Microsoft® Excel/Visual Basic® for Windows® 95 编程指南

[美] Microsoft Corporation 著
盛 新 黎军英 译
熊桂喜 审校

清华大学出版社

目　录

试读结束，需要全本PDF请购买 www.ertongbook.com

序言

欢迎使用 Microsoft Excel 使用的编程语言——面向应用程序的 Visual Basic。使用 Visual Basic，你就能够自动化日常任务、增加自定义特征和函数以满足你的需求，并且还可以创建完整的应用程序。

在 Microsoft Excel 中，你可以通过宏来自动执行各项任务。所谓“宏”就是一个用 Visual Basic 写成的指令序列，它告诉 Microsoft Excel 应该做些什么。

Microsoft Excel 提供了一种被称为宏记录器的功能，它通过保存你完成的操作以及在使用 Microsoft Excel 时所选择的命令来为你编写宏。此后，你可以回放或运行该宏来自动地重复宏记录器记录下的操作，从而为你节省大量的时间和精力。

学会了记录和运行宏之后，你就会发现宏的作用有多么大，并且你将身不由己地想加上自己的 Visual Basic 代码并改进原记录的代码，以使得宏具有更强大的功能。

不管你是一位新手还是一位老练的宏开发者，Visual Basic 都能帮你提高工作效率。有了 Visual Basic，你就能创建自定义的命令、菜单、对话框、消息和按钮；并且还可为上述所有项显示自定义的 Help 标题。从自动地完成重复的任务到开发强大的、功能齐全的应用程序，Visual Basic 的众多工具都允许你定制 Microsoft Excel，以使它符合你的特殊需求。

0.1 入　门

*本序言介绍的是已公开印刷的联机文档，随着你不断深入地学习 Visual Basic，你将*会逐步用到这些信息。以下几节将帮助你确定如何让这本书更好地满足你的需求。

0.1.1 摆正你的位置

本书要求读者已经熟悉如何使用 Microsoft Excel。如果你对 Microsoft Excel 感到陌生，那么这本书目前还不适合你读。关于 Microsoft Excel 的一般信息，可参阅《Getting Results with Microsoft Excel for Windows 95》或《Getting Results with Microsoft Office for Windows 95》。

在没有学习 Visual Basic 之前，用户可以用 Microsoft Excel 来记录和使用宏。如果你从未记录或运行过 Visual Basic 宏，可先参阅 Help 中的 Contents 标签上列于“Recording and Running Macros”之下的标题。

注意： 在安装 Microsoft Excel 时，如果单击了 Typical，则还得再运行 Setup 来安装 Visual Basic Help 文件。详细的信息可参见后面的“使用联机帮助”。

本书是为 Microsoft Excel 中的 Visual Basic 的中级用户设计的。书中所提供的信息都是围绕着主题展开的，而不是针对某些特定任务。如果你对 Microsoft Excel 中的 Visual Basic 比较陌生，或者你更乐于接受一种更接近于任务的方法，那么你最好先读《Microsoft Excel Visual Basic Step by Step》——一本 Microsoft Press 出版的自学教程，然后你才能用本书的知识来定制并优化你所掌握的内容。

注意： 可直接从 Microsoft Press 订购《Microsoft Excel/Visual Basic Step by Step》和《Microsoft Excel/Visual Basic Reference》（一种联机 Visual Basic Help 的书面版本）。在订购时，输入你的信用卡号，拨通 615-793-5090 或拨对方付费的 1-800-MSPRESS。为了保证订购工作尽快地完成，务请留下你的基准索引代码 FXL。CompuServe 用户还可打印 GO MSP 来从 Microsoft Press Electronic Book Store 征订这两本书。

0.2 几种学习 Visual Basic 的方法

下面就如何最有效地学习 Visual Basic 提供几点建议。

0.2.1 先学习 Microsoft Excel

对 Microsoft Excel 了解得越多，就能为更好地学习 Visual Basic 打下更坚实的基础。大多数宏通过 Microsoft Excel 来实现一系列动作，并且宏中的大多数指令等价于 Microsoft Excel 中的命令或作用。一般情况下，使用 Visual Basic 与使用没有用户界面的 Microsoft Excel 是有一些相似之处的，所不同的是，在 Microsoft Excel 中使用的是命令和对话框，而在 Visual Basic 中使用的是 Visual Basic 指令。因此，若熟悉 Microsoft Excel 中出现的语句和函数的功能，那么用这些语句和函数来编写指令就会容易得多。

