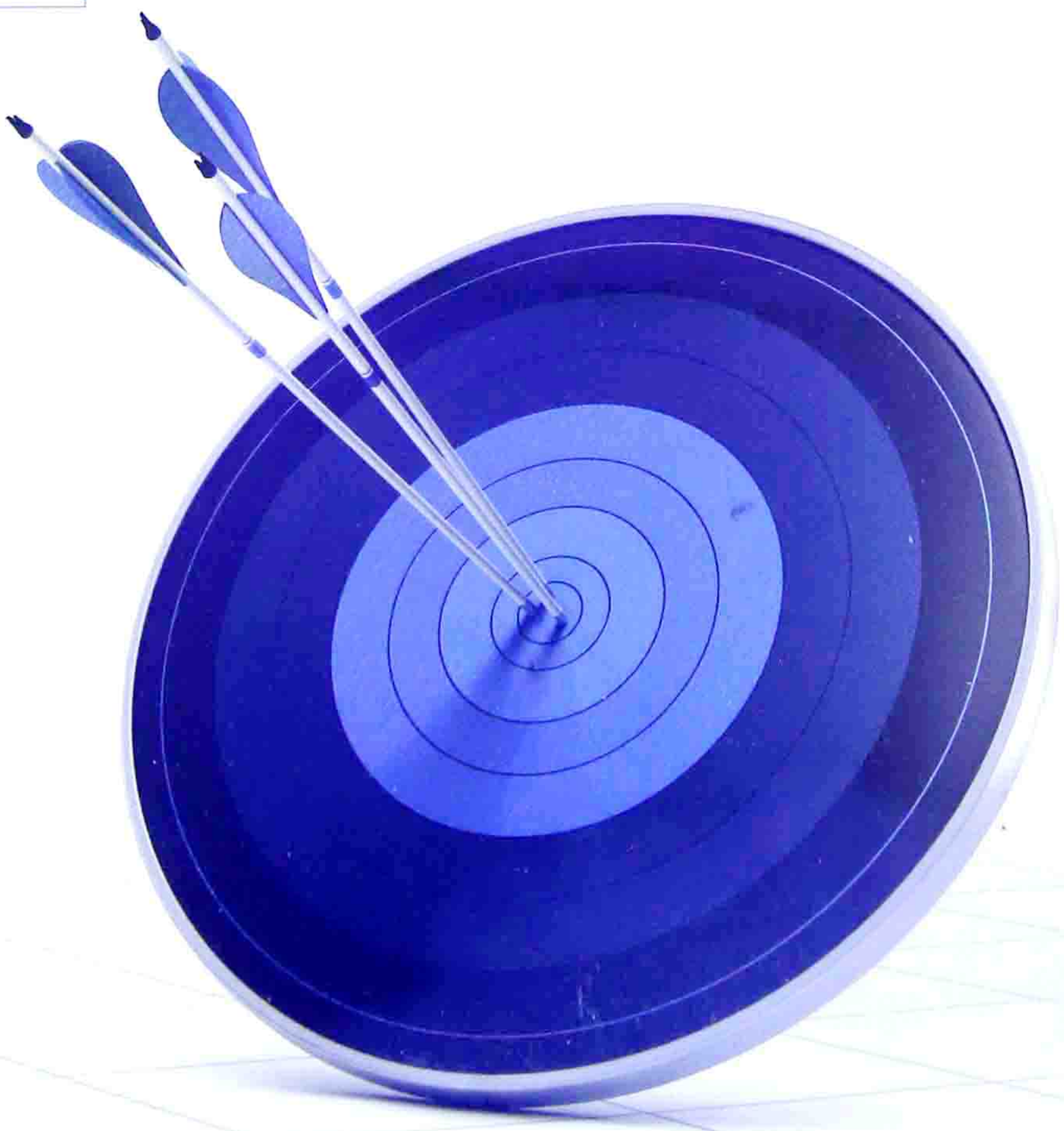
Broadview®
www.broadview.com.cn

VBA入门经典之作 畅销书升级版

Excel VBA 程序开发自学宝典 第3版

罗刚君 著

- Excel百宝箱安装程序
- Excel百宝箱动画教学
- VBA入门视频教材
- 全书案例文件及源代码



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

Excel VBA
程序开发自学宝典 第3版

写作特点

本书是Excel VBA初学者的基础教材，全书分上下两篇，上篇讲述了VBA语言的基础语法与常见对象的综合应用，下篇讲述数组、正则表达式、自定义函数、设计窗体、FSO、类模块知识、注册表、功能区设计、插件设计和封装代码等高级知识。

通读本书，您可以利用VBA解决工作疑难，让以往可能需要一小时的工作量在1到2秒钟内完成；您也可以借助本书的知识点编写出自己的商业插件。大型Excel插件《E灵》(早期名称为《Excel百宝箱》)即为Excel VBA开发的商业插件，至今拥有数十万名用户。

本书基于Excel 2010写作，但是代码通用于Excel 2007、2010和2013。由于Excel 2003即将被淘汰，因此本书不再讲述传统菜单的设计方法，而是重点讲述功能区的开发思路，并提供若干功能区模板，从而让读者可以快速设计功能区组件。

本书比较注重代码的通用性和效率，总结了多条优化代码的规则。

本书的随书光盘中提供了书中的一切案例源代码，并对每一句代码提供含义注释，便于读者快速理解过程的含义与设计思路。

售后服务

为了帮助读者快速掌握本书的教学内容，作者特为本书提供售后服务QQ群4661142，读者可以申请加入该群参与讨论，以及就阅读过程中遇到的疑难向作者提问，作者会随时在线提供解答。

除了QQ群外，本书还提供售后服务论坛，网址如下：

http://www.exceltip.net/

读者可以通过以上QQ群或者论坛反馈阅读心得，或者提交对本书的建议。

在阅读本书时，有疑问可以随时在售后论坛发帖，有专人负责解答。

也可以向作者发送邮件(邮箱地址：888@excelbbx.net或者Excelbbx@163.com)获得帮助与答疑。

此外作者还向读者赠送Excel百宝箱，包括安装文件、源代码和动画帮助。其界面如下：

博文视点Broadview

@博文视点Broadview

策划编辑：张慧敏
责任编辑：王　静
封面设计：侯士卿

上架建议：办公软件>Excel

ISBN 978-7-121-24032-4

9 787121 240324 >

定价：75.00元（含光盘1张）

Excel VBA

程序开发自学宝典 第3版

罗刚君 著

電子工業出版社
Publishing House of Electronics Industry
北京•BEIJING

内 容 简 介

《Excel VBA 程序开发自学宝典（第 3 版）》是 VBA 入门与提高的经典教材。全书分上下两篇，上篇包含入门知识，对 VBA 的基础理论、语法规则、编写思路、代码优化思路等都提供了详尽的理论阐述和案例演示。下篇包含进阶知识，提供窗体设计、正则表达式、字典、FileSystemObject、类模块、注册表、功能区设计、开发加载宏、封装代码等高级应用。

本书基于 Excel 2010 撰写，不过代码可在 Excel 2007、Excel 2010 和 Excel 2013 中通用。

本书是《Excel VBA 程序开发自学宝典（第 2 版）》的升级版，在升级过程中做了大量（不少于 60%）的修改，包括调整章节顺序、舍弃部分实用性不大的内容、修改书写方式、完善代码的含义注释、删除已经过时的一些技巧、增加全新案例等。

本书向读者赠送了更新版的 Excel 百宝箱的安装文件、源代码和动画帮助，保存在随书光盘中。此外本书还提供售后服务 QQ 群（QQ 群号码：4661142），以及售后服务论坛（网址为 http://www.exceltip.net/），在阅读过程中如有任何疑问，读者可以随时与作者沟通与反馈。

图书在版编目（CIP）数据

Excel VBA 程序开发自学宝典 / 罗刚君著. —3 版. —北京：电子工业出版社，2014.9
ISBN 978-7-121-24032-4

Ⅰ. ①E…　Ⅱ. ①罗…　Ⅲ. ①表处理软件　Ⅳ. ①TP391.13

中国版本图书馆 CIP 数据核字（2014）第 183620 号

策划编辑：张慧敏
责任编辑：王　静
印　　刷：北京天宇星印刷厂
装　　订：三河市鹏成印业有限公司
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
开　　本：787×1092　1/16　印张：33　字数：930 千字
版　　次：2009 年 10 月第 1 版
　　2014 年 9 月第 3 版
印　　次：2014 年 9 月第 1 次印刷
印　　数：4000 册　定价：75.00 元（含光盘 1 张）

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。

如何学习 VBA

VBA 是什么

Excel VBA 是依附在 Excel 中的二次开发语言，全称为“Visual Basic For Application”。VBA 已有 20 年历史，目前最新版本是 7.1，其开发环境和语法已趋于完善。

VBA 不仅是 Excel 的二次开发平台，同时还大量应用在其他软件中，包括机械设计软件 AutoCAD、平面设计与排版软件 CorelDraw、办公排版软件 Word、网页设计软件 FrontPage、项目管理软件 Project、国产 Office WPS 等，VBA 的应用前景相当广阔。

学 VBA 有用吗

这是很多网友问过的问题，其实答案只有一个——任何软件都有用，只看你学到什么程度。

不管是什么软件都能提升工作效率，以及带来经济效益，但前提是学得足够好，以及用得恰到好处。

当然，也可以换一种方式回答：别问有没有用，你有几分耕耘？

学习 VBA 的必要性

VBA 能做什么？是否有必要学习？VBA 有很多事都不能做，例如，不能开发独立的应用程序、不能开发 ERP 系统、不能实现网页设计、不能防范病毒等，但是在它的专业领域可以实现诸多令人惊奇的功能，常常让人眼前一亮：咦，制表原来可以这么快！

在工作中是否有很多需求是 Excel 做不到的？诸如底端标题、隔 N 行插入 M 行、金额大小写批量转换等。

再或者说，Excel 的某些内置函数是否让你一直不满意？例如，CONCATENATE 函数不能合并数组也不能合并区域；Vlookup 函数只能返回第一个找到的对象；SUM 函数不能实现按颜色汇总，也不能对超过 15 位的数据求和……这些都是 VBA 的强项，所有问题都可用 VBA 轻松化解。

当然，VBA 更重要的应用在于开发插件和设计运算系统（财务报表、人事管理系统、仓库进销存等）。当 VBA 的功能发挥到极致时，很多平常需要数小时的计算工作，VBA 能在三两秒内完成。“秒杀”对于 VBA 而言易如反掌。

学习 VBA 的基础

学 VBA 需要会英语，需要懂 VB 或者 C++，这是在与网友们交流中看到的最多的一种说法。其实不然，VBA 与英语没有任何关系，一个不认识任何英语单词的人也可以学好 VBA，就像笔者自己，在完全不懂英语的情况下自学了 6 个月，就掌握 VBA 的初、中级应用，至今已经出版了 7 本 VBA 相关的图书。

当然，懂英语对学习 VBA 是一个辅助条件，可以看懂一些外国的参考资料，但绝不是必要条件，国内的 VBA 资料已足够丰富。

从另一个角度讲，如果学 VBA 需要懂英语，那岂不是懂英语的人对于 VBA 一看就会？事实上并非如此。一个英国人要学习 VBA 并不比一个中国人的学习时间短，而且国际知名的 VBA 插件和 VBA 高手也基本与英国无关，这也反证了英语不是学习 VBA 的基本要求。

VB 和 C 语言是否是学习 VBA 的基础呢？当然也不是。不过懂 VB 或者 C 语言对于学习 VBA 是有帮助的，主要体现在编程的理念和思路上。一个 VB 或者 C 语言高手必定已经养成程序员的严谨和逻辑性等良好习惯，这种习惯和思维对学习 VBA 有比较大的帮助，但并非 VB 和 C 语言程序本身构成学习 VBA 的基础，它们所涉及的对象大不相同。一个 C 语言专业程序员转学 Excel VBA 仍然需要逐个学习 Excel 的对象、属性和方法，没有捷径可走。

那么学习 VBA 的基础究竟是什么呢？笔者的看法是：了解什么是单元格、工作表、工作簿，会使用条件格式、定义单元格格式，懂得排序、筛选、填充、插入图形对象、分列、创建图表等操作。当然，还需要认识 26 个英文字母。

简单吗？是的，学习 VBA 的基础条件就这些，如果你都会，那么祝贺你已步入 VBA 潜在用户之列。

当然，若要成为好的程序员，还需要有耐心、周密的思维能力、充分的逻辑性，以及举一反三的能力等。

学习 VBA 需要背英文单词吗

当然不需要。举一个例子：100 以内的加减法人人皆会吧？会计算 100 以内的加减法是因为背下了 100 以内的所有加减法表达式的答案，还是因为掌握了加减法运算技术呢？

1+1,1+2,1+3,1+4…2+2,2+3,2+4…也就是说，仅 100 以内的加法表达式就有 5000 多个，减法表达式也有 5000 多个，还不包括小数。把这么多题目的答案背下来是不可能的事，但是懂得计算方法后要快速地得到 100 以内的加减法答案却是极容易的事。

掌握了方法，一通百通。

VBA 同样也是这个道理。

如何发挥 VBA 的潜能

VBA 的理论不多，但是极其重要，是解决一切 VBA 问题的基石。学习 VBA 需要深入理解 VBA 的对象、属性、方法、事件，以及它们的调用方式，其后则一通百通。

本书有三分之一的篇幅展示 VBA 的基础理论，三分之一的篇幅展示罗列应用案例，另外三分之一的篇幅展示分析思路及过程，阐述代码的优化与提速方法。

通读本书，并且反复操作案例，必可掌握 VBA 的精髓。

罗刚君

前　言

Excel 是所有制表软件中最优秀、市场占有率最高的一款软件，这归功于它强大且灵活的制表功能和二次开发平台，通过二次开发平台可以让用户开发新的工具，从而实现 Excel 本身不具备的功能，或者弥补 Excel 自身的不足。

Excel VBA 还可以实现操作自动化，让某些工作全自动完成，进而全方位提升工作效率，这使得 Excel 从众多制表软件中脱颖而出。

通过 VBA 进行二次开发可以强化 Excel 的功能，将某些繁杂或者重复的日常工作简化，还可以通过 VBA 开发商业插件或者小型财务系统等。可以说 Excel VBA 已完全融入办公文员的日常工作，拥有 VBA 就等于拥有效率。

本书结构

《Excel VBA 程序开发自学宝典（第 3 版）》是适合自学的 VBA 教材，它包含了 Excel VBA 的所有基础理论和高级应用。全书 24 章，分上下篇，各包含 12 章。上篇讲述 VBA 相关的基础理论及综合练习，下篇提供 VBA 高级应用的相关知识。

上篇主要介绍 Excel VBA 的基础知识，并通过这些知识的综合应用加深读者的理解。具体包含 VBA 代码的产生方式、存放方式、调用方式、保存方式、程序结构、四大基本概念(对象、属性、方法和事件)、变量与数据类型、常用语句的语法介绍(包含创建输入框、条件判断语句、循环语句、错误处理语句、选择文件与文件夹)，然后提供综合应用案例，帮助读者理解前面所介绍的基础知识，从而让知识系统化。

最后还提供编程规则与代码优化技巧，以及编程的捷径，教读者掌握更高效的编程方式，以及提升程序的效率。具体的章节名称如下：

第 1 章　初步感受 VBA 的魅力　　第 2 章　VBA 程序入门
第 3 章　VBA 的程序结构分析　　第 4 章　VBA 四大基本概念
第 5 章　通过变量强化程序功能　　第 6 章　深入剖析常见对象的引用方式
第 7 章　常用语句的语法剖析　　第 8 章　让代码自动执行
第 9 章　综合应用案例　　第 10 章　编程规则与代码优化
第 11 章　利用参数强化过程　　第 12 章　编程的捷径

下篇主要介绍 Excel VBA 的高级应用，包含数组、正则表达式、自定义函数、设计窗体、FSO、类模块知识、注册表、功能区设计、插件设计和封装代码等知识。其中重点在于数组、字典、窗体、功能区、开发插件，对于任何一个 VBA 高级用户而言这些领域都是不可或缺的，掌握这些应用后才能开发出大中型的高效的程序。具体的章节名称如下：

第 13 章　利用数组提升程序效率　　第 14 章　正则表达式与 VBA
第 15 章　详解字典应用　　第 16 章　开发自定义函数
第 17 章　设计窗体　　第 18 章　处理文件与文件夹
第 19 章　认识类和类模块　　第 20 章　VBA 与注册表
第 21 章　Ribbon 功能区设计　　第 22 章　开发通用插件
第 23 章　代码封装技巧　　第 24 章　开发逐步提示的数据录入助手

本书特点

相对于同类书籍，本书在内容编排上具有以下特点。

（1）除了对 VBA 语言的基础语法与常见对象的综合应用介绍以外，重点展示如何开发一个独立、完善、拥有专用菜单的通用程序，E 灵（早期名称为“Excel 百宝箱”，官方网址为 http://excelbbx.net）正是基于本书所介绍的知识点而开发的。

通过本书，你完全可以编写出自己的商业插件，也可以通过 Excel 插件大幅度提升工作效率，让以往可能需要一个小时的工作量在几秒内即可完成。

（2）本书基于 Excel 2010 写作，但是代码通用于 Excel 2007、Excel 2010 和 Excel 2013。由于 Excel 2003 即将被淘汰，因此本书不再讲述传统菜单的设计方法，而是重点讲述功能区的开发思路，并提供若干功能区模板，从而让读者可以快速设计功能区组件。

（3）本书比较注重代码的通用性和效率，总结了多条优化代码的规则。

（4）正则表达式可以强化 VBA 的字符处理能力，本书详细地阐述了正则表达式的调用方法、语法，并提供了大量案例与思路，这在所有 VBA 书籍中是独一无二的。

（5）详细教学保护代码的方法，防止他人查看自己的程序源代码。同时展示 VBA 代码加工成 exe 格式的可执行程序的思路，提升代码的易用性和专业性。

（6）随书光盘中提供了书中的所有案例源代码，并对每一句代码提供含义注释，便于读者快速理解过程的含义与设计思路。

光盘文件介绍

本书提供随书光盘一张，光盘中存放了案例文件、视频教材和 Excel 百宝箱。

1. 案例文件

本书的随书光盘中存放了书中的所有案例文件的源代码，读者在学习本书前应该将随书光盘中的文件复制到磁盘中，然后将案例文件配合图书阅读，从而提升学习速度。切不可通过手工逐字摘抄书中代码的方式来测试代码，因为摘抄代码的出错概率太高了。

2 . 视频文件

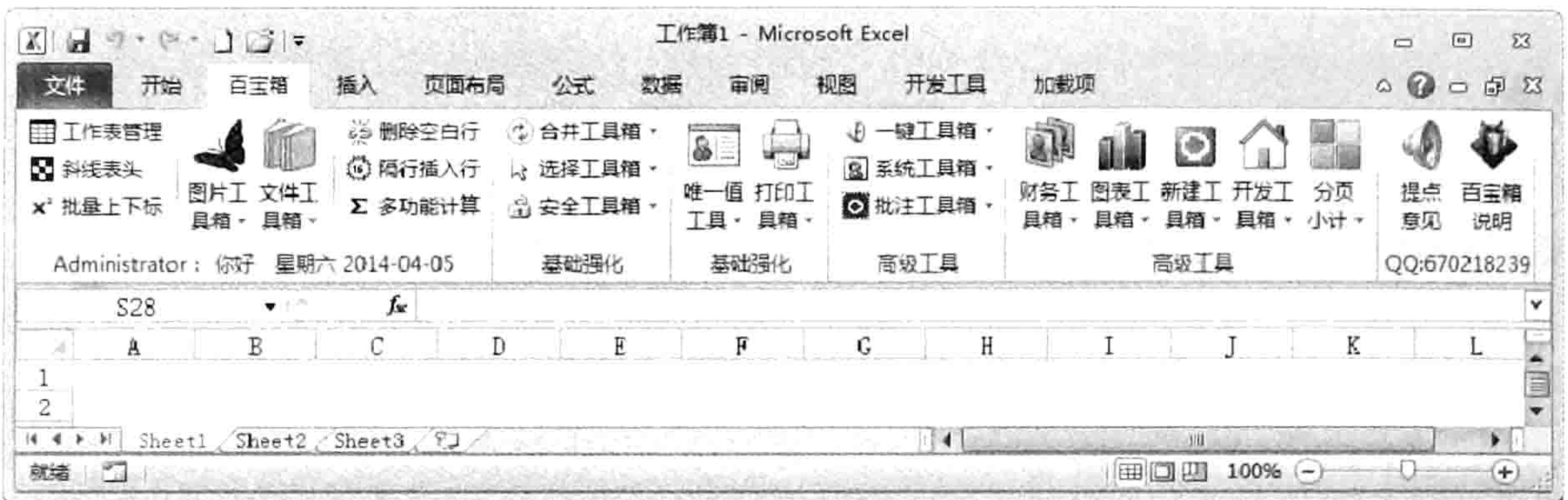
本书的随书光盘提供了 8 集 VBA 视频教材，读者可以看完本书后再观看视频教材，但不宜先看视频后看图书。

3. Excel 百宝箱

为了感谢读者对本书的支持，特赠送大型插件“Excel 百宝箱”。“Excel 百宝箱”是一个 Excel 插件，拥有 100 多个功能，可以强化 Excel，弥补 Excel 的某些缺陷——例如合并单元格再取消合并后会丢失数据的毛病，再如 Excel 只支持顶端标题不支持低端标题、可以新建工作表但不能批量新建等问题。

随书光盘中提供了“Excel 百宝箱”的安装文件，插件是开源的，未设置密码保护，读者可以随时查看其中的所有源代码。为了方便读者使用，光盘中还同步提供“Excel 百宝箱”的动画教材，动画教材来自录像工具所生成的插件安装步骤和使用步骤，读者可以通过该动画教材快速掌握插件的每一个功能。

下图是“Excel 百宝箱”的操作界面：



4. 365 个疑难问题

为了提升读者解决问题的能力，以及扩展读者的知识面，特在光盘中提供 365 个常见疑难问题及答案。

适合读者群

本书对 VBA 的基础理论有比较详尽的介绍，并提供了大量的案例引导读者逐步深入。

本书上篇介绍 Excel VBA 的入门知识，适合编程零基础的人群阅读。

本书下篇提供 Excel VBA 的高级知识应用，包括数组、类模块、正则表达式、功能区、窗体、封装代码等高级应用，同时还提供多个插件的开发过程演示，强化读者的动手能力，从实战中将基础理论转换为实战技能。

因此，适合本书的读者包括三类：

（1）VBA 零基础者，通过本书踏入 VBA 的大门。

（2）已有 VBA 基础但需要扩充知识面者。本书涉及的 VBA 知识相当全面，包含了学习 VBA 所必需的基础知识，也提供了正则表达式、FSO、字典、注册表、功能区设计和封装代码等边缘性知识，从而让读者对 VBA 掌握得更全面。

本书还提供代码优化的诸多规则，掌握这些规则可让程序具有更强的通用性和执行效率。

（3）已对 VBA 有相当的认识，但想开发更专业的商业软件者。本书对开发加载宏、自动化加载项和 COM 加载项都有详细的阐述。同时还提供制作安装程序的教学思路。

售后服务

为了帮助读者快速掌握本书的教学内容,作者特为本书提供售后服务 QQ 群（QQ 群号码：4661142），读者可以申请加入本群参与讨论,以及就阅读过程中遇到的疑难向作者提问,作者会随时在线提供解答。

除了 QQ 群外，本书还提供售后服务论坛，网址如下：

http://www.exceltip.net/

读者可以通过以上 QQ 群或者论坛反馈阅读心得，或者提交对本书的建议。同时，如果读者发现书中有错别字，也请读者在 QQ 群中进行反馈，我们在下次印刷时将会改进。

在阅读本书时，有疑问可以随时在售后论坛发帖，有专人负责解答，周末无休。

也可以向作者发送邮件（邮箱地址：888@excelbbx.net 或者 Excelbbx@163.com）获得帮助与答疑。

感谢

参与本书编写工作的还有杨嘉恺、王丽云、姚书杰、龚莉、马世梅、余敏键、龚丹、王军、吴炎翠、王益明、杨阳、胡清春、葛度金、章兰新，在此表示感谢！

罗刚君
2014 年 7 月

目　录

上　篇

下　篇

第 1 章　初步感受 VBA 的魅力

简单地说，Excel VBA 是依附于 Excel 程序的一种自动化语言，它可以使程序自动执行、批量执行、定时执行……类似于 DOS 操作系统中后缀名为“.bat”的批处理文件，但它比 DOS 系统的批处理功能更强大。在进入 VBA 的实质性开发阶段之前，先来感受 VBA 的独特魅力吧！

需要特别说明的是——本章仅向读者展示 VBA 的优越性，让读者通过两个案例了解 VBA 的自动化，对于案例中所涉及的每句代码有何含义及程序设计思路请完全忽略，在后面的章节自有详解。

1.1　批量任务一键执行

VBA 可以一键执行批量任务，大幅度提升制表效率。某些原本需要几小时方可完成的工作量改用 VBA 程序来实现则往往仅需要几秒钟，此类案例不胜枚举。本节通过从身份证号码中提取信息向读者展示 VBA 的魅力，同时也引出后续章节的 VBA 编程教学。

1.1.1　准备工作

本书的所有案例文件都存放在随书光盘中，请读者将光盘中的“案例文件”文件夹复制到计算机的硬盘中，然后再跟随书中的操作步骤测试代码。

如果你的计算机没有光驱，或者光盘在运输过程中被损坏，请与作者罗刚君联系。联系方式包括 QQ（QQ 号码：670218239）和电子邮箱（电子邮箱地址：888@excelbbx.net）。

本书以 Excel 2010 为蓝本进行讲解，默认采用 xlsm 格式的文件，Excel 2003 或者 Excel 2013 的用户也可以按相同的步骤学习。

1.1.2　程序测试

假设你已经将随书光盘中的“1-1 一键提取身份证信息.xlsm”文件复制到硬盘中，请按以下步骤操作。

step 1　双击打开“1-1 一键提取身份证信息.xlsm”文件。

step 2　如果在工作表上方弹出如图 1.1 所示的安全警告，则单击右方的“启用内容”按钮。

图 1.1　安全警告

step 3　选择 B 列所有存放身份证号码的区域——假设为 B3:B6，然后单击首行单元格中的“从身份证号码获取信息”按钮，程序会根据所选身份证号码瞬间生成对应的性别、出生日期和年龄等信息。效果如图 1.2 所示。

	A	B	C	D	E	F	G	H	I
1	从身份证号码获取信息			人事资料表					
2	姓名	身份证	性别	出生日期	年龄	部门	工号	学历	住址
3	赵	511025198905126171	男	1989/5/12	24				
4	钱	440104198403265142	女	1984/3/26	29				
5	孙	120221780915326	女	1978/9/15	35				
6	李	310101200112205137	男	2001/12/20	11				

图 1.2　根据身份证号码提取职工年龄、出生日期与性别

本例文件参见光盘：..\第一章\ 1-1 一键提取身份证信息.xlsm

1.1.3　案例点评

在前面的案例中，已知身份证号码可以提取身份证号码持有人的性别、出生日期和年龄，而且不管选中的是单个还是数万个身份证号码都可以在几秒钟提取所有信息。如果在制作人事资料表时手工逐一录入身份证号码所对应的性别、出生日期和年龄，那么录入 10000 条数据估计得耗费 10 小时，而利用 VBA 代码可以在几秒钟内完成，这正是 VBA 的魅力体现。

1.2　多工作簿自动汇总

将文件夹中所有工作簿的所有工作表汇总到一个工作表中，这是很常见的工作需求。按常规的操作方式——逐一打开工作簿并逐一复制所有工作表中的数据到活动工作表中再汇总，这可能耗费几十分钟，还无法确保没有遗漏某些数据。而采用 VBA 跨工作簿汇总不仅快捷、准确，甚至都不需要按快捷键或者单击菜单，只要打开工作簿就全自动完成。

在接触 VBA 之前，读者可能会产生疑问：这有可能吗？完全有可能！在 VBA 的世界里，瞬间完成操作和全自动执行命令是极其常见的。本节将展示打开工作簿时全自动汇总的案例。

1.2.1　案例需求

在“生产日报表”文件夹中存放了若干个工作簿，每个工作簿中有若干个工作表，每个工作表中有若干行产品生产记录，这些数据的行数都不确定。图 1.3 和图 1.4 分别展示了文件夹中的工作簿，以及工作簿中的数据结构。

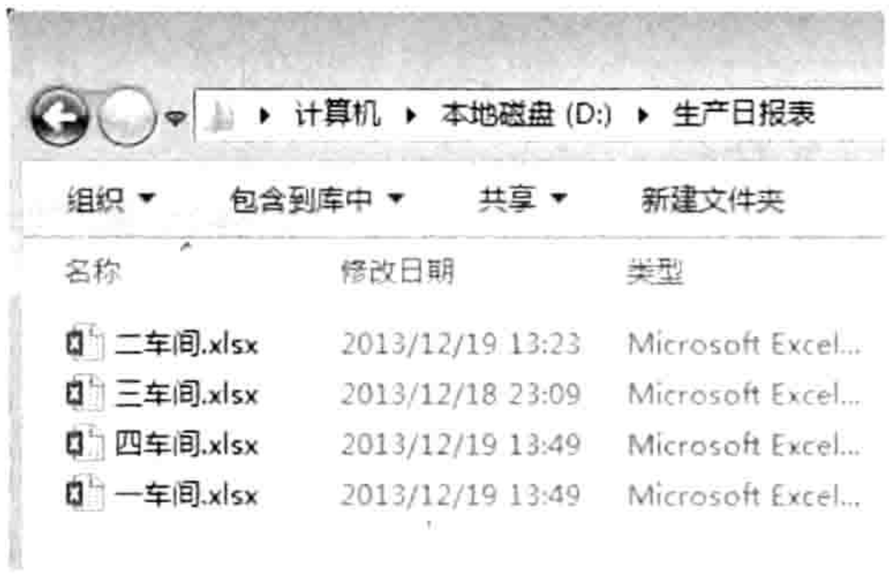


图 1.3　生产日报表

	A	B	C	D
1	姓名	产品	产量	不良品
2	诸华	连接螺丝	87	11
3	周至强	方螺帽	93	1
4	宁湘月	方螺帽	95	2
5	周鉴明	内六角螺丝	83	11
6	黄土尚	铝连杆	86	6
7	刘专洪	内六角螺丝	112	14
8	张庆	连接螺丝	101	14
9	朱未来	双头螺丝	109	8
10	陈秀雯	扳手螺丝	104	0
11	周华章	异形螺帽	104	9

A线 B线 C线

图 1.4　生产数据

现要求对“生产日报表”文件夹中所有工作簿的生产数据按产品名称分类汇总，并且该文件夹中的工作簿数量增减或者工作簿中的数据增减后，汇总结果也会相应变化。

假设“生产日报表”文件夹中有数百个工作簿，每个工作簿中有数十个工作表，人工逐一汇总将是相当浩大的工程，可能会用 10 分钟，也可能会用 1 小时，视工作簿数量的多少而定。

然而，采用 VBA 代码汇总可以全自动完成，打开“汇总表.xlsm”后不用做任何事情就已经汇

总成功。你是否怀疑 VBA 能此等智能呢?

1.2.2 程序测试

随书光盘中已经提供了 4 个待汇总的工作簿和用于汇总的程序代码，读者可以使用它们测试代码的正确性和执行效率。具体操作步骤如下。

step 1 将光盘中“第一章”文件夹下的“1-2 合并工作簿”文件夹复制到硬盘中。

step 2 双击打开“汇总表.xlsm”文件，假设弹出了如图 1.1 所示的安全警告，单击右方的“启用内容”按钮。

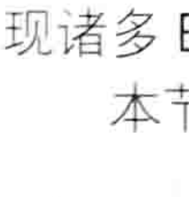
在打开工作簿的瞬间，工作簿中的 VBA 代码会全自动汇总当前路径下的所有生产数据。如图 1.5 所示是汇总结果。

事实上，“汇总表.xlsm”中的代码并非是专门针对图 1.3 中所示的 4 个工作簿所写的，而是具有通用性，即使该文件夹中的工作簿增加到 100 个，每个工作簿中的工作表数量也增加到 100 个，不需要修改代码，打开“汇总表.xlsm”后同样可以瞬间完成汇总。

	A	B	C
1	产品名称	产量	不良品
2	异形螺帽	416	22
3	扳手螺丝	1535	91
4	铝连杆	634	40
5	方螺帽	1497	72
6	连接螺丝	911	87
7	压花螺帽	516	31
8	方管连杆	1033	56
9	钢轴	1675	133
10	双头螺丝	1183	83
11	外六角螺丝	651	36
12	内六角螺丝	708	35
13	十字盘头螺钉	515	46
14	旋转螺丝	451	26

Sheet1

图 1.5 汇总结果



本例文件参见光盘：..\第一章\1-2 合并工作簿\汇总表.xlsm

1.2.3 案例点评

以上案例主要涉及了 VBA 的多个知识点，包括变量与数据类型、工作簿事件、循环语句、数组、合并计算、区域引用等。将它们组合后可以自动汇总具有相同格式的工作簿，既快捷又准确，也不会遗漏任何数据，比人工分类汇总能千百倍地提升效率。

事实上，这也是 VBA 之所以吸引用户的原因之一。

1.3 浅谈 VBA 优势

Excel 是制表行业中最优秀的表格软件，它提供了诸多的数据处理工具，其中最强大的莫过于二次开发工具 VBA，它不仅能批量执行任务，全自动执行命令，还可以改善 Excel 的内置功能，实现诸多 Excel 原本无法实现的功能。

本节向读者阐述 Excel VBA 相对于 Excel 其他功能在制表工作上的一些优势。

1.3.1 批量执行任务

常规的制表手法只能一次执行一项任务，例如删除 100 个工作表中的错误值，需要手工操作 1000 次以上，包括逐个激活工作表，然后定位错误值所在的单元格，再删除错误值。如果改用 VBA 删除 100 个工作表中的错误值可以一键完成，整个过程不超过 1 秒钟。

删除错误值的代码可以重复使用，一键调用，而且自动适应数据变化。

1.3.2 将复杂的任务简单化

Excel 的诸多小功能可以搭配使用，从而实现比单个工具更强大的功能。例如公式、定位、插入行三者配合可以实现在工作表中隔行插入行。然而此操作过程过于烦琐，也很难在短时间内教会他人使用。用 VBA 开发一个隔行插入行的工具则可以一键完成，既提升操作效率又减少教导他人使用的时间成本。

再如使用公式从身份证号码中提取年龄和性别信息，需要多个函数嵌套使用，录入长长的公式既费时费力，又加大查看报表者的理解难度。使用 VBA 一键生成结果可以全方位地简化工作流程，以及减少教学成本。当公司有新人进来时，可以不用再花太多的时间教其函数与数组公式的用法，或者多个内置功能的嵌套技术，仅需要告知单击某个按钮能实现某个功能即可。

1.3.3 提升工作表数据的安全性

利用 VBA 代码可以对数据进行多层保护，在特殊需求下，VBA 可以保护数据不让普通用户胡乱修改，甚至可以让未授权的用户只能看到乱码，授权后才能看到原文。

1.3.4 提升数据的准确性

VBA 的准确性体现在录入数据和运算数据两个方面。

首先，VBA 可以对用户录入的数据执行限制，从而防止用户意外录入不规范字符。例如录入数值时误录入了两个小数点，或者意外插入了字母导致后期运算出错。

其次，使用公式统计数据时公式不会随数据增减变化而变化。例如使用公式统计所有工作表 B 列的产量，当新建一个工作表后公式的计算结果无法自动更新，而 VBA 可以自动适应数据的增减变化，这是 VBA 独有的一个优势。

1.3.5 完成 Excel 本身无法完成的任务

录入阿拉伯数字时自动转换成英文大写形式的金额、录入公历日期自动提示对应的农历日期、行程安排与预告，或者修改注册表等需求皆无法通过 Excel 的常规方式实现，而 VBA 处理此类问题则得心应手。

1.3.6 开发专业程序

利用 VBA 还可以开发专业性的程序，例如报表汇总软件、进销存管理系统、人事管理系统等，也可以通过 VBA 开发表格插件。笔者本人就开发了一个大型的 Excel 插件——E 灵，它包括 180 多项功能，可以大大扩展 Excel 的应用领域。软件的官方网址为：http://excelbbx.net/。

E 灵的界面如图 1.6 所示。

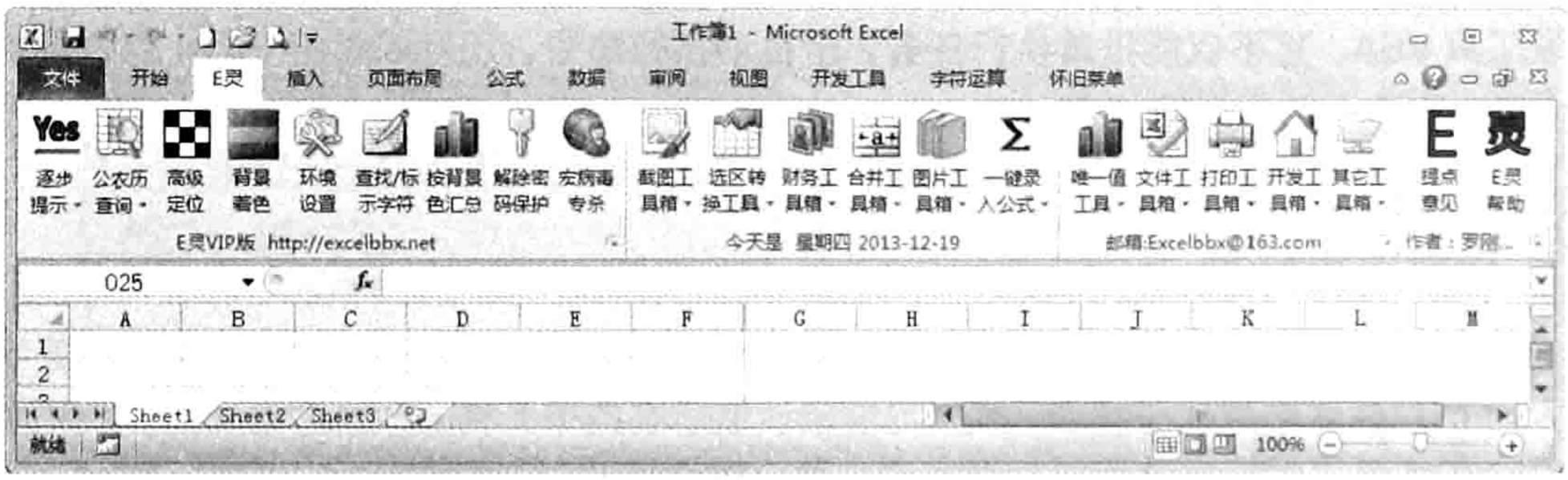


图 1.6 Excel 插件 E 灵的操作界面

通过前面的案例演示和对 VBA 优势的阐述，你是否已经被 Excel VBA 的魅力所折服呢？

从第 2 章开始，本书将带领读者走进 Excel VBA 的世界，去尽情领略 VBA 的强大功能，同时也要学会驾驭 Excel VBA，以致为我所用，提升制表工作的效率。

