Application.ActiveSheet.Cells(1, 2) = Time
```

其中的 Time 是 VBA 函数，返回当前的计算机时间。我们通过“对象.属性”或者“对象.方法”的格式来使用 Excel 对象。上面的例子使用了 Application 对象的 ActiveSheet 属性，返回当前的活动工作表，然后使用活动工作表的 Cells 属性，操作其单元格。

ActiveCell 返回当前活动的单元格或单元格区域，例如下面的例子使用 MsgBox 输出当前活动单元格的值：

```
MsgBox ActiveCell.Value
```

下面的例子修改了活动单元格的外观：

```
With ActiveCell.Font
    .Bold = True
    .Italic = True
```



```
End With
```

Workbok 对象

Workbok 对象代表 Excel 的一个工作薄，通过 Workbok 对象可以得到工作薄中的工作表（Sheets 属性或 Worksheets 属性）和活动工作表（ActiveSheet 属性，使用 Application 对象也可以），获得当前工作薄的名称、保存路径、保存工作薄等。例如以下语句保存当前工作薄：

```
Application.ActiveWorkbook.Save
```

Worksheet 对象

Worksheet 对象代表了一个工作表，通过 Worksheet 对象可以拷贝、粘贴、打印工作表，或者操作其中的数据，例如前面 Application 对象的 ActiveSheet 返回了一个代表当前工作表的 Worksheet 对象，然后使用 Worksheet 对象的 Cells 属性来操作其中的单元格。

Range 对象

Range 对象应该是 Excel 里使用频率最高的对象之一，Range 可以代表工作表的一个单元格或者单元格区域，例如：

```
MsgBox Worksheets("Sheet1").Range("A1").Value
Worksheets("Sheet1").Range("A7").Value = 20
Worksheets("Sheet1").Range("B7").Formula = "=Sum(A1:A5)"
```

上面的例子使用了“Worksheets("Sheet1")”集合对象来返回工作表 Sheet1，我们可以使用 Worksheets(工作表名称)的形式返回不同的工作表。Range 的 Value 属性可以返回 Range 的值，如果是一个单元格，那么就是一个具体的数值或者字符串等，如果是一个单元格区域，则返回一个数组。同样，我们也可以通过 Value 属性给 Range 赋值。最后一句是使用了 Formula 属性，可以使用此属性给单元格输入一个公式。

3.4. 应用实例

下面的实例是一些可能会在实际中用到的 VBA 代码或代码片断，一方面用来复习前面学习的内容，一方面也可以在实际工作中直接使用。

3.4.1. 使用 VBA 合并列

有时候需要将某些列合并，我们可以使用公式实现，也可以使用 VBA 代码实现，下面的代码将第一列和第二列的单元格的内容连接后，赋给第三列的单元格。

```
Sub MergeTest()
    Dim i As Long
    For i = 3 To 30
        Cells(i, 3) = Cells(i, 1) & " " & Cells(i, 2)
    Next
End Sub
```

上面的例子使用了 Application 对象的 Cells 属性返回当前工作表的 Cells 对象（Application 对象可以省略），使用 Cells 对象来操作工作表的数据。这种方法也是使用 VBA 来操作 Excel 数据的常见方法。

3.4.2. 自动隐藏或显示表格中无数据的行

以下代码可以实现工作表中第一列中没有数据的行（可能有公式或其他内容）自动隐藏， 代码循环到了 300 行，可以根据实际更改。

```
Sub HideCell()
    Dim i As Long
    For i = 1 To 300
        If Cells(i, 1).Value = "" Then
            Rows(i).Hidden = True
        End If
    Next i
End Sub
```

下面的代码将增加了数据的行从隐藏改为显示：

```
Sub ShowCell()
    Dim i As Long
    For i = 1 To 300
        If Cells(i, 1).Value <> "" Then
            Rows(i).Hidden = False
        End If
    Next i
End Sub
```

和上面的例子一样，本例里也省略了 Application 对象，其中 Rows 代表整个一行，使用其 Hide 属性可以设置其是否隐藏。

3.4.3. 使用 VBA 操作工作表单元格

这个例子演示了如何操作单元格的一些方法，可以直接使用在实际工作中。

```
Public Sub WritesCell()
    '最简单在 "[  ]" 中输入单元格名称。
    [A1] = 100  '在 A1 单元格输入 100。
    [A2:A4] = 10  '在 A2:A4 单元格输入 10。

    '采用 Range(" "), " " 中输入单元格名称。
    Range("B1") = 200  '在 B1 单元格输入 200。
    Range("C1:C3") = 300  '在 C1:C3 单元格输入 300。

    '采用 Cells(Row,Column)，Row 是单元格行数，Column 是单元格栏数。
    Cells(1, 4) = 400  '在 D1 单元格输入 400。
    Range(Cells(1, 5), Cells(5, 5)) = 50  '在 E1:E 5 单元格输
入 50。
End Sub

Public Sub ReadCell()
    MsgBox [A1]
    MsgBox Range("B1")
    MsgBox Cells(1, 4)
End Sub
```

对于多个工作表之间的数据的操作，和前面的单元格操作不同的是，在读取的单元格之前要增加工作表对象，例如：

```
Sheets(4).[A1]
Sheets(4). Range("B1")
```

下例中，变量 i 代表了行号。此过程将在单元格区域 D1:D20 中循环，将所有绝对值小于 0.1 的数字都设置为 0（零）。