此外，若熟知 Microsoft Excel，还能更好地回答在编写宏时会经常考虑到的问题，即“最佳的方法是什么?”如今人们已经知道如何为那些曾经由单个 Microsoft Excel 命令处理的任务编写长的宏。

0.2.2 需要什么学什么

可以在实际操作中，遇到什么问题就去学习相关的知识。开始学习时，Visual Basic 可能让人感到不知从何处着手，尤其是当你没有使用宏编程语言的经验时，情况更是如此。学习这种语言的一种很好的办法就是学习怎样实现手边现有的特定宏概念。一旦具备了编写不同类型宏的经验时，所有困难也都会迎刃而解。

0.2.3 使用宏记录器

宏记录器能记录你通过 Microsoft Excel 所采取的每个虚拟动作的 Visual Basic 指令。可以用宏记录器来看看在 Microsoft Excel 中的动作如何转换成 Visual Basic 指令，反之亦然。并且，你还将发现，记录下部分宏往往比写出指令更快，且更容易。

0.2.4 使用 Visual Basic Help

Help 是学习 Visual Basic 的一种强有力的工具。在 Visual Basic 模块中，可以打印 Visual Basic 关键字，并且，按下 F1，便可立即显示该关键字的 Visual Basic Help。大多数关键字的 Visual Basic Help 项都包括有一个你可以复制并粘贴到宏中的实例。更多的信息可参见下面的“使用联机帮助”。

0.3 本书的内容安排

本书分章介绍 Visual Basic 的主要功能块以及使用这些功能的技巧。前三章“模块和过程”、“变量、常量和数据类型”以及“控制程序流程”主要是介绍编写 Visual Basic 代码的技巧。

第 4 章“对象和集合”详细地介绍了 Microsoft Excel 对象模型，并举例介绍如何操纵和引用对象来实现 Visual Basic 宏中的任务。

第 5 章“优化大小和速度”介绍了几种可使 Visual Basic 代码更快且更简洁的技巧。

第 6 章“调试”和第 7 章“处理运行时出错”介绍了如何在运行代码之前发现并消除其中的故障，以及如何处理在代码运行时出现的错误。

第 8 章“控制和对话框”以及第 9 章“菜单和工具栏”介绍如何添加定制的用户接口元素到 Visual Basic 宏中。

第 10 章“与其它应用程序通信”详细地介绍了如何使用 OLE 自动化来与其它应用程序通信，并控制这些应用程序。

第 11 章“自动化过程和 OnEvent 过程”介绍了如何编写当某个指定事件（如打开工作簿或单击控制）发生时将运行的过程。

第 12 章“创建附加程序”介绍了为宏创建和使用附加程序的有关信息。

第 13 章“访问外部数据”介绍了如何使用 Data Access Objects（DAO）来输入和输出保存在数据库中的信息。

附录 A 至附录 D 介绍了如何编写国际化应用程序，如何为正在切换到 Visual Basic 的高级 Microsoft Excel 用户提供帮助信息，并且列出了 Microsoft Excel 中的所有工具栏按钮，还绘出了 Microsoft Excel 对象模型图。

0.4 代 码 实 例

Microsoft Excel 包中包括了一些你可打开和运行的代码实例。既可以把其中的任何一部分代码复制到应用程序中，也可以根据需要来修改它们。这些代码保存在 Microsoft Excel 文件夹中的 Examples 子文件夹的 Samples.xls 工作簿里。

0.5 使 用 联 机 帮 助

Microsoft Excel 还为 Visual Basic 语言、Microsoft Excel 支持的对象以及这些对象的特

征和方法提供了较为详尽的 Help 系统。

在安装 Microsoft Excel 时，如果单击了 Typical，则还需要再次运行 Setup 来安装用于 Visual Basic for Microsoft Excel 的 Help。

