

Mastering VBA Second Edition

[美] Guy Hart-Davis 著



杨密 杨乐 柯树森 译 马茂盛 审校

 SYBEX

VBA

从入门到精通
(第二版)



电子工业出版社

PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

<http://www.phei.com.cn>



经典 实用 权威

Mastering VBA Second Edition

VBA从入门到精通

(第二版)

本书提供了最快、最简便的方法学习使用VBA，以便自定义和增强Office软件以及许多可使用VBA的软件的功能。本书不仅提供了VBA宿主软件的一般技巧，而且涉及了在Word、Excel、PowerPoint、Access以及Outlook中的应用。本书针对Office 2003软件，也适用于Office XP以及Office 2000。

- 什么是VBA，用VBA可以做什么
- 在Office软件中使用录制宏、编辑宏的功能
- 使用Visual Basic编辑器生成和编辑代码
- 找出程序中需要的对象、属性和方法
- 使用变量、常量和数组存储和操作数据
- 创建确定和不确定的循环
- 用条件选择不同的决策
- 用信息框、输入框以及内置对话框进行交互
- 生成自定义对话框并对控件编程
- 创建清晰、易维护的模块
- 测试、调试以及完善程序
- 用数字证书和安全特性保证代码安全

VBA从入门到精通（第二版）

Project 2007中文版从入门到精通（普及版）

Excel 2007与VBA编程从入门到精通

Excel 2007中文版从入门到精通（普及版）

Word 2007中文版从入门到精通（普及版）

PowerPoint 2007中文版从入门到精通（普及版）

Access 2007中文版从入门到精通（普及版）

Windows Vista中文版从入门到精通（普及版）

Windows Vista从入门到精通（中文版）

Windows Server 2003中文版从入门到精通

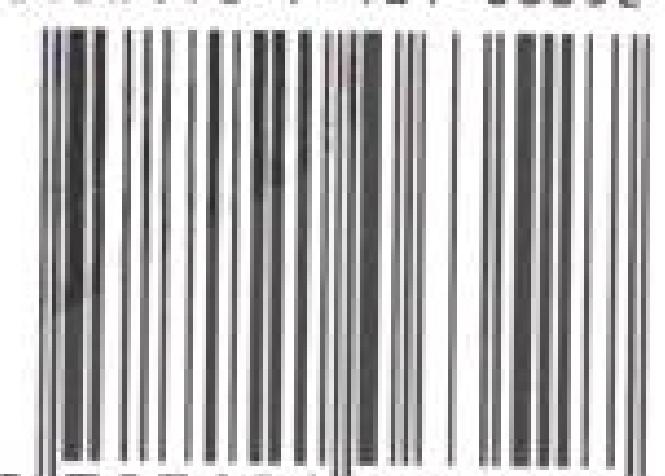


责任编辑：王军花

本书贴有激光防伪标志，凡没有防伪标志者，属盗版图书。



ISBN 978-7-121-06352-7



9 787121 063527 >

定价：72.00元

Mastering VBA Second Edition

VBA 从入门到精通（第二版）

[美] Guy Hart-Davis 著

杨密 杨乐 柯树森 译

马茂盛 审校

了幾本編著。 1900-1910年

電子工業出版社

电子工业出版社

Publishing House of Electronics Industry

·8342586(010)· 中山地質系

10.1101/010416; this version posted April 1, 2016. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted 10.1101 a license to display the preprint in perpetuity. It is made available under a CC-BY-NC-ND 4.0 International license.

北京 · BEIJING

内 容 简 介

本书是一本系统全面地介绍 VBA 的书,包括 6 大部分、30 章。本书以目前最流行的 Word、Excel、Access 等软件为基础,由浅入深,层层递进,精辟地阐述了 VBA 的应用方法和使用范围。无论是新手还是有经验的开发人员,都可以从中获得有益的内容和信息。通过这本书,读者可以了解 VBA 是附着在应用软件上的工具,而这些应用软件可能正是我们最常用的软件(例如 Word、Excel),用好了 VBA 我们就可以将这些常用软件的功能大大加强,解决一些我们原来认为不可能解决的问题。

本书不仅可以作为 VBA 初学者的入门读物,也可作为具有一定 VBA 基础的读者进一步学习和掌握的教材,还可以作为大专院校相关专业师生的参考书。

Copyright © 2005 by Wiley Publishing, Inc., Hoboken, New Jersey. All rights reserved. This translation published under license. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning, without the prior written permission of the publisher. The SYBEX Brand trade dress is a trademark of Wiley Publishing, Inc. in the United States and/or other countries.

本书英文版由美国 Wiley 公司出版,Wiley 公司已将中文版独家版权授予中国电子工业出版社及北京美迪亚电子信息有限公司。未经许可,不得以任何形式和手段复制或抄袭本书内容。

版权贸易合同登记号 图字:01-2007-2001

图书在版编目(CIP)数据

VBA 从入门到精通·第 2 版/(美)戴维斯(Davis, G. H.)著;杨密,杨乐,柯树森译.一北京:电子工业出版社,2008.6

书名原文:Mastering VBA, 2nd Edition

ISBN 978-7-121-06352-7

I. V… II. ①戴…②杨…③杨…④柯… III. BASIC 语言—程序设计 IV. TP312

中国版本图书馆 CIP 数据核字(2008)第 047368 号

责任编辑:王军花

印 刷:北京天竺颖华印刷厂

装 订:三河市金马印装有限公司

出版发行:电子工业出版社

北京市海淀区万寿路 173 信箱 邮编:100036

北京市海淀区翠微东里甲 2 号 邮编:100036

开 本: 787×1092 1/16 印张: 35.375 字数: 900 千字

印 次: 2008 年 6 月第 1 次印刷

定 价: 72.00 元

电子工业出版社

凡所购买电子工业出版社图书有缺损问题,请向购买书店调换。若书店售缺,请与本社发行部联系。联系及邮购电话:(010)88254888。

质量投诉请发邮件至 zlts@phei.com.cn, 盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线:(010)88258888。

译者的话

《VBA 从入门到精通》(第二版)一书翻译完成了，作为译者我们简单地谈一谈自己学习(翻译的过程也是学习的过程)的体会，希望对其他读者(译者也是读者)有所帮助，因为我们自己就是 VBA 的热心用户。

VBA 是嵌入在应用软件中的工具，其使用情况和被嵌入的软件(也称宿主软件)有关。目前来看，大体有三种状态：其一，很少使用。例如 Word 软件，使用 Word 软件的用户很多，但使用其 VBA 工具的寥寥无几。主要是因为一般的文字处理已经满足了绝大多数用户，而不再需要 VBA 的帮助。其二，作为辅助工具使用。例如 Excel 软件，使用 Excel 软件的人越来越多，主要用于财务、统计等需要进行一些数字分析的领域。这时候，VBA 用得越好，操作越简便。其三，作为开发工具使用。例如 Access 软件，Access 是一个数据库软件，和 Word、Excel 都不相同，如果不能开发就很难使用。本书正是以这三个软件为主要内容进行描述的(但不限于这三个软件，还包括 PowerPoint、Outlook 等)，所以非常实用。读完这本书，我们有了新的想法，例如，可以按需要制作特定的 Word 模板，自动生成文件格式，填写内容后就产生新的文件，从而节省了格式处理的时间，这就特别适合发文较多的大型企业或者政府部门。另外，Word 和 Excel 都可以作为报表工具，即把 Access 等数据库处理的内容打印出来，后者还可以建立自己的计算公式。如果不想涉足 Access 的窗体开发，可以只作为数据库使用，输入的界面用 Excel 来处理。

根据我们的体会，入门侧重于方法，提高侧重于认识。本书在入门和提高两方面都很有特色。在入门的方法上，本书着重介绍了宏录制的功能。虽然具有宏录制功能的软件不多，但入门已经足够了。若干年前，我们正是借助 Excel 的宏录制功能学到了 VBA 的编程技术。例如可以利用宏录制功能处理工作表的格式(即建立一个处理特定格式的小程序反复使用)。假设要进行这样的处理，可以把工作表上的字号设定为 10，把 D、E、F 三列设定为保留 2 位小数并含有千分节的数字格式，然后根据单元格中的内容自动设定列的宽度。然而，录制下来的宏至少有 17 行之多(录制内容略)，读者可以自己测试。读完这本书，我们将学会用下面的语句实现同样的效果，前后一共只有 5 行：

```

With ThisWorkbook. ActiveSheet
    . Cells. Font. Size = 10
    . Columns ("D: F"). NumberFormatLocal = "#, ##0. 00"
    . Columns ("A: F"). AutoFit
End With

```

在提高方面，本书详细介绍了面向对象的概念。有了这样的概念，可以真正成为 VBA 的使用高手。这里，我们不妨也举个例子。

因工作需要，我们用 Access 开发了一个库存软件。在软件中设计了入库和出库窗体，在阅读该书之前，我们分别设计了两个窗体，而两个窗体的颜色不同。读完这本书，我们就有了新的思路，只用一个窗体就能够达到两个窗体的效果。颜色用下面的语句处理。

```

Dim mCon As Control          1
If mInOtID = 1 Then          2
    For Each mCon In Me. Controls      3
        mCon. BorderColor = 16711680  4
        mCon. ForeColor = 16711680  5
    Next                         6
    Me. Caption = " 入库"           7
End If                         8
If mInOtID = 2 Then          9
    For Each mCon In Me. Controls      10
        mCon. BorderColor = 128       11
        mCon. ForeColor = 128       12
    Next                         13
    Me. Caption = " 出库"           14
End If                         15

```

上面的语句写入窗体的打开或加载事件中。基本思路是把窗体中的控件作为对象处理，改变了对象的属性。这些控件都是用来输入数据的文本框或组合框等。如果窗体处理（输入）入库信息，控件的边框和内容就变成绿色（第 2 行至第 8 行）；如果窗体处理出库信息，相应的内容就变成红色（第 9 行至第 15 行）。程序中的 mInOtID 为全局变量，进入该窗体之前首先赋值。要进行入库处理，令该值为 1；要进行出库处理，令该值为 2。

读者对书的评价大多仁者见仁、智者见智，但我们认为读者不会对本书的实用性和系统性提出异议，总之，这是一本不可多得的 VBA 工具书。

```

    . Cells("D1"). Value = "商品类别"
    . Cells("E1"). Value = "商品名称"
    . Cells("F1"). Value = "商品数量"
    . Cells("G1"). Value = "商品单价"
    . Cells("H1"). Value = "商品金额"
    . Cells("I1"). Value = "操作员"
    . Cells("J1"). Value = "操作时间"
    . Cells("K1"). Value = "备注"

```

VBA 程序运行结果如图 1-1 所示。图 1-1 展示了在 Excel 中输入商品信息后的效果。可以看到，表格中显示了商品类别、商品名称、商品数量、商品单价、商品金额、操作员、操作时间以及备注等信息。操作员和操作时间列显示了“操作员”和“操作时间”字样，而备注列显示了“无”。

前 言

VBA (Visual Basic for Application) 是一个功能强大的工具，可以在微软公司的办公软件以及其他嵌入了 VBA 的软件中自动运行。用 VBA 自动处理，可以减少大量的时间和人力。使用它，用户感到轻松自如，事半功倍，大大增强职业安全感。

本书介绍怎样用 VBA 编程，并使用微软的 Office 2003 办公软件给出具体的实例。读者可以把在本书中所学到的原理用于其他具有 VBA 功能的软件，例如 AutoCAD、WordPerfect 等其他大量的相关软件。

VBA 能做什么

具有 VBA 功能的软件几乎可以用 VBA 进行所有相应的交互式（手动式）操作。例如，在 Word 中，可以生成一个文档，加入文字，进行格式化处理，进行编辑；在 Excel 中，可以将许多工作簿的数据汇总到一个工作簿里；在 PowerPoint 中，可以生成一个自定义的演示文稿，包括来自多种数据源的最新数据。

VBA 使操作更加快捷、准确、可靠，更加节省人力。正如可以根据公司的状况进行决策一样，VBA 也可用来决策。在程序中加入决策结构和循环（有条件的循环），能够使处理能力大大超出一般人的能力范畴。

除了使手动操作自动化外，VBA 还提供了交互界面——消息框、输入框和用户窗体。这些图形界面用来制作窗体和自定义对话框。VBA 可以在软件中生成用户自己的应用程序。例如，可以在 PowerPoint 中制作一个自定义程序，自动生成一份演示文稿。

使用 VBA，还可以在一个软件中访问另一个软件。例如，在 Word 中使用 VBA 可以启动 Excel 进行计算处理，并把结果填入 Word 文档中。同样地，在 Excel 中使用 VBA 可以把特定的对象输出到 PowerPoint 中自动生成的演示文稿中。

因为 VBA 提供了一套标准的工具，只是不同软件的功能不同，因此，在一个软件中学 会使用 VBA，就可以很快将学到的知识用于另一个软件。例如，先在 Excel 中学习 VBA，接着可以在 Outlook 中使用 VBA 技术。当然，必须学习 Outlook 中的功能，因为其功能不同于 Excel。不过，VBA 的框架是完全相同的。

正如学习任何一种编程语言一样，一开始接触 VBA 有相当大的难度。不过，可以使用微软 Office 软件中的宏录制工具（主要有 Word、Excel 和 PowerPoint）减少难度。本书将使用宏录制功能作为学习编程的起点。首先介绍使用宏录制方法，接着说明如何修改宏录制的程序代码，然后才进入 VBA 的基本语法结构。在这个基础上，再进一步加深难度。

本书的内容

本书介绍如何在具有 VBA 功能的软件中进行自动操作。用 Word、Excel 和 PowerPoint

软件作为一般性的例子，因为这些软件是微软办公软件，一般都会安装。本书的最后一部分讨论如何针对这三个软件进行编程，同时也涉及到 Outlook 和 Access。

第一部分“录制宏和 VBA 入门”包括以下内容：

- ◆ 第 1 章介绍如何用软件中的宏录制功能录制宏。在 Word、Excel、PowerPoint 和 Project 软件中录制宏。还要介绍如何指定运行宏的方法以及如何删除宏。
- ◆ 第 2 章介绍 Visual Basic 编辑器，这是一个用于生成 VBA 代码的软件（无论是编辑录制的代码还是从零开始编写的代码），以及用户窗体。本章后半部分讨论如何自定义 Visual Basic 编辑器，以便使用起来更快捷有效。
- ◆ 第 3 章介绍如何编辑已录制的宏，它使用了在第 1 章中录制的宏。读者可以循序渐进，并学会在 Visual Basic 编辑器中测试宏。
- ◆ 第 4 章介绍如何在 Visual Basic 编辑器中从零开始编写代码。针对 Word、Excel 和 PowerPoint 各编写一段程序。

第二部分“使用 VBA”包括以下内容：

- ◆ 第 5 章介绍 VBA 的基本语法。给出整体概念，同时学习在 Visual Basic 编辑器中编写语句。
- ◆ 第 6 章介绍如何使用变量和常量。变量和常量用来在程序中存储信息。
- ◆ 第 7 章讨论如何使用数组。数组类似于大型变量，可以同时存入多条信息。
- ◆ 第 8 章介绍用来编程的对象。说明如何使用宏录制器、对象浏览器和帮助系统查找对象；如何用对象变量描述对象；以及了解对象模型的概念及其作用。

第三部分“指令决策、使用循环与函数”包括以下章节：

- ◆ 第 9 章介绍怎样使用 VBA 的内部函数，包括字符转换函数、数学函数以及文件管理函数。
- ◆ 第 10 章介绍怎样生成自己的函数以补充内置函数，包括在具有 VBA 功能的软件中通用的函数，以及 Word、Excel 和 PowerPoint 特定软件中使用的函数。
- ◆ 第 11 章介绍在代码中怎样使用条件语句（例如 If 语句）做决定。条件语句很关键，可以使代码更灵活。
- ◆ 第 12 章介绍怎样使用循环重复执行操作，包括固定循环次数的循环和用条件指定循环次数的循环。同时讨论怎样避免死循环。死循环会永远执行下去或者直到计算机崩溃。

第四部分“使用消息框、输入框和对话框”包括以下内容：

- ◆ 第 13 章介绍怎样使用消息框和用户交流，让用户对程序如何运行做一个简单的决定，并使用输入框提供程序运行需要的信息。
- ◆ 第 14 章讨论怎样使用 VBA 用户窗体生成简单的对话框，让用户提供信息，做出选择，以指导程序的结果。
- ◆ 第 15 章讨论怎样建立复杂的对话框。这包括动态对话框，可以让用户单击按钮时自动更新；隐藏内容的对话框，在特定的情况下显示出来；多页信息的对话框；以及带有选择性控件的对话框。

第五部分“生成有效的代码”包括以下内容：

- ◆ 第 16 章介绍反复使用的模块如何比一次性模块优越，以及怎样生成。
- ◆ 第 17 章介绍 VBA 的调试原则，解释各种发生的错误，以及怎样处理这些错误。

- 面处◆第 18 章讨论怎样建立运行出色的代码，在错误的环境下也很稳定；即使程序结束也一个一致使用户保持最佳状态。
- ◆第 19 章讨论 VBA 提供的安全机制，防止用户不知不觉遭遇恶意代码。该章讨论数字认证和数字签名，怎样选择合适的安全设定，怎样创建及删除密码。
- 第六部分“办公软件编程”包括以下 11 章：
- ◆第 20 章介绍 Word 对象模型，解释怎样使用 Word 中的关键对象、包括 Document 对象、Selection 对象和 Range 对象；以及在 Word 中设定选项。
 - ◆第 21 章讨论怎样使用 Word 中最常用的对象，包括查找、替换；页眉、页脚、页码、分段、页面设置、窗口、视图和表格。
 - ◆第 22 章介绍 Excel 对象模型，解释怎样使用 Excel 中的关键对象，包括 Workbook 对象、Worksheet 对象、ActiveCell 对象和 Range 对象，以及在 Excel 中设定选项。
 - ◆第 23 章介绍怎样使用图表、窗口，以及在 Excel 中通过 VBA 查找和替换。
 - ◆第 24 章开始介绍 PowerPoint 对象模型以及重要对象，包括 Presentation 对象、Windows 对象、Slide 对象和 Master 对象。
 - ◆第 25 章进一步介绍 PowerPoint 中的 VBA 如何与图形、页眉、页脚以及用来自动设置和运行幻灯片的 VBA 对象工作。
 - ◆第 26 章介绍 Outlook 对象模型及其重要对象。介绍 Outlook 的可生成对象、主要界面项目，操作 Outlook 对象的一般方法以及操作信息、日历项目、任务与任务要求以及搜寻的方法。
 - ◆第 27 章介绍 Outlook 中的事件。有两种事件（应用程序级事件和项目级事件），可以编程以响应 Outlook 动作（例如新邮件到达）和用户动作（例如生成新的联系人）。
 - ◆第 28 章介绍怎样使用 Access 对象模型以及一些主要对象的关键功能。
 - ◆第 29 章介绍怎样用 VBA 处理 Access 数据库中的数据。
 - ◆第 30 章介绍怎样用 VBA 在一个软件中操作另一个软件。介绍有哪些工具，怎样自动处理，怎样使用 Shell 函数，以及怎样使用数据对象、DDE 和 SendKeys。

本书怎样使用

本书的内容环环相扣。为了避免不必要的内容重复，各章节内容彼此相关，后面的章节建立在前面章节的基础上。

本书前面五部分针对 Word、Excel 和 PowerPoint 给出各种程序示例。如果读者使用这些软件（或者某个软件），尽量参照这些例子。某些例子或许可以直接用于工作当中，然而大多数只给出技巧与原理。读者可以利用这些技巧和原理生成自己的代码。

第六部分也是最后一部分，介绍具体的程序，用 VBA 对 Word、Excel、PowerPoint、Outlook 和 Access 编程。若希望在这些软件中使用 VBA，就应当学习该部分。最后一章介绍怎样在一个软件中调用另一个软件，例如在 Word 中调用 Excel。

本书是否适合所有的读者

是的。

本书适合所有希望在内置 VBA 的软件中学习使用 VBA 的读者。自动完成任务涉及面很广，可能是一个简单的程序，用一个按键就可能进行复杂烦琐的任务，也可能是开发一个用户软件，其界面与原软件的界面完全不一样。

本书试图将理论尽可能地用实际操作表现出来，在操作中给出理论示例。例如，介绍循环时，给出短小的程序表现每个循环，运行结果一目了然。

本书的约定

本书采用几个约定表达特定含义。

- ◆ ➤ 表示在菜单中选择命令。例如“选择‘文件’➤‘打开’”的意思是单击“文件”菜单并选择“打开”命令。
- ◆ + 表示按键组合。例如“Ctrl+Shift+F9”表示按住 Ctrl 键和 Shift 键后再按下 F9 键。有些组合意义含混（例如，“Ctrl++”表示按住 Ctrl 键再按下+键，也可表示按住 Ctrl 键和 Shift 键，再按=键），因此应仔细阅读。
- ◆ ←、→、↑ 和 ↓ 表示键盘上出现的箭头。必须注意，←不是退格键（许多键盘上以此表示退格键）。退格键用“Backspace”表示。
- ◆ 程序字体表示来自程序的内容。完整的程序另起一段，较短的表达式直接在文章中出现。
- ◆ 斜体表示新术语或可变信息（例如驱动器名称在各个计算机上都不同，因此读者需要自己确定）。
- ◆ ➡ (连接箭头) 表示一个整行被分开。输入这样的代码必须写成一行，例如以下三行实际为一行。

```
MsgBox System.PrivateProfileString("",  
    "HKEY_CURRENT_USER\Software\Microsoft\  
    "Office\11.0\Common\AutoCorrect", "Path")
```

书中还有注意、提示和警告。

注意：指与当前主题相关的附加事项。

用剪贴簿

提示：指有用的信息与建议，通常和当前主题有关。

警告：指出与当前主题有关的可能出现的问题。

善用组合键

致 谢

感谢对本书做出贡献的人，他们是：

- ◆ 市场采集和推广编辑 Tom Cirtin，他们使本书被大众认可。
- ◆ 通过编辑和制作过程指导本书的项目编辑 Katherine Perry 和 Rachel Gunn。
- ◆ 既聪明又认真的技术审阅人 John Mueller。
- ◆ 考虑周全的技术编辑 Linda Recktenwald。
- ◆ 为本书设计版式的排版人员 Craig Woods。
- ◆ 校对本书的 Nancy Riddiough。
- ◆ 创建本书索引的 Nancy Guenther。

01 第二个一章的 PowerPoint 方式 02 第二个一章的 Excel 方式

A-Z 用处 目录

量变质变 章 2

量变质变 章 2

第一部分 录制宏和 VBA 入门

第 1 章 在 Office 程序中录制和**运行宏** 2**什么是 VBA 以及 VBA 能做什么** 2**VBA 与 VB 的差别** 2**宏基础** 3**录制宏** 3**计划宏** 4**打开宏录制器** 4**命名宏** 5**设定运行宏的方式** 8**运行宏** 12**在 Word 中录制样本宏** 13**在 Excel 中录制样本宏** 15**创建一个个人宏工作簿，如果还没有****它的话** 15**在 Excel 中录制样本宏** 16**在 PowerPoint 中录制样本宏** 17**指定运行宏的方式** 18**将宏指定到工具栏按钮或菜单项** 18**将宏指定到组合键** 21**删除宏** 21**第 2 章 从 Visual Basic 编辑器入手** 24**打开 Visual Basic 编辑器** 24**和所选择的宏一道打开 Visual Basic****编辑器** 24**打开 Visual Basic 编辑器** 25**导引到一个宏** 25**使用 Visual Basic 编辑器的****主窗口** 26**工程资源管理器** 26**对象浏览器** 29**代码窗口** 29**属性窗口** 32

长篇二章

量变质变 章 2

量变质

为 Excel 创建一个过程	67	为 PowerPoint 创建一个过程	70
----------------	----	---------------------	----

第二部分 使用 VBA

第 5 章 VBA 的基本语法	78	第 7 章 数组变量	99
准备	78	什么是数组	99
过程	78	声明数组变量	99
函数	79	在数组中存储数值	101
子过程	79	多维数组	102
语句	79	声明动态数组	102
关键词	81	再定义数组的大小	102
表达式	82	从数组中返回信息	103
运算符	82	清除数组的内容	103
变量	82	检查某个变量是否为数组变量	103
常量	83	检查数组的边界	104
参数	83	数组排序	104
指明参数的名称和忽略参数的名称	83	数组查询	106
对象	84	在数组中实现线性查询	107
集合	85	在数组中实现二分法查询	110
属性	85	第 8 章 寻找所需的对象、方法和属性	114
方法	85	什么是对象	114
事件	85	属性	115
第 6 章 了解变量、常量以及枚举	86	方法	116
常量	86	集合	117
使用变量	86	集合中的对象	117
变量命名的方法	86	在集合中添加对象	118
声明变量	87	查寻需要的对象	118
设定变量的数据类型	93	使用宏录制器录制对象	118
常量	97	使用对象浏览器	120
声明自己的常量	97	使用帮助查找对象	123
语法	97	使用列表属性/方法的特色	124
设定常量的范围和有效事件	98	使用对象变量代表对象	125
常量列表	98		

第三部分 指令决策、使用循环与函数

第 9 章 使用函数	130	使用 Val 函数取出字符串中的数字	134
函数	130	使用 Str 函数将数值转换为字符	135
使用函数	131	使用 Format 函数设定格式	135
将参数传递给函数	132	使用 Chr 函数以及常数输入特殊字符	138
数据类型转换	133		
使用 Asc 函数返回字符代码	134		

用函数处理字符串	139	002 函数	159
使用 Left、Right 和 Mid 函数返回字符		002 编写在 Excel 中使用的函数	160
串的一部分	140	002 编写在 PowerPoint 中使用的	
使用 InStr 和 InStrRev 函数在一个字符		002 函数	161
串中查找另一个字符	142	第 11 章 用代码决策	164
使用 LTrim、RTrim 和 Trim 去除		010 怎样比较	164
空格	144	010 使用逻辑运算符测试多种条件	165
用 Len 函数检查字符串的长度	144	011 If 语句	167
用 StrConv、LCase 和 UCase 改变字符串的		011 If ... Then	167
大小写	145	011 If...Then...Else 语句	169
使用 StrComp 函数进行比较	146	011 If...Then...Else If...Else 语句	170
使用 VBA 的数学函数	147	012 用 If 和 GoTo 产生循环	174
使用日期和时间函数	147	If 语句嵌套	175
用 DatePart 函数分解日期	148	013 Select Case 语句	177
使用 DateDiff 函数	149	013 语法	177
使用 DateAdd 函数	149	第 12 章 使用循环重复执行	180
使用文件管理函数	150	014 何时使用循环	180
使用 Dir 函数判断一个文件是否		014 循环的基本知识	180
存在	150	015 用 For 循环于固定的循环次数	181
返回当前的路径	151	015 For...Next 循环	181
第 10 章 编写自定义函数	152	015 For Each...Next 循环	186
函数的组成部分	152	015 使用 Exit For 语句	186
编写一个函数	153	016 用 Do 循环于可变的循环次数	187
手动生成函数	153	016 Do While ... Loop	187
通过“添加过程”对话框生成		016 Do ... Loop While 循环	190
函数	154	016 Do Until ... Loop 循环	191
向函数传递参数	154	016 Do ... Loop Until 循环	193
声明参数的数据类型	155	016 使用 Exit Do 语句	194
指定可选参数	155	017 While...Wend 循环	195
控制函数的使用范围	155	018 循环嵌套	196
可用于任何 VBA 宿主软件的函数		019 防止死循环	198
实例	156		
编写在 Word 中使用的自定义			

第四部分 使用消息框、输入框和对话框

第 13 章 使用消息框和输入框以获得用户输入		000 消息框	202
打开某一过程以进行工作	200	000 使用消息框的优点和缺点	203
在 Word 和 Excel 中显示状态栏信息	201	000 消息框的语法	203
		001 显示简单的消息框	204
		001 显示多行消息框	205

为消息框选择 buttons 206	显示和隐藏对话框 241
为消息框选择图标 206	设置默认命令按钮 242
为消息框设置默认按钮 207	从对话框中取回用户的选择 242
控制消息框的模式 208	从文本框中返回一个字符串 242
为消息框指定标题 209	从选项按钮中返回一个值 243
向消息框添加“帮助”按钮 210	从复选框中返回一个值 243
为消息框指定帮助文件 210	从列表框中返回一个值 244
只使用部分参数 211	从组合框中返回一个值 245
从消息框取回一个值 211	把对话框连接到过程的例子 245
输入框 212	Word 例子：移动段落过程 245
输入框的语法 213	通用例子：从列表框中打开文件 253
从输入框取回输入 214	生成对应于该用户窗体的代码 255
当消息框和输入框均不满足需要时 214	使用应用程序的内置对话框 257
第 14 章 生成简单的自定义对话框 215	显示内置对话框 258
什么时候需要使用自定义对话框 215	在内置对话框中设置和恢复选项 260
生成自定义对话框 216	返回用户在对话框中选择的按钮 261
设计对话框 216	为对话框指定超时 261
插入用户窗体 217	第 15 章 生成复杂对话框 262
重命名用户窗体 219	生成和使用复杂对话框 262
向用户窗体添加控件 221	更新对话框以反映用户的选择 262
重命名控件 224	显示对话框的附加部分 263
移动控件 224	在对话框中跟踪过程 266
复制和粘贴控件 225	使用多页对话框和 TabStrip 控件 267
更改控件上的标签 225	生成无模型对话框 274
工具箱控件的重要属性 226	为对话框选择位置 275
使用控件组工作 237	使用事件来控制窗体 276
对齐控件 238	仅仅应用于 UserForm 对象的事件 278
放置控件 239	应用于 UserForm 对象和容器控件的事件 282
调节对话框的 Tab 顺序 239	应用于大多数控件的事件 286
把对话框连接到过程 240	仅仅应用于极少数控件的事件 297
加载和卸载对话框 241	

第五部分 生成有效的代码

第 16 章 构建模块及使用类 300	调用过程 301
构建模块化代码 300	代码的逻辑改进 303
什么是模块化代码 300	改进代码布局 307
模块化代码的优点 301	生成及使用类 310
怎样编写模块化代码 301	类模块的作用 311
模块中的代码布局 301	简述 311

383 ··· 使用 ActiveDocument 对象进行工作中	371	311 ··· 使用 Sections, PageSetup, Windows 和 Views 进行工作	389
384 ··· 工作	371	311 ··· 给文档添加一节	389
385 ··· 使用 Selection 对象进行工作	372	312 ··· 更改页面设置	390
386 ··· 检查所选内容的类型	372	312 ··· 打开含有打开文档的新窗口	390
387 ··· 检查所选内容的文字部分类型	373	312 ··· 关闭除第一个窗口之外的所有与活动	
388 ··· 获得关于当前所选内容的其他		312 ··· 文档对应的窗口	391
389 ··· 信息	374	312 ··· 拆分窗口	391
390 ··· 在所选内容处、所选内容之后或		318 ··· 显示与窗口对应的文档结构图	391
391 ··· 之前插入文本	375	318 ··· 滚动窗口	391
392 ··· 在所选内容中插入段落	376	318 ··· 排列多个窗口	392
393 ··· 应用某种样式	376	318 ··· 定位窗口和调整窗口的大小	392
394 ··· 扩展所选内容	376	318 ··· 确认项目已在窗口内显示	393
395 ··· 折叠所选内容	377	320 ··· 改变文档的视图	393
396 ··· 创建和使用 Range	377	321 ··· 放大视图以显示多个页面	393
397 ··· 定义带名称的 Range	378	321 ··· 使用表格进行工作	394
398 ··· 重定义 Range	378	322 ··· 创建表格	394
399 ··· 使用 Duplicate 属性存储或复制格		323 ··· 选择表格	395
400 ··· 式设置	379	323 ··· 将文本转换为表格	395
401 ··· 设置 Options 对象	379	323 ··· 确认所选内容在表格之内	396
402 ··· 确认超链接需用 Ctrl+单击	379	324 ··· 找出所选内容在表格中什么地方	397
403 ··· 关断改写模式	379	324 ··· 对表格排序	398
404 ··· 设置默认的文件路径	380	325 ··· 给表格添加一列	398
405 ··· 关闭跟踪更改	380	325 ··· 从表格中删除一列	399
第 21 章 使用 Word 中广泛使用的对象		326 ··· 设置列的宽度	399
406 ··· 进行工作	381	326 ··· 选择某一列	400
407 ··· 在 VBA 中使用 Find 对象和		327 ··· 给表格添加一行	400
408 ··· Replacement 对象	381	327 ··· 从表格中删除一行	400
409 ··· 了解对应于 Execute 方法的语法	382	328 ··· 设置一行或多行的高度	400
410 ··· 使用 ClearFormatting 方法	383	328 ··· 选择某一行	401
411 ··· 使查找和替换投入工作	383	329 ··· 插入单元格	401
412 ··· 使用 Headers、Footers 和		329 ··· 返回单元格内的文本	401
413 ··· PageNumbers 进行工作	384	329 ··· 将文本输入单元格	402
414 ··· 了解 VBA 如何构成页眉和页脚	384	329 ··· 删除单元格	402
415 ··· 接触到页眉和页脚	384	329 ··· 选择单元格区域	403
416 ··· 检查页眉或页脚是否存在	385	329 ··· 将表格或行转换为文本	403
417 ··· 与前一节的页眉或页脚相链接	385	第 22 章 了解 Excel 对象模型和重要	
418 ··· 生成不同的第一页页眉	385	对象	405
419 ··· 生成不同的奇数页页眉和偶数页		428 ··· 获得 Excel 对象模型的概括性	
420 ··· 页眉	386	428 ··· 知识	405
421 ··· 将页码添加到页眉和页脚	386		

· 了解 Excel 的可创建对象	405	· 给图表添加标题	430
· 使用 Workbooks 进行工作	407	· 使用图表的坐标轴进行工作	430
· 创建工作簿	407	· 使用 Windows 进行工作	430
· 保存工作簿	408	· 在工作簿上打开新窗口	431
· 打开工作簿	410	· 关闭窗口	431
· 关闭工作簿	412	· 激活窗口	431
· 共享工作簿	413	· 排列窗口和调整窗口的大小	431
· 保护工作簿	413	· 缩放窗口和设置显示选项	432
· 使用 ActiveWorkbook 对象进行工作	413	· 使用 Find 和 Replace 进行工作	433
· 使用 Worksheets 进行工作	414	· 使用 Find 方法进行搜索	433
· 插入工作表	414	· 使用 FindNext 方法和 FindPrevious	
· 删除工作表	415	· 方法继续搜索	434
· 复制或移动工作表	415	· 使用 Replace 方法进行替换	434
· 打印工作表	416	· 搜索和替换格式设置	435
· 保护工作表	417	第 24 章 了解 PowerPoint 对象模型和重要对象	436
· 使用 ActiveSheet 对象进行工作	418	· 获得 PowerPoint 对象模型的概括	
· 使用 ActiveCell 或 Selection 进行工作	418	· 性知识	436
· 使用 ActiveCell 进行工作	418	· 了解 PowerPoint 的可创建对象	436
· 使用 ActiveCell 进行工作	418	· 使用 Presentations 进行工作	437
· 使用 Selection 进行工作	420	· · 创建基于默认模板的新演示文稿	438
· 使用 Range 进行工作	420	· · 创建基于模板的新演示文稿	438
· 使用单元格区域进行工作	421	· · 打开演示文稿	439
· 创建带名称的区域	421	· · 保存演示文稿	439
· 删除带名称的区域	422	· · 关闭演示文稿	441
· 使用带名称的区域进行工作	422	· · 将演示文稿或幻灯片导出到图形	
· 使用已用区域进行工作	422	· · 文件	441
· 使用特定单元格进行工作	422	· · 打印演示文稿	442
· 将公式输入单元格	423	· · 将模板应用于演示文稿、幻灯片或	
· 设置 Options 对象	423	· · 幻灯片子集	442
· 在 Application 对象中设置选项	423	· · 使用 ActivePresentation 进行工作	443
· 在工作簿中设置选项	424	· 使用 Windows 和 Views 进行工作	443
第 23 章 使用 Excel 中广泛使用的对象		· 使用 ActiveWindow 进行工作	444
· 进行工作	426	· 在演示文稿上打开新窗口	444
· 使用 Charts 进行工作	426	· 关闭窗口	444
· 创建图表	426	· 激活窗口	445
· 指定图表的源数据区域	427	· 排列各窗口和调整窗口的大小	445
· 指定图表类型	427	· 改变视图	445
· 使用图表中的系列进行工作	428	· 使用窗格进行工作	446
· 给图表添加图例	429		

使用 Slides 进行工作	446	创建自定义放映	465
添加幻灯片到演示文稿	446	删除自定义放映	465
从现有演示文稿中插入幻灯片	447	开始幻灯片放映	466
使用标识符查找幻灯片	448	改变幻灯片放映的大小和位置	466
改变现有幻灯片的版式	448	在各幻灯片之间移动	466
删除现有的幻灯片	448	暂停放映和使用白屏与黑屏	467
复制和粘贴幻灯片	449	切换到自定义放映和结束自定义	
创建幻灯片副本	449	放映	467
移动幻灯片	449	退出幻灯片放映	468
借助名称来访问幻灯片	449	第 26 章 了解 Outlook 对象模型和重要对象	469
对幻灯片子集进行工作	450	获得 Outlook 对象模型的概括性知识	469
为幻灯片设置格式	450	了解 Outlook 将 VBA 项目存储在何处	471
为幻灯片、幻灯片子集或母版设置切换方式	451	了解 Outlook 的可创建对象和主要的用户界面项目	471
使用 Masters 进行工作	452	使用 Application 对象进行工作	471
使用幻灯片母版进行工作	452	使用 NameSpace 对象进行工作	472
使用标题母版进行工作	452	使用检查器和浏览器进行工作	474
使用讲义母版进行工作	453	创建项目	475
使用备注母版进行工作	453	关闭 Outlook	476
删除母版	453	了解使用 Outlook 对象进行工作的	
第 25 章 使用 Shapes 进行工作和运行		一般方法	476
幻灯片放映	454	使用 Display 方法	476
使用 Shapes 进行工作	454	使用 Close 方法	476
将 Shapes 添加到幻灯片	454	使用 Delete 方法	477
删除形状	457	使用 Printout 方法	477
选择所有形状	457	使用 Save 方法	477
重定位形状和调整形状的大小	458	使用 SaveAs 方法	477
复制形状的格式设置	458	针对邮件进行工作	478
使用形状中的文本进行工作	458	创建新邮件	478
为形状或形状范围设置动画	461	针对邮件内容进行工作	478
使用 HeadersFooters 进行工作	462	给邮件添加附件	479
返回所需要的页眉或页脚对象	462	发送邮件	480
显示或隐藏页眉或页脚对象	463	针对日历项目进行工作	480
设置页眉或页脚中的文本	463	创建新的日历项目	480
为使用日期和时间的页眉和页脚设置		针对日历项目内容进行工作	480
格式	463	针对任务和任务请求进行工作	481
使演示文稿中所有页眉和页脚标		创建任务	481
准化	463		
设置和运行幻灯片放映	464		
控制放映类型	464		

针对任务项目内容进行工作	481	使用 CurrentDb 方法转回当前数	
将任务指派给同事	482	据库	502
搜索项目	482	打开和关闭当前数据库	502
第 27 章 使用 Outlook 中的事件进行		同时打开多个数据库	503
工作	485	关闭数据库	504
使用应用程序级事件进行工作	485	创建和移除工作区	504
使用 Startup 事件	485	为数据库设定启动选项	505
使用 Quit 事件	486	应用 Screen 对象	508
使用 ItemSend 事件	486	使用 DoCmd 对象执行命令	509
使用 NewMail 事件和 NewMailEx		使用 OpenForm 方法打开窗体	510
事件	487	使用 PrintOut 方法打印对象	511
使用 AdvancedSearchComplete 事件和		使用 RunMacro 方法运行宏	511
AdvancedSearchStopped 事件	488		
使用 MAPILogonComplete 事件	488	第 29 章 使用 VBA 处理 Access 数据库	
使用 Reminder 事件	489	中的数据	513
使用 OptionsPagesAdd 事件	489	了解如何进行操作	513
使用项目级事件进行工作	489	准备访问数据库中的数据	513
声明对象变量并初始化事件	490	引用相应的对象库	514
了解应用于所有消息项目的事件	490	与数据库建立连接	514
了解应用于浏览器、检查器和视图		打开记录集	516
的事件	492	用 ADO 打开记录集	516
了解应用于文件夹的事件	493	选择如何访问 ADO 记录集中的	
了解应用于项目和结果的事件	493	数据	517
了解应用于 Outlook 面板的事件	494	用 DAO 打开记录集	518
了解应用于提醒的事件	494	访问记录集中的特定记录	521
了解应用于同步处理的事件	495	使用 MoveFirst、MoveNext、Move-	
第 28 章 了解 Access 对象模型和重要		Previous 和 MoveLast 方法	521
对象	496	使用 Move 方法移动到指定记录	522
开始使用 Access 中的 VBA	496	查找记录	522
创建模块	497	在 ADO 记录集中查找记录	522
创建函数	497	在 DAO 记录集中查找记录	523
创建一个宏来运行函数	498	返回记录中的字段	524
使用 AutoExec 宏设置 Access 的		编辑记录	524
启动项	499	插入和删除记录	524
运行子过程	499	关闭记录集	525
了解 Option Compare Database			
语句	499	第 30 章 实现不同软件间的访问	526
纵览 Access 对象模型	500	不同软件相互交流的工具	526
了解 Access 的可创建对象	500	使用自动方式传输信息	526
打开和关闭数据库	502	前期绑定和后期绑定	527
		用 CreateObject 函数创建对象	528
		用 GetObject 函数返回一个对象	528

第一部分 录制宏和 VBA 入门

- ◆ 第 1 章 在 Office 程序中录制和运行宏
- ◆ 第 2 章 从 Visual Basic 编辑器入手
- ◆ 第 3 章 编辑已录制的宏
- ◆ 第 4 章 在 Visual Basic 编辑器中从头生成代码

第 1 章 在 Office 程序中录制和运行宏

本章将向读者介绍如何在 Microsoft Office 程序中录制宏。通过录制宏，用户可以自动执行一系列操作，从而提高工作效率。本章将介绍录制宏的基本步骤，并通过一个具体的例子来说明如何录制宏。最后，还将讨论如何运行录制好的宏以及如何修改宏。

录制宏的基本步骤如下：

- 启动 Microsoft Word、Excel 或 PowerPoint 等 Office 程序。
- 执行所需的操作，例如插入表格、格式化文本或插入图表等。
- 按 **Alt + F11** 键打开 Visual Basic 编辑器。
- 在“工具”菜单中选择“宏”命令，然后选择“录制新宏”。
- 在“录制宏”对话框中输入宏的名称并设置其他选项（如保存位置）。
- 开始录制宏，所有操作都会被记录下来。
- 完成录制后，停止录制。
- 保存宏并退出 Visual Basic 编辑器。

录制好的宏可以在以后使用时直接运行。要运行宏，可以在“工具”菜单中选择“宏”命令，然后选择录制好的宏。宏将在后台运行，直到你停止运行它。如果想要停止运行宏，可以在宏运行时按 **Esc** 键。