```
#001 Sub RoundToZero()
#002     Dim i As Long
#003     Dim rCell As Range
#004     For i = 1 To 20
#005         Set rCell = Worksheets("Sheet2").Cells(i, 4)
#006         If IsNumeric(rCell.Value) Then
#007             If Abs(rCell.Value) < 0.1 Then
#008                 rCell.Value = 0
#009             End If
#010         End If
#011     Next i
```



```
#012 End Sub
```

代码中，第 3 行定义了变量 rCell，是 Range 类型，表示 Excel 的一个单元格或单元格区域；第 4 到 11 行循环工作表 Sheet2 的单元格区域 D1:D20，第 6 行使用了 VBA 的函数 IsNumeric 判断单元格的值是否数字，如果是数字才处理；第 7 行使 VBA 的用 Abs 函数返回 rCell 值的绝对值，判断绝对值是否小于 0.1，如果小于 0.1，则在第 8 行将其的值改为 0。

3.4.4. 查找工作表的第一个空行

以下代码查找当前工作表第 2 列（B 列）的第一个空行，然后报告结果并写入当前时间。不断运行这段程序，会发现报告的空行数逐次增加。

```
#001 Sub FindEmpty()
#002    Dim x As Long
#003    x = 1
#004    Do Until (IsEmpty(Cells(x, 2).Value))
#005        x = x + 1
#006    Loop
#007    MsgBox "空行为 " & x
#008    Cells(x, 2) = Time
#009 End Sub
```

代码中第 4 行的 IsEmpty 函数为 VBA 函数，判断当前值是否是空（没有任何值），最后一行给当前空行的 B 列写入当前时间，Time 我们前面使用过，可以返回当前时间。

3.4.5. 改变 Excel 界面的标题

我们来看几个轻松的例子：

```
Sub ChangeCaption()
   Application.Caption = "我的 Excel! "
End Sub
```

这个过程改变 Excel 标题为"我的 Excel！"。

```
Sub MyTime()
   Application.Caption = Now()
   Application.OnTime Now + TimeValue("00:00:01"), "MyTime"
End Sub
```

这个过程在 Excel 标题栏上显示当前时间并不断刷新，程序第一句设置 Application 的 Caption 为当前日期和时间，使用了 Now 这个 VBA 函数，返回当前日期和时间。后面一句中的 OnTime 是 Application 的一个方法，可以设置一个“定时器”，安排一个过程在将来

的特定时间运行（既可以是具体指定的某个时间，也可以是指定的一段时间之后）。OnTime 的使用方法如下：

```
Application.OnTime(开始运行的时间, 运行的过程名, 运行的最晚时间,
是否安排一个新的 OnTime 过程)
```

因此可以使用 Now + TimeValue(time) 可安排经过一段时间（从现在开始计时）之后运行某个过程。例如，本示例设置 15 秒后运行 my_Procedure 过程，从现在开始计时：

```
Application.OnTime Now + TimeValue("00:00:15"), _
    "my_Procedure"
```

本示例设置 my_Procedure 在下午 5 点开始运行:

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

本示例撤消前一个示例对 OnTime 的设置：

```
Application.OnTime EarliestTime:=TimeValue("17:00:00"), _
   Procedure:="my_Procedure", Schedule:=False
```

3.4.6. 隔行格式化工作表

很多时候，我们需要把表格的底色更改为隔行一致，以便于阅读和使用。如果手动来做这件事情，不仅枯燥乏味，而且浪费时间。使用 VBA 可以很快的实现这样的任务。

在 Excel 的 VBA IDE 中输入以下代码：

```
#001 Sub ColorSheet()
#002    Dim i As Long
#003    For i = 1 To Application.Selection.Rows.Count
#004        If i Mod 2 = 1 Then
#005            Selection.Rows(i).Interior.Color = _
#006                RGB(255, 255, 0)
#007        End If
#008    Next i
#009 End Sub
```

在 Excel 中，选中要隔行改变背景色的区域，然后选择“**工具 — 宏 — 宏**”，显示宏对话框，选择“ColorSheet”，运行后可以看到选中的单元格的背景已经被设置为黄色（图 3-11）。

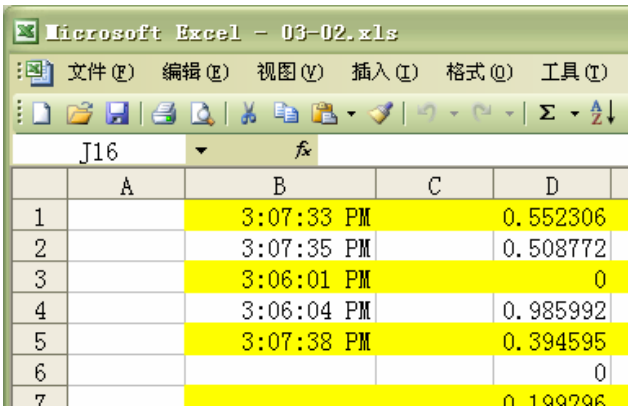


图 3-11 隔行设置单元格背景宏运行结果

下面我们来说明一下以上代码，第 3 行定义了一个循环，循环的范围是 1 到选中部分的行数，使用“Application.Selection.Rows.Count”返回，Selection 表示了工作薄当前选中部分；然后在第 4 行通过条件语句来判断 i 是否奇数，Mod 是数学运算符，用来对两个数作除法并且只返回余数，这样奇数才返回 1，偶数返回 0，因此奇数行将符合条件；最后将背景使用 RGB 函数设置为黄色。