采用下列四种方法中的任何一种，你都能访问 Visual Basic Help。

- 在 Visual Basic 模块中，在对象、特征、方法、函数或其它关键字中设置插入点，并按下 F1 来获取上下文有关的 Help。
- 在 Help 菜单上单击 Microsoft Excel Help Topics。此后，既可单击 Contents 标签上的 "Getting Started with Visual Basic"，在 Index 标签上查找指定标题或 Visual Basic 术语，也可以在 Find 标签上进行全文搜索。
- 在 Help 菜单上单击 Microsoft Excel Help Topics 。随后，可以在 Answer Wizard 标签上提出某个问题，并阅读在该对话框的 Programming and Language Reference 部分中显示出的内容。
- 保持 Visual Basic 模块继续活动，单击 View 菜单上的 Object Browser，并接着单击 Element Help 按钮（以问号形式出现在 Objects/Modules 框之下）以获取对象、方法、特征或函数的有关信息。

0.6 文档约定

本书采用下表列出的印刷约定。也许你一下子记不住所有这些术语或 Visual Basic 关键字，不过没关系，随着本书的不断深入，你将会了解它们的更详细信息。

约定实例	描　述
setup	要求用黑体打印的字或字符。
Sub、**If**、**ChDir**、**MsgBox**、**True**、**Add**、**Height**、**Application**、**Range**、**Row**	这些第一个字母为大写的黑体字都是语言特有的术语，要么为一种特征、方法或对象名，要么为其它的 Visual Basic 关键字。
对象	在文本中，**斜体**用来表示重要的新术语，通常都是本书中第一次出现的术语。
Propertyname	在代码语法中，斜体表示的是你应提供的信息。
enter	小型大写字母通常用作键和键组合（如 enter 和 ctrl + r）名。
ctrl + v	在两个键名之间加上一个加号（+）表示键组合。例如，CTRL + V 表示同时按下 CTRL 键和 V 键
down arrow	每个箭头键都是指对应键上箭头的方向（左、右、上或下）。箭头键是这四个键的总称。
backspace, home	其它方向键，其方向与名称对应。
myVar	这种字体用来表示实例代码。
Sub StockSale () . . . End Sub	这三个句号中的行或列都表明实例程序中的这一部分已被省略。

0.7 编 程 风 格

本书对代码实例采用以下几条编程风格标准。关于 Visual Basic 代码，更多的信息可参见第 1 章“模块和过程”。

- 下面的字体用于代码。

```
sub HelloWorld
    Cells (1, 1). Value = "Hello, world!"
End Sub
```

- 在代码中，撇号（'）用于注释。

```
' This is a comment; these two lines
' are ignored when the program is running.
```

- 在本书中，所有宏的名称和用户定义的函数的第一个字母都是大写的。注意，宏和函数名中不能出现空格。因此，若名称是由一个以上的单词组成的，则该名称中的其它单词的第一个字母也必须大写。

```
'The AuditResult user-defined function is in the Finance module.
Function AuditResult (latestIncome, expenses, taxes, commissions)
```

参数和变量名的第一个字母必须小写（以便同宏、函数、特征、方法以及对象名区分开来）。

- 关键字的第一个字母大写，并且内带的常量的前面应冠以小写字母“xl”或“vb”。

```
'Sub is a keyword.
Sub Title (titleText)
'xlManual is a built-in constant.
Application. Calculation = xlManual
```

- 在 **Sub** 和 **Function** 过程中的控制流块和语句被缩排在包围着它们的代码中。

```
Sub CheckRecordSound
    SoundRecordCapable = Application. CanRecordSounds
    If SoundRecordCapable Then
        Cells (1, 1). SoundNote. Record
    End If
End Sub
```

- 行连接字符，即一条下划线（_）表示从一行至下一行的代码都属于同一逻辑行。在 Visual Basic 模块中，你可以把这些语句全部打印在一行中。也可以分成几行打印该代码，并在每一行末尾添加一条行连接字符。

```
ActiveSheet. Rectangles. Add _
    Width := 200, _
    height := 200, _
    left := 50, _
    top := 50
```


试读结束，需要全本PDF请购买 www.ertongbook.com

0.8 Microsoft 产品支持服务

Microsoft 提供了多种支持项以帮助你从 Microsoft 产品中获取最大的收益。关于 Microsoft Product Support Services 的详细情况可参阅《Getting Results with Microsoft Excel for Windows 95》或《Getting Results with Microsoft Office for Windows 95》。