录制宏时，应注意以下几点：

- 确保录制宏时没有其他宏正在运行，以免发生冲突。
- 避免在录制宏时进行任何可能会影响宏正常运行的操作。
- 在录制宏时，尽量使用相对引用而不是绝对引用，以便在不同的工作簿或工作表中使用宏时能够正确地引用单元格。
- 录制宏时，尽量避免使用复杂的公式或函数，以免影响宏的运行速度。

第 1 章 在 Office 程序中录制和运行宏

- ◆ 什么是宏
- ◆ 在 Word 中录制宏
- ◆ 在 Excel 中录制宏
- ◆ 在 PowerPoint 中录制宏
- ◆ 运行宏
- ◆ 删除宏

Office 宏录制器
从 VBA 到 Visual Basic
第 1 章 ◆
第 2 章 ◆
第 3 章 ◆
第 4 章 ◆

本章介绍了解 VBA 最简便的方法：用 Office 程序中的宏录制器录制简单的宏，然后运行这些宏，以重复宏中所包含的动作。可以直接把需要反复操作的任务录制为宏，提高工作效率。还可以用宏录制器生成 VBA 代码，执行需要的动作，然后编辑这些代码增加功能和灵活性。

什么是 VBA 以及 VBA 能做什么

VBA 是微软公司开发的程序语言，可以嵌入到软件中。在支持 VBA 的软件中可以用 VBA 进行自动化工作。所有主要的 Office 程序如 Word、Excel、PowerPoint、Outlook、Access、FrontPage 和 Project 都支持 VBA，因此可以说大多数 Office 软件都可以实现自动化。然而，微软还许可其他软件公司和开发商使用 VBA（清单见 <http://msdn.microsoft.com/isv/technology/vba/partners/default.aspx>）。因此，VBA 还可以用于许多其他软件。

前面提到“在软件中进行自动化工作”，并不是很明确，因为可以用 VBA 做的事情很多，举例如下：

- ◆ 可以录制一个宏，自动运行一系列标准动作。例如，在 Word 文档中插入一张图片，调整图片的尺寸和样式，再加上图题并选择适合的字体。或者将 Excel 图表插入 PowerPoint 幻灯片，进行格式处理后加上文字说明。
- ◆ 可以写一些代码自动运行多次命令，根据运行的条件给出判定。例如，可以在 PowerPoint 中每次打开文件时执行一系列的动作。
- ◆ 可以生成用户窗体或者自定义对话框，让用户进行选择，为运行的代码设定条件。
- ◆ 可以用 VBA 执行在用户窗体中不能直接操作的动作。例如在使用大多数软件时，只能对当前文件进行字面操作，如 Word 中的当前文档，Excel 的当前工作簿，等等，而使用 VBA，就可以操作未激活的文件。
- ◆ 可以用一个软件控制另一个软件。例如，可以用 Word 把表格从文档写入 Excel 工作表。

VBA 与 VB 的差别

VBA 来源于 Visual Basic，是从 BASIC 中派生出来的编程语言。BASIC 是缩写，即

Beginner's All-Purpose Symbolic Instruction Code (初学者全方位象征性指令)。BASIC容
易被用户掌握使用，因为使用了可以识别的英语单词（或者是基本可识别的词汇），而不是
简单抽象的程序术语。Visual Basic 具有很好的图形效果，用于支持 Windows 操作系统的图
形用户界面 (GUI)，提供拖曳编程的工具，且可以使用共享的图形元素。

VBA 包括 Visual Basic 中运行的通用命令代码以及特定软件的对象。每种软件的对象
都不相同，因为其特点和功能都不相同。

例如，Word 中的 VBA 对象不同于 Excel 中的对象，因为 VBA 在 Word 中的特点和命
令不会在 Excel 中出现。然而，VBA 的命令形式和结构在 Word 和 Excel 中都是相同的，两
者的知识可以迅速转换。例如，在 Excel 的 VBA 中，Word 的 VBA 中以及 PowerPoint 的
VBA 中都可以使用 Save 方法（方法从根本上说就是命令）来保存一个文件。在 Excel 中，
该命令写成 ActiveWorkbook. Save，在 Word 中写成 ActiveDocument. Save，而在 Power-
Point 中应当为 ActivePresentation. Save。

VBA 总是和主软件（例如 Word，Quattro Pro，或者 Visio）一同使用。除了一些独立
的项目可以用微软 Office 开发版实现以外，必须打开主软件才能使用 VBA。就是说，不能
像 Visual Basic 那样用 VBA 开发独立的软件。若有必要，可以将主软件隐藏，用户只能见到用
VBA 开发的界面，这样做可以制造独立软件的感觉，是否有必要这样做取决于程序的要求。

宏基础

宏是一连串可以重复使用的指令，可以使用一个命令反复运行宏，有些软件可以让宏自
己运行。例如，可以在 Word 软件中做一个宏自动对格式不正确的文档进行格式处理，可以
在打开文档时手动或自动运行宏。

宏是一种子过程，子过程有时也称做子程序。一般说来，人们并不严格称其为“子”，
而直接叫“过程”或“程序”。宏有时被看做录制的代码而不是写入的代码，不过，很多人
采用了更广泛的含义，即也包括写入的代码。例如，先录制一个宏，经过编辑使其更紧凑更
有效率，而且加入其他命令，人们也将这看做为宏。

在支持录制宏的软件（Word、Excel 和 PowerPoint）中，可以有两种方法生成宏：

- ◆ 打开宏录制器，执行一系列希望宏执行的动作；
- ◆ 打开 Visual Basic 编辑器输入 VBA 指令。

可以用宏录制一些基本操作，然后打开宏把不必要的命令删去。在对宏进行编辑时，还
可以加入其他命令。可以加入控件结构和用户界面素材（如信息框和对话框），这样宏可以
做出决定并选择结果。

一旦生成了宏，可以规定执行的方法。在大多数软件中，可以将宏加入菜单、键盘组
合、工具按钮，并且随时运行它。你也可以在录制宏时规定它运行的方法。不过，大多数情
况下，最好将宏脱离默认环境，然后再设定运行的方法。

录制宏

创建 VBA 代码最简单的方法就是用宏录制器录制宏，但只有一些软件支持宏录制器。
打开宏录制器，选择某个使用宏的方法（使用工具按钮、菜单或按键组合），接着实施操作，

然后关闭宏录制器，在用软件进行操作时，宏录制器将操作指令以 VBA 的编程语言录制下来。录制一结束，可以在 Visual Basic 编辑器中见到代码，并进行修改。如果代码运行正确就不需要再处理，只要在需要的时候选择工具按钮、菜单、组合键，或直接由宏对话框运行宏即可。

在下面的章节中，可以见到录制宏的步骤。过程很简单，但若从未录制过宏就必须熟悉一些背景。了解基本情况之后就可以在 Word、Excel 和 PowerPoint 中录制示例宏，学会怎样使用 Visual Basic 编辑器之后就可以阅读并熟悉书中的宏了。

计划宏

在打开宏录制器之前，先要计划做什么。大多数情况下，必须设定好软件，一切准备就绪，再把命令录制下来。例如，用 Word 录制一个编辑宏，应先打开一个包括文本内容的文档，并选择为当前窗口。

另一种情况是，设定也是宏的一部分。这时，必须保证软件的状态正是宏希望的，然后才开始录制宏。例如，如果希望 Excel 的工作簿在激活时是空的，宏就必须生成一个新的工作簿而不是当时正使用的工作簿。

注意：有些软件（例如 Word）在执行不希望录制的动作时，可使宏录制器暂停，

该功能有助于在计划宏时解决不希望出现的问题。例如录制宏已经开始，要用的文件还没有打开。通常应当在开始录制宏之前做好准备。

打开宏录制器

打开宏录制器的方法是选择“工具”>“宏”>“录制新宏”。宏录制器显示“录制宏”对话框，给出默认的宏名（Macro1, Macro2 等）以及说明，用户可以接受或更改。在图 1.1 中，用户已在对话框中修改了宏名和说明。

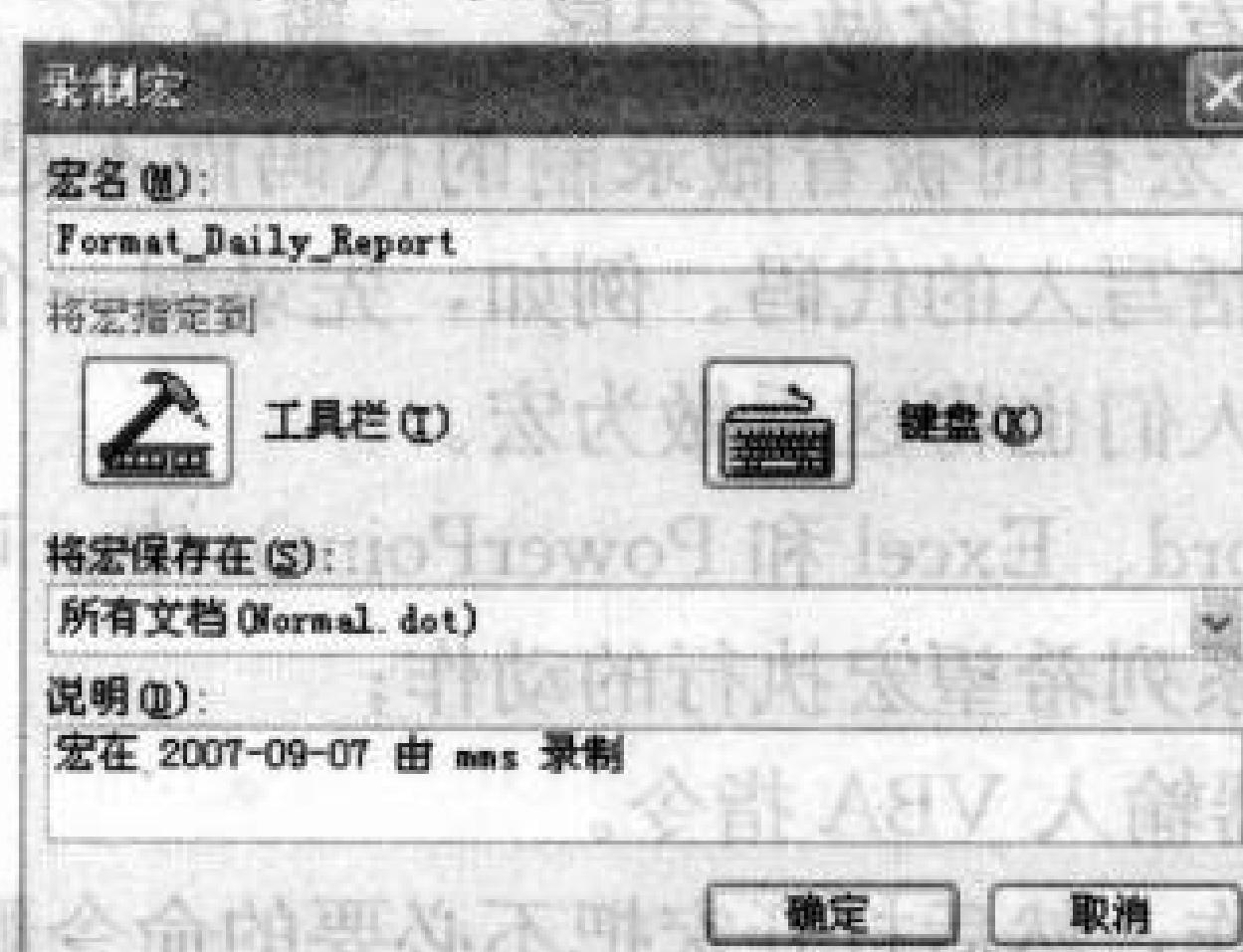


图 1.1 在录制宏对话框中，为要录制的宏输入名称。在说明一栏中

给出简明扼要的文字。该图为 Word 软件的录制宏对话框

提示：在 Word 中，要启动（或终止）宏录制器，可以双击状态条上的录制指示器（如果状态条显示出来，如默认状态下）。在苹果机上的 Word，应单击而非双击指示器。

注意：在使用办公软件自己的菜单时，宏只有在使用了一次以后才会出现在工具菜单上。因此，将短菜单放下，稍等片刻待菜单显示所有不常用的项目，或者直接单击菜单底部向下的箭头使其显示出来（如果不使用事先设定的菜单，选择“工具”>“自定义”，并选择“选项”页面上的“显示完整菜单”复选

复选框。在Office 2000中，应清除“选项”页面上的“最新使用的命令”复选框。该处理对所有的办公软件都有作用)。

“录制宏”对话框的外观对于各软件有所不同，因此对话框必须提供符合各自特点的选项。每一次都必须对宏命名和为其提供说明。一般情况下，必须指明宏存放的地点，Word、Excel和PowerPoint都需要这样做。大多数“录制宏”对话框还需要指明使用宏的方法，要么提供文本框输入按键组合，要么通过命令按键进入另一个对话框(例如：Word软件所使用的自定义对话框)。

大多数带有VBA的微软公司的软件都有工具栏，可以从工具栏上操纵宏和Visual Basic编辑器。

这些Visual Basic工具栏因命令不同也有所不同，实际上是按键的数量不同。如果软件中的Visual Basic工具栏含有录制宏按钮，就可以单击该按钮显示“录制宏”对话框。图1.2为Word软件中的Visual Basic工具栏。

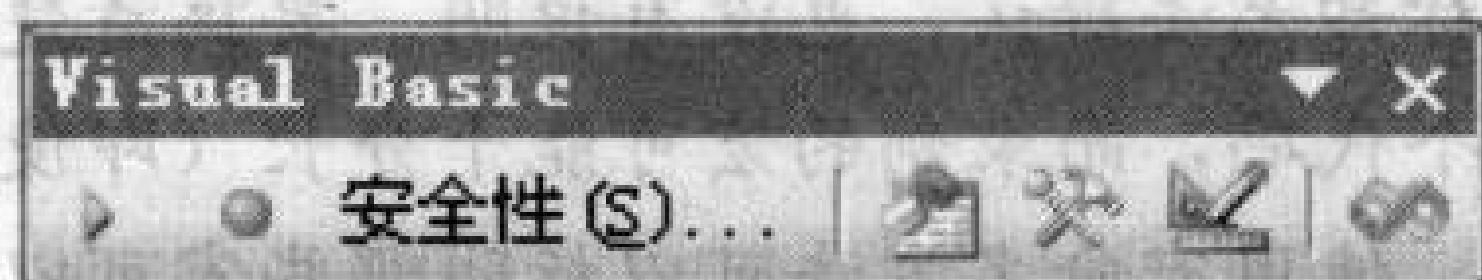


图1.2 可使用VB工具栏操作宏

注意：Outlook软件和Access软件没有Visual Basic工具栏。

Visual Basic工具栏的按钮的功能如下：

“运行宏”按钮：显示“宏”对话框，可在框中选择运行宏(还可选择用该对话框在宏编辑器中生成宏)。

“录制宏”按钮：显示“录制宏”对话框。录制宏的时候，若Visual Basic工具栏在屏幕上，录制宏按钮显示为按下状态，单击可停止录制。

“安全性”按钮：显示“安全性”对话框，详见第19章。“安全性”对话框用来决定软件使用命令代码的安全级别。可以规定命令代码的信任来源(信任来源是将个人或组织认定为安全的代码的来源)。

“Visual Basic编辑器”按钮：打开“Visual Basic编辑器”。第2章介绍Visual Basic编辑器(本书此后大多数篇幅涉及编辑器)。

“控件工具箱”按钮：显示“控件工具箱”，用来在文档中加入控件。有的软件还可以从工具栏菜单中显示“控件工具箱”(右击菜单条或工具栏以显示菜单)或者使用“视图”>“工具栏”子菜单。

“设计模式”按钮：将当前文档切换为设计模式，若“控件工具箱”未显示可使其显示出来。并显示“退出设计模式”工具栏(用来退出显示模式)。“设计模式”按钮是个触发式按钮。不过，若退出设计模式，控件工具箱并不会隐藏起来。即使控件工具箱是在设计模式中显示出来的。

“微软脚本编辑器”按钮：显示微软脚本编辑器，用来生成网页脚本。

命名宏

在“录制宏”对话框中的“宏名称”栏中输入一个名称。该名称：

- ◆ 必须以字母开头，后面可以跟有字母和数字；
- ◆ 可以长达80个字符；
- ◆ 可包括下划线，用来隔开字符；
- ◆ 不能包括空格，标点符号和特殊字符(例如：!或*)。

(对如何命名宏的建议见下面的补充资料。)

必须对宏命名并加上说明

若生成了许多宏，应认真管理，了解什么该保留，什么该舍弃；录制宏可以很快生成宏，但不易管理；

工作匆忙时或者正在解决一个问题的时候容易忘记写上说明，而且，不能肯定哪个宏应当保留。即使如此，应当给出几个字的说明。否则，结果往往会出现一大堆录制的宏，根据名称却无法辨认，也没有说明。要搞清楚每个宏做什么，哪个宏可以放心删去，就必须研究每个代码，而录制的宏可能十分长（即使相关操作仅仅是在对话框中做几个选择）。

使用宏命名方法可以标明哪些宏可以删去而不会出错。名称开始用固定字符，后面加序号以表示不同的版本。例如，草案（草案 01，草案 02，等等）、临时（临时 01，临时 02，等等），甚至可以用 aaa（可以保证在宏对话框中排在前面）。千万不要使用 VBA 提供的默认名称（如：Macro1，Macro2，等等，则序列号加上 Macro），除非希望用该名称录制临时宏。宏名称不明确，事后就很难搞清宏的作用了。

VBA 代码增加了文件的数据量，最好将不用的宏删去，这样做可以防止文件变得过大，从而降低运行速度。

微软公司的软件不会阻止在“录制宏”对话框的“宏”名称栏中输入无效的名称。但是当单击“确定”按钮开始录制时会给出提示。图 1.3 给出 Word、Excel 和 PowerPoint 对无效宏名称的提示。

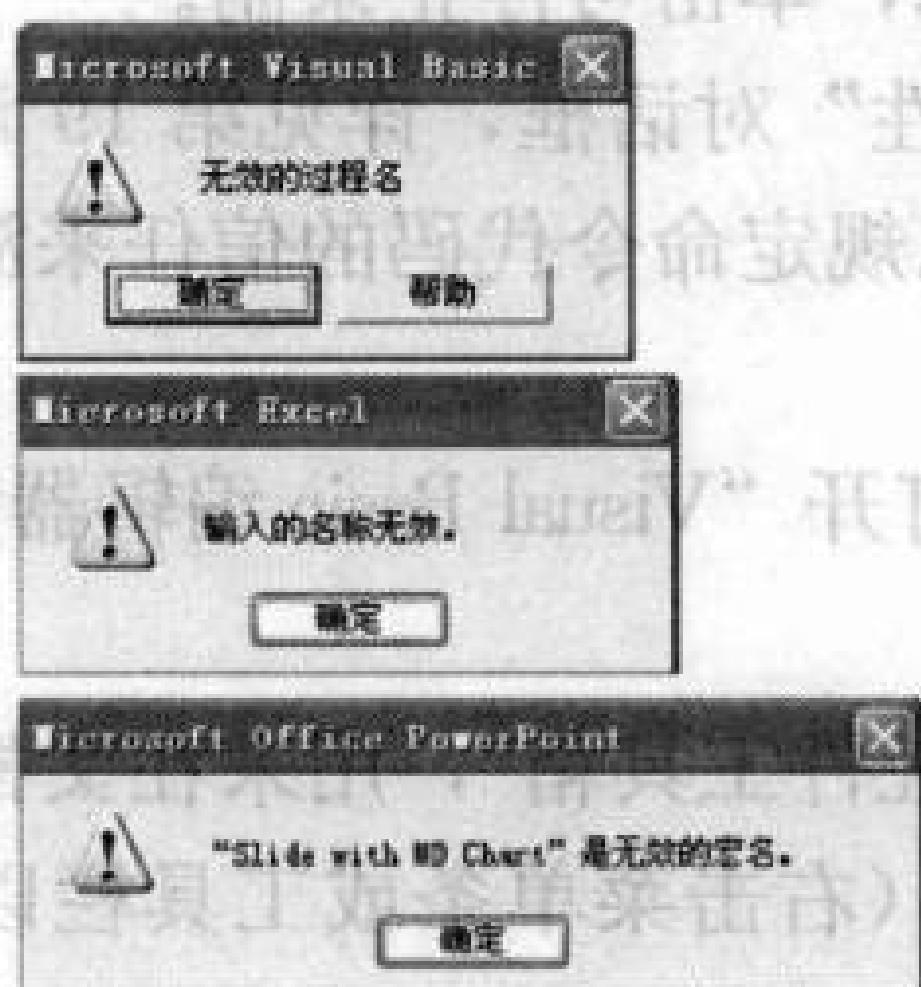


图 1.3 若在“录制宏”对话框中输入不合要求的名称，在单击确定时，软件会给出提示。Word 软件的对话框信息没有什么帮助，只显示了微软公司 Visual Basic 对话框标题、Excel 和 PowerPoint 给出软件自己的名称，提示信息也很有帮助

在“说明”栏中输入说明，该说明有助于识别宏以及宏的用途。若宏只有在特定的条件才能使用，应在说明栏中简要注明。例如：某个 Word 软件的宏在使用前需要在文档中进行选择，应把要求注明。如果宏对 PowerPoint 软件中的当前演示文稿进行格式调整，也应当说明。

另一种命名的考虑：关于屏幕提示

若在办公软件中创建一个宏的工具栏按钮，软件会自动为按钮给出包括宏名的屏幕提示。默认的屏幕提示似乎要求命名一致性，或者至少应当复杂一些，例如：每个单词的第一个字母大写，或者在每个单词之前使用下划线就要比全用大写字母或全用小写字母更容易看懂。

Word 软件是个例外，它产生的屏幕提示比其他软件复杂。它自动在每个大写字母前加空格，这时大写字母被判断为一个单词的开始。例如，若宏命名为 FormatDailyReport（对

每天的报告进行格式处理)并生成工具栏按钮,Word软件将给出的屏幕提示为Format Daily Report。如果名称中包括一组大写字母,Word软件将该组大写字母的第一个字母前加空格而不会再分隔。例如,若宏名为FixTCPIPS,Word给出的屏幕提示为Fix TCPIPS,而不是人们希望的Fix TCPIP Settings。在这种情况下,最好使用下划线:FixTCPIP _ Settings,这样屏幕提示的后半部分就会分开,容易读懂。

选择宏的存放位置。Word,Excel和PowerPoint软件中的情况分别如下:

◆在Word软件中,若希望限制宏在当前的模板或文档中使用,应当在“将宏保存在”

下拉列表中选择模板或文档。若希望宏用于任何模板,必须使所有文档(Normal.dot)出现在“将宏保存在”下拉列表中。(若读者不了解什么是Word软件的模板以及其作用,可参阅附文“了解Word软件的Normal.dot、模板和文档”。)

◆在Excel软件中,可选择把宏存放在当前工作簿、新工作簿或个人宏工作簿中。个人

宏工作簿是一个特殊工作簿,称做PERSONAL.XLS,保存在%userprofile%\Application Data\Microsoft\Excel\XLSTART\目录中,在第一次将宏保存在个人宏工作簿中时,Excel生成个人宏工作簿。若在个人宏工作簿中保存宏以及其他程序,任何情况都可使用该宏。这样说来,个人宏工作簿类似于Word软件的Normal.dot文档。若选择新工作簿,Excel将生成新工作簿,并在其中产生宏。

注意:%userprofile%是Windows环境变量。将含有用户资料的路径存入目录,用

户资料包括详细的目录和文件,含有对Windows和应用软件的设定。环境变量是个容器,含有很容易查找的一条信息。例如,%userprofile%可以进入用户资料目录,无论资料存在哪个驱动器上。

◆在PowerPoint中,可将宏存在当前的演示文稿上,或者任何一个打开的演示文稿以及模板上。PowerPoint不提供公共的宏存储点,尽管宏可以用于别的演示文稿,只要把某个演示文稿或模板作为公共存储地并始终打开(或者希望使用宏的时候打开)即可。

针对特定的文档、模板、工作簿或演示文稿,Word、Excel和PowerPoint都提供了默认地点自动存放录制的宏。

◆Word软件在选定的模板或文档中把宏存在NewMacros的模块中,因此总是可以得知宏存在何处。若模块不存在,宏录制器可以创建。宏录制器把文档或模板的每个宏都录制下来,若录制宏太多,NewMacros模块很快变大。NewMacros模块在默认的通用模板Normal.dot中很有可能膨胀,因为若不指定另一个文档或模板就会在此一直录制下去。最好经常清除NewMacros,把希望保留的录制宏存入另一个模块,清除所有无用的宏。

◆Excel和PowerPoint将录制宏按阶段存在新模块中,名称为Module_n,n表示最小的数字序列(Module1,Module2,等等),在下一个阶段中将生成的宏存入新的模块,用新序列号标识。如果经常在Excel或PowerPoint软件中录制宏,就特别需要把宏合并起来,否则会出现许多模块。

了解Word软件的Normal.dot、模板和文档

在Word文档中,存储宏和VBA代码时往往不知所措,因为必须在几个地点中进行选择,因此应当了解Normal模板(Normal.dot),以及其他模板和文档之间的关系。

Word 软件有四层结构，每一层会影响 Word 软件的外观和功能，然而，并不是全部的四层结构必须同时使用。

最低层是 Word 应用层，必须使用，包括了所有的 Word 对象和内置命令。应用层包括的界面对象包括菜单、工具栏，等等。应用层最难说明，因为不能直接看见，可看到的是 Normal.dot，即通用模板，这是第二层，也必须使用。

启动 Word 软件时，自动加载 Normal.dot，Normal.dot 始终存在直到退出 Word 软件（可使用特别开关阻止 Normal.dot 加载，即 Winword/n，若要查寻错误可以这么做）。Normal.dot 包括样式、自动文本、自动纠错以及自定义信息，这些自定义信息除非刻意排除，否则会在别的层中显现出来。

默认的文档（指启动 Word 软件出现的文档以及单击标准项栏上新文档按钮时出现的任何文档）以 Normal.dot 为基础。因此，使用默认的空白文档，出现的界面就是在 Normal.dot 中设定的界面。

当前的模板在 Word 应用层和 Normal.dot 层之上。该模板可以包括样式、自动文本、宏模块、自定义工具栏的各种设定，以及该文档需要的样板文本。这是第三层。不过，该层只有在当前文档（或者活动的文档）与 Normal.dot 以外的文档相结合时才会使用。

在当前模板之上是当前文档，包括文档中的字、图形、格式和外观。文档还可包括宏模块、自定义工具栏、自定义菜单、自定义快捷键，因此，文档本身为第四层。该层在某个文档打开时始终存在，然而对 Word 界面或功能无任何影响，除非该文档包括自定义的内容。

自定义的内容自顶层向下起作用。因此，在当前文档中的自定义内容优先于当前模板的自定义内容，而当前模板的自定义内容优先于任何 Normal.dot 以外的通用模板或附加项目。这些通用模板或附加项目中的自定义内容优先于 Normal.dot 中的自定义内容。因此，如果从 Normal.dot 中的菜单栏中移除表菜单，连接其他模板的文档也会看不到，除非存储在这些模板中，这时候，该设定优先于 Normal.dot 中的设定，因为这些文档是以该模板为基础的。

再举一例，如果把组合键功能 Ctrl+Shift+k 安排在 Normal.dot、加载的通用模板、文档模板，以及文档本身的不同程序中，按下组合键时，只能在文档内的程序中起作用，因为这是最顶层。如果把文档中的组合键移去，模板为顶层，那么将在模板中的程序中起作用。如果也从文档中移去组合键，那么只在加载的通用模板中的程序中起作用。最后，如果也移去这层组合键，在 Normal.dot 中的程序中才会起作用。

Word 软件可以加载 Normal.dot 以外的两个或更多的通用模板。如果加载多个通用模板，将按字母顺序排列。就是说，通用模板 Alpha Global.dot 和 Beta Global.dot 加载后，Beta Global.dot 中的自定义内容优先于 Alpha Global.dot 中的自定义内容。

设定运行宏的方式

这时候，当命名了宏以后，给出了说明，并选择了宏的存放位置后，Word 和 Excel 还要求你选择运行宏的方式：键盘快捷方式（Word 或 Excel）或者命令工具栏（在 Word 中）。如果录制少量的宏不做修改，也不在模块中复制使用，这一特色是很方便的。然而，如果自己生成许多宏，并且要进行编辑而且在模块中复制时使用该特色的帮助不大，这是因为把宏由一个模块移到另一个模块就改变了事先设定的运行方式。

注意：命令栏是菜单，工具栏或隐藏菜单（指在程序中右击对象可以显现的菜单）。

在具体的应用中，可以自定义所有或部分命令栏。

这样的限制意味着如果打算按录制的形式和默认的位置使用宏，设定宏的使用方法是符合常理的。如果移动了宏或重新命名，就不要立刻设定宏的使用方法，应当等到宏的形式和位置最终完成，然后再设定其运行方式。详见本章后面设定宏的运行方式。

提示：把录制的宏移动到其他模块中可以重新组织宏，从而比较代码，进行调试或者给别人使用。

设定宏的运行方式时，在Word、Excel、PowerPoint中都需要按下面的提示来做。

指定在Word中运行宏的方式

在Word中使用“自定义”对话框以及“自定义键盘”对话框设定宏的运行方式。下面介绍用“自定义”对话框将宏设定到工具按钮、菜单，或者快捷菜单。

1. 在“录制宏”对话框中，单击“工具栏”按钮，显示“自定义”对话框。
2. 如果不能显现，选择“命令”选项卡，以显示“命令”页（如图1.4所示）。在“类别”清单中，只列出“宏”的类别，而且必须选中。
3. 必须保证Word在“自定义”对话框的底部“保存于”下拉列表框中选择了正确的自定义内容（即自定义涉及的范围），可将自定义用于默认的全局模板（Normal.dot）、当前的文件或者和当前文件关联的模板（如果当前文件和Normal.dot关联，在“保存于”组合框中就不会再有其他模板）。
4. 如果将宏设定到工具栏按钮或快捷菜单项中，必须保证工具栏或快捷菜单工具栏显示出来，要显示工具栏或快捷菜单工具栏，单击“自定义”对话框中的“工具栏”选项卡，以显示“工具栏”页，然后选择工具栏或“快捷菜单工具栏”的复选框。单击“命令”选项卡，再次显示“自定义”对话框的“命令”页。
5. 在“命令”列表框中，单击宏的名称，并将宏拖入工具栏、快捷菜单，或者相应的菜单中，详见图1.5。
6. 快捷菜单选项，然后将宏项目从命令页中拖入相应的快捷菜单，Word给宏增加一个按钮或者菜单项目，并给出宏的全名，例如Normal.NewMacros.CreateDailyReport。该名称由保存宏的模板或文件的名称、包含宏的模板的名称和宏的名称组成。
7. 要对按钮或菜单项目重新命名，先右击（或者单击“自定义”对话框中的“修改选择”按钮），在快捷菜单的“命名”栏中输入自己喜欢的名称，详见图1.6。

提示：有两点需要记住，一是宏的菜单项目名称或者按钮名称和宏的名称不需要有任何关联；二是如果需要，也可以生成新的工具栏和新的菜单。

8. 要对某个项目使用快捷键，将字符（&）放在对应快捷键的字母之前，然后按Enter键。

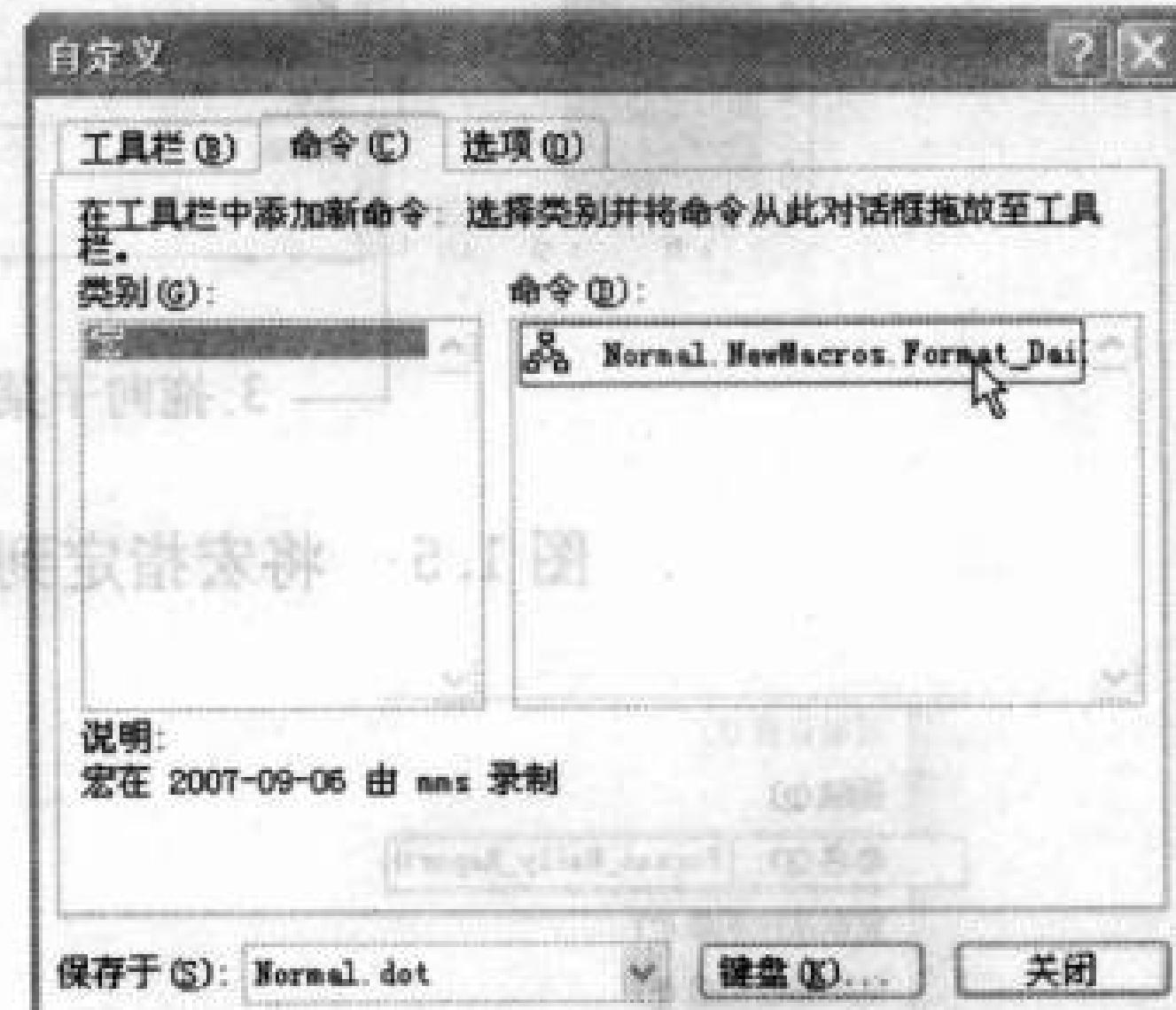


图1.4 在“自定义”对话框中选择宏的运行方式



图 1.5 将宏指定到快捷菜单，选择自定义对话框命令页的



图 1.6 Word 针对菜单项目或者工具栏按钮给出宏的全名。使用“修改选择”命令可使名称变短，方便使用

提示：快捷键并不需要唯一，然而唯一之后可以方便使用。如果多个菜单或者命令使用同样的快捷键，Word 将选择最先使用的一个项目。可按下 Enter 键，以显示该菜单或执行该命令，也可再次按快捷键调用与之相关的下一个项目。例如，如果将快捷键 T 用于命名为 Transpose _ Word 的宏，Word 在第一次按下 Alt+T 键时选择工具菜单（如果没有被移除），然后再按下 Alt+T 键时选择 Transpose _ Word 按钮。

9. 单击“关闭”按钮，关闭“自定义”对话框。将宏指定到组合键的方法如下：

(1) 单击“录制宏”对话框中的“键盘”按钮，以显示“自定义键盘”对话框。

注意：同运行宏的其他方式一样，可以在录制宏的时候，也可以在完成了录制宏之后的任何时候，指定某一组合键来运行宏。如果打算将宏从 NewMacros 模块移到另一个模块，那就只能在宏到达它的最终目的地之后，才可以将其指定到组合键。

(2) 以单击来定位“请按新快捷键”框内的插入点，再按下想要输入的组合键。图 1.7 表示出带有已选择的新快捷键的“自定义键盘”对话框。组合键可以是：

◆ Alt键加上任一功能键或没有用做菜单访问键的任一通用键

◆ Ctrl键加上任一功能键或任一通用键

◆ Shift键加上任一功能键

◆ Ctrl+Alt键, Ctrl+Shift键, Alt+Shift键, 或是Ctrl+Alt+Shift键加上任一通用键或功能键。但是, 频繁按下Ctrl+Alt+Shift键和另一个键是很不方便的。

(3) 检查一下“当前键”列表框, 确认选择的组合键不在已用之列。如果是已用的, 那就要按Backspace键将当前组合键清除掉(除非打算重新指定组合键), 再按入另一组合键。

提示: 可以采用两步法来设定快捷键——例如, Ctrl

+Alt+F键, 1和Ctrl+Alt+F键, 2。方法是: 先按下组合键, 再按第二个键(此例中是1或2)。但是, 这种快捷键设定方式比较麻烦, 有些失大于得, 除非是在逐个指定数百个额外的快捷键。

(4) 单击“指定”按钮, 以便将组合键指定给宏。

(5) 单击“关闭”按钮, 以关闭“自定义键盘”对话框。

指定在Excel中运行宏的方式

当录制宏时, 在Excel中要做的, 只是指定一个Ctrl快捷键来运行它。如果想增添菜单项或工具栏按钮, 必须在录制宏完成之后进行增添。因为极有可能想把宏从VBA自动保存它的模块中移至另一个不同模块之内, 这样做也许是件好事。

指定Ctrl快捷键来运行所录制的宏:

1. 在“快捷键”文本框内输入该键。如果希望快捷键中包括Shift键, 则按下Shift键。

2. 在“将宏保存在”下拉列表中, 指明想让宏录制器把宏保存在什么地方。可用选择如下:

◆ 当前工作簿。它将宏保存进活动工作簿内。这一选择对于专属某个特定的工作簿而其他地方不用的宏是很有用的。

◆ 新工作簿。它会引起Excel创建一个新的工作簿, 以便将宏存入其中。这一选择对于试验性的宏是很有用的, 在将这些宏投入到实际工作中去之前, 需要对它们进行编辑。

◆ 个人宏工作簿。将宏保存在名为PERSONAL.XLS的特殊工作簿内, 而该工作簿保存在\Application Data\Microsoft\Excel\XLSTART\文件夹之内。把宏和其他自定义内容保持在“个人宏工作簿”里, 就能够使这些宏为所有过程所用——这样一来, “个人宏工作簿”类似于Word里的Normal.dot。如果还没有“个人宏工作簿”的话, 宏录制器会自动创建一个。

3. 单击“确定”按钮以启动录制宏。

指定在PowerPoint中运行宏的方式

当录制宏时, PowerPoint不能指定运行宏的方式, 但用户可以在以后指定运行宏的方



图1.7 在“自定义键盘”对话框内, 为宏设定快捷键

式。在本章后部分“指定运行宏的方式”一节中会加以讨论。可以选择将宏保存在任何打开的演示文稿或模板中。

在宏里录制行动

当从“自定义”对话框、“自定义键盘”对话框或“录制宏”对话框退出时，“宏录制器”准备启动录制宏。对于大多数应用程序，“宏录制器”显示“停止录制”工具栏（通常出现在屏幕的左上角）；对于 Word，“宏录制器”会在鼠标指针上加上一个盒带图标，并使状态栏上的 REC（录制）指示器从默认的灰色转变成黑色。

现在来实施想要录制的一系列行动。用户能做的事，视应用程序的不同而有所不同，但是，一般来说，可以使用鼠标来选择菜单和工具栏的项目，在对话框中做出选择，也可以在文档中选择指定的项目（例如，工作表中的单元格，或者 PowerPoint 幻灯片中的各形状）。但是，有很多事情用鼠标是做不了的。例如，在 Word 的文档窗口之内选择项目。为了在文档窗口内选择项目，必须使用键盘命令。

注意：当在对话框中做出选择并单击“确定”按钮时，“宏录制器”将对话框的那一页上的所有选项的当前设置录制下来。例如，当在 Word 的“段落”对话框上改变某一段落的左缩进时，“宏录制器”会把对缩进和间距的所有其他设置（对齐，段前和段后间距等）都录制下来。如果不使用它们，以后编辑时可以剔除表示这些设置的代码。

在 Word 中，如果必须实施某些行动，但又不想把它们录制下来的话，可以单击“停止录制”工具栏上的“暂停录制”按钮，使“宏录制器”暂停。“暂停录制”按钮是一种推入式按钮，即也可用做“恢复录制”按钮。再次单击这个按钮后，录制又恢复进行。

要想停止录制，选择“工具”>“宏”>“停止录制”（有的应用程序里是“停止录制器”），或者单击“停止录制”工具栏上的“停止录制”按钮（如果显示了这一工具栏的话）。在 Word 中，也可以双击状态栏上的 REC 指示器以停止录制。

现在，宏录制器已经录制好了宏，并将它指定到所选择的控件。如果没有指定控件，可以在以后再指定（本章后面还要讨论）。

运行宏

运行已录制的宏，可以使用下面的任何一种方法：

- ◆ 如果为运行宏指定了控件，则可：单击工具栏按钮，选择菜单项或上下文菜单项，或按下组合键。
- ◆ 如果未指定控件，则按下 Alt+F8 键，或选择“工具”>“宏”>“宏”以显示“宏”对话框，选择宏之后，单击“运行”按钮（也可以双击列表框中的“宏名”）。

注意：也可以通过单击 Visual Basic 工具栏上的“运行宏”按钮（带有“play”符号的按钮）来显示“宏”对话框。还可以从 Visual Basic 编辑器上运行宏，当正在这个编辑器上工作时，这样做是很有用的。

宏运行时，实施录制好的一系列行动。例如，假设在 Excel 里创建了一个宏，它选择当前工作表里的单元格 A2，在该单元格内使用粗体字，输入“年销售额”文本，再选择单元

格B2，在该单元格中输入数字100000。宏录制器会把这五项行动登记下来：选择单元格A2，使用粗体字，输入文本，选择单元格B2，输入第二份文本。以后，每次运行宏时，VBA逐次实施这五个行动。

提示：要停止运行宏，需按下Ctrl+Break键，VBA会停止运行代码并显示一个对话框，告诉用户代码执行已经停止。单击“结束”按钮可从对话框退出。

在宏停止运行之后，某些应用程序（如Word）会让用户撤销通过VBA执行的大多数行动（使用“编辑”>“撤销”菜单项或标准工具栏上的“撤销”按钮，每一次撤销一条命令）；其他应用程序则不会如此。

注意：如果运行宏出现了错误，有可能是宏试图对不可用的文件或对象进行操作。例如，如果在Excel里录制了一个宏，这个宏是在活动工作簿上工作的，但当运行宏时工作簿并未打开，宏就会引起错误。同样，如果在PowerPoint里录制了一个宏，它针对活动幻灯片上第三个形状进行工作，但当运行宏时幻灯片上此形状并不存在，宏就会出错。为使宏正确运行，需重新创建它所需要的条件，然后再次进行尝试。

在Word中录制样本宏

本节将要在Word中录制一个以后会用到的样本宏。这个宏是选择一个当前字，剪切它，将插入点右移一个字，再将字粘贴回去。这个宏是一系列简捷行动，以后在“Visual Basic编辑器”里，还会对它查看和进行编辑。