◆ ◆ ◆

第 2 章　VBA 程序入门

编程的重点在于熟悉语法、思路灵活，以及善用代码模板。然而进入实质性的编写代码之前有必要了解一些与编程相关的基本常识，包括代码放在哪里、如何产生代码、如何保存代码，以及如何放行被宏安全性挡住的代码等入门知识。

2.1　如何存放代码

学习 VBA 往往并不是从自己编写代码开始，而是先从网上复制他人写好的代码，或者摘抄教材中的现成代码，然后再逐步掌握语法，从而学会修改与编写代码。在这个过程中，涉及了代码存放位置的问题，只有将代码保存在正确的位置才能发挥代码的功效。

2.1.1　认识模块

VBA 的前身是宏（Macro），一段 VBA 程序也曾被称为一个宏，而在 VBA 中更专业的称谓是过程。一段完整的 VBA 程序就是一个过程。

过程分为三种——以 Sub 开头的子过程、以 Function 开头的函数过程，以及以 Property 开头的属性过程。在实际工作中 90%以上的情况下都在使用子过程，因此本书前 15 章都只涉及子过程，在第 16 章才详解开发自定义函数，展示 Function 过程的结构、语法和开发思路。属性过程在工作中一般不用，本书不涉及属性过程的教学。

子过程有多种用法，采用不同用法时代码的存放处所也各不相同，但是比较通用的办法是将子过程代码存放在模块中。至于其他的存放方式会在后面更高阶的章节中有相应的介绍。

那么什么是模块？如何调出模块的界面？请按以下步骤操作。

step 1　打开 Excel 进入工作表界面。

step 2　按<Alt+F11>组合键打开 VBE 窗口，如图 2.1 所示是默认的 VBE 窗口，它包含了 VBA 程序相关的菜单、工具栏、工作簿名称、工作表名称、属性窗口和帮助查询窗口。

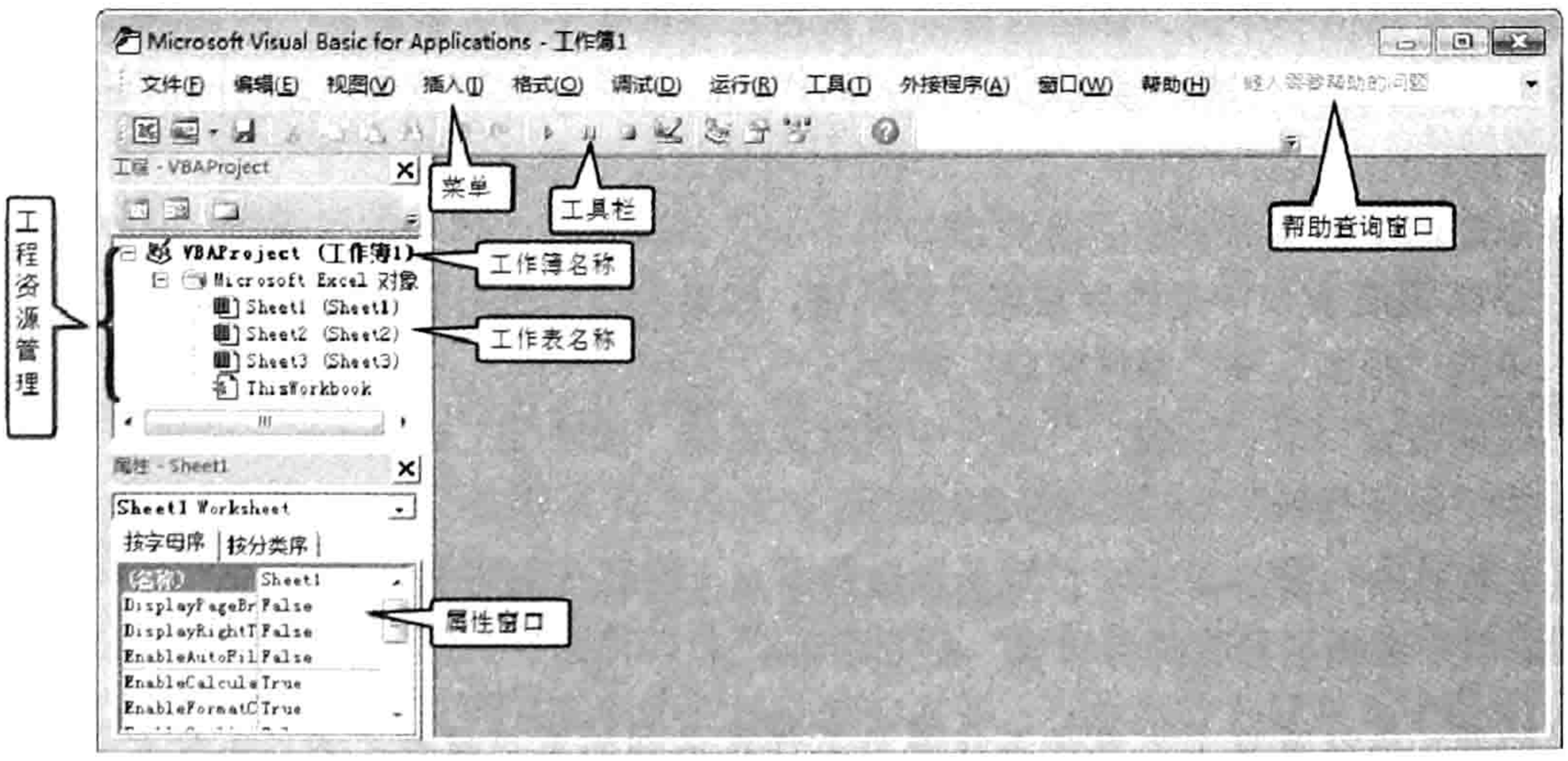


图 2.1　默认的 VBE 窗口

默认的 VBE 窗口没有任何模块，需要手工添加模块。

知识补充：在 VBA 中，一个工作簿拥有一个工程（VBAProject），Excel 允许同时打开多个工作簿，因此在 VBE 界面中也可能存在多个工程。如果读者确认自己只打开了单个工作簿，但却在 VBE 界面中发现了多个工程，这说明你安装了若干个加载宏。可以在工作表界面按<Alt+T+I>组合键打开“加载宏”对话框，在该对话框中会罗列出当前已经安装的加载宏，例如“分析工具库”、“规划求解加载项”等。

step 3 单击菜单中的“插入”→“模块”命令，在属性窗口上方将会出现一个默认名称为“模块 1”的模块，右侧的空白窗口则是此模块对应的代码窗口，用于存放 VBA 程序代码，如图 2.2 所示。

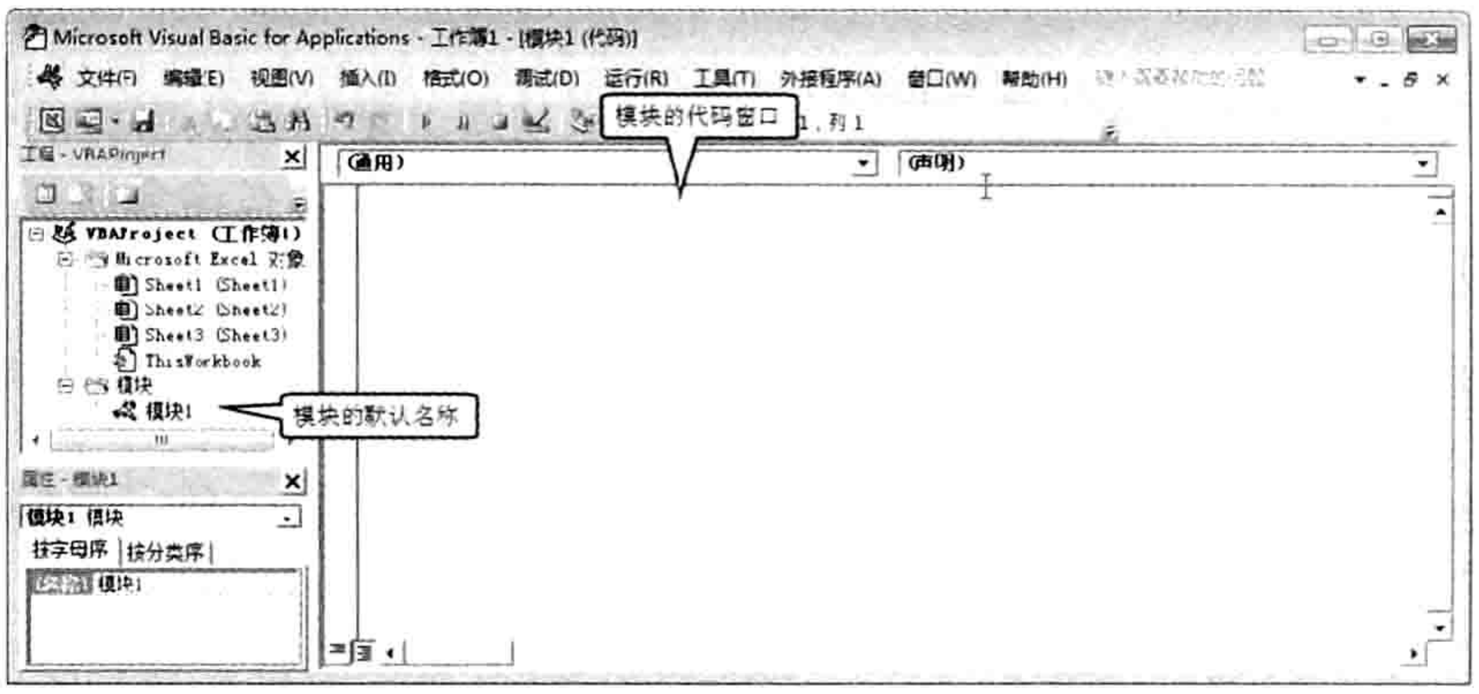


图 2.2 模块与模块代码窗口

step 4 如果需要更多的模块，那么再次单击菜单中的“插入”→“模块”命令，在属性对话框上方将出现名为“模块 2”的空白模块。

以上步骤用于插入模块，在 2.2 节中会讲述如何在模块中产生代码。

2.1.2 管理模块

模块用于存放程序代码，同一个模块中不宜存放太多的过程代码。为了便于管理，通常将代码分类存放，例如第一个模块存放与“财务”相关的程序代码，第二个模块存放与“人事”相关的程序代码。

当模块数量超过一个时，需要对模块重命名，尽量使任何人看到模块名称就能明白每个模块中存放了哪方面的代码。对模块重命名可按以下步骤操作。

step 1 单击选中要重命名的“模块 2”。

step 2 查看 VBE 窗口中是否存在属性对话框，如果没有则单击菜单中的“视图”→“属性窗口”命令。

step 3 在属性窗口的“（名称）”属性中将默认名称“模块 2”修改为“财务模块”，效果如图 2.3 所示。

图 2.3 将“模块 2”重命名为“财务模块”

如果需要删除“财务模块”，那么在模块上单击鼠标右键，从弹出的右键菜单中选择“移除财务模块”命令即可。

知识补充：属性窗口上方包含工作簿与工作表名称的窗口被称之为“工程资源管理器”，如果不小心被关闭会给查看代码和编程带来障碍，此时可以单击菜单中的“视图”→“工程

资源管理器”命令将它调出来。

2.2 如何产生代码

有 4 种方式可以产生代码，包括复制他人已经编好的程序代码、录制宏产生宏代码、手工编写程序，以及调用代码模板然后加以修改，本节一一剖析它们的区别与利弊。

2.2.1 复制现有的代码

几乎所有人在初学 VBA 时都是复制他人编好的代码来使用，待掌握好 VBA 的对象、属性、方法和事件等基础概念，以及循环语句、条件语句等常用语句后才自己编写代码。

复制代码的渠道有很多，主要包含以下两种。

1. 从随书光盘中获得代码

介绍 VBA 的图书都配有光盘，读者在初学 VBA 时应尽量复制光盘中的代码去测试，而不是将代码中的字符逐个录入到模块中。因为手工抄写代码出错的几率比较高，除了人为录入失误和看错字符导致出错外，还有某些代码比较长以致在书中印刷为 2 行或者 3 行，而读者很难正确地判断书中印刷的多行代码是一句代码还是多句代码，特别是代码中的空格刚好位于行末或者行首时比较容易判断失误。

鉴于以上分析，初学者在学习代码时应该复制随书光盘中的代码来使用，待学完 VBA 的基础知识后再手写代码，否则一旦代码出错将无所适从。初学者根本不具备调试程序的能力，一旦出错的次数过多又不能及时纠正，就会影响继续看书的兴趣和动力，同时打击初学者的信心，以致放弃学习 VBA。

2. 从论坛复制代码

各大 Excel 论坛都有大量公开分享的代码，可以从论坛中复制需要的代码到自己的模块中，或者学习他人的编程思路。

目前中国比较有影响力而且内容更新比较快的两大论坛有：

http://www.exceltip.net/；

http://club.excelhome.net/。

2.2.2 录制宏

录制宏是学习 Excel VBA 的便捷工具，不管是初学者还是具有多年编程经验的老程序员都会通过录制宏来产生宏代码，然后再根据需求修改宏代码。

录制宏时使用的是 Excel 自带的操作记录器，它可以用代码记录下用户的当前操作。当结束录制后可以重播代码，让代码代替手工操作去批量执行相同的命令，从而减轻用户的工作量。它类似于生活中的录音机，例如老师教第一批学生时可以将声音录制下来，教第二批学生、第三批学生时只要播放录制好的光盘或者磁带即可，不再需要一遍遍重复口述，从而减轻工作量，这与 VBA 的录制宏如出一辙。

下面以录制“清除工作表中所有图形对象”宏为例，介绍录制宏与重播宏代码的步骤。

step 1 新建一个空白工作簿，在默认的 3 个工作表中都插入多张任意图片。

step 2 单击状态栏左方的“录制宏”按钮，从而调出“录制新宏”对话框。“录制宏”按钮的外观如图 2.4 所示。如果读者使用的是 Excel 2013，由于 Excel 2013 在默认状态下关闭了“录制宏”按钮，因此需要在状态栏单击鼠标右键，在弹出的右键菜单中选择“录制宏”选项。如图 2.5 所示为 Excel 2013 中“录制宏”按钮的外观。

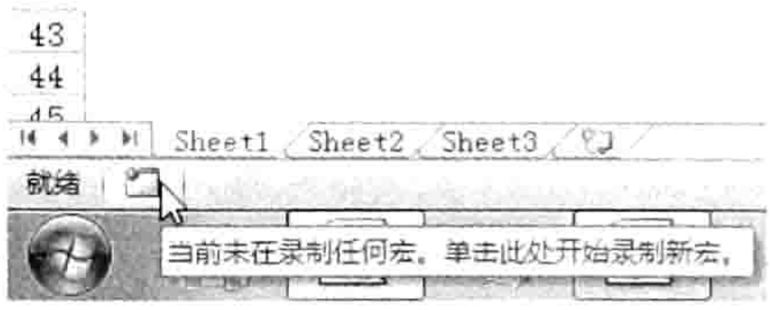


图 2.4　Excel 2010 的录制宏按钮

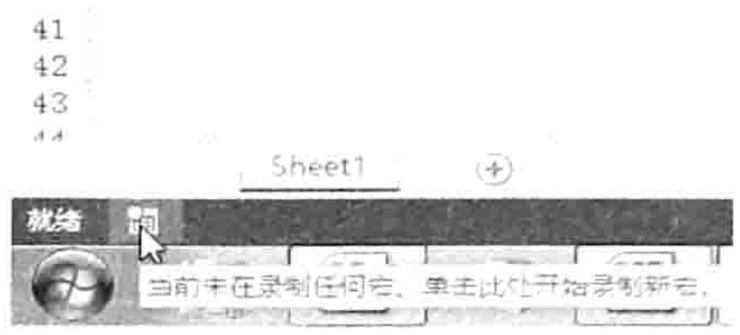


图 2.5　Excel 2013 的录制宏按钮

step 3　在“录制新宏”对话框中按图 2.6 所示的方式设置。其中“宏名”保持为默认值“宏 1”即可，在“快捷键”文本框中输入小写字母 q，表示为当前宏指定的快捷键为<Ctrl+Q>，在“保存在”组合框中选择“当前工作簿”，在“说明”文本框中输入“删除所有图形对象”。

step 4　单击“录制新宏”对话框中的“确定”按钮启动录制宏。

知识补充：在录制宏阶段，所有操作都会被记录器记录下来。为了避免产生不必要的代码，启动录制宏后应小心翼翼地操作，确保每一个步骤都是必要的，不能随意操作。

step 5　按<F5>或者<Ctrl+G>组合键启动“定位”对话框，然后单击对话框左下角的“定位条件”按钮，从而打开“定位条件”对话框。

step 6　在“定位条件”对话框中选择“对象”单选框，如图 2.7 所示。

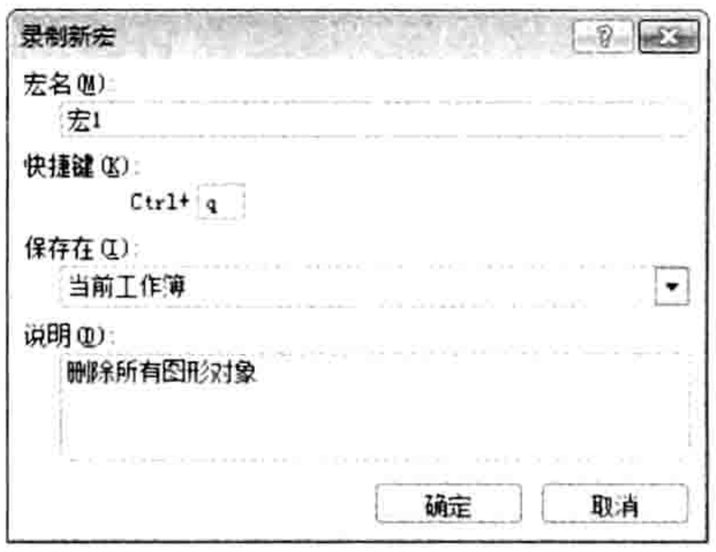


图 2.6　“录制新宏”对话框

图 2.7　定位对象

step 7　在“定位条件”对话框中单击“确定”按钮从而选中活动工作表中的所有图形对象，然后按<Delete>键删除所选对象。

step 8　单击状态栏左方的“停止录制”按钮（图标■）。

step 9　按<Alt+F11>组合键打开 VBE 界面，在模块 1 中将会看到如图 2.8 所示的宏代码。

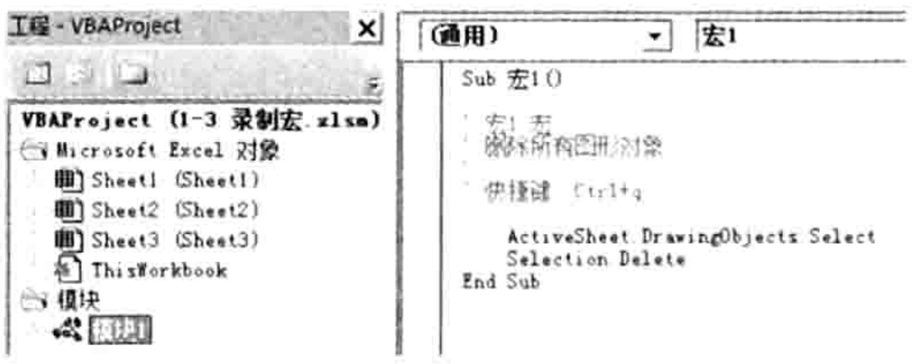


图 2.8　录制宏产生的宏代码

step 10　再次按<Alt+F11>组合键返回工作表界面。

step 11　进入 Sheet2 工作表中，按<Ctrl+Q>组合键，如果工作表中存在图形对象，此时将会发现所有图形对象都已被删除干净。

知识补充：图形对象包括图片、艺术字、文本框、图表、剪贴画和对象等。

根据以上操作步骤，读者们应该对录制宏有了初步认识，具体可以总结为以下 4 点。

◆ 如实记录操作

录制宏的目的是记录自己的所有操作，将人工操作转换成对应的 VBA 代码。这对于学习 VBA 而言极为重要，既可以通过录制宏与重播宏代码来简化某些重复性的工作，又可以为编程提供参考，避免编写程序时逐个录入字符。

通过录制宏自动产生代码，然后根据自己对 VBA 的了解对代码进行后期加工，这可以提升编程效率，而且即使忘记代码的书写方式也能编程。

◆ 宏代码保存在模块中

录制宏产生的过程一律属于子过程，以 Sub 开始，保存在模块中。

◆ 可用快捷键调用宏

录制宏时可以为宏指定包含 Ctrl 的快捷键，从而方便用户调用宏命令。当指定为小写的字母 q 时表示快捷键为<Ctrl+Q>，当使用大写字母 Q 时表示快捷键为<Ctrl+Shift+Q>。

◆ 一键执行批量命令

录制宏时不管操作了多少个步骤，调用宏时只需要单个步骤。

以上 4 点中第一点最重要。对于 VBA 爱好者而言，录制宏的目的不是使用宏，而是借助录制宏产生自己需要的代码，从而提升编程效率，以及减少记忆代码的学习时间。例如插入行的代码忘记了、创建条件格式和数据有效性的代码也忘记了，这完全不重要，只要录制宏马上就产生代码，开发者只需要将宏代码中需要的代码提取出来即可，而不再像学英语那样得反复背诵一个个单词。以此立场评价录制宏，可以说录制宏工具是 VBA 爱好者终生的良师益友，而且可以随时请教，不分场合与时间。

本例文件参见光盘：..\第二章\2-1 录制宏.xlsm

2.2.3　手工编写代码

录制宏能给编程带来莫大的帮助，不过 Excel 中的部分操作（不到 40%）无法通过录制宏产生代码，因此对于这部分操作只能自己手工编写代码。当然，如果你学完 VBA 后，掌握了 VBA 中的常用对象、方法和属性名称，也可以不录制宏，可以随手编写程序代码。

手工编写程序可以借助 VBE 中的“插入”菜单完成，具体步骤如下。

step 1　单击菜单中的“插入”→“过程”命令，弹出“添加过程”对话框，如图 2.9 所示。

step 2　在“添加过程”对话框的“名称”文本框中输入“计算工资”，将过程的类型设置为“子过程”，将范围设置为“公共的”，然后单击“确定”按钮，从而产生一个子过程的程序外壳，效果如图 2.10 所示。

图 2.9　插入子过程

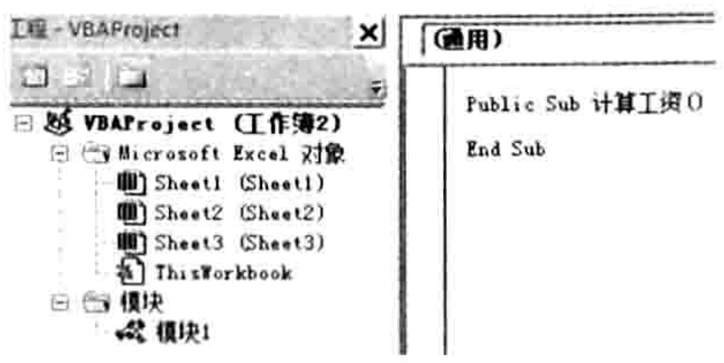


图 2.10　自动产生子过程的程序外壳

图 2.10 中的代码是一个空白的 VBA 程序，具有程序的声明语句和结束语句，没有需要执行的具体命令，该部分内容需要等读者了解 VBA 的四大基本概念，以及掌握常用语句的语法后再自行补充。

事实上，当足够熟悉 VBA 后可以直接在模块中手工录入代码，不需要借助菜单中的“插入”→“过程”命令。由于 VBA 具有智能补充代码的功能，手工声明过程同样高效。例如在模块中录入“Sub 工资条”，然后按 Enter 键，VBA 会自动将缺失的部分补充完整，产生以下代码：

```
Sub 工资条()

End Sub
```

知识补充：VBA 中的子过程的结构将在第 3 章进行详细分析，这里读者仅需要明白如何录入过程的程序外壳（即 VBA 程序的声明语句和结束语句）即可。

2.2.4 从模板中获取代码

Excel 有数百种对象，平均每个对象的属性和方法也有几十种，因此不管是多么专业的 VBA 程序员都不会将它们一一记在大脑中。通过记忆逐个录入字符的编程方式过于低效。

笔者在此给读者们一个很好的建议：在学习 VBA 的过程中，将自己每一次编写的程序的最终版本分类保存起来，当以后遇到同类需求时不需要再逐字录入代码，复制自己以前保存的代码并稍加修改即可，这样既提升编程的速度，又降低手工编码过程中的出错几率。

除了自制代码模板外，各大论坛也提供了 VBA 代码辅助工具。使用工具录入代码将大大提升录入速度，而且 100%不会出现拼写错误。笔者本人就开发过一个 VBA 辅助工具“代码百宝箱 2.0 版”，读者有兴趣的话可以通过以下网址进行下载：

http://excelbbx.net/tishi/VBAsetup.rar

http://club.excelhome.net/thread-1080867-1-1.html

安装“代码百宝箱 2.0 版”工具后的界面如图 2.11 所示。

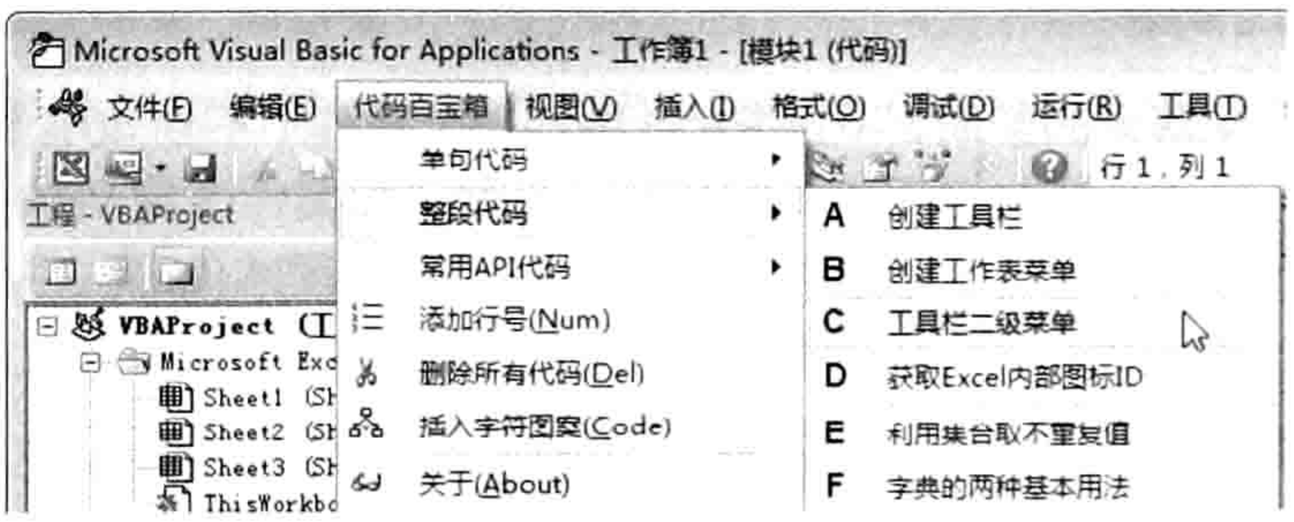


图 2.11 “代码百宝箱 2.0 版”工具

2.3 如何调用代码

VBA 程序代码有多种不同的存在形式，不同的形式有不同的调用方式。本节仅讲解模块中不带参数的子过程的调用方法，对于带有参数的子过程、函数过程、事件过程或者类模块的程序代码将在后面的章节进行讲解。

2.3.1 F5 键

保存在模块中的子过程有 5 种常用的调用方式，包括使用<F5>键、<Alt+F8>组合键、自定义快捷键、按钮和菜单。下面以调用弹出当前时间对话框的过程为例，介绍编写代码并调用过程的

具体步骤。

step 1　在 Excel 的工作表界面按<Alt+F11>组合键进入 VBE 窗口。

step 2　单击菜单中的“插入”→“模块”命令，从而创建一个新模块。

step 3　在模块中录入以下代码：

```
Sub 报告当前时间()
  MsgBox Now
End Sub
```

以上代码的含义是通过 Msgbox 函数弹出一个信息框，在信息框中显示当前的时间。如果要显示“您好”，那么将函数“Now”替换成字符串“"您好"”即可，文字前后有一对半角双引号。

step 4　单击过程“报告当前时间”的任意行代码，表示将它设置为当前过程。

step 5　按<F5>键执行当前过程，VBA 会弹出如图 2.12 所示的对话框，表示调用成功。

图 2.12　在信息框中展示当前时间

Msgbox 是一个 VBA 函数，可以返回值。不过绝大多数情况下仅仅通过它弹出一个包含指定信息和具有指定按钮的对话框。关于它的语法在本书的附录 A 中有详细说明，请打开随书光盘中的“附录 A Msgbox 函数用法说明.pdf”文件，此处了解如何调用过程即可。

知识补充：VBE 中的<F5>键只调用当前过程，如果一个模块中有多个过程时应该先将需要执行的过程切换为当前过程。单击过程的任意位置，该过程就会成为当前过程。

2.3.2　Alt+F8 组合键

在 VBE 界面中调用子过程使用<F5>键，而返回工作表界面后调用模块中的子过程则应改用<Alt+F8>组合键。下面以调用“报告今天星期几”过程为例，介绍<Alt+F8>组合键的调用步骤。

step 1　在模块中录入过程“报告今天星期几”，代码如下：

```
Sub 报告今天星期几()
  MsgBox Format(Now, "AAAA")
End Sub
```

代码中的 Format 是一个 VBA 函数，和工作表函数 Text 的功能相近，用于将数值转换成需要的格式。本例中 Format 的作用是将 Now 函数产生的时间值转换成中文的星期。

CD-ROM　本例文件参见光盘：..\第二章\2-2 调用子过程的五种方法.xlsm

step 2　按<Alt+F11>组合键返回工作表界面。

step 3　按<Alt+F8>组合键打开“宏”对话框。在此对话框中会将当前工作簿所有模块中的所有子过程都罗列出来（有参数的过程例外，后面的章节会介绍带参数的过程），如图 2.13 所示。

step 4　选择列表中的过程“报告今天星期几”，然后单击“执行”按钮，弹出如图 2.14 所示的信息框，说明调用过程成功。

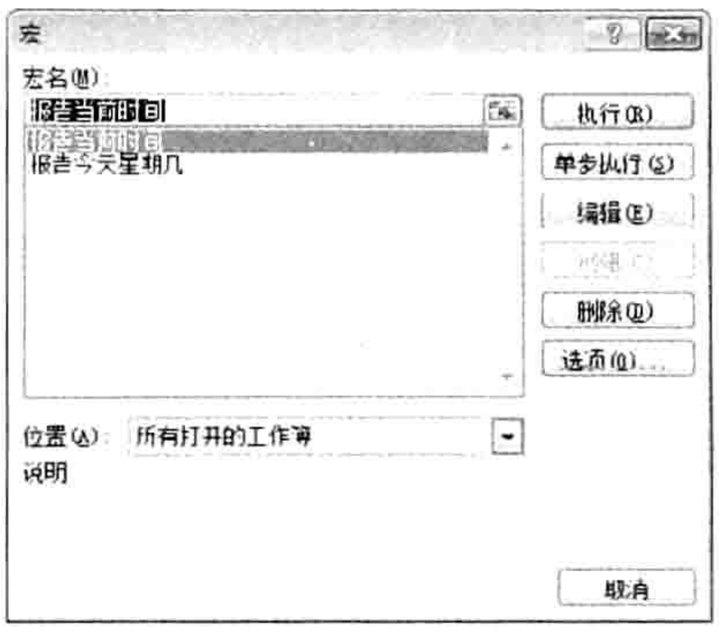


图 2.13 “宏”对话框

图 2.14 在信息框中展示今天星期几

知识补充：假设同时打开了多个工作簿，而且每个工作簿中都有无参数的子过程，那么“宏”对话框可以将所有工作簿中的子过程一并列出来。为了便于识别，它会在非活动工作簿的过程名称前添加工作簿名称，而活动工作簿中的过程则如图 2.13 所示的那样只显示过程名称。

2.3.3 自定义快捷键

在录制宏时可以为当前宏指定快捷键（在图 2.6 中有说明）。事实上，在模块中手工编写的子过程中也可以用相似的办法指定快捷键。下面以过程“报告上午还是下午”为例，介绍为其指定快捷键的步骤。

step 1 在模块中录入以下过程代码：

```
Sub 报告上午还是下午()
  MsgBox Format(Now, "AM/PM")
End Sub
```

代码中 AM 表示上午，PM 表示下午。

step 2 按<Alt+F11>组合键返回工作表界面。

step 3 按<Alt+F8>组合键打开“宏”对话框，从宏名列表中选择“报告上午还是下午”。

step 4 单击“宏”对话框中的“选项”按钮，打开“宏选项”对话框，在“快捷键”文本框中输入小写字母 q，如图 2.15 所示。

step 5 在“宏选项”对话框中单击“确定”按钮返回“宏”对话框，然后再单击“取消”按钮返回工作表界面。

step 6 按<Ctrl+Q>组合键调用过程，程序会弹出如图 2.16 所示的信息框。

图 2.15 为过程指定快捷键

图 2.16 提示当前是上午还是下午

知识补充：使用“宏选项”对话框为程序指定快捷键有所限制，其指定的快捷键只能是 Ctrl 键配合字母，而不能包含 Alt 和 Shift 两个功能键，也不能包含数值。在后面的章节会介绍使用更强大的 OnKey 方法为程序指定快捷键。

2.3.4　按钮

当模块中过程比较少时可以使用快捷键调用，若过程比较多仍采用快捷键调用则不利于记忆，使用按钮调用过程才是首选。下面以对子过程“报告今天星期几”指定按钮为例，介绍具体的操作步骤。

step 1　依次按下 Alt、T、O 三个键从而打开“Excel 选项”对话框。

step 2　选择“自定义功能区”选项，在右边的“自定义功能区”列表中在“开发工具”前打钩，然后单击“确定”按钮返回工作表界面。

step 3　单击菜单中的“开发工具”→“插入”→“按钮（窗体控件）”命令，如图 2.17 所示。然后在工作表中任意位置拖曳鼠标，从而绘制一个命令按钮。

step 4　当 Excel 弹出如图 2.18 所示的“指定宏”对话框时，在宏名列表中选择过程“报告今天星期几”，然后单击“确定”按钮。

图 2.17　命令按钮的外观

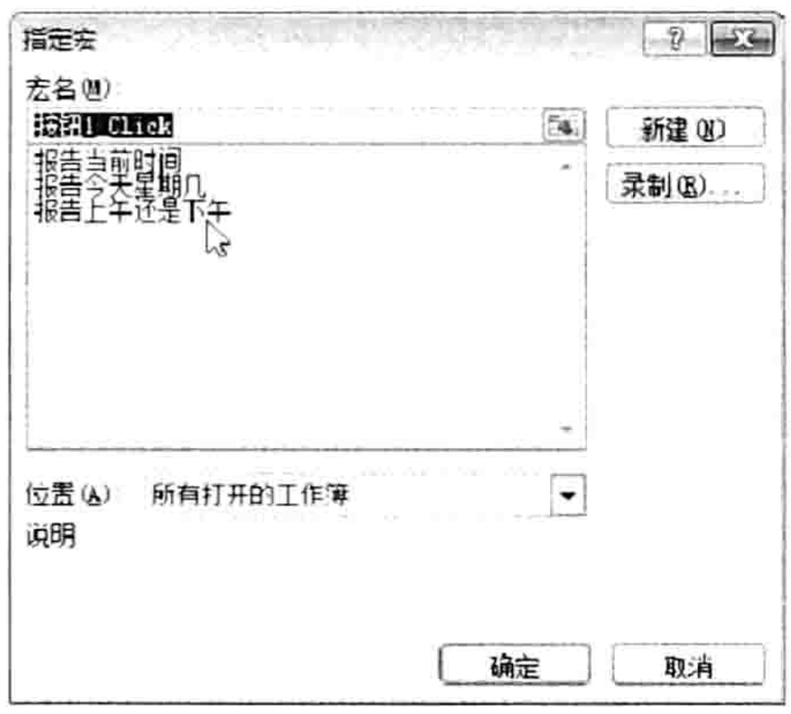


图 2.18　为命令按钮指定宏

step 5　单击第三步创建的按钮，程序会弹出今天是星期几的信息框，这表示通过按钮调用过程已经成功。

2.3.5　菜单

通过自定义菜单调用子过程是最标准的做法，这样既方便调用又显得专业。不过设计菜单属于 VBA 的高级知识，本书将它放在最后 4 章讲述，此节暂且略过。

2.4　如何保存代码

当工作簿中有 VBA 代码时，保存工作簿将有诸多讲究与规则，否则可能会令代码丢失。

本节讲述工作簿格式与保存代码之间的关系。

2.4.1　工作簿格式

自从 Excel 从 Excel 2003 版升级以后，工作簿的常用格式不再局限于 xls 一种，而有 xls、xlsm 和 xlsx 三种格式。这三种常用格式的区别如下。

1. 兼容格式：xls

从 Excel 2007 开始，微软公司推出了新的文件格式，但是为了保持文件的兼容性仍然允许使用老版本的 xls 格式，因此在高版本的 Excel 中也把 xls 称之为兼容格式。

将文件保存为兼容格式，文件发送给低版本的用户后也可以正常打开，缺点是会丢失高版本的某些特性，例如65 536行以上的行数、256列以上的列数、超过3项以上的条件格式、超过7层的IF函数嵌套公式等。

2. 压缩格式：xlsx

从Excel 2007开始，微软推出了压缩格式xlsx，将工作簿保存为压缩格式后可以应用更多的新功能，例如支持1 048 576行、16 384列、64个条件格式等，而且将xls格式的文件另存为xlsx格式后其体积会大大缩小，基于以上优势，微软公司将xlsx格式设置为Excel 2007、2010和2013的默认格式。

3. 启用宏的压缩格式：xlsm

在推出xlsx的同时，微软也推出了xlsm格式，它既拥有xlsx格式的一切优势又能弥补xlsx格式的不足——xlsx格式不能保存宏代码，而xlsm格式可以保存宏代码。

2.4.2 解决代码丢失问题

初学者在使用VBA时可能会常遇到此类问题——向工作簿中写入代码，当重启工作簿后发现代码不见了。这其实是由于Excel的文件格式引起的。

根据2.4.1节的分析，Excel 2007、2010和2013的默认格式是xlsx格式，而xlsx格式不能保存宏代码，因此保存工作簿后VBE中的模块和模块中的代码都会自动丢失。