在美国之外的任何地方，用户可与本地区的 Microsoft 代表处中的 Microsoft Product Support Services 机构（产品支持部）联系。Microsoft 代表处及它们所服务的国家都详细列在《Getting Results with Microsoft Excel for Windows 95》和《Getting Results with Microsoft Office for Windows 95》中。

第1章 模块和过程

模块和过程是 Visual Basic 程序的基本结构单位。通过学习怎样把复杂的任务分解成一系列简单的过程，怎样从一个过程中调用另一个过程，以及如何在两个过程之间传送数据，我们就能创建易于编写、阅读、调试以及修改的模块代码。

本章主要内容有：

- 结构化 Visual Basic 程序
- 创建过程
- 调用过程
- 使用参数与过程通信

1.1 结构化 Visual Basic 程序

现代编程的目标之一就是产生**模块化**程序，也就是由小型的、不连续的代码单元构成的程序，程序中的每个模块都能独立地实现某个特定的任务。朝着这个目标，Visual Basic 允许你把代码组织成过程、模块和工作簿。

同单一结构的程序相比，由小型而稳定的组件构成的程序具有许多优点。具有完美结构的模块化程序拥有如下优点：

- 易于编写，因为复杂的问题已被分解成一系列不连续的且易于弄懂的任务。
- 易于阅读，因为最高级的过程典型地只含有一系列发向带有描述性名字的低级过程的调用。
- 易于调试，因为每一个任务都是在一个过程中得到实现的，并且还能容易地找出问题并改正问题。
- 高效，因为实现某项普通任务的代码只出现在一个过程中，可以随着调用该过程而多次调用它，而不是像以前那样，在整个程序的不同位置上多次重复出现。
- 易于修改，因为实现某个指定任务的代码只出现在程序中的某一个位置上。这样，在有必要修改这一代码时，就只需在某一个位置上进行改变。
- 十分稳定，因为你可以建立一个可靠的过程库，并且指导程序使用这些相互嵌套的、且已得到验证的组件。
- 十分安全，因为可以把不要求访问数据的程序中各个部分的数据隐藏起来，这样就减少了因偶然事故导致改变数据的危险。
- 易于管理和发布，因为可以把一系列相关的或经常使用的过程作为一个单元来保存和发布。

这一章的其余内容将介绍如何把代码组织成各个组件以及如何控制保存在程序不同部

分中的过程之间的相互关系。

1.1.1 把代码分解为过程

一个**过程**就是由 **Sub** 和 **End Sub** 语句或 **Function** 和 **End Function** 语句所包括在一起的代码单元。一个过程必须能够实现一个简单的、定义完整的任务。通常把过程用于实现重复或共享的任务。

如果代码段包含着一个可实现复杂任务的过程,则可以把该过程分解为几个更小的过程,并编写一个单独的过程来按顺序调用这些小过程。如果还嫌不足,还可继续检查并分解代码中的每一个过程,直到每一个过程都非常简单,并且独一无二。过程实现的任务越简单,你就越容易编写并调试这一过程。同时,过程实现的任务越独特,该过程的适用面就越广。

1.1.2 了解模块的结构

模块就是含有代码的工作簿表。每一个模块都含有一个声明部分以及出现在其后的过程。位于模块最前面的声明部分可以包括以下内容:

- **选项语句。**如果模式中含有任何 **Option** 语句(设置各种模块级选项),则这些语句必须出现在所有过程之前。更多的信息可参见 Help 中的“option”。
- **用户定义的类型定义。**如果模块中含有任何用户定义的类型,则它们的定义必须出现在所有过程之前。关于用户定义的类型,更多的信息可参见第 2 章“变量、常量和数据类型”。
- **声明。**任何不包括在过程中的变量或常量声明都必须设置在模块中的所有过程之前。关于说明变量和常量的更详细情况可参见第 2 章“变量、常量和数据类型”。

除 **Option** 语句之外,用户定义的类型定义、声明以及 Visual Basic 中的所有代码都可以出现在过程中;Visual Basic 模块不能包含自由飘移的(未被包括进来的)命令。