遵循以下步骤来录制这个宏：

1. 如果没有新的已打开文档（或者有一个文档，但并不关心该文档），则创建一个新文档。单击标准工具栏上的“新建空白文档”按钮。
2. 双击状态栏上的REC指示符，或选择“工具”>“宏”>“录制宏”。无论用哪种方法，Word都会显示出“录制宏”对话框。
3. 在“宏名”文本框内，输入Transpose_Word_Right。
4. 在“将宏保存在”下拉列表中，确认“所有文档(Normal.dot)”已被选择，除非想把宏指定到另一个不同模板（本例假设宏在Normal.dot中，而且，如果已将它置于别处，用户自会考虑其后果）。
5. 在“说明”框中，输入对该宏的说明（如图1.8所示）。“说明”框内可包括诸如“宏在2007-09-07由mms录制”之类的话，这是Word力图帮助使用人员以后能够识别这个宏。要想更清楚些的话，也可以输入说明为“将当前的文本移至文本的右侧，宏在2007-09-07由mms创建”。

提示：以后，可以在“宏”对话框里或在“Visual Basic编辑器”里对该宏的说明做改动，

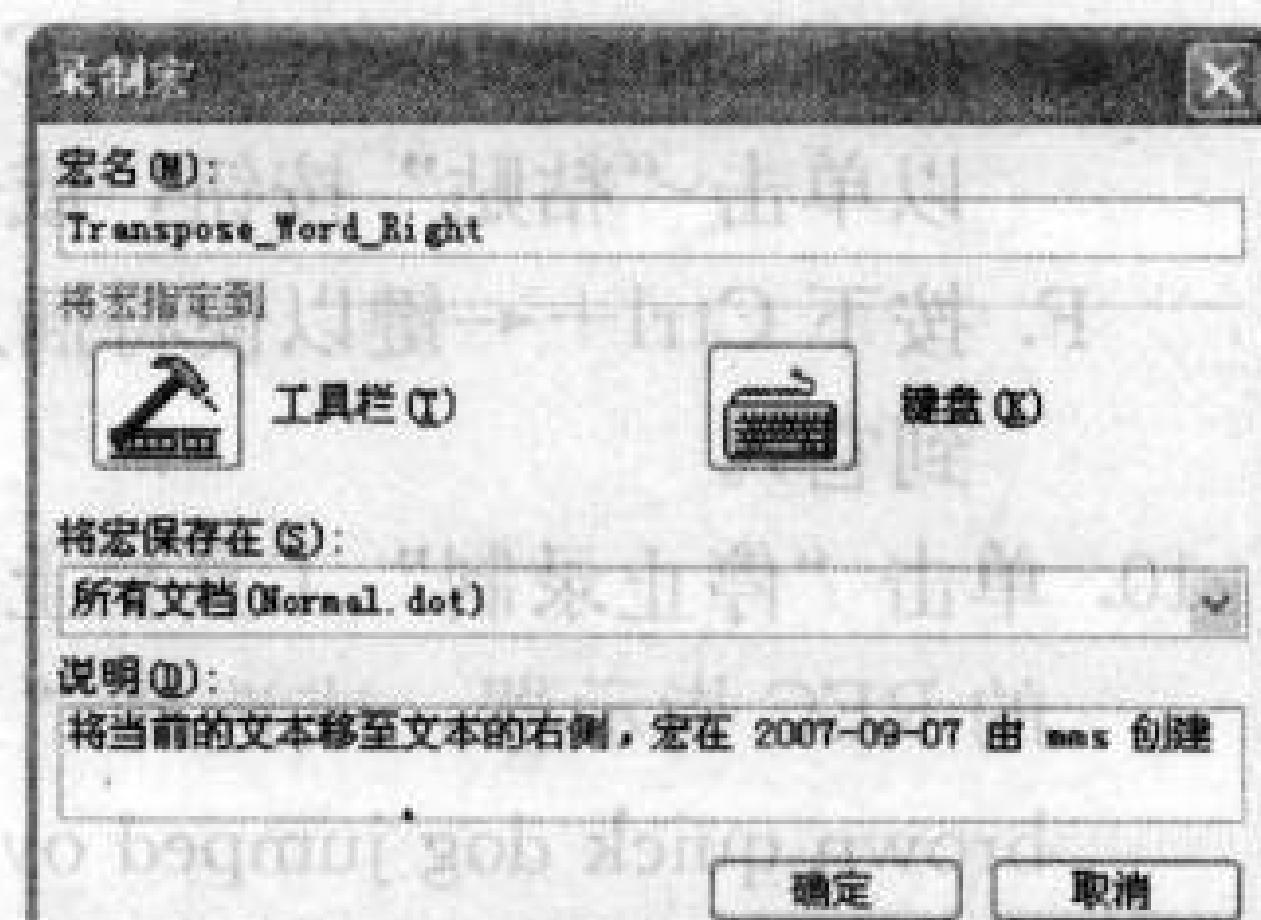


图1.8 在Word中创建样本宏

但是，最好在录制宏时首先就输入适当的说明。如果想在创建宏之后再来说明宏，很可能会忘记这样做。结果可能是，做完了几十个宏，创建它们时宏名很清楚，时间长了，对这些宏的功能却感到茫然。

6. 如愿意的话，照前节所述，指定一个运行宏的方式。创建一个工具栏按钮，一个菜单项，或一个上下文菜单项，或指定一个快捷键方式（选择何法纯属个人爱好）。如果以后会把这个宏移至另一不同模块（或另一不同模板、文档）的话，此时不要指定运行宏的方式。
7. 单击“关闭”按钮，从“自定义”对话框或“自定义键盘”对话框退出（如果选择不指定运行宏的方式，则可单击“确定”按钮从“录制宏”对话框退出）。现在已经准备好了录制宏。“停止录制”工具栏出现在屏幕上，而且鼠标指针有一个盒带图标附在其上。



8. 作为暂停录制的简捷说明，单击“停止录制”工具栏上的“暂停录制”按钮时，盒带图标从鼠标指针上消失，同时，“暂停录制”按钮变成了“恢复录制”按钮。在文档中输入一行文本：The quick brown dog jumped over a lazy fox. 将插入点定位于 quick 这个字的任何地方，再单击“停止录制”工具栏上的“恢复录制”按钮，就能重新激活宏录制器。

9. 录制该宏的操作如下：

- A. 连按 F8 键两次，从而使用“扩展选定范围”特性来选择 quick 这个字。状态栏上的 EXT（扩展）指示器会变黑，以显示扩展方式已激活。
- B. 按下 Escape 键以取消扩展方式。状态栏上的 EXT 指示器会再次变淡（对于该宏，此步不是绝对必要的，但还是实施为好，以便宏录制器录制它）。
- C. 按下 Shift+Delete 键或 Ctrl+X 键，以便将所选的字剪切到剪贴板（也可以单击“剪切”按钮，或选择“编辑”>“剪切”）。
- D. 现在插入点位于 brown 这个字的起始处。按下 Ctrl+→ 键以便将插入点右移一个字，让它位于 dog 这个字的起始处。
- E. 按下 Shift+Insert 键或 Ctrl+V 键，以便将被剪切的字从剪贴板粘贴进来（也可以单击“粘贴”按钮，或选择“编辑”>“粘贴”）。
- F. 按下 Ctrl+← 键以便将插入点左移一个字（这是一条附加指令，在编辑宏时会用到它）。
10. 单击“停止录制”工具栏上的“停止录制”按钮，以停止录制宏（或双击状态栏上的 REC 指示器，或选择“工具”>“宏”>“停止录制”）。句子现在读为“The brown quick dog jumped over a lazy fox.”

现在，可以使用指定过的（如果选择指定某一个的话）工具栏按钮、菜单或上下文菜单项，或者快捷键方式，来运行这个宏。也可以选择“工具”>“宏”>“宏”，从“宏”对话框来运行这个宏。可以试一试把插入点定位在 brown 这个字再运行宏，这样可以把句子

中的各个字恢复成最初的次序。

提示：按下 Shift 键不放并单击“文件”菜单，会使“全部保存”这一项目出现在“保存”项目的位置上，以及“全部关闭”这一命令出现在“关闭”项目的位置上。“全部保存”这一命令，对于迫使 Word 保存针对模板的更改和针对活动文档的更改特别有用。

此时，Word 已将宏保存进 Normal.dot，但还没有将更改保存起来。如果在退出 Word 之前仍不保存这些更改，Word 会提示保存。但是，为了避免一旦 Word 或 Windows 崩溃时丢失宏，最好是现在就保存 Normal。为此需按下 Shift 键不放，单击“文件”菜单，再单击“保存全部”这一项。如果 Word 提示要保存更改的话（如图 1.9 所示），应单击“是”按钮。此例中的“更改”是指已在 Normal 模板中创建的宏。

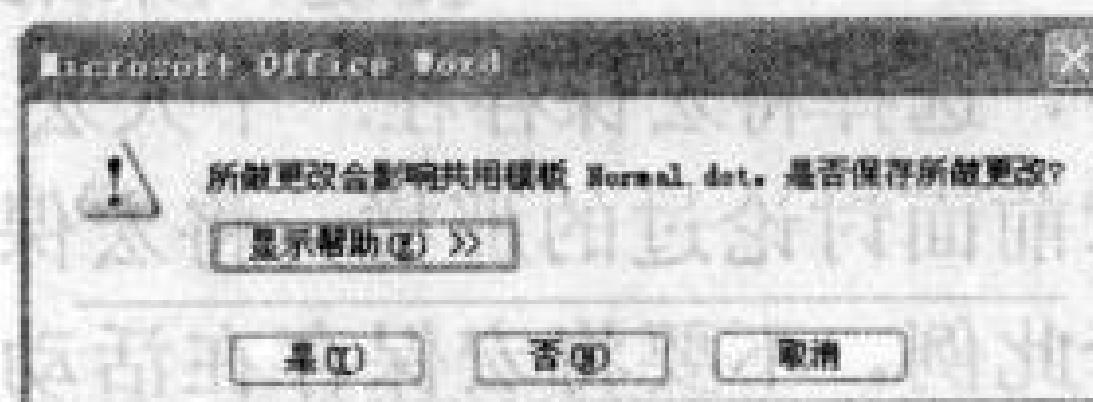


图 1.9 当在模板中创建宏之后，Word 可能会提示保存针对 Normal 模板的改动

注意：如果愿意让 Word 自动地将针对 Normal 模板的更改保存起来而不必提示，可选择“工具”>“选项”以显示“选项”对话框，单击“保存”选项卡，清除掉“提示保存 Normal 模板”复选框，然后单击“确定”按钮。

在 Excel 中录制样本宏

本节将要在 Excel 中录制样本宏。这个宏创建一个新工作簿，将月份顺序输入工作簿中并保存。在第 3 章中会用到这个宏。

创建一个个人宏工作簿，如果还没有它的话

如果还没有在 Excel 中创建“个人宏工作簿”，在创建这一过程之前，必须要创建一个“个人宏工作簿”。遵循如下步骤：

1. 选择“工具”>“宏”>“录制新宏”，以显示“录制宏”对话框。
 2. 接受针对此宏的默认名，因为很快就会删除它。
 3. 从“将宏保存在”下拉列表中，选择“个人宏工作簿”。
 4. 单击“确定”按钮以从“录制宏”对话框退出，并开始录制宏。
 5. 在任何一个已激活的单元格中输入一个字符，并按 Enter 键。
 6. 单击“停止录制”工作栏上的“停止录制”按钮，以停止录制宏。
 7. 选择“窗口”>“取消隐藏”，以显示“取消隐藏”对话框。选择 PERSONAL.XLS 并单击“确定”按钮。
 8. 选择“工具”>“宏”>“宏”，以显示“宏”对话框。
 9. 选择录制的宏并单击“删除”按钮以删除它。在确认消息框内单击“是”按钮。
- 现在有一个可供使用的“个人宏工作簿”了。

在 Excel 中录制样本宏

为了录制这个宏，启动 Excel 并遵循如下步骤：

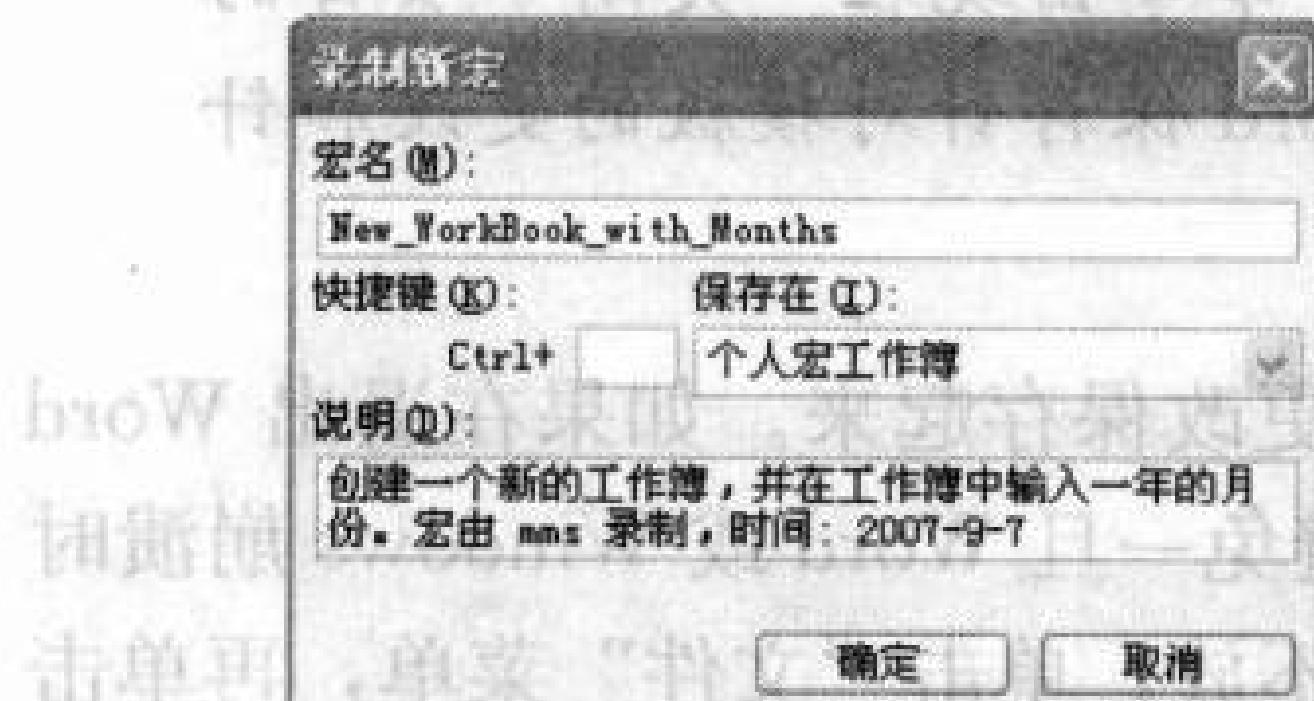


图 1.10 显示针对 Excel 的“录制宏”对话框，可在框中做出选择

1. 选择“工具”>“宏”>“录制新宏”，以显示“录制宏”对话框，如图 1.10 所示（图中含有已输入信息）。

2. 在“宏名”文本框内输入宏的名称，New_Workbook_with_Months。

3. 如想使用快捷键的话，在“快捷键”文本框内，输入一个快捷键（以后可以更改快捷键，所以此时并不一定要输入快捷键）。

4. 从“保存在”下拉列表中，选择将宏保存在“个人宏工作簿”内，或活动工作簿内，或新工作簿内。正如本章前面讨论过的那样，将宏保存在“个人宏工作簿”内，可提供最大的灵活性。对于此例，不要将宏保存在活动工作簿内，因为很快就将删除活动工作簿。
5. 在“说明”文本框中，输入对该宏的说明。
6. 单击“确定”按钮以从“录制宏”对话框中退出，并启动录制宏。
7. 选择“文件”>“新建”以显示“新建工作簿”任务窗格，然后单击“本机上的模板”以显示“模板”对话框。（在 Excel 2000 和更早的版本上，选择“文件”>“新建”以显示“新建”对话框。）
8. 在对话框的“常用”页上选择“工作簿”项目，然后单击“确定”按钮。Excel 创建出一个新工作簿并选择簿上第一张工作表。
9. 单击单元格 A1 以选中它（即使它已被选中，仍需单击它，因为必须录制此指令）。
10. 输入 January 2006，并按→键以选择单元格 B1。Excel 自动地将日期变化成默认的数据格式。这就好了。
11. 输入 February 2006，并按←键以再次选择单元格 A1。
12. 从单元格 A1 拖曳到单元格 B1，以使两单元格都被选中。
13. 将填充“手柄”从 B1 拖曳到 L1，这样，利用 Excel 的自动填充特性，就可将 March 2006 至 December 2006 各月份输入到这些单元格中。
14. 单击标准工具栏上的“保存”按钮（或选择“文件”>“保存”），以显示“另存为”对话框。将工作簿保存在合适的文件夹中（例如，My Documents 文件夹），Sample Workbook.xls 之类的名称之下。
15. 单击“停止录制”工具栏上的“停止录制”按钮，以使宏录制器停止（也可选择“工具”>“宏”>“停止录制”）。

关闭样本工作簿，使用 Windows Explorer 去导引并删除它。然后运行宏并观察有何事发生。（如果不删除此工作簿，当步骤 14 试图使用已有工作簿的相同名称保存新工作簿时，Excel 会提示用户是否决定覆盖它。）

在PowerPoint中录制标本宏

本节将在PowerPoint中录制样本宏。为这个宏创建一张新幻灯片，对幻灯片上某一部分进行移位和调整大小，在该部分中输入文本并对其设置格式。在第3章中会考察这个宏。为了创建这个宏，启动PowerPoint并遵循如下步骤：

1. 基于所选择的模板打开一个新的演示文稿。（如果在启动PowerPoint时，它已自动打开了一个新演示文稿的话，就可使用该文稿。）如果此时出现了“新幻灯片”对话框，需暂时退出该对话框。
2. 选择“工具”>“宏”>“录制新宏”，以显示“录制新宏”对话框，如图1.11所示（图中含有已输入信息）。

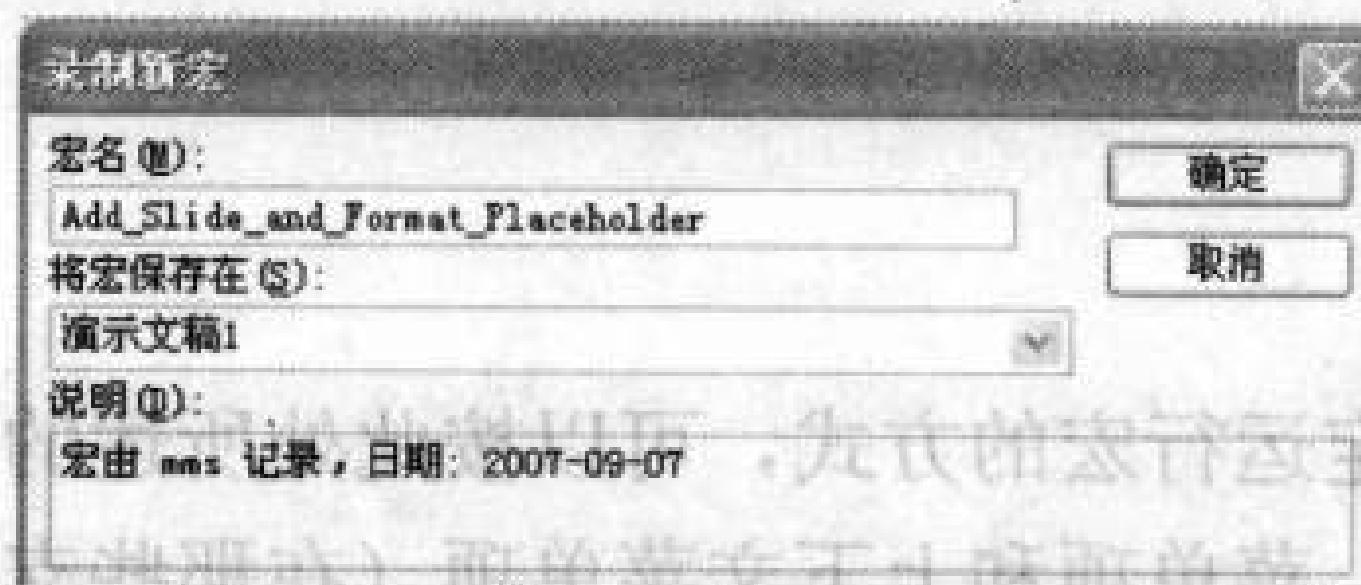


图1.11 显示针对PowerPoint的“录制宏”对话框，可在框中做出选择

3. 在“宏名”文本框内输入宏的名称：Add_Slide_and_Format_Placeholder。
4. 在“将宏保存在”下拉列表中，选择将宏保存在活动演示文稿中，不要保存在别的已打开的演示文稿或模板中。此下拉列表内的演示文稿，是按名称来识别的，而不是按诸如“这一文稿”或“活动文稿”之类的说明来识别的。默认情况下，PowerPoint选择将宏保存在下拉列表内的活动演示文稿之中，无需更改这一设定。
5. 在“说明”文本框内，输入对该宏的说明。
6. 单击“确定”按钮以从“录制新宏”对话框中退出，并启动录制宏。
7. 选择“插入”>“新幻灯片”，以便插入一张新幻灯片，并显示“幻灯片版式”任务窗格。

注意：在PowerPoint2000和更早版本上，选择“插入”>“新幻灯片”可以显示“新幻灯片”对话框。单击“标题幻灯片版式”，再单击“确定”按钮就可插入幻灯片。

8. 单击“标题幻灯片版式”，使其对幻灯片起作用，然后单击“关闭”按钮，以关闭“幻灯片版式”任务窗格。
9. 单击幻灯片上第一个占位符以选中它。
10. 在占位符的被选边界单击，并将它朝幻灯片顶端方向向上拖动，以移动该占位符。
11. 单击占位符底边中央处的调整大小手柄，并将它朝着较下的占位符的方向下拖。
12. 在该占位符内（在“添加标题”区）单击并键入文本：The quick brown dog jumped over a lazy fox。
13. 按住Ctrl键和Shift键不放，再按Home键，以选中占位符内全部文本。
14. 选择“格式”>“字体”以显示“字体”对话框。
15. 选择字体格式设置，例如字体为常规字体，或加粗字体，字号为54等。单击“预

览”按钮，以证实选择结果与调整大小后的占位符内部相适宜，且看上去很协调。

16. 单击“确定”按钮，以便从“字体”对话框退出。
17. 按→键使所选文本脱离选择。
18. 单击“停止录制”工具栏上的“停止录制”按钮，或选择“工具”>“宏”>“停止录制”，以停止录制宏。
19. 以 Sample Presentation.ppt 名称保存演示文稿。(必须保存它，以便在本书的后面部分能针对所录制的宏进行工作。)

现在，选择“工具”>“宏”>“宏”以显示“宏”对话框，选择宏，并单击“运行”按钮。VBA 会通过一系列操作再运行一遍，添加新幻灯片，对幻灯片上第一个占位符进行移位和调整大小，输入文本和为文本设置格式。

如对运行的宏感到满意，可在不保存更改的情况下关闭演示文稿。

指定运行宏的方式

如果在录制宏时没有指定运行宏的方式，可以按此处所述的方法来指定运行方式。

将宏指定到工具栏按钮、菜单项和上下文菜单项（在那些支持自定义上下文菜单的应用中）的过程几乎都相同，所以本节把它们放在一块研究，但请注意其不同之处。

将宏指定到工具栏按钮或菜单项

要将宏指定到工具栏按钮、菜单项或上下文菜单项（在 Word 或 PowerPoint 中），遵循如下步骤：

1. 右击任一已显示的工具栏或菜单栏，及从上下文菜单中选择“自定义”以显示“自定义”对话框。也可以选择“工具”>“自定义”以显示该对话框。
2. 如果“自定义”对话框的“工具栏”页没有显示出来，可单击“工具栏”标签使其显示。图 1.12 显示出了 Word 上的“自定义”对话框的“工具栏”页。Excel 和 PowerPoint 上的“自定义”对话框的“工具栏”页与此相似，只是没有“键盘”按钮。

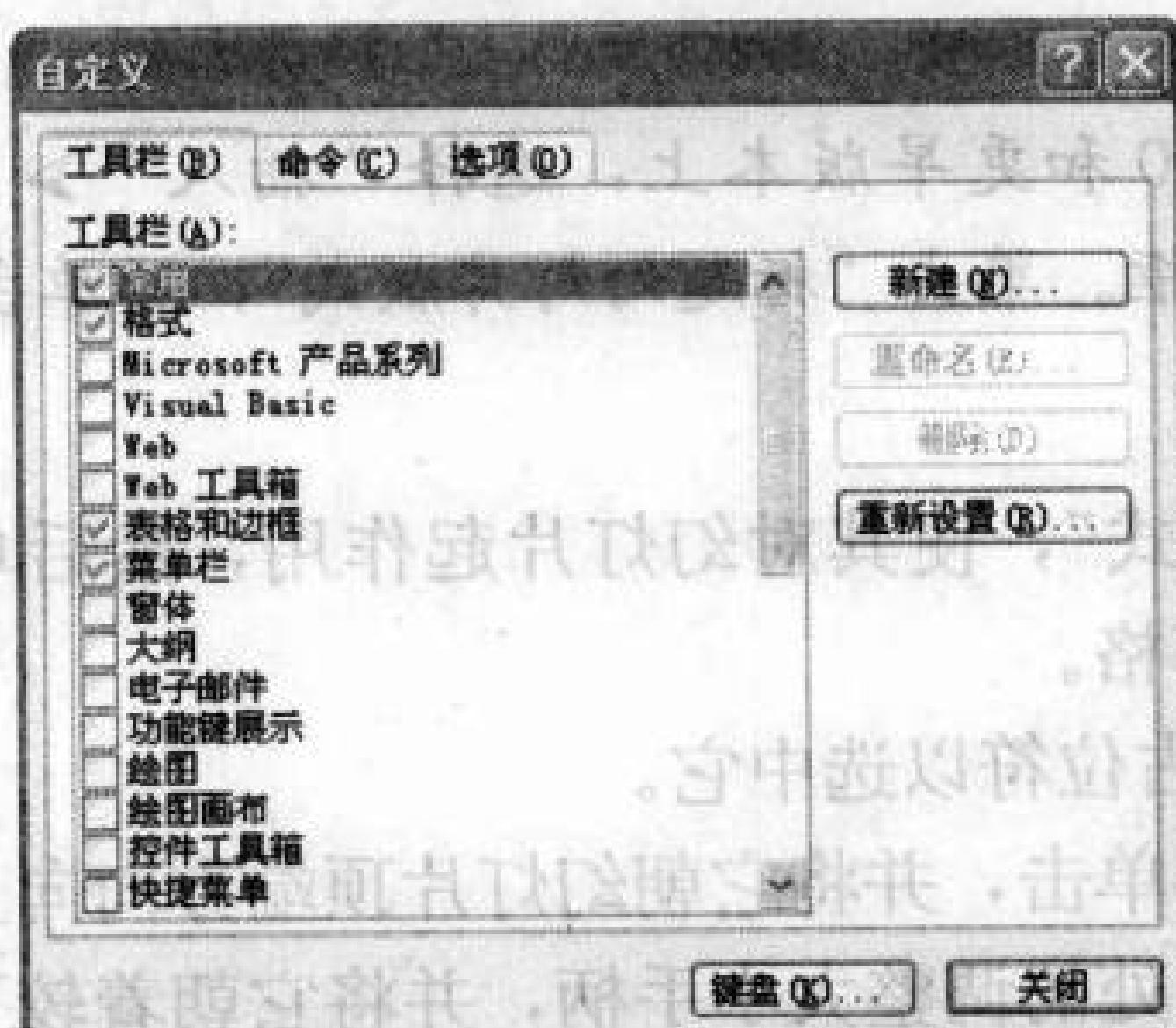


图 1.12 在“自定义”对话框的“工具栏”页，选择想要自定义的工具栏

3. 确认宏指定到的那个工具栏已显示出来了。
 - ◆ 要显示某一工具栏，需选中“工具栏”列表框中该工具栏的复选框。
 - ◆ 要隐藏某一工具栏，需清除该工具栏复选框的选择。

◆要在Word和PowerPoint的上下文菜单添加一个项目，需选中“快捷菜单”复选框。应用程序会显示一个带有若干按钮的工具栏，这些按钮引出对应不同种类工具栏的下拉列表。图1.13左边展示Word上的“快捷菜单”工具栏，右边则是PowerPoint上的“快捷菜单”工具栏。你能看到Word的“快捷菜单”工具栏（左边）和PowerPoint的“快捷菜单”工具栏（右边）。

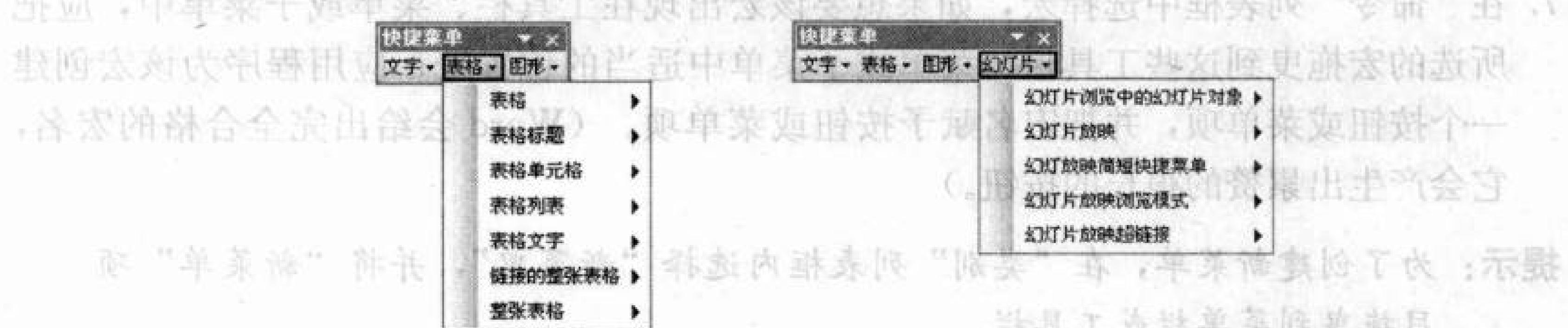


图1.13 在Word或PowerPoint的上下文菜单添加一项，显示“快捷菜单”工具栏，这里可看到Word的“快捷菜单”工具栏（在左边）和PowerPoint的“快捷菜单”工具栏（在右边）

◆要创建一个新工具栏，需单击“新建”按钮，以显示“新建工具栏”对话框（如图1.14所示，其中名称已输入），在“工具栏名称”文本框内输入与按钮对应的名称，然后单击“确定”按钮。在Word中，还要选择在何处保存新工具栏：是在Normal.dot中，在活动文档中，还是在附属于活动文档的模板中。与选择有关的提示参见步骤5。

4. 单击“命令”选项卡以显示“命令”页。图1.15显示出了Word中的“自定义”对话框的“命令”页，因为这一对话框的Word版本较其他应用程序的版本更复杂些。其他Office应用程序既没有“保存于”下拉列表，也没有“键盘”按钮。

5. 在Word中，确认“保存于”下拉列表正在显示把这个自定义命令保存进去的文档或模板。如不是的话，从下拉列表中选择出合适的文档或模板。

◆如果将自定义项保存于Normal.dot，即保存在全局模板，它可以为所有文档使用。

◆如果将自定义项保存于附属于活动文档的模板，自定义项可以为该模板所包括的所有文档使用。（如果Normal.dot作为活动文档的模板附属于该活动文档，“保存于”下拉列表就不显示出另外的模板。）

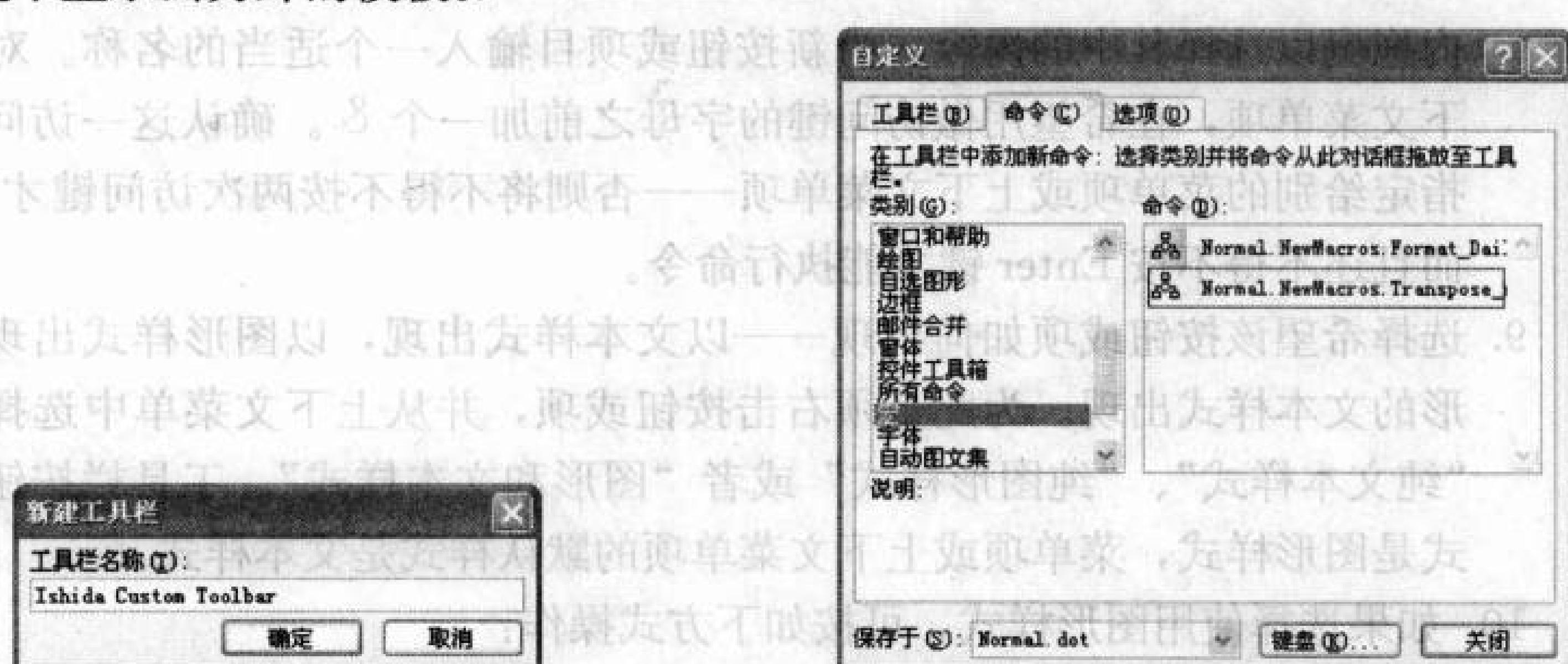


图1.14 创建已含有对应于宏的按钮的新建工具栏

图1.15 自定义对话框的命令页

◆如果将自定义项保存于活动文档，它只能为这个文档使用。

6. 在任何一种应用程序中，从“类别”列表框中选择“宏”项目，除了Excel之外，所有应用程序都会显示出可供使用的宏的列表。

警告：在某些场合，Word 会以这些宏的完全合格的名称来列出这些宏，名称中包括含有宏的项目（模板或文档）和模块，以及宏名。例如，有一个宏，名叫 Example_Macro，它保存在名为 Demos 的模块中，该模块又在 Normal 模板内，那么，这个宏可以列名为 Normal.Demos.Example_Macro。

7. 在“命令”列表框中选择宏，如果想要该宏出现在工具栏、菜单或子菜单中，应把所选的宏拖曳到这些工具栏、菜单或子菜单中适当的位置处；应用程序为该宏创建一个按钮或菜单项，并把宏名赋予按钮或菜单项。（Word 会给出完全合格的宏名，它会产生出累赘的很长的按钮。）

提示：为了创建新菜单，在“类别”列表框内选择“新菜单”，并将“新菜单”项目拖曳到菜单栏或工具栏。



图 1.16 在 Excel 中，使用“指定宏”对话框，以将宏指定到所创建的按钮或菜单项

◆在 Excel 中，将“自定义按钮”图标拖曳到合适的工具栏上的适当位置，或将“自定义菜单项”图标拖曳到合适的菜单。Excel 创建一个附有“笑脸”图标的自定义按钮，或创建一个附有文本“自定义菜单项”的自定义菜单项。此时，该按钮或菜单项与宏尚未连接。右击该按键或菜单项，及从上下文菜单中选择“指定宏”，可以显示出“指定宏”对话框，如图 1.16 所示。

◆为了在 Word 和 PowerPoint 中创建上下文菜单项，将宏项目拖曳到“快捷菜单”工具栏的按钮上，它表示想要在其上添加该项的上下文菜单。然后将该项拖曳到单独的上下文菜单以显示此菜单。将水平条定位于你希望该项出现的地方，然后将其放下。

8. 右击工具栏按钮、菜单项或创建的显示上下文菜单的上下文菜单项。在“名称”框内拖曳以选择其中的内容。为新按钮或项目输入一个适当的名称。对于菜单项或上下文菜单项，在希望用做访问键的字母之前加一个 &。确认这一访问键字母并未被指定给别的菜单项或上下文菜单项——否则将不得不按两次访问键才能访问第二项，而且还不得不按 Enter 键才能执行命令。

9. 选择希望该按钮或项如何出现——以文本样式出现，以图形样式出现，还是以带图形的文本样式出现。为此必须右击按钮或项，并从上下文菜单中选择“默认样式”、“纯文本样式”、“纯图形样式”或者“图形和文本样式”。工具栏按钮对应的默认样式是图形样式，菜单项或上下文菜单项的默认样式是文本样式。

10. 如果选择使用图形样式，可按如下方式操作：

- ◆使用上下文菜单上的“复制按钮图形”项，以便从别的按钮复制图形，然后使用“粘贴按钮图形”项，将图形粘贴到新按钮上。
- ◆使用“编辑按钮图形”项，以显示“按钮编辑器”对话框（如图 1.17 所示），在框内可以以像素为单位绘出自定义按钮图形，然后用到新按钮上去。
- ◆选择“更改按钮图形”项，以显示出内置按钮图形的弹出式面板，供立即选用。

11. 完成了添加工具栏按钮和菜单项之后，单击“关闭”按钮，以关闭“自定义”对话框。

将宏指定到组合键

在本节中，将学习如何将宏指定到组合键。

本章前面的“指定在Word中运行宏的方式”一节，解释了如何在Word中做这件事。PowerPoint中不能将宏直接指定到组合键。Excel所用方法则与Word也有所不同。

在Word中将宏指定到组合键

在Word中，为了将宏指定到组合键，选择“工具”

- “自定义”，以显示“自定义”对话框，然后单击“键盘”按钮，以显示“自定义键盘”对话框。然后，就按本章前面“指定在Word中运行宏的方式”一节中讨论过的方法去做。

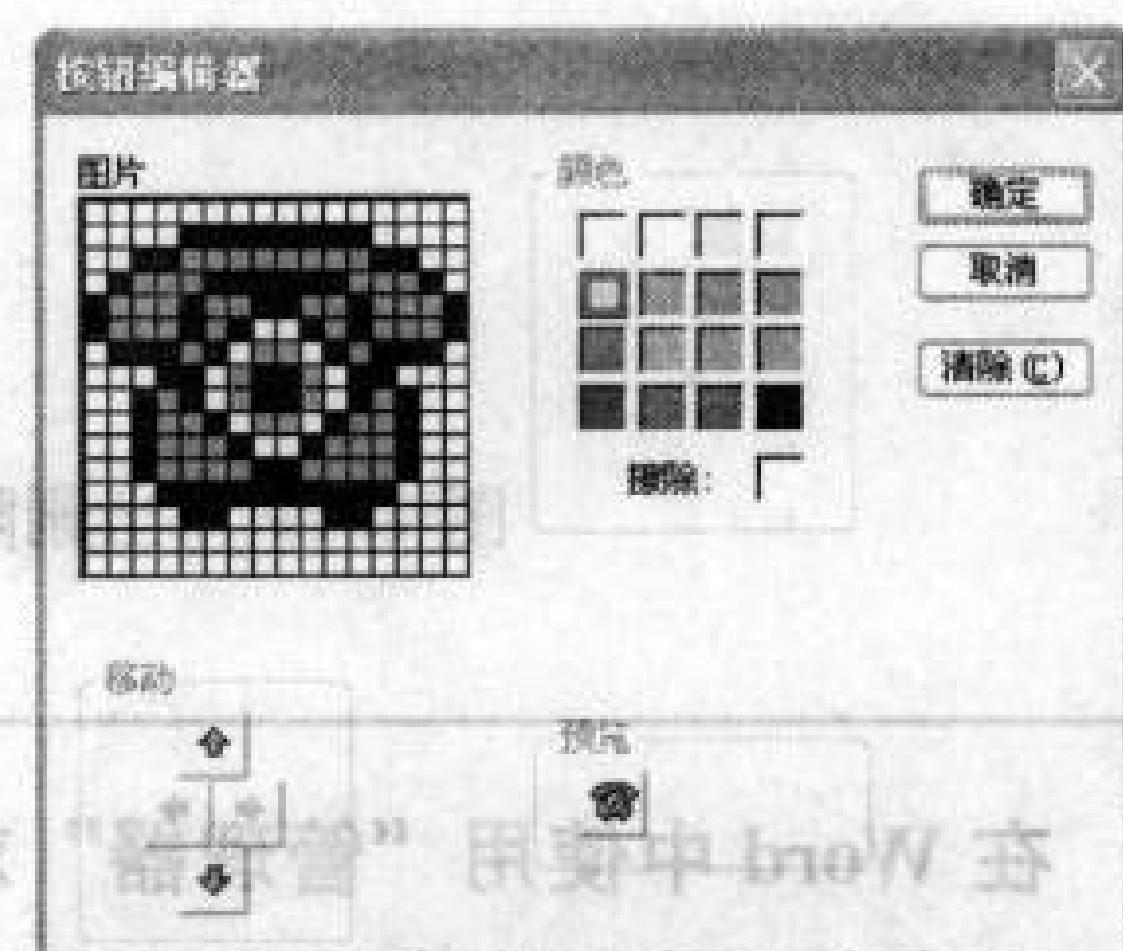


图1.17 如果认为已定义按钮均不合适的话，可以使用“按钮编辑器”对话框创建自定义按钮

在Excel中将宏指定到组合键

在Excel中，为了将宏指定到组合键，遵循如下步骤：

1. 选择“工具”>“宏”>“宏”以显示“宏”对话框。

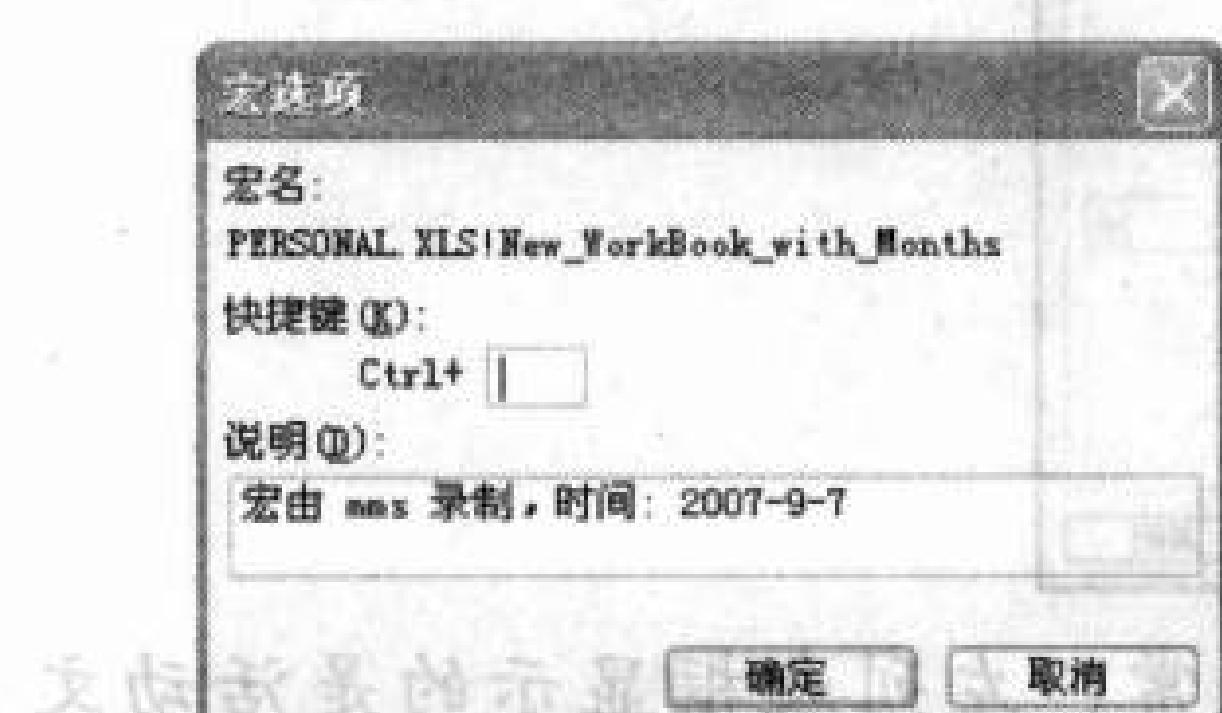


图1.18 在Excel中，使用“宏选项”对话框，以便将宏指定到组合键

2. 选择要指定到组合键的那个宏。

3. 单击“选项”按钮以显示“宏选项”对话框，如图1.18所示。

4. 在“快捷键”文本框内输入组合键。可以只用Ctrl键组合（反之，也可用Ctrl+Alt键组合），也可以把Shift键添加到组合键中，在按字母键的同时按Shift键即可。