其实在保存文件时，如果工作簿中存在VBA代码、模块或者宏表函数，Excel会弹出如图2.19所示的提示信息。

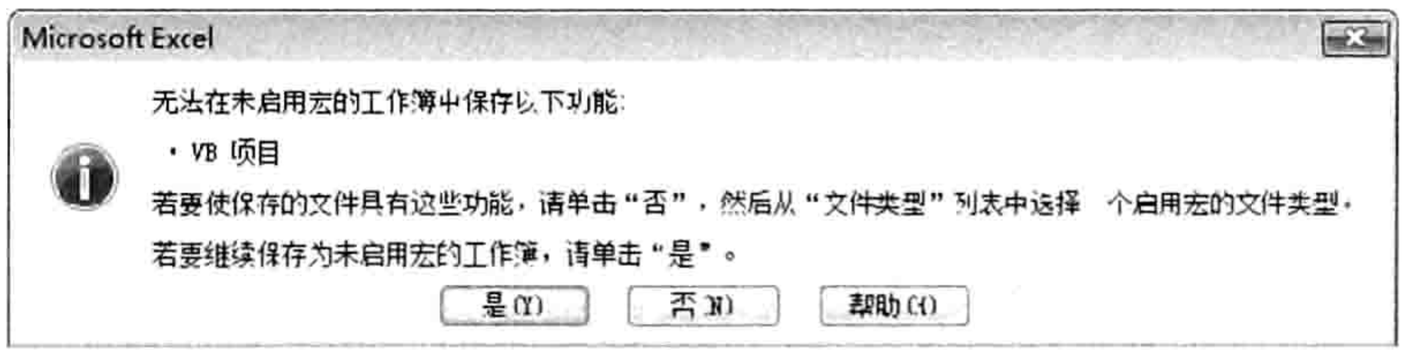


图2.19 不能保存宏代码时的提示信息

当产生如图2.19所示的提示时，应该单击“否”按钮，然后重新选择文件格式。如果需要将文件发送给其他人查看，而且不确定他人使用的是高版本还是低版本的Excel，那么应改用xls格式，否则应改用xlsm格式，从而支持高版本的新特性的同时又缩减文件的体积。

2.4.3 显示文件扩展名

Windows系统的默认设置会隐藏文件的扩展名，因此在保存文件时无法看到文件的扩展名。按以下步骤操作可以让文件扩展名显示出来。

step 1 按<Win+E>组合键打开“计算机”窗口（在Windows XP系统中也称“我的电脑”窗口）。

step 2 单击菜单栏中的“工具”→“文件夹选项”命令，打开“文件夹选项”对话框。

step 3 进入“查看”选项卡，将“隐藏已知文件类型的扩展名”前面的钩去掉，如图2.20所示，然后单击“确定”按钮。

step 4 打开Excel，按<Ctrl+S>组合键保存工作簿，在“另存为”对话框中单击保存类型，在其列表中会罗列出Excel所支持的所有文件格式，其中包含了每种格式的扩展名称，效果如图2.21所示。

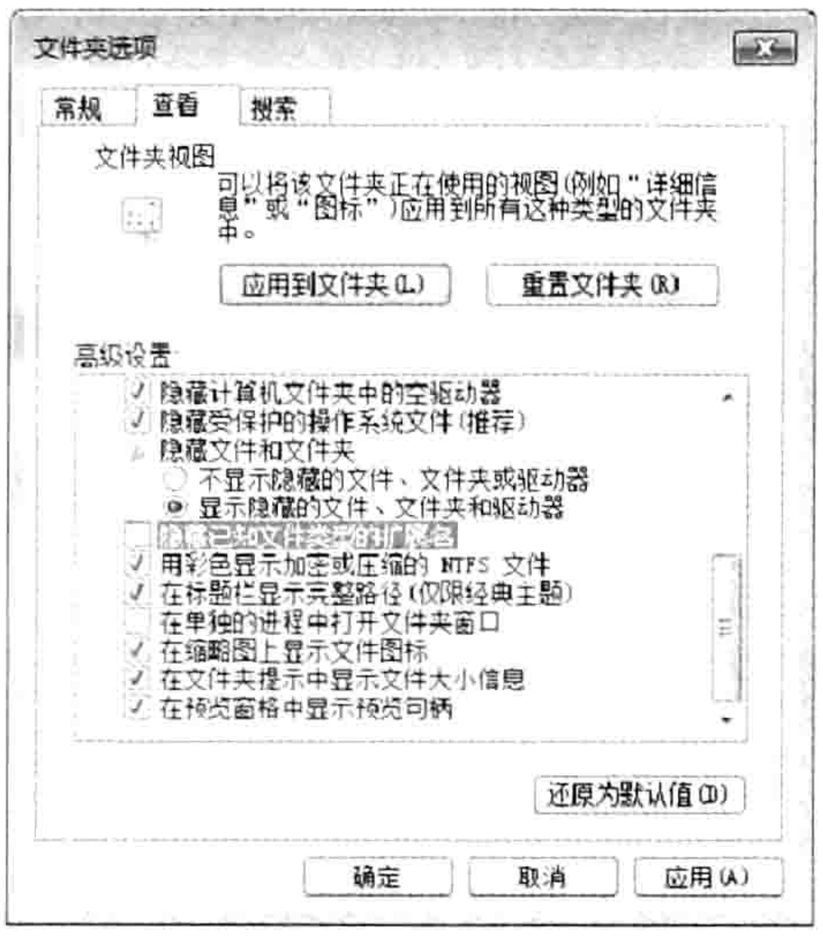


图 2.20　设置文件夹选项

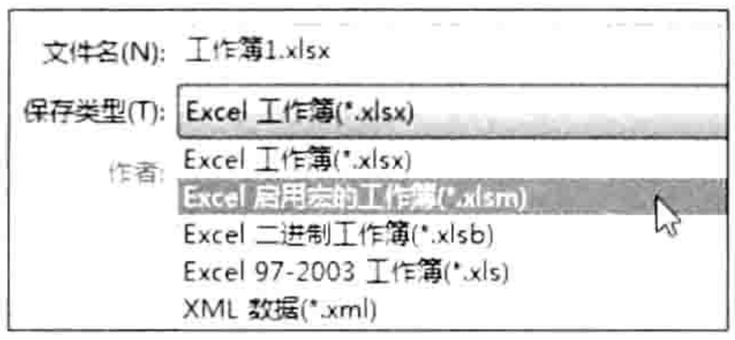


图 2.21　保存文件时的保存类型列表

当按以上方式调整后，保存文件时可以清晰地看到当前文件的格式，避免选错类型。

事实上还有一个更为快捷的办法确保代码不会丢失，方法是进入“Excel 选项”对话框的“保存”选项卡中，将文件的默认保存类型由 xlsx 格式切换成 xlsm 格式。设置界面如图 2.22 所示。

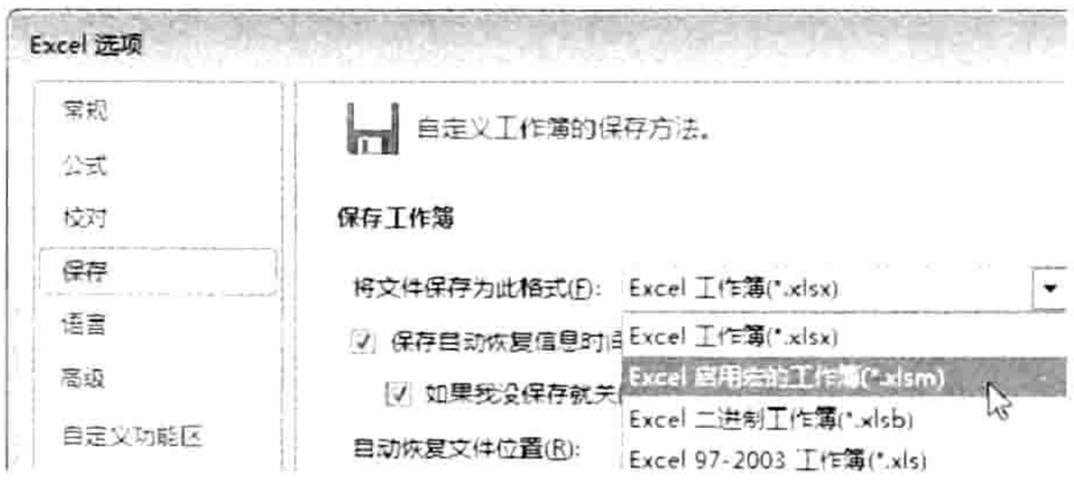


图 2.22　设置 Excel 的默认保存格式

2.5　如何放行代码

Excel 在默认状态下是禁止代码运行的，从而避免感染宏病毒。然而此方法会将正常的 VBA 代码也一并禁止，从而影响工作。因此有必要通过设置让 Excel 既避免感染宏病毒又能放行用户所指定的代码。

VBA 代码可以批量执行命令，也可以实现很多 Excel 原本无法实现的功能，它的强大功能与灵活性让它在 20 世纪 90 年代得以迅猛发展。然而正因为宏过于强大，使用宏代码开发的病毒也日益猖獗，同样是在 20 世纪 90 年代，由于宏病毒泛滥给 Office 用户带来了比较大的破坏。

微软为了阻止宏病毒发展，现在所有版本的 Office 都默认禁止运行宏，如果工作簿中存在宏代码，打开工作簿时将会产生如图 2.23 所示的提示信息。在此状态下，一切 VBA 代码都无法运行，只有单击“启用内容”按钮后活动工作簿中的 VBA 代码才能正常执行。

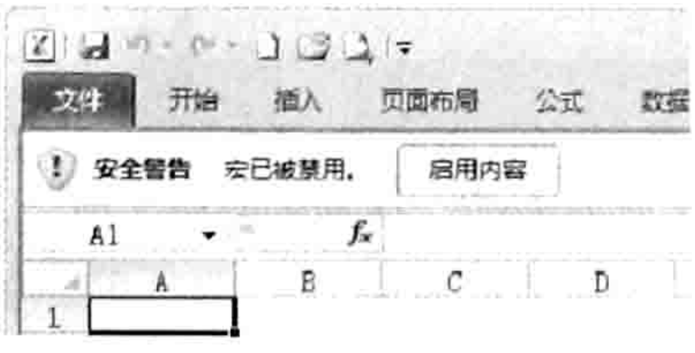


图 2.23　禁用宏的提示

不过，每打开一次工作簿都得单击“启用内容”按钮显然很费事。使用以下两种方式可一劳永逸。

1. 调整“宏设置”

在“宏设置”中启用宏可以让任何工作簿的宏代码都畅通无阻，具体步骤请按图 2.24 中的箭头编号操作。

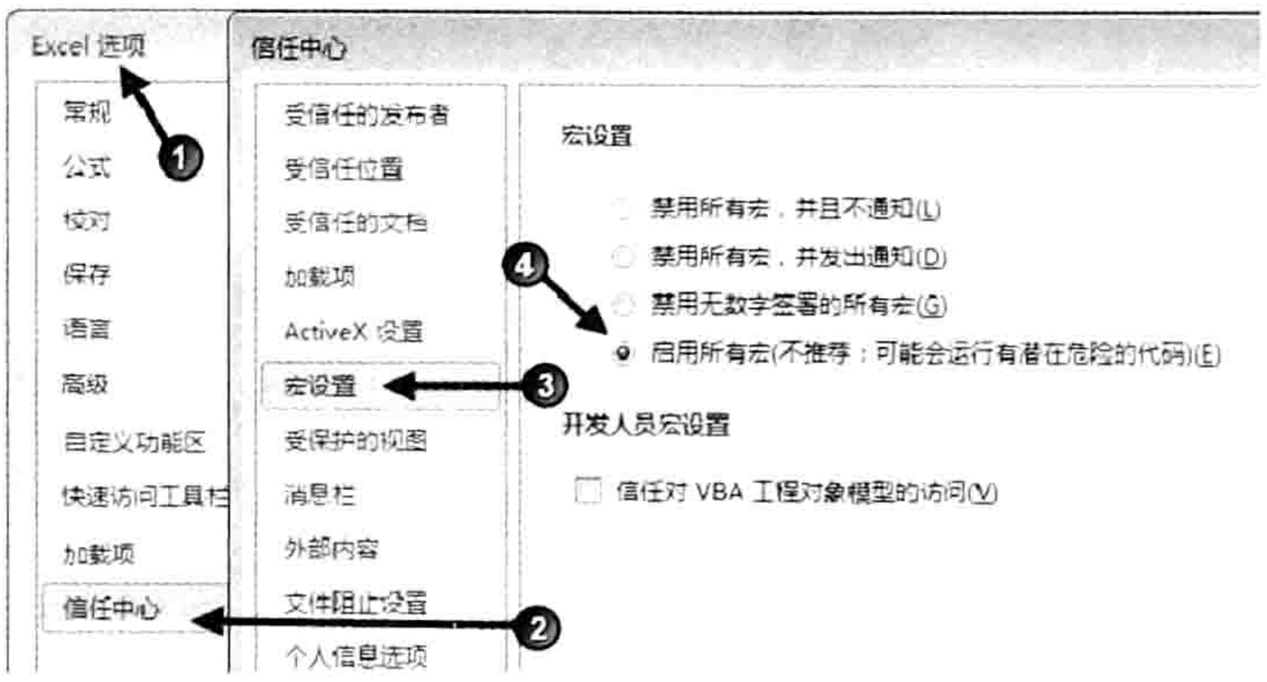


图 2.24　启用宏

2. 添加受信任位置

前面介绍的解决方案设置简便又有效，然而可能会带来隐患，让你的计算机感染宏病毒（尽管这种可能性极小）。

既放行代码又避免中毒的唯一可行的办法是设置受信任位置。

所谓的受信任位置是指用户指定的已确认其代码安全的一个文件夹，在此文件夹中的所有文件中的代码都被允许运行，而其他路径下的文件是否执行代码则视“宏设置”而定。

下面以设置“D:\生产表”为受信任位置为例，介绍具体操作的步骤。

step 1　通过如图 2.24 所示的方式将宏设置还原为“禁用所有宏，并发出通知”。

step 2　按如图 2.25 所示的方式将“D:\生产表”添加到受信任列表中。

其中最后一步勾选“同时信任此位置的子文件夹”复选项表示“D:\生产表”文件夹下的所有文件夹中的文件也受信任，如果取消勾选则只信任“D:\生产表”文件夹下的文件。

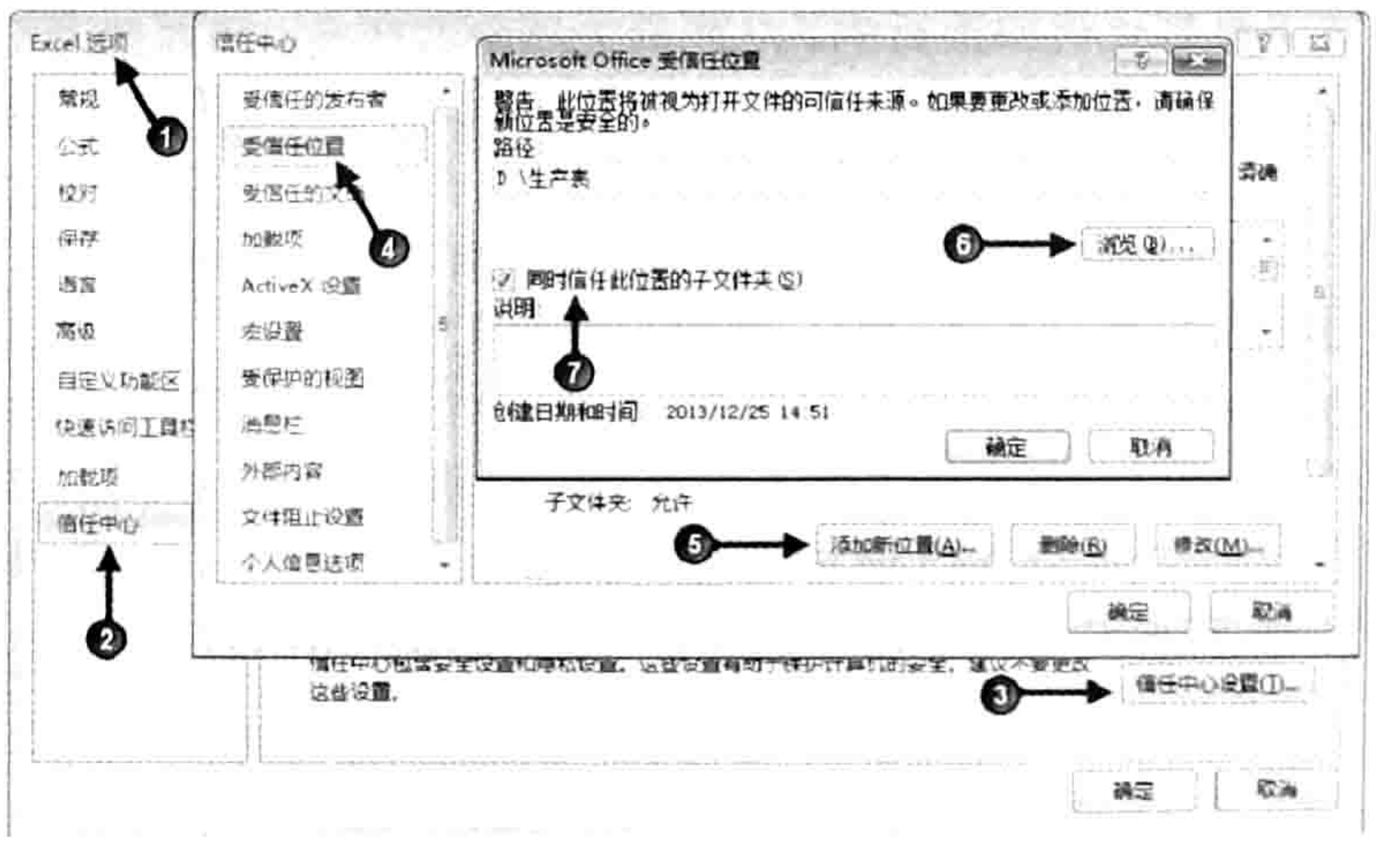


图 2.25　设置受信任位置

按以上方式设置后，可以将自己的文件或者自己信任的人发来的文件放置在“D:\生产表”路径下，打开此路径下的文件时代码运行畅通无阻，而打开其他位置的文件时，如果文件中包含了 VBA 代码或者模块，会弹出如图 2.23 所示的提示信息，从而避免意外中毒。

2.6 如何查询代码帮助

Excel 2010 自带相当完善的 VBA 帮助，可以通过帮助系统找到所有 VBA 相关的概念解释，以及代码含义释疑。Excel 2013 不含本地帮助，只有网络帮助，而且网络帮助也不够全面，其系统性和方便性不及 Excel 2010。建议初学 VBA 者应使用 Excel 2010，而非 Excel 2013。

2.6.1 调用帮助系统

Excel 的帮助和 Excel VBA 的帮助是不一样的，调用方式也不相同。查询 Excel 的帮助方法是在 Excel 工作表界面中按<F1>键，然后输入要查询的关键字；而查询 Excel VBA 的帮助需要在 VBE 界面中按<F1>键打开帮助系统，然后再输入关键字。

在查询 VBA 的代码含义时要掌握以下两个原则。

1. 只查询关键字而不是查询整行代码

在查询 VBA 代码时，应只查询关键字而非整行代码，和百度搜索的原理一致。假设有以下代码需要查询其含义：

```
Range("a1").AddComment "VBA 很神奇"
```

其中 Range 是关键字，AddComment 也是关键字。参数不能算关键字，“"a1"”是 Range 的参数，“"VBA 很神奇"”是 AddComment 的参数，因此代码中只有两个关键字。

Range 的正确查询步骤如下。

step 1 在 VBE 界面中按<F1>键打开帮助系统。

step 2 在查询框中输入关键字“Range”，然后按 Enter 键，Excel 会罗列出与当前关键字相关的所有信息，如图 2.26 所示。

查询帮助时往往有多条结果与关键字相关，对 VBA 熟悉者可以快速识别哪一条才是自己实际需要的目标，单击即可查看详细解释。VBA 初学者可以逐条查看，直到看懂为止。

图 2.27 所示的是单击图 2.26 中第三条搜索结果后的帮助信息，可以看出该窗口中包含了“Range.Range 属性”相关的功能说明、语法、参数和示例。从帮助中可以看出 Range 是一个对象，代表单元格或者区域。

图 2.26 查询 Range 的帮助

图 2.27 Range("a1")的语法、参数和示例

step 3 继续输入关键字“AddComment”，然后按 Enert 键，Excel 会罗列出与关键字相关的所有信息。单击第一条“Range.AddComment 方法”打开详细页面，从中可看到“Range.AddComment 方法”的含义、语法、参数和示例。根据帮助中的解说可以明白“Range.AddComment 方法”的作用是向单元格中插入一个批注。

2.6.2 为什么查看不了帮助

对于学习 VBA 而言，帮助极为重要。如果读者安装的 Office 属于精简版，那么必定无法查询 VBA 帮助，甚至可能连 VBA 都不支持。精简版 Office 首先就是去除帮助，然后是去除 Access、OneNote 之类不常用组件。如果还需要继续精简，那么就需要删除 VBA 组件。

为了确保 VBA 和 VBA 帮助可用，应尽量安装完整版的 Office 2010。完整版的安装包大小一般在 500MB 以上。

◆ ◆ ◆

第 3 章　VBA 的程序结构分析

在进入实质性的 VBA 开发之前，有必要了解 VBA 的程序结构。
VBA 程序包含子过程、函数过程和属性过程，本章仅以子过程为例阐述过程的结构。

3.1　子过程的结构

初学 VBA 者通常都是从录制宏开始的，录制宏产生的代码其实就是一个子过程。一个完整的子过程由作用范围、变量的生命周期、过程声明语句、参数、命令、中断过程语句、结束语句和注释 8 个部分组成，而录制宏时产生的代码不可能包含这 8 个组成部分，因此如果仅仅停留在录制宏阶段，是无法了解子过程的完整结构的。

3.1.1　认识程序结构

子过程包含作用范围、变量的生命周期、过程声明语句、参数、命令、中断过程语句、结束语句和注释 8 个部分。其中除注释以外的 7 个部分可以在帮助中找到，在帮助中搜索关键字“Sub 语句”即可得到子过程的结构。

子过程的结构如图 3.1 所示。

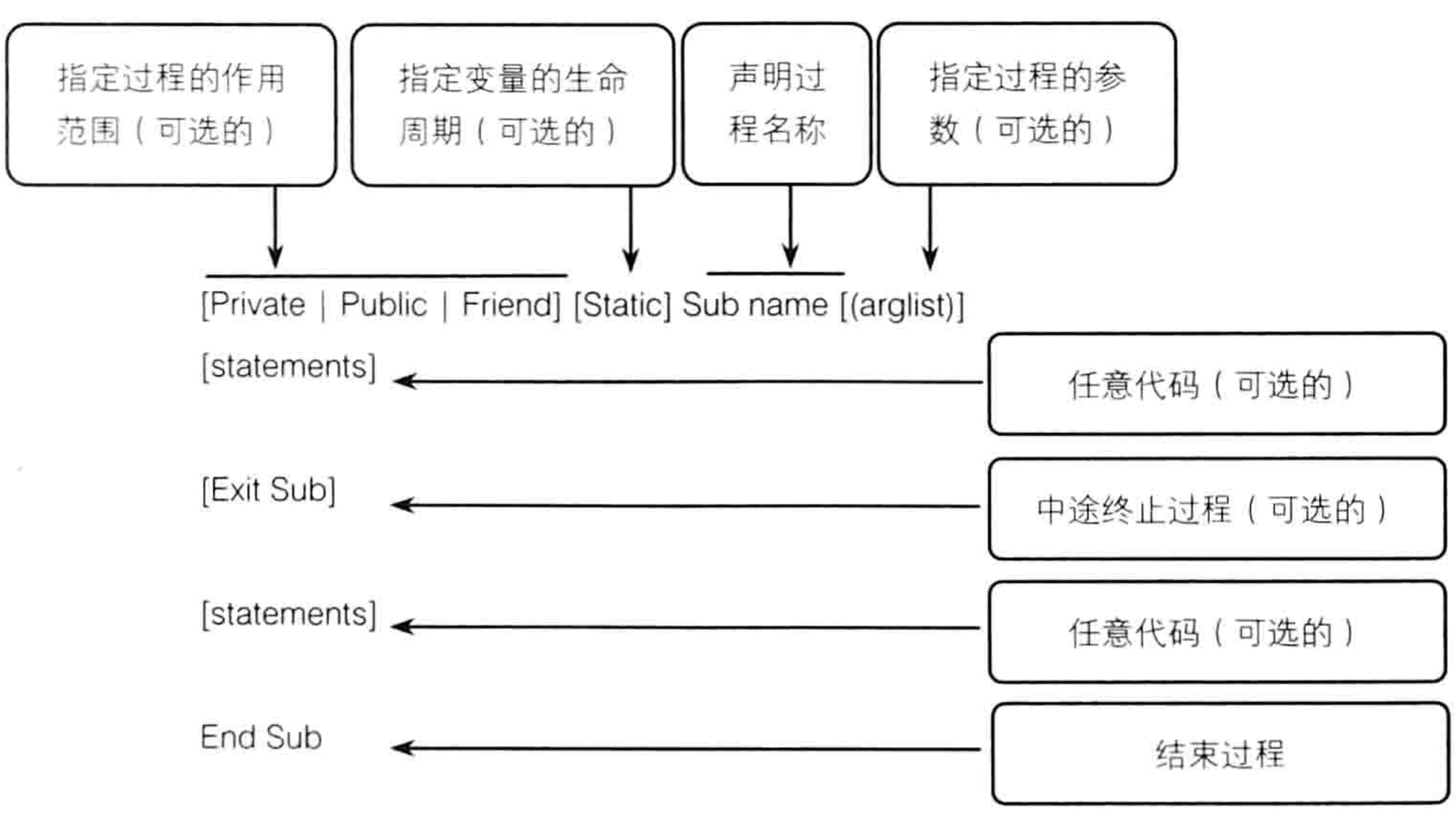


图 3.1

图 3.1 中带有方括号“[]”的部分表示是可选项，在编程时允许忽略。例如用于中断过程的代码“Exit Sub”就是可选项，程序执行过程中可以有条件地中断过程，也可以不中断过程；过程的参数“arglist”也是可选项，可以根据需求决定是否使用参数。

图 3.1 中带有“|”的部分表示是并列项，例如 Private、Public 与 Friend 之间属于并列关系，三者不能同时存在，只能根据需求任选一种。同时又由于它是可选项，因此三者都可以忽略。

剔除可选项后，子过程的简写模式如下：

```
Sub name()
```

```
End sub
```

在实际工作中，用得最多的都是简写结构的过程。

知识补充：程序的声明语句和结束语句组成程序的外壳。程序外壳是必不可少的组成部分，通常说一段程序用了多少行代码时都会忽略程序外壳。

以下过程“对 A1 赋值”忽略了作用范围、变量的生命周期、中断过程语句等部分，这是最常见的代码书写方式。

```
Sub 对A1赋值()'代码存放位置：模块中
Range("a1") = 123
End Sub
```

再如以下过程“修改工作表名称”，可将活动工作表命名为今日日期：

```
Sub 修改工作表名称()'代码存放位置：模块中
ActiveSheet.Name = Format(Date, "yyyy年mm月dd日")
End Sub
```

除此之外，过程还包括代码注释，3.1.2 节专门讲解与注释相关的知识。

3.1.2 为 VBA 程序添加注释

为程序添加注释有两个目的。其一是说明程序的功能、版号、作者、更新时间、更新内容之类，也可以是程序的思路说明；其二是为某一行代码添加代码的含义说明。

代码注释默认显示为绿色，代码颜色为黑色，两者形成反差，从而便于识别。

添加注释有两种方法。其一是先写“Rem”加一个空格，然后写入注释内容，VBA 在执行代码时只要检测到“Rem ”就自动忽略该行；其二是先写一个半角状态的单引号“'”，然后再追加注释内容。

例如在图 3.2 中，分别使用了两种方式为程序添加注释，在注释中既有程序的功能说明，也有每句代码的含义注释，从而给他人查看代码提供方便。

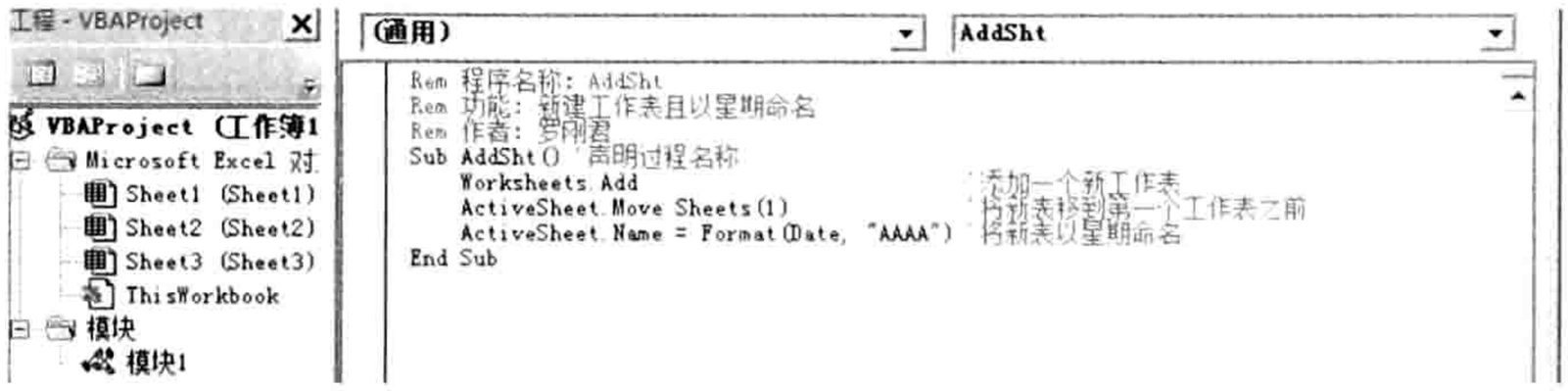


图 3.2 为 VBA 程序添加注释

代码注释不是必需的，但是很有必要完善代码的注释，这样既给他人提供方便，也给自己维护代码提供方便。因为 VBA 开发者都不是专业程序员，当几个月不用 VBA 后再回来查看代码可能会连自己写的代码也无法理清思路，鉴于此，代码的注释显得格外重要。

本例文件参见光盘：..\第三章\3-1 为过程添加注释.xlsm

3.2 子过程的作用范围

在调用程序时，除了通过菜单、快捷键、命令按钮、<Alt+F8>组合键等方式调用，还可以用

另一个过程来调用当前过程，而用过程调用过程时就涉及过程的作用范围。

本节阐述过程的作用范围，也称之为作用域。

3.2.1 何谓作用范围

过程的作用范围是指一个过程有多大的被调用权限，是只能被当前模块中的其他过程调用还是可以被其他模块中的过程调用。对于只有当前模块才能调用的过程称之为私有过程，使用 Private 关键字声明过程即可；所有模块都能调用的过程则称之为公有过程，使用 Public 关键字声明过程。由于 Public 属于默认值，因此声明公有的过程时可以忽略不写。

3.2.2 公有过程与私有过程的区别

公有过程与私有过程的区别主要体现在两方面。

1. 是否允许其他模块调用

私有过程只允许当前模块的其他过程调用，而公有过程允许所有模块的过程调用。通过以下步骤可以了解公有过程与私有过程的区别。

step 1 如果当前的 Excel 处于打开状态，那么重启 Excel。

step 2 在“工作簿 1”中按<Alt+F11>组合键进入 VBE 界面。

step 3 单击菜单中的“插入”→“模块”命令，并在产生的“模块 1”中录入以下 3 段子过程：

```
Rem 声明一个私有的过程“问候 1”
Rem 此过程只能当前模块才能调用
Private Sub 问候 1()
 MsgBox "早上好"
End Sub

Rem 声明一个公有的过程“问候 2”
Rem 此过程可被所有模块都调用
Sub 问候 2()
 MsgBox "下午好"
End Sub

Sub test()'调用过程“问候 1”和“问候 2”
 Call 问候 1
 Call 问候 2
End Sub
```

step 4 将光标定位于过程“test”中的任意位置，然后按<F5>键执行过程，程序会依次弹出“早上好”和“下午好”两个信息框，说明“问候 1”和“问候 2”都被调用成功。

step 5 单击菜单中的“插入”→“模块”命令，并在产生的“模块 2”中录入以下过程：

```
Sub test()'调用过程“问候 1”和“问候 2”
 Call 问候 1
 Call 问候 2
End Sub
```

step 6 将光标定位于过程“test”中的任意位置，然后按<F5>键执行过程，程序会弹出如图 3.3 所示的错误提示窗口，说明过程“问候 1”无法被调用。

图 3.3　出错提示

step 7　删除代码“Call 问候 1”，再次执行过程“Test”，程序会弹出“下午好”的提示信息。

知识补充：使用代码调用其他过程时，可使用 Call 语句，将被调用的过程名称作为它的参数即可，参数不能使用双引号，例如写成“Call 问候 1”。事实上在 Call 语句中允许忽略关键字“Call”，因此调用过程“问候 1”也可以直接写为“问候 1”。

2. 是否出现在“宏”对话框中

公有过程的名称会罗列在“宏”对话框中，而私有过程则处于隐藏状态，虽然也可以在工作表界面中通过按<Alt+F8>组合键来调用私有过程，但是私有过程的名称不会出现在该列表中。

在实际工作中极少使用私有过程，因此直接忽略 Private 和 Public 即可。

CD-ROM　本例文件参见光盘：..\第三章\ 3-2 公有过程与私有过程.xlsm

3.3　过程的命名规则

对过程命名需要遵循以下规则。

（1）第一个字符必须使用英文字母或者汉字，禁止用数字开头。

（2）不能使用空格、句号、感叹号、 @、&、$、，、、# 等标点符号。

（3）名称的长度不能超过 255 个字符。

（4）过程名称不宜与模块同名，否则跨模块调用该过程时必须同时书写模块名称和过程名称，当采用与其他过程一样的调用方式时无法调用成功。

（5）不能与同一个模块中的其他任意过程同名，但允许与不同模块中的过程同名。

（6）不能与 VBA 的保留字一致。例如 Dim、Sub、End、as 和 Exit 等都是保留字。

此外，过程名称不区分字母大小写，命名为 A 时，也可以用代码“Call a”来调用。

3.4　过程的参数

过程允许携带参数，有参数的过程比无参数的过程更强大、灵活。在编写比较大型的程序时会用到带参数的过程。

不过带有参数的过程在编写和应用上都比较复杂，本书在第 11 章中将它专门放到一章中来详解，因此本节暂且略过，待读者掌握 VBA 的基础知识后再做研究。

3.5　过程的执行流程

一个复杂的过程可能会有几百行代码。在执行程序时遵循从上到下、从左到右的顺序。不过

VBA也提供了改变程序流程的诸多方法，可让程序随时更改执行顺序。

3.5.1 正常的执行流程

VBA的程序执行流程是自上而下、从左到右的。

```
Sub 问候()'代码存放位置：模块中
  MsgBox "早上好"
  MsgBox "中午好"
  MsgBox "下午好": MsgBox "傍晚好"
  MsgBox "晚上好"
End Sub
```

在以上过程中，总共5句代码，其中第3句和第4句并列在一行中，使用了冒号作为分隔符。此处冒号的作用将两行代码显示在一行中，代码仍然算作两句。

当选择模块中的过程“问候”，然后按<F5>键执行过程时，Excel会依次弹出“早上好”、“中午好”、“下午好”、“傍晚好”和“晚上好”。

也可以使用<F8>键查看代码的执行流程。

选择过程“问候”，然后按<F8>键，可以看到VBA将声明语句“Sub 问候()”标示为黄色，表示它是即将执行的语句。再次按<F8>键，VBA将“MsgBox "早上好"”标示为黄色，表示“MsgBox "早上好"”是即将执行的语句……

3.5.2 改变程序的执行流程

VBA提供了条件语句和Goto语句来改变程序的执行流程。

以下两个过程使用了条件语句和GoTo语句，程序不再是从上到下执行，读者可以按<F8>键逐步执行代码，从而了解它的实际执行顺序。

```
Sub test()'代码存放位置：模块中
    a = 61                          '对变量A赋值为61
    If a < 60 Then                  '如果变量A的值小于60
        MsgBox "不及格"             '提示“不及格”
    Else                            '否则
        MsgBox "及格"               '提示“及格”
    End If                          '结束条件语句
End Sub
Sub test2()'代码存放位置：模块中
    On Error GoTo line              '当程序出错时，跳转到line处执行代码
    MsgBox 100 / Range("a1")        '报告100除以A10单元格的值
    Exit Sub                        '结束程序
line:                               '设置一个标签
    Range("a1") = 10                '将A1单元格的值赋值为10
    Resume                          '返回出错的地方继续执行
End Sub
```

本例文件参见光盘：..\第三章\3-3 改变程序的执行流程.xlsm

关于条件语句If Then和GoTo语句会在本书第7章有详解，此处仅需了解通过代码可以改变执行流程即可。

3.6 中断过程

执行过程时，VBA 会按预设的顺序连续地执行代码，直到遇到暂停语句或者终止过程的代码。VBA 提供了几个可以中断过程或者终止过程的代码，本节将一一解析这些语句。

3.6.1 结束过程：End Sub

End Sub 属于程序外壳的组成部分，它标志着程序结束，处于过程的末端。当 VBA 遇到此句代码时会自动结束过程。

一个 VBA 过程中只能存在一个 End Sub 语句。

3.6.2 中途结束过程：Exit sub

Exit Sub 语句用于中途结束过程，可以在一个过程中存放任意数量的 Exit Sub 语句，不过在执行代码时只要遇到第一个 Exit Sub 语句就会结束过程，完全忽略其他的 Exit Sub 语句。

在测试一个有 200 行代码的过程时，假设只需要测试前 100 行，忽略后 100 行，可在第 100 行代码之后插入 Exit sub 语句，那么后 100 行代码不再执行。

```
Sub 问候()'代码存放位置：模块中
  MsgBox "早上好"      '第一次提示
  MsgBox "中午好"      '第二次提示
  Exit Sub             '中途结束过程
  MsgBox "下午好"      '第三次提示
End Sub
```

以上过程中使用了 3 个 MsgBox 函数，用于弹出 3 个信息框，然而在第 3 句代码之前插入 Exit sub 语句之后，原本的第 3 句代码就不再执行，读者可以按<F8>键测试。

3.6.3 中途结束一切：End

End 语句具有 Exit Sub 语句的功能，可以中途结束过程。不过它比 End Sub 语句的功能强大得多，它不仅可以结束过程，甚至可以结束一切，包括清除所有公共变量的值，关闭当前窗体（假设 End 语句在窗体中）。

基于以上原因，End 语句不能随意使用，它可能会带来破坏性的后果，致使其他程序无法正常运行。

3.6.4 暂停过程：Stop

Stop 语句用于中途暂停过程，当程序执行过程中遇到 Stop 语句时会自动暂停，再次按<F5>键时则继续执行后面的代码。一个过程中允许存放多个 Stop 语句。

在调试代码时，Stop 语句是比较有用的工具。

```
Sub 问候 2()'代码存放位置：模块中
  MsgBox "早上好"      '第一次提示
  MsgBox "中午好"      '第二次提示
  Stop                 '中途暂停过程
  MsgBox "下午好"      '第三次提示
End Sub
```