1.1.3 把过程组织成模块和工作簿

既然可以毫不费力地找到某个特定过程或访问普通数据乃至提供一系列相关服务的一系列过程,那么也可以把过程组织成模块并给这些模块赋上一个描述性的名称。例如,如果已编写成一系列实现计算任务的过程,则可以把它们组织成一个模块,并将该模块命名为“Accounting_Procedures”。随后,你就会知道在哪里找到任何一个计算过程。

还可更进一步地把相关模块组织成一个简单的工作簿。这样做可使得把程序转换成附加程序并分布它变得更为容易。关于创建附加程序,具体信息可参见第 12 章“创建附加程序”。

尽管你可能出于组织方面的考虑想把过程保存在不同的模块乃至工作簿中,但你仍然希望能在它们之间交换信息。为了支持在模块和工作簿之间进行数据交换,Visual Basic 允许过程访问保存在其它模块或工作簿中的代码和数据。关于访问保存在其它模块或工作簿中的数据,详细情况可参见第 2 章“变量、常量和数据类型”。关于访问其它模块或工作簿中的信息,详细情况请参阅 1.3.3“调用其它模块中的过程”。

一些简单的应用程序可以刚好组成一个模块;并且应用程序的所有代码也都驻留在该模块中。如果应用程序较大并且较复杂,还可以添加附加模块。实际操作中你还可能发现

你想把一些普通代码用于几个模块中。由于你并不希望在每个模块中都重复使用这类代码，因此，可以单独创建一个模块以包含实现这种普通代码的过程。这样，经过一段时间以后，你就能建立起一个包含共享进程的模块库，这一方面的内容稍后我们再单独讨论。

1.1.4 创建库工作簿

可以把常用的过程组织成模块，并把这些模块保存在**库**中（只含模块的工作簿），并使这些过程在整个程序中都是可见的，从而不必把它们复制到将调用它们的每一个模块和工作簿中。通过从含有调用过程的工作簿中创建对库工作簿的引用，就能使工作簿中的过程能看见库工作簿中的过程。如何创建对库工作簿的引用，详细信息可参见 1.3.3 中介绍“创建对工作簿的引用”这一专题。

1.2 创 建 过 程

在创建过程中，首先必须确定你希望该过程具有多大的可访问范围，你想为该过程赋以什么名称，你希望该过程实现什么任务，为实现这一任务该过程将需要哪一种信息，你希望该过程接受并返回哪一种数据以及你是否打算让该过程影响程序中其它部分的数据。下面我们就分别介绍 Visual Basic 如何允许你为你所编写的每一个过程指定上述信息。

1.2.1 指定过程范围

术语**范围**指的是程序中某一指定程序项（如变量、常量、用户定义的数据类型或过程等）的部分区域是可访问的或“可见的”。在 Visual Basic 中，过程既可以有私用范围也可以有公共范围。具有私用范围的过程只对同一模块中的其它过程是可见的，具有公共范围的过程则对该过程所在的工作簿中的每个模块内的所有过程都是可见的，并且对那些与该工作簿有关系的所有工作簿来说，这一过程同样是可见的。如果没有加以特别声明，过程都拥有公共范围。

如果想限制过程在模块中的可用性，可使用 **Private** 关键字。由于默认时过程都是公共的，因而没有必要使用 **Public** 这一关键字来声明公共过程；但是，这样做也存在着一个好处，这就是更容易让人一眼就看出哪些程序元素具有公共范围、哪些拥有私用范围。下面给出的代码表明了两个公共过程和一个私用过程。

```
Public Sub CalcPayments()
    ' place statements here
    MsgBox "I'm public!"
End Sub

Sub CalcIncome()     ' Public by default
    ' place statements here
    MsgBox "I'm public, too!"
End Sub

Private Sub CalcSavings()
    ' place statements here
    MsgBox "I'm private!"
End Sub
```

注意：不能把声明为 **Private** 的过程作为孤立的过程来运行，因为只能从其它过程中调用私用过程。

若想限制过程在工作簿中的可用性，可以把该过程声明为 **Public** 过程，但是应在模块的声明部分中添加 **Option Private Module** 语句，把该模块变为私用模块。