提示：如果需要的话，可以在“宏选项”对话框内更改对宏的说明。

5. 单击“确定”按钮，以关闭“宏选项”对话框。

6. 单击“关闭”按钮或“取消”按钮，以关闭“宏”对话框。

删除宏

删除不再需要的宏，遵循如下步骤：

1. 选择“工具”>“宏”>“宏”以显示“宏”对话框。
2. 在“宏名”列表框内选择宏。
3. 单击“删除”按钮。
4. 在出现的警告消息框内，单击“是”按钮。如图1.19所示为Excel中的警告消息框。
5. 单击“关闭”按钮或“取消”按钮，以关闭“宏”对话框。

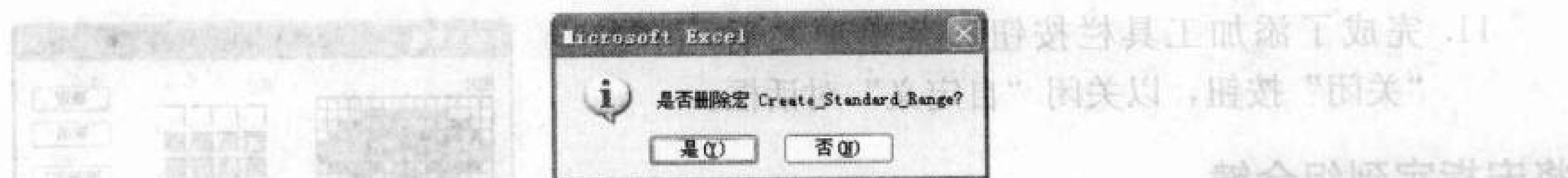


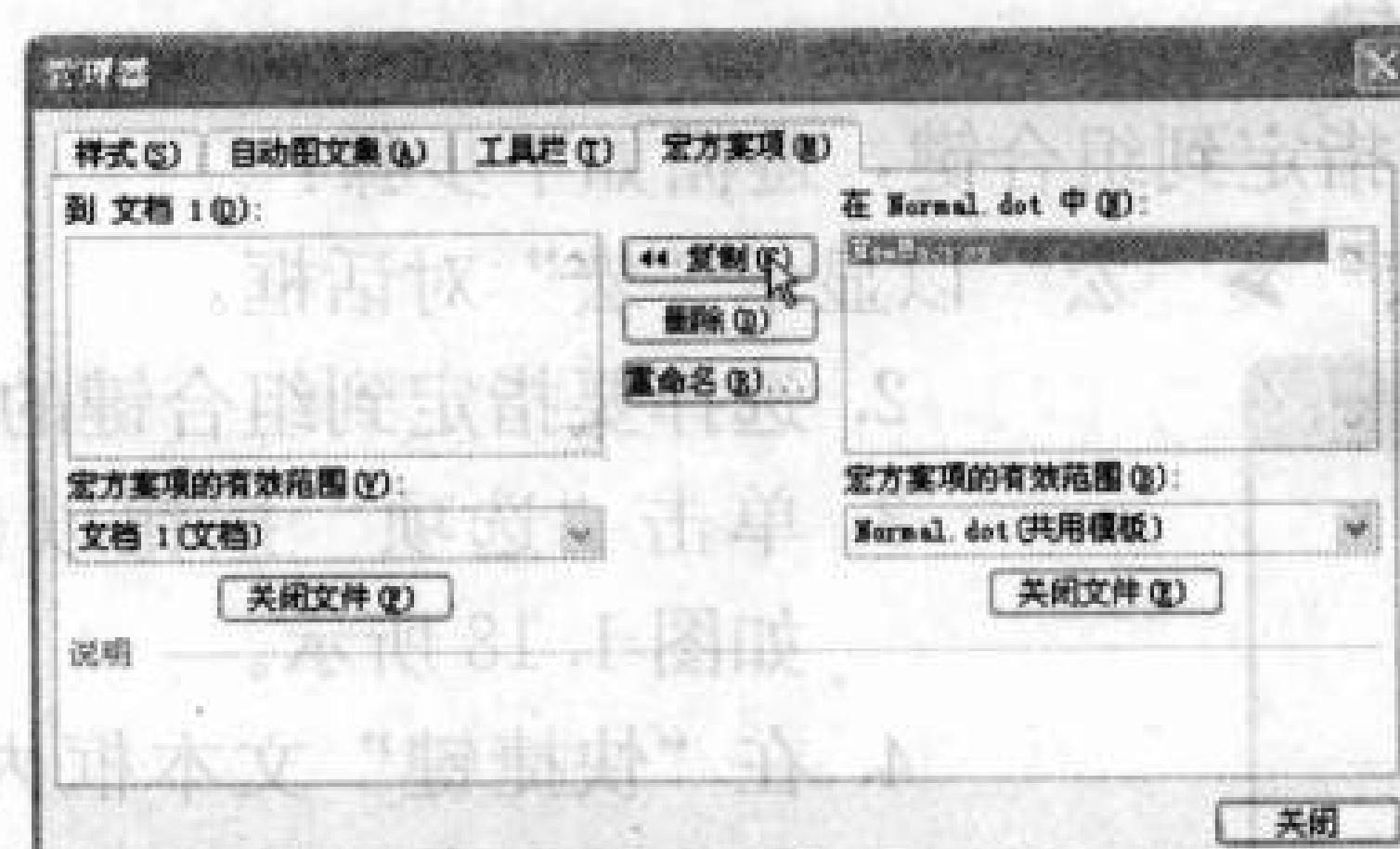
图 1.19 当删除宏时，应用程序进行检查，以确认是想这样做

在 Word 中使用“管理器”对话框来管理宏

大多数 VBA 功能的应用程序要求使用 Visual Basic 编辑器（下章讨论），以便将代码模块、用户窗体和其他代码项从一个文件移至另一个文件。（代码模块是用来保存宏的容器。用户窗体是自定义对话框。）但是，Word 提供了一个称为“管理器”对话框的有用工具，可以使用它直接从 Word 界面来复制、移动、重命名和删除代码模块、用户窗体及其他代码项。

使用“管理器”对话框，遵循如下步骤：

1. 在 Word 中，按下 Alt+F8 键或选择“工具”>“宏”>“宏”，以显示“宏”对话框。
2. 单击“管理器”按钮以显示“管理器”对话框，并单击“宏方案项”，如果“宏方案项”页面（见下图）没有自动显示的话。



3. 观看两个列表框上方显示出来的两个文档或模板。通常，左列表框显示的是活动文档，而右框显示 Normal.dot。更改这些框，使一个列表框显示文档或模板，它们包含有想要复制或移动的代码；而另一个框显示目的文档或模板。（如果只想删除或重命名代码项的话，需要做的只是让“管理器”对话框仅列出包含这些代码项的文档或模板。）为了更改所列文档或模板，单击那一边列表框下方的“关闭文件”按钮。“关闭文件”按钮将变成“打开文件”按钮。单击该按钮以显示“打开”对话框，选择出想要的文档或模板，然后单击“打开”按钮。
4. 然后可以删除、重命名、复制及移动宏方案项：
 - ◆ 要从模板上删除一个或多个宏方案项，需从“管理器”对话框上选出这个项或这些项，并单击“删除”按钮。在确认消息框内单击“是”按钮。其他模板中这些项的复印件不受影响。
 - ◆ 要重命名某个宏方案项，需从图板上选中它，单击“重命名”按钮以打开“重命名”对话框。输入新名并单击确定按钮，其他模板中该项的复印件不受影响。
 - ◆ 要将一个或多个宏方案项从一个模板复制到另一模板，需在“管理器”对话框内打开这些模板。在对话框图板上选中要复制的这个项或这些项，然后单击“复制”按钮，如果接收模板中已有与欲复制方案项同名的项，Word 会显示警告消息框，提示不能

复制该项。如果仍想复制该项，可以将欲复制项重命名，或将目的模板中的同名项重命名，然后再做复制操作。

- ◆要将宏方案项从一个模板移至另一模板，可按上段所述的方法先复制这个项，再将此宏方案项从源模板中删除。

5. 删除、重命名、复制或移动宏方案项完成之后，单击“关闭”按钮，以关闭“管理器”对话框。如果Word提示是否要将受影响的文档或模板的变更保存起来，可单击“是”按钮。

重取各同的中达处而目补类，这命重取佛莫必补人下，取斯待莫殊对果味。取忘佛莫
。补斯待莫始再试然，名命

第 2 章 从 Visual Basic 编辑器入手

- ◆ 打开 Visual Basic 编辑器
- ◆ 打开 Visual Basic 编辑器中的宏
- ◆ 使用 Visual Basic 编辑器的主窗口
- ◆ 为工程设置属性
- ◆ 自定义 Visual Basic 编辑器
- ◆ 关闭 Visual Basic 编辑器并返回到主应用程序

在本章中，将开始学习如何使用 Visual Basic 编辑器，它是微软提供的一个工具，用以配合 VBA 代码和用户窗体一起工作。充当 VBA 应用宿主的所有应用程序，均可使用 Visual Basic 编辑器，所以，当使用 VBA 工作时，不管正在使用哪种应用程序，开发环境几乎是一样的。

本章介绍 Visual Basic 编辑器的基础知识、它的各个组成部分、这些组成部分都能干什么，以及如何去使用它们。以后在本书中将可以学到使用 VBA 的更先进的技巧。

本章也介绍如何自定义 Visual Basic 编辑器，使工作起来更加方便。这种自定义费时不多，但它带来的使用上的便利，与投入的时间相比，要划算得多。

打开 Visual Basic 编辑器

可以从正在使用的主应用程序中打开 Visual Basic 编辑器。例如，如果正在使用 Word，就从 Word 上打开 Visual Basic 编辑器。这样，所打开的 Visual Basic 编辑器实例就和 Word 相关联。可以打开两个或更多个 Visual Basic 编辑器实例。例如，如果已经从 Word 打开了一个 Visual Basic 编辑器实例，还可以从 Excel 再打开 Visual Basic 编辑器的另一个实例，然后从 PowerPoint 打开又一个实例。

可以以两种方法打开 Visual Basic 编辑器：

- ◆ 确定想要编辑的宏。然后，从主应用程序打开 Visual Basic 编辑器并显示这个宏，这样，就可以和它一起工作了。
- ◆ 直接打开它，然后导引至要编辑的代码上。

下两节说明打开 Visual Basic 编辑器的两种方法，第三节说明如何导引到某个宏。当希望针对一个特定的宏进行工作时，间接法比较省时；而当需要立即打开 Visual Basic 编辑器并投入工作时，直接法比较方便。

和所选择的宏一道打开 Visual Basic 编辑器

如果知道要针对哪个宏进行工作，那就使用这种方法来打开 Visual Basic 编辑器，并同时打开宏。本例使用 Word 来打开在第 1 章里录制的宏 Transpose_Word_Right。

1. 如果没有运行 Word，则打开 Word；如果正在运行，则激活 Word。
2. 按下 Alt+F8 或选择“工具”>“宏”>“宏”，以显示“宏”对话框。
3. 选择 Transpose_Word_Right 宏并单击“编辑”按钮。Word 将打开 Visual Basic 编辑器及所显示的宏，并准备进行编辑，如图 2.1 所示。
4. 选择“文件”>“关闭并返回到 Microsoft Word”，以暂时关闭 Visual Basic 编辑器，以后，可以使用下节所述的方法再打开它。

打开 Visual Basic 编辑器

遵循以下步骤，可直接打开 Visual Basic 编辑器。

1. 打开或激活主应用程序。在本例中是打开 Word。
2. 按下 Alt+F11，或选择“工具”>“宏”>“Visual Basic 编辑器”，打开 Visual Basic 编辑器。

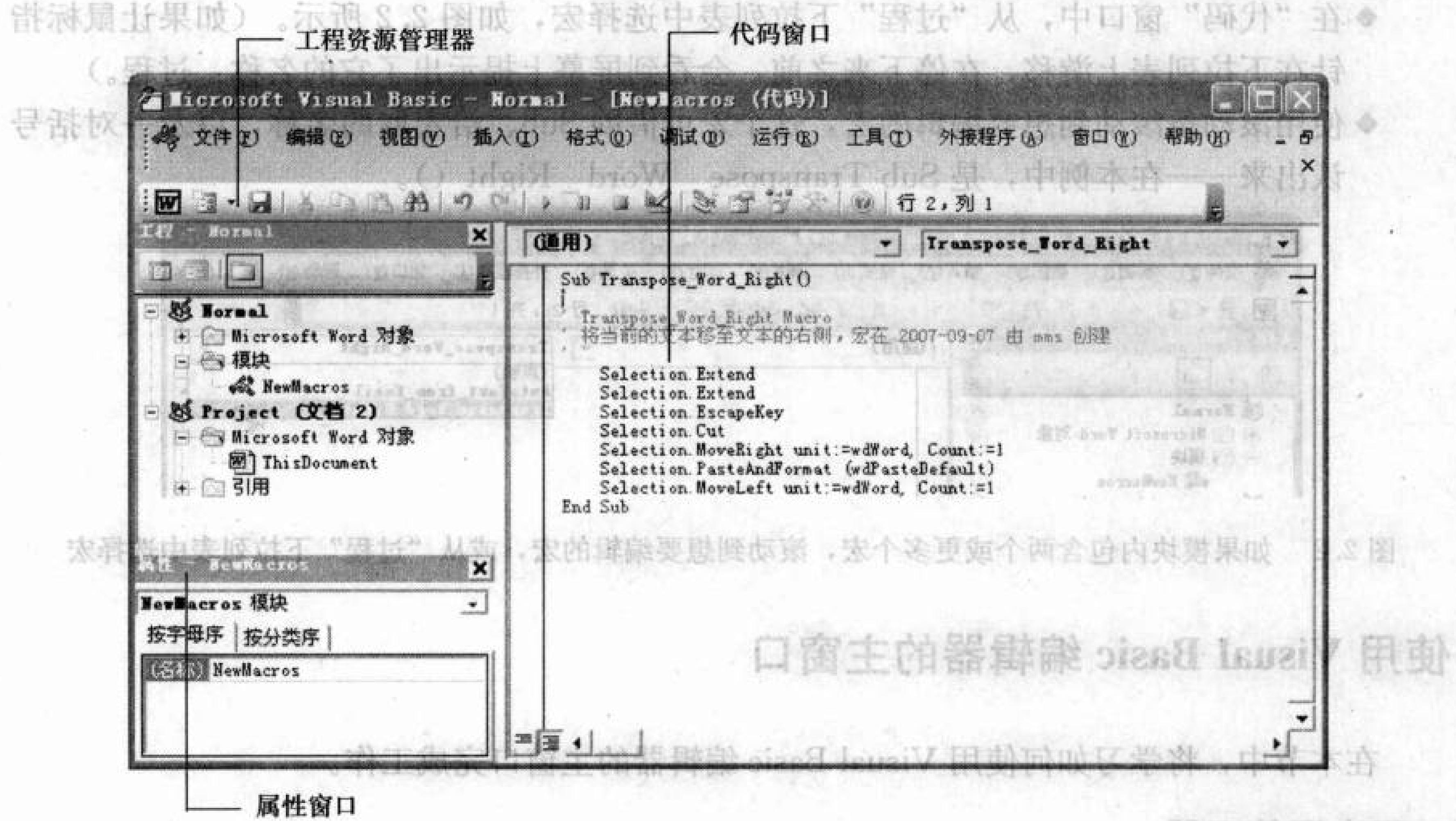


图 2.1 Visual Basic 编辑器，带有在“代码”窗口中打开的 Transpose_Word_Right 宏

注意：可能会看到一个或多个打开的“代码”窗口，这与最近一次关闭 Visual Basic 编辑器时它的状态有关。例如，如果前次是让对应于 NewMacros 模板的“代码”窗口保持打开的话，Visual Basic 编辑器将可能再次显示这一“代码”窗口。

导引到一个宏

直接打开 Visual Basic 编辑器之后，使用工程资源管理器窗口可导引到宏。当正在 Visual Basic 编辑器中工作时，也可以使用工程资源管理器，在几个打开的工程和模块之间进行切换。

注意：工程资源管理器窗口以类似于标准的 Windows Explorer 树的方式工作。在

树中将会看到不同的工程，这取决于正在使用的应用程序（本章后面将有进一步介绍）。

要导引到 Transpose_Word_Right，遵循以下步骤：

1. 在 Visual Basic 编辑器左上角的工程资源管理器窗口内，找到对应于 Normal（它表示 Normal.dot, Normal 模板）的条目，单击它名称左边的 + 号，使其展开（如已扩展，可跳过这一步）。
2. 双击“模块”对象以展开它。
3. 双击 NewMacros 模块。（在此模块中，Word 已自动创建了录制的宏。）Visual Basic 编辑器在右边的“代码”窗口中显示出该模块的内容。

如果模块内包含有一个以上的宏，必须选择打算与之工作的那个宏——在本例中是 Transpose_Word_Right。（如果只录制了 Transpose_Word_Right 这一个宏，“代码”窗口里就只有它出现。）要做到这一点，可以使用如下任一种方法：

- ◆ 在“代码”窗口中，从“过程”下拉列表中选择宏，如图 2.2 所示。（如果让鼠标指针在下拉列表上游移，在停下来之前，会看到屏幕上提示出了它的名称：过程。）
- ◆ 使用滚动条滚动到想要编辑的宏，这个宏可借助 Sub、给宏取的名称，以及一对括号认出来——在本例中，是 Sub Transpose_Word_Right ()。

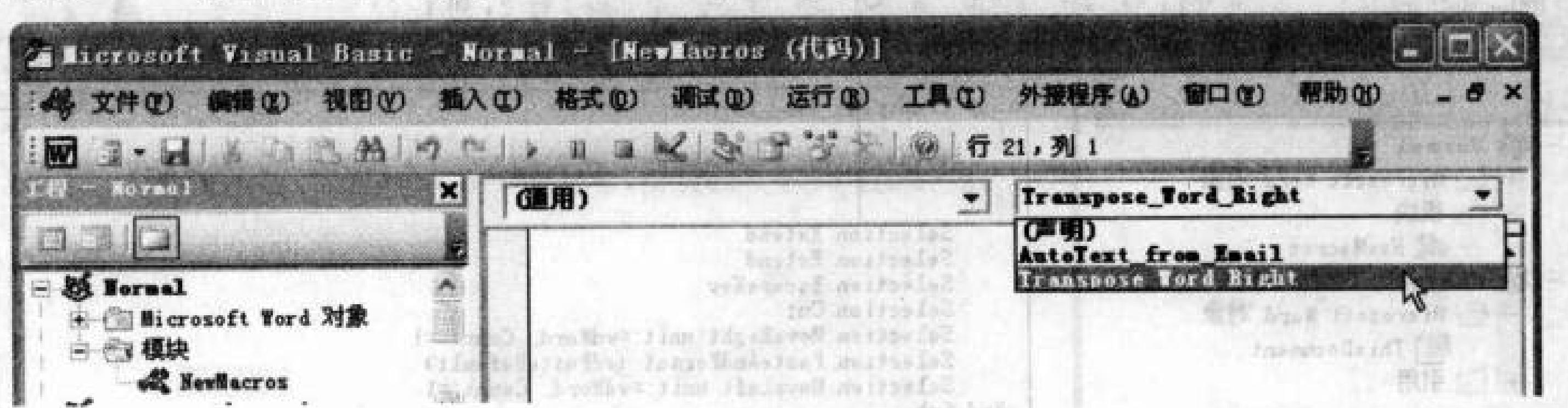


图 2.2 如果模块内包含两个或更多个宏，滚动到想要编辑的宏，或从“过程”下拉列表中选择宏

使用 Visual Basic 编辑器的主窗口

在本节中，将学习如何使用 Visual Basic 编辑器的主窗口完成工作。

工程资源管理器

工程资源管理器是在 Visual Basic 编辑器内各组成部分之间进行导引的工具。图 2.3 所示即为以 Word 为主应用程序的 Visual Basic 编辑器的工程资源管理器。

根据主应用程序及其功能的不同，每个工程包含如下成分的一部分或全部：

- ◆ 用户窗体（窗体用以组成应用程序的用户界面，例如自定义对话框）。
- ◆ 包含宏、过程以及函数的模块。
- ◆ 类模块（类模块用来定义对象、对象的属性和值）。
- ◆ 引用其他工程或库文件（如 DLLs——动态链接库）。
- ◆ 与应用有关的对象。例如，每个 Word 文档和模板都包含有 Microsoft Word Objects 文件夹，文件夹内放有名为 This Document 的类对象。This Document 让人访问对应此文档或模板的属性和事件。每个 Excel 工作簿包含有名为 This Workbook 的类对象，让人访问对应此工作簿的属性和事件，而且，对于每个工作表有一个表对象

(名为 Sheet1, Sheet2, 等等)。

对于大多数主应用程序，每个打开的文档和模板都可以视为一个工程，并显示为工程树的一个根。工程树还包含有全局宏存储——例如 Word 中的 Normal.dot 模板或 Excel 中的个人宏工作簿——以及被加载的外接程序。

作为例子，在图 2.3 中，Normal.dot 被标识为 Normal，活动文档被标识为 Project (Document2)：一个名为 Document2 的文档（它还未被保存）。附属于活动文档的模板被标识为 TemplateProject (Sales Memo) ——换而言之，模板名为 Sales Memo.dot。还加载了一个名为 TemplateProject (Sales Global Template) 的全局模板，也可以说该模板名为 Sales Global Template.dot。从列表中看，不能说这是一个全局模板（与它的说明名称不同），但单击它的 + 号来扩展它时，消息框告知该模板已被锁住。

提示：可以使用两种方法来更改工程名：使

用“工程资源管理器”对话框（本章后面讨论），或在“属性”窗口内选中该工程并输入新名。一旦更改了名称，该工程就以工程资源管理器中的这个名称，后随文档名或模板名来识别。例如，如果把文档 Document2 的工程名改为 Testing，在工程资源管理器中，该文档工程就标识为 Testing (Document2)，不再是 Project (Document2)。

可以用导引 Windows Explorer 树的同样方式来导引工程资源管理器。单击某一工程项左侧的 + 号，可扩展视图并显示出工程所含的各项目；单击 - 号，则回缩视图并重新隐藏各项目。双击某一模块，可在“代码”窗口中显示其代码。双击某一用户窗体，则在“代码”窗口中显示该窗体（用户窗体可视为代码，因为它是使用代码创建的对象）。

Visual Basic 编辑器以默认方式显示工程资源管理器，而且，因为工程资源管理器能在 VBA 工程的各成分之间进行快速而有效的导引，所以，通常应该保持显示它，除非屏幕空间不足，或者要在“代码”窗口工作很长时间，不需要切换到其他成分。要关闭工程资源管理器，需单击 Ctrl+R 键，或选择“视图”>“工程资源管理器”。在本章后面会看到，也可以不连接工程资源管理器，即在需要更多空间时把它移开。

图 2.3 中，在工程资源管理器顶部的工具栏上有三个按钮：

查看代码 为选中的对象显示“代码”窗口。例如，如果选择了工程资源管理器中的某一用户窗体，单击“查看代码”按钮后，Visual Basic 编辑器就显示出包含附属于此用户窗体的代码的“代码”窗口。如果选择了工程资源管理器中的某一模块或类模块，单击“查看代码”按钮后，Visual Basic 编辑器就显示出包含此模块的代码的“代码”窗口。也可以右击对象和从快捷菜单中选择“查看代码”。

提示：对于模块或类模块，也可以双击对象以查看其代码。这往往比先选中模块再单击“查看代码”按钮来得快。但是，对于用户窗体或文件，双击则是发出“查看对象”命令（随后要讨论），而不是“查看代码”命令。

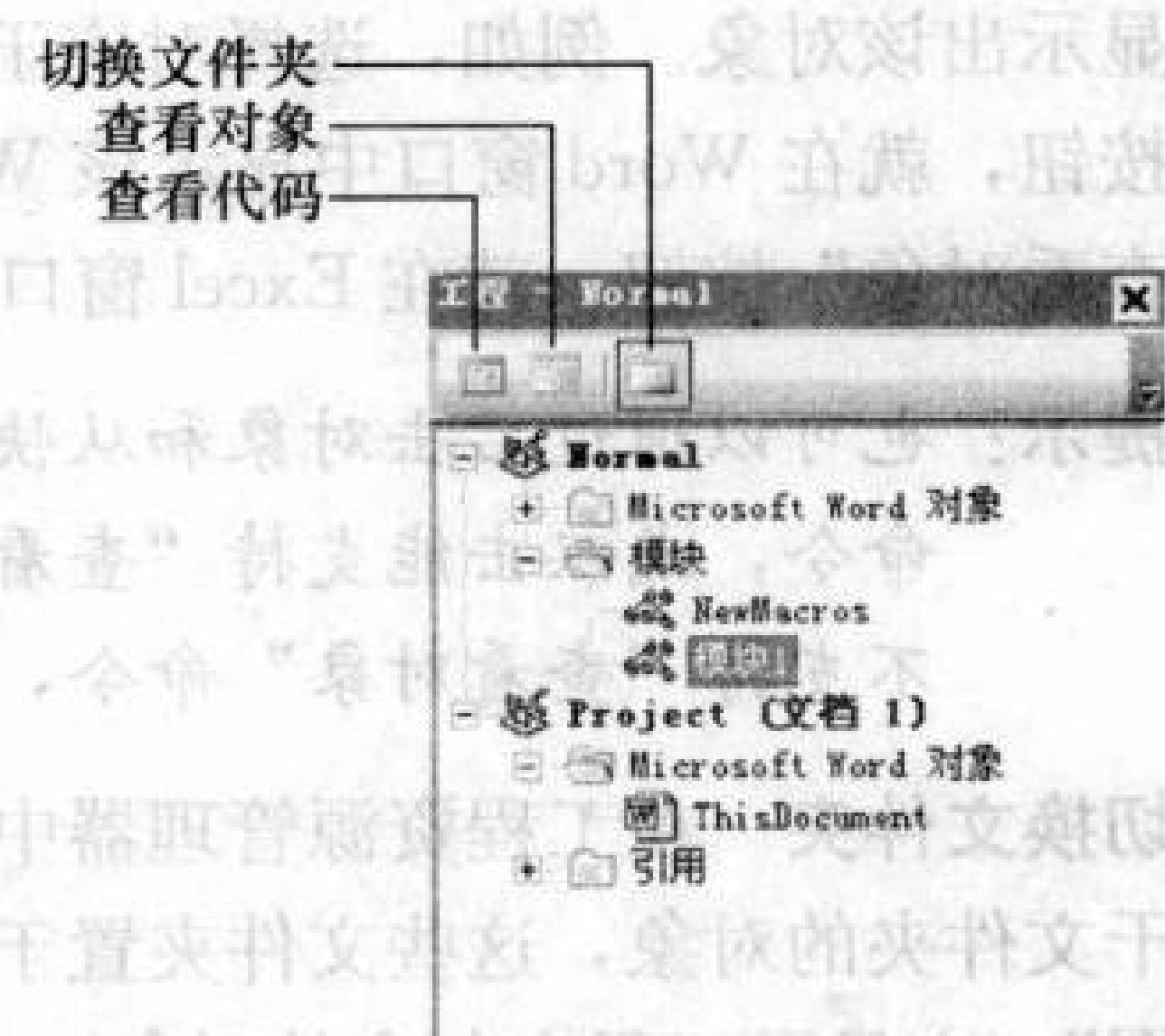


图 2.3 使用工程资源管理器，导引到想与之一道工作的模块

查看对象 显示出包含被选对象的窗口。在选择到一个可显示的对象（如用户窗体或文件，或文件内的对象）之前，“查看对象”按钮是颜色暗淡和不可使用的。如果选中的对象是用户窗体，单击“查看对象”按钮之后，在主应用程序的窗口内就显示出该用户窗体；如果选中的对象是文件或文件内的一个对象，单击“查看对象”按钮后，就在主应用程序的窗口中显示出该对象。例如，选择对应于 Word 文档的 This Document 对象，并单击“查看对象”按钮，就在 Word 窗口中显示该 Word 文档。选择 Excel 工作簿中的 Sheet1 对象，并单击“查看对象”按钮，就在 Excel 窗口中显示 Excel 工作簿内的这张工作表。

提示：也可以通过右击对象和从快捷菜单中选择“查看对象”来发出“查看对象”命令，或双击能支持“查看对象”命令的对象来发出这一命令。（如果对象不支持“查看对象”命令，则双击它时发出的是“查看代码”命令。）

切换文件夹 在工程资源管理器中切换对象的视图，在“文件夹视图”（视图显示分离为若干文件夹的对象，这些文件夹置于包含它们的文档工程或模板工程之下）和“文件夹内容视图”（它显示工程内包含的对象）之间进行切换。图 2.4 左部显示对应于某一应用程序的工程资源管理器，归类为文件夹视图；右部显示同一情况下的工程资源管理器，则以文件夹内容视图形式显示。是把更多时间花在文件夹视图上，还是花在文件夹内容视图上，取决于屏幕大小，用户置入给定工程的对象的数目，以及思考的方式，但不是一定要按这个次序。出于很多目的，用户可能愿意在文件夹视图和文件夹内容视图之间切换，以便更容易地找出对象。

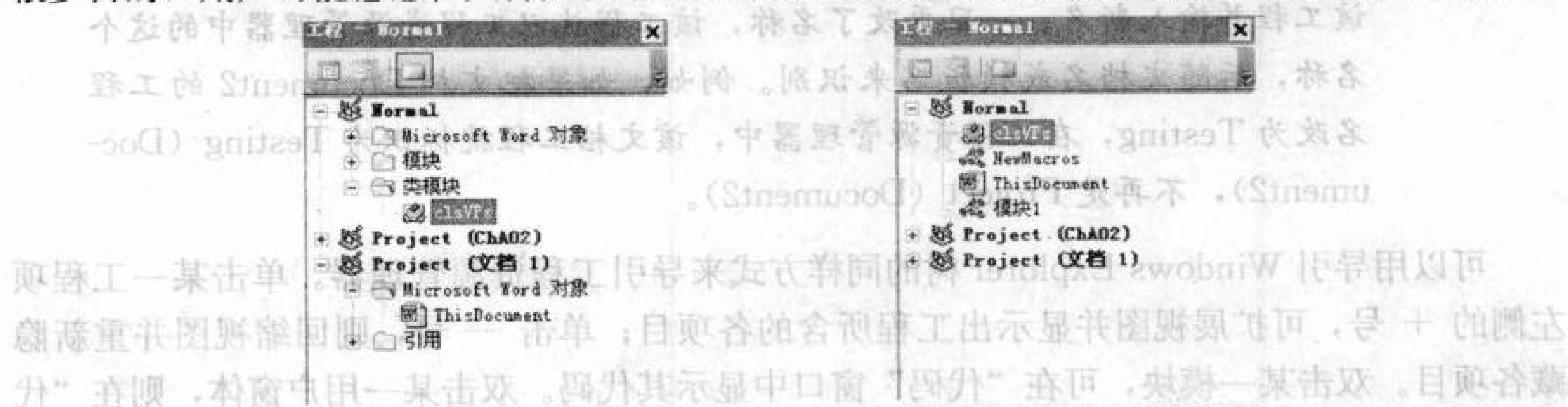


图 2.4 文件夹视图（左）显示分离为若干文件夹的对象。文件夹内容视图（右）显示工程内的对象

除了导引到必须与之一道工作的项目之外，还可以使用工程资源管理器完成如下任务：

- ◆ 向工程添加某些组成部分，或从工程中移出某些组成部分。例如，可以使用工程资源管理器向工程添加模块或添加用户窗体。
- ◆ 把一个工程的组成部分与另一工程的组成部分做比较。当需要迅速确定两个或多个工程之间的差别时，这种比较是很有用的。
- ◆ 从一个工程向另一工程移动项目或复制项目。可以在工程资源管理器中，把某一代码模块、类模块或用户窗体从一个工程拖曳至另一工程，以进行复制；也可以把它们从 Visual Basic 编辑器的某一实例中的工程资源管理器，拖曳至另一实例中的工程资源管理器，以进行复制。例如，可以把某个用户窗体从以 Excel 为宿主的 Visual Basic 编辑器实例，拖曳至以 PowerPoint 为宿主的 Visual Basic 编辑器实例，以复制该用户窗体。
- ◆ 在一个工程与另一工程之间导入或导出代码模块或用户窗体。

注意：通过工程资源管理器来实现的很多操作，通过 Visual Basic 编辑器的菜单项也可以实现。当工程资源管理器未显示时，这一点很有用。一般来说，工程

资源管理器提供了最容易的方式，在Visual Basic编辑器内进行模块间导引，特别是在有几个复杂工程同时打开的时候。可以通过右击工程资源管理器中的对象以显示快捷菜单的方法，访问针对该对象的最常用命令。

对象浏览器

Visual Basic编辑器提供了一个全对象浏览器，以帮助浏览VBA中所有可用对象。在第8章中要详细学习对象浏览器，在本书的最后部分，还要考查对应于各种应用程序的对象模型。但在这里先简单看看图2.5，它显示了对应于Word VBA实例的对象浏览器。在左侧图板中选中Document对象，其属性列表便会在右侧图板中。

可以看出，这些属性中，有很多与已掌握的Word文档方面的知识是相通的。例如，AttachedTemplate属性说明文档目前附属于哪个模板。与之类似，Bookmarks属性包含了文档中所有书签方面的信息。属性信息显示在对象浏览器底部。

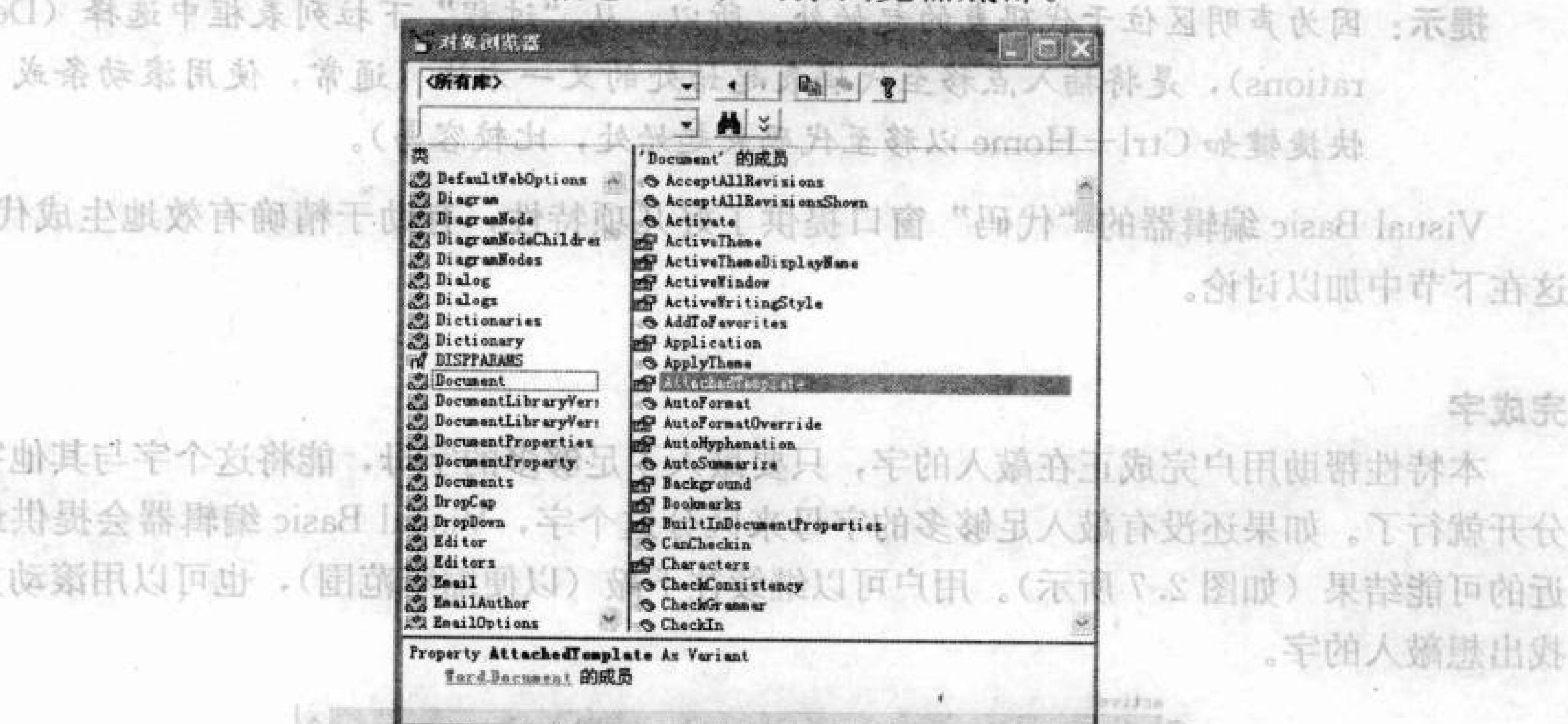


图2.5 对象浏览器提供了迅速查看对象及其属性的方法。这里可看到Document对象包含的属性

代码窗口

创建和编辑宏的大部分实际工作，是在Visual Basic编辑器的“代码”窗口中进行的。Visual Basic编辑器中每个打开的工程提供了“代码”窗口，工程内可以包含代码的每个文档片段，以及工程内的每个代码模块和用户窗体。每个“代码”窗口以工程名、工程内的模块名以及括号中的字Code来标识。图2.6显示了Visual Basic编辑器“代码”窗口，内有打开的宏Transpose_Word_Right。

从图中可见，在“代码”窗口的标题栏下方有两个下拉列表框：

- ◆“代码”窗口左上角的“对象”下拉列表框，提供了在不同对象之间导引的快速方式。
- ◆“代码”窗口右上角的“过程”下拉列表框，使用户能够在当前模块内各过程之间迅速切换。单击↓键以显示“过程”下拉列表，可以看到第一个过程是(Declarations)，这就把用户带到当前代码表顶端Declarations(声明)区，这是声明公共变量和其他VBA信息的地方，各过程都需要知道上述信息。

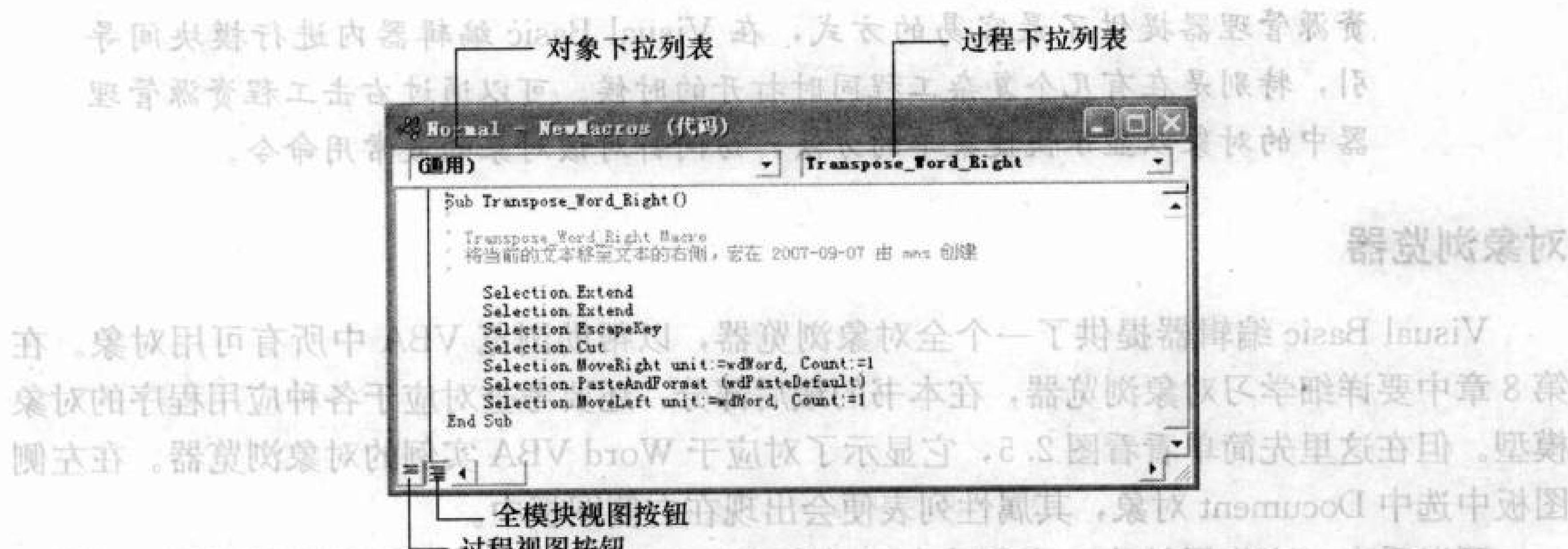


图 2.6 在“代码”窗口内创建和编辑宏

提示：因为声明区位于代码表的起始处，所以，从“过程”下拉列表框中选择（Declarations），是将插入点移至代码表起始处的又一方法（通常，使用滚动条或快捷键如 Ctrl+Home 以移至代码表起始处，比较容易）。