以上过程中使用了 Stop 语句，因此按<F5>键后只执行前两句代码，遇到 Stop 后会就暂停。当再次按<F5>键时才会继续执行完剩下的代码。

3.6.5 手动暂停程序：Ctrl+Break

除了前面所讲的通过代码结束或者暂停过程以外，Excel 允许通过<Ctrl+Break>组合键暂停过程，不过只能在程序刚开始运行不久时才管用。

假设某程序需要比较长的执行时间，想要中途停止过程时，可以按<Ctrl+Break>组合键让程序暂停。例如以下过程需要 10 秒钟以上才能执行完毕，在程序启动后的前几秒可以按<Ctrl+Break>组合键让程序暂停：

```
Sub 填充数值()'代码存放位置：模块中
    For i = 1 To 1000000      '从1到100万
       Cells(i, "A") = I      '对A列第i个单元格写入序号
    Next                      '下一个单元格
End Sub
```


本例文件参见光盘：..\第三章\ 3-4 结束或暂停过程.xlsm

◆ ◆ ◆

第 4 章　VBA 四大基本概念

本书的第 2 章和第 3 章介绍了 VBA 程序的入门知识，包括 VBA 程序代码的存放方式、产生方式、调用方式、保存方式、查询方式，以及 VBA 程序的基本结构等内容，对这些应用常识有所认识后，从本章开始正式接触 Excel VBA 编程。

本章主要向读者介绍 VBA 中的四大基本概念，此后的一切高级应用都是这四大概念的延伸，因此请读者务必认真学习本章的理论，切不可跳过本章直接进入后面的综合应用环节。

4.1　Excel 的对象

在 Excel VBA 的帮助中对于对象的定义是“代码和数据的组合，可将它看作单元，例如，控件、窗体或应用程序部件。每个对象由类来定义”。很显然，这个定义不够通俗、直白，无法从中明白对象究竟是什么。本节将向读者详解 Excel 对象的概念、对象的特性，以及展示 Excel 常用对象的名称与书写方式。

4.1.1　什么是对象

不管是简单还是复杂的故事都会有人物、事件、时间和地点等要素，其中人物是故事的核心，故事的推进、发展总是围绕人物而展开。

在 Excel VBA 编程过程中，也相应地有对象、属性、方法和事件等要素，其中对象是 VBA 的核心，一切操作皆以对象为基础。如果没有了对象，VBA 编程就失去了存在价值。

为了让读者更易于理解，笔者打算使用类比的方法来描述对象。尽管一切类比都会有疏漏之处，无法完整、恰到好处地传达目标信息，然而在无法明了官方定义时，类比可以作为有效的补充，让用户从中获得对概念的基本认知。

Excel 的对象就像一个物体，实实在在地呈现在每一个 Excel 用户眼前。物体是能看见的，或者能感受到它的存在。Excel 中的单元格、工作表、工作簿、窗口、批注、图表、艺术字、菜单等都是对象，一个对话框就是一个对象，Excel 本身也是一个对象。

事实上，Excel 中也有一些比较隐蔽、难以判断的对象。例如通过<Ctrl+F3>组合键定义的名称就是一个对象，一个筛选器、超级链接、页面设置、分页符也是一个对象。

操作 Excel，其实就是操作这些对象。

正确地理解对象，对于 VBA 编程有着深远的意义。

Excel 2010 有 261 类对象，不过常用的对象不超过 10 类，对于这些常用对象的名称与含义有必要熟记，如表 4-1 所示。

表 4-1　常见对象及其含义

对象类别	含　义
Application	代表整个 Excel 应用程序
Workbook	代表 Excel 工作簿对象
Worksheet	代表工作表对象
Window	代表窗口对象

续表

对象类别	含　义
Range	代表单元格对象
Shape	代表嵌入到工作表中的图形对象，包括自选图形、OLE 对象、图片、图表、艺术字、文本框、批注等
Name	代表名称对象，可以是内置名称也可以是自定义名称
Chart	代表图表对象
WorksheetFunction	代表工作表函数对象
Comment	代表单元格中的批注对象

本书随书光盘的附录中有Excel 2010的对象名称与含义大全，文件名称为“附录B　Excel 2010对象大全.pdf”，读者有兴趣的话可以查看该表。不过了解对象大全对于编程而言没有实际作用，只要了解几个常用的对象就足够了，当工作中需要使用代码操作这些对象时，录制宏即可获得代码，而不用将它们全记在大脑中。

4.1.2　对象与对象集合

Excel VBA将单一的对象和同类别的多个对象分别定义为对象与对象集合，对象集合通常以字母“s”结束，表示复数。例如：

Workbooks——工作簿集合，代表当前打开的所有工作簿，可能是一个也可能数百个。

Worksheets——工作表集合，代表工作簿中的所有工作表，可能是一个也可能数百个。

Comments——批注集合，代表工作表中的所有批注。

Cells——单元格集合，代表工作表中的所有单元格，有1 048 576×16 384个，如果使用兼容格式的工作簿，则只有65 536×256个。

Shapes——图形对象集合，代表插入到工作表中的一切图形对象。

不是所有对象都有对象集合，Excel应用程序对象Application就只有一个，不存在集合，工作表函数对象WorksheetFunction和字体对象Font也不存在集合，只能单个访问。

使用以下代码可以关闭当前已经打开的所有工作簿：

```
Sub 关闭所有工作簿()  '代码存放位置：模块中
  Workbooks.Close
End Sub
```

其中Close表示关闭，因此整句代码表示关闭所有工作簿。

```
Sub 计算工作表数量()'代码存放位置：模块中
  MsgBox Worksheets.Count
End Sub
```

其中Count表示数量，因此整句代码表示在信息框中显示工作表集合的总数量。

单个对象是指对象集合中的其中一个对象，通过参数来表示。

引用单个对象时有两种参数书写方式，包括使用序号和使用名称。

1. 使用序号引用单个对象

使用序号引用单个对象比较方便，书写时也比较简单，直接在对象集合的括号中加入一个序号即可。该序号不能小于1、不能大于对象集合的总数量。语法如下：

```
对象集合(序号)
```

例如 Worksheets(1)表示第一个工作表，参数“1”代表工作表的序号，按从左到右的顺序计算。在图 4.1 中展示了工作表序号的编号方式。

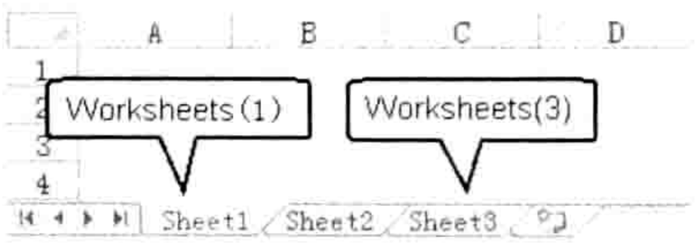


图 4.1　工作表序号示意图

图 4.1 中只有 3 个工作表，因此只能使用 Worksheets(1)、Worksheets(2)和 Worksheets(3)引用这 3 个工作表，使用 Worksheets(0)或者 Worksheets(4)都会出错。

同理，Comments(3)代表第 3 个批注，如果工作表中没有批注或者批注数量少于 3 个，那么执行代码时将会出错。

假设工作表中有超过 3 个批注，那么以下代码可以获得第 3 个批注的内容：

```
Sub 获得活动工作表第 3 个批注的内容() '代码存放位置：模块中
  MsgBox ActiveSheet.Comments(3).Text
End Sub
```

代码中 ActiveSheet 代表活动工作表，Comments(3)代表第 3 个批注，Text 代表批注的内容，因此整句代码表示在信息框中显示活动工作表第 3 个批注的内容。

本例文件参见光盘：..\第四章\ 4-1 获取第 3 个批注内容.xlsm

Cells 代表单元格集合，可以通过序号引用集合中的任意一个单元格。序号是先横向后纵向计算的，因此 Cells(8)代表横向第 8 个单元格，即 H1 单元格；Cells(16400)代表横向第 16 400 个单元格，即 P2 单元格。P2 的计算规则是：Excel 2010 的工作表具有 16 384 列，第 16 384 个单元格即第一列的最后一个单元格 XFD1，当序列号大于 16 384 时则从第二行开始继续累加。16 400 减去 16 384 等于 16，因此 16 400 个单元格即第二行的第 16 个单元格 P2。

2. 使用名称引用单个对象

使用名称引用单个对象比较直观，但是书写方式比使用序号更复杂。其语法如下：

```
对象集合（"对象名称"）
```

如果序号引用图 4.1 中的第二个工作表，应使用代码 Worksheets(2)，而使用名称引用图 4.1 中的第二个工作表则应采用 Worksheets("Sheet2")。具体应用如下：

```
Sub 删除 Sheet2()'代码存放位置：模块中
Worksheets("sheet2").Delete
End Sub
```

代码中 Delete 的含义是删除，因此整行代码的作用是删除名为“sheet2”的工作表。

对象名称需要使用双引号，忽略双引号时会执行出错；使用序号引用对象时则绝对不能使用双引号，否则 VBA 会将序号作为名称处理。例如 Worksheets(3)代表第 3 个工作表，而 Worksheets("3")则代表名称为 3 的工作表，两者含义大相径庭。

知识补充：单元格对象的写书方式比较特殊，它的对象集合是 Cells，可以通过序号作参数引用单个单元格对象，例如 Cells(5)，但是不能对 Cells 使用名称参数，例如“Cells ("B5")”。Excel 提供了 Range("地址")这种形式的单元格引用方式，从而所见即所得，可通过参数了解它所引用的单元格对象的地址，例如 Range ("A1")，Range ("b2:c10")，Range ("C:D")等。

4.1.3　对象的层次结构

Excel 有 200 多类对象（包含对象集合），这些对象不是并列存在的，而是像公司职员的组织架构一样具有一定的层次结构，一级级泾渭分明。

Excel 最高级别的对象即 Excel 应用程序本身，其对象名称为 Application。在它的下层有工作簿对象（类别名称为 Workbook）、窗口对象（类别名称为 Window）、对话框对象（类别名称为 Dialogs）、应用程序级别的名称对象（类别名称为 Name）、单元格对象（类别名称为 Range）、活动工作表对象（此对象的写书方式是 Activesheet）等。

工作簿对象处于 Excel 对象的第 2 个层级，它的下层有工作表对象（类别名称为 Worksheet）、工作簿级别的名称对象（类别名称为 Name）、样式对象（类别名称为 Style）、活动工作表（此对象的写书方式是 Activesheet）等。

工作表对象处于 Excel 对象的第 3 个层次，在它的下层有单元格对象（类别名称为 Range）、批注对象（类别名称为 Comment）、行对象（行的集合用 Rows 表示，可通过参数引用单个行）、列对象（列的集合用 Columns 表示，可通过参数引用单个列）、页面设置对象（类别名称为 PageSetup）、分页符对象（类别名称为 HpageBreak）、图形对象（类别名称为 Shape）、工作表及级的名称对象（类别名称为 Name）等。

行对象或者列对象处于 Excel 对象的第四层，在它们的下层都是 Range 对象。

对于几个常用的对象，可以按以下形式展示它们的层次关系：

Application 对象→Workbook 对象→Worksheet 对象→Rows/Columns 对象→Range 对象→Comment 对象。

有 3 种对象比较特殊，下面有必要特别说明。

（1）Name 对象

Name 对象是自定义名称，名称包括工作簿级名称和工作表级名称。工作表级名称处于工作表对象的下一层，可以通过工作表对象的属性访问工作表名称的内容。例如获取第一个工作表中的名为 A 的工作表名称可用代码“Worksheets(1).Names("A")”来实现。

工作簿级名称处于工作簿对象的下一层，不能通过工作表对象的属性来访问，需要通过工作簿对象的属性访问。例如“生产表.xlsm”中有一个工作簿级名称 B，那么可用代码“Workbooks("生产表.xlsm").Names("B")”获取该名称的值。

Excel VBA 为了简化引用方式，允许通过 Application 对象直接访问工作表级名称和工作簿级名称，因此 Application.Names("A")和 Application.Names("B")也能访问成功。

基于以上原因，Application 对象、Workbook 对象和 Worksheet 对象都有一个名为 Name 的下层对象，但是它们三者并不相同。

本例文件参见光盘：..\第四章\4-2 名称的三种访问方式.xlsm

（2）ActiveSheet 对象

ActiveSheet 对象代表活动工作表。工作簿对象 Workbook 和应用程序对象 Application 都有一个名为 Activesheet 的下级对象，然而它们两者是不同的。Application 的子对象 ActiveSheet 代表活动工作簿中的活动工作表，由于活动工作簿只有一个，因此 Application 的子对象 ActiveSheet 永远只有一个；一个 Workbook 对象虽然也只有一个 Activesheet 子对象，但是由于 Excel 允许同时打开多个工作簿，因此每一个工作簿都有一个名为 Activesheet 的子对象。

基于此，Application 的子对象 ActiveSheet 和 Workbook 的子对象 ActiveSheet 是不同的。代码“Application.ActiveSheet”永远只能引用活动工作簿中的活动工作表，不能引用非活动工作簿

中的活动工作表。换而言之，“Workbook.ActiveSheet”包含“Application.ActiveSheet”对象。

（3）Range 对象

Range 对象即单元格和区域，它既是工作表对象 Worksheet 的下层对象，同时又是 Application 对象的下层对象。当作为 Application 对象的下层对象时，Range 只代表活动工作表中的单元格；当作为工作表对象 Worksheet 的下层对象时，Range 可以代表任何工作表中的单元格，因此“Worksheet 对象.Range”大于“Application.Range”对象。

下面用具体的案例来演示以上差异。

step 1 新建一个工作簿，在 Sheet1 工作表的 A1 单元格中输入字符 A，在 Sheet2 工作表的 A1 单元格中输入字符 B，在 Sheet3 工作表的 A1 单元格中输入字符 C。

step 2 按<Alt+F11>组合键进入 VBE 窗口。

step 3 单击菜单中的“插入”→“模块”命令，然后在模块中录入以下两段代码：

```
'此过程可以获取 Sheet1、Sheet2 和 Sheet3 中 A1 单元格的值
Sub 通过 Worksheet 对象访问 Range 对象的值()
    MsgBox Worksheets("Sheet1").Cells(1)     '在信息框中显示 Sheet1 工作表中 A1 的值
    MsgBox Worksheets("Sheet2").Cells(1)     '在信息框中显示 Sheet2 工作表中 A1 的值
    MsgBox Worksheets("Sheet3").Cells(1)     '在信息框中显示 Sheet3 工作表中 A1 的值
End Sub
'此过程只能获取活动工作表中 A1 单元格的值
Sub 通过 Application 对象访问 Range 对象的值()
    MsgBox Application.Cells(1)              '在信息框中显示活动工作表中 A1 的值
End Sub
```

step 4 分别执行两个过程，第一个过程可以获取 3 个工作表中 A1 单元格的值，而第二个过程只能获取活动工作表中 A1 的值。这证实了 Range 对象分别作为 Application 对象和工作表对象的下层对象时是有区别的。

本例文件参见光盘：..\第四章\4-3 单元格对象的两种访问方式.xlsm

4.1.4 父对象与子对象

由于 Excel 的对象具有层次结构，因此就产生了父对象与子对象的说法。

Excel 将上一层对象称之为父对象，下一层对象称之为子对象。例如工作簿是工作表的父对象，工作表是单元格的父对象；工作表是工作簿的子对象，批注是单元格的子对象。

访问子对象的语法如下：

对象名称.子对象名称

要注意，对象与它的子对象之间使用半角状态下的圆点符号，切不可使用全角状态下的圆点符号。

例如以下代码都是标准的访问子对象的方法：

Worksheets(1).Cells (10)——访问第 1 个工作表的第 10 个单元格。

Workbooks(3).Worksheets ("生产表")——访问第 3 个工作簿中名为“生产表”的工作表。

Cells(2).Comment——访问第 2 个单元格中（B1 单元格）的批注。

访问父对象的语法如下：

```
对象名称.Parent
```

其中 Parent 代表父对象。以下代码可以访问父对象：

Cells(2).Parent——访问第 2 个工作表的父对象，也就是活动工作表。

Comments(2).Parent——访问第 2 个批注的父对象，也就是该批注所在单元格。

读者看到此处或许会产生一个疑问：为什么以上代码每一句都无法执行？例如以下两句代码中任意一句都会出错：

```
Sub test()'代码存放位置：模块中
  Worksheets(1).Cells (10)
  Workbooks(3).Worksheets ("生产表")
End Sub
```

对于 VBA 而言，一句完整的代码必定有一个动作，例如赋值、修改某个属性、打开或者关闭对象、声明一个变量、调用一个过程等。没有动作的代码都是不完整的，在执行过程中必定出错。

“Worksheets(1).Cells (10)”和“Workbooks(3).Worksheets ("生产表")”是两个对象，它们属于名词，不算动词，因此不构成一句完整的代码。

以下代码属于完整的代码：

Workbooks("生产表.xlsm").Close——关闭工作簿，其中 Close 表示关闭，是一个动作。

MsgBox Workbooks("生产表.xlsm").Name——提示工作簿对象的名称，提示即为动作。

Worksheets.Add——创建一个新工作表，创建即为动作。

Range("A1")=123——对单元格 A1 赋值，此处的等号表示赋值，也算是一个动作。

Worksheets(1).Name = "总表"——对第一个工作表重命名为“总表”，命名就是一个动作。

通过以上比较，读者应该足以判断怎样才算一句完整的代码。

4.1.5　活动对象

Excel 将当前处于激活状态的对象定义为活动对象，活动对象是可以直接访问的，不用指定名称。常用的活动对象包括活动工作表 Activesheet、活动单元格 ActiveCell 和活动图表 ActiveChart、活动窗口 ActiveWindow、活动工作簿 ActiveWorkbook。

1. Activesheet

图 4.2 中有 3 个工作表，其中 Sheet2 处于激活状态，因此只有 Sheet2 属于 Activesheet。

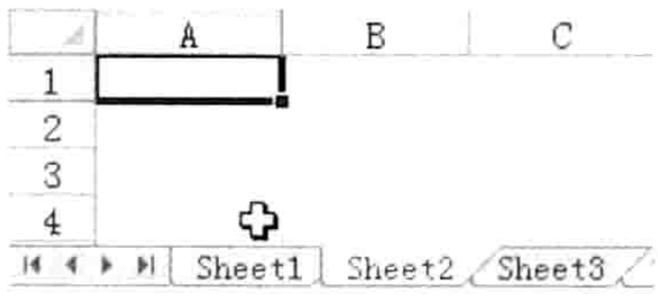


图 4.2　活动工作表

以下过程可以获得活动工作表的名称：

```
Sub 获取活动工作表的名称()'代码存放位置：模块中
 MsgBox ActiveSheet.Name
End Sub
```

如果需要获取第 3 个工作表的名称，那么可以先进入第 3 个工作表中再执行以上代码，也可以使用代码“MsgBox Worksheets(3).Name”。

一个工作簿中只有一个活动工作表。

将一个非活动工作表变成活动工作表应使用 Worksheet.Activate 方法，具体代码如下：

```
Worksheets("sheet3").Activate
```

引用活动工作表中的单元格时可以省略 Activesheet，例如对活动单元格的 A1 赋值为 123，不再使用代码“ActiveSheet.Cells(1) = 123”，而是简化为“Cells(1) = 123”。

2. ActiveWorkbook

活动工作簿由于处于激活状态，总在其他工作簿窗口的上层。不管打开了多少个工作簿，活动工作簿只有一个。活动工作簿的书写方式是 ActiveWorkbook。

当打开了一个新工作簿后，新工作簿就是活动工作簿。

引用活动工作簿中的工作表时，可以忽略活动工作簿。

3. ActiveCell

活动单元格是指处于激活状态的单元格，在活动单元格中能直接录入字符。当选择单个单元格时，被选择的单元格就是活动单元格；当选择了一个区域时，该区域中背景为白色的单元格即为活动单元格，也可以根据名称框来确定活动单元格，只有活动单元格的地址会显示在名称框中。在图 4.3 中选区是 B2:C4，只有 C4 单元格才是活动单元格。

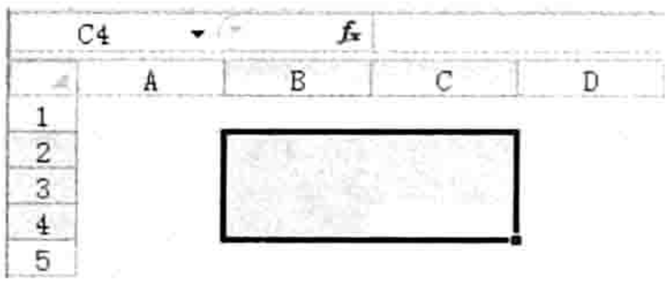
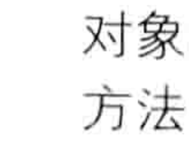


图 4.3　活动单元格（C4）示意图

引用活动单元格的子对象时不允许忽略 Activecell。例如以下过程表示向活动单元格中添加一个内容为“VBA 好犀利！”的批注，然后将批注内容输出到信息框中。过程中使用了两次 ActiveCell，两个 ActiveCell 都不允许被忽略：

```
Sub 批注() '代码存放位置：模块中
 ActiveCell.AddComment "VBA 好犀利！"'向活动单元格创建批注
 MsgBox ActiveCell.AddComment         '在信息框中显示活动单元格的批注内容
End Sub
```



本例文件参见光盘：..\第四章\4-4 活动对象的省略与否问题.xlsm

4.2 对象的方法和属性

每个对象都有方法和属性，掌握了某个对象的方法和属性才算真正掌握了这个对象。

本节主要从概念上区分对象的方法与属性，以及介绍如何查看方法与属性，对于具体的应用将在本书的第 9 章中提供大量的案例。

4.2.1 属性与方法的区别

对象的属性是指对象的某个特征，例如颜色、大小、地址、名称等。一切对象都有属性，而且有多个属性。

对象的方法是指处理对象的过程，通俗而言就是对对象执行某种操作。

方法都是动词，例如创建、删除、关闭、插入、激活、计算、复制、查找等。以下是 Range 对象的部分方法：

Range ("A1").Copy——复制 A1 单元格。
Range ("A1").Insert——在 A1 单元格上方插入单元格。
Range ("A1"). AutoFill——填充 A1 单元格。
以下是 Worksheet 对象的部分方法：
Worksheets("Sheet2").Activate——激活工作表 Sheet2。
Worksheets("Sheet2").Delete——删除工作表 Sheet2。
Worksheets("Sheet2").Move——移动工作表 Sheet2。

对象的属性属于名词，例如大小、地址、位置序号、名称等。以下是 Workbook 对象的部分属性：

Workbooks(2).Name——获取第 2 个工作簿的名称。
Workbooks("生产表.xlsm").FileFormat——获取工作簿“生产表.xlsm”的文件格式。
ActiveWorkbook.Password——获取活动工作簿的密码。

4.2.2　查询方法与属性的两种方法

尽管方法和属性通过词性就相当容易区分，但是 Excel 还是提供了两种简便的查询方法，既帮助用户区分方法与属性，又让用户快捷地找到每个对象的方法和属性的含义解释。

1. 查询帮助

Excel VBA 的帮助系统中罗列了一切对象的属性和方法，只要正确地输入关键字即可调用。下面以查询 Workbook 对象的方法与属性为例，介绍具体的操作步骤。

step 1　在 VBE 界面中单击菜单中的“帮助”→“Microsoft Visual Basic for Applications 帮助”命令。

step 2　在 VBA 帮助界面的查询窗口中输入“Workbook 对象成员”，然后按<Enter>键。

step 3　单击查询结果中的第一条“Workbook 对象成员”，此时会打开 Workbook 对象相关的方法、属性和事件列表，如图 4.4 所示。

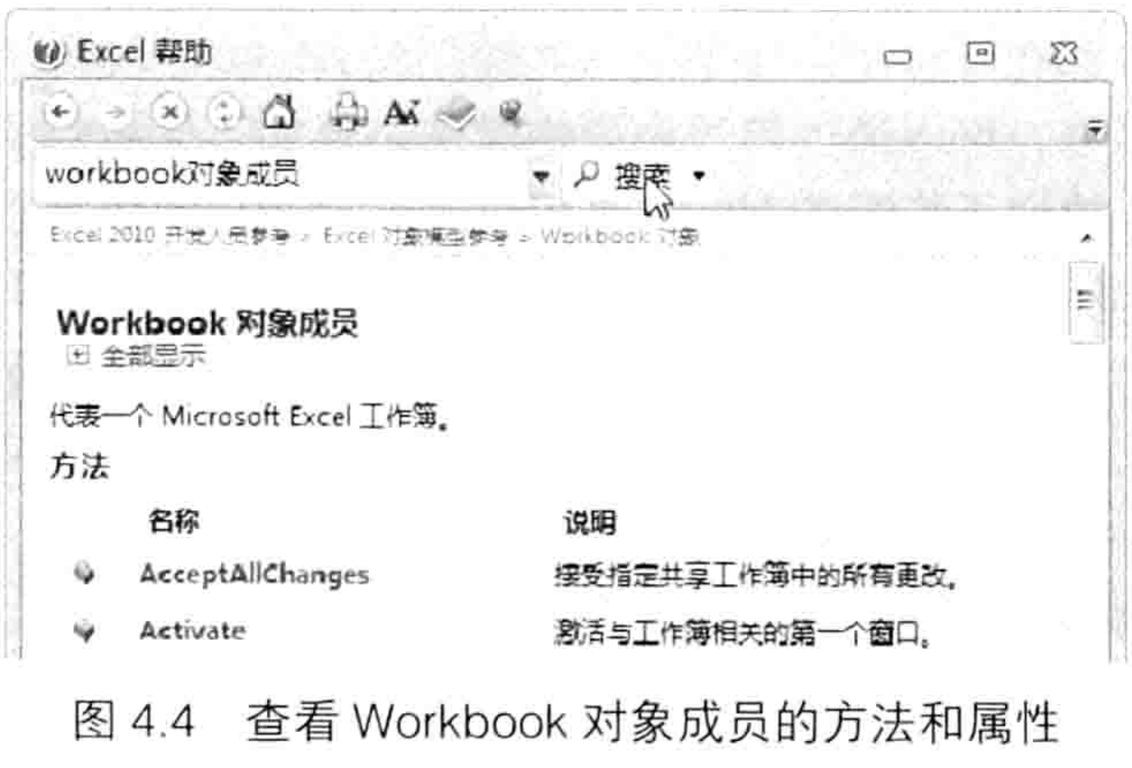


图 4.4　查看 Workbook 对象成员的方法和属性

2. 属性与方法列表

VBA 为用户提供了对象的属性与方法列表，录入代码时会自动弹出该列表，用户可以从列表中选择属性或者方法名称，从而既加快录入速度又确保代码的准确性。

下面以调用工作表对象的属性与方法为例，介绍具体的操作步骤。

step 1　使用代码“Dim a As Worksheet”声明一个 Worksheet 类型的对象变量。

step 2　再输入代码“a.”，此时会弹出与工作表相关的所有属性与方法列表，如图 4.5 所示。

Worksheet 是工作表的类别名称，因此变量 a 此时就代表一个工作表，输入“a.”之后可以调用工作表相关的一切方法与属性。在图 4.5 中带有绿色图标的是方法，带有手形黑色图标的是属性。

step 3 如果需要查看单元格对象的属性和方法，那么将变量 a 的类型改为 Range 即可，弹出的属性与方法列表如图 4.6 所示。

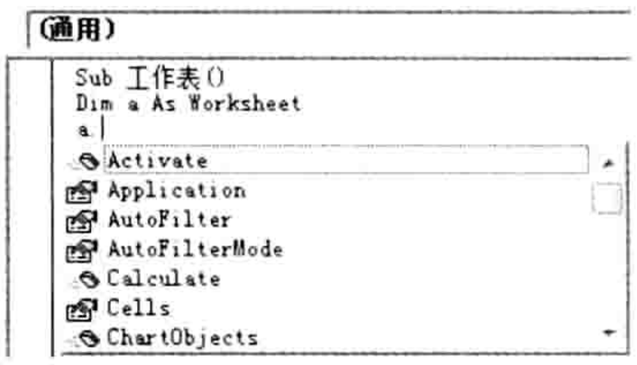


图 4.5　与工作表相关的方法与属性

图 4.6　单元格对象的方法与属性

关于定义变量的相关知识在本书的第 5 章会详述，此处知道调用属性与方法的步骤即可。

4.2.3　方法与属性的应用差异

1. 对象的属性

对象的属性包含只读属性和可读、可写属性，其中只读属性只能获取不能修改，而可读、可写属性则既可以获取该属性的值又可以根据需求修改属性。

例如 Workbook 对象的 FullName 属性就是一个只读属性。FullName 属性代表工作簿的路径，可以获取该属性值，但不能修改该值，因为工作簿的路径是不允许修改的（要注意，将文件另存到其他路径下，看似修改了路径，其实这是偷换了概念，因为原文件的路径仍然没有变化，变化的是多了一个文件，新文件的路径与原文件不同不代表修改了原文件的 FullName 属性值）。

获取活动工作簿的路径可使用以下代码：

```
Range("a1") = ActiveWorkbook.FullName '将活动工作簿的路径和名称保存在 A1 单元格中
```

FullName 属性包含了工作簿的路径和名称，如果工作簿未保存，那么只能获取到工作簿的名称，如果工作簿已保存，那么可将工作簿路径一并输出到 A1 单元格中。

对于可读、可写的属性，可以通过等号为属性赋值。例如工作表对象的 Name 属性是可读、可写的属性，因此既可以读取工作表的 Name 属性又可以随意修改该属性值。

以下代码分别获取了两个工作表的名称，也分别修改了两个工作表的名称：

```
Range("a1") = Worksheets(1).Name '将第一个工作表的名称输出到 A1 单元格
Range("b1") = Worksheets(2).Name '将第二个工作表的名称输出到 B1 单元格
Worksheets(1).Name = "分表" '将第一个工作表重命名为“分表”
Worksheets(1).Name = "总表" '将第二个工作表重命名为“总表”
```

如果对只读属性赋值会产生“不能给只读属性赋值”的错误提示。

2. 对象的方法

对象的属性是一个名词，其应用包含取值和修改值两种方式。对象的方法属于动词，其操作结果千变万化，远比属性的应用复杂。

所有对象的方法加起来有近万个，每一种方法的操作结果都是独一无二的，因此没法按规律将它们分类。不过常用的方法并不多，包含激活、删除、新建、关闭、打印、保护、保存、移动、选择、复制、粘贴、查找、合并、排序、筛选等，而且这些方法都可以通过录制宏产生对应的代码，因此不必花费精力去记这些方法的书写方式，学会录制宏和查询帮助即可。

对象的方法通常都可以录制，例如创建条件格式、新建工作表、删除单元格的值、筛选等都能通过录制宏得到一切代码。不过学习这些方法的重点在于方法的参数，大多数方法具有多个参数，而且参数还分必选参数和可选参数，因此比较复杂。

通过录制宏可以了解各种方法的书写方式及其参数名称。下面以录制“选择性粘贴”宏为例，学习 PasteSpecial 方法的正确方式。

step 1 新建一个工作簿，然后单击“录制宏”按钮，保持默认的宏名称“宏 1”不变，将“保存在”设置为“当前工作簿”，然后单击“确定”按钮。

step 2 选择 A1 单元格，然后按<Ctrl+C>组合键复制 A1 的值。

step 3 选择 B1 单元格，单击鼠标右键，从弹出的右键菜单中选择“选择性粘贴”→“数值”命令。

step 4 单击“停止录制”按钮，然后按<Alt+F11>组合键进入 VBE 窗口。

step 5 双击“模块 1”，可在模块中看到以下代码：

```
Sub 宏1()
Range("A1").Select
    Selection.Copy
Range("B1").Select
    Selection.PasteSpecial Paste:=xlPasteValues, Operation:=xlNone, SkipBlanks _
:=False, Transpose:=False
End Sub
```

将宏代码与前面的操作步骤对应起来查看即可明白代码“Range("A1").Select”表示选择 A1 单元格，代码“Selection.Copy”表示复制选区中的值，“Range("B1").Select”表示选择 B1 单元格，而“Selection.PasteSpecial”表示将复制对象选择性粘贴到当前选区中，其后面的部分属于 PasteSpecial 方法的参数。

step 6 打开 VBA 的帮助窗口，以“Range.PasteSpecial”为关键字进行查询，从而可以得到以下语法描述：

```
表达式.PasteSpecial(Paste, Operation, SkipBlanks, Transpose)
```

根据帮助中的说明得知 Range.PasteSpecial 方法的 4 个参数分别对应于要粘贴的内容、粘贴时的操作方式、是否跳过空白单元格，以及是否转置 4 个项目。而“选择性粘贴”对话框中正好也有这 4 项，它们是一一对应的，如图 4.7 所示。因此只要熟悉 Excel 的基本操作，再配合录制宏及查询帮助可以快速掌握各种方法的语法和参数含义。

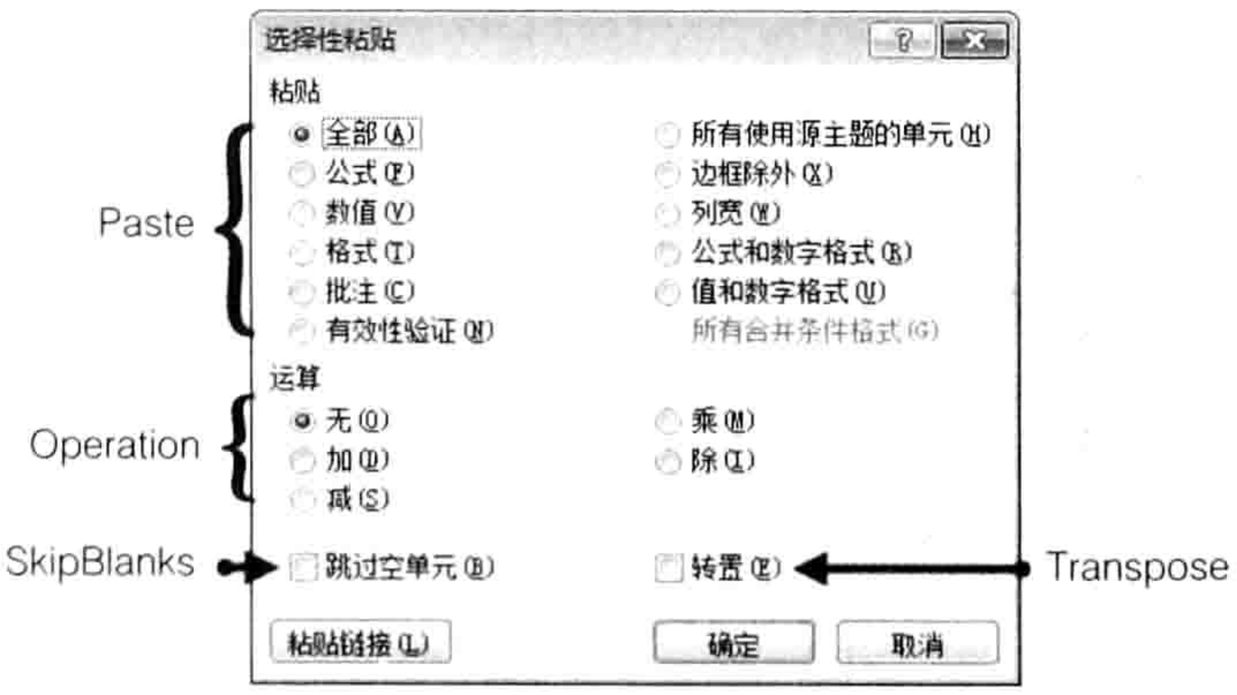


图 4.7　Range.PasteSpecial 方法的操作界面与参数的对应关系

当然，并非对象的所有方法都是与操作界面一一对应的，有些方法的部分参数无法在操作界面

的选项中找到，好在仍然可以通过录制宏并查看 VBA 帮助从而掌握每个参数的含义。

例如选择 A2 单元格，然后执行 4 次删除单元格操作，在如图 4.8 所示的操作选项中分别选择 4 个不同的选项对应 4 种操作方式。将此过程录制下来后所产生的宏代码如下：

```
Sub 宏2()
Range("A2").Select
  Selection.Delete Shift:=xlUp
  Selection.Delete Shift:=xlToLeft
  Selection.EntireRow.Delete
  Selection.EntireColumn.Delete
End Sub
```

很显然，代码中的“Selection.Delete”代表删除单元格，此处的 Selection 代表单元格对象。单元格对象的类别名称是 Range，因此应以“Range.Delete”为关键字去查询帮助。

帮助中关于 Range.Delete 方法的语法说明是：

```
表达式.Delete(Shift)
```

图 4.8　删除单元格的操作界面

虽然它只有一个参数，参数 Shift 只有 xlShiftToLeft 和 xlShiftUp 两个选项，分别对应于“右侧单元格左移”和“下方单元格上移”，但是宏代码的最后两句中明显增加了 EntireRow 和 EntireColumn 两段，分别以 EntireRow 和 EntireColumn 为关键字到 VBA 的帮助中查询可以得知它们分别代表整行和整列，这说明图 4.8 中后两个操作选项对应于 EntireRow 和 EntireColumn。换而言之，VBA 不是提供两个参数对应图 4.8 中的两个操作选项，而是通过重置对象的方式解决问题，将对象 Selection 重置为整行或者整列。

以上两个案例说明了对象的方法虽然可能有比较多的参数或者参数比较复杂，但是配合录制宏和 VBA 帮助就足以快捷地掌握这些方法，当然其前提是熟悉 Excel 的基本操作，只有对 Excel 的基本操作比较熟悉才能录制宏，从而根据宏代码去查询帮助、查看语法。

4.3　对象的事件

对象是组成 Excel 软件的基本单元，对象的属性用于指明对象的各种特征，对象的方法则用于操作这些对象。学习 VBA 其实就是用对象的方法去操作对象，以及读、写对象的属性。

为了提升程序的效率，Excel VBA 还为常用对象提供了事件，通过事件可以让程序自动化执行。本章对事件进行简要的阐述，在本书的第 9 章中会有大量关于事件的综合应用。

4.3.1　什么是事件

事件是对象在某个状态下触发的动作，事件是 VBA 为特定对象赋予的一个特性。不是每个对象都有事件，只有几类最常用的对象才有事件。

每一个事件都对应一个事件过程，通过事件过程可以让对象在满足特定条件时自动执行命令，从而更方便地控制对象，同时也提升工作效率。

可以用一个形象化的比喻来阐述事件：假设冰块是一个对象，冰块在坠落后会摔碎，冰块在坠落时就发生了坠落事件，摔碎是该事件的结果。若对冰块加温，那么会发生加温事件，在这个事件过程中冰块会融化，体积会膨胀，这是事件的结果……

具体到 Excel 中，工作簿是一个对象，打开工作簿时就触发了工作簿的打开事件，工作簿打开事件过程的书写方式如下：

```
Private Sub Workbook_Open()

End Sub
```

在以上过程中，Workbook 代表对象，Open 代表过程，在它们之间用“_”串连起来形成事件过程的完整名称。一切事件过程的名称都不允许被修改，否则该过程无法自动运行。