1.2.2 命名过程

在为过程命名时，必须遵守以下几条原则。一个过程名：

- 必须以字母开头。
- 不能包括嵌入的句号、数学或比较算符以及类型说明字符。
- 不能超过 255 个字符。
- 在其范围内必须是独一无二的。
- 不能为 Visual Basic 方法、特征、函数、参数的名称，也不能以其它限定的关键字作为过程名。关于限定的关键字究竟是哪些，可参见 Help 中的"restricted keywords"。

提示：在调用过程时，为了避免使用模块限定词，最好对每个过程名都只使用一次。

1.2.3 过程的类型

Visual Basic 允许创建两种过程，即 **Sub** 过程和 **Function** 过程。

Sub 过程是介于 **Sub** 和 **End Sub** 语句之间的代码单元，它实现任务但不返回值。

Function 过程是介于 **Function** 和 **End Function** 语句之间的代码单元。与 **Sub** 过程一样，**Function** 过程也实现任务。但是，所不同的是，**Function** 过程还返回一种可通过某种表达式来使用的值，如下例所示。

```
totalSales = CalculateTotal (x) 'CalculateTotal is a Function procedure
```

也可用 **Function** 过程来替代复杂的工作表公式，就好像使用内在的 Visual Basic 函数如 **Sqr**、**Cos** 和 **Chr** 一样。

注意：从工作表上的公式中调用的 **Function** 过程一定不得以任何方式改变工作簿中的数据或改变 Microsoft 环境。详细情况可参见 1.2.6"在工作表上使用 Function 过程"。

1.2.4 从函数中返回值

对于返回值的函数而言，它必须包括一个函数指定语句以指定一个值给该函数的名字，如下例所示。指定给 ConeSurface 的值将是函数返回的值。

```
Function ConeSurface(radius, height)
    Const Pi = 3.14159
```

```
    coneBase = Pi * radius ^ 2
    coneCirc = 2 * Pi * radius
    coneSide = Sqr(radius ^ 2 + height ^ 2) * coneCirc / 2
    ConeSurface = coneBase + coneSide
End Function
```

当 **Function** 过程返回一个值时，这个值可能会成为一个更大表达式的一部分。例如，在另一过程中，下面这一行把 ConeSurface 和 ScoopSurface 函数的返回值结合到了其算式中。

```
totalSurface = ConeSurface (3, 11) + 2 * ScoopSurface (3)
```

必须为过程提供以便于它实现其任务的信息（如前例中的 radius 和 height）是以参数形式传送给过程的。关于参数，具体情况可参见 1.4 “使用参数与过程通信”。

1.2.5 为 Function 的返回值指定数据类型

如果想把 **Function** 返回的值限定为一种指定的数据类型，可在函数名之后的括号后面用 **As** 关键字来指出返回值的类型。

```
Function CalcPay (hours, rate) As Currency
```

没有 **As** 项，返回值将通过默认方式采用**变体**类型。

注意：也可以把 **As** 关键字用在给出参数的括号内以指定过程参数的数据类型，如下例所示：

```
Function CalcPay (hours As Integer, rate As Currency) As Currency.
```

更具体的信息可参见 1.4.1 “指定参数数据类型”。

1.2.6 在工作表上使用 Function 过程

可用 **Function** 过程来替代工作表单元中的长工作表公式（就像是使用内在的工作表函数一样），我们称这类函数为**用户定义的工作表函数**。

例如，假定名为 “MyBook” 的工作簿中含有函数 ConeSurface（这一函数我们在 1.2.4 “从函数中返回值” 里已定义过）。如果你想让 MyBook 中的工作表上的某个单元（可以是任何引用 MyBook 的工作簿中的单元）来保存一个半径为 5 高度为 10 的锥体面积，那么必须把公式 = ConeSurface（5，10）置于该单元内。如果想在不引用 MyBook 的工作簿中的工作表单元内使用 ConeSurface 函数，则必须在函数名前面加上工作簿名和一个感叹号，例如，=MYBOOK .XLS！ ConeSurface（5，10），并且，MyBook 必须是打开的。

注意：可以在已装载的附加程序中调用函数而不必打开添加的工作簿，并且也无须指定工作簿名。关于创建附加程序的有关信息，可参见第 12 章 “创建附加程序”。

自定义的工作表函数不得改变工作簿中的任何数据，也不能以下述任何方式改变 Mi-

试读结束，需要全本PDF请购买 www.ertongbook.com