Visual Basic 编辑器的“代码”窗口提供了好几项特性，有助于精确有效地生成代码，这在下节中加以讨论。

完成字

本特性帮助用户完成正在敲入的字，只要敲入了足够多的字母，能将这个字与其他字区分开就行了。如果还没有敲入足够多的字母来区分这个字，Visual Basic 编辑器会提供最接近的可能结果（如图 2.7 所示）。用户可以继续往下敲（以便缩窄范围），也可以用滚动方法找出想敲入的字。

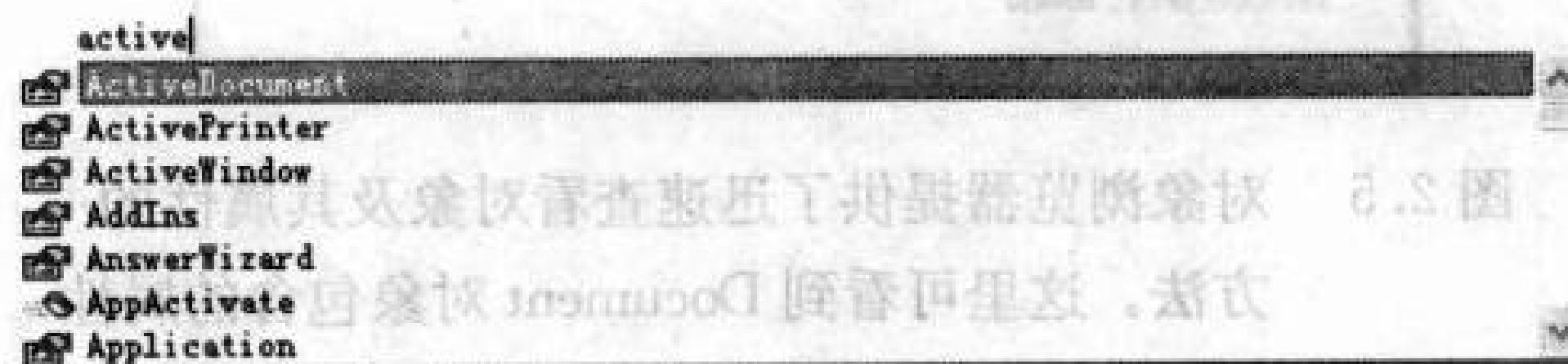


图 2.7 本特性能自动完成一个术语，只要已敲入足够的字母来区分它。如果敲得不够多，可以从一个列表中选择

激活本特性的最简单方法，是在敲代码时按下 Ctrl+空格键。也可以选择“编辑”>“完成字”，或单击“编辑”工具栏上的“完成字”按钮（如图 2.8 所示）。

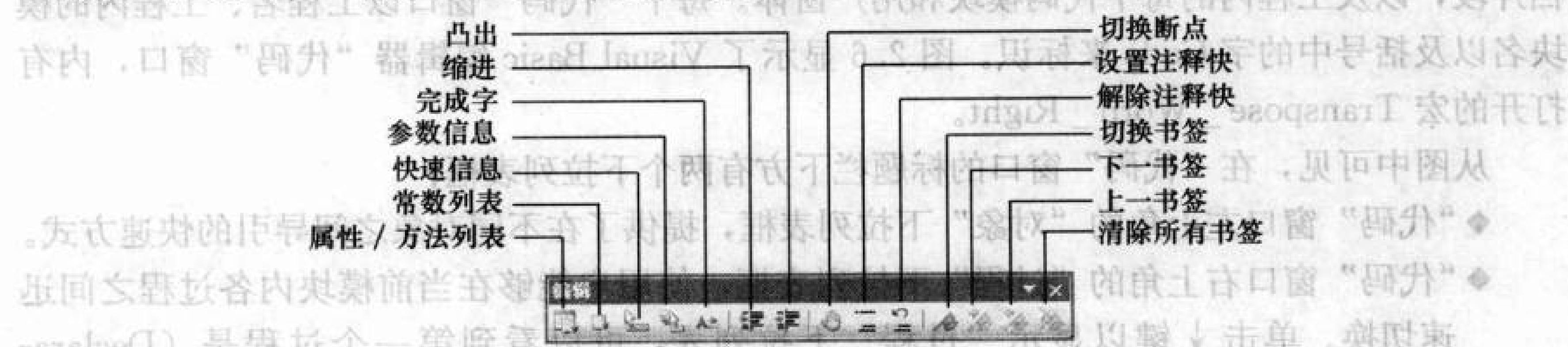


图 2.8 (“编辑”工具栏含有在“代码”窗口上工作时用到的命令

快速信息

“快速信息”命令显示出屏幕提示，内有关于当前变量、函数、方法、语句或过程的语法信息。图 2.9 给出“快速信息”屏幕提示的例子。

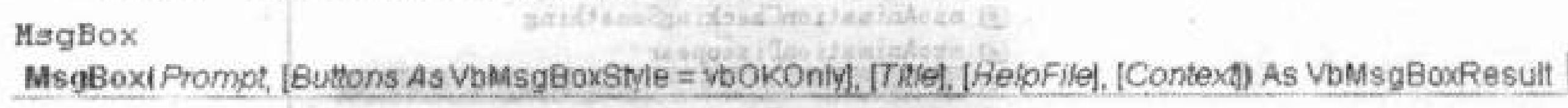


图 2.9 使用“快速信息”命令，可获得对语法的快速提示，或状态的快速读出

为了发出“快速信息”命令，可使用如下任一种方法：

- ◆ 右击此术语和从快捷菜单中选择“快速信息”。

- ◆ 将插入点定位于此术语并按下 **Ctrl+I**。

- ◆ 将插入点定位于此术语并选择“编辑”>“快速信息”。

属性/方法列表

“属性/方法列表”命令显示出一个弹出式列表框，它包含有用户刚键入的对象的属性和方法，以使用户迅速完成表达式。“属性/方法列表”是默认接通的，当在一个表达式内键入一个句号时，会自动弹出该列表框。也可以单击“编辑”工具栏上的“属性/方法列表”按钮，以显示该列表框。

为了使用“属性/方法列表”，遵循如下步骤：

1. 按下 **↓** 以向下滚动至属性或方法，或使用鼠标向下滚动（如图 2.10 所示）。也可以敲进属性/方法名的前几个字母，以跳到该处。

2. 将属性/方法输入代码项：

- ◆ 如果希望在输入了属性或方法之后，仍在相同行上继续工作的话，按下 **Tab** 键，或双击该属性或方法。

- ◆ 如果希望在输入了属性或方法之后，从新的一行开始工作的话，则按 **Enter** 键。

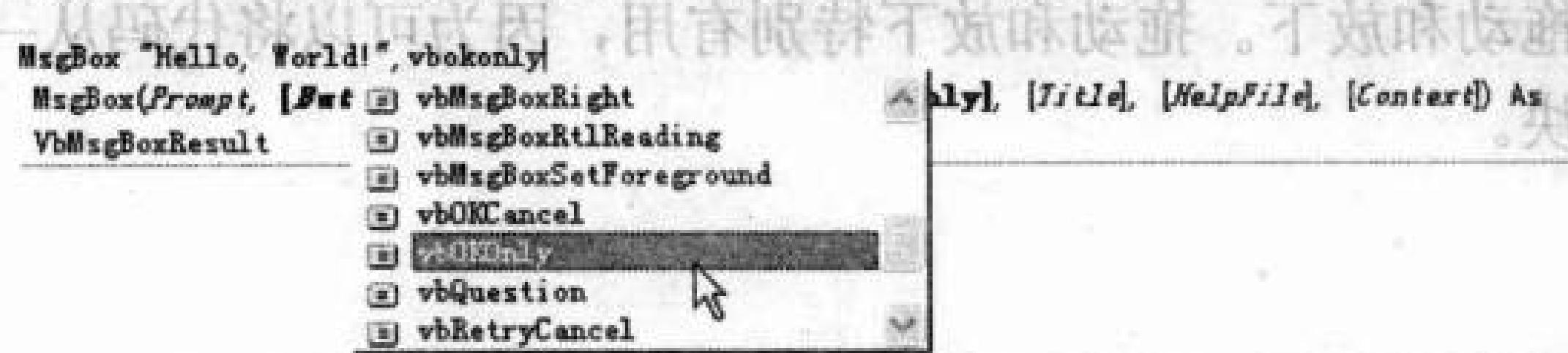


图 2.10 使用“属性/方法列表”命令，以迅速精确地输入代码项

常数列表

“常数列表”命令显示出一个弹出式列表框，它包含有对应于用户刚键入的某一属性的若干常数，以使用户迅速完成表达式。“常数列表”是默认接通的。也可以单击“编辑”工具栏上的“常数列表”按钮，以显示该列表框。

使用“常数列表”（如图 2.11 所示），遵循如下步骤：

1. 按下 **↓** 以向下滚动至常数，敲进它的第一个字母（或前几个字母），或使用鼠标向下滚动。

2. 将常数输入代码：

- ◆ 如果希望在输入了常数之后，仍在相同代码行上继续工作的话，按下 **Tab** 键，或

双击该常数。

◆ 如果希望在输入了常数之后，从新的一行开始工作的话，则按 Enter 键。

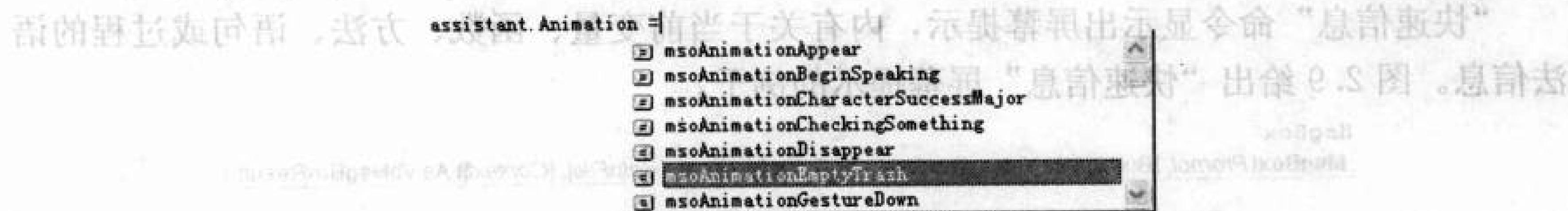


图 2.11 “常数列表”特性为用户省时省力，特别是在敲入复杂的常数名称时

数据提示

当 Visual Basic 编辑器处于中断模式时（该模式在测试和调试宏时使用），鼠标指针在某一变量上方移动，“数据提示”特性会显示出包含该变量的值的屏幕提示。图 2.12 显示出一个例子。“数据提示”特性是默认运行的。

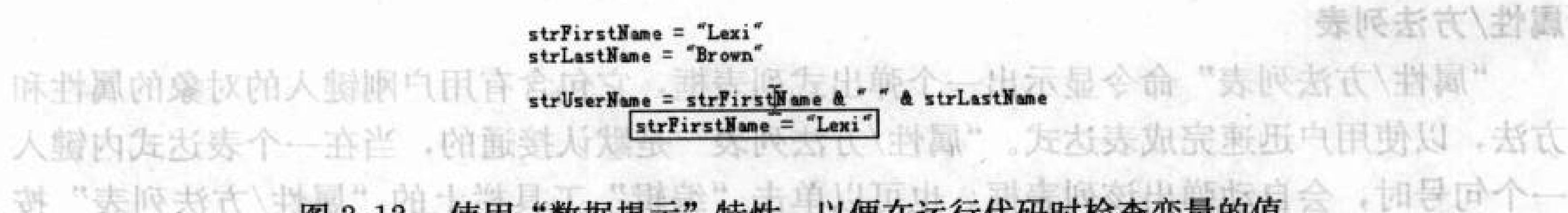


图 2.12 使用“数据提示”特性，以便在运行代码时检查变量的值

边界指示符

“边界指示符”特性，使用户能够快速设置断点、下一语句或书签，只需在“代码”窗口边界内单击即可。在本书后面可看到如何设置断点、设置下一语句，以及设置书签。

其他编辑特性

除了上述特性之外，“代码”窗口还包括有标准的 Office 编辑特性。例如复制、移动、剪切、粘贴，以及拖动和放下。拖动和放下特别有用，因为可以将代码从一个过程或模块拖曳至另一过程或模块。

属性窗口

Visual Basic 编辑器提供有“属性”窗口，可以用它来查看和修改 VBA 对象，如工程、模块或类模块、用户窗体或控件（如对话框中的按钮或复选框）的属性。“属性”窗口顶部的下拉列表用来挑选对象，以便对该对象的属性进行查看或修改。在按字母序页面中，项目中的属性是按字母排序的，在按分类序页面中，属性则是分类列出。

在图 2.13 中，左边显示出对应于某个 Excel 工作簿的属性，是按字母排序的；右边显示出对应于某个用户窗体（自定义对话框）的属性，是按分类列出的（为 Excel 工作簿显示按分类序属性没有多大用处，因为所有属性均属杂项）。许多工作簿属性是容易掌握的。例如，HasRoutingSlip 属性设置为 False，表示该工作簿没有附属于它的电子邮件传送名单；Saved 属性设置为 True，表示该工作簿不含有任何未保存的更改。在第 14 和第 15 章会碰到用户窗体属性，但是现在可以先看看 Caption 属性（它列在 Appearance 类，意为设定用户窗体标题栏文本）和 Height 属性（它列在 Position 类，意为以点数指明用户窗体的高度）。



图 2.13 使用“属性”窗口，以查看某一工程、用户窗体、模块、类模块或控件的属性了解设计模式、运行模式和中断模式

Visual Basic 编辑器使用三种模式

设计模式 也称设计时间。当在 Visual Basic 编辑器中针对代码进行工作时，处在设计模式中。

运行模式 也称运行时间。代码运行时，处在运行模式。

中断模式 当代码在运行但执行暂时中止时，处在中断模式。中断模式可让用户单步运行其代码，每次一条命令或一个过程（而不是一次连续运行所有命令）。使用此模式来调试或鉴定代码。在中断模式上会花很多时间。

Visual Basic 编辑器以默认方式显示“属性”窗口，但是可以单击它的“关闭”按钮（ \times 按钮）来关闭它。为了再次显示“属性”窗口，需按下 F4，或选择“视图” \triangleright “属性窗口”。

为了更改某一属性，需单击含有该属性名称的小格（如果想要为该属性选择一个不同的值），或在取值格内单击（如果想要编辑该值），然后再更改这个值。根据属性类型，可以选择不同的值，对于 True/False 属性，在下拉列表中仅有这两种选择；对于文本属性，例如 Name，可以输入任何有效的 VBA 名称。

按默认方式，“属性”窗口连接在工程资源管理器之下。可以拖曳二者之间的边框，以调整“属性”窗口或工程资源管理器的大小；也可以将边框拖至它们的右边，以立即调整二者的大小。如果恢复到“属性”窗口，可以调整它的大小，即拖曳它的边框或四角以显示更多属性，或缩小该窗口，以便在 Visual Basic 编辑器中少占空间。

立即窗口

除了工程资源管理器，“代码”窗口，以及“属性”窗口之外，Visual Basic 编辑器还有很多默认为不显示的其他窗口。两个重要的这种窗口是对象浏览器（本章前面已讲过）和“立即”窗口，在第 5 章讨论 VBA 语言时会用到它们。

如图 2.14 所示的“立即”窗口，是一个很小的、很简洁的窗口。可以把它作为草稿本来用，写上打算要测试而尚未写入宏的代码行。当把代码行敲进“立即”窗口并按 Enter 键时，Visual Basic 编辑器便执行该代码。

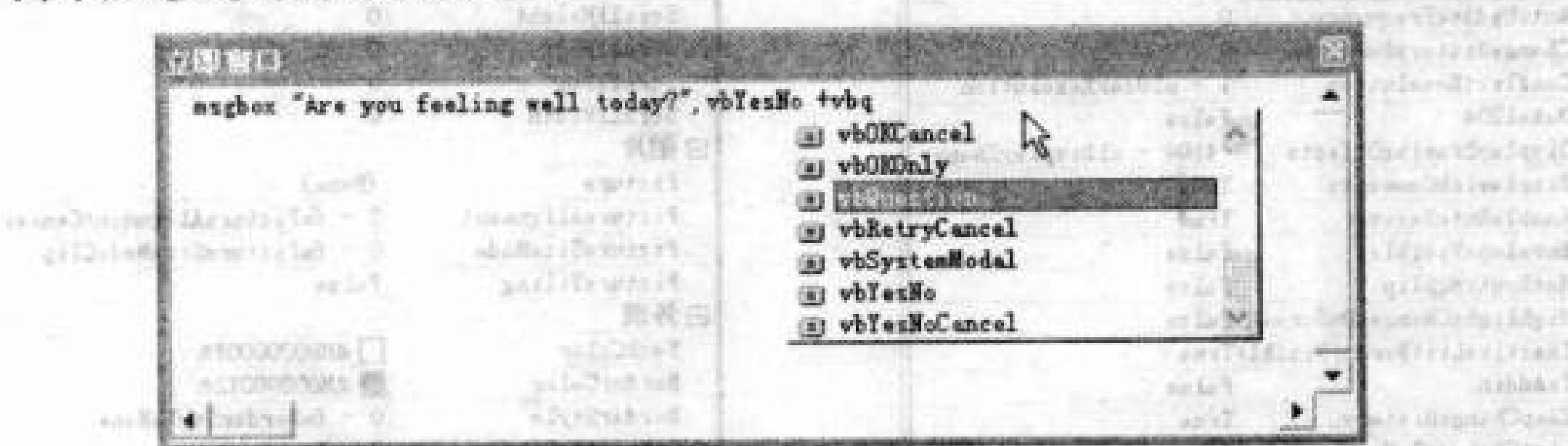


图 2.14 使用“立即”窗口，写入思考中的工作和信息

为了显示“立即”窗口，按下 Ctrl+G，或选择“视图”>“立即窗口”。如果“立即”窗口被连至 Visual Basic 编辑器窗口旁边，双击其标题栏，可使立即窗口不连接。为使“立即”窗口返回其被连接位置，需再次双击其标题栏。

要关闭“立即”窗口，需单击其“关闭”按钮（×按钮）。按下 Ctrl+G 或选择“视图”>“立即窗口”可再次使“立即”窗口不关闭。

注意：也可以使用“立即”窗口，来显示在执行代码时帮助用户检查变量和表达式的值的信息。

为工程设置属性

可以为每个 VBA 工程设置几项属性，包括它的工程名，它的说明和是否锁定以防别人查看。为了考查或设置工程的属性，需在工程资源管理器内右击该工程或其中的一个组成部分，和从上下文菜单中选择属性项目，以显示“工程属性”对话框。菜单项和得到的对话框均可通过工程说明来标识——例如，对应于 Word 中某一模板的属性对话框，可标识为“模板工程—工程属性”，而对应于 Excel 工作簿的属性对话框，可标识为“VBA 工程—工程属性”。图 2.15 显示的是对应于 Excel 工作簿工程的“工程属性”对话框。

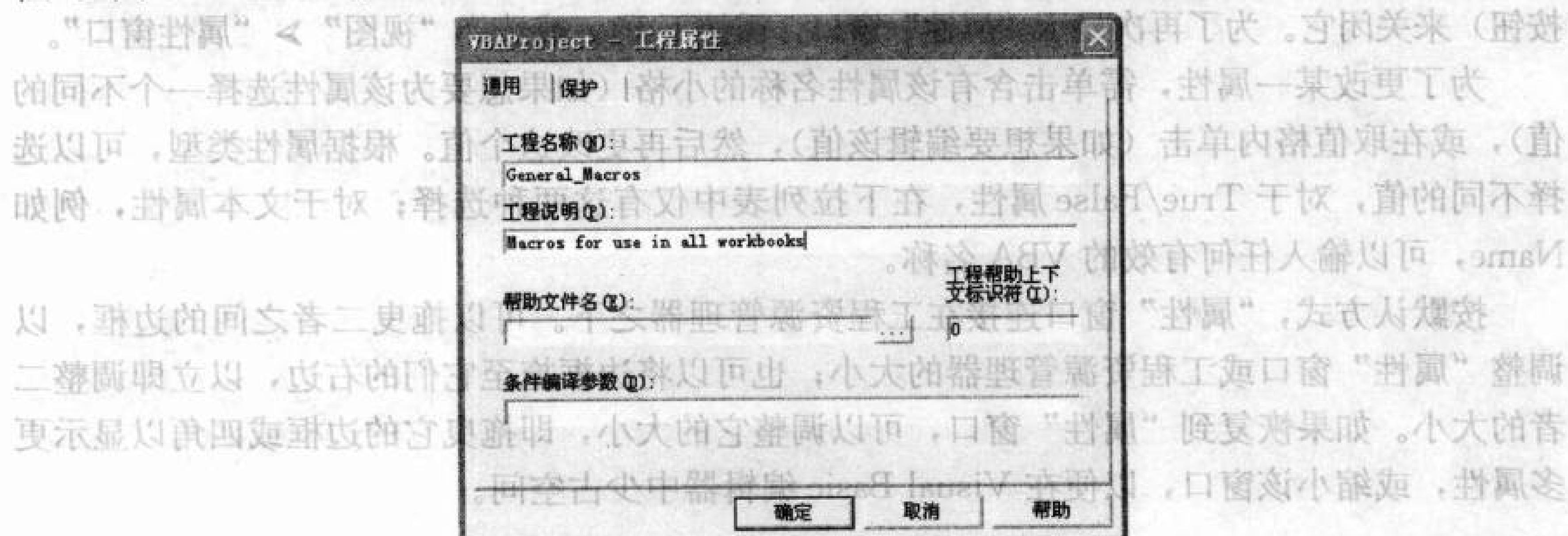


图 2.15 使用“工程属性”对话框，以查看和设置工程属性，并锁定工程以防更改

在“工程属性”对话框的“通用”页上可做如下工作：

- ◆ 在“工程名称”文本框内设置工程名。这个名称用以在对象浏览器中，必要时在

Windows Registry 中识别出该工程。确认这个名称是唯一的，以避免与其他工程混淆。从技术上来说，工程名是对应于该工程的“类型库”的名称（类型库描述该工程所包含的对象，如模块和用户窗体）；它用来构建工程中类别的完全合格的类名（本书后面要进一步讨论）。工程名中可包含有下划线，但不可包含空格。

- ◆ 在“工程说明”文本框内输入工程说明。该说明会出现在对象浏览器的说明图板上，帮助用户了解该工程为何物。说明要尽量简明。
- ◆ 为该工程指明“帮助”文件，方法是：在“帮助文件名”文本框内输入“帮助”文件的名称和路径。单击“帮助文件名”文本框右侧标有省略号（…）的按钮，以显示“帮助文件”对话框。然后选择文件并单击“打开”按钮，以便在文本框内输入“帮助”文件名。（也可以键入或粘贴名称和路径。）
- ◆ 在“工程帮助上下文标识符”文本框内为工程指定“帮助”上下文。“帮助”上下文指的是在“帮助”文件中的位置。默认的“帮助”上下文是 0，它使得“帮助”文件显示打开的“帮助”文件屏幕（如果从“运行”对话框来运行“帮助”文件，或双击资源管理器内的该文件，将看到相同的屏幕）。可以指定不同的“帮助”上下文，从而将用户带到一个特定的题目上——例如，带到与正在寻求帮助的工程有关的题目上。
- ◆ 指定工程需要的条件编译参数。

如图 2.16 所示，在“工程属性”对话框的“保护”页上可做如下工作：

- ◆ 选中“查看时锁定工程”复选框，使别人在不知道密码时不能打开工程、查看它和更改它。

- ◆ 在“查看工程属性的密码”组框内，在“密码”文本框里输入对应于本工程的密码，然后在“确认密码”文本框里输入同样的密码。单击“确定”按钮并关闭该工程。现在，如果不知道密码的话，没有人能够打开和查看（更谈不上更改）该工程。据说，Office 的安全性比较差，能够轻易被破坏。更多信息请参阅第 19 章。

提示：如果在“密码”文本框和“确认密码”文本框里输入了密码，但是没有选中“查看时锁定工程”复选框，下次在试图显示“工程属性”对话框时，Visual Basic 编辑器将提示用户要留意密码。但是，不提供密码，仍能打开和查看该工程及其内容。

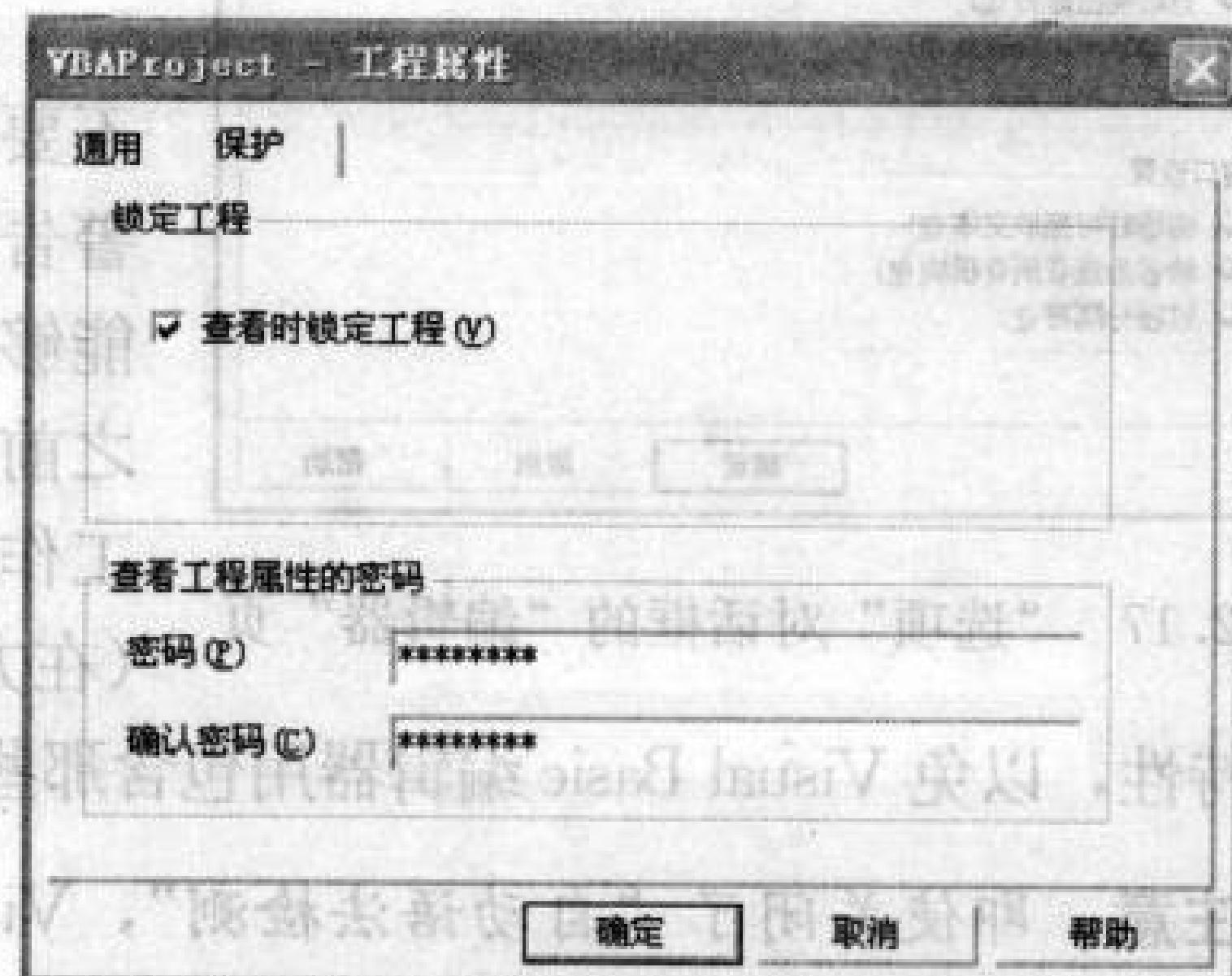


图 2.16 “工程属性”对话框的“保护”页，可用来锁定工程，使别人不能查看或编辑它

自定义 Visual Basic 编辑器

不管打算在 Visual Basic 编辑器上花多少时间，用户最好自定义此编辑器，以便使工作尽量迅速而舒服。用户可以：

- ◆ 选择在 Visual Basic 编辑器中喜欢的编辑器设置和视图设置，以便控制如何与其互动。

- ◆ 选择在 Visual Basic 编辑器中显示哪些窗口，以及如何布局它们，以便尽可能有效地使用工作空间。
 - ◆ 自定义 Visual Basic 编辑器的工具栏和菜单，使所需用的命令很方便（不搞乱工作空间）。
 - ◆ 自定义工具箱，让它包含构建用户窗体时所需要的工具。
- 下节将介绍选项。

警告： 用户所做出的自定义，会作用到使用 VBA 版本的所有宿主。例如，如果更改了以 Excel 2003 为宿主的 Visual Basic 编辑器实例中的字体，以 Word, PowerPoint, Outlook 等为宿主的 Visual Basic 编辑器实例中的字体也就相应改变了。

选择喜欢的编辑器设置和视图设置

为了着手选择喜欢的编辑器设置和视图设置，选择“工具”>“选项”以打开“选项”对话框（见图 2.17）。

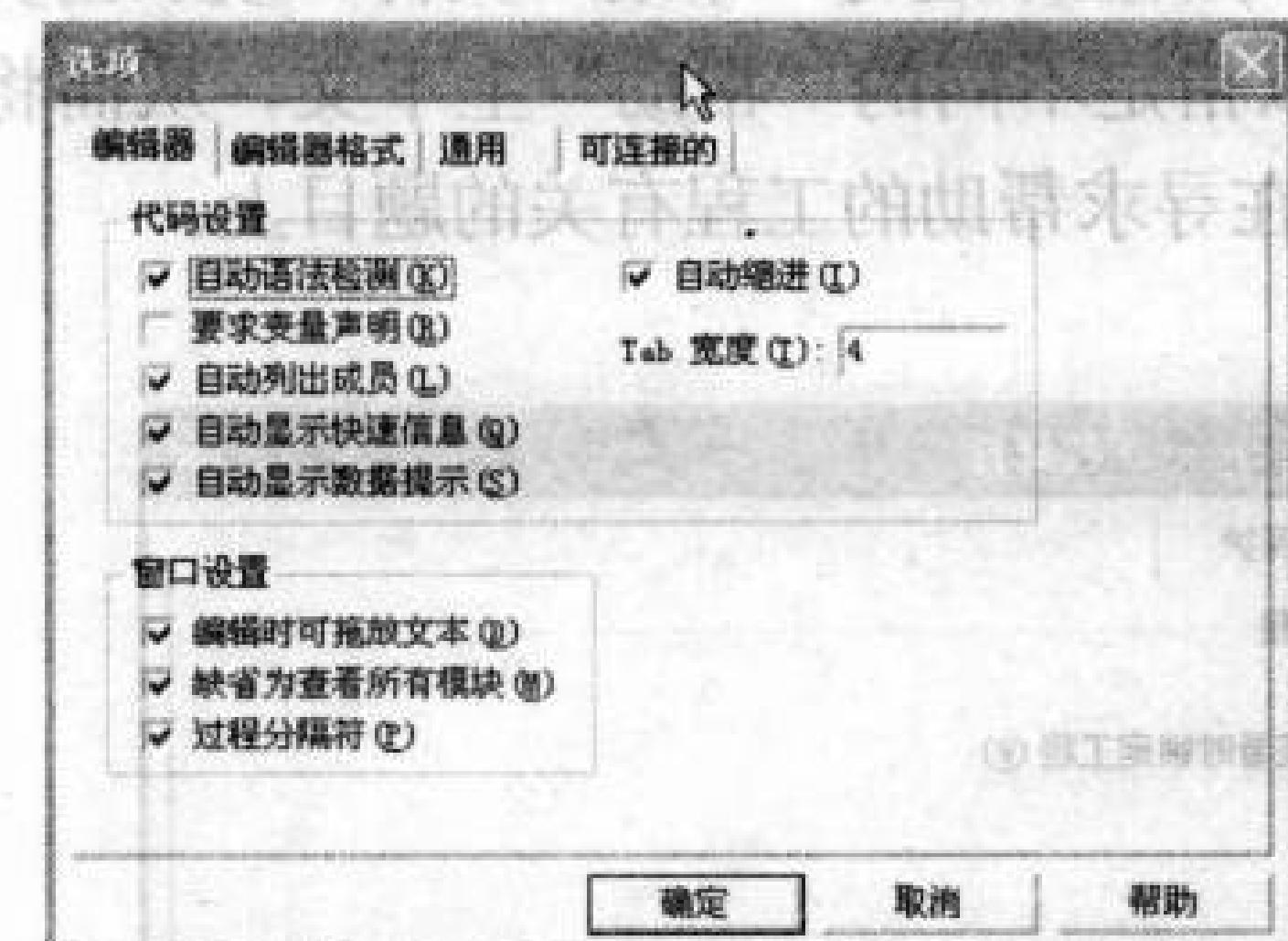


图 2.17 “选项”对话框的“编辑器”页
这一特性，以免 Visual Basic 编辑器用包含那些已发觉但未及更正的错误的信息框来打扰人。

编辑器页选项

“选项”对话框的“编辑器”页包含如下设置：

自动语法检测 它控制：当键入表达式时，要不要自动检测语法，并在发现错误时让 VBA 显示警告消息框。这一特性常常是有帮助的，因为 VBA 能够立即指出错误，否则，在试图运行或调试代码之前，往往看不到这些错误。但是，如果某用户的工作风格是喜欢从一个未完成代码行移到另一行（在方便时最终完成全部代码行），他可能宁愿关闭这一特性，以免 Visual Basic 编辑器用包含那些已发觉但未及更正的错误的信息框来打扰人。

注意： 即使关闭了“自动语法检测”，Visual Basic 编辑器仍会把出错的代码行变为红色，以引起用户对它的注意。

要求变量声明 它控制：是否必须显式声明变量。显式声明变量较隐式声明变量要多做点工作，但是显式声明变量是一个好的实践，将会为用户节省以后的时间——所以要确认已选择了这一复选框。（第 6 章中讨论如何针对变量进行工作。）

自动列出成员 它控制：当在“代码”窗口中工作时，是否让“属性/方法列表”和“常数列表”特性对属性、方法和常数的设置提出建议。很多人发现此特性是很有帮助的，但也有些经验丰富的编程者将其关闭，因为他们对自己需要的所有属性、方法和常数都很了解，不愿意受到忙碌的界面的打扰。

注意： 用户也可能想把诸如“自动列出成员”、“自动显示快速信息”和“自动显示数据提示”这类特性关闭，因为它们占据存储空间和处理能力，使计算机变慢。速度差别相对不大，但在较老的计算机上还是明显的。

自动显示快速信息 它控制：当在代码窗口中使用到函数时，是否让“快速信息”特性自动地显示有关函数及其参数的信息。

自动显示数据提示 它控制：当在中断模式中使鼠标指针在某一变量或表达式上方游移

时，是否让Visual Basic编辑器显示屏幕提示，以使用户能够快速检查该变量或表达式的值。（也可以为此使用“本地”窗口或“监视”窗口，但它们会占据更多空间。）

自动缩进 它控制：当已经让某一代码行缩进时，Visual Basic编辑器是否让后续的各代码行都缩进。当“自动缩进”接通时，Visual Basic编辑器让每个新的代码行都缩进到前一行相同程度（相同数量的制表符或空格，或相同数量的二者结合）。当“自动缩进”关闭时，Visual Basic编辑器仍在“代码”窗口的左边界处开始新的代码行。通常，“自动缩进”可以节省时间，但是，当需要减小新代码的缩进程度时，必须按下Shift+Tab，单击“编辑”工具栏上的“凸出”按钮，或删除制表符或空格。

Tab（制表符）宽度 设置一个制表符的空格数量。该设置可在1到32个空格之间调节。默认的设置是4个空格，这对默认字体来说比较适宜。如果在写代码时选用均衡型字体（如Times或Arial）而不是单空格字体（如Courier）的话，用户可能希望增加一个制表符所代表的空格数，以便代码缩进的程度很清楚。

编辑时可拖放文本 它控制：Visual Basic编辑器是否支持拖放。大多数人发现这一特性很有帮助。可以将一段代码在“代码”窗口内拖动，或从一个“代码”窗口拖曳到另一个“代码”窗口。也可以将代码拖至“立即”窗口，或将一个表达式拖至“监视”窗口。

缺省为查看所有模块 它控制：Visual Basic编辑器是显示一个程序清单中一个模块内的所有过程（“全模块”视图），还是一次只显示一个过程（“过程”视图）。如果正在一个短过程上做工作，用户可能会发现“全模块”视图很有用；对大多数其他过程而言，单个视图可减少混乱和提高效率。工作于“过程”视图时，用户从“代码”窗口顶端的“过程”下拉列表中选择出要与之工作的过程，也就打开了该过程。为了在“全模块”视图和“过程”视图之间切换，需单击“代码”窗口左下角的“全模块视图”按钮或“过程视图”按钮。

注意：当工作于“全模块”视图时，也可以使用“过程”下拉列表，以便按名称迅速地移到一个过程上去。

过程分隔符 它控制：在“代码”窗口内显示“全模块”视图时，Visual Basic编辑器是否显示一条条横线来分隔一个模块内的各个过程。通常，这些横线是有帮助的，能让人很快知道一个过程在何处结束，而下一过程从何处开始。（如果正在使用“过程”视图，这个复选框是不用考虑的。）

编辑器格式页选项

如图2.18所示的“选项”对话框的“编辑器格式”页，控制代码如何在Visual Basic编辑器中显现。

要改变各过程所使用的各种类型文本的默认颜色，需在“代码颜色”列表框内选择文本类型，再从“前景色”、“背景色”、“标识色”各下拉列表中选择颜色。以下说明“代码颜色”选择的含义：

标准文本 常用于典型过程中的文本。用户可能愿意为它选择自己习惯的颜色（如默认的黑色）。

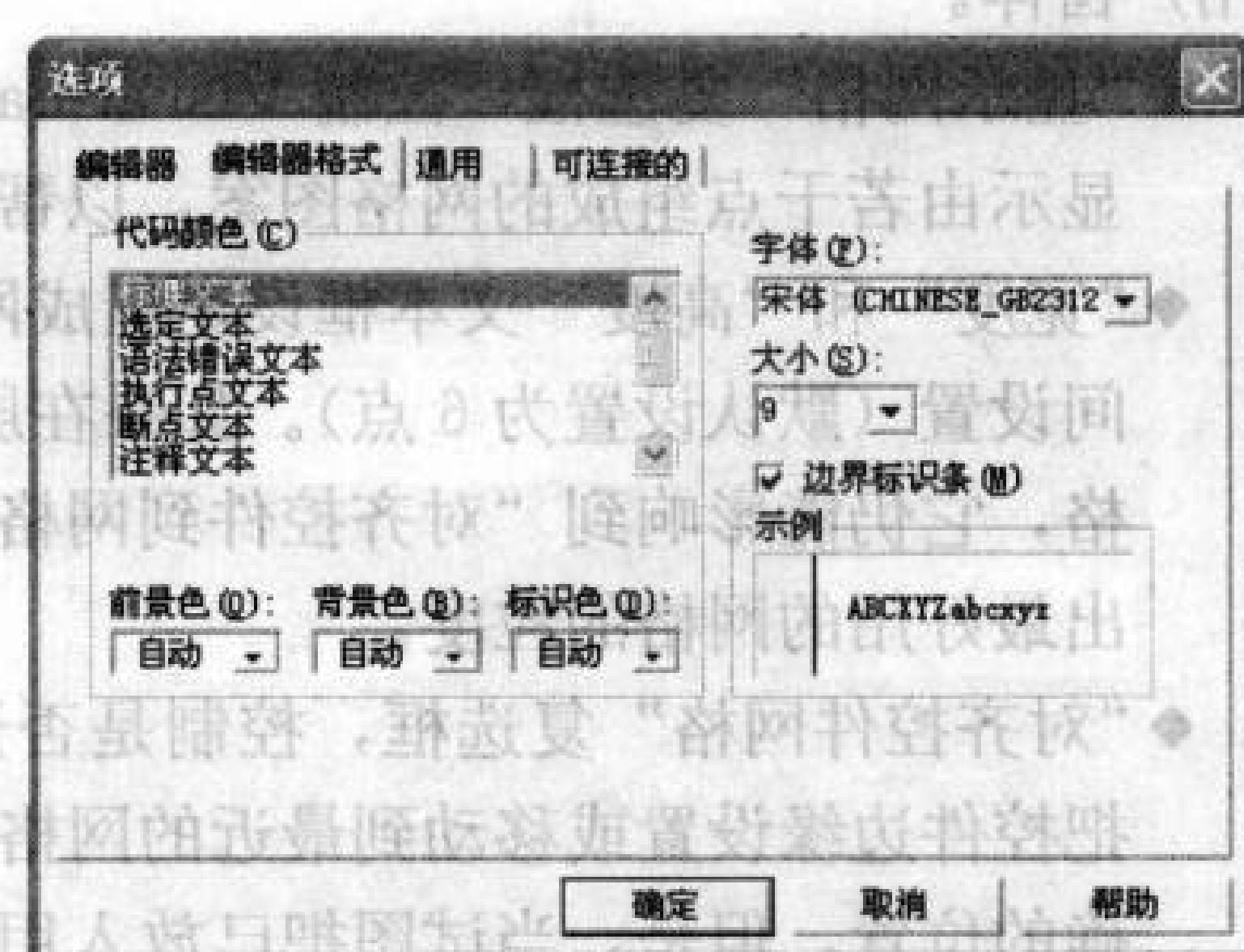


图2.18 “选项”对话框的“编辑器格式”页

选定文本 影响到被选定(高亮)文本的颜色。

语法错误文本 影响到 VBA 用于出错行的颜色。默认色为红色。

执行点文本 影响到 VBA 用于中断模式中当前执行行的颜色。通常会使其为高亮色(Visual Basic 编辑器使用萤光黄为默认色),以便能立即看出当前行。

断点文本 影响到 VBA 用于显示断点的(过程执行停止处)的颜色。

注释文本 影响到注释行的颜色。默认色为深绿色。

关键字文本 影响到关键字(认作为 VBA 语言的一部分的字)的颜色。这种文本在每个过程中有相当大份额。关键字以不同于标准文本的颜色来显示,因为这有助于不读代码就能认出关键字。默认色为深蓝色。

标识符文本 影响到 VBA 用于标识符的颜色。标识符包括定义的变量名,常数名和过程名。

书签文本 影响到 VBA 用于代码中书签的颜色。

调用返回文本 影响到 VBA 用于调用其他过程的颜色。Visual Basic 编辑器使用灰绿色作为调用返回文本的默认色。

使用“编辑器格式”页内的“字体”下拉列表和“大小”下拉列表,可以改变“代码”窗口内所有文本的字体和大小。还可以避免显示边界标识条(在标识条中会有下一语句和断点图标之类项目出现),方法是清除“边界标识条”复选框。(通常,这些图标是有帮助的,但是,去掉该标识条可使屏幕上代码区稍有增加。)