在事件过程中允许随意指定命令，表示触发事件要执行的操作。假设需要在打开作簿时在第 2 个工作表的 A1 单元格中显示今天的日期，那么事件过程的代码如下：

```
'Workbook 代表工作簿，Open 代表事件过程，Workbook_Open 则形成事件过程的完整名称
Private Sub Workbook_Open()
  Worksheets(2).Range("a1") = Date '在第二个工作表的 A1 单元格显示当前日期
End Sub
```

以上过程是工作簿事件过程，代码必须放在 ThisWorkbook 窗口中才生效，如图 4.9 所示。

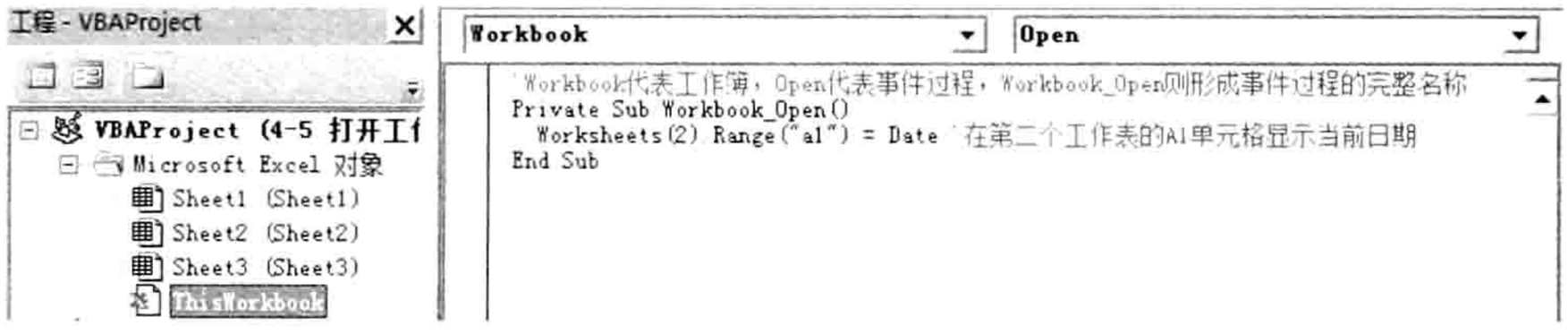


图 4.9　工作簿事件过程的存放位置

CD-ROM　本例文件参见光盘：..\第四章\4-5 打开工作簿时在 A1 显示日期.xlsm

4.3.2　事件的分类及其层级关系

Excel 2010 中具有事件的对象包含 Excel 应用程序对象、工作簿对象、工作表对象、图表对象、ActiveX 控件对象、窗体对象、窗体中的控件。

本章和第 8 章主要讲述工作簿事件和工作表事件。由于应用程序事件的应用比较复杂，需要使用类模块的知识，因此在本书的第 19 章再讲述应用程序事件。而窗体和窗体中的控件相关的事件则会出现在本书的第 17 章中。

应用程序对象、工作簿对象、工作表对象的事件被称作应用程序事件、工作簿事件和工作表事件，它们之间是有层级关系的，应用程序事件包含工作簿事件、工作簿事件包含工作表事件。当触发工作表事件时，同时也触发工作簿事件和应用程序事件，当触发工作簿事件时也会触发应用程序事件，但是反过来不成立，即触发高级别事件时不一定触发低级别事件。

4.3.3　工作簿事件与工作表事件一览

Excel 2010 的工作簿事件共有 36 种，现罗列如下，如表 4-2 所示。

表 4-2　工作簿事件一览

事件名称	功能说明
Activate	激活工作簿、工作表、图表工作表或嵌入式图表时发生此事件
AddinInstall	当工作簿作为加载宏安装时，发生此事件
AddinUninstall	当工作簿作为加载宏卸载时，发生此事件

续表

事件名称	功能说明
AfterSave	在保存工作簿之后发生此事件
AfterXmlExport	在 Microsoft Excel 保存或导出指定工作簿中的 XML 数据之后发生此事件
AfterXmlImport	当刷新现有的 XML 数据连接或新的 XML 数据被导入任意一个打开的 Microsoft Excel 工作簿之后，发生此事件
BeforeClose	在关闭工作簿之前，先产生此事件。如果该工作簿已经更改过，则本事件在询问用户是否保存更改之前产生
BeforePrint	在打印指定工作簿（或者其中的任何内容）之前，发生此事件
BeforeSave	保存工作簿之前发生此事件
BeforeXmlExport	在 Microsoft Excel 保存或导出指定工作簿中的 XML 数据之前发生此事件
BeforeXmlImport	在刷新现有的 XML 数据连接或新的 XML 数据被导入任意一个打开的 Microsoft Excel 工作簿之前，发生此事件
Deactivate	图表、工作表或工作簿被停用时发生此事件
NewChart	在工作簿中创建新图表时发生
NewSheet	当在工作簿中新建工作表时发生此事件
Open	打开工作簿时，发生此事件
PivotTableCloseConnection	在数据透视表的连接关闭之后发生此事件
PivotTableOpenConnection	在数据透视表的连接打开之后发生此事件
RowsetComplete	如果用户在 OLAP 数据透视表上深化记录集或调用行集操作，则会引发此事件
SheetActivate	当激活任何工作表时发生此事件
SheetBeforeDoubleClick	当双击任何工作表时发生此事件，此事件先于默认的双击操作发生
SheetBeforeRightClick	右键单击任意一个工作表时发生此事件，此事件先于默认的右键单击操作
SheetCalculate	在重新计算工作表时或在图表上绘制更改的数据之后发生此事件
SheetChange	当用户或外部链接更改了任何工作表中的单元格时发生此事件
SheetDeactivate	当任何工作表被停用时发生此事件
SheetFollowHyperlink	单击 Microsoft Excel 中的任何超链接时发生此事件
SheetPivotTableAfterValueChange	在编辑或重新计算（针对包含公式的单元格）数据透视表中的单元格或单元格区域后发生此事件
SheetPivotTableBeforeAllocateChanges	在向数据透视表应用更改前发生此事件
SheetPivotTableBeforeCommitChanges	在针对 OLAP 数据源提交对数据透视表的更改前发生此事件
SheetPivotTableBeforeDiscardChanges	在放弃对数据透视表所做的更改之前发生此事件
SheetPivotTableChangeSync	在对数据透视表进行更改之后发生此事件
SheetPivotTableUpdate	在数据透视表的工作表更新之后发生此事件
SheetSelectionChange	任意一个工作表上的选定区域发生更改时，将发生此事件（但图表工作表上的选定区域发生改变时，不会发生此事件）

续表

事件名称	功能说明
Sync	当作为“文档工作区”一部分的工作簿的本地副本与服务器上的副本进行同步时，发生此事件
WindowActivate	工作簿窗口被激活时，将发生此事件
WindowDeactivate	任何工作簿窗口被停用时将发生此事件
WindowResize	任何工作簿窗口调整大小时将发生此事件

工作簿事件的代码必须放在 ThisWorkbook 对象的代码窗口中，放在其他地方则只能当作普通的子过程处理，不具备自动执行的功能。

而工作表事件只有 14 种，现罗列如下，如表 4-3 所示。

表 4-3　工作表事件一览

名　　称	说　　明
Activate	激活工作簿、工作表、图表工作表或嵌入式图表时发生此事件
BeforeDoubleClick	当双击工作表时发生此事件，此事件先于默认的双击操作
BeforeRightClick	右键单击工作表时发生此事件，此事件先于默认的右键单击操作
Calculate	对于 Worksheet 对象，在对工作表进行重新计算之后发生此事件
Change	当用户更改工作表中的单元格，或外部链接引起单元格的更改时发生此事件
Deactivate	图表、工作表或工作簿被停用时发生此事件
FollowHyperlink	当单击工作表上的任意超链接时，发生此事件
PivotTableAfterValueChange	在编辑或重新计算（针对包含公式的单元格）数据透视表中的单元格或单元格区域后发生此事件
PivotTableBeforeAllocateChanges	在向数据透视表应用更改前发生此事件
PivotTableBeforeCommitChanges	在针对 OLAP 数据源提交对数据透视表的更改前发生此事件
PivotTableBeforeDiscardChanges	在放弃对数据透视表所做的更改之前发生此事件
PivotTableChangeSync	在对数据透视表进行更改之后发生此事件
PivotTableUpdate	工作簿中的数据透视表更新后发生此事件
SelectionChange	当工作表上的选定区域发生改变时发生此事件

工作表事件的代码必须放在工作表的代码窗口中，放在其他地方例如模块中或者 ThisWorkbook 中将不具备自动执行的功能。

4.3.4　工作簿与工作表事件的作用对象

工作簿事件和工作表事件的作用对象是不同的，不过触发机制是一致的。

1. 工作簿事件

部分事件过程由对象和触发条件两部分组成，例如打开工作簿的事件过程名称是“Workbook_Open”，其中的 Workbook 是作用对象，代表代码所在的工作簿，Open 是触发条件。当满足 Open 这个条件时，代码所在的工作簿就触发了“Workbook_Open”事件。

其他的部分工作簿事件由 3 部分组成，包括对象、触发条件和参数，参数主要用于获取事件过程中的某些信息，或者作为事件的开关。对于事件过程的参数将在本书第 8 章有详细说明。

简单地说，工作簿事件只作用于代码所在工作簿，对其他工作簿无效。

2. 工作表事件

工作表事件过程的名称也包含对象和触发条件，部分事件还包含参数。

工作表的 Change 事件的完整名称是“Worksheet_Change(ByVal Target As Range)”，其中 Worksheet是作用对象，代表代码所在的工作表，Change代表触发条件，事件过程Worksheet_Change 的整体含义是改变代码所在工作表的单元格的值时触发 Worksheet_Change 事件。Change 要理解为“改变”，它是一个动词，而非形容词“变化”，因为单元格的值没有变化时也可能触发该事件，例如删除空白单元格的值。

删除空白单元格这个动作的原本目的是去“改变”单元格 A1 中的值，尽管单元格 A1 中的值最终没有产生变化，但是执行命令的初衷是“改变”，这个动作已经执行，不管执行的结果是什么。

工作表事件的参数用于获取事件相关的信息。例如 Worksheet_Change 事件的参数 Target 代表此事件过程中的单元格对象，事件过程改变了哪些单元格，参数 Target 就代表哪些单元格对象。以下步骤可以让读者了解事件过程与参数的关系。

step 1 新建一个空白工作簿。

step 2 按<Alt+F11>组合键进入 VBE 窗口。

step 3 在工程资源管理器中双击 Sheet1 工作表从而进入工作表的代码窗口。

在工作表的代码窗口上方有一个对象窗口和过程窗口，如图 4.10 所示，它们是录入事件过程代码的辅助工具，单击这两个列表可以自动产生任意工作表事件过程的程序外壳。

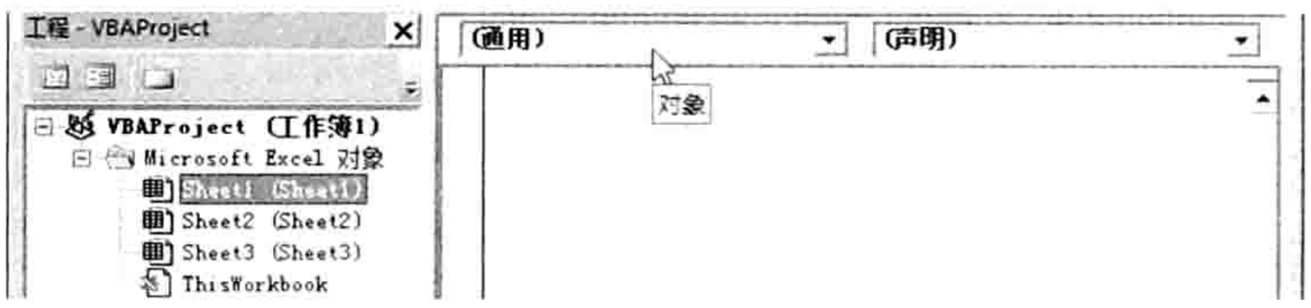


图 4.10 Sheet1 工作表的代码窗口

step 4 单击对象窗口，从列表中选择“Worksheet”，此时在代码窗口中会自动产生默认的工作表事件“SelectionChange”的程序外壳。

step 5 单击过程窗口，从列表中选择“Change”，如图 4.11 所示。此时在 Sheet1 工作表代码窗口中包含了工作表的 SelectionChange 事件和 Change 事件的程序外壳。

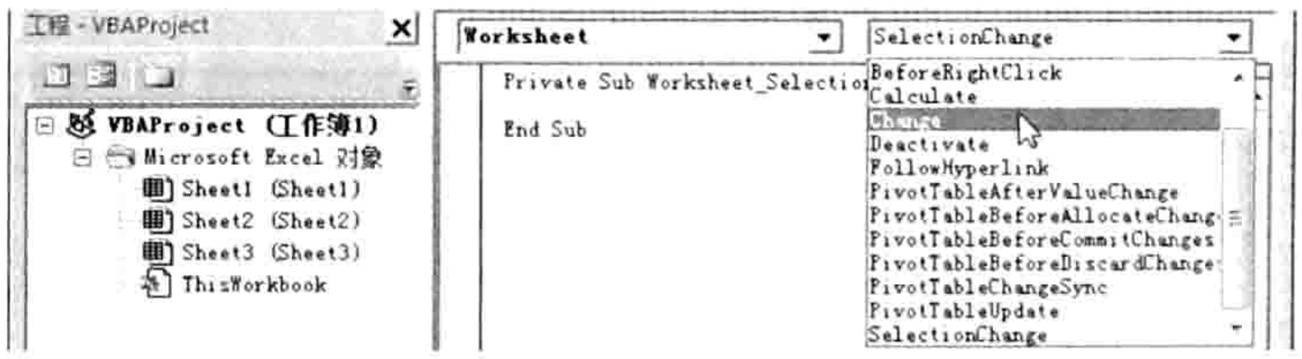


图 4.11 选择“Change”过程

step 6 删除 SelectionChange 事件的程序外壳，在 Change 事件的程序外壳中录入以下代码：

```
Application.StatusBar = "您当前正在修改" & Target.Address & "的值"
```

代码的含义是当修改单元格中的值时在状态栏中显示被修改的单元格的地址。

step 7 按<Alt+F11>组合键返回工作表界面，进入 Sheet1 工作表中。

step 8 在 A1 单元格中输入数值 1，然后按 Enter 键，此时可在状态栏中看到如图 4.12 所示的提示信息。由于此命令是通过事件过程执行，因此它是全自动的，在符合条件——改变 Sheet1 工作表的任意单元格的值时自动执行。

step 9　选中 B1:C2 区域，然后输入 10，并按<Ctrl+Enter>组合键结束，此时状态栏将会显示如图 4.13 所示的提示信息。

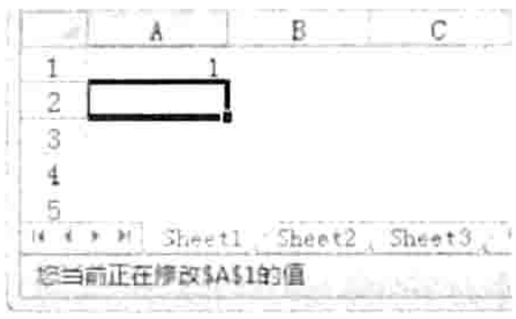


图 4.12　修改 A1 的值

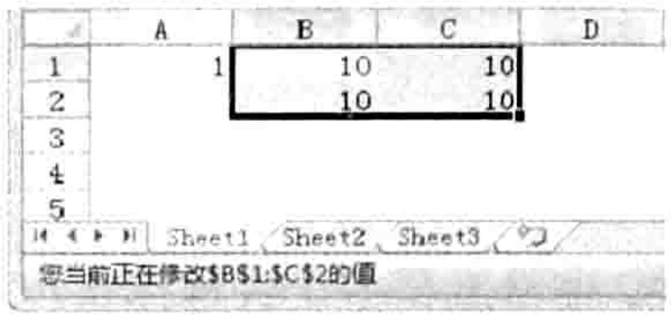


图 4.13　修改 B1:C4 的值

step 10　进入 Sheet2 工作表，修改任意单元格的值，状态栏将不产生任何变化。

通过以上 10 个步骤的测试可以了解工作表事件的以下几个知识点：

其一，工作表事件的代码需要放在工作表对应的代码窗口中。

其二，事件过程的程序外壳不需要手工拼写，而是用鼠标单击对象与过程列表，从而让代码自动产生。

其三，工作表事件的代码仅对代码所在工作表生效，不能跨表调用事件过程。

其四，工作表事件的参数 Target 代表当前正在操作的区域对象，可以通过该参数获得与该区域相关的一切信息。

其五，事件过程在满足条件时会全自动执行，无须人工调用。

4.3.5　快速掌握事件过程

Excel 总共有几百个事件，对于每个事件过程的书写方式不需要花费任何精力和时间去记忆，通过鼠标在事件代码窗口的对象列表和过程列表中选择即可产生代码。而对于每个事件过程的含义也同样不需要花费时间去记忆，一是完全没有必要，二是由于事件的数量比较多，很难将它们记住。笔者在此向读者们推荐两种快捷、有效的方法解决此问题。

1. 查询帮助

当需要了解某个事件过程的功能时，以事件过程名称为关键词在 VBA 的帮助中进行查询，不超过两秒钟即可找到对应的帮助信息。例如要查询工作表的 Change 事件的含义，由于工作表的类别名称是 Worksheet，那么关键词是“Worksheet.Change 事件”，如果要查询工作簿的 SheetCalculate 事件，由于工作簿的类别名称是 Workbook，因此关键词是“Workbook.SheetCalculate 事件”，其他事件以此类推。

2. 打印笔记

对于常用的表格式的知识点，应将其整理到笔记（Word 文件或者 Excel 表格文件）中，要用时从笔记中翻查答案。也可以将笔记打印出来放在桌边，需要时浏览一眼即可，这样既能瞬间找到需要的答案，又能避免因记忆出错而导致代码无法达成预期效果。

4.3.6　何时需要使用事件过程

事件过程的存在价值在于“自动化”三个字，只要符合指定的条件就可以自动执行预先编好的代码，因此当需要自动化执行时才使用事件过程，其他情况下宜在模块中编写子过程，通过按钮、<Alt+F8>组合键或者菜单调用。

本节主要介绍事件的概念、类别、用法与用途，对于具体的应用将在本书第 8 章进行详细讲解。

◆　◆　◆

第5章 通过变量强化程序功能

通俗地讲，变量就是一个在程序执行期间可以改变其值的量，它由用户指定名称，同时分配数据类型。在过程中使用变量可以使程序的功能更强大，让程序满足繁复的工作需求。

本章主要介绍变量的概念、声明变量的方式，以及与变量相关的另外两个概念——常量和数据类型。在以后的每一章都会大量涉及变量的相关应用。

5.1 数据类型

在学习变量之前，有必要了解一下变量的相关概念——数据类型。

数据类型是一类数据的集合，它决定变量的占用空间及变量的种类。在声明变量时指定合理的数据类型可以使程序具有更高的执行效率。

5.1.1 为什么要区分数据类型

数据类型用于指定数据以何种方式储存在内存中，正确地分配数据类型可以节约数据的占用空间和执行效率。

每一个数据类型都有一个独有的名称，它限制了数据的有效范围。

VBA 将 0～255 的整数定义为 Byte 型，如果一个变量被定义为 Byte 型，那么它只允许在 0～255 中变化，否则将会出错；VBA 将–32 768～32 767 的整数定义为 Integer 型，将–3.402 823 E38～–1.401 298 E-45（科学记数法）的整数和小数定义为 Single 型，将文本定义为 String 型，每一种数据类型都有它的专用范围，超过范围赋值则会出错……

根据以上分析，似乎在 VBA 中引入数据类型这个概念只会带来麻烦，为什么还要区分数据类型呢？答案是方便管理，就和每个班级都会将学生分组一样，分组后再管理学生将高效得多。

其实也可以通过人民币的面值分配来理解 VBA 中的数据类型。人民币的面值分为 1 分、2 分、5 分、1 毛、2 毛、5 毛、1 元、2 元……到 50 元、100 元，通过不同面值的钞票解决不同的需求。例如要支付 5 分、20 元、70 元或者 100 元等不同范围的金额，如果只有面值为 1 元或者只有面值为 100 元的钞票，那么很难处理此类支付问题。如果提供从 1 分、2 分、3 分到 99 元、100 元共 10000 种面值的钞票，尽管可以解决以上支付问题，但是钞票的管理问题又会显得极为烦琐，因此现实中采用了折中的方案——提供 13 种面值的钞票。

数据类型的存在价值也与人民币面值分配相近，给不同范围和不同类型的数据指定不同的类型名称，并对每个类别的数据按需分配储存空间。如果在使用过程中正确地指定了数据类型，那么程序的调用时间可以更短、数据的占用空间可以更小，从而实现优化程序代码的目的。

5.1.2 认识 VBA 的数据类型

VBA 中支持 10 多种数据类型。不同数据类型的差异主要体现在 3 个方面：类型名称、占用内存空间的大小和取值范围。表 5-1 中罗列了 VBA 所有数据类型的存储空间大小与范围。

表 5-1　VBA的数据类型

数据类型	存储空间大小	范围
Byte	1 个字节	0～255
Boolean	2 个字节	true 或 false
Integer	2 个字节	–32 768～32 767
Long（长整型）	4 个字节	–2 147 483 648～2 147 483 647
Single（单精度浮点型）	4 个字节	负数时从–3.402 823 E38～–1.401 298 E-45；正数时从 1.401 298 E–45～3.402 823 E38
Double（双精度浮点型）	8 个字节	负数时从–1.797 693 134 862 31E 308～4 840 656 458 412 47E-324
Currency（货币型）	8 个字节	从–9 223 372 036 854 775 808～922 337 203 685 477.580 7
Decimal	14 个字节	没有小数点时为±79 228 162 514 264 337 593 543 950 335，而小数点右边有 28 位数时为±7.922 816 251 426 433 759 354 395 033 5；最小的非零值为±0.000 000 000 000 000 000 000 000 1
Date（日期）	8 个字节	100 年 1 月 1 日到 9999 年 12 月 31 日
String（变长）	10 字节加字符串长度	0 到大约 20 亿
String（定长）	字符串长度	1 到大约 65 400
Variant（数字）	16 个字节	任何数字值，最大可达 Double 的范围
Variant（字符）	22 个字节加字符串长度	与变长 String 有相同的范围

从表 5-1 中分析得知，如果某个变量用于处理的数据是单科成绩，其取值范围在 0～100，那么就应该将该变量声明为 Byte 型，用 Long 型虽然也可以正确执行运算，但它占用的内存是 Byte 的 4 倍，将使程序的工作效率降低。

假设变量需要处理的数据是 10 个科目的考试成绩汇总，那么不宜使用 Byte 型，否则将发生“溢出”错误，因为 Byte 型的上限是 255。

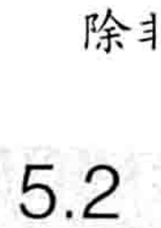
假设在过程中某个变量总是代表数值去参与数学运算，那么不宜将变量声明为 String 型，因为 String 型变量占用的空间远远大于数值型变量（包括 Byte，Integer，Long，Single，Double，Currency 等）。

假设过程中某变量总是代表文本去参与运算，那么此变量不宜声明为数值专用的数据类型（包含 Byte，Integer，Long，Single，Double，Currency 等），否则执行代码时会产生“类型不匹配”错误。

以下提供三个错误的数据类型应用案例，读者可以对照表 5-1 自行修改代码，使代码不再出错。

```
Sub 错误地声明变量的数据类型1()'代码存放位置：模块中
    Dim a As Byte      '对变量a声明为Byte型(有效值在0～255)
    a = "长江"         '对变量赋值为文本“长江”，执行代码时必定出错——“类型不匹配”
End Sub
Sub 错误地声明变量的数据类型2()
    Dim b As Byte      '对变量b声明为Byte型(有效值在0～255)
    b = 300            '对变量赋值为300，执行代码时必定出错——“溢出”
End Sub
```

```
Sub 错误地声明变量的数据类型3()
    Dim c As Integer '对变量c声明为Integer型(有效值-32768~32767,整数)
    c = 300.05       '对变量赋值为300.05，赋值后将丢失小数部分，因为它只能储存整数
    MsgBox c
End Sub
```



本例文件参见光盘：..\第五章\5-1 三个数据类型错误案例.xlsm

由于数据类型必须与 Dim 语句共用才有意义，因此以上案例中使用了 Dim 语句为变量指定数据类型。对于 Dim 语句的语法与功能在 5.2 节进行详细阐述。

知识补充：数据类型中比较特别的是 Variant 型，它也被称作变体型，是变量的默认类型，任何未指定类型的变量都是变体型变量。变体型变量的特征是根据数据变化而自动分配数据类型。例如对变体型变量 A 赋值为文本“ABC”时，VBA 会自动将该变量转换成 String 型，若对变量赋值为 1 123.45 时，VBA 则将该变量转换成 Double 型，若对变量赋值为 1 000，那么 VBA 会将该变量转换成 Integer 型。将变量声明为变体型的优点是书写方便，因为其在默认状态下就是变体型，因此编写代码时可以不用指定变量的类型名称；缺点是其占用空间大，处理速度慢。编程的初衷是提升程序工作效率，因此不宜为了少写几个字而牺牲程序工作的效率，除非无法确定变量的赋值范围，不能提前指定具体类型，否则应尽可能不使用变体型变量。

5.2 声明变量

变量对于 VBA 程序而言举足轻重，没有变量的程序只能处理一些简单的问题。

本节讲述变量的定义、用途、声明方式，以及区分动态变量和静态变量等知识。

5.2.1 变量的定义

“变量是一个已经被命名的存储位置，它包含了程序每个执行阶段所修改的数据”，这是 VBA 帮助中对于变量的定义。如果要通俗地描述变量，那么它就是在程序执行中可以任意修改的量。

变量的本质是可以随意修改和代表一切未知数，变量的作用也体现在这两个方面。

变量分为数据变量、数组变量和对象变量，它们的声明方式和使用对象各不相同。如果要从使用难度上升序排序，那么应按以下顺序排列：

数据变量→对象变量→数组变量

录制宏时可以产生 Sub 过程代码，但是永远不可能产生变量，因此宏代码都是比较简单的程序，需要在其中加入循环语句、判断语句和变量之后才能变得更强大、更灵活。

5.2.2 变量的声明方式

声明变量有两种方式——显式声明和隐式声明。所谓的隐式声明其实就是不声明，直接在代码中调用变量。使用隐式声明变量后患无穷，既可能导致程序出错还会降低代码的执行效率，因此本节所说的声明变量一律指显示声明。

1. 语法

显示声明变量的前提是了解数据类型的有效范围和书写方式，因此笔者建议读者在编程前将表 5-1 打印出来放在身边，在声明变量时查看该表即可，这样既确保不出错，又可以避免浪费太多的时间去背单词。

声明变量有 4 种方式：Dim，Public，Private，Static，其中最常用的是 Dim 语句。使用 Dim 语句声明变量的语法如下：

```
Dim [WithEvents] varname[([subscripts])] [As [New] type] [, [WithEvents]
varname [([subscripts])] [As [New] type]]
```

各参数的含义如表 5-2 所示。

表 5-2 Dim语句的参数说明

参　数	功能描述
WithEvents	可选的。声明 varname 是一个用来响应由 ActiveX 对象触发的事件的对象变量。只有在类模块中才是合法的
varname	必需的。变量的名称；遵循标准的变量命名约定
subscripts	可选的。数组变量的维数；最多可以定义 60 维的多维数组
New	可选的。可隐式地创建对象的关键字。如果使用 New 来声明对象变量，则在第一次引用该变量时将新建该对象的实例，因此不必使用 Set 语句来给该对象引用赋值。New 关键字不能与 WithEvents 一起使用
type	可选的。变量的数据类型。所声明的每个变量都要是一个单独的 As type 子句

针对表 5-2 有必要做以下补充：

（1）WithEvents 仅在类模块中才使用，VBA 的初、中级教程是不涉及类模块相关知识的。

（2）subscripts 部分仅用于数组变量，可为数组变量指定维数，在本书的第 13 章会有相关的教学内容。

（3）New 关键字用于在声明变量的同时创建一个对象，在声明数据变量时不能使用 New 关键字，而工作中使用最多的是数据变量。

（4）type 部分是指变量的类型，尽管这是可选的，但是为了确保代码的执行效率尽可能指定变量的类型。

2. 命名规则

声明变量时变量名称是必选项，为变量指定名称时必须遵循以下规则。

（1）第一个字符必须使用英文字母或者汉字，禁止使用数字。

（2）不能使用空格和句号、感叹号、 @、&、$、，、、# 等标点符号。

（3）名称长度不能超过 255 个字符。

（4）变量不宜与过程同名，否则调用过程时可能产生混乱。

（5）同一个过程不允许存在多个同名的变量，不同过程中允许存在同名的变量。

（6）不能与 VBA 的保留字一致。例如 Dim、Sub、End、as 和 Exit 等。

下面列举一些常见的数据变量声明方式：

```
Dim name
Dim a as Byte
Dim 姓名 as String
Dim 产量 as long
Dim ShuiLv As Single
```

在声明变量时允许同一行中声明多个变量，也允许部分变量指定数据类型、部分变量不指定数据类型。在同一行代码中声明多个变量时只能使用一次 Dim 语句，例如：

```
Dim a, b, c——声明了三个变体型变量，三个变量皆未指定类型名称
```

Dim a As Byte, b, c, d As String——声明一个 Byte 型变量、一个 String 型变量和两个变体型变量

下面是两种错误的声明方式：

```
Dim a as String,b,dim c as Variant
```

以上代码错在同一句代码中使用了两个 Dim 语句。

```
Dim a As String, b
c As String
```

以上代码错在声明变量 c 时未使用 Dim 语句。

5.2.3 变量的赋值方式与初始值

在声明变量的同时变量就拥有了一个初始值，不同类型的变量的初始值是不同的。

按值的类型可以将变量分成六类：数值型、布尔值、文本型、日期型、数组型和对象型。

1. 数值型变量

数值型变量包括 Byte、Integer、Long、Single、Double 和 Currency 型，它们的初始值都是 0，即声明变量后、对变量赋值之前，用变量去参与运算时皆当作 0 处理。

对数值型变量的赋值方式如下：

```
Let 变量 = 值
```

其中 Let 是可选的，允许忽略。等号右边的值只能是数字，允许在数字前后添加双引号，但应尽量不添加引号。

如果将数值型变量赋值为文本，那么会产生“类型不匹配”错误。以下是常见的两种赋值方式：

```
a = 125
b = "800"
```

2. 布尔型变量

布尔型变量是 Boolean 型的变量，它的赋值范围包括 True 和 False，默认值是 False。

对布尔型变量的赋值语法与数值型变量一致。

在对布尔型变量赋值时可赋值为 True 和 False，也可以赋值为“True”和“False”，但是尽量不要添加引号。

如果对布尔值变量赋值为数值，那么 0 当作 False 处理，0 以外的值都当作 True 处理。例如代码“变量=10”相当于“变量=True”，而代码“变量=0”相当于“变量=False”。

3. 文本型变量

文本型变量是指 String 型的变量，它的赋值范围比较广，赋值为文本、数值、布尔值和日期都可以，只不过赋值后全部当作文本处理。

文本型变量的初始值是空文本（即""），也称作零长度的文本。

对文本型变量的赋值方式和对数值型变量的赋值方式一致。

4. 日期型变量

日期型变量是 Date 型的变量，包含日期和时间，它的日期赋值范围是 100 年 1 月 1 日到 9999 年 12 月 31 日，赋值范围在 0:00:00 到 23:59:59。时间初始值是“0:00:00”。

对日期型变量赋值时必须在前后添加“#”，例如：

```
Dim MyDate1 As Date, MyDate2 As Date
```

```
MyDate1 = #5/4/2014#
MyDate2 = #9:25:48 #
```

5. 数组变量

前面所说的四类变量都只能存放单个值，而数组变量可以存放无限个值，仅受内存限制。换而言之，只要内存足够大，多少数据都可以放在数组变量中去。数组变量的声明方式、应用方式都比较复杂，本书将它放在第 13 章讲述。

6. 对象型变量

对象型变量相对于数据变量是比较特殊的一种变量，它用于储存一个对象，而不是一个或者一组数据，因此声明方式和赋值方式都有所不同。

对对象变量赋值的语法如下：

```
Set 变量 = 对象
```

Set 是必需的，不允许被忽略。赋值时只能使用对象，如果赋值为数值、文本或者日期将会产生“类型不匹配”的错误提示。正确的赋值方式如下：

```
Dim Sht As Worksheet
Set Sht = Worksheets("Sheet2")
```

对于未赋值的对象变量，其初始值是 Nothing。

5.2.4 如何确定变量的数据类型正确

声明变量的同时指定变量的数据类型的目的是提升代码的运行效率，如果为变量指定了错误的数据类型则可能适得其反，不仅降低程序的运行效率，还可能导致程序运行出错。

那么如何确保声明变量时指定的数据类型是正确的呢？其实很简单，符合“一大一小”两个原则即可。

1. 原则一：大

大是指数据类型的有效取值范围必须大于变量的赋值范围。例如有一个名为“产量”的变量，在编写代码前需要先了解“产量”变量的最大值和最小值，假设最高产量是 800，最低产量是 400，那么参照表 5-1 中的各种数据类型的有效范围就可以发现——不能声明为 String 型，因为它适合于文本；不能声明为 Byte 型，因为它的最大值是 255；也不能声明为 Date 型，Date 仅用于日期；更不能声明为 Boolean 型，它只适用于逻辑值 True 和 False.

2. 原则二：小

小是指在所有符合条件的数据类型中取占用空间最小的一个。例如变量“产量”的赋值上限是 800，下限是 400，那么将它赋值为 Integer，Long，Single，Double，Currency，Decimal，Variant 等类型都不会产生“溢出”错误，但是为了程序的运行效率，必须在它们之间找一个占用空间最小的数据类型——那就是 Integer 了，仅 2 个字节。

换而言之，原则一的目的是防止代码出错，原则二的目的是提升程序运行效率，两者同等重要。

如果某个变量的赋值范围不确定呢？如果知道变量是数值，不知道值的大小、是否带有小数，那么可以声明为 Double 型；如果只知道变量是整数，不确定值的大小，那么宜用 Long 型，工作中超过 2 147 483 647 的值极为少见；如果变量是文本还是数值都不确定，那么宜用变体型 Variant。

5.2.5 正确声明变量的数据类型的优势

对于初学者而言，为每一个变量都指定其数据类型尽管并不复杂，但是需要有耐心才能完成。

那么为每一个变量指定数据类型在工作中会有哪些优势呢?