通用页选项

如图 2.19 所示的“选项”对话框的“通用”页,包括有若干项设置,在下面各节中分组进行讨论。

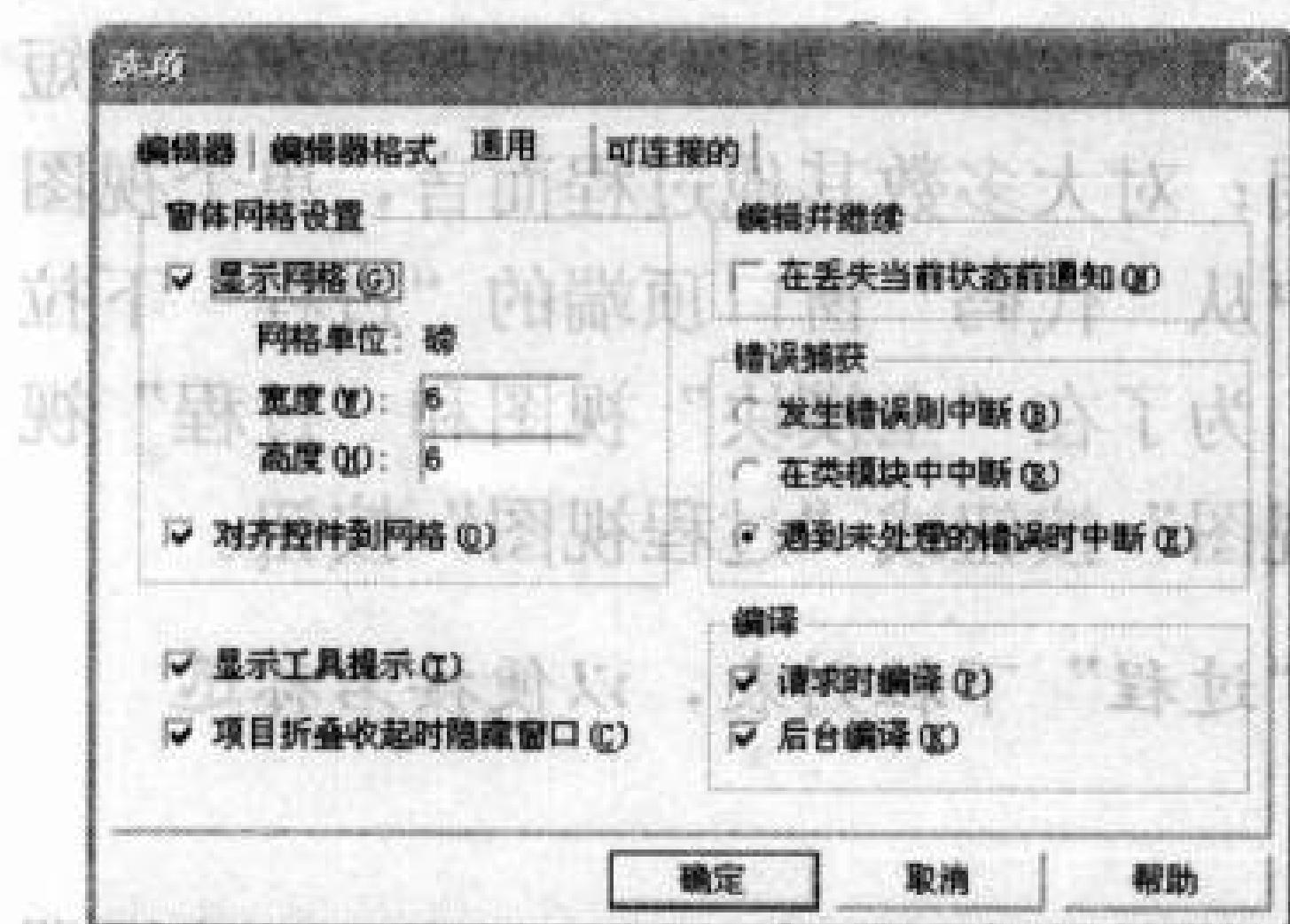


图 2.19 “选项”对话框的“通用”页
窗体网格设置组框

窗体网格设置控制 Visual Basic 编辑器如何处理用户窗体。

- ◆“显示网格”复选框,控制 Visual Basic 编辑器是否要在设计模式中,在用户窗体内,显示由若干点组成的网格图案,以帮助放置和对齐控件。这一复选框是默认选中的。
- ◆“宽度”和“高度”文本框设定组成网格的点的间隔。可以在 2 点至 60 点的任何数值间设置(默认设置为 6 点)。如果在屏幕上显示网格,将看到很多点;如果不显示网格,它仍能影响到“对齐控件到网格”特性,这在下面会讨论。用户可以通过试验找出最好用的网格的粒度。
- ◆“对齐控件网格”复选框,控制是否让 Visual Basic 编辑器自动抓住控件边缘,以便把控件边缘设置或移动到最近的网格线。这一选项使人能迅速而容易地把控件放到正确的位置,但是,当试图把已放入用户窗体的控件的布局加以改进时,会带来麻烦。(如果这样的话,可以选择清除“对齐控件到网格”复选框,另一选择是仍选中它,但减小网格的大小。)

编辑并继续组框

“编辑并继续”组框只含一个控件——“在丢失当前状态前通知”复选框。这个复选框控制：当正在运行代码时，如果用户试图采取某一行动，而这一行动要求 VBA 重新设定模块中所有变量的值，此时 Visual Basic 编辑器是否对用户发出警告。

错误捕获组框

“错误捕获”组框包含三个选项按钮，使用它们来指明，VBA 应如何处理运行代码时出现的错误。

发生错误则中断 它告诉 VBA，在碰到任何错误时都要进入中断模式，不论错误处理程序（为处理错误而设计的一段代码）是否已激活，或代码是否在类模块中。“发生错误则中断”对精确定位错误发生在何处是有用的，它有助于跟踪并排除错误。如果代码中已有错误处理程序，可能不需要这一选项。

在类模块中中断 有人认为这是日常应用中最有用的选项。当 VBA 遇到类模块（是定义对象类型的模块）中一个未处理错误时，VBA 在出错代码行处进入到中断模式。

遇到未处理的错误时中断 这是默认设置，当已构建了一个处理当前模块中可预测错误的错误处理程序时，这是很有用的。如果有了这个错误处理程序，VBA 允许由该程序来捕获错误而不进入中断模式；但如没有针对所产生的错误的处理程序的话，VBA 会在出错代码行处进入中断模式。不过，类模块中的未处理错误，会引起工程在调用该类中的出错过程的代码行上进入中断模式，这就能让人识别出（从而修改）引起问题的代码行。

编译组框

“编译”组框控制 VBA 何时为工程编译出可执行代码。在执行代码之前需要做编译；但是，在 Visual Basic 编辑器启动执行第一部分代码之前，并不是工程的所有代码都必须编译好。

如果希望 VBA 仅在需要时才编译代码，可选择“请求时编译”复选框。VBA 只是在启动执行某一过程之前，编译用户正运行的该过程的代码，而不编译同一模块内其他过程的代码，除非在运行时要调用这些过程。结果是，只要 VBA 完成了对某一过程的代码的编译，用户就可以先在模块中开始执行该过程。如果该过程以后要调用模块中另一过程，VBA 可以在第一个过程调用第二个过程时，编译第二个过程的代码，而不必在开始运行第一个过程时编译。

“请求时编译”通常是个好选项。当用户已经在模块中构建了很多过程，而其中一些已经是半完成代码时，这个选项特别有用。反之，如果清除了“请求时编译”复选框，在启动执行用户欲运行的过程之前，VBA 就必须编译好模块中所有过程的全部代码。这意味着，不仅过程启动的时间要推后（代码越多，编译所花时间越长），而且，模块中任何过程内的语言错误或编译错误，都会使用户无法运行当前过程，即使当前过程代码并无错误。

设有一个名为 Compilation 的模块，它含有两个过程，GoodCode 和 BadCode，表示如下：

```
Sub GoodCode()
    MsgBox "This code is working."
End Sub

Sub BadCode()
    Application.Delete
End Sub
```

GoodCode 只是显示一个消息框，指出它正在工作，而 BadCode 包含一个无效语句（Application 对象没有 Delete 方法）。GoodCode 运行没有发生问题，但 BadCode 每次都会引起错误。

如果试图在打开“请求时编译”后运行 GoodCode，该过程运行正常；VBA 编译 GoodCode 中的代码，没有发现错误，就运行它。但是，如果试图在关闭“请求时编译”后运行 GoodCode 的话，在启动运行 GoodCode 之前，VBA 也要编译 BadCode 中的代码——在无效语句 Application.Delete 处，它停下来了，有了一个编译错误。在运行代码之前进行全面检查，这对一道工作的已完成的各模块是件好事，但是，当正对某一个模块中的代码进行试验时，它就会出问题。

另一方面，可以看到在运行 GoodCode 之前调用 BadCode 编译模块中全部代码的好处。这个调用表示在该过程这一版本的第 3 行中：

```
Sub GoodCode()
    MsgBox "This code is working."
    BadCode
End Sub
```

这里，在启动运行 GoodCode 之前编译 BadCode 中的代码是个好主意，因为这样做之后，如果 BadCode 包含错误，GoodCode 就不运行了。如果运行此版本的 GoodCode 同时，让“请求时编译”接通，VBA 就编译 GoodCode 并启动运行它，显示出第 2 行中的消息框。在第 3 行中 BadCode 被调用，引起 VBA 去编译 BadCode，而在此处 VBA 因编译错误而停止。我们并不希望在一个编译过程的中间发生这种事，在这种情况下，不如让“请求时编译”关闭。

当选择“请求时编译”复选框时，“后台编译”复选框可以使用，它控制：当已经编译好的代码正在运行时，Visual Basic 编辑器是否利用 CPU 的空余时间来编译另外的代码。应该让“后台编译”接通，除非怀疑这减慢了代码的执行。

显示工具提示和项目折叠收起时隐藏窗口

“选项”对话框的“通用”页上最后两个选项是“显示工具提示”和“项目折叠收起时隐藏窗口”。“显示工具提示”复选框，控制 Visual Basic 编辑器是否要显示对应于它的工具栏按钮的工具提示（屏幕提示）。工具提示应该是有用的，除非非常想节省出它们会耗费的存储器和处理器时间。

“项目折叠收起时隐藏窗口”复选框，控制项目折叠收起时，Visual Basic 编辑器是否隐藏代码窗口和其他工程窗口。这个复选框是默认选中的，一般来说，这是个有用的特性。当在工程资源管理器中折叠收起某一工程时，Visual Basic 编辑器把属于该工程的“代码”窗口或用户窗体窗口隐藏起来，并将它们从“窗口”菜单上出现的列表中移出。当再次展开该工程时，Visual Basic 编辑器在它们原先的位置上显示这些窗口，并使它们在“窗口”菜单列表上恢复出现。

可连接的页选项

如图 2.20 所示的“选项”对话框的“可连接的”页，控制 Visual Basic 编辑器中的各种窗口是否是可连接的——这就是说，当移动它们时，它们是否能自动附着到窗口一旁。保持这些窗口为可连接的，对拥有更多的可管理界面往往是有利的。但是，用户可能想使某些窗口为不可连接的，这样，就可以在必要时把它们拖离屏幕边缘，并由用户随意安排。

编辑器窗口的选择和布局

下面，可以选择怎样对 Visual Basic 编辑器中的各窗口进行布局。用户的布局在很大程度上取决于屏幕的尺寸和分辨率，以及他个人的爱好，但是这里有几条建议：

- ◆ 要使“代码”窗口总最大化。如果写了很长的代码行，而只在迫不得已的情况下才把它们分成合理的长度时，会希望在 Visual Basic 编辑器窗口内有尽可能大的空间。
- ◆ 很多时间用户都在努力地写代码，可以暂不理睬工程资源管理器，只有在需要时才调用它。让它重新显示的一个方便办法，是将工程资源管理器命令放在“代码窗口”、“代码窗口中断”、“监视窗口”、“立即窗口”和“本地窗口”上下文菜单上。（在下节中，将学习如何做到这一点。）
- ◆ 如果不连接某些窗口，可以将它们收入 Visual Basic 编辑器底部的图标中。
- ◆ 如果正在使用多监视器配置，会希望能够将子窗口拖出 Visual Basic 编辑器父窗口之外，再拖至第二个监视器。遗憾的是，不能在父窗口的界限外拖得很远。但是，将 Visual Basic 编辑器窗口从右手边的监视器扩展到左手边的监视器，可以达到类似的效果，然后可以将“属性”窗口和工程资源管理器窗口连接到左手边的监视器。菜单栏和工具栏的显现会受不利影响，但是，用户将拥有更大的“代码”窗口空间，而且所有这三个窗口均可使用。

自定义工具栏和菜单栏

Visual Basic 编辑器支持与 Office 应用程序相同的工具栏和菜单栏定制，使人能够自定义 Visual Basic 编辑器界面，让所有需要的命令都很便手。

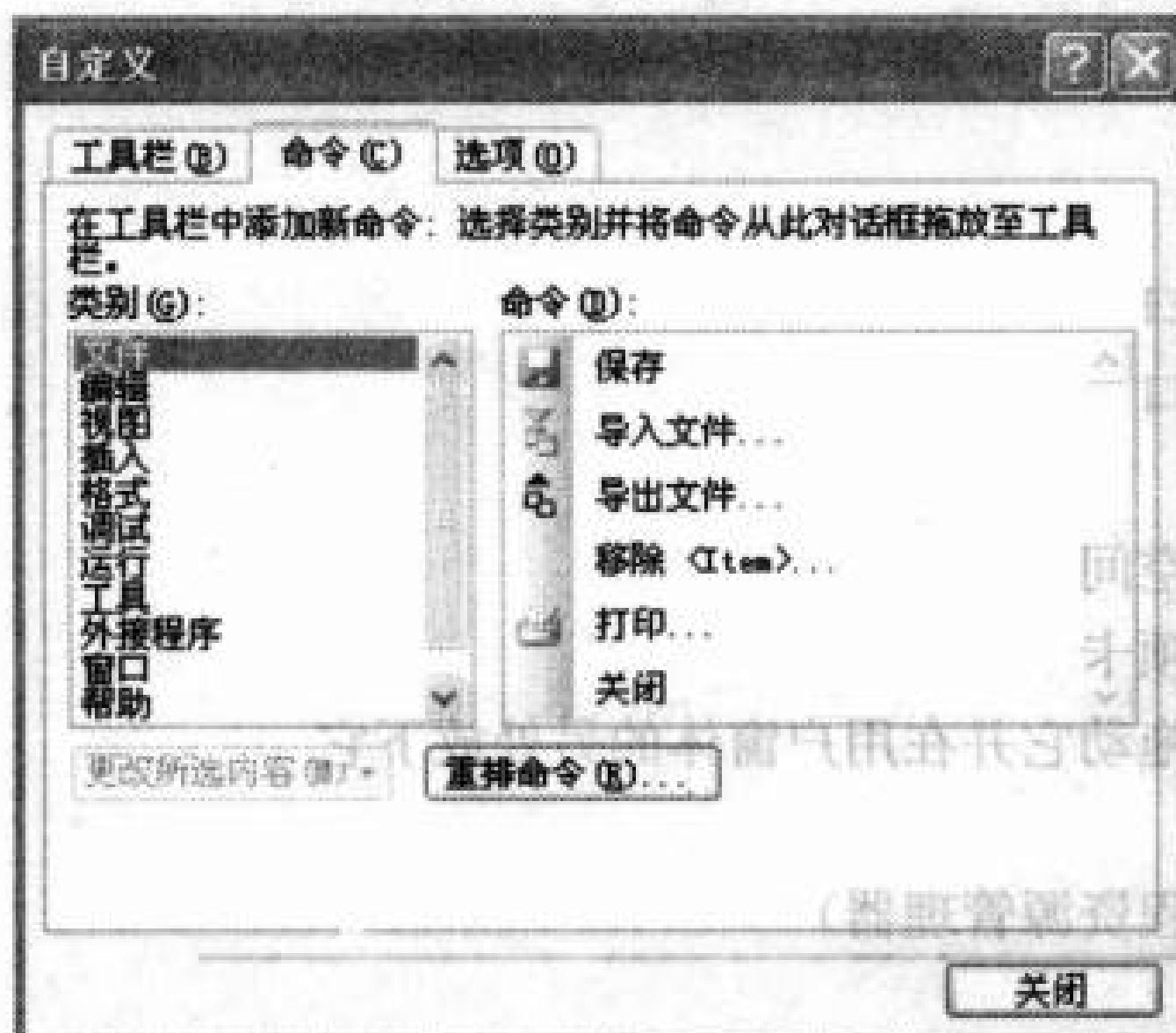


图 2.21 使用“自定义”对话框，以自定义 Visual Basic 编辑器的菜单栏、工具栏和上下文菜单

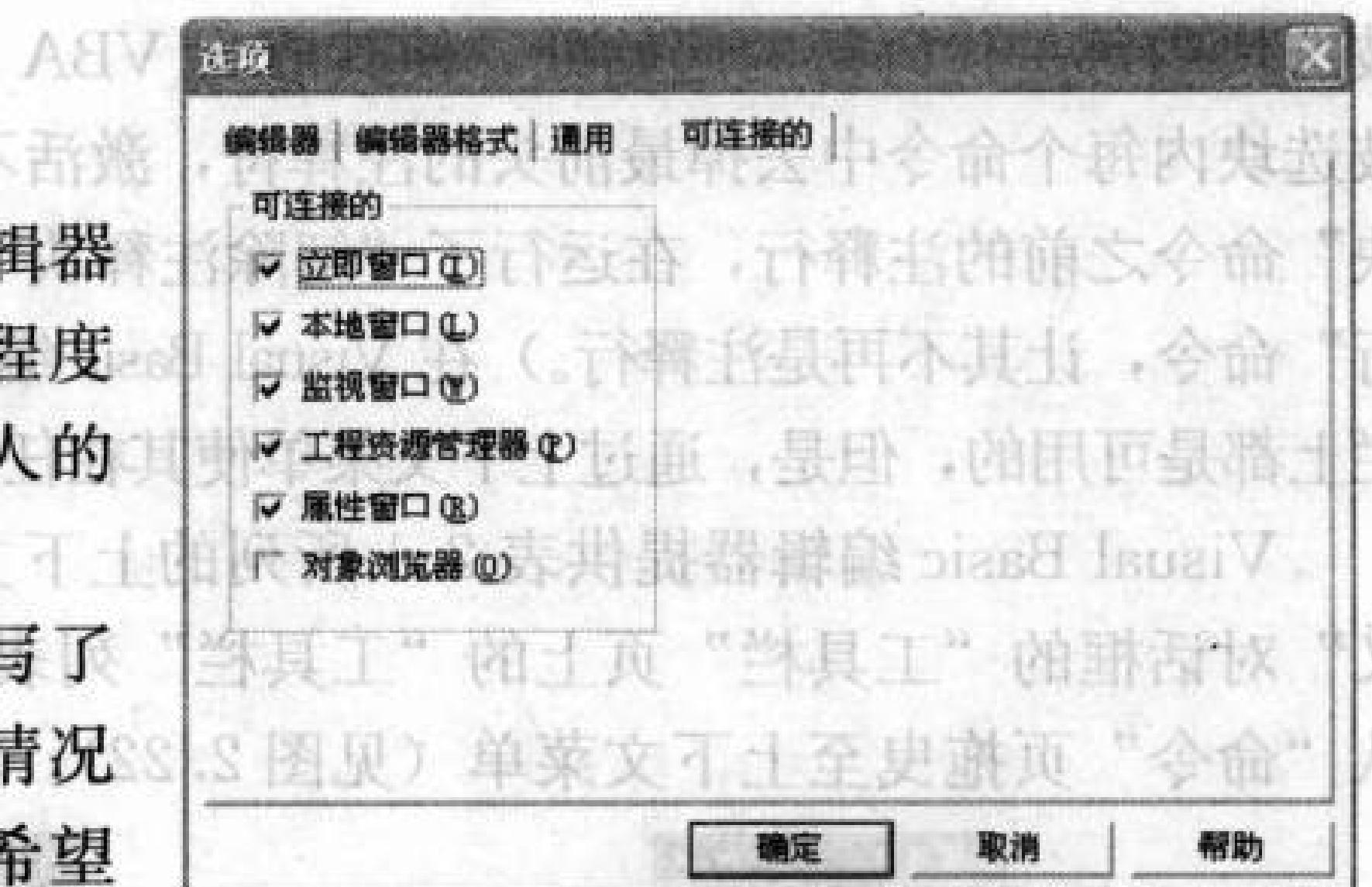


图 2.20 “选项”对话框的“可连接的”页

为了自定义 Visual Basic 编辑器，选择“视图”>“工具栏”>“自定义”（或右击已显示的工具栏或菜单栏，并从上下文菜单中选择“自定义”），以显示“自定义”对话框，如图 2.21 所示。

注意：与 Office 应用程序不同的是，Visual Basic 编辑器不能让人创建新菜单或自定义键盘快捷键。

现在可以自定义适应自己工作方式的工具栏、菜单和上下文菜单了。尤其是，如果使用上下文菜单的话，一定要将它们定制成能提供所需要的各种命令。

你可能特别希望把两个重要命令添加到上下文菜单：“设置注释块”和“解除注释块”。“设置注释块”命令，是把注释符（'）添加到

被选块中每一代码行的起始处，使其成为 VBA 不执行的注释。“解除注释块”命令，是从被选块内每个命令中去掉最前头的注释符，激活不再有注释符的各代码行。（运行“设置注释块”命令之前的注释行，在运行了“解除注释块”之后仍为注释行。可以再次运行“解除注释行”命令，让其不再是注释行。）在 Visual Basic 编辑器的正常配置中，这些命令在“编辑”工具栏上都是可用的，但是，通过上下文菜单使其在任何时间在“代码”窗口内可用，要省事许多。

Visual Basic 编辑器提供表 2.1 所列的上下文菜单。为了自定义上下文菜单，在“自定义”对话框的“工具栏”页上的“工具栏”列表内，选择“快捷菜单”，然后把想要的命令从“命令”页拖曳至上下文菜单（见图 2.22）。

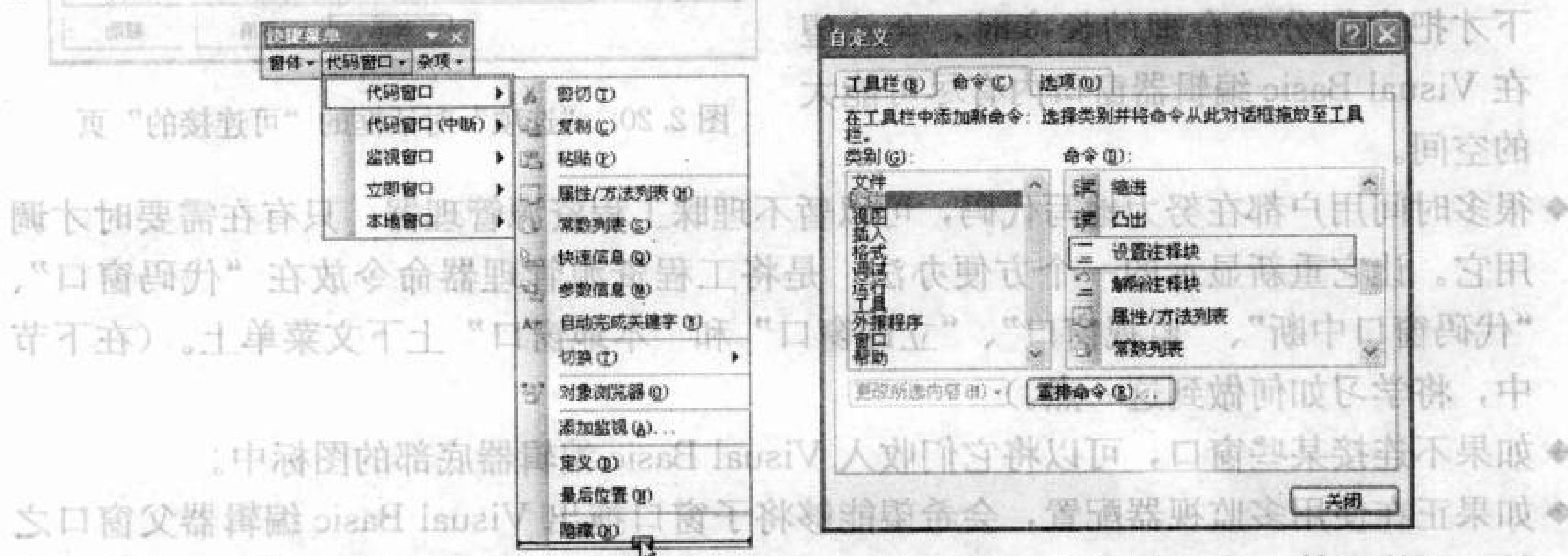


图 2.22 使用“快捷菜单”工具栏，把重要命令置入 Visual Basic 编辑器的上下文菜单

表 2.1 Visual Basic 编辑器中的上下文菜单

上下文菜单	当右击时，出现
MSForms	一个用户窗体
MSForms 控件	用户窗体上的一个控件
MSForms 控件组	用户窗体上的一个控件组
MSForms MPC	用户窗体上的一个多页控件
代码窗口	处于设计模式下的代码窗口
代码窗口 (中断)	处于中断模式下的代码窗口
监视窗口	监视窗口
立即窗口	立即窗口
本地窗口	本地窗口
工程窗口	处于设计模式下的工程窗口
工程窗口 (中断)	处于中断模式下的工程窗口
对象浏览器	对象浏览器
MSForms 调色板	工具箱内一个页面上的净空间
MSForms 工具箱	工具箱内一个页面上的选项卡
MSForms DragDrop	用户窗体上的一个项目：拖动它并在用户窗体的另处放下它
属性浏览器	属性窗口中的一个属性
泊定的窗口	一个泊定的窗口（例如工程资源管理器）

下面给出对于自定义 Visual Basic 编辑器的几条建议：

- ◆ 如果经常使用“本地”窗口，在单步执行代码时跟踪变量的值的话，可为此在工具栏上设一个按钮，总是保持显示它（标准按钮在“调试”工具栏上），或者为此在对应于“代码窗口”（设计模式和中断模式）、“监视窗口”和“立即窗口”的上下文菜单上放一个项目。

- ◆ 将“监视”窗口和“立即”窗口放在对应于要调用它们的窗口的上下文菜单上。
- ◆ 如果有一个中等大小的监视器，可以考虑把要用的所有工具栏按钮组合在一个工具栏中，这样就不会由于显示多个工具栏而浪费了空间。

自定义工具箱

也可以自定义工具箱，它是一种专门的工具栏，包含有构建用户窗体的各个控件，并且只有在用户窗体被选中时才能使用。（第 14 和 15 章介绍如何构建用户窗体。）

可以通过添加控件，删除控件和增加新的“工具箱”页来自定义工具箱。在自定义时，可以把最常用的控件放在工具箱中，并尽可能放在一个页面上，以节省自己的时间。这些控件将包括常规工具箱控件的自定义变体；将它们放在工具箱，可以避免再次定制它们。

例如，所创建的很多对话框都需要有“确定”按钮，以便于从对话框退出，完成某些代码，然后继续执行某过程。每个“确定”按钮都要求把它的 Name 属性设置为 cmdOK，它的 Caption 属性设置为 OK，它的 Default 属性设置为 True，它的 Height 和 Width 属性设置为小于 Visual Basic 编辑器默认指定的一个尺寸。一旦自定义了某个命令按钮，就可以把它的复件放在工具箱，在后续窗体中再使用它。

自定义工具箱的另一个理由是增添先进控件，这些控件能使我们可以利用对话框和用户窗体做更多的事情。

添加控件至工具箱

添加控件的首要方式，是将控件直接从某个用户窗体添加到工具箱去。例如，一旦完成了自定义“确定”和“取消”按钮，就可以将它们从用户窗体复制到工具箱，这样，在以后创建的任何用户窗体中，都可以再使用它们。

为了把一个控件从某个已显示的用户窗体复制到工具箱，只需按如图 2.23 所示那样拖放它即可。（第 14 章介绍如何将控件放入自己创建的用户窗体。）



图 2.23 添加控件至工具箱的最快方法，是把它从用户窗体拖出

为了将控件添加到工具箱，先确定想把控件加到哪个页面，在该页内右击（在本章后面“添加页面至工具箱”中，将学习如何将页面加至工具箱），并从上下文菜单中选择“附加控件”，以显示“附加控件”对话框，如图 2.24 所示。在“可用控件”列表框内，选中与欲加至工具箱的那些控件对应的复选框，然后单击“确定”按钮。（如果想让该列表



图 2.24 在“附加控件”对话框中，选中对应于要添加的控件的复选框，然后单击“确定”按钮

只留有当前选中的项目，在“显示”组框中选择“只显示所选项”复选框即可。)

可以把一个控件从工具箱内的一页移至另一页。这只需将它从所在页上拖出，让鼠标指针在目的页的标签上游移（保持拖动）以显示目的页，然后将鼠标指针移入目的页体内（仍保持拖动），并放下此控件。

自定义工具箱

为工具箱控件重命名

当让鼠标指针在工具箱中一个控件上方游移时，屏幕提示会出现，以显示该控件的名称。要给它重命名，需在工具箱内右击它，并从上下文菜单中选择“自定义”项，以显示“自定义控件”对话框。（菜单项以控件的名称来标识——例如，如果该控件被标识为“新标签”，菜单项即称为“自定义新标签”。）在“工具提示”文本框内，键入控件名称（如有必要，可删除或更改其现用名称）；当用户让鼠标指针在工具箱内该控件上方游移时，屏幕提示出现此名称。然后，如果愿意的话，可以为该控件的工具箱图标指定一个不同的图片，这在下节中要讲到。否则，可单击“确定”按钮，以关闭“自定义控件”对话框。

为控件的工具箱图标指定图片

工具箱内的每个控件都以一个图片来识别。可以为控件指定新图片，方法是：显示出“自定义控件”对话框，单击“加载图片”按钮，并在弹出的对话框中选择图片或图标。

注意：可以使用专用图标编辑器（如 freeware IconEdit 32，可从 www.pcmag.com 下载）来创建自己的图标。

对于某些控件，可以用下述方法来编辑指定给控件的图片：显示“自定义控件”对话框，单击“编辑图片”按钮；再用“编辑图形”对话框来为组成图片的像素选颜色。

从工具箱中删除控件

为了从工具箱中删除某一控件，需右击该控件，并从上下文菜单中选择“删除”项目。项目是以控件名来识别的——例如，如果右击了一个名为“公司名组合框”的控件，菜单项目就称为“删除公司名组合框”。

如果该项目是一个自定义控件，本操作会删除这一控件，并不能恢复它（除非在另处有个复印件）。如果该项目是微软支持的控件（由 Microsoft Forms 2.0 包——它是 VBA 的一部分——来支持），那么，只要选中对应于适当对象（如 Microsoft Forms 2.0 命令按钮）的复选框，就可以从“附加控件”对话框中恢复它。

也可以通过删除某控件所在页的整个页面，把这个控件从工具箱中删除掉。请参看本章后面的“从工具箱中删除页面”。

添加页面至工具箱

要向工具箱添加一个页面，需右击页顶端的选项卡，并从上下文菜单中选择“新页”。Visual Basic 编辑器把这个页面命名为“新页”，它添加“选择对象”控件到该页面。这个控件出现在工具箱的每一页上（所以总是能用得很顺手），而且不能移掉它。

有可能会需要立即为新页重新命名。

为工具箱页重命名

要更改工具箱页名，需右击其选项卡或标签，并从上下文菜单中选择“重命名”，以显示“重命名”对话框。在“标题”文本框中键入名称，在“控件提示”文本框中键入控件提示文本，并单击“确定”按钮以关闭此对话框。

从工具箱中删除页面

要从工具箱中删除一个页面，需右击其选项卡，并从上下文菜单中选择“删除”页。Visual Basic 编辑器将从工具箱删除该页，这无需确认，也不管该页是否含有控件。

导入和导出工具箱页面

如果需要与别人共用“工具箱”页，可以将它们保存为单独的文件，并把它们分配给同事。“工具箱”页具有.pag 文件扩展名。

为了导入“工具箱”页，需右击工具箱内一个现有页上的选项卡，并从上下文菜单中选择“导入页”，以显示“导入页”对话框。选择想要导入的页面，并选择“打开”按钮。Visual Basic 编辑器会在工具箱最后一个页面之后，添加上该新页，并命名为“新页”。右击该页的选项卡，选择“重命名”，键入新页名和说明，然后单击“确定”按钮。

类似地，也可以导出“工具箱”页，右击它的选项卡，并从上下文菜单中选择“导出页”，以显示“导出页”对话框。键入该页的名称，选择保存它的文件夹，然后单击“保存”按钮来保存它。现在，任何人都可以照前述方法导入该页了。

在工具箱内移动页面

要移动工具箱中某一页，需右击它的选项卡，并从上下文菜单中选择“移动”，以显示“页顺序”对话框。在此对话框内，选择欲移动的一页或数页（Shift+单击以选择连贯的多页，Ctrl+单击以选择单独的多页），并使用“上移”和“下移”按钮，根据意愿来重新安排页面。做完后，单击“确定”按钮，以关闭“页顺序”对话框。

关闭 Visual Basic 编辑器并返回到主应用程序

结束了 Visual Basic 编辑器的编辑工作之后，可以关闭 Visual Basic 编辑器，也可以让它运行，但切换至另外的窗口：

- ◆ 要关闭 Visual Basic 编辑器，选择“文件”>“关闭并返回到<应用程序>”，按下 Alt+Q 键，或单击 Visual Basic 编辑器窗口的“关闭”按钮。
- ◆ 让 Visual Basic 编辑器运行，但在另外的应用程序中工作，可以使用任务栏或按下 Alt+Tab 键，以切换到其他应用程序。

命令重页群工具式

第 3 章 编辑已录制的宏

- ◆ 在 Visual Basic 编辑器中测试宏
- ◆ 设置断点和使用注释
- ◆ 编辑已录制的 Word 宏
- ◆ 编辑已录制的 Excel 宏
- ◆ 编辑已录制的 PowerPoint 宏

在本章中，将使用 Visual Basic 编辑器来编辑在第 1 章中使用宏录制器在 Word、Excel 和 PowerPoint 中录制的宏。

在 Visual Basic 编辑器中针对宏进行工作，有如下三条理由：

- ◆ 首先，是为了在执行已录制宏的过程中发现和确定问题。例如，如果在录制宏时有一个步骤出错，每次运行宏的时候，它总是会执行那个错误的指令，除非去除或更改该指令。
- ◆ 第二，是为了向宏添加更多的指令，以使其操作有所不同（以前所述）。这是一个着手了解 VBA 的重要途径，因为，对已录制宏做出相对简单的更改，往往能极大地提高它的功能和灵活性。
- ◆ 第三，是为了用 Visual Basic 编辑器中编写宏的方式，而不是录制宏的方式来创建新宏。

可以根据实际情况，从头开始编写新宏，或利用已有宏的部分成果来编写新宏。

注意：即使没有可支持宏录制器的应用程序，仍应好好阅读这一章，因为它介绍了如何使用 Visual Basic 编辑器的一些重要编辑特性。

在 Visual Basic 编辑器中测试宏

如果从主应用程序来运行一个宏遭到失败的话，找出错在哪里的最快方法是在 Visual Basic 编辑器中打开这个宏，运行它，看看错在什么地方。

1. 在主应用程序中，按下 Alt+F8 键，或选择“工具”>“宏”>“宏”，以显示“宏”对话框。
2. 选中该宏，然后单击“编辑”按钮。主应用程序打开 Visual Basic 编辑器的这一实例，并显示该宏以便编辑。
3. 运行宏，方法是按下 F5 键，或选择“运行”，或单击 Visual Basic 编辑器中“标准”工具栏上的“运行子过程/用户窗体”按钮（如图 3.1）。
4. 如果宏出错和崩溃，VBA 会在屏幕上显示错误消息框，并在“代码”窗口中选出有错的语句。可以编辑该语句以确定问题所在。这样做过之后，再按下节所述方法，单步执行这个宏。

警告：要在用户不看的文件（或文件的复件）上测试宏，以防因在文件上使用未经测试的宏而破坏对用户有价值的工作。把代码保存在中心位置（如 Word 中的 Normal.dot, Excel 中的个人宏工作簿，或 PowerPoint 中仅有代码的演示文稿），这样，它就可以让所有文件，而不是仅仅让包含它的文件来访问。如果在不恰当的文件中创建了一个宏，应将它从这个文件中导出，再导入到中心存储里去。为了导出宏，需在工程资源管理器中右击其模块，从快捷菜单中选择“导出文件”，使用“导出文件”对话框以指明文件夹和文件名，然后单击“保存”按钮。为了导入模块，需在工程资源管理器中右击目的工程，在“导入文件”对话框中选择“导入文件”，然后单击“打开”按钮。

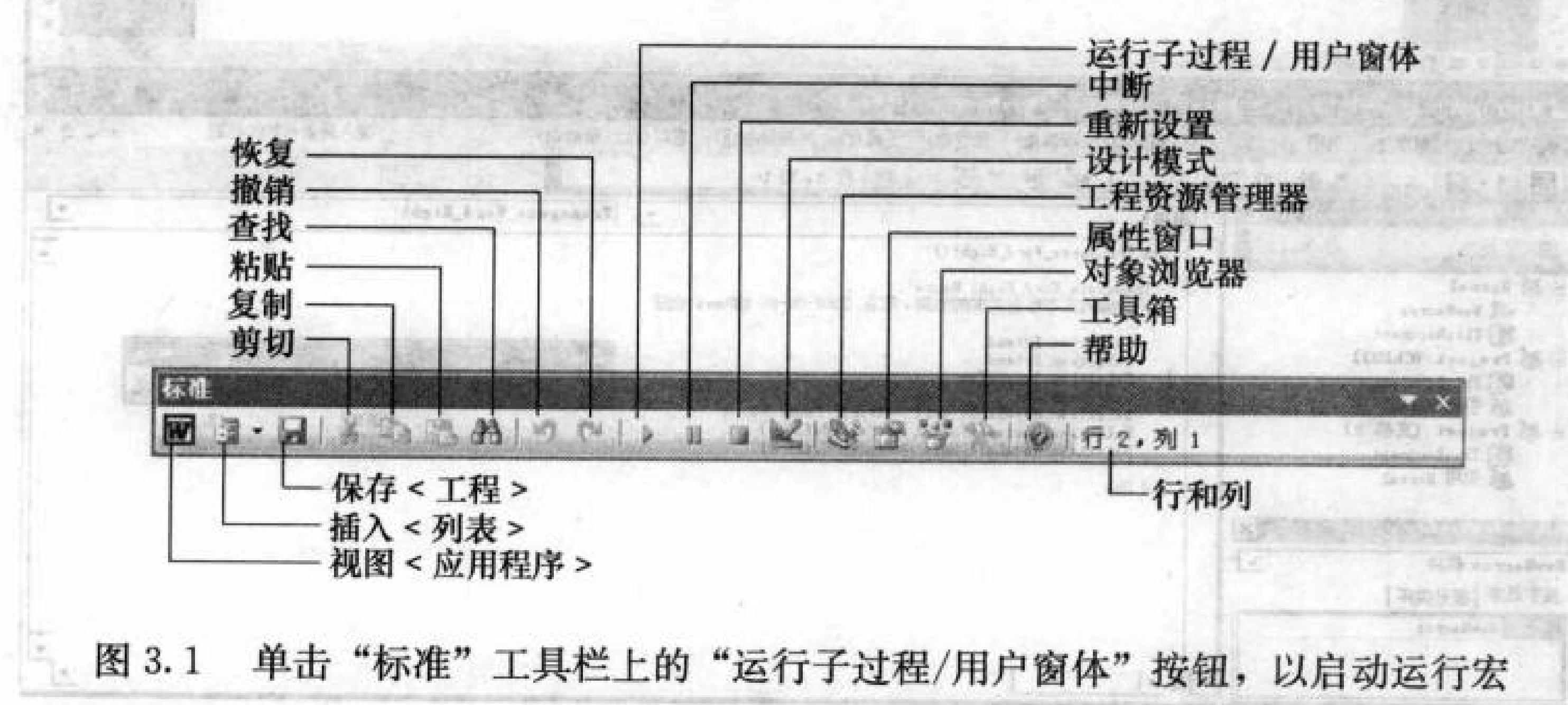


图 3.1 单击“标准”工具栏上的“运行子过程/用户窗体”按钮，以启动运行宏

单步执行宏

为了看看某个宏到底在做什么（以及错在哪里），可以单步执行这个宏，即从头到尾，一次一条命令地执行——这样就能看到每条命令的效果。单步执行宏是很耗时的，但是它能发现和确定问题所在。

单步执行宏，遵循如下步骤：

1. 打开主应用程序，然后打开宏以进行编辑：按下 Alt+F8 键，或选择“工具”>“宏”>“宏”，选中该宏，然后单击“编辑”按钮。
2. 安排 Visual Basic 编辑器窗口和主应用程序的窗口，使二者均能看到。可以手工进行安排，也可以使用 Visual Basic 命令来安排。例如，如果 Visual Basic 编辑器窗口和主应用程序的窗口，是仅有的两个已打开且未最小化的窗口，可右击 Windows 任务栏的背景，并从上下文菜单中选择“横向平铺窗口”或“纵向平铺窗口”。
3. 根据宏的需要，将插入点定位在应用程序窗口一个适当位置上。（例如，可能需要将插入点定位在一个特定的地方，或选择一个对象，以使宏能够适当运行。）
4. 单击 Visual Basic 编辑器窗口以激活它，再将插入点定位在欲运行的宏上。
5. 按下 F8 键，以便逐条命令地单步执行宏。每按一次 F8 键，VBA 代码就执行一行。当执行某行时，Visual Basic 编辑器使该行增亮，所以，用户可以在应用程序窗口中监视执行效果并发现错误。

提示：也可以选择“调试”>“逐语句”或在“调试”工具栏上单击“逐语句”按钮，来单步执行宏，但是最常用的还是使用 F8 键。

图 3.2 提供了单步执行录制在 Word 里的宏的例子。在第 17 章里将要详细学习如何调试宏。现在介绍两个可尝试迅速解决宏出现的问题的简捷技术：设置断点和在代码中插入注释行。

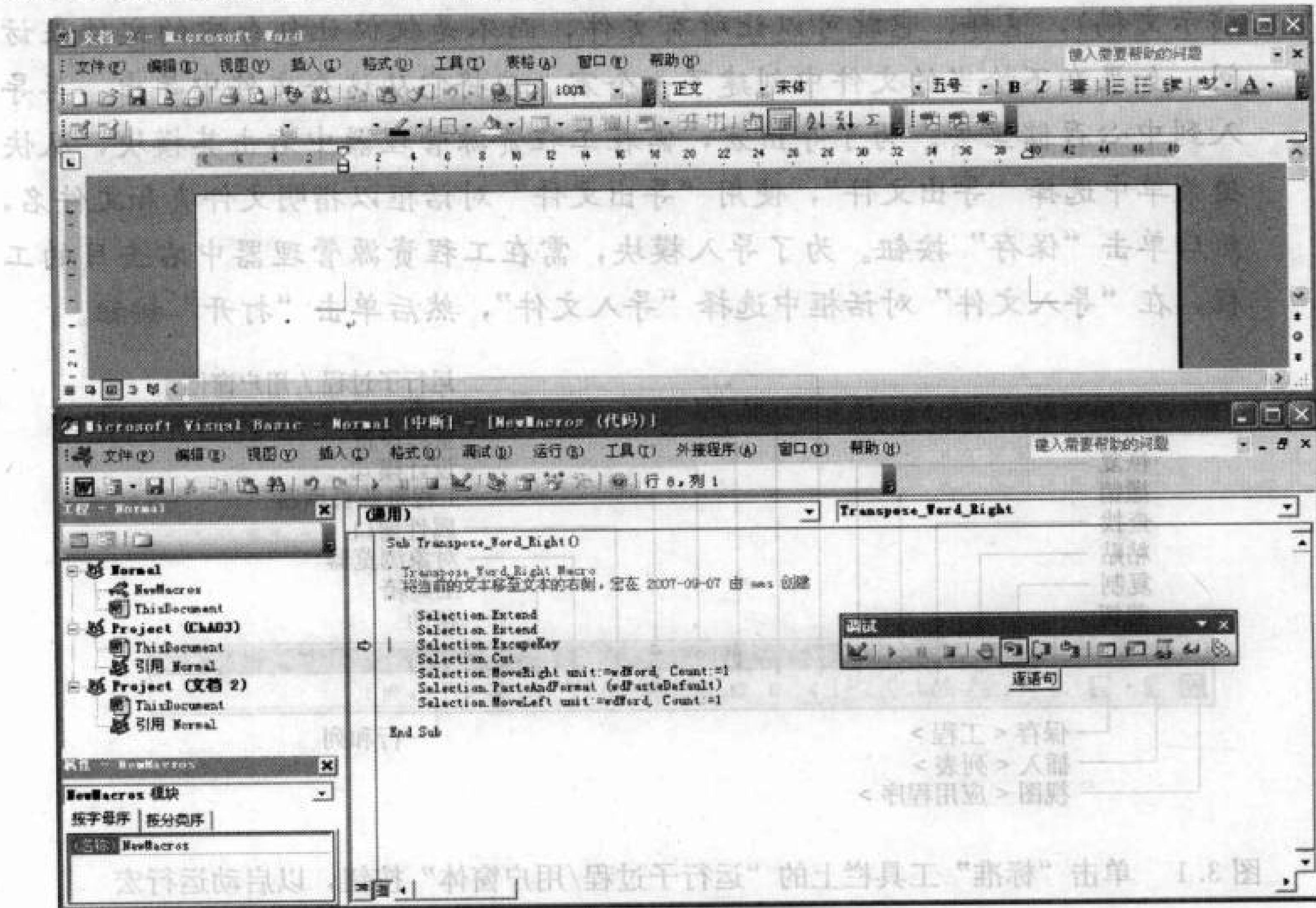


图 3.2 为了发现宏的错误，安排好应用程序窗口和 Visual Basic 编辑器窗口，使二者均能看到。然后按下 F8 键或使用“逐语句”命令来单步执行宏

设置断点

断点是程序人员设置在代码行里的触发开关，它告诉 VBA 在此处停止宏的执行。使用断点，可以在运行宏的时候全速略过宏的各功能部分，而停在用户想要开始监视代码逐语句执行的地方。

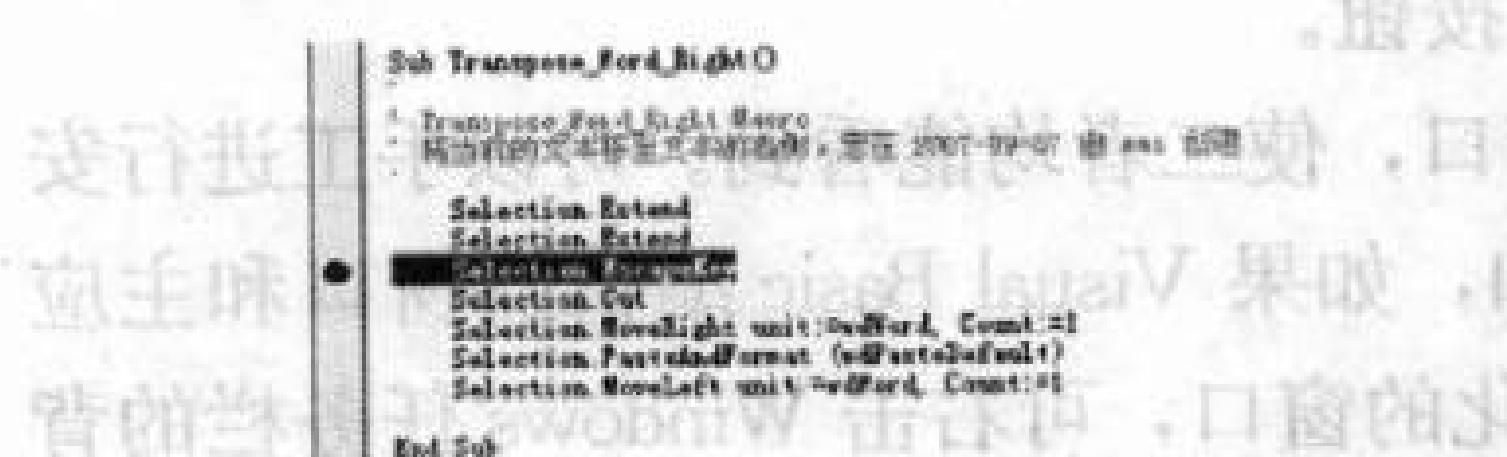


图 3.3 使用断点（以出现在边界指示条内的深色圆圈来指明），以便在所选择的行上停止代码执行

为了切换断点，在一个可执行代码行（不是注释行）内右击，并从上下文菜单中选择“切换”>“断点”，或单击“编辑”工具栏上的“切换断点”按钮。也可以在紧挨该代码行的边界标识条内单击。按默认方式，设有断点的那一行的颜色会变深。断点本身以边界标识条内一个深色圆圈来指明（见图 3.3）。

注意：断点是临时的——Visual Basic 编辑器不把它们和代码一起保存。必须在每次编辑时放置它们。

插入注释行

和大多数编程语言一样，VBA 让程序人员在代码中添加注释，以使代码易于理解。在

创建代码时，注释是很有价值的；当程序人员很久之后再看自己的代码时，由于时间过久，会忘记了该代码是干什么的——或者，更糟的是，要搞清楚别人的代码是干什么的，这时候，注释显得很宝贵。

也可以把那些不想让 Visual Basic 编辑器执行的代码行设成注释行。有些代码行是有疑问的，但实际上还没有从宏里面移出，对排除这种代码来说，将其改为注释是一种有用的技术。

要用手工方法指明某一行是注释，只需要在该行代码前面键入注释符（'）。通常，把注释符放在一行的起始处是最容易的。也可以使用 Rem 来代替注释符。（Rem 是 remark 即“附注”的缩写，注释行有时也称“附注行”。）要用手工方法解除注释行，删除注释符或 Rem 即可。

Visual Basic 编辑器提供“设置注释块”和“解除注释块”命令，以自动进行有关注释设置和解除的操作。选择代码行（或在想对其施加影响的单行内单击），然后单击“编辑”工具栏上的“设置注释块”按钮，以便在每行的起始处放置注释符；要解除注释行，需单击“解除注释块”按钮，这样，Visual Basic 编辑器便从每行中去除注释符。

“设置注释块”和“解除注释块”命令只针对注释符做工作，不能对 Rem 行起作用。如果用户偏爱使用 Rem，必须用手工方法设置和解除注释行。

注意：“设置注释块”命令，是逐个把注释符加到被选块中每一行，包括已经加过注释符的那些行的起始处。同样，“解除注释块”命令，是逐个从被选块的每一行去除注释符，不是一次去除所有注释符。这一方式有助于保持注释行，并能让人使用不同等级的注释方法。

跳出某个宏

一旦已经发现和确定了宏里面的问题，用户可能不愿意再逐条命令地去单步执行该余下部分。为了运行宏的余下部分和要调用该宏的任何余下部分，可以按下 F5 键，单击“标准”工具栏或“调试”工具栏上的“继续”按钮（如图 3.4 所示），或选择“运行”>“继续”。如果只是想运行这个宏的余下部分，然后再返回到单步执行调用这个宏的宏，就使用“跳出”命令。“跳出”命令可以全速结束当前宏或过程的执行，但是，如果此后的代码随另一个过程继续的话，Visual Basic 编辑器便反转到中断模式，让用户能够考查这个过程的代码。

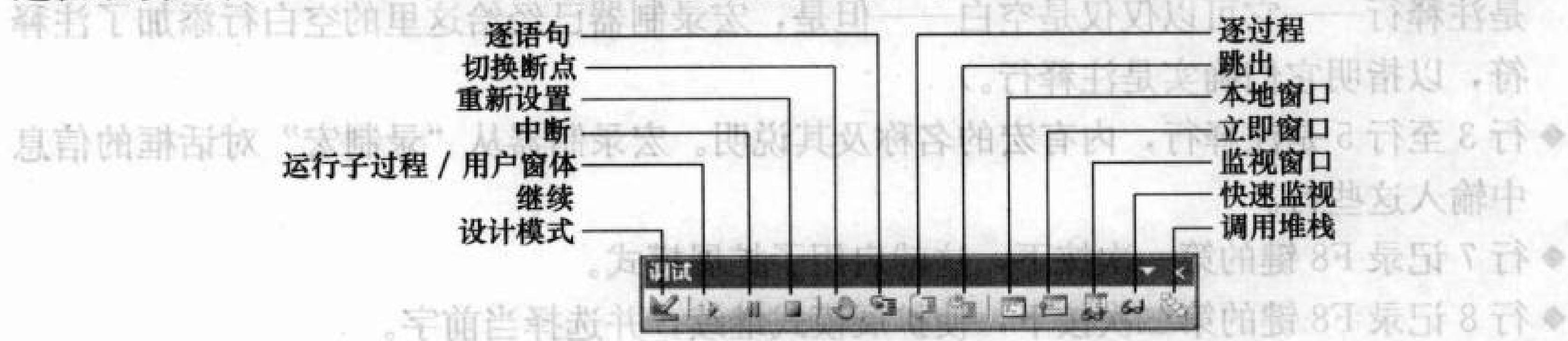


图 3.4 “调试”工具栏包含运行代码、逐语句执行、跳出等命令，及显示用于调试的重要窗口

要发出“跳出”命令，可按下 Ctrl+Shift+F8 键，或单击“调试”工具栏上的“跳出”按钮，或选择“调试”>“跳出”。

编辑 Word 宏

现在来编辑在 Word 中录制的宏 Transpose_Word_Right，并使用它来构建另一个宏。

首先，在 Visual Basic 编辑器中打开该宏：

1. 启动 Word，如果它不在运行的话；或激活 Word。
2. 按下 Alt+F8 键，或选择“工具”>“宏”>“宏”，以显示“宏”对话框。
3. 选中 Transpose_Word_Right 宏，然后单击“编辑”按钮。

在“代码”窗口，应能看到类似于程序清单 3.1 的代码，只是没有行号。行号是本书为了识别代码行而加上的。

程序清单 3.1

```

1. Sub Transpose_Word_Right()
2. ' Transpose_Word_Right Macro
3. ' Transposes the current word with the word to its right. -
4. ' Created 6/1/2006 by Joanna Bermudez.
5. '
6. Selection.Extend
7. Selection.Extend
8. Selection.EscapeKey
9. Selection.Cut
10. Selection.MoveRight Unit:=wdWord, Count:=1
11. Selection.Paste
12. Selection.MoveLeft Unit:=wdWord, Count:=1
13. Selection.Cut
14. End Sub

```

以下说明该宏是干什么的：

- ◆ 行 1 以 Sub Transpose_Word_Right()语句来表示宏的开始，行 14 则以 End Sub 语句来结束该宏。Sub 行和 End Sub 行分别标志宏的始与终。（与它们用于任何子过程时一样）。
- ◆ 行 2 和行 6 是空白的注释行，宏录制器插入它们，是为了让宏便于阅读。可以在宏里使用任意多的空白行或空白注释行，以便把所有语句分隔成若干组。（空白行不一定是注释行——它可以仅仅是空白——但是，宏录制器已经给这里的空白行添加了注释符，以指明它们确实是注释行。）
- ◆ 行 3 至行 5 是注释行，内有宏的名称及其说明。宏录制器从“录制宏”对话框的信息中输入这些行。
- ◆ 行 7 记录 F8 键的第一次按下，这就启用了扩展模式。
- ◆ 行 8 记录 F8 键的第二次按下，使扩展模式继续，并选择当前字。
- ◆ 行 9 记录 Escape 键的按下，它取消扩展模式。
- ◆ 行 10 记录“剪切”命令，它将 Selection 即选择内容（在此例中是被选的字）剪切到剪贴板。
- ◆ 行 11 记录 Ctrl+→命令，它将插入点右移一个字。
- ◆ 行 12 记录“粘贴”命令，它将被选的字粘贴到文档中的当前插入点位置处。
- ◆ 行 13 记录 Ctrl+←命令，它将插入点左移一个字。

首先，将行 13 变成注释行，用户已经录制了这一行，所以可以由此构建另一个宏 Transpose_Word_Left。在该行的起始处键入一个注释符——在该指令起始处前面的任何

地方均可，但是，最容易的是将注释符输入在最左列处，那里最为清晰可见：

```
' Selection.MoveLeft Unit:=wdWord, Count:=-1
```

也可以先单击第13行内的任意地方，再单击“设置注释块”命令，让Visual Basic编辑器来输入注释符。

当把插入点从第13行移出时，VBA检查该行，将它识别为注释行，并将它的颜色更改成当前为注释文本设定的颜色。当运行宏时，VBA忽略这一行，不执行该命令，所以，Word窗口中的插入点并不像它以前做的那样左移一个字。

单步执行 Transpose_Word_Right 宏

试一试使用“逐语句”命令在中断模式中单步执行这个宏。

1. 安排屏幕，使既能看到活动的Word窗口，又能看到Visual Basic编辑器窗口（例如，可右击任务栏，并从上下文菜单中选择“横向平铺窗口”或“纵向平铺窗口”）。
2. 在Visual Basic编辑器内单击，然后将插入点置于“代码”窗口内的Transpose_Word_Right宏之中。
3. 按下F8键以单步执行代码，每次执行一个活动行。可以注意到，VBA跳过空白行和注释行，因为它们不是活动行。当按下F8键时，VBA使当前语句增亮，同时能看到发生在Word窗口内的操作。

到达宏的结束处时（本例是执行到行14的End Sub语句时），Visual Basic编辑器关闭中断模式。可通过单击“标准”工具栏或“调试”工具栏上的“重新设置”按钮，或选择“运行”>“重新设置”实现，也可以在任何时刻退出中断模式。

运行 Transpose_Word_Right 宏

如果该宏在单步执行时工作正常，用户可能想从Visual Basic编辑器来运行它，这就要单击“标准”工具栏或“调试”工具栏上的“运行子过程/用户窗体”按钮。也可以单击这个按钮（此时它被认做“继续”，而不是“运行子过程/用户窗体”），从中断模式来运行该宏，它从当前指令开始运行。

创建 Transpose_Word_Left 宏

现在，只要对Transpose_Word_Left宏做少量调整，就可以创建一个Transpose_Word_Left宏。

1. 在“代码”窗口中，选中Transpose_Word_Right宏的所有代码，从Sub Transpose_Word_Right()行到End Sub行。选择时，可以使用鼠标拖曳的方法，可以保持Shift键按下并使用箭头键来扩展选择内容，或将插入点定位在该宏的一端，按下Shift键，再单击宏的另一端。
2. 发出一个“复制”命令来复制该代码（例如，右击并从上下文菜单中选择“复制”，或按下Ctrl+C键或Ctrl+Insert键）。
3. 将插入点移到“代码”窗口中Transpose_Word_Right宏的End Sub语句下面那一行。
4. 发出一个“粘贴”命令来粘贴该代码（例如，右击并从上下文菜单中选择“粘贴”，或按下Ctrl+V键或Shift+Insert键）。Visual Basic编辑器自动在Transpose

Word_Right 宏的 End Sub 语句和粘贴的新宏之间输入一个横行。

- 通过编辑 Sub 行，把第二个 Transpose_Word_Right 宏的名称更改为 Transpose_Word_Left：

```
Sub Transpose_Word_Left()
```

- 相应地编辑宏起始处的注释行——例如：

```
'Transpose_Word_Left Macro
```

```
'Transposes the current word with the word to its left.
```

```
'Created 6/1/2006 by Joanna Bermudez.
```

- 现在需要做的事是把 MoveRight 方法用 MoveLeft 方法来取代，以便将插入点左移一个字而不是右移一个字。可以用键入校正字符的方法，或者使用“剪切”和“粘贴”，把注释行之外的 Selection.MoveRight 换成 Selection.MoveLeft，或者尝试使用“属性/方法列表”特性，即：

- ◆ 单击以将插入点定位于 MoveRight 方法。
- ◆ 单击“编辑”工具栏上的“属性/方法列表”按钮，以显示属性和方法列表。
- ◆ 双击 MoveLeft 方法，以使其取代 MoveRight 方法。

- 从宏的结束处删除作为注释的 Selection.MoveLeft 行，因为不再需要它了，甚至也不需要它做引用。

工作结束了！获得了程序清单 3.2 所示的宏。

程序清单 3.2

```
Sub Transpose_Word_Left()
```

```
'Transpose_Word_Left Macro
```

```
'Transposes the current word with the word to its left.
```

```
'Created 6/1/2006 by Joanna Bermudez.
```

```
Selection.Extend
```

```
Selection.Extend
```

```
Selection.EscapeKey
```

```
Selection.Cut
```

```
Selection.MoveLeft Unit:=wdWord, Count:=-1
```

```
Selection.Paste
```

```
End Sub
```

试一试单步执行这个宏，以确认它工作正常。如果正常的话，可以准备将它保存起来——可能要为它在 Word 中创建一个工具按钮，菜单项，上下文菜单项，或者快捷键。

保存作品

当完成了针对上述宏或任何其他宏的工作时，文档或模板中已包含了所创建的宏，文档或模板也有了改动，此时需从 Visual Basic 编辑器中选择“文件”>“保存”，以保存文档或模板。然后按下 Alt+Q 键，或选择“文件”>“关闭并返回到 Microsoft Word”，以关闭 Visual Basic 编辑器，并返回到 Word。

编辑 Excel 宏

在本节中，要编辑在第 1 章中录制的 Excel 宏。这次，不用创建一个新宏——只需要向已有的宏添加一些东西。

取消隐藏个人宏工作簿

在编辑 Excel 宏之前，如果个人工作簿当前是被隐藏的话，必须取消隐藏：

1. 选择“窗口”>“取消隐藏”，以显示“取消隐藏”对话框。
2. 选择 PERSONAL.XLS，并单击“确定”按钮。

注意：如果已经把宏保存在另一个工作簿，而不是个人宏工作簿里，在试图往下做之前，打开那个工作簿。为了在编辑宏之后重新隐藏个人宏工作簿，在个人宏工作簿为活动工作簿时，选择“窗口”>“隐藏”。

打开宏以便编辑

采取如下步骤打开宏，以便查看和编辑：

1. 选择“工具”>“宏”>“宏”（或按下 Alt+F8 键），以显示“宏”对话框。
2. 选择名称为 New_ Workbook _ with _ Months 的宏。（如果已经给该宏取了一个与此不同的名称，则选择那个名称。）
3. 单击“编辑”按钮，在 Visual Basic 编辑器内显示该宏以便编辑。程序清单 3.3 显示出应该看到的代码。

程序清单 3.3

```
1. Sub New_ Workbook _ with _ Months()
2.
3. ' New_ Workbook _ with _ Months Macro
4. ' Sample macro that creates a new workbook and enters a year's worth of months
   ' into it. Recorded 12/2/05.
5.
6.
7.     Workbooks.Add
8.     Range("A1").Select
9.     ActiveCell.FormulaR1C1 = "Jan-2006"
10.    Range("B1").Select
11.    ActiveCell.FormulaR1C1 = "Feb-2006"
12.    Range("A1:B1").Select
13.    Selection.AutoFill Destination:=Range("A1:L1"), Type:=xlFillDefault
14.    Range("A1:L1").Select
15.    ActiveWorkbook.SaveAs Filename:= _
   "C:\Documents and Settings\Jack Ishida\My Documents\Sample Workbook.xls", _
   FileFormat:=xlNormal, Password:="", WriteResPassword:="", _
   ReadOnlyRecommended:=False, CreateBackup:=False
16. End Sub
```

以下说明在程序清单 3.3 的宏里面有何事发生：

- ◆ 行 1 以 Sub New_ Workbook_ with_ Months() 语句来表示宏的开始，行 16 则以 End Sub 语句来结束该宏。
- ◆ 行 2、5 和行 6 是宏录制器自动添加的注释行。（行 6 里的注释行似乎多余，之所以如此，是因为 Excel 允许在“录制宏”对话框内的“说明”文本框中输入两行，但该宏只用了一行。）
- ◆ 行 3 是注释行，给出宏的名称，说明它是一个宏；行 4 包含来自“录制宏”对话框的说明。
- ◆ 行 7 是在 Workbooks 集合对象上使用 Add 方法来创建一个新的空白工作簿。（所谓一个集合对象，或者更简洁地说，一个集合，也是一个对象，它又包含某一给定类型的若干对象。）
- ◆ 行 8 选择 Range 对象 A1，使单元格 A1 为活动单元格。
- ◆ 行 9 在该活动单元格中输入 Jan-2006。注意：宏录制器已经保存了分拆的日期值，而不是被输入的文本（January 2006）。同样要记住，单元格里显示的日期，其格式可能不同于 MMM-YYYY。
- ◆ 行 10 选择 Range 对象 B1，使单元格 B1 为活动单元格；行 11 在该单元格中输入 Feb-2006。
- ◆ 行 12 选择区域 A1: B1。
- ◆ 行 13 在区域 A1: L1 内实施默认的自动填充操作；行 14 选择该区域。注意：宏录制器已经录制了两个独立的行动，尽管在 Excel 界面里，实施的只是一次行动。
- ◆ 行 15 在给定的名称和文件夹之下保存工作簿。注意：宏录制器已经使用了延续符，即一个空格后跟一条下划线，自动地将这个长语句拆分成四行。可以在各关键字之间的任何地方拆分语句，以使代码行有合适的长度，便于工作。

编辑这个宏

现在，通过如下步骤来展开这个宏：

1. 选中行 8 至行 13。
2. 按下 Ctrl+C 键，或单击“标准”工具栏上的“复制”按钮，或选择“编辑”>“复制”或在选定范围内右击并从上下文菜单中选择“复制”，以发出“复制”命令。
3. 按下→键使选定范围脱离选择，并将插入点移至原选定范围的末尾处，即使得插入点定位于行 14 的起始处。
4. 按下 Ctrl+V 键，或单击“标准”工具栏上的“粘贴”按钮，或选择“编辑”>“粘贴”，或在插入点处右击并从上下文菜单中选择“粘贴”，以发出“粘贴”命令。

得到的新宏应如程序清单 3.4 所示。

程序清单 3.4

- ```

1. Sub New_ Workbook_ with_ Months()
2.
3. ' New_ Workbook_ with_ Months Macro
4. ' Sample macro that creates a new workbook and enters a year's worth

```

```

 of months into it. Recorded 12/2/05.
5. '
6. '
7. Workbooks.Add
8. Range("A1").Select
9. ActiveCell.FormulaR1C1 = "Jan-2006"
10. Range("B1").Select
11. ActiveCell.FormulaR1C1 = "Feb-2006"
12. Range("A1:B1").Select
13. Selection.AutoFill Destination:=Range("A1:L1"), Type:=xlFillDefault
14. Range("A1").Select
15. ActiveCell.FormulaR1C1 = "Jan-2006"
16. Range("B1").Select
17. ActiveCell.FormulaR1C1 = "Feb-2006"
18. Range("A1:B1").Select
19. Selection.AutoFill Destination:=Range("A1:L1"), Type:=xlFillDefault
20. Range("A1:L1").Select
21. ActiveWorkbook.SaveAs Filename:=_
 "C:\Documents and Settings\Jack\My Documents\Sample Workbook.xls", _
 FileFormat:=xlNormal, Password:="", WriteResPassword:="", _
 ReadOnlyRecommended:=False, CreateBackup:=False
22. End Sub

```

现在采取如下步骤来更改这个宏：

1. 删除行 16，它没有什么用处，还在“代码”窗口内占据空间。
2. 删除行 20，对于宏所干的事来说，它不是必需的——不需要该宏选择这个区域，因为行 13 中的自动填充指令已经足够了，它可以实施自动填充操作而无需去选择这个区域。
3. 将行 14 的选择单元格 A1 改成选择单元格 A2：  
`Range("A2").Select`
4. 更改行 15，使其输入数值 100 以取代 Jan-2006：  
`ActiveCell.FormulaR1C1 = 100`

5. 更改行 16 以选择单元格 B2 而不是单元格 B1：  
`Range("B2").Select`
6. 更改行 17，使其输入数值 200 以取代 Feb-2006：  
`ActiveCell.FormulaR1C1 = 200`
7. 更改行 18，使其选择区域 A2:B2：  
`Range("A2:B2").Select`
8. 更改行 19，使其在区域 A2:L2 实施自动填充操作：  
`Selection.AutoFill Destination:=Range("A2:L2"), Type:=xlFillDefault`

9. 使用空格和下划线，以及在 Type 参数之前按回车，将行 13 分拆成两行，如下所示。  
 第二行缩进一个制表符位置：

```

Selection.AutoFill Destination:=Range("A1:L1"), _
 Type:=xlFillDefault

```

10. 同样地，使用空格和下划线，以及在 Type 参数之前按回车，将行 19 拆分。

11. 单击“保存”按钮，或选择“文件”>“保存”，以保存所做出的更改。

该宏显示如程序清单 3.5 所示。

### 程序清单 3.5

```

1. Sub New_Workbook_with_Months()
2.
3. ' New_Workbook_with_Months Macro
4. ' Sample macro that creates a new workbook and enters a year's worth -
 ' of months into it. Recorded 12/2/05.
5.
6. Workbooks.Add
7. Range("A1").Select
8. ActiveCell.FormulaR1C1 = "Jan-2006"
9. Range("B1").Select
10. ActiveCell.FormulaR1C1 = "Feb-2006"
11. Range("A1:B1").Select
12. Selection.AutoFill Destination:=Range("A1:L1"), _
 Type:=xlFillDefault
13. Range("A2").Select
14. ActiveCell.FormulaR1C1 = "100"
15. Range("B2").Select
16. ActiveCell.FormulaR1C1 = "200"
17. Range("A2:B2").Select
18. Selection.AutoFill Destination:=Range("A2:L2"), _
 Type:=xlFillDefault
19. ActiveWorkbook.SaveAs Filename:=_
 "C:\Documents and Settings\Jack\My Documents\Sample Workbook.xls", _
 FileFormat:=xlNormal, Password:"", WriteResPassword:"", _
 ReadOnlyRecommended:=False, CreateBackup:=False
20. End Sub

```

因  
个  
个

现在单步执行该宏以监视有何事发生：它像先前那样，创建一个新工作簿并输入月份，但随后它输入数值 100 至 1200 到第二行各单元格。在结束处，它像先前那样，保存工作簿，并提示用户，原先的工作簿会被覆盖（除非已删除它）。

### 保存作品

当完成了针对这个宏的工作时，工作簿中已包含了该宏，也使工作簿有了改动，此时需从 Visual Basic 编辑器中选择“文件”>“保存”，以保存工作簿。然后，按下 Alt+Q 键，或选择“文件”>“关闭并返回到 Excel”，以关闭 Visual Basic 编辑器，并返回到 Excel。

### 编辑 PowerPoint 宏

在本节中，要编辑在第 1 章中录制的 PowerPoint 宏，从打开这个宏开始，步骤如下：

1. 打开包含该宏的 PowerPoint 演示文稿。
2. 选择“工具”>“宏”>“宏”以显示“宏”对话框。
3. 选择 Add\_Slide\_and\_Format\_Placeholder 宏。
4. 单击“编辑”按钮。

程序清单 3.6 显示出该宏，它与录制时几乎一样，只是使用了下划线延续符，把一些较长的代码行分成了若干较短的行。

### 程序清单 3.6

```

1. Sub Add_Slide_and_Format_Placeholder()
2.
3. ' Sample macro that adds a slide, formats its placeholder, and adds text -
 to it. Recorded 12/4/05 by Maria Kruger.
4.
5. ActiveWindow.View.GotoSlide Index:=1
6. ActivePresentation.Slides.Add(Index:=2, _
 Layout:=ppLayoutText).SlideIndex
7. ActiveWindow.Selection.SlideRange.Layout = ppLayoutTitle
8. ActiveWindow.Selection.SlideRange.Shapes("Rectangle 4").Select
9. With ActiveWindow.Selection.ShapeRange
10. .IncrementLeft -6#
11. .IncrementTop -125.75
12. End With
13. ActiveWindow.Selection.ShapeRange.ScaleHeight 1.56, msoFalse, _
 msoScaleFromTopLeft
14. ActiveWindow.Selection.ShapeRange.TextFrame.TextRange.Select
15. ActiveWindow.Selection.ShapeRange.TextFrame.TextRange.Characters _
 (Start:=1, Length:=0).Select
16. With ActiveWindow.Selection.TextRange
17. .Text = "The quick brown dog jumped over a lazy fox"
18. With .Font
19. .Name = "Arial"
20. .Size = 44
21. .Bold = msoFalse
22. .Italic = msoFalse
23. .Underline = msoFalse
24. .Shadow = msoFalse
25. .Emboss = msoFalse
26. .BaselineOffset = 0
27. .AutoRotateNumbers = msoFalse
28. .Color.SchemeColor = ppTitle
29. End With
30. End With
31. ActiveWindow.Selection.ShapeRange.TextFrame.TextRange.Characters _
 (Start:=1, Length:=42).Select
32. With ActiveWindow.Selection.TextRange.Font
33. .Name = "Impact"
34. .Size = 54
35. .Bold = msoFalse
36. .Italic = msoFalse
37. .Underline = msoFalse

```

```

38. .Shadow = msoFalse
39. .Emboss = msoFalse
40. .BaselineOffset = 0
41. .AutoRotateNumbers = msoFalse
42. .Color.SchemeColor = ppTitle
43. End With
44. End Sub

```

以下看看该宏内有何事发生：

- ◆ 行 1 是宏的开始，行 44 是宏的结束。
- ◆ 行 2 和行 4 是空白注释行，用来隔开宏说明，宏说明在行 3。
- ◆ 行 5 是向演示文稿添加幻灯片。这一语句稍显复杂，但对它不必担心过多，现在要注意两点：第一，该语句是对 Slides 集合对象使用 Add 方法，以便向该集合添加一张幻灯片（换而言之，是创建一张新的幻灯片），这和 Excel 宏使用 Add 方法向 Workbooks 集合添加一个工作簿相类似。第二，幻灯片的版式是 ppLayoutText，它是对应于文本幻灯片版式的 VBA 常量，PowerPoint 将它用于默认的新幻灯片。

**注意：**如果使用的是 PowerPoint 2000，那么，行 5 和行 6 将是一个语句，用来创建一张新幻灯片和指定该幻灯片版式。

- ◆ 行 6 是应用标题版式 (ppLayoutTitle)，录制宏时选择了它。（如果选择了一个不同的幻灯片版式，将会看到一个不同于 ppLayoutTitle 的常量。）
- ◆ 行 7 在活动幻灯片上的 Shapes 集合中，选择名称为 Rectangle 4 的形状。（此时，不要管怎样才能接触到活动幻灯片。）
- ◆ 行 8 至行 11 包含与已选择的形状 (ActiveWindow.Selection.ShapeRange) 一道工作的一个 With 语句。With 语句是一种简化的对象引用方式，With 语句和 End With 语句之间的一切，都引用 With 语句所叙述的对象。在本例中，行 9 使用带有负值的 IncrementLeft 方法，沿着水平方向左移形状；而行 10 则使用带有负值的 IncrementTop 方法，沿着垂直方向上移形状。

**提示：**With 语句有两条优点：简化了代码（因为不必在 With 与 End With 之间的每一行中说明对象）和使代码运行更快。

- ◆ 行 13 选择名称为 Rectangle 4 的形状，而行 14 选择该形状内 TextFrame 对象中的 Text Range 对象。当以交互方式工作时，PowerPoint 使这一选择过程成为无缝过程：在显示有图例“单击以添加标题”（或诸如此类）的形状中单击，PowerPoint 就选择该形状内文本框中的文本范围——但是，用户所看到的是，该形状内的文本成了被选文本。在 VBA 中，在接触到文本之前，必须在对象模型的几个看不见的层次中穿过。
- ◆ 当选择占位符文本时，PowerPoint 弃用它。当通过 VBA 选择占位符文本时也是如此。所以，行 15 是在文本范围内第一个字符起始处，找出一个新的选择内容。该选择内容的 Length 为 0，意味着该选择内容只是落到一个插入点，而不包含任何字符。
- ◆ 行 16 开始运行一个 With 语句，此语句持续到行 30。行 16 的 With ActiveWindow.Selection.TextRange 语句，让行 17 能够很简单地引用 ActiveWindow 对象内 Selection 对象

中 TextRange 对象的 Text 属性（代替 ActiveWindow.Selection.TextRange.Text），同时让行 18 很容易地引用 ActiveWindow 对象内 Selection 对象中 TextRange 对象的 Font 属性（代替 ActiveWindow.Selection.TextRange.Font）。

- ◆ 行 17 将 ActiveWindow.Selection.TextRange 对象的 Text 属性设定为键入的文本。
- ◆ 行 18 开始了一个嵌套的 With 语句，为对应于 TextRange 对象的 Font 对象设定属性。行 19 将 Font 对象的 Name 属性设定为 Arial；行 20 将 Font 对象的 Size 属性设定为 44；行 21 将 Font 对象的 Bold 属性设定为 msoFalse，即 Microsoft Office (mso) 常量为 False，等等。宏录制器创建所有这些独立的语句，来录制文本键入时的字体格式设置，它对该宏并不是必不可少的。行 29 结束嵌套的 With 语句。

**提示：**嵌套的 With 语句，是放在另一个 With 语句之内的语句，它用来说明在外层的 With 语句内说明过的对象之内的某一个对象。必要时可使用多重嵌套 With 语句。

- ◆ 行 31 使用 Select 方法，以选择文本范围内的字符 1 到字符 42。这是按下 Ctrl+Shift + Home 组合键的结果。因为这一语句是说明字符选择的，所以，如果更改了宏插入的文本，就必须更改这一语句。（如果在少于 42 个字符的文本范围上运行此语句，它将返回一个错误。如果在多于 42 个字符的文本范围上运行此语句，它将只选择用户想要的文本范围中的前 42 个字符。）
- ◆ 行 32 开始另一个 With 语句，它与 TextRange 对象的 Font 对象一道工作——但是，这个 With 语句录制用户在“字体”对话框中采取的行动，所以希望保持它。
- ◆ 行 43 结束此 With 语句，行 44 结束宏。

编辑这个宏，让它瘦身，并且更改一下它插入的文本：

1. 删除行 18 至行 29 之内的不必要的 With 语句。
2. 删除行 30。
3. 把行 16 和行 17 改成不带 With 的单个语句：

```
ActiveWindow.Selection.TextRange.Text = "The quick brown dog jumped over a lazy fox"
```

4. 现在，更改新的行 16 插入的文本。在双引号之间键入选择的文本。
5. 更改行 31，使用文本范围上的 Select 方法，不再说明字符选择。删除 Characters (Start: =1, Length: =42)，只留下如下语句：  

```
ActiveWindow.Selection.ShapeRange.TextFrame.TextRange.Select
```
6. 单击“标准”工具栏上的“保存”按钮，或选择“文件”>“保存”，以保存对演示文稿的改动。

现在得到了如程序清单 3.7 所示的代码。

**程序清单 3.7**

```
1. Sub Add_Slide_and_Format_Placeholder()
2. '
```

```

3. ' Sample macro that adds a slide, formats its placeholder, and adds text to it. -
 Recorded 12/4/05 by Maria Kruger.
4.
5. ActiveWindow.View.GotoSlide Index:=_
 ActivePresentation.Slides.Add(Index:=2, _
 Layout:=ppLayoutText).SlideIndex
6. ActiveWindow.Selection.SlideRange.Layout = ppLayoutTitle
7. ActiveWindow.Selection.SlideRange.Shapes("Rectangle 4").Select
8. With ActiveWindow.Selection.ShapeRange
9. .IncrementLeft -6#
10. .IncrementTop -125.75
11. End With
12. ActiveWindow.Selection.ShapeRange.ScaleHeight 1.56, msoFalse, _
 msoScaleFromTopLeft
13. ActiveWindow.Selection.SlideRange.Shapes("Rectangle 4").Select
14. ActiveWindow.Selection.ShapeRange.TextFrame.TextRange.Select
15. ActiveWindow.Selection.ShapeRange.TextFrame.TextRange.Characters _
 (Start:=1, Length:=0).Select
16. ActiveWindow.Selection.TextRange.Text = "Welcome to Acme Industries"
17. ActiveWindow.Selection.ShapeRange.TextFrame.TextRange.Select
18. With ActiveWindow.Selection.TextRange.Font
19. .Name = "Impact"
20. .Size = 54
21. .Bold = msoFalse
22. .Italic = msoFalse
23. .Underline = msoFalse
24. .Shadow = msoFalse
25. .Emboss = msoFalse
26. .BaselineOffset = 0
27. .AutoRotateNumbers = msoFalse
28. .Color.SchemeColor = ppTitle
29. End With
30. End Sub

```

现在可以单步执行这个修改后的宏，并确认它工作正常，符合用户的期望。

## 保存作品

当完成了针对这个宏的工作时，演示文稿已包含了该宏，也使演示文稿有了改动，此时，需从 Visual Basic 编辑器中选择“文件”>“保存”，以保存演示文稿。然后，按下 Alt + Q 键，或选择“文件”>“关闭并返回到 Microsoft PowerPoint”，以关闭 Visual Basic 编辑器，并返回到 PowerPoint。

### 什么时候应该使用宏录制器

至今为止，正如在本书中看到的那样，用户既可以使用宏录制器（在提供宏录制器的应用程序中），以交互工作方式，在应用程序中录制一系列行动来创建 VBA 代码；也可以在 Visual Basic 编辑器中，将 VBA 语句输进“代码”窗口来创建 VBA 代码。因此，用户可能会问：什么时候应该录制宏？又什么时候应该从头创建代码？编写一个过程比录制一个过程更困难但更先进——所以，当能够编写过程的时候，就不应该去录制过程吗？

实际上，使用宏录制器既有优点也有缺点。优点是：

- ◆任何时候宏录制器都能创建可用代码（只要能在适合的条件下运行宏）。
- ◆宏录制器使用起来迅速而容易。
- ◆宏录制器能够帮助你找出哪一个VBA对象、方法和属性，对应于应用程序的界面的哪一部分。

使用宏录制器的缺点是：

- ◆在宏录制器中创建的代码，可能含有一些不必要的语句，因为宏录制器录制用户在应用程序中所做的一切——包括录制宏时使用的每个内置对话框里的所有选项。例如，如果从Word启动宏录制器，选择“工具”>“选项”，以显示“选项”对话框的“视图”页，单击“编辑”选项卡以显示“编辑”页，以及更改“自动键盘切换”设置，此时，宏录制器将录制“编辑”页上的所有设置，以及“视图”页上的所有设置。结果是产生出约40行不必要的代码。（如果在访问“编辑”页之前又访问了“选项”对话框中的其他页的话，宏录制器也要录制这些页面中的所有设置。）而在Visual Basic编辑器中手工创建代码时，能够通过使用一个语句来达到与其相同的效果。
- ◆由宏录制器创建的代码，只能在活动文档中工作，不能使用于其他文档，这是因为，以交互方式与之工作的任何文档，都只能成为活动文档。在本书后面，会学习到如何使用应用程序的对象模型中的对象，来与同活动文档不一样的其他文档一道工作。与其他文档一道工作是会有好处的，例如，能够隐藏正实施的操作而不为用户所知，还能够使代码运行得更快。
- ◆宏录制器只能针对在主应用程序中实施的某些行动创建VBA代码。例如，如果想在某一过程中显示一个对话框或用户窗体，必须手工编写适当的语句——不能够录制它。宏录制器中可用的VBA行动的子集，同以交互方式工作时在主应用程序中可采取的行动集相类似，所以，可以用它做很多事。但是，同在VBA中能够实施的行动的整个范围相比较，它仍然是很有限的。

即使是VBA专家，也应承认宏录制器能够生成虽粗糙但可用的宏，或生成更复杂过程的基本部分，它不失为一个有用的工具。使用者常常会发现，让宏录制器尽量分担创建过程的紧张是可行的。如果使用宏录制器能够迅速地找出所需要的VBA对象或属性，从而节省时间，那就这么干吧。

## 第 4 章 在 Visual Basic 编辑器中从头生成代码

- ◆ 设置 Visual Basic 编辑器以创建过程
- ◆ 为 Word 创建一个过程
- ◆ 为 Excel 创建一个过程
- ◆ 为 PowerPoint 创建一个过程

在本章中，将在 Visual Basic 编辑器中进行实践，学习如何从头开始创建过程。本章假设用户已经有了按默认配置方式安排的 Visual Basic 编辑器，而且（为了更好地实践）已经将其设置为要求显式声明变量，所以，本章假设已经将编辑器设置为这样了，然后通过几个例子来学习如何在 Word、Excel 和 PowerPoint 中创建过程。

本章的目的是在学习语言的详细内容之前，给读者一个在 Visual Basic 编辑器中创建代码的感觉。在本章中，很快就会接触到 VBA 的各种成分（如对象，属性，方法，变量，以及常量等），在本书后面，将对它们进行更全面的学习。Visual Basic 编辑器提供有多种辅助功能形式，在学习进程中，会遇到其中的几种，包括宏录制器、对象浏览器和“帮助”系统。以后也将在本书中研究这些辅助功能。

### 设置 Visual Basic 编辑器以创建过程

如果让 Visual Basic 编辑器以默认配置方式设置（就像第一次从 VBA 宿主中显示 Visual Basic 编辑器时会看到的那样），会发现遵循下述过程中的指令是很容易的。为了更好地实践，也应该要求显式声明变量。下面各步骤叙述如何设置 Visual Basic 编辑器。过程中的每一步都会说明，何时确认 Visual Basic 编辑器已经正确设置。

1. 如果“工程资源管理器”没有显示，选择“视图”>“工程资源管理器”，或按下 Ctrl+R 键以显示它。
2. 如果“属性”窗口没有显示，选择“视图”>“属性窗口”，或按下 F4 键以显示它。
3. 除非另有偏好，否则，应将“工程资源管理器”放在它的合适位置上，即 Visual Basic 编辑器主区域的左上角。将“属性”窗口放在“工程资源管理器”下面，这同样也是它的默认位置。（要更改连接，需选择“工具”>“选项”，单击“可连接”标签，然后在“选项”对话框的“可连接”页面上工作。）
4. 关闭已打开的任何“代码”窗口或用户窗体窗口。
5. 将 Visual Basic 编辑器设置成要求显式声明变量，这样，在使用每个变量之前，都必须正式声明这个变量。选择“工具”>“选项”以显示“选项”对话框，在“编辑器”页上选择“要求变量声明”复选框，然后单击“确定”按钮。这一设置，使得 Visual Basic 编辑器自动地对今后创建的所有模块和用户窗体输入 Option Explicit 语句。