简要地说，显式声明变量并且正确地指定其类型有两个作用：提升效率和防止出错。现分别使用两个案例来印证。

1. 提升效率

为变量正确地分配数据类型可以节约变量所占用的空间，提升程序执行效率。以下两个过程的功能完全一致，区别仅在于过程 A 为变量指定的类型名称，而过程 B 中的一切变量都是变体型，当分别执行两段程序后可以发现过程 B 的执行时间是过程 A 的两倍左右。

```
Sub A()'代码存放位置:模块中
    Dim tim As Date, x As Integer, y As Integer, z As Integer
    tim = Timer
    For x = -100 To 10000
        For y = 1 To 10000
            z = x + y
        Next y, x
        Application.StatusBar = "程序执行时间: " & Format(Timer - tim, "0.000秒")
    End Sub
Sub B()'代码存放位置:模块中
    tim = Timer
    For x = -100 To 10000
        For y = 1 To 10000
            z = x + y
        Next y
    Next x
    Application.StatusBar = "程序执行时间: " & Format(Timer - tim, "0.000秒")
End Sub
```

对于以上过程的含义可以完全忽略，通过案例了解指定数据类型的重要性即可。

本例文件参见光盘：..\第五章\5-2 正确指定变量类型的优势.xlsm

2. 防止溢出

在 xlsx 或者 xlsm 格式的工作簿中执行以下 VBA 代码将会产生“溢出”错误：

```
Sub 计算单元格数量()'代码存放位置:模块中
  MsgBox Rows.Count * Columns.Count   '利用行数乘以列数得到单元格总数量
End Sub
```

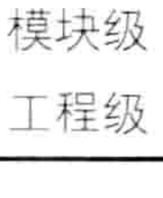
以上代码的思路是用总行数乘以总列数得到单元格的总数量，在理论上此思路完全可行。然而 VBA 在执行乘法运算时，总取最精确的乘数作为结果的数据类型，这种策略偶尔会导致程序出错，当乘积的值远远大于乘数时就可能产生“溢出”错误。本例中 rows.count 的值 1 048 576 属于 Long 型，VBA 会预先为表达式的结果也分配为 Long 型，然而计算结果 17 179 869 184 已经远超 long 型数据的上限，从而执行代码时产生了“溢出”错误。

如果预先声明两个 Double 型的变量，将 Rows.Count 和 Columns.Count 的值赋予变量就可以解决此问题：

```
Sub 单元格数量2()'代码存放位置:模块中
  Dim RowCount As Double, ColCount As Double '声明两个 Double 型的变量
```

```
  RowCount = Rows.Count          '将行数赋予变量 RowCount
  ColCount = Columns.Count       '将列数赋予变量 ColCount
  MsgBox RowCount * ColCount     '将行数与列数相乘，然后显示在信息框中
End Sub
```

修改后的代码不再出错是因为 Double 数据类型的上限大于乘法表达式的乘积。



本例文件参见光盘：..\第五章\5-3 正确指定变量类型的优势 2.xlsm

5.2.6 变量的作用域

变量的作用域是指允许在什么地方调用变量，或者可以理解为变量的可调用范围。

变量分为公有变量和私有变量，其中公有变量又区分于模块级公有变量和工程级公有变量。作用范围的差异取决于变量的声明方式和存放位置。具体请参考表 5-3。

表 5-3 变量的作用域

级　　别	作 用 域	存放位置	变量的声明方式
过程级	当前过程	过程中	使用 Dim 或者 Static 声明变量
模块级	当前模块	模块顶部	使用 Dim 或者 Private 声明变量
工程级	所有模块	模块顶部	使用 Public 声明变量

其中工程级变量可以被当前工程的任意模块中的任意过程调用；模块级变量只能在声明变量的模块中调用，允许跨过程调用；而过程级变量则只能在声明变量的过程中调用。

通过以下步骤可以加深对变量的作用域的理解。

step 1 新建一个工作簿，按<Alt+F11>组合键进入 VBE 界面。

step 2 单击菜单中的“插入”→“模块”命令，然后在模块中录入以下代码：

```
Public a As String
Dim b As String
Private c As String
Sub test()                    '代码存放位置:模块中
  Dim d As String             '声明过程级变量
  a = "工程级变量"            '对变量赋值
  b = "模块级变量"
  c = "模块级变量"
  d = "过程级变量"
End Sub
Sub 主程序()
  Call test '调用过程 Test,即对四个变量赋值
  '分别将 4 个变量的串连起来，用换行符分隔符，然后显示在信息框中
  MsgBox a & Chr(13) & b & Chr(13) & c & Chr(13) & d
End Sub
```

step 3 执行过程“主程序”，弹出如图 5.1 所示的信息框。根据执行结果可以得出结论：过程级变量 d 不能跨过程调用。

step 4 单击菜单中的“新建”→“模块”命令，然后将过程“主程序”的代码复制到模块 2 中。

step 5 执行模块 2 中的过程“主程序”，弹出如图 5.2 所示的信息框。根据执行结果可以得出结论：过程级变量 d 和模块级变量 c、d 不能跨模块调用。

图 5.1 调用结果 1

图 5.2 调用结果 2

CD-ROM 本例文件参见光盘：..\第五章\ 5-4 变量的作用域.xlsm

为什么要区分过程级、模块级和工程级变量呢？这是由工作需求决定的。当要求一个模块中的多个过程共用一个变量时，就需要使用模块级的变量，例如在过程 A 中对变量已经赋值过，那么过程 B 直接调用该变量参与运算即可，不再需要重新声明一个变量并且对它赋值。而需要跨模块调用变量时，则只有工程级变量能适应需求。工程级变量和模块级变量的声明语句都只能放在模块的顶部，在 Sub 过程或者 Function 过程之外。

知识补充：过程级变量区分于动态变量和静态变量，分别采用 Dim 和 Static 语句声明。动态变量的特点是过程结束后自动释放变量的值，静态变量只有关闭工作簿后才释放变量的值。

5.2.7 变量的生存周期

变量保留其值的这段时间称为生存周期，通俗地讲就是指在从声明变量到释放变量的值这期间的时间长短。

与生存周期相关的一个概念是“释放变量”。所谓的释放变量是指清空变量的值，释放其占用的空间，也可以理解为让变量还原到初始值。释放变量的值包括手动释放和自动释放，关于变量的生命周期的计算方式是针对自动释放而言的。

过程级变量的生命周期等于过程的执行时间，当变量所在的过程结束后变量的生存周期也相应结束。模块级变量和工程级变量的生命周期比较长，只要不关闭工作簿，那么它的值就会一直保留下去，可以随意调用该变量的值。在文件“5-4 变量的作用域.xlsm”中已经将这几类变量的生存周期演示过，读者可以反复测试该文件中的三个 Sub 过程，从中印证上述理论。

当然，也可随时手动释放变量的值。对于对象变量，将它赋值为 Nothing 即可，例如：

```
Set Sht = Nothing
```

对于单个的数据变量，将它赋值为初始值即可。如果需要一次性释放所有变量，那么可以在过程中使用 End 语句。不过 End 语句不仅仅是释放变量，它同时也会结束当前过程。如果当前过程在窗体中还会将窗体也一并关闭，因此一般不用 End 语句释放变量，它可能产生“误伤”，将其他需要保留值的公共变量也一并被释放了。

在关闭 Excel 工作簿后，该工作簿中的一切变量的值都会被释放。

5.3 对象变量

对象变量是指可以引用对象的变量，它可以代替对象去参与各种运算，拥有对象的一切属性、方法。在工作中关于对象变量的应用相当广泛，本节将详细阐述对象变量的相关知识。

5.3.1　如何区分对象变量和数据变量

数据变量用于存放数据，数值、文本、日期、逻辑值等都是数据，假设需要一个变量来存放这些数据，那么应将变量声明为数据变量，包括 Byte，Boolean，Integer，Long，Single，Double，Currency，Date，String 等类型。

对象变量用于存放储存对象，工作簿、工作表、单元格、批注、图表、图片等即为对象。声明对象变量和声明数据变量的语法完全一致，有区别的是类型名称不同。常用的对象名称见表 5-4。

表 5-4　常见的对象类型名称

对象类型名称	含义描述
Application	代表整个 Excel 应用程序
Workbook	代表 Excel 工作簿对象
Worksheet	代表工作表对象
Window	代表窗口对象
Range	代表单元格对象
Shape	代表嵌入到工作表中的图形对象，包括自选图形、OLE 对象、图片、图表、艺术字、文本框、批注等
Name	代表名称对象，可以是内置名称也可以是自定义名称
Chart	代表图表对象
WorksheetFunction	代表工作表函数对象
Comment	代表单元格中的批注对象

例如声明一个代表单元格的对象变量可以用以下语句：

Public TargetRng As Range——声明一个公共变量，名称为 TargetRng，类型为 Range；

Dim Rng As Range——声明一个名为 Rng 的变量，其类型为 Range。

要注意单元格对象的类别名称是 Range，不是 Cell 或者 Cells。

如果声明一个代表工作表的变量可用以下代码：

```
Dim Sht As Worksheet
```

要注意工作表的类别名称是 Worksheet，不是 Worksheets 或者 Sheet。

数据变量在赋值之前是 0 或者空文本（String 型），而对象变量在赋值之前是 Nothing。

知识补充：当无法把握数据变量应该使用何种类型时，可用 VBA 提供的通用类型——变体型 Variant；如果对一个对象变量该声明为何种类型感到迷茫，可用 VBA 提供的另一个通用类型——Object，它可以代替任意对象类型。

5.3.2　对变量赋值

对数据变量赋值用 Let 关键字，不过由于允许忽略 Let 关键字，因此在实际工作中对数据变量赋值采用的是“变量名称 = 值”方式。

对对象变量赋值用 Set 关键字，因此假设需要将第二个工作表赋值给变量 Sht，那么应按以下方式编写代码：

```
Sub test()
  Dim sht As Worksheet     '声明一个工作表对象变量
```

```
  Set sht = Worksheets(2)  '将第 2 个工作表赋值给变量
  '...更多代码...
End Sub
```

当使用 Set 为变量 Sht 赋值后，变量 Sht 就代表 Worksheets(2)对象，拥有 Worksheets(2)对象的一切方法和属性。此时直接使用对象变量去参与各种运算即可。

需要特别注意的是，对对象变量赋值时只能用对象，而不是对象的某个属性。以下演示 3 种错误的赋值方式：

```
Sub test()'代码存放位置:模块中
  Dim rng As Range, sht As Worksheet    '声明一个单元格对象变量，一个工作表对象变量
  Set rng = Cells(1, 1).Value           '将 A1 的值赋予变量 rng
  Set sht = Worksheets(2).Name          '将第 2 个工作表的名称赋予变量 sht
  Set sht = Worksheets(3).Cells         '将第 3 个工作表的所有单元格赋予变量 sht
End Sub
```

其中第一句代码的含义是将 A1 单元格的值赋予变量 rng，由于单元格的值属于数据，单元格本身才是对象，因此应去掉“.Value”部分；第二句的错误原因与第一句相同；第三句代码中的“Worksheets(3).Cells”尽管也是对象，但是它属于 Range 对象，不是 Worksheet 对象，因此赋值时会产生“类型不匹配”错误。

本例文件参见光盘：..\第五章\5-5 三种错误的对象变量赋值方式.xlsm

5.3.3 使用对象变量的优势

调用对象变量去参与运算比直接使用对象参与运算有诸多优势，主要体现在以下 3 个方面。

1. 简化书写方式

Worksheets(2)代表第 2 个工作表对象，它包含 13 个字符，而变量 sht 仅有 3 个字符，当代码中需要多次调用这个对象时，显然采用变量会提升代码的录入速度和准确度，代码越长则出错的几率越高。

2. 提升执行效率

Excel VBA 调用对象变量比调用对象的速度更快，假设在一段程序中需要反复调用某个对象时应使用对象变量，而非每次都调用对象去参与运算。以下两个过程的功能一致，其中第二个过程使用了对象变量，直接用对象变量去参与运算，因此效率比第一个过程提升了 3～4 倍。

```
'使用 Range("a1")对象参与运算
Sub 直接引用对象()'代码存放位置:模块中
  Dim tim1 As Date, tim2 As Date, item As Long, Length As Byte    '声明变量
  tim1 = Timer                          '将当前时间赋予变量 tim1
  For item = 1 To 1000000               '循环执行 1000000 次
    Length = Len(Range("A1"))           '计算 Range("a1")的字符长度
  Next item
  tim2 = Timer                          '计算当前的时间
  MsgBox Format(tim2 - tim1, "程序执行时间为: 0.000 秒") '根据两个时间的差值得到程序执行时间
End Sub
'使用变量代替 Range("a1")对象参与运算，效率约为前一个过程的 3～4 倍
```

```
Sub 直接引用对象 2()'代码存放位置:模块中
  Dim tim1 As Date, tim2 As Date, item As Long    '声明变量
  Dim Length As Byte, rng As Range '声明变量
  tim1 = Timer                          '将当前时间赋予变量 tim1
  Set rng = Range("a1")                 '将 A1 单元格赋值给变量 rng
  For item = 1 To 1000000               '循环执行 1000000 次
    Length = Len(rng)                   '计算变量所代表的单元格的字符长度
  Next item
  tim2 = Timer                          '计算当前的时间
    MsgBox Format(tim2 - tim1, "程序执行时间为: 0.000 秒")'根据两个时间的差值得到程序执行时间
End Sub
```

以上两段代码仅用于比较时间差异，对于代码中使用到的循环语句 For Next 在本书的第 7 章中会详细讲解，此处知道变量的价值即可。

本例文件参见光盘：..\第五章\5-6 使用对象变量提升程序效率.xlsm

在引用对象时，对象的层级数量会影响引用对象的时间，对象的书写方式越复杂、层级越多则效率越低。若改用对象变量，可以缩短更多的执行时间。

在随书光盘的“5-6 使用对象变量提升程序效率.xlsm”文件中有相应的代码做比较，此处不再罗列代码，读者打开工作簿后进入 VBE 界面即可看到代码。

3. 提供属性与方法列表

引用对象集合时，VBA 会弹出对象集合的属性与方法列表，开发者可以从列表中选择单词的方式代替手工录入代码，这样既提升速度又确保准确度。然而引用部分对象时不会产生提示信息，例如在 VBE 中录入“worksheets(1).”，VBA 不会弹出属性与方法列表，这给编程工作带来了障碍。如果将对象赋予对象变量，然后录入变量名称加小圆点则可以弹出属性与方法列表。图 5.3 和图 5.4 分别说明了两种引用方式的效果，明显后者对编程更有帮助。

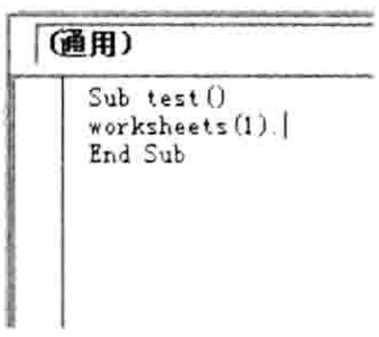


图 5.3　录入对象时无提示

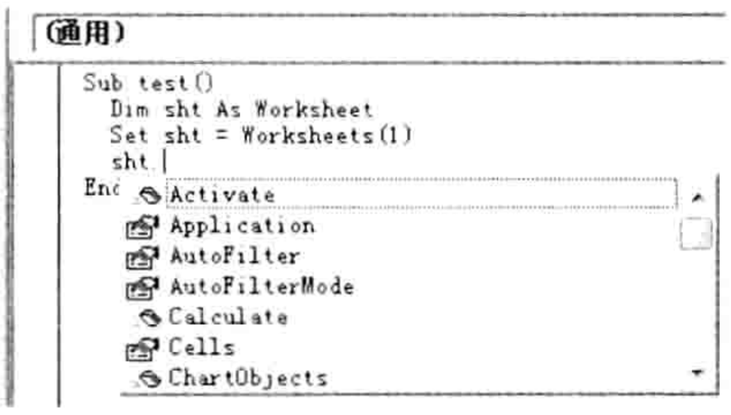


图 5.4　录入对象变量时有提示

知识补充：录入变量名称和小圆点后弹出的属性与方法列表受声明变量时所用的类型影响，由于图 5.4 中变量的类型是 Worksheet，因此弹出了与工作表相关的属性与方法列表。假设声明变量时采用 Range 类型，那么将弹出与单元格对象相关的属性与方法列表。假设声明变量时使用了 Object 或者 Variant 类型，那么将不会弹出属性与方法列表。

5.4　声明常量

常量是相对于变量而言的，它是在程序执行中不产生变化的量。

常量与变量相比，没有那么重要，使用频率也更低，但是它仍然有存在的必要性。

5.4.1 常量的定义与用途

常量也称常数，在帮助中 Excel 对常量的定义是“执行程序时保持常数值的命名项目。常数可以是字符串、数值、另一个常数、任何算术运算符或逻辑运算符的组合（除乘幂与 Is 之外的）”。更通俗地讲，常量就是在程序执行过程中永远不变的量。

“你我他”是一个常量，“Ace”是一个常量，128 也是一个常量。

常量的用途主要体现在以下两方面。

1. 简化输入

在某个程序中需要多次使用一个数值 3.141 592 6 时，如果每次都录入这个长长的数值会降低工作效率，同时也有录入错误的潜在危险，例如漏掉了一位数或者多了一个小数点。

通常的解决办法是定义一个常量 P，对常量赋值为 3.141 592 6，后续直接使用 P 参与运算即可，结果完全相同，但在录入时却轻松许多。

2. 方便识别

仍用前一个实例来说明，在程序中大量运用数值 3.141 592 6，而程序的终端用户却不一定知道这个数值代表什么。如果我们在代码中定义一个常量“圆周率”，并对其赋值为 3.141 592 6，则可以让用户顾名思义。

5.4.2 常量的声明方式

声明变量采用 Dim，Public，Private，Static 等语句，而声明常量必须使用 Const 语句来完成。

声明常量的语法如下：

```
[Public | Private] Const constname [As type] = expression
```

当在过程中声明过程级常量时，直接使用 Const 指定常量的名称和值即可；若要声明模块级的常量，则在 Const 之前添加 Private，而声明工程级常量，需要在前面添加 Public，具体见表 5-5。

表 5-5 Const语句的使用说明

级　别	作 用 域	声明方式	存放位置	举　例
过程级	当前过程	Const	过程中	Const City As String = "乌鲁木齐"
模块级	当前模块	Private Const	模块顶部	Private Const City As String = "乌鲁木齐"
工程级	当前工程	Public Const	模块顶部	Public Const City As String = "乌鲁木齐"

其中“As type”部分是可选的，表示允许在声明常量时不指定其类型。

和变量一样，不指定类型时则是变体型，会耗用更多的资源，影响代码的效率。

声明常量时必须为常量赋值。

以下有 3 种常见的常量声明方式：

```
Const 圆周率 As Double = 3.1415926
Const 圆周率 = 3.1415926
Public Const 圆周率 As Double = 3.1415926
```

第一句指定了常量名称和常量类型；第二句未指定常量类型；第三句使用了 Public，因此代码只能放在模块顶部。如果在过程中通过 Public 声明常量将会产生编译错误。正确的声明工程级常量的代码如下：

```
Public Const 圆周率 As Double = 3.1415926 '在模块顶部声明工程级的常量
Sub test()
  MsgBox 圆周率'在过程中调用常量
End Sub
```

5.4.3 常量的命名规则

常量有两类：内置常量和用户自定义常量。

其中内置常量是 Excel 定义的，用户直接调用即可。例如：xlLandscape，xlPortrait，vbCancel，vbYes，xlDown 和 xlup 等。其中 xlLandscape 和 xlPortrait 两个常量分别代表 2 和 1，用于设定页面的方向是横向还是纵向的。在代码中既可以调用这两个常量去参与运算，也可以直接用 2 或者 1 去参与运算，因为 xlLandscape 等于 2，xlPortrait 等于 1。

自定义的常量由开发者指定常量名称。

常量的命名规则和变量的命名规则完全一致，具体请参考变量命名规则。

下面列举一些不符合要求的常量声明方式：

Const 12st as Byte——数值开头，未赋值；

Const ?asc as Integer——问号开头，未赋值；

Const dim As Byte = 11——不能使用 Dim；

Const sub as String = "中国"——使用了保留字作为常量名称；

Const "ABC" As String="中国"——常量名称不能包含引号。

◆ ◆ ◆

第 6 章　深入剖析常见对象的引用方式

Excel 制表工作就是频繁地操作各种对象，因此对象对于 Excel VBA 而言是极其重要的，几乎每一段程序中都会出现对象。尽管也可以刻意编写出几段不含任意对象的 VBA 代码，然而那已经偏离了 Excel 制表软件的初衷。

本章首先和读者一起回忆一下对象相关的概念，然后在此基础上延伸开，向读者展示单元格、图形对象、工作表、工作簿等常用对象的引用方式。

当本章和第 7 章“常用语句的语法剖析”的知识点都掌握好后，Excel VBA 的筑基过程就算完成了，剩下的将是综合应用，读者要从应用中去总结经验、完善思路。

6.1　关于对象

在本书的第 4 章已经介绍过对象、对象的属性和对象的方法三方面的概念，读者们对于对象应该已经有初步的认知。本章在该基础之上重点向读者介绍常用对象的引用方式，展示其中的一些技巧。不过在介绍引用方式之前有必要复习一下对象相关的基本概念，从而可以让读者更顺利地吸收新的知识。

6.1.1　对象的结构

Excel 有数百类对象，它们之间有着层次分明的结构，就像公司职员的组织架构一样。

然而在日常工作中需要接触的对象类别并不多，一般不超过 10 个。如图 6.1 所示是常用对象的基本结构。

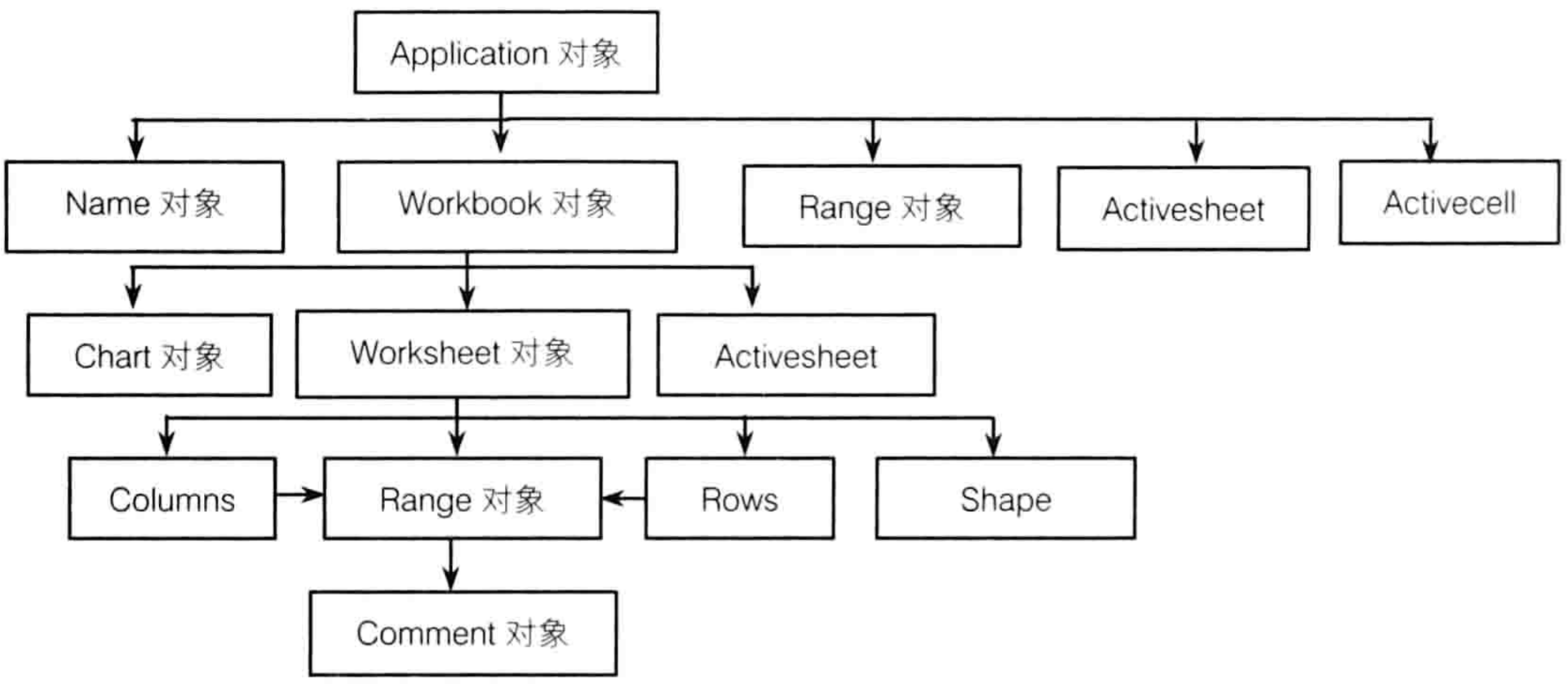


图 6.1　Excel 常用对对象的基本结构

由于对象有着严格的层级关系，Excel VBA 要求引用对象时也按层级关系引用对象。例如引用工作簿“财务.xlsx”中“一月报表”的 A1 单元格可以用以下代码：

```
Application.Workbooks("财务.xlsx").Worksheets("一月报表").Range("a1")
```

其中顶层的对象 Application 允许被忽略。

假设“财务.xlsx”是活动工作簿，那么代码可以简写为：

```
Worksheets("一月报表").Range("a1")
```

假设“一月报表”是活动工作表，那么代码可以继续简写为：

```
Range("a1")
```

6.1.2　对象与对象的集合

Excel 的部分对象有对象与对象集合之分，对象集合包含了同类的所有对象，采用数组形式表示。以下罗列几个常见的对象集合：

Worksheets——工作表集合，代表工作簿中的所有工作表；

Cells——单元格集合，代表工作表中的所有单元格；

Shapes——图形对象集合，代表工作表中的所有图形对象。

对象包含于对象集合中，但是对象和对象集合各自拥有自己的方法和属性，它们之间不相通。例如对象集合有 Count 属性，表示对象集合的数量，而单个对象是没有 Count 属性的。

6.1.3　引用集合中的单一对象

当需要引用对象集合中的单个对象时，可通过参数指定对象的序号或者名称，从而引用集合中的指定对象。例如引用第二个工作簿中的第 4 个工作表时可以用以下代码：

```
Workbooks(2).Worksheets(4)
```

当序号小于 1 或者大于对象集合的总数量时，执行代码时会产生“下标越界”错误。

如果以名称引用对象，那么应在名称前后添加双引号。例如引用“生产报表.xls”工作簿对象应用以下代码：

```
Workbooks("生产报表.xls")
```

VBA 允许使用变量或者常量作为参数去引用对象，例如：

```
Sub 计算指定工作簿中的工作表数量()                    '代码存放位置:模块中
  Const WbName As String = "生产报表.xls"     '声明常量
  '以常量作参数引用指定工作簿，然后计算它的工作表数量
  MsgBox Workbooks(WbName).WorkSheets.Count
End Sub
```

6.1.4　父对象与子对象

父对象是指某个对象的上一层对象，在 VBA 中使用 Parent 来表示父对象。例如要获取工作表中第 2 个批注所在单元格的地址，可以用以下代码实现：

```
MsgBox ActiveSheet.Comments(2).Parent.Address
```

代码中的“ActiveSheet.Comments(2)”表示活动工作表中第 2 个批注；“Parent”代表父对象，即批注所在单元格；“Address”代表单元格地址。如果工作表中的批注少于两个，执行代码将会出错。

子对象是指某个对象的下一层对象。对象集合中的单个对象不是对象集合的子对象，必须是上下级关系才算子对象。

工作簿是 Excel 应用程序对象的子对象，工作表是工作簿的子对象，单元格是工作表的子对象……但 Excel 的对象之间并非都是这种单线式的层级关系，部分对象可能有多个父对象，或者说

多个对象拥有同一个子对象。

例如任意工作表都有单元格这个子对象，活动工作表也有单元格子对象；行对象、列对象、Excel 应用程序对象都有单元格这个子对象。

6.1.5 活动对象

活动对象是当前可以直接操作的对象，例如安装了多个打印机，直接单击打印按钮就能调用的打印机是活动打印机，VBA 中用 ActivePrinter 表示。

Excel 常用的活动对象如表 6-1 所示。

表 6-1 常用活动对象一览

活动对象	书写方式
活动工作簿	ActiveWorkbook
活动工作表	ActiveSheet
活动单元格	ActiveCell
活动窗口	ActivveWindow

6.2 对象的简化引用

Excel 的对象之间皆有层级关系，因此在引用下级对象时可能需要比较长的代码。当面对名称比较长而且需要反复引用时，应该寻求简化的引用方式。

本节向读者展示两种简化引用方式的新技术。

6.2.1 使用对象变量

在引用对象时，小圆点越多，引用的时间就会越长，书写出错的几率也越大。正确的做法是将被引用的对象赋值给对象变量，然后使用这个变量代表对象参与各种运算和操作。

例如以下的“设置序号”过程可以在 A2:A16 的区域中生成如图 6.2 所示的大写序号。其中代码“Workbooks("生产表.xlsm").Sheets("一月").Range("a2:a16")”重复出了 3 次：

```
Sub 设置序号()'代码存放位置:模块中
  Workbooks("6-1 使用对象变量和 With 语句简化引用.xlsm").Sheets("一月
").Range("a2:a16").FormulaR1C1 = "=COUNTA(R1C1:R[-1]C)" '设置公式
  Workbooks("6-1 使用对象变量和 With 语句简化引用.xlsm").Sheets("一月
").Range("a2:a16").NumberFormatLocal = "[DBNum2][$-804]G/通用格式" '设置单元格
的数字格式
  Workbooks("6-1 使用对象变量和 With 语句简化引用.xlsm").Sheets("一月
").Range("a2").EntireColumn.AutoFit  '自动调整列宽
End Sub
```

如果改用对象变量来简化过程，应按以下方式编写：

```
Sub 设置序号2()        '代码存放位置:模块中
  Dim rng As Range   '声明一个对象变量
  Set rng = Workbooks("6-1 使用对象变量和 With 语句简化引用.xlsm").Sheets("一月
").Range("a2:a16")   '将 Range 对象赋值给变量
  rng.FormulaR1C1 = "=COUNTA(R1C1:R[-1]C)"          '设置公式
```

```
    rng.NumberFormatLocal = "[DBNum2][$-804]G/通用格式"      '设置单元格的数字格式
    rng.EntireColumn.AutoFit                                  '自动调整列宽
End Sub
```

很显然，使用变量简化程序后不仅代码更简短，程序的执行效率也更高。

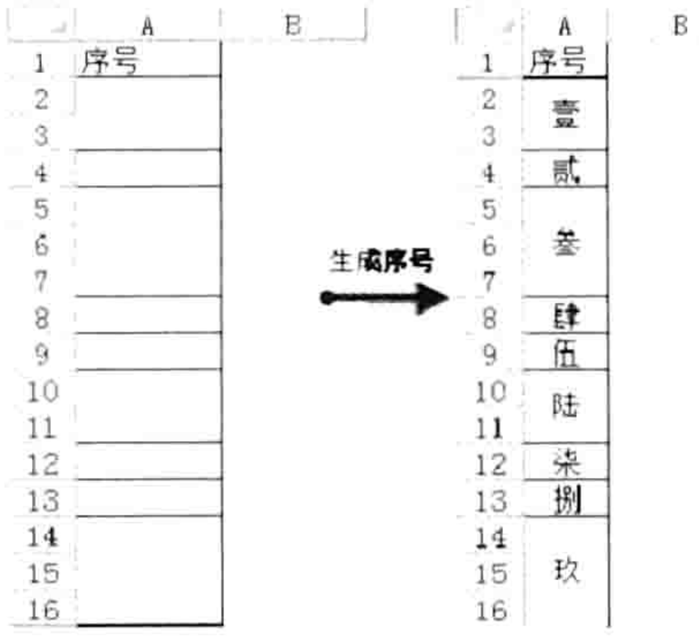


图 6.2　生成序号前后效果比较

本例文件参见光盘：..\第六章\ 6-1 使用对象变量和 With 语句简化引用.xlsm

过程的 FormulaR1C1 属性代表单元格中 R1C1 样式的公式，可以通过此属性向单元格中插入公式。所谓的 R1C1 样式是单元格相对引用的一种形式，其中的 R 代表行、C 代表列，R 和 C 后面的数字相当于公式所在单元格的行偏移量和列偏移量。当偏移量是负数时需要使用方括号“[]”将它括起来，当偏移量是 0 时可以忽略不写。

例如要求在 B4 单元格录入公式“=a1”，由于 A1 单元格相对于 B4 单元格的行偏移量为-3，列偏移量为-1，因此代码如下：

```
Range("b4").FormulaR1C1 = "=r[-3]c[-1]"
```

执行代码后可得到如图 6.3 所示效果。

假设要求在 B4 单元格中录入公式“=sum(b1:b3)”，那么应使用以下代码：

```
Range("b4").FormulaR1C1 = "=sum(r[-3]c:r[-1]c)"
```

代码是“r[-3]c”代表 B1 单元格，“r[-1]c”代表 B3 单元格，因此“r[-3]c:r[-1]c”代表 B1:B3”区域。代码的执行结果如图 6.4 所示。

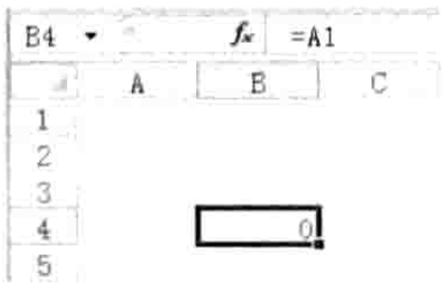


图 6.3　效果一

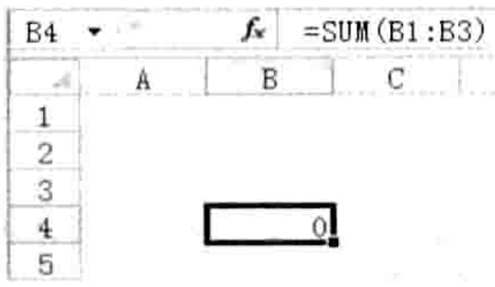


图 6.4　效果二

事实上，对单元格录入公式（Range.FormulaR1C1 属性），以及设置单元格的格式（Range.NumberFormatLocal 属性）、自动调整列宽（Range.AutoFit 方法）等都可以录制宏产生代码，不需要花时间记准它们的语法，更不需要背下这些代码。

6.2.2　使用 With 语句

With 语句是专门用于简化引用并提升效率而存在的，当某个对象重复出现时就有必要使用 With 语句。With 语句的语法如下：

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

其中 object 代表对象，statements 代表针对 object 而执行的一条或多条语句。鉴于 With 语句的存在价值包含简化代码，因此在实际工作中 statements 必定包括多条语句。

以上一个案例中的“设置序号”过程为例，使用 With 语句简化后的过程如下：

```
Sub 设置序号 3()     '通过 With 简化程序（代码存放位置：模块中）
  '利用 With 语句引用对象
  With Workbooks("6-1 使用对象变量和 With 语句简化引用.xlsm").Sheets("一月
").Range("a2:a15")
    .FormulaR1C1 = "=COUNTA(R1C1:R[-1]C)"                '对当前引用的对象设置公式
    .NumberFormatLocal = "[DBNum2][$-804]G/通用格式"     '设置单元格的数字格式
    .EntireColumn.AutoFit                                '自动调整列宽
  End With
End Sub
```

需要注意的是，以上过程中 With 和 End With 之间的三句代码都是基于同一个对象，三句代码都必须以小圆点开头。

With 语句允许嵌套使用，即里层的 With 语句所引用的对象是外层 With 语句引用的对象的子对象。以下过程的功能是对活动工作表重命名、移动位置，并且在其中查找“中国”，将找到后的单元格的字体设置为红色、加粗、倾斜，最后在状态栏中显示值为“中国”的单元格的地址。过程使用了 With 语句嵌套，里层 With 引用的对象“Cells.Find("中国").Font”是外层 With 语句所引用对象的子对象，因此在它之前需要添加小圆点。

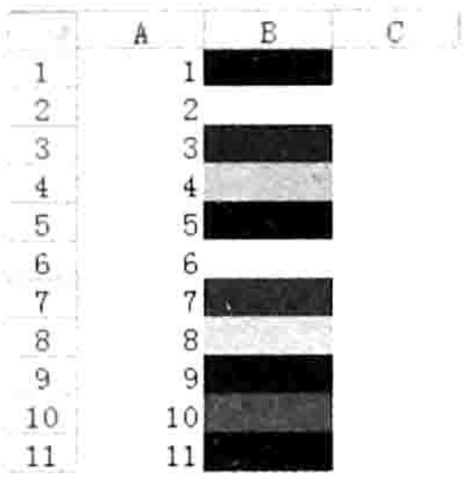
```
Sub test()'代码存放位置：模块中
  With ActiveSheet                          '引用活动工作表
    .Name = "XX 公司" & .Name               '对引用的名称前添加“XX 公司”
    .Move before:=Worksheets(1)             '将引用对象移动到第一个工作表之前
    With .Cells.Find("中国").Font           '引用活动工作表中值为“中国”的单元格的字体
      .ColorIndex = 3                       '将字体的颜色编号设置为 3（即红色）
      .Bold = True                          '将字体的加粗属性赋值为 True
      .Italic = True                        '将字体的倾斜属性赋值为 True
      Application.StatusBar = .Parent.Address '在状态栏显示被引用对象的父对象的地址
    End With
  End With
End Sub
```

本例文件参见光盘：..\第六章\6-2 With 语句的嵌套应用.xlsm

以上代码中 ColorIndex 代表单元格的字体颜色地址编号，当对它赋值为 3 时代表红色。可以用以下代码获得 0～56 个颜色与颜色编码之间的对应关系（大于 56 时会出错），结果如图 6.5 所示。

```
Sub 颜色编码与颜色的对应关系()   '代码存放位置：模块中
  Dim i As Byte                  '声明一个 Byte 型的变量
  For i = 1 To 56                '从 1 循环到 56
    Cells(i, 1) = I              '在第 i 行第 1 列的单元格中输出编号
    Cells(i, 2).Interior.ColorIndex = i '在第 i 行第 2 列的单元格中显示编号为 i 的颜色
```

```
    Next
End Sub
```



	A	B	C
1	1		
2	2		
3	3		
4	4		
5	5		
6	6		
7	7		
8	8		
9	9		
10	10		
11	11		

图 6.5　颜色编码与颜色的对应关系

本例文件参见光盘：..\第六章\6-3 颜色编码与颜色的对应关系.xlsm

6.3　单元格对象

Excel 中有很多对象，最常用的当属单元格对象，它是数据最基本的载体。而 VBA 中对单元格的表示方法也比其他对象更多、更复杂，本节专门讲述单元格与区域的表示方法。

单元格的最基本的表示方式有三种：Range("a1")方式、cells(1,1)方式和[a1]方式，此外还有交集、合集、偏移量、已用区域、当前区域、End 等单元格引用的相关概念。

6.3.1　Range("A1")方式引用单元格

用 Range 可以将文本型的单元格地址转化为单元格对象引用，类似于工作表函数 Indirect。它可以引用单元格、区域、整行、整列及工作表中的所有单元格。

1. 引用单元格

Range 引用单元格对象的方式为：单元格的列标加行号作为参数，并且加入引号。例如：

Range ("A1")——表示 A1 单元格；

Range ("C25")——表示 C25 单元格；

Range ("ZZ1048576")——表示 ZZ1048576 单元格，在 Excel 2003 中这是无效的引用，因为 Excel 2003 的最大行不超过 65 536 行，最大列不超过 IV 列。

Range 参数中的引号必须是在半角状态下输入的，否则必将产生编译错误。

2. 引用单元格区域

Range 引用单元格区域时是利用要引用的区域中左上角单元格地址加冒号再加右下角单元格地址为其参数。不过参数也可以写成右下角单元格地址加冒号再加左上角单元格地址，VBA 会自动将其转换成左上角单元格地址加冒号再加右下角单元格地址的形式。

例如使用以下两种方式引用单元格区域都可以得到相同结果：

```
MsgBox Range("A2:D1").Address
MsgBox Range("D1:A2").Address
```

以下是单元格区域引用的三个案例：

Range ("A1:V10")——代表引用从 A1 到 V10 的矩形区域，包括 220 个单元格；

Range("F2:F10000")——代表引用从 F2 到 F10000 的矩形区域，包括 9999 个单元格；

Range("D2:ZZ10000")——代表引用从 D2 到 ZZ10000 的矩形区域，包括 6 989 301 个单元格，

不过它在兼容模式的工作簿中是不合法的引用，因为兼容模式的工作表最末列是 IV 列。

区域和单元格的默认属性都是 Value，但是区域的 Value 是一个数组，包括多个值，在 VBA 中无法直接将其显示在屏幕上。如果利用 Msgbox 函数来显示区域的属性值将会产生如图 6.6 所示的运行错误。如果需要将区域中每个单元格的值都显示在信息框中，需要使用循环语句将每一个单元格的值串连起来，形成一个字符串，在本书的第 7 章中会讲述循环语句的语法和应用。

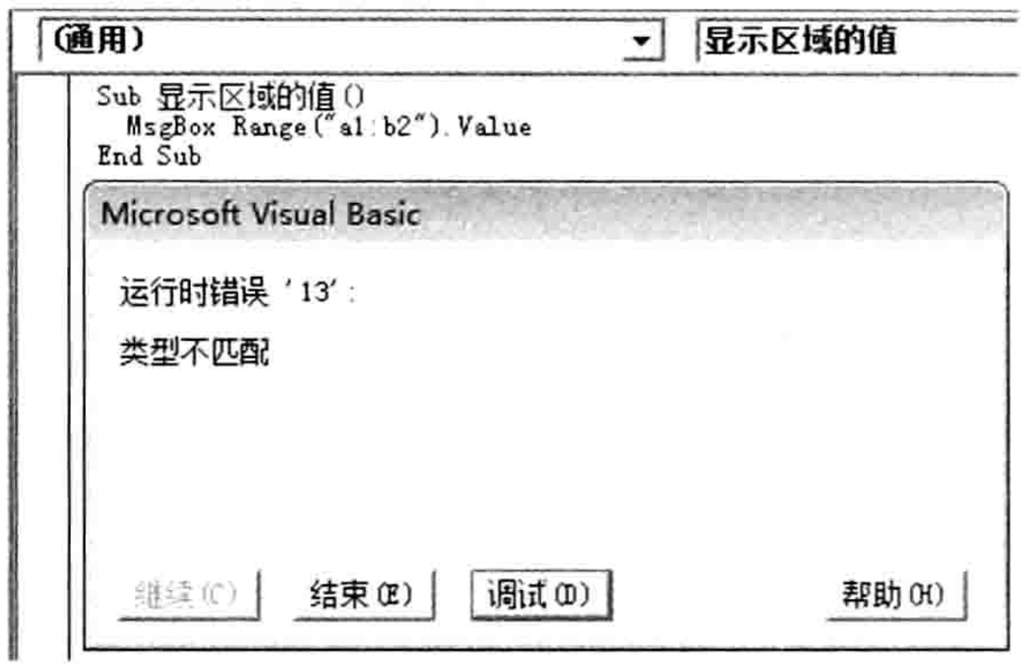


图 6.6　引用区域默认属性时产生运行时错误

不过可以通过参数引用区域中的单个单元格的值，例如：

Range("a1:b2")(1)——代表 a1:b2 区域中第 1 个单元格，即 a1；

Range("a1:b2")(2)——代表 a1:b2 区域中第 2 个单元格，即 b1；

Range("a1:b2")(3)——代表 a1:b2 区域中第 3 个单元格，即 a2；

Range("a1:b2")(4)——代表 a1:b2 区域中第 4 个单元格，即 b2。

也就是说，索引号代表区域中从左到右、从上到下的序号，它以区域左上角单元格为参照点进行相对引用。

事实上，引用区域中的单个单元格时也可以使用双索引号，第一参数表示行的索引，第二参数表示列的索引。那么参数“(4,5)”就可以引用区域第 4 行中第 5 列的单元格，它以区域左上角单元格为参照，而非以工作表中 A1 单元格为参照。