**注意：**如果有可供使用的所有三种应用程序，最好在它们上面都实践一下上述这些过程。

## 为Word创建一个过程

要为Word创建这个过程，可引出“修订”特性，以便在“删除线”和“隐藏”之间切换显示被删除文本的方式。可以在下述两种方式之间即时切换：让被删除文本留在屏幕上，但带有一条贯通被删除文本的删除线；或是让它直接消失。

遵循如下步骤来创建该过程：

1. 启动Word。如果Word已经在运行，先退出再重新启动它。
2. 录制一个宏，以便接触到所需要的对象、属性和设置。遵循如下步骤：
  - A. 选择“工具”>“宏”>“录制新宏”，以显示“录制宏”对话框。
  - B. 接受宏录制器自动指定的宏名(Macro1, Macro2, 等等)，或者给它取一个临时性的名字，这个名字将提醒用户要删除这个宏，如果忘了这样做的话。本例使用的名称是Macro\_to\_Delete\_1。
  - C. 让“将宏保存在”下拉列表的设置停留在“所有文档(Normal.dot)”。如果觉得在本节结束时不会忘记删除这个宏，可将说明部分丢下不管。
  - D. 单击“确定”按钮以启动录制宏。
  - E. 选择“工具”>“选项”，以显示“选项”对话框。按默认方式，Word首先显示“视图”页。单击“修订”选项卡以显示“修订”页(如图4.1所示)，在“删除内容”下拉列表中选择“删除线”，然后单击“确定”按钮以关闭“选项”对话框。
  - F. 选择“工具”>“选项”，以再次显示“选项”对话框。这一次，“修订”页应该处在页面的顶部，现在，在“删除内容”下拉列表中选择“隐藏”，并再次单击“确定”按钮，以关闭“选项”对话框。
  - G. 双击状态栏上的REC指示符，或单击“停止录制”工具栏上的“停止录制”按钮，或选择“工具”>“宏”>“停止录制”，以使录制宏停止。
3. 选择“工具”>“宏”>“宏”，以显示“宏”对话框。选中刚才录制的宏，再单击“编辑”按钮打开该宏，以便在Visual Basic编辑器中进行编辑。所得代码应如下所示(用户名称和日期除外)：

```

1. Sub Macro_to_Delete_1()
2.
3. ' Macro_to_Delete_1 Macro
4. ' Macro recorded 1/7/2006 by Jack Ishida
5.
6. Application.DisplayStatusBar = True
7. Application>ShowWindowsInTaskbar = True
8. Application>ShowStartupDialog = True
9. With ActiveWindow

```



图4.1 Word中“选项”对

话框的“修订”页

```

10. .DisplayHorizontalScrollBar = True
11. .DisplayVerticalScrollBar = True
12. .DisplayLeftScrollBar = False
13. .StyleAreaWidth = InchesToPoints(0)
14. .DisplayRightRuler = False
15. .DisplayScreenTips = True
16. With .View
17. .ShowAnimation = True
18. .Draft = False
19. .WrapToWindow = False
20. .ShowPicturePlaceHolders = False
21. .ShowFieldCodes = False
22. .ShowBookmarks = False
23. .FieldShading = wdFieldShadingWhenSelected
24. .ShowTabs = False
25. .ShowSpaces = False
26. .ShowParagraphs = False
27. .ShowHyphens = False
28. .ShowHiddenText = False
29. .ShowAll = False
30. .ShowDrawings = True
31. .ShowObjectAnchors = False
32. .ShowTextBoundaries = False
33. .ShowHighlight = True
34. .DisplayPageBoundaries = True
35. .DisplaySmartTags = True
36. End With
37. End With
38. With Options
39. .InsertedTextMark = wdInsertedTextMarkUnderline
40. .InsertedTextColor = wdByAuthor
41. .DeletedTextMark = wdDeletedTextMarkStrikeThrough
42. .DeletedTextColor = wdByAuthor
43. .RevisedPropertiesMark = wdRevisedPropertiesMarkNone
44. .RevisedPropertiesColor = wdByAuthor
45. .RevisedLinesMark = wdRevisedLinesMarkOutsideBorder
46. .RevisedLinesColor = wdAuto
47. .CommentsColor = wdByAuthor
48. .RevisionsBalloonPrintOrientation = _
 wdBalloonPrintOrientationPreserve
49. End With
50. With ActiveWindow.View
51. .RevisionsMode = wdBalloonRevisions
52. .RevisionsBalloonShowConnectingLines = True
53. .RevisionsBalloonSide = wdRightMargin
54. .RevisionsBalloonWidthType = wdBalloonWidthPoints
55. .RevisionsBalloonWidth = InchesToPoints(2.5)
56. End With
57. With Options
58. .InsertedTextMark = wdInsertedTextMarkUnderline
59. .InsertedTextColor = wdByAuthor
60. .DeletedTextMark = wdDeletedTextMarkHidden
61. .DeletedTextColor = wdByAuthor
62. .RevisedPropertiesMark = wdRevisedPropertiesMarkNone
63. .RevisedPropertiesColor = wdByAuthor
64. .RevisedLinesMark = wdRevisedLinesMarkOutsideBorder

```

```

65. .RevisedLinesColor = wdAuto
66. .CommentsColor = wdByAuthor
67. .RevisionsBalloonPrintOrientation =
68. wdBalloonPrintOrientationPreserve
69. End With
70. With ActiveWindow.View
71. .RevisionsMode = wdBalloonRevisions
72. .RevisionsBalloonShowConnectingLines = True
73. .RevisionsBalloonSide = wdRightMargin
74. .RevisionsBalloonWidthType = wdBalloonWidthPoints
75. .RevisionsBalloonWidth = InchesToPoints(2.5)
76. End Sub

```

4. 针对上述行动产生的代码量大得惊人。这是因为，宏录制器录制了所访问的“选项”对话框页面上所有选项的设置。简单看一下与很多设置对应的代码：

◆ 行 6 至行 37 录制“视图”页上的设置

(如图 4.2 所示)。因为没有改变这些设置中的任何一项，所以它们应与该宏无关，但是，如果看看图 4.2，就能看到代码是如何反映这些设置的。

例如，Application.DisplayStatusBar = True 语句意味着“状态栏”复选框被选中；Application.ShowWindowsInTaskbar = True 语句表示“任务栏中的窗口”复选框被选中；而 Application.ShowStartupDialog = True 语句表示“启动任务窗格”复选框被选中。(被选中的复选框为 True，被清除的复选框为 False。)

◆ 行 38 至行 49 录制第一次访问时“选

项”对话框的“修订”页上的设置。这些语句可分为两个片段：第一个片段是对应于该页的标记区和打印区（包括批注框）的设置（它们出现在 With Options 结构之内）；而第二个片段是对应于批注框区的设置（它们出现在 With ActiveWindow.View 结构之内）。

◆ 行 50 至行 75 录制第二次访问时“选项”对话框的“修订”页上的设置。

5. 要分辨清所需要的命令——对于 Options 对象的 DeletedTextMark 属性——和与它对应的所需要的值：wdDeletedTextMarkStrikeThrough（将“删除内容”下拉列表设置为“删除线”时）和 wdDeletedTextMarkHidden（将“删除内容”下拉列表设置为“隐藏”时）。

6. 选择整个已录制的宏，从 Sub 语句直到 End Sub 语句，再按下“删除”键，以将宏删去。

7. 确认 Visual Basic 编辑器已经按照本章前面“设置 Visual Basic 编辑器以创建过程”一节的叙述进行了设置。



图 4.2 Word 中“选项”对话框的“视图”页

8. 在“工程资源管理器”窗口内，右击 Normal 项目中的任意处。并从上下文菜单中选择“插入”>“模块”。Visual Basic 编辑器将一个新模块插入进 Normal.dot 全局模板，为它显示一个“代码”窗口，如果 Normal 树已折叠，则展开 Normal 树。
9. 按下 F4 键，以便为新模块激活“属性”窗口。Visual Basic 编辑器选择（名称）属性，这是唯一的可用属性（属性的名称包括括号）。
10. 在“属性”窗口中为新模块键入名称。本例中使用的名称是 Procedures \_ to \_ Keep \_ 1。
11. 按下 F7 键或在“代码”窗口内单击以激活它。
12. 确认 Visual Basic 编辑器已经在“代码”窗口内代码表（代码区）顶部的声明区里输入了 Option Explicit 语句。如果没有，现在可将该语句键入。
13. 在 Option Explicit 语句下面，为过程键入 Sub 语句并按下 Enter 键。将过程命名为 Toggle \_ Track \_ Changes \_ between \_ Hidden \_ and \_ Strikethrough:

```
Sub Toggle_Track_Changes_between_Hidden_and_Strikethrough
```

14. 当按下 Enter 键时，Visual Basic 编辑器输入一个括号（在 Sub 语句结束处），一个空白行，以及 End Sub 语句，并将插入点定位于空白行。

```
Sub Toggle_Track_Changes_between_Hidden_and_Strikethrough()
```

```
End Sub
```

15. 按下 Tab 键以使 Sub 语句下面第一行缩进。
16. 键入 if options.（用小写，包括句号）以显示“属性/方法列表”下拉列表。
17. 键入 d、e 和 l 并使用↓键，或以↓键或鼠标做简单的滚动，以选择 DeletedTextMark 条目。
18. 键入 =Visual Basic 编辑器输入 DeletedTextMark 关键字，后跟等号，然后显示对应于 DeletedTextMark 属性的“属性/方法列表”常量表（见图 4.3）。

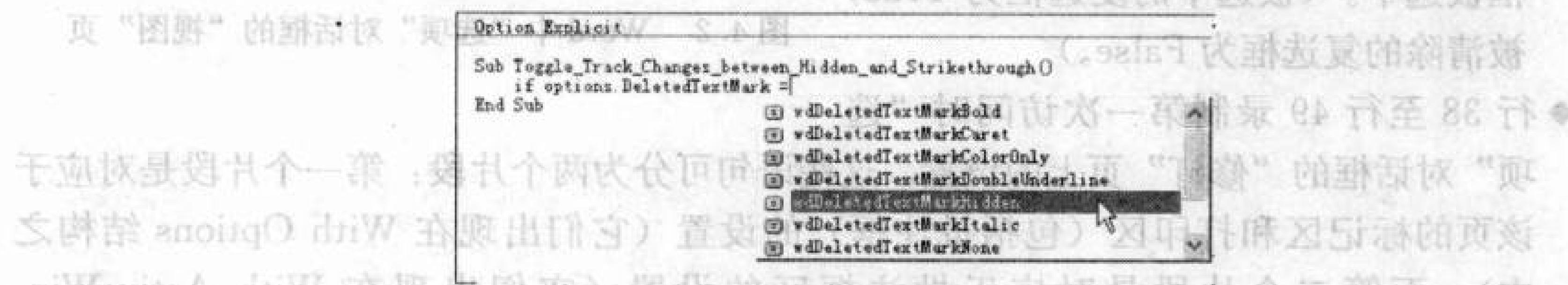


图 4.3 Visual Basic 编辑器的“属性/方法列表”常量表，

提供了对应于 DeletedTextMark 属性的可用常量

19. 选择 wdDeletedTextMarkHidden 项并输入它，方法是按下 Tab 键或双击该项。
20. 键入 Then 并按下 Enter 键。注意：当开始下一行代码时（通过按下 Enter 键），Visual Basic 编辑器为本代码行检查错误。如果对语句的 If options 部分使用了小写，Visual Basic 编辑器会使首字母变为正确的大写。如果等号的任一边不是空格，Visual Basic 编辑器都会给加上。
21. 使用 Visual Basic 编辑器特性提供的辅助方式，输入 Options. DeletedTextMark = wdDeletedTextMarkStrikethrough，然后按下 Enter 键。

22. 按下 Backspace 键或 Shift+Tab 键，以使新代码行除去缩进一个制表符的距离。  
 23. 键入 ElseIf 关键字，然后输入过程的其余部分，如下所示：

```
ElseIf Options.DeletedTextMark = wdDeletedTextMarkStrikeThrough Then
 Options.DeletedTextMark = wdDeletedTextMarkHidden
End If
```

24. 确认过程与下面所示的一致：

```
Sub Toggle_Track_Changes_between_Hidden_and_Strikethrough()
 If Options.DeletedTextMark = wdDeletedTextMarkHidden Then
 Options.DeletedTextMark = wdDeletedTextMarkStrikeThrough
 ElseIf Options.DeletedTextMark = wdDeletedTextMarkStrikeThrough Then
 Options.DeletedTextMark = wdDeletedTextMarkHidden
 End If
End Sub
```

25. 按下 Alt+F11 键以切换到 Word，然后创建一个短文档，该文档使用“修订”标记，包括被删除文本。  
 26. 将 Word 窗口和 Visual Basic 编辑器窗口安排成并排显示。在 Word 中，选择“工具”>“选项”，检查“选项”对话框的“修订”选项卡上“删除内容”下拉列表的当前设置，然后单击“确定”按钮。在 Visual Basic 编辑器中，按下 F5 键或单击“标准”工具栏或“调试”工具栏上的“运行子过程/用户窗体”，以运行这个宏。返回到 Word，选择“工具”>“选项”，证实“删除内容”设置已经更改。  
 27. 单击 Visual Basic 编辑器中“标准”工具栏上的“保存”按钮。

**注意：**也可以像使用 Options 对象的 With 语句那样设置这个过程，使它看起来像如下样子：

```
Sub Toggle_Track_Changes_between_Hidden_and_Strikethrough_2()
 With Options
 If .DeletedTextMark = wdDeletedTextMarkHidden Then
 .DeletedTextMark = wdDeletedTextMarkStrikeThrough
 ElseIf .DeletedTextMark = wdDeletedTextMarkStrikeThrough Then
 .DeletedTextMark = wdDeletedTextMarkHidden
 End If
 End With
End Sub
```

## 为 Excel 创建一个过程

将要为 Excel 创建的过程虽短但很有用：当用户运行 Excel 时，这个过程使 Excel 窗口最大化，并打开在其上工作过的最后一个文件。该过程提供了一个与 Excel 中各事件一道工作，及使用对象浏览器以寻找所需要的对象、方法和属性的例子。

遵循如下步骤来创建该过程：

1. 启动 Excel，如果它没有运行的话。
2. 如果个人宏工作簿当前是隐藏着的话，选择“窗口”>“取消隐藏”，以显示“取消隐藏”对话框，在“取消隐藏工作簿”列表框内选择 PERSONAL.XLS，然后单击“确定”按钮。

3. 按下 Alt+F11 键，或选择“工具”>“宏”>“Visual Basic 编辑器”，以打开 Visual Basic 编辑器。
4. 确认 Visual Basic 编辑器已经按照本章前面“设置 Visual Basic 编辑器以创建过程”一节的叙述进行了设置。
5. 在“工程资源管理器”窗口内，展开 VBAProject (PERSONAL.XLS)，如果它是折叠的话，双击它的名称或单击名称左边的+号。
6. 展开 Microsoft Excel Objects 文件夹。
7. 双击 ThisWorkbook 项以打开代码窗口内的代码表。ThisWorkbook 对象表示工作簿。
8. 确认 Visual Basic 编辑器已经在代码表顶部的声明区里输入 Option Explicit 语句。如果没有，现在可将该语句键入。
9. 在“代码”窗口左上角的“对象”下拉列表中，选择 Workbook。Visual Basic 编辑器为这个工作簿对象创建一个 Open 事件的程序的头和尾，并将插入点置于空白行：

```
Private Sub Workbook_Open()
End Sub
```

**注意：** Private 关键字限制该宏的范围——即它可操作的区域。它使得该宏为包含它的模块中的所有过程可用，但其他模块中的过程则不可用。第 6 章将对此做更详细的说明。

10. 按下 F2 键，或选择“视图”>“对象浏览器”，或单击“标准”工具栏上的“对象浏览器”按钮，以显示对象浏览器窗口（如图 4.4 所示）。



图 4.4 使用对象浏览器以寻找所需要的对应于某一过程的对象、方法和属性

11. 要在该宏中采取的第一个操作，是最大化应用窗口。如在任何应用程序中一样，VBA 使用 Application 对象来表示 Excel 应用，但是需要找出针对其进行工作的属性。在“工程/库”下拉列表中选择 Excel，在“搜索”文本框内键入 maximize，然后单击“搜索”按钮或按下 Enter 键。对象浏览器在“搜索结果”窗格内（在图 4.4 中该窗格是折叠的，没有显示出来）显示搜索的结果（如图 4.5 所示）：常量 xlMaximized 是类 xlWindowState 的一个成员。



图 4.5 对象浏览器中的与“maximize”对应的搜索结果

12. 按下 F7 键以激活“代码”窗口。（也可以单击“代码窗口”，或选择“视图”>“代码窗口”，或从“窗口”菜单中选择“代码窗口”。）
13. 键入 application.（用小写，包括句号）以使 Visual Basic 编辑器显示下拉列表，键入 w 以跳到以 W 开头的项目，然后选择WindowState 项。
14. 键入 = 以输入WindowState 至代码中，并显示WindowState 可用的常量的列表（如图 4.6 所示）。
15. 选择 xlMaximized 项并按下 Enter 键，以开始一个新语句。
16. 针对该宏采取的第二个操作，是打开用过的最后一个文件——“最近使用过的文件”的列表上的文件 1（选中 Excel 的“选项”对话框“常规”页上的“最近使用过的文件列表”复选框，这个列表就会出现在“文件”菜单底部）。按下 F2 键，以再次激活对象浏览器。
17. 在“工程/库”下拉列表中仍选择 Excel，键入 recent，按下 Enter 键，或单击“搜索”按钮。对象浏览器显示搜索的结果（如图 4.7 所示）。所需要的项目是 Application 对象的 RecentFiles 属性。RecentFiles 属性返回 RecentFiles 集合，这是一个包含“最近使用过的文件列表”中各文件详细情况的对象。
18. 按下 F7 键以激活“代码”窗口。键入 application，并从“属性/方法列表”下拉列表中选择 RecentFiles。然后键入 (1)，以指明是 RecentFiles 集合中的第一项，再从“属性/方法列表”中选择 Open 方法：

```
Application.RecentFiles(1).Open
```

19. 这样一来得到的过程应该如下所示：

```
Private Sub Workbook_Open()
 Application.WindowState = xlMaximized
 Application.RecentFiles(1).Open
End Sub
```

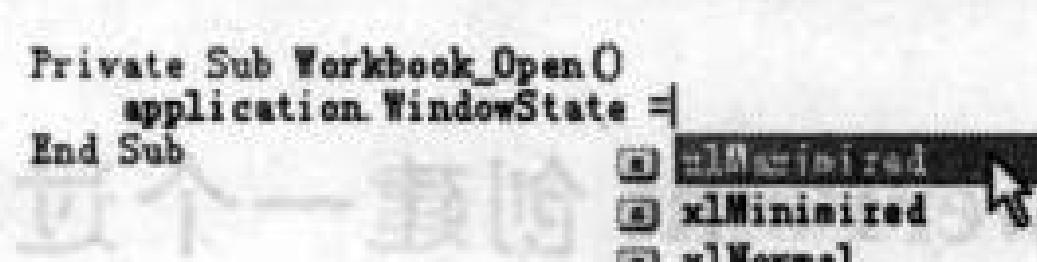


图 4.6 使用常量列表以便迅速而容易地输入常量

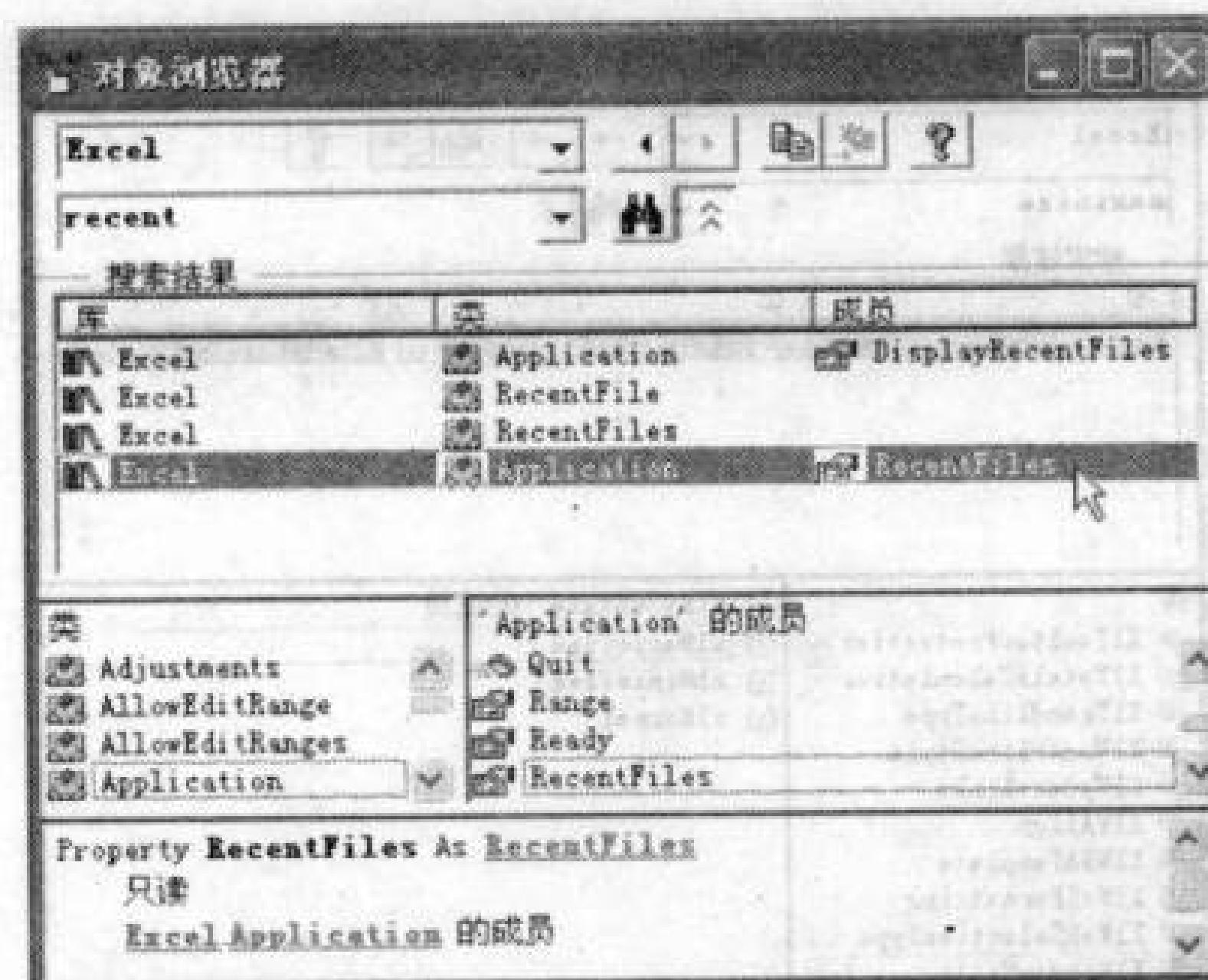


图 4.7 对象浏览器中对于“recent”的搜索结果

20. 按下 Alt+Q 键，或选择“文件”>“关闭并返回到 Microsoft Excel”，以返回到 Excel。
21. 选择“文件”>“保存”，以保存个人宏工作簿。
22. 选择“窗口”>“隐藏”，以隐藏个人宏工作簿。
23. 打开一个样本文档，对它做一个更改，保存并关闭它。
24. 选择“文件”>“退出”，以退出 Excel。
25. 重新启动 Excel。Excel 使应用窗口最大化，并打开最近使用过的文件。

## 为 PowerPoint 创建一个过程

将要为 PowerPoint 创建的这个过程既短又简单，但它能够节省用户的精力，从长远来看很有价值。它把一张标题幻灯片添加到活动的演示文稿上，即插入起封面作用的标题幻灯片，上面含有当前日期和作为演示者的公司的名称。

遵循如下步骤创建这个过程：

1. 启动 PowerPoint。如果 PowerPoint 已经在运行，先关闭它，再重新启动它。如果 PowerPoint 在启动时创建了一个默认的演示文稿，则关闭这个演示文稿。
2. 以所选择的模板为基础，创建一个新的演示文稿。确认演示文稿上的默认幻灯片具有标题幻灯片版式。如果不是的话，选择“格式”>“幻灯片版式”，然后单击“标题幻灯片版式”，以将其应用于默认幻灯片。
3. 选择“工具”>“宏”>“Visual Basic 编辑器”，或按下 Alt+F11 键，以打开 Visual Basic 编辑器。
4. 确认 Visual Basic 编辑器已经按照本章前面“设置 Visual Basic 编辑器以创建过程”一节的叙述进行了设置。
5. 在“工程资源管理器”窗口内，右击 VBAProject（演示文稿 1）项内的任意地方，并从上下文菜单中选择“插入”>“模块”。Visual Basic 编辑器在工程中插入一个新模块，显示包含对应于该模块的代码表的代码窗口，并扩展工程树。
6. 确认 Visual Basic 编辑器已经在代码表顶部的声明区里输入了 Option Explicit 语句。如果没有，现在将该语句键入。

7. 按下 F4 键，以激活与新模块对应的“属性”窗口。也可以在“属性”窗口内单击。
8. 在“属性”窗口中键入新模块的名称：General\_Procedures。
9. 按下 F7 键，或在“代码”窗口中单击以激活它。
10. 在 Option Explicit 语句下面，键入对应于该过程的 Sub 语句，并按下 Enter 键：

```
Sub Add_Title_Slide
```

11. Visual Basic 编辑器在 Sub 语句结束处输入括号，再输入一个空白行和 End Sub 语句，并把插入点放在空白行。

```
Sub Sub Add_Title_Slide()
```

```
End Sub
```

12. 按下 Tab 键，使 Sub 语句下面的第一行缩进。
13. 现在，使用帮助系统来识别所需要的对象。用户将与以 ActivePresentation 对象来表示的活动演示文稿一道工作，所以，键入 activepresentation，把插入点定位于该字中的任意地方，再按下 F1 键，以打开“Microsoft Visual Basic 帮助”。它将展示如图 4.8 所示的 ActivePresentation 属性的屏幕帮助画面。

**注意：**如果按下 F1 键，并且插入点位于 activepresentation 这个字之中时不能工作的话，可选择“帮助”>“Microsoft Visual Basic 帮助”，以显示“Visual Basic 帮助任务”窗格。在“搜索”框内键入 activepresentation，并按下 Enter 键，或单击“开始搜索”按钮。在“搜索结果”窗口内，单击“ActivePresentation 属性”链接。

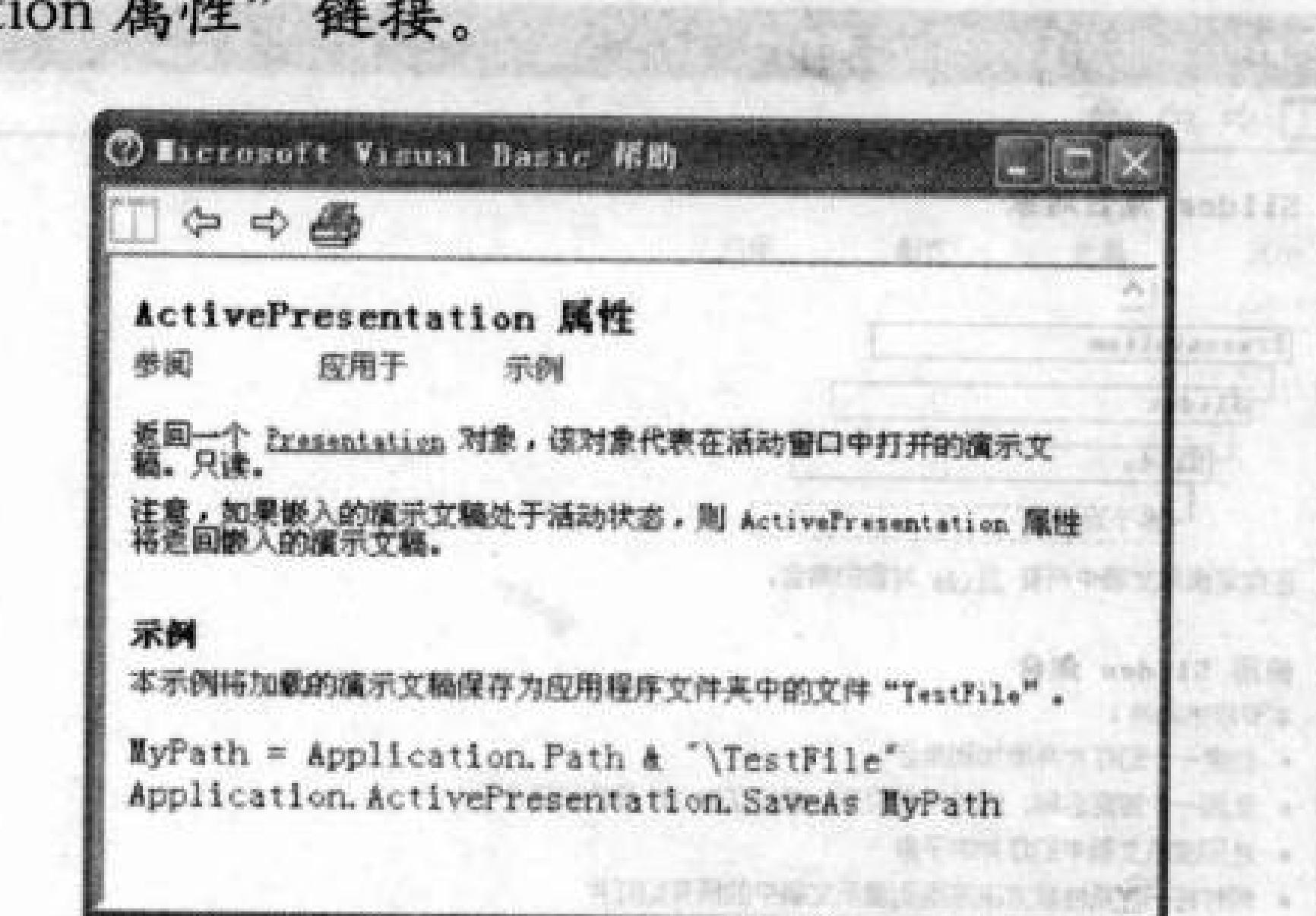


图 4.8 “ActivePresentation 属性”

的屏幕帮助画面

14. 单击“ActivePresentation 属性”屏幕帮助画面上的 Presentation 对象处，以显示 Presentation 的屏幕“帮助”画面。
15. 单击位于 Presentation 对象帮助画面上方的对象层次图中的“多个对象”框，然后单击弹出式菜单中的 Slides 集合对象（如图 4.9 所示），以显示 Slides 集合对象帮助画面（如图 4.10 所示）。
16. 从这一屏幕画面中，可取得两部分信息：第一，一张幻灯片，是用一个 Slide 对象来表示的（它被组织在 Slides 集合之中）；第二，要使用 Add 方法来创建一张新幻

。击单击幻灯片。“幻灯片”或幻灯片窗格“幻灯片”窗格右侧的“幻灯片”按钮不对。

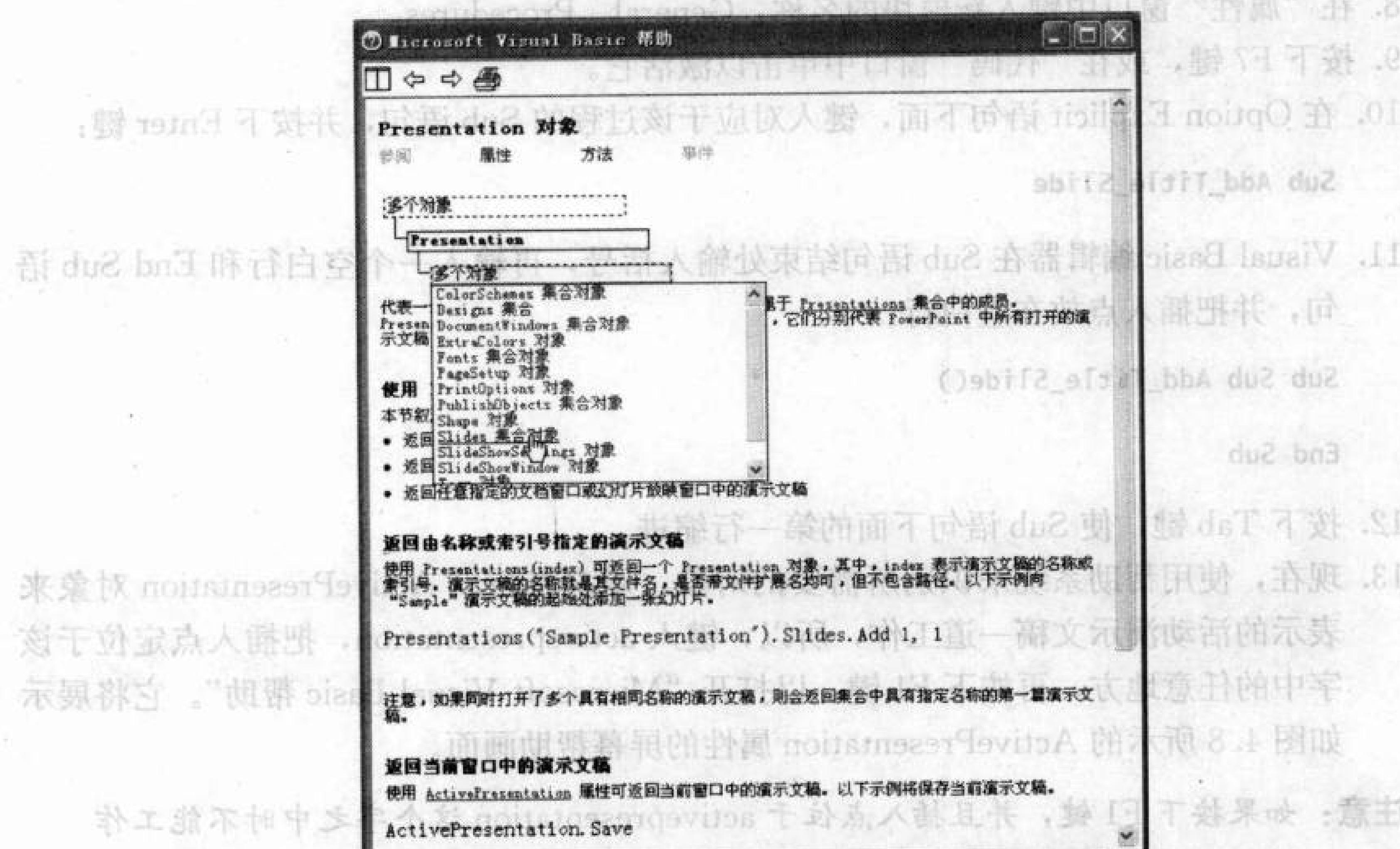


图 4.9 单击 Presentation 对象屏幕帮助画面中的“多个对象”框，并从弹出式菜单中选择 Slides 集合对象

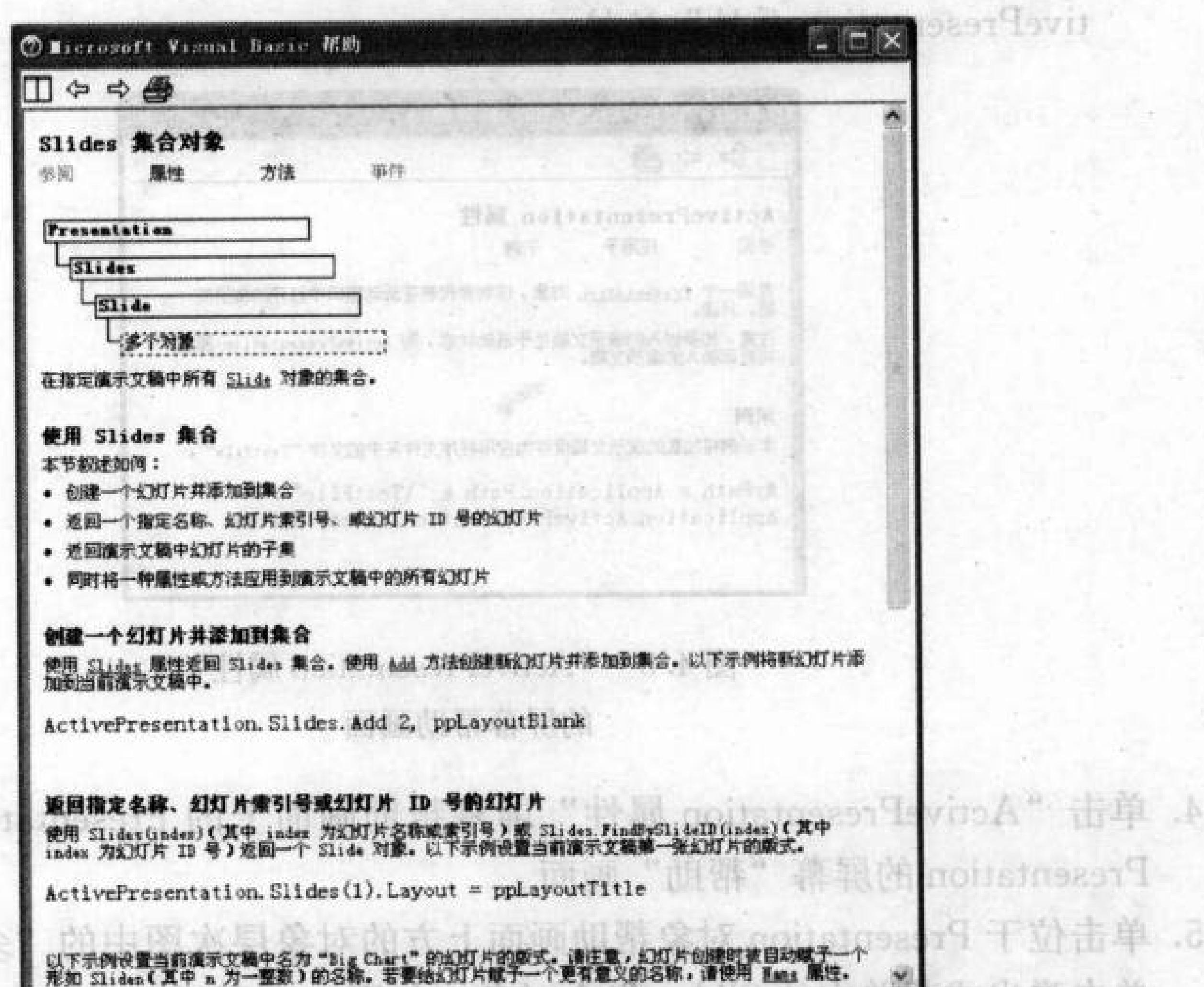


图 4.10 Slides 集合对象的屏幕帮助显示画面

17. 返回到 Visual Basic 编辑器，并删除添加的 activepresentation。
18. 键入针对 Slide 对象类型的对象变量的声明，以表示该过程所创建的幻灯片。注意：

当键入了 as 和一个空格之后，Visual Basic 编辑器会显示出可用关键字列表。键入 s 和 l 后选中 Slide，然后按 Enter 键，以完成该术语，并开始一个新的代码行：

```
Dim sldTitleSlide As Slide
```

19. 然后，使用一个 Set 语句，将用 Add 方法创建的新幻灯片指定给 sldTitleSlide 对象。键入 set sld 然后按下 Ctrl+空格键，使用字特性帮助输入 sldTitleSlide。然后再使用 Visual Basic 编辑器的辅助功能，键入 ActivePresentation. Slides. Add ()，这样，这一行显示如下：

```
Set sldTitleSlide = ActivePresentation.Slides.Add()
```

20. 键入括号时，“自动显示快速信息”特性显示出对应于 Add 方法的语法，如图 4.11 所示。

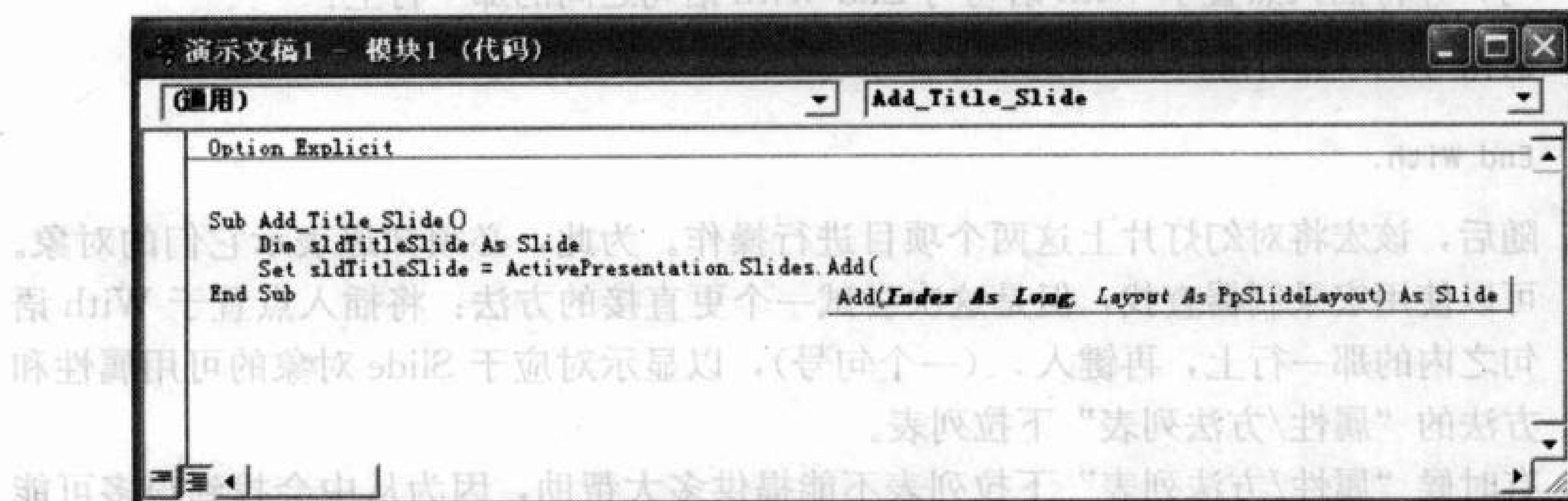


图 4.11 键入 Add 之后，“自动显示快速信息”特性显示出对应于 Add 方法的语法

21. 键入 Index 参数，一个冒号，一个等号，数值 1（因为标题幻灯片是演示文稿中第一张幻灯片），以及一个逗号：

```
Set sldTitleSlide = ActivePresentation.Slides.Add(Index:=1,
```

**注意：**当某一方法要使用参数时，如这里 Add 方法所做的那样，可以在两种方式中选用：指定参数的名称；或者略去名称，让 VBA 从值或常量的次序中推断出参数。例如，在本例中，既可以指定 Add (Index: = 1, Layout: = ppLayoutTitle)，也可以指定 Add (1, ppLayoutTitle)。后者较简洁且易于输入，但前者读起来更清楚。

22. 使用行延续符（一个空格，后随一条下划线），将该语句拆成两行。然后按下 Tab 键使新行缩进，键入 Layout 参数，一个冒号，一个等号，再从“属性/方法列表”下拉列表中选出 ppLayoutTitle 常量，如图 4.12 所示。

23. 键入括号以结束该语句：

```
Set sldTitleSlide = ActivePresentation.Slides.Add(Index:=1, _
Layout:=ppLayoutTitle)
```

24. 按下 Enter 键以开始新行，按下 Backspace 键或 Shift+Tab 键，以使新行除去缩进一个制表符的距离。
25. 此后，用户将要与 sldTitleSlide 一道工作，所以，需要使用它来创建一个 With 语