MsgBox Range("D3:F7")(1, 3).Address——结果为“$F$3”，表示 D3:F7 区域第 1 行中第 3 列的单元格地址；

MsgBox Range("D3:F7")(4, 2).Address——结果为“$E$6”，表示 D3:F7 区域第 4 行中第 2 列的单元格地址；

MsgBox Range("D3:F7")(9, 4).Address——结果为“$G$11”，即 D3 单元格向下偏移到第 9 行，再向右偏移到第 4 列。虽然其行数与列数都已超过区域的大小，仍然可以正确地引用单元格。

Range 的参数也支持表达式，即字符或者数值运算的结果。例如：

Range("F" & 3 + 2)——表示引用 F5 单元格；

Range("F" & Range("D5").Value)——表示列标为 F、行号等于 D5 单元格中的值的单元格。

还可以使用变量作为参数，例如：

Range("D" & i)——表示列标为 D，行号为变量 i 的值决定的单元格，变量 i 的值不能是小于等于 0 或者大于工作表的总行数，否则会引用失败。

3. 引用多区域单元格

将单元格或者单元格区域的地址以半角的逗号为分隔符串连成字符串，然后作为 Range 的参数，可以引用不连续的多个单元格区域。例如以下引用方式：

Range("D3,F7")——表示 D3 和 F7 两个区域，包括两个单元格；

Range("D3:F4,G10")——表示 D3:F4 和 G10 两个区域，包括 7 个单元格；

Range("A1,B3:F4,Z1:ZB2")——表示 A1、B3:F4 和 Z1:ZB2 3 个区域，包括 1317 个单元格。

此方式引用单元格区域有限制——参数的长度不能超过 256 个字符，否则运行时将会产生错误。

4. 引用整行、整列单元格

利用“行号:行号”作为参数时可引用整行单元格，同理利用“列标:列标”作为参数时可引用整列单元格，如果两个行号或者两个列标不一致时，可以引用多行或者多列。以下是引用案例：

Range("2:2")——表示引用第 2 行；

Range("2:10")——表示引用第 2 到第 10 行；

Range("D:d")——表示引用第 D 列，列标不区分大小写；

Range("D:Z")——表示引用从 D 列开始、Z 列结束的区域；

Range("D:A")——表示引用 A 列到 D 列，顺序不一致时，VBA 会自动转换成升序格式。

参数中的冒号可以用半角也可以用全角，VBA 会将全角冒号转成半角冒号。但是引号却只能使用半角的，否则将产生编译错误。

整行、整行引用对象除了使用 Range 方法外，还可以用 Rows 和 Columns 来完成。其中 Rows 引用行，以阿拉伯数字作为参数；Columns 引用列，既可用阿拉伯数字也可用列标做参数。

Rows(2)——表示引用第 2 行；

Rows("2")——同样表示引用第 2 行；

Rows("2:2")——仍然表示引用第 2 行；

Rows("2:4")——表示引用第 2 到第 4 行；

Columns(2)——表示引用第 2 列，相当于 Range("B:B")；

Columns("B")——同样表示引用第 2 列；

Columns("B:B")——仍然表示引用第 2 列；

Columns("B:D")——表示引用第 2 到第 4 列。

如果不带参数，那 Rows 代表整个工作表所有行，包括 17 179 869 184 个单元格。而 Columns 代表整个工作表所有列，也包括 17 179 869 184 个单元格。

5. Range 嵌套使用

除上面的四种方法外，Range 还支持利用单元格作为参数，其具体语法为：

```
Range(Cell1, Cell2)
```

其中 Cell1 和 Cell2 是必选参数。Cell1 用于指定目标区域的左上角单元格，Cell2 用于指定目标区域右下角单元格。如果参数为一个或者三个单元格则产生编译错误。

以下为 Range 嵌套引用的案例：

Range(Range("A1"), Range("D2"))——表示引用 A1:D2 区域，包含 8 个单元格；

Range(Range("A4"), Range("A100"))——表示引用 A4:A100 区域，包含 97 个单元格。

6.3.2　Cells(1,1)方式引用单元格

利用 Cells 引用单元格有三种用法。

1. WorkSheet.Cells（横坐标,纵坐标）

引用某工作表中行、列坐标所指定的单元格，可以使用本方式，基本语法为：

```
[WorkSheet].Cells([RowIndex],[ColumnIndex])
```

其中工作表对象 WorkSheet 可选，行坐标 RowIndex 与列坐标 ColumnIndex 也可选。

当行坐标与列坐标皆为 1 时表示引用工作表左上角的 A1 单元格。

MsgBox Cells(1,1).Address——计算结果为“$A$1”。

以上代码中忽略了工作表对象，因此表示引用活动工作表中的 A1 单元格；如果忽略横坐标与纵坐标，则表示引用所有单元格。

以下为 Cells(1,1)形式的单元格引用案例：

WorkSheets(1).Cells(5,4)——表示引用第 1 个工作表中行坐标为 5、列坐标为 4 的单元格 D5；

WorkSheets("生产表").Cells(10000, 1000)——表示引用“生产表”中的 ALL10000 单元格。在兼容模式的工作簿中执行代码将会出错，因为其最大列为 256，不足 1000 列。

2. WorkSheet.Cells（行号,列标）

本引用方式依靠目标地址的行号与列标来确定目标单元格。其中行号与列标两个参数都是必选参数。而工作表对象 Worksheet 则是可选参数。

以下三个引用为合法的单元格对象引用：

WorkSheets("生产表").Cells(2, "C")——表示引用“生产表”中 C2 单元格；

Cells(12, "ZZ")——表示引用当前工作表中 ZZ12 单元格；

Cells("12", "ZZ")——仍然表示引用 ZZ12 单元格，其中行号并非一定要使用双引号，但列标一定要使用双引号。

本方法永远只能引用一个单元格，不能引用区域。

3. Range.Cells（横坐标,纵坐标）

本方式引用单元格是以 Range 对象的横向坐标与纵向坐标的交叉点为目标，它有别于“WorkSheet.Cells（横坐标,纵坐标）”方式，坐标的计算方式相同，但是参照点不同。

Range("B2:G10").Cells(2, 2)——表示 B2:G10 单元格中横坐标为 2、纵坐标为 2 的单元格 C3。图 6.7 展示了目标单元格与 Range("B2:G10")的关系。

其中黄色单元格相对于工作表的横、纵坐标分别为 3 和 3，但对于 B2:G10 区域，其横、纵坐标则为 2 和 2。

Cells 的参数还可以使用小数，不过 VBA 会将其进行四舍五入后再计算坐标。例如：

Range("B2:G10").Cells(1.5, 4.4)——表示引用 B2:G10 区域中位于第二行、第四列的 E3 单元格。

还可以使用负数或者 0 作为参数，那么其坐标计算方式则向左、向上偏移。例如：

Range("D4:G10").Cells(–1, –1)——表示引用 B2 单元格；

Range("D4:G10").Cells(0, –2)——表示引用 A3 单元格。

如图 6.8 所示是负数坐标的示意图。

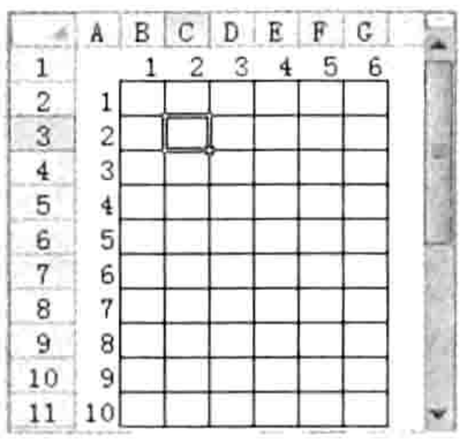


图 6.7　Range("B2:G10").Cells(2, 2)示意图

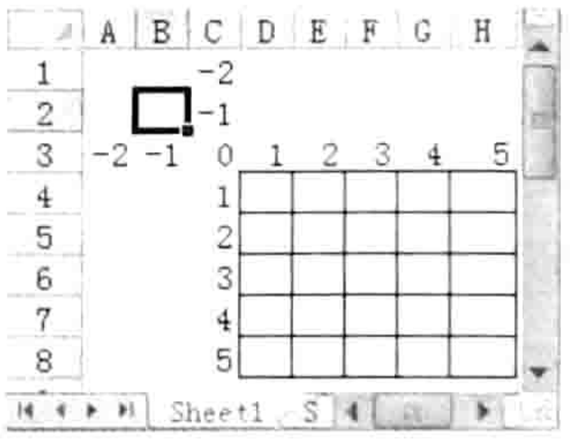


图 6.8　Range("D4:H8").cells(–1,–1)示意图

4. Range.Cells(索引号)

当使用单个索引号作为参数时，它表示父对象中的一个索引子集。其编号方式是先行后列、先左后右。

Range("B2:G10").Cells(5)——表示 B2:G10 区域中第 5 个单元格 F2，从 B2 开始向右数 5 个单位。

Range("B2:G10").Cells(7)——表示 B2:G10 区域中第 7 个单元格 B3，因父对象只有 6 列，那么第 2 行第 1 个单元格即为该区域的第 7 个单元格。

Range("B2:G10").Cells(60)——表示引用单元格 G11。在 B2:G10 区域中仅有 54 个单元格，而参数 60 超过区域的最大个数后，则继续向其下一行开始累加，直到超出整个工作表的边界。

6.3.3　[a1]方式引用单元格

[a1]引用单元格方式是在左、右方括号中直接录入单元格或者区域地址来引用目标单元，单元格地址不区分大小写，也不区分是相对引用还是绝对引用。

[a1]引用单元格方式在写法上占有比较大的优势，而且可以引用单元格、区域、整行、整列等，既书写简单又功能强大。

以下是[a1]方式的单元格引用案例：

[a1]——表示引用单元格 A1；

[B$10]——表示引用单元格 B10；

[D2:F500]——表示引用 D2:F500 区域，包括 1497 个单元格；

[D2,F2]——表示引用 D2 和 F2 两个单元格；

[D2:D3,F2:G10,Z100]——表示引用 D2:D3 和 F2:G10、Z100 三个区域，包括 21 个单元格；

[D2：D3，D5]——表示引用 D2:D3 和 D5 两个区域，中间的冒号和逗号允许使用全角，VBA 会自动将其转换为半角。

以下是错误的单元格引用：

["D2:D3"]——参数不能使用引号；

[A1:F2500000]——行数超过允许的最大值 1 048 576。

6.3.4　Range ("A1")、Cells (1,1) 与[a1]引用单元格方式比较

在实际工作中，Range ("A1")、Cells (1,1) 和[a1]这三种引用单元格方式各有所长。如表 6-2 所示是它们在功能的效率上的一些差异比较。

表 6-2　三种单元格引用方式比较

比较项目 \ 引用方式	Range ("A1")	Cells (1,1)	[a1]
可以引用的对象	单元格、区域、多区域、整行、整列	单元格	单元格、区域、多区域、整行、整列
属性与方法列表	支持	不支持	不支持
用于代码循环	行循环	行循环、列循环	不支持
输入简便性	差	差	好
支持参数	索引号、Item 和 Cells	Item 和 Cells	不支持
引用效率	中	高	低

从以上的比较中可以发现，Cells(1,1)引用方式的优势在于将它用于循环语句比较方便，而且效率极高，缺点是无法引用单元格区域；Range 引用方式的优势在于支持属性与方法列表，支持行循环和参数，缺点是书写时不够方便；而[A1]方式的优势在于书写方便、可以引用单元格区域，缺点是不支持循环和不能弹出属性与方法列表、效率极低。

为了加深读者对表 6-2 的理解程度，下面对其中四点详加说明。

1. 属性与方法列表

对象是否支持属性与方法列表是很重要的一个特点，在编写代码时，可能程序员对某些属性或

者方法不够熟悉，需要借助属性与方法列表来快速录入代码。

在单元格的三种引用方式中，仅 Range("A1")引用方式支持属性与方法列表，在代码窗口中录入“[A1].”或者“Cells（1,1）”后不会有任何反应。

2. 通过参数访问子集

能否通过参数访问区域中的单个对象也是衡量引用方式是否人性化的标准之一。

Range("A1:A10")支持三种方式访问其子集，例如获取 A1:A10 区域的第 2 个单元格，那么有以下三种方式：

```
Range("A1:A10").Item(2)
Range("A1:A10").Cells(2)
Range("A1:A10")(2)
```

而[A1]方式引用单元格时仅仅支持两种方式引用子集。例如：

```
[A1:A10].Item(2)
[A1:A10].Cells(2)
```

Cells(1,1)引用方式只能引用单个单元格，因此不存在子集。

3. 支持循环

所谓的支持循环是指参数是否能配合变量在循环语句中实现累加或者递减。支持循环会给编程带来若大的便利。Range("A1")引用方式支持单循环，例如代码中“Range("A"&变量)”的行号采用变量，当变量配合循环语句实现累加或者递减时，引用的对象也会相应地变化，当变量是 2 时就引用 A2 单元格，变量是 3 时就引用 A3 单元格。

Cells(1,1)引用方式支持双循环，横坐标和纵坐标都允许使用数值变量，在循环语句中它会占有比较大的优势。本书第 7 章中讲述循环语句时展示了 Cells(1,1)的优势。

[a1]引用方式不支持循环。

4. 引用效率

Cells(1,1)引用方式的效率最高，[a1]引用方式的效率最低，前者大约是后者的 20 倍，因此需要反复引用一个单元格去参与某些运算时应采用 cells(1,1)引用方式，如果只引用一次，那么 Range("A1")、Cells（1,1）、[a1]三个引用方式任选，它们之间的差异几乎可以忽略。例如 0.0001 秒和 0.00005 秒完全感受不到有何区别，但是如果引用 10 万次后，它们之间的差值就比较明显了。

6.3.5 Selection 与 ActiveCell：当前选区与活动单元格

在选择了单元格或者单元格区域的情况下，Selection 代表当前选中的所有单元格，通常简称为选区。而 ActiveCell 则表示活动单元格。选区可以是一个单元格，也可以包含多个单元格甚至多个区域，但活动单元格永远只有一个，而且活动单元格一定包含于选区中。

如果选区仅有一个单元格，那么选区与活动单元格完全相同，它们代表相同的对象。

如果需要在选区不变的情况下改变活动单元格，可用 Range.Activate 方法来实现，其语法如下：

```
Range.Activate
```

其中 Range 表示待激活的单元格，只能包含一个单元格，如果超过一个则会在运行时产生错误。

假设让选区中第 4 个单元格成为活动单元格，使用以下语句即可：

```
Selection(4).Activate
```

如果需要让选区中的第 2 列第 3 行的单元格成为活动单元格，使用以下语句即可：

```
Selection.Cells(3, 2).Activate
```

Activecell 和 Selection 都只能引用活动工作表中的单元格，如果在前面加上其他工作表名称，引用必将在运行时产生错误。例如代码“WorkSheets(2).Selection”并不能成功引用第 2 个工作表的选区。

6.3.6　已用区域与当前区域

已用区域是指 UsedRange，其父对象是工作表。
当前区域是指 CurrentRegion，其父对象是单元格。

1. UsedRange

WorkSheet.UsedRange 表示工作表中的已用区域，即用户已经使用过的区域。

例如工作表 Sheet1 中 A1:V2 存放了数据，其他区域一直保持空白，那么 A1:V2 区域是该表的已用区域。如果在 A1:V2 区域之外的 B10 单元格中录入一个数据，那么该表的已用区域应为 A1:V10。

一个工作表只有一个已用区域，该区域是一个矩形、包含了所有已用单元格的区域。

当单元格中有数据时，它无疑属于已用区域的一份子；如果单元格中没有数据但是曾经设置过格式，而且格式信息尚未被清除，那么此单元格也属于已用区域。

图 6.9 中的 B3:B6、D2:F2 单元格区域都有数据，G8 单元格没有数据但有格式信息，它们三者组成的最小矩形区域是 B2:G8,，因此活动工作表的已用区域即为 B2:G8。可用以下代码测试实际的已用单元格区域的大小：

```
MsgBox ActiveSheet.UsedRange.Address
```

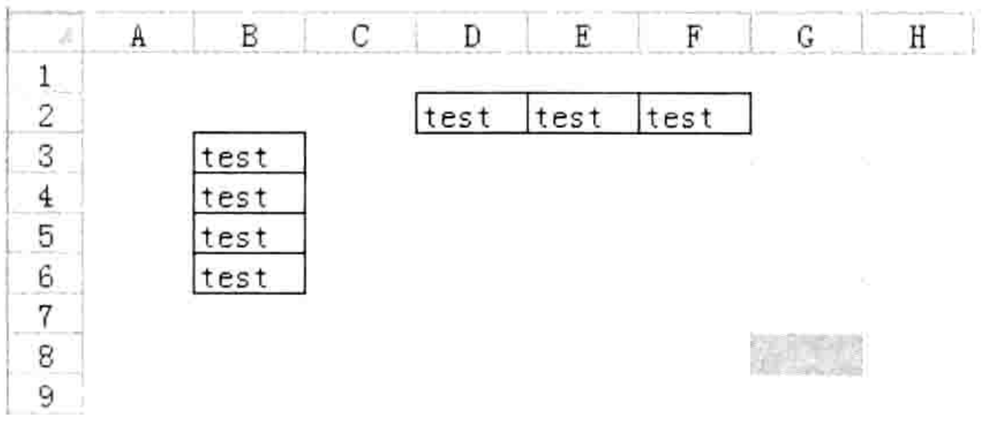


图 6.9　工作表的已用区域 Usedrange

假设工作表中任何单元格都未曾使用过，那么工作表的 UsedRange 是 A1 单元格，而非 Nothing。

2. CurrentRegion

Range.CurrentRegion 表示当前区域。当前区域是指包含指定单元格且以空行、空列组合为边界的区域。与 WorkSheet.UsedRange 相比较，它们的不同点如表 6-3 所示。

表 6-3　已用区域与当前区域的比较

比较项目 \ 比较对象	UsedRange	CurrentRegion
判断标准	是否有格式及有数据	是否有数据及是否与 Range 之间存在空白行、空白列
工作表中的拥有数量	1 个	多个
父对象	Worksheet	Range

每个单元格都拥有自己的 CurrentRegion，如果某个单元格有数据而且与指定对象 Range 之间不存在空行或者空列间隔，那么它就属于该 Range 的 CurrentRegion 成员。

例如在图 6.10 中，工作表中的已用区域和单元格 A1 的当前区域获取方法如下：

```
Sub UsedRange与CurrentRegion()'代码存放位置:模块中
  MsgBox "UsedRange:" & ActiveSheet.UsedRange.Address & Chr(10) _
& "CurrentRegion:" & Range("a1").CurrentRegion.Address, vbInformation, "提示"
End Sub
```

其中 Chr(10)表示编码为 10 的字符，它可以将信息框中的字符串换行，代码的执行结果如图 6.11 所示。

	A	B	C	D	E
1	姓名	成绩		姓名	成绩
2	赵	80		吴	86
3	钱	81		郑	78
4	孙	73		王	67
5	李	72		冯	89
6	周	76		陈	68

Sheet1 Sheet2 Sheet3

图 6.10　成绩表

图 6.11　UsedRange 与 CurrentRegion

Excel 的排序与筛选操作其实就是以 Range.CurrentRegion 为操作对象的，当工作表中有多个不相邻的数据区域时，排序与筛选操作仅针对当前区域有效。

6.3.7　SpecialCells：按条件引用区域

Range.SpecialCells 方法可以返回与指定类型和值相匹配的区域。它的具体参数为：

```
Range.SpecialCells(Type, Value)
```

其中 Range 可以是任意的单元格对象或者区域，Type 参数表示单元格类型，Value 参数是可选参数，只有 Type 参数为 xlCellTypeConstants 或 xlCellTypeFormulas 时才可用。Value 参数用于确定数值类型，它的取值范围是数值、文本、逻辑值和错误值 4 个选项。

其中 Type 参数即为 XlCellType 常量，可选值如表 6-4 所示。

表 6-4　XlCellType常量

XlCellType 常量	含义	值
xlCellTypeAllFormatConditions	包含条件格式的单元格	–4172
xlCellTypeAllValidation	包含有效性验证条件的单元格	–4174
xlCellTypeBlanks	空单元格	4
xlCellTypeComments	含有注释的单元格	–4144
xlCellTypeConstants	含有常量的单元格	2
xlCellTypeFormulas	含有公式的单元格	–4123
xlCellTypeLastCell	已用区域中的最后一个单元格	11
xlCellTypeSameFormatConditions	含有相同条件格式的单元格	–4173
xlCellTypeSameValidation	含有相同有效性条件的单元格	–4175
xlCellTypeVisible	所有可见单元格	12

Value 参数即为 XlSpecialCellsValue 常量，其可选值如表 6-5 所示。

表 6-5　XlSpecialCellsValue 常量可用值详解

XlSpecialCellsValue 常量	含义	值
xlNumbers	数值	1
xlTextValues	文本	2
xlLogical	逻辑值	4
xlErrors	错误值	16

表 6-5 中的 4 项常量允许任意组合，例如用 23 代表所有值，用 6 代表文本与逻辑值。参数的默认值是所有值，也就是说如果忽略第 2 参数时相当于使用 23 作为参数。

根据表 6-4 所示，引用 Range ("A1:G10")区域中的空白单元格可以使用以下代码：

```
Range("A1:G10").SpecialCells(xlCellTypeBlanks)
```

删除 Range("A1:G10")区域中带有批注的单元格可以使用以下代码：

```
Range("A1:G10").SpecialCells(xlCellTypeComments).Delete
```

而引用所有带有公式的单元格则用以下代码：

```
Range("A1:G10").SpecialCells(xlCellTypeFormulas, 23)
```

如果 SpecialCells 的参数所指定的单元格不存在，那么引用结果并非 Nothing，而会直接弹出错误提示，表示不存在指定类型的单元格。例如选择公式所在区域的代码如下：

```
Sub 选择公式所在区域()'代码存放位置:模块中
  Range("A1:E14").SpecialCells(-4123, 23).Select
End Sub
```

当 A1:E14 区域中无公式时，执行上述代码后会弹出一个错误提示；当 A1:E14 区域中有公式时，执行上述代码后会选中所有公式所在的单元格。

知识提示：使用 Range.SpecialCells 方法引用单元格可以通过录制“定位条件”的宏来完成。在录制宏时可以产生关于 Range.SpecialCells 方法的所有常量名称，用户在不熟悉参数用法的时候可以通过录制宏来获取代码。“定位条件”对话框如图 6.12 所示。

图 6.12　SpecialCells 方法对应的“定位条件”对话框

在图 6.12 中，右下角的“全部”和“相同”都对应两个常量，具体对应哪一个由上面的“条件格式”和“数据有效性”决定。例如选中“条件格式”单选框时，那么“全部”对应的常量是 xlCellTypeAllFormatConditions，而“相同”对应的常量是 xlCellTypeSameFormatConditions。

6.3.8 CurrentArray：引用数组区域

数组公式分单元格数组公式和区域数组公式，数组区域是针对数组公式而言的，指区域数组公式所占用的区域。数组区域有以下两个显著特点。

（1）公式两端有大括号“{}”

在录入数组公式后必须按<Ctrl+Shift+Enter>组合键，录入完成后在公式的首尾会自动产生大括号“｛｝”。如图 6.13 所示即为典型的数组公式。

（2）无法单独编辑其中某个单元格

区域数组公式跨越多个相邻的单元格，用户无法单独编辑其中任何一个单元格，包括修改、删除等。如图 6.14 所示为编辑数组区域中单个单元格时产生的错误提示。

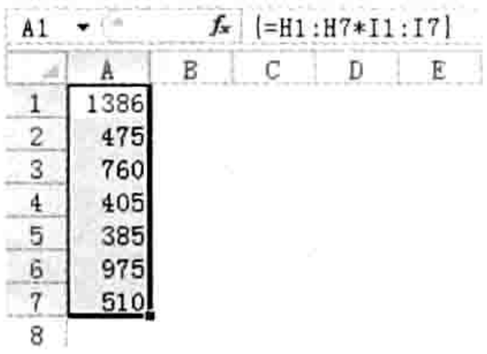


图 6.13　典型的数组公式与数组区域

图 6.14　编辑数组区域中单个单元格时产生的错误提示

如果需要修改区域数组公式，需要先选择整个数组区域，然后对整个区域一并修改。

CurrentArray 代表数组区域，其具体语法为：

```
Range.CurrentArray
```

Range 可以是隶属于数组区域中的任意单元格。例如在图 6.14 中，单元格 A1 的数组区域和 A2 的数组区域完全一致。

如果 Range 不在数组区域中，则引用 Range.CurrentArray 时将产生“未找到单元格”的错误信息。

假设要删除活动单元格中的值，由于活动单元格可能处于数组区域中也可能不处于数组区域中，为了让代码更通用，应按以下方式编写代码：

```
Sub 删除活动单元格的值()    '代码存放位置：模块中
  On Error Resume Next   '当代码出错时，继续执行下一句代码
  ActiveCell.CurrentArray.ClearContents    '删除活动单元格的数组区域的值
  '如果有错误(表示活动单元格未处于数组区域内)，那么只删除活动单元格的值
  If Err.Number > 0 Then ActiveCell.ClearContents
End Sub
```

过程中首先使用“On Error Resume Next”防错语句，避免活动单元格未处于数组区域中时导致代码出错并中断程序，然后使用 Range.ClearContents 方法清除活动单元格的数组区域的值，此时利用条件语句 IF Then 判断程序是否有错误，如果有错误，重新调用代码“ActiveCell.ClearContents”只清除活动单元格中的值。

对于防错语句和条件语句的更多详细内容请参阅本书第 7 章。

本例文件参见光盘：..\第六章\6-4 删除活动单元格的值.xlsm

6.3.9 Resize：重置区域大小

Range.Resize 属性用于调整区域的大小，返回代表调整后的区域。它的具体语法如下：

```
Range.Resize(RowSize, ColumnSize)
```

其中参数 RowSize 代表重置后的区域行数，ColumnSize 代表重置后的区域列数。两个参数皆为可选参数，如果省略参数则表示新区域中的行数或者列数保持不变。

以下为 Range.Resize 属性的应用案例：

Range("a1").Resize(2, 2)——表示 A1:B2 单元格区域，包括 2 行 2 列共 4 个单元格；

Cells(3, 2).Resize(1, 4)——表示 B3:E3 单元格区域，包括一行 4 列共 4 个单元格；

Range("B1:C2").Resize(4, 4)——表示 B1:E4 单元格区域，包括 4 行 4 列共 16 个单元格；

Range("B1:C2").Resize(1)——表示 B1:C1 单元格区域，将原区域两行重置为一行，而列数保持不变；

Range("B1:C2").Resize(1, 1)——表示 B1 单元格区域，行与列都调整为 1。为了简化代码通常不使用这种引用方式，而是改用索引号来引用第一个单元格——Range("B1:C2")（1）；

Range([a2], [c10]).Resize(4, 5)——表示 A2:E5 单元格区域，将原区的域列数增大、行数减小。

Range.Resize 属性的参数不可以使用负数和 0，否则运行时将产生错误。

还可以使用表达式作为 Resize 属性和参数，例如：

Range("A4:B7").Resize(1, [A1] + 5)——重置后的新区域行数为 1，列数为 A1 的值加 5。

根据以上的参数解说，读者对 Resize 属性应该已有比较深入的认识。以下是 Range.Resize 属性的一个应用案例——将第一个工作表的已用区域复制到第二个工作表中。

```
Sub 复制工作表的已用区域的值()          '不保留公式和格式，仅复制数值(代码存放位置:模块中)
  With Worksheets(1).UsedRange  '引用第一个工作表的已用区域
 '以第二个工作表的 A1 单元格为基准，重置为第一个工作表的已用区域的大小，
    '然后将其赋值为第一个工作表的已用区域的值
   Worksheets(2).Range("a1").Resize(.Rows.Count, .Columns.Count).Value = .Value
  End With
End Sub
```

在将一个区域赋予另一个区域时，需要确保两个区域的高度和宽度一致，否则将产生错误。而利用 Resize 属性可以计算数据源的大小，再将目标区域重置为相同的大小。

代码中“.Rows.Count”表示区域的总行数，“.Columns.Count”表示区域的总列数，当两个区域的行数与列数都相同时就可以通过等号赋值。

本例文件参见光盘：..\第六章\6-5 将第 1 个工作表的已用区域的值复制到第 2 个工作表.xlsm

6.3.10　Offset：根据偏移量引用新区域

Range.Offset 属性可以返回一个 Range 对象，代表位于指定单元格或单元格区域的一定的偏移量位置上的新区域。其具体语法如下：

```
Range.Offset(RowOffset, ColumnOffset)
```

RowOffset 表示行偏移量，ColumnOffset 表示列偏移量。两个参数均为可选参数，如果忽略某个参数，表示该参数采用默认值为 0，即不执行偏移。

以下是 Range.Offset 属性的应用案例：

[a1].Offset(2, 3)——表示相对于 A1 单元格向下偏移 2 行、向右偏移 3 列，即 D3 单元格；

Range("D2").Offset(, 4)——表示相对于 D2 单元格，行偏移为 0、列偏移为 4，即 H2 单元格。

VBA 中的 Range.Offset 属性与工作表函数 Offset 在使用上有比较大的差异。工作表函数中 Offset 有 5 个参数，后两个参数用于指定目标区域的高度与宽度，而 VBA 中的 Range.Offset 属性

则没有提供表示高度与宽度的参数，而是由其前置对象 Range 来决定。

Range("D2:C3").Offset(1, 1)——表示相对于 D2:C3 单元格区域向下偏移 1 行、向右偏移 1 列，且高度与宽度与 D2:C3 一致的新区域 D3:E4；

Range("D2:D10").Offset(, 4)——表示引用 H2:H10 单元格区域，从原区域向右移动 4 列，高度与宽度一致；

Range.Offset 属性的参数也可以使用负数。当 RowOffset 参数使用负数时则表示向上偏移，而 ColumnOffset 参数使用负数时则表示向右偏移。

Range("D4:D10").Offset(–2, 4) ——表示引用 H2:H8 单元格区域，在原有区域基础上向上偏移 2 行、向右偏移 4 列；

Cells(3, 4).Offset(–1, –2)——表示引用 B2 单元格，在原单元格 D3 基础上上移 1 行、左移 2 列；

Cells(3, 4).Offset(–1, –4)——由于向左偏移 4 列后已超过了工作表的边界，所以产生错误。

也可以使用表达式作为 Range.Offset 属性的参数，例如：

Range("F2").Offset(2*3, [a1] + 5)——表示向 F2 单元格向下偏移 6 行，向右偏移 A1 单元格中的值加 5 列。

以下提供一个 Range.Offset 属性的应用案例。

假设有如图 6.15 所示的工作簿，要求将 A 组、B 组和 C 组 3 个工作表的数据复制到总表中，将数据按先后顺序向下排列。

	A	B	C	D
1	业绩表			
2	姓名	业绩	奖金	
3	柳红英	372840	1870	
4	罗正宗	208992	1050	
5	古山忠	253987	1270	
6	钱单文	396139	1990	
7	徐大鹏	325777	1630	
8				

A组 B组 C组 总表

图 6.15　业绩表

要实现本例所要求的功能主要涉及以下三个知识点。

（1）已用区域：即 Worksheet.Usedrange 属性，它可以自动适应工作表的数据多少，不管工作表的数据如何增减变化都不用修改代码。

（2）Range.Offset 属性：通过此属性可以得到工作表中已用单元格区域下方的区域，复制数据时用它作为目标区域可以避免覆盖数据。

（3）Range.Copy 方法：此方法有一个可选参数，它的语法如下所示。

```
表达式.Copy(Destination)
```

参数 Destination 是一个单元格对象，通常使用单个单元格即可。当忽略参数时表示将数据复制到剪贴板中；当使用了参数时表示将数据复制到参数所指定的单元格中，当被复制的对象是一个区域时，将会沿参数所指定的单元格向下、向右扩展，形成与被复制的单元格区域的大小一致的新区域，然后将数据粘贴在此区域中。例如代码“Range("a1:c2").Copy Range("b5")”中的目标单元格实际上不是 B5 单元格，而是 B5:D6，其高度和宽度由 Range("a1:c2")决定。

综合以上 3 个知识点，代码如下：

```
Sub 复制数据()'代码存放位置:模块中
   Worksheets("总表").Cells.Clear '清除“总表”中的所有内容，包括格式信息
  '将“A 组”工作表的已用区域复制到“总表”中，存放区域的左上角单元格为 A1
  Worksheets("A 组").UsedRange.Copy Worksheets("总表").Cells(1)
  '将“B 组”工作表的已用区域复制到“总表”的已用区域下一行中
```

```
  Worksheets("B 组").UsedRange.Copy  Worksheets("总表").UsedRange.Cells(1).
Offset(Worksheets("总表").UsedRange.Rows.Count).Cells(1)
  '将“C 组”工作表的已用区域复制到“总表”的已用区域下一行中
  Worksheets("C 组").UsedRange.Copy  Worksheets("总表").UsedRange.Cells(1).
Offset(Worksheets("总表").UsedRange.Rows.Count).Cells(1)
End Sub
```

当执行以上代码后可得到如图 6.16 所示结果。

以上过程在合并 3 个工作表时没有考虑标题的问题，复制 3 个工作表后在“总表”中就产生了 3 个标题，如果要求只保留一个标题，那么应使用 Range.Offset 属性排除第 2 个、第 3 个标题，完整代码如下：

```
Sub 复制数据 2()'代码存放位置:模块中
  Worksheets("总表").Cells.Clear '清除“总表”中的所有内容
  '将“A 组”工作表的已用区域复制到“总表”中，存放区域的左上角单元格为 A1
  Worksheets("A 组").UsedRange.Copy Worksheets("总表").UsedRange.Cells(1)
 '将“B 组”工作表的已用区域(排除前 2 行)复制到“总表”的已用区域下一行中
  Worksheets("B 组").UsedRange.Offset(2).Copy  Worksheets("总表").UsedRange.
Cells(1).Offset(Worksheets("总表").UsedRange.Rows.Count).Cells(1)
 '将“C 组”工作表的已用区域(排除前 2 行)复制到“总表”的已用区域下一行中
  Worksheets("C 组").UsedRange.Offset(2).Copy  Worksheets("总表").UsedRange.
Cells(1).Offset(Worksheets("总表").UsedRange.Rows.Count).Cells(1)
End Sub
```

新的代码执行后的效果如图 6.17 所示。

	A	B	C	D
1	业绩表			
2	姓名	业绩	奖金	
3	柳红英	372840	1870	
4	罗正宗	208992	1050	
5	古山忠	253987	1270	
6	钱单文	396139	1990	
7	徐大鹏	325777	1630	
8	业绩表			
9	姓名	业绩	奖金	
10	朱贵	306088	1540	
11	游三妹	344978	1730	
12	罗新华	226341	1140	
13	罗至贵	375649	1880	

A组 B组 C组 总表

图 6.16　合并结果 1

	A	B	C	D
1	业绩表			
2	姓名	业绩	奖金	
3	柳红英	372840	1870	
4	罗正宗	208992	1050	
5	古山忠	253987	1270	
6	钱单文	396139	1990	
7	徐大鹏	325777	1630	
8	朱贵	306088	1540	
9	游三妹	344978	1730	
10	罗新华	226341	1140	
11	罗至贵	375649	1880	
12	张未明	385038	1930	
13	钟秀月	272435	1370	

A组 B组 C组 总表

图 6.17　合并结果 2

修改后的代码使用了“Offset(2)”，表示将已用单元格区域向下移动两行从而产生新的区域，该区域不再包含标题行。

CD-ROM　本例文件参见光盘：..\第六章\6-6 将三个工作表的细明数据复制到总表中.xlsm

6.3.11　Union：多区域合集

多区域的合集是将多个单元格或者区域合并为一个 Range 对象，它与合并单元格的概念不同，仅仅为了引用方便，不会影响单元格的状态和数值。

在工作中，需要用到合集的地方比较多，通常配合循环语句一起使用。

引用多区域的合集可用 Application.Union 方法，其具体语法为：

```
Application.Union(Arg1, Arg2, Arg3, Arg4, Arg5, Arg6, Arg7, Arg8, Arg9, Arg10,
Arg11, Arg12, Arg13, Arg14, Arg15, Arg16, Arg17, Arg18, Arg19, Arg20, Arg21,
```

```
Arg22, Arg23, Arg24, Arg25, Arg26, Arg27, Arg28, Arg29, Arg30)
```

其中前 2 个参数为必选参数，其余 28 个参数为可选参数，其返回结果为 Range 对象。

例如需要同时引用 A2:B2 和 D3:G4 两个单元格区域，那么可以利用 Union 方法将两个单元格区域合并为一个 Range 对象，后续使用此对象参与运算即可。代码如下：

```
Application.Union(Range("A2:B2"),Range("D3:G4"))'其中 Application 对象允许被忽略
```

如果读者还记得 Range 引用多区域的方法，那么可能会认为不使用 Union 方法仍然可以引用多区域的合并区域。例如以上区域可以表示为：

```
Range("A2:B2,D3:G4")
```

然而，Range 参数的字符限制使它在多区域应用方面无法取代 Application.Union 方法，当参数长度超过 256 个字符时必将产生编译错误。而 Application.Union 方法配合循环语句可以突破这个屏障，在学习本书第 7 章循环语句后 Application.Union 方法将大有用武之地。

6.3.12 Intersect：单元格、区域的交集

交集是指两个或者多个区域的重叠部分，交集也是一个 Range 对象。

获取多区域的交集可用 Application.Intersect 方法，其具体语法如下：

```
Application.Intersect(Arg1, Arg2, Arg3, Arg4, Arg5, Arg6, Arg7, Arg8, Arg9,
Arg10, Arg11, Arg12, Arg13, Arg14, Arg15, Arg16, Arg17, Arg18, Arg19, Arg20,
Arg21, Arg22, Arg23, Arg24, Arg25, Arg26, Arg27, Arg28, Arg29, Arg30)
```

其中前 2 个参数为必选参数，其余 28 个参数为可选参数。

相对于多区域的合集，多区域的交集在工作中有更广阔的应用天地，通常利用它来提升代码的执行效率。

以下为 3 种常见引用交集的方法：

Application.Intersect(Range("a1:a10"), Range("2:2"))——表示引用 A1:A10 与第 2 行的交界处 A2 单元格，通过图 6.18 可以观察多区域取交集的算法。

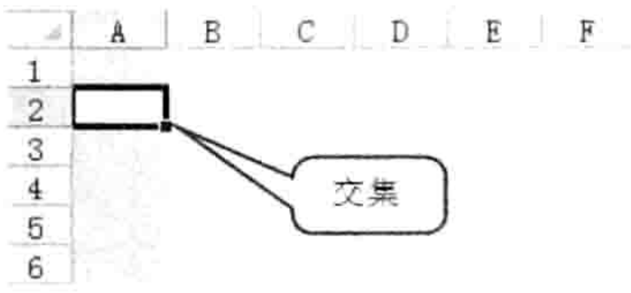


图 6.18　两个区域的交集

Intersect([B2:F10], Range("A2:G3"))——表示引用 B2:F3 单元格区域，Application 可以忽略不写；

Intersect(Range("B2:F10"), Cells(3, 11))——两个区域不存在重叠区，运行代码时将产生错误。可以利用 TypeName 函数计算 Intersect 的类型，如果返回 Nothing 表示两者无交集。

下面通过一个案例解读 Intersect 方法的使用思路。

要求在活动工作簿的任何工作表中的 D 列、E 列和 G 列双击就可以输入日期。

解决本题需求的重点在于如何选择事件，以及如何限定事件的触发条件，具体操作步骤如下。

step 1　新建一个工作簿，按<Alt+F11>组合键进入 VBE 窗口。

step 2　如果未显示工程资源管理器，那么单击菜单中的“视图”→“工程资源管理器”命令。

step 3　双击工程资源管理器中的 ThisWorkbook 从而打开工作簿事件代码窗口。

step 4　单击代码窗口上方的对象列表，从中选择 Workbook。

step 5　单击代码窗口上方的过程列表，从中选择 SheetBeforeDoubleClick，此时在代码窗口中已经产生了 Workbook_Open 事件和 Workbook_SheetBeforeDoubleClick 事件的程序外壳。

step 6　删除 Workbook_Open 事件的程序外壳，然后在 Workbook_SheetBeforeDoubleClick 事件中加入判断语句。完整代码如下：

```
Private Sub Workbook_SheetBeforeDoubleClick(ByVal Sh As Object, ByVal Target
As Range, Cancel As Boolean)'代码存放位码：ThisWorkbook 中
 '首先使用 Union 将 D、E 列和 G 列合并为一个 Range 对象，然后利用 Intersect 判断该对象是否
与 Target 存在交集
 '如果有交集，那么在 Target 中录入当前日期
  If Intersect(Union(Range("D:E"), Range("g:g")), Target) Is Nothing Then
    Target = Date
  End If
End Sub
```

step 7　关闭 VBE 窗口返回工作表界面，然后双击 C2 单元格，C2 单元格中将会产生当前的系统日期，效果如图 6.19 所示。

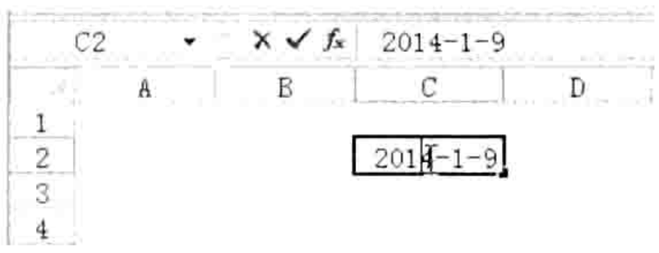


图 6.19　双击 C2 单元格录入当前日期

step 8　双击 G10 单元格，同样会产生当前日期，但是双击 D 列、E 列和 G 列以外的任意区域则不会产生当前日期。

代码中 Union 方法的作用是将三列合并为一个 Range 对象，再用此对象与事件过程中的 Target 进行比较，如果重叠就允许双击录入日期。Target 变量是一个 Range 对象，代表事件过程中被双击的那一个单元格。

If Then 是条件语句，在本例中用于判断被双击的单元格与 D 列、E 列和 G 列是否存在交集，当不存在交集时，Intersect 方法的返回值是 Nothing。

Workbook_SheetBeforeDoubleClick 事件是工作簿级别的事件，双击单元格时触发此事件，对代码所在工作簿的任意工作表都生效。此事件的代码必须录入在 Thisworkbook 窗口中。

CD-ROM　本例文件参见光盘：..\第六章\ 6-7 在指定区域双击录入日期.xlsm

6.3.13　End：引用源区域的区域尾端的单元格

如果活动单元格在一个比较大的数据区域中间，按组合键<Ctrl+↑>、<Ctrl+↓>、<Ctrl+←>、<Ctrl+→>可以迅速激活已用区域边缘的单元格；如果整个工作表均为空白则可以在最小行、最小列或者最大行、最大列之间切换；如果在空白区域向数据区域所在方向使用快捷键，则可以定位于数据区域的第一个或者最后一个非空单元格。

Range.End 属性正好对应于以上 4 个快捷键，可以实现最上端、最下端、最左端和最右端的切换。它的具体语法为：

```
Range.End(Direction)
```

其中必选参数 Direction 表示方向，可选值是表 6-6 中的 4 个常量。

表 6-6　End属性的参数详解

名称	值	描述
xlDown	–4121	向下
xlToLeft	–4159	向左
xlToRight	–4161	向右
xlUp	–4162	向上

以下语句配合图示可以让读者更清晰地理解 Range.End 属性的用法与技巧，所有代码都基于图 6.20 的数据进行演示，如果工作表中数据有变化，则代码的执行结果也会相应变化。

（1）引用 C 列第一个非空单元格

Range("C1").End(xlDown)——从空白单元格 C1 向下，遇到第一个非空单元格时停止；

Range("C4").End(xlUp)——从非空区域中间向上移，遇到第一个非空单元格时停止；

Range("C1048576").End(xlUp). End(xlUp)——从 C 列最后一个非空单元格向上，遇到第一个非空单元格时停止，再次向上遇到最后一个非空单元格时停止。

以上 3 种方法都可以得到正确结果 C2。

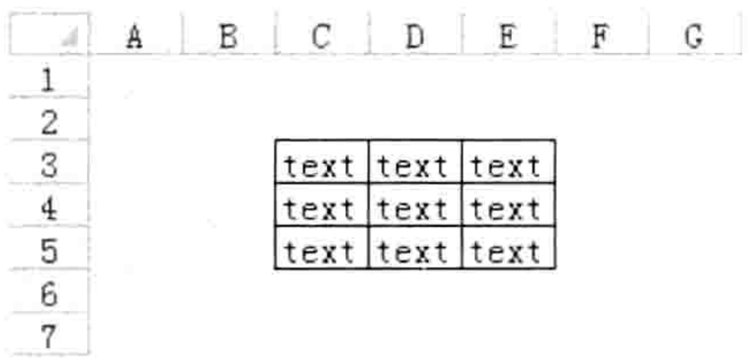


图 6.20　测试数据

（2）引用第 4 行最后一个非空单元格

Range("XFD4").End (xlToLeft)——从 XFD4 单元格向左遇到的第 1 个非空单元格；

Cells(4, Columns.Count).End(xlToLeft)——第 4 行最后一列向左遇到的第 1 个非空单元格。

以上两句代码都能成功引用目标单元格 E4，不过第一句代码通用性比较差，只有在 xlsx 和 xlsm 格式的工作簿中可用，在 xls 格式的工作簿中执行代码会产生错误。

本例文件参见光盘：..\第六章\ 6-8 Range.End 属性的应用.xlsm

为了确保代码的通用性，引用 A 列最后 1 个单元格应使用 Cells(Rows.Count, 1)而不是 Range("A65535")或者 Range("A1048576")；引用第 1 行最后 1 个单元格应使用 Cells(1,Column.Count)而不是 Range("IV1")或者 Range("FXD1")。

（3）获取 D 列已用单元格区域下面第一个空白单元格的地址

当需要在 D 列插入新数据时，通常需要定位最后一个非空单元格下方的第一个空单元格，将新数据存放在此处既避免覆盖原有数据又确保新数据与原始数据间不产生空行。

获取 D 列已用单元格区域下面第一个空白单元格地址的代码如下：

```
Sub D列已用区域下面第一个空白单元格地址() '代码存放位置:模块中
  MsgBox Cells(Rows.Count, 4).End(xlUp).Offset(1).Address
End Sub
```

代码 Cells(Rows.Count, 4)代表第 4 列最后一个单元格，而 End(xlUp).Offset(1)表示从该单元格向上遇到第一个非空单元格，然后再向下偏移一行，从而得到“D 列已用区域下面第一个空白单元格”。本代码对于 Excel 2003、Excel 2007、2010 和 2013 版本都通用。

6.3.14　RangeFromPoint：屏幕坐标下的单元格

RangeFromPoint 是窗口对象 Window 的一个方法，可通过关键字“Window.RangeFromPoint”从 VBA 的帮助中找到详细的语法说明。

RangeFromPoin 方法用于获取位于屏幕上指定坐标位置的 Shape 对象或 Range 对象。如果指定坐标位置上没有任何形状，则此方法将返回 Nothing；如果在指定坐标的单元格上方有 Shape 对象，那么 RangeFromPoint 方法可返回一个 Shape 对象，如果指定坐标处只有单元格则返回 Range 对象。

在图 6.21 中，鼠标指针下是图片，所以此时该坐标下的 Window.RangeFromPoint 对象就是 Shape 对象，即工作表中的图片。如果删除图片，那么此时该坐标下的 Window.RangeFromPoint 对象就是 Range 对象，即 B5 单元格。

图 6.21　鼠标指针下的对象

RangeFromPoint 方法的具体语法如下：

```
Window.RangeFromPoint(X,Y)
```

其中参数 X 表示屏幕的横坐标，Y 表示屏幕的纵坐标，单位皆为像素，坐标原点是屏幕左上角。例如需要获取横坐标为 400、纵坐标为 400 处的对象名称，可使用以下过程：

```
Sub 获取坐标下的对象名称() '放置存放位置：模块中
    Dim Obj As Object      '声明一个 Object 型的对象变量
    Set Obj = ActiveWindow.RangeFromPoint(400, 400) '将 RangeFromPoint 获得的对象赋予变量
    If Obj Is Nothing Then  '如果对象不存在
        MsgBox "没有单元格和图形对象", vbOKOnly, "友情提示" '提示没有对象
    Else                    '否则(表示该坐标下方有单元格对象或者图形对象)
        '如果对象的类型是 Range，那么提示对象的地址，否则提示对象的名称(图形对象)
        If TypeName(Obj) = "Range" Then MsgBox Obj.Address Else MsgBox Obj.Name
    End If
End Sub
```

过程中首先定义了一个 Object 型的变量，之所以不定义为 Range 型是因为指定坐标下可能是图片也可能是单元格，如果定义为 Range，那么该坐标下是图片时则会出错。

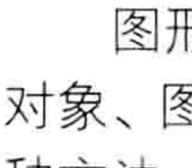
然后通过代码 ActiveWindow.RangeFromPoint(400, 400)引用指定坐标下的对象，并且赋值给变量 Obj。如果此时变量值是 Nothing 则表示坐标下不存在图形对象也不存在单元格对象；如果变量不是 Nothing，那么需要区分变量所代表的对象是图形对象还是单元格对象，因此代码中使用了 TypeName 函数计算变量 Obj 的类型名称，返回值是 Range 时表示是单元格对象，取其 Address 属性值即可，否则取其 Name 属性值——Shape 对象没有 Address 属性。

TypeName 函数用于判断一个参数的类型（查询帮助的关键字：TypeName 函数），当用字符串与 TypeName 函数的返回值做比较时要注意一个问题——Excel 在执行比较运算时是区分大小写的，而 TypeName 函数的返回值总是首字母大写，因此本例代码中的“Range”不能写作“range”

或者“RANGE”。

假设确定该横坐标、纵坐标为 400 处只有 Range 对象，那么代码就可以大大简化，一句代码即可解决：

```
MsgBox ActiveWindow.RangeFromPoint(400,400).Address
```



本例文件参见光盘：..\第六章\6-9 获取指定坐标下的对象.xlsm

RangeFromPoint 方法在实际工作中用处不大，而且即使要用也需要配合 API 函数使用。例如通过 API 函数获取鼠标移动时鼠标指针的坐标，以该坐标作为 RangeFromPoint 方法的参数获取鼠标指针下方的单元格或者图片。本书不涉及 API 知识，不演示 RangeFromPoint 方法与 API 函数的搭配应用，不过可以为读者提供一个案例，供大家休闲时娱乐，同时了解 API 与 RangeFromPoint 搭配应用时产生的奇效。本书光盘中路径为“..\第六章\鼠标移过单元格时在状态栏显示地址.xlsm”的文件中采用了 API 配合 RangeFromPoint 应用，实现鼠标移过单元格时在状态栏显示单元格的地址。

6.4 图形对象

图形对象的类别名称是 Shape，形状（在 Excel 2003 中称之为自选图形）、剪贴画、SmartArt 对象、图表、艺术字、文本框和插入到工作表中的图片等都属于图形对象。本节讲解图形对象的多种方法。

6.4.1 Shapes：图形对象集合

Shapes 代表图形对象集合，其父对象是 Worksheet，书写时不可以省略父对象名称。

可以通过关键字“Shapes 对象成员”在 VBA 的帮助中查到图形对象集合的方法与属性。

Shapes 对象集合包含形状（在 Excel 2003 中称之为自选图形）、剪贴画、SmartArt 对象、图表、艺术字、文本框和插入到工作表中的图片，不过由于它的父对象是工作表而不是工作簿，因此可以通过以下代码统计活动工作表中的图形对象数量：

```
MsgBox ActiveSheet.Shapes.Count
```

但没有办法通过 Shapes 对象集合直接计算工作簿中的图形对象数量。

从“Shapes 对象成员”的帮助中可以得知——Shapes 对象集合不支持批量操作，包括删除、移动、调整比例等，唯一的一个可以用于 Shapes 对象集合的方法是 SelectAll，它可以批量选中 Shapes，但不能对图形对象集合批量删除、批量移动、批量调整比例。

Shapes 对象集合有一个 Item 方法用于获取对象集合中的单个对象，例如需要引用活动工作表中的第 2 个图形对象，那么应采用以下代码：

ActiveSheet.Shapes.Item(2)——其中 Item 可以忽略，直接使用索引号即可引用单个对象。

单个图形对象和图形对象集合有着截然不同的方法和属性。可以通过关键字“Shape 对象成员”到 VBA 的帮助中查到单个图形对象的所有方法和属性名称。

根据 Shapes 的帮助和 Shape 对象的帮助可以得到以下拓展应用：

Msgbox ActiveSheet.Shapes(1).Name——获取活动工作表的第 1 个图形对象的名称；

Msgbox WorkSheets(2).Shapes.Count——计算第 2 个工作表的图形对象的总数量；

Activesheet.Shapes(Activesheet.Shapes.count).Delete——删除活动工作表的最后一个对象；

Activesheet.Shapes(3).Select——选中活动工作表的第 3 个图形对象；

Worksheets(1).Shapes(3).left = 0——将第 1 个工作表的第 3 个图形对象移到工作表最左边位

置。Left 属性代表图形对象的左边距，赋值为 0 则表示左移到边界处。

需要注意的是不管引用哪一个工作表中的图形对象都不能忽略父对象。例如“ActiveSheet.Shapes(1)”不能写作“Shapes(1)”。

6.4.2　图形对象的名称

图形对象有两种命名方式，一种是插入图形对象时在名称栏可以看到的名称，采用汉字加编号的形式命名，可以随意修改；一种是 VBA 中专用的名称，采用英文加编号的形式命名，不可以修改。

通过以下步骤可以了解这两种命名方式。

step 1　在工作表中插入任意一张图片，在名称栏中可以看到图片名称为“图片 1”，如图 6.22 所示。

step 2　按<Alt+F11>组合键打开 VBE 窗口，然后单击“插入”→“模块”命令，并在模块中录入代码：

```
Sub 获取图片名称()'代码放置位置：模块中
  MsgBox ActiveSheet.Shapes(1).Name '获取第 1 个图形对象的名字
End Sub
```

step 3　选择代码，然后按<F5>键执行代码，VBA 弹出如图 6.23 所示的提示信息。

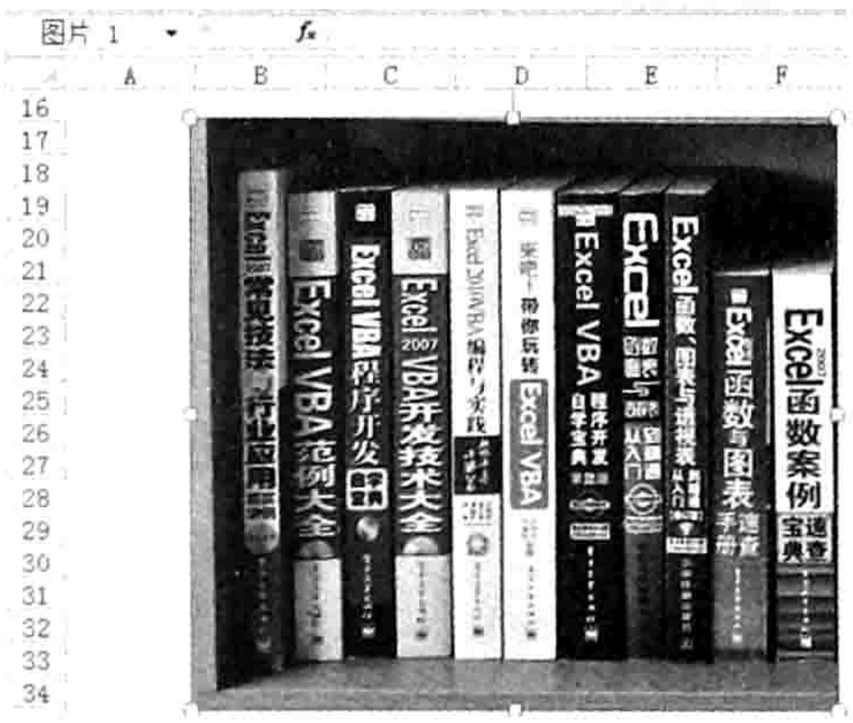


图 6.22　名称栏中显示的图片名

图 6.23　VBA 内部的图片名称

step 4　返回工作表界面，进入名称栏将“图片 1”修改为“图书”，然后按<Enter>键确认。

step 5　按<Alt+F11>组合键打开 VBE 窗口，在模块中继续录入代码：

```
Sub 获取图片的边距() '代码放置位置：模块中
  MsgBox ActiveSheet.Shapes("图书").Left '获取第 1 个图形对象的左边距
End Sub
```

执行代码后将获得图片的左边距，说明手工对图片命名后可以将该名称应用到代码中。但是此命名方式对于 VBA 内部的名称没有影响，原名称“Picture 1”保持不变。

当工作表中有一个图形对象时，其名称是图形对象的类型加编号 1，当插入第二个图形对象时，则用第二个图形对象的名称加编号 2 命名，不管两个图形对象的类型是否一致。也就是说图形对象的编号方式与图形对象的类型无关，而是与插入顺序相关。假设表中有两个矩形和图表，编号时不会采用“矩形 1”、“矩形 2”和“图表 1”、“图表 2”，而是采用升序的自然数序号，不可能重复，除非手工重新命名。

不过不需要纠结于图形对象的具体名称的写法，在实际工作中通常采用循环语句访问每个图片，不需要知道图片的具体名称也可以正常工作。

本例文件参见光盘：..\第 6 章\ 6-10 图形对象名称.xlsm

6.4.3 DrawingObjects：隐藏的图形对象集合

Shapes 代表工作表中的图形对象集合，但是不能通过它对图片执行批量操作。VBA 提供了一个隐藏的图形对象集合——DrawingObjects，它可以弥补 Shapes 的缺陷。

由于 DrawingObjects 是 VBA 的一个隐藏对象，因此无法在 VBA 的帮助中查询到它的任何帮助信息，但事实上可以借用它来实现很多 Shapes 不具备的功能。下面提供几个应用案例。

1. 删除工作表中所有图形对象

```
Sub 删除活动工作表中所有图形对象()'代码存放位码：模块中
 ActiveSheet.DrawingObjects.Delete 'Delete 方法用于删除图形对象
End Sub
```

Shapes 对象集合没有 Delete 方法，但是 DrawingObjects 对象集合有 Delete 方法，借助“DrawingObjects.Delete”可以弥补 Shapes 不足之处。

2. 让所有图形对象按 B 列对齐

```
Sub 让所有图形对象按 B 列对齐()'代码存放位码：模块中
'DrawingObjects 对象没有 Left 属性，它的子对象 ShapeRange 具有 Left 属性
'因此借用 ShapeRange 实现批量调整边距
'代码的含义是让活动工作表中所有图形对象的左边距等于 B 列的左边距
 ActiveSheet.DrawingObjects.ShapeRange.Left = Range("B:B").Left
End Sub
```

DrawingObjects.ShapeRange 代表对象中的所有形状，它的 Left 属性即为图形对象的左边距，修改此属性值可以移动图形对象。本例以 Range("B:B").Left 作为图形对象的左边距，因此可以实现所有图形对象以 B 列对齐。

3. 缩小所有图形对象

```
Sub 让所有图形对象显示为原图的一半大小()'代码存放位置：模块中
 'ShapeRange.ScaleHeight 方法用于设置图形对象的高度，参数 0.5 表示缩小为一半大小
 '第 2 参数使用 msoFalse 表示以图片的当前大小为参照标准进行缩放
 '第 3 参数 msoScaleFromTopLeft 表示缩放图片时，其左上角位置保持不变
 ActiveSheet.DrawingObjects.ShapeRange.ScaleHeight 0.5, msoFalse, msoScale
FromTopLeft
End Sub
```

ShapeRange.ScaleHeight 方法用于调整图形对象的高度，其语法如下：

```
表达式.ScaleHeight(Factor, RelativeToOriginalSize, Scale)
```

其中参数 Factor 代表形状调整后的高度与当前或原始高度的比例，值小于 1 时表示缩小，大于 1 时表示放大；RelativeToOriginalSize 代表参照标准，赋值为 msoTrue 表示相对于形状的原有尺寸来调整，而赋值为 msoFalse 时表示相对于形状的当前尺寸来调整；第 3 参数 Scale 表示调整形状大小时，该形状哪一部分的位置将保持不变，可选项包括 msoScaleFromBottomRight、msoScaleFromMiddle 和 msoScaleFromTopLeft。

4. 批量复制图形对象且调整左边距

```
Sub 批量复制图形对象且调整左边距()          '代码存放位置：模块中
   ActiveSheet.DrawingObjects.Copy   '复制活动工作表的所有图形对象
   Worksheets(2).Activate            '激活第 2 个工作表
   Worksheets(2).Paste               '粘贴被复制的图形对象
   Selection.ShapeRange.Left = Range("A:A").Left  '设置对象的左边距
End Sub
```

Shapes 对象集合没有 Copy 方法，无法批量复制，DrawingObjects.Copy 方法可以弥补该缺陷。不过复制图形对象不如复制单元格方便，DrawingObjects.Copy 方法没有参数，不能指定目标区域或者目标工作表，因此复制图形对象需要分两步执行——先复制，后粘贴。

本例文件参见光盘：..\第六章\ 6-11 DrawingObjects 应用.xlsm

6.5　表对象

Excel 有 4 种表，包含工作表、图表、4.0 宏表和 5.0 对话框，它们统称表对象，不过用得最多的是工作表。本节介绍表对象的一些特性和引用方式。

6.5.1　表的类别

表对象包含工作表、图表、4.0 宏表和 5.0 对话框。当在工作表标签处单击鼠标右键，从弹出的快捷菜单中选择“插入”命令后即可看到这 4 类表的类别名称，如图 6.24 所示。

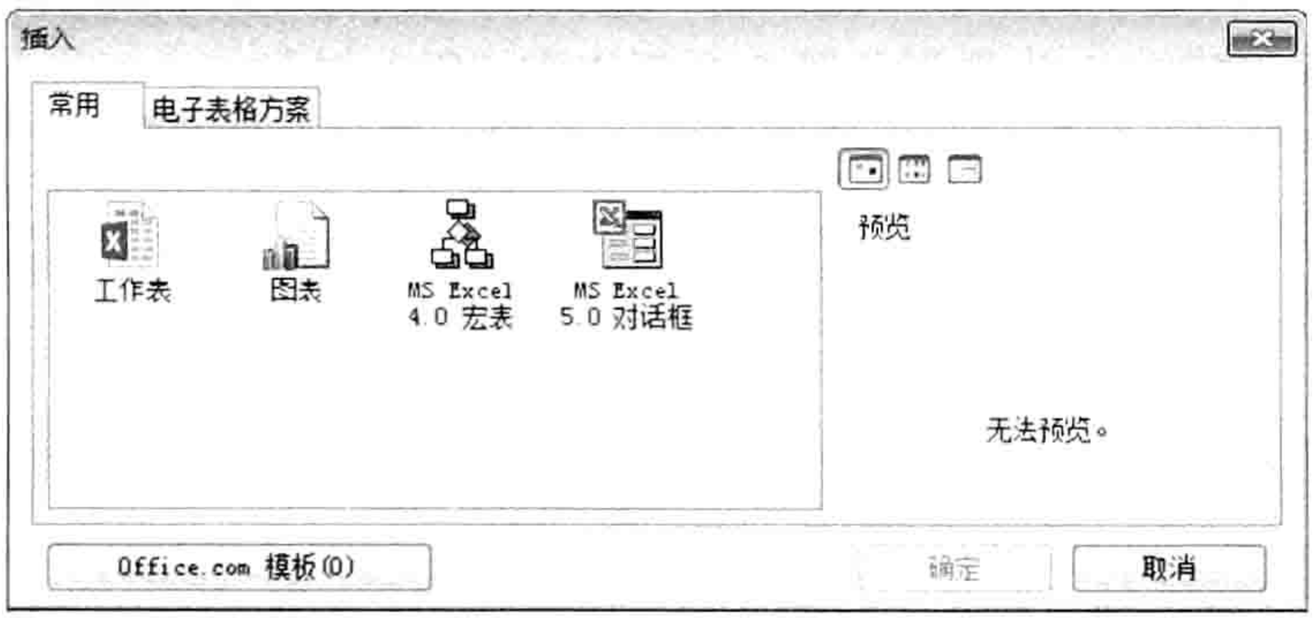


图 6.24　表的类别

表对象的集合采用 Sheets 表示，它代表工作簿中的所有表。

图表对象集合采用 Charts 表示，它代表工作簿中的所有图表。

5.0 对话框对象集合采用 DialogSheets 表示，它代表工作簿中的 5.0 对话框。

4.0 宏表对象集合采用 Excel4MacroSheets 表示，它代表工作簿中的所有 4.0 宏表。

工作表对象集合采用 Worksheets 表示，它代表工作簿中的所有工作表。

在图 6.25 中包含一个图表、一个 4.0 宏表、一个 5.0 对话框和两个工作表，使用以下代码可以得到表对象的总量 5：

```
Sub 计算表对象集合的数量()'代码存放位置：模块中
  MsgBox "共有" & Sheets.Count & "个表", vbInformation, "友情提示"  '结果为 5
End Sub
```

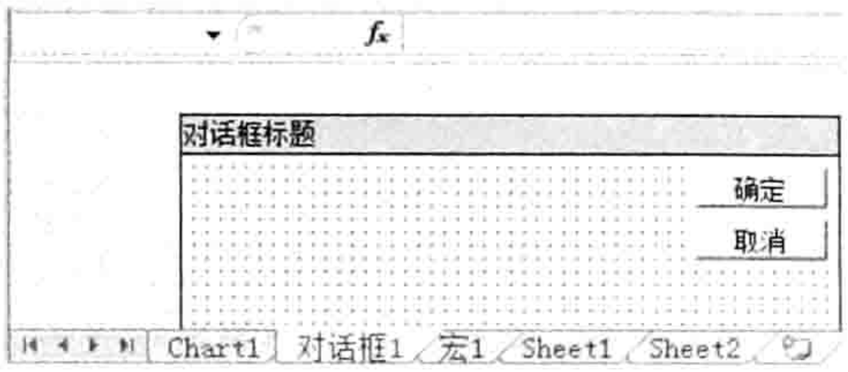


图 6.25　工作簿中的四类表

如果改用以下代码则可得到结果 2，表示活动工作簿的所有表中只有两个是工作表。

```
Sub 计算工作表对象集合的数量()    '代码存放位置：模块中
  MsgBox Worksheets.Count    '结果为 2
End Sub
```

6.5.2　Worksheets：工作表集合

尽管 Excel 支持 4 类表，但是常用的表是工作表，其类别名称为 Worksheet。

工作表对象集合的书写方式是 Worksheets，因此假设要求在工作簿中查找数据或者删除工作簿中所有图片时，那么循环语句的循环对象应该是 Worksheets，而非 Sheets，否则工作簿中存在图表时执行代码将会出错。

6.5.3　引用工作表子集

WorkSheets 代表工作表集合，可以通过参数引用其中单个工作表对象。参数包含数值参数和文本参数两种，当以数值作为参数时，VBA 将它当作工作表的顺序；当以文本作为参数时，VBA 将它当作工作表的名称。针对图 6.26 有以下结果：

Worksheets(2)——代表引用第 2 个工作表“Sheet1”，按从左向右的顺序计算；

Worksheets ("Sheet1") ——也是引用第 2 个工作表“Sheet1”；

Worksheets(Worksheets.count)——代表最后 1 个工作表“总表”；

Worksheets("Sheet"&变量)——当变量的值等于 2 时，那么可以引用“Sheet2”工作表。

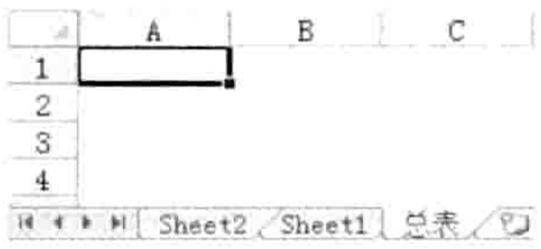


图 6.26　工作表

假设工作表的名称是数字该怎么引用呢？例如名为“10”的工作表。

如果采用 Worksheets(10)引用工作表，它只能引用第 10 个工作表，而不是名字为“10”的工作表。正确的办法是对名称“10”添加双引号——Worksheets ("10")。

如果“10”来自单元格，那么引用单元格作为 Worksheets 的参数时必须将单元格的值转换成文本，主要有以下两种办法。

第一种方法：

```
Worksheets(Range("a1") & "").Delete
```

当 A1 单元格的值是 10 时，直接引用 Range("a1")只能得到一个数值，而将数值与空文本串连后会变成文本。可以通过代码“MsgBox TypeName(Range("a1") & "")”验证是否为文本，此代码的执行结果是“String”。

第二种方法：

```
Worksheets(Cstr(Range("a1"))).Delete
```

此代码中使用了类型转换函数 Cstr，该函数可以将任意类型的数据转换成文本。同样可以使用代码“MsgBox TypeName(CStr(Range("a1")))”验证，其结果为“String”。

6.5.4 ActiveSheet：活动表

VBA 采用 ActiveSheet 来表示活动表，不管当前活动的表是何种类型。也就是说没有活动工作表、活动图表或者活动宏表之分，ActiveSheet 可以代表任意类型的活动表。但是实际工作中可能99%的情况都在使用工作表，因此在工作中常将 ActiveSheet 解释为活动工作表。

一个工作簿只允许有一个活动表，但允许每一个工作簿中都有一个活动表。

当访问活动工作簿的活动表时可以忽略工作簿名称，而访问非活动工作簿的活动表时需要指明活动表的父对象。例如将工作簿“生产.xlsm”的活动表重命名为“二月”，代码如下：

```
Workbooks("生产.xlsm").ActiveSheet.Name = "二月"
```

其中 Name 属性表示表的名称，可以直接赋值从而对表重命名，不过需要遵循 Excel 的命名规则，例如“\”、“/”、“:”、“?”、“*”等符号不允许作为表名称。

如果要将非活动表变成活动表，那么可以使用 Worksheet.Activate 方法去激活它。例如：

WorkSheets(4).Activate——激活第 4 个工作表；

WorkSheets(WorkSheets.Count).Activate——激活最后一个工作表。

6.5.5 工作表的特性

工作表有很多特性，现针对在实际工作中可能会影响到工作的几个特性进行逐一讲解。

1. 工作表的数量限制

Excel 的工作表没有数量限制，不过工作表越多则占用计算机的内存就越大，因此正确的说法是工作表数量仅受内存限制，计算机的内存越大，工作簿中可以存放的工作表就越多。

如果使用代码“Worksheets.Add”创建工作表，那么一次允许创建 1～255 个工作表。Worksheets.Add 方法的语法如下：

```
表达式.Add(Before, After, Count, Type)
```

其中参数 Count 表示新工作表的数量，默认值为 1，最大值是 255，因此使用以下代码可以一次性创建 255 个工作表，如果连续执行 10 次就能创建 2550 个新工作表：

```
Worksheets.Add Count:=255
```

2. 工作表的名称限制

对工作表命名时必须遵守工作表名称的规则，其规则如下：

（1）字符数量不能大于 31 位。

（2）不能包含“:”、“\”、“/”、“?”、“*”、“[”和“]”等特殊字符。

（3）名称不能是空白。

假设调用单元格中的字符为工作表命名，应先检查单元格的值是否符合以上三个规则。

3. 工作表的显示状态

使用代码“Worksheets(1).Visible =0”可以隐藏第一个工作表，不过当工作簿中仅有一个工作表时则无法隐藏。工作簿中必须保留至少一个可见的工作表。

4. 隐藏工作表的特性

隐藏工作表有两种办法，一是将它的 Visible 属性赋值为 0，二是将它的 Visible 属性赋值为 2。两者之间的区别在于后者是深度隐藏，只能修改工作表的 Visible 属性值才能显示工作表，前者是普通隐藏，可以在工作表标签的右键菜单中选择“取消隐藏”命令从而显示隐藏的工作表。

处于隐藏状态的工作表无法对其做任何操作，包括重命名、修改单元格的值、插入行、设置格式等，但是 VBA 可以对普通隐藏的工作表执行任意操作。

假设第 2 个工作表处于隐藏状态，以下所有语句都可以正常执行：

Worksheets(2).Name = "生产表"——对隐藏工作表重命名；

Worksheets(2).Range("B:B").Insert——在隐藏工作表的第 2 列前插入 1 列；

Worksheets(2).Range("B:B").Copy Worksheets(1).Range("a1")——将隐藏工作表的第 2 列数据复制到第 1 个工作表的 A 列。

如果工作表的 Visible 属性值为 2，即工作表处于深度隐藏状态下，那么使用 VBA 代码也无法对该表执行删除、复制、移动等操作。

6.6 工作簿对象

工作簿就是 Excel 文件，它保存着一切用户数据。本节展示引用工作簿的一些常识和技巧，以及不同格式的工作簿特性。

6.6.1 工作簿格式与特性

Excel 2003 使用 xls 格式的工作簿，它支持 65 536 行、256 列。从 Excel 2007 开始新增了 xlsx 和 xlsm 格式，将文件存为新格式可以拥有更大的空间、更小的体积。

xlsx 和 xlsm 格式的文件都支持 1 048 576 行、16 384 列，将同一个文件保存为 xlsx 或者 xlsm 格式，文件的体积大约只有 xls 格式工作簿的一半大小。

xlsx 格式的文件不能保存宏代码，因此本书随书光盘中的文件都采用了 xlsm 格式。

6.6.2 Workbooks：工作簿集合

VBA 中使用 Workbooks 表示工作簿集合，代表当前已打开的所有工作簿。对于没有打开的工作簿，VBA 没有任何办法引用。

以下是关于工作簿集合的一些简单应用：

Workbooks.Count——获取已打开的工作簿的总数量；

Workbooks.Close——关闭当前打开的所有工作簿。

MsgBox Workbooks.Item(2).Name——Item 属性可以产生集合中的单个对象引用，不过由于直接在括号中添加索引号也能引用，因此实际工作中都不使用 Item 属性。

6.6.3 引用工作簿子集

可以通过索引号引用 Workbooks 对象集合的单个工作簿，也可以通过名称来引用。

Workbooks（10）——引用第 10 个工作簿；

Workbooks（"工作簿 1"）——由于新建工作簿在保存前不存在扩展名，因此可以忽略；

Workbooks（"财务损益表.xlsm"）——引用已经保存过的工作簿必须添加扩展名；

Workbooks（Workbooks.count）——引用最后一个打开的工作簿；

Workbooks (Range（"a1"）.Value)——表示引用名字等于 A1 单元格的值的工作簿。

6.6.4　活动工作簿

活动工作簿是指当前可以直接操作的处于激活状态的工作簿，与活动表不同（每个工作簿都有一个活动表，所以当前打开多个工作簿时会有多个活动表）活动工作簿只有一个。

VBA 中采用 ActiveWorkbook 来表示活动工作簿。

如果需要将一个非活动工作簿转换成活动工作簿，可采用 Activate 方法激活它：

```
Workbooks("五月.xlsm").Activate
```

引用活动工作簿中的工作表时可以忽略 ActiveWorkbook，直接引用表名即可。

VBA 中还有一个特殊的对象——ThisWorkbook，ThisWorkbook 表示代码所在工作簿，而活动工作簿指当前正在使用的工作簿，与代码保存位置无关。

在任何一个打开的工作簿中通过代码调用 ActiveWorkbook 时都指向同一个工作簿，但是在不同工作簿中调用 ThisWorkbook 时却会指向不同的工作簿。

这是启用宏的压缩格式，可以使用新版本的一切新功能，也具有文件压缩功能。

◆　◆　◆

第 7 章　常用语句的语法剖析

编写代码时最重要的是了解语法，而对象、对象的方法和对象的属性只要录制宏或者借助属性与方法列表即可获取对应代码，不需要花费时间去记忆。

VBA 中涉及的语法相当多，本章对其中常见的几类语句的语法进行详细讲解，并通过诸多案例加深读者对这些语句的理解。

7.1　创建输入框

在前面的章节中涉及引用区域和指定数值时都是直接在代码中指定，而实际工作中的需求往往是不确定的，在编写代码阶段不能预知准确的信息，而是在执行代码阶段由用户指定。这引申出一个新的问题——有必要给用户创建一个输入框。

7.1.1　Application.Inputbox 方法

Excel VBA 提供了两个可以创建输入框的语句，一个是 Inputbox 函数，另一个是 Application.Inputbox 方法。后者的功能比前者强大，而且还可以对用户输入的数据进行校验，因此在实际工作中用 Application.Inputbox 方法代替了 Inputbox 函数。本书也仅介绍 Application.Inputbox 方法的语法并提供相应的案例应用，如果读者对 Inputbox 函数有兴趣可以在 VBA 的帮助中输入关键字“Inputbox 函数”查询。

Application.Inputbox 方法的存在价值在于提升程序的灵活性，让程序在执行过程中弹出一个输入框等待用户指定 Range 对象或者数据，而不是在编写代码期间由程序员指定。

Application.Inputbox 方法允许用户录入公式、数字、文本、逻辑值、单元格引用、错误值和数值数组等 7 种类型的值，同时也提供一个参数让程序去限制用户只能录入某种类型的值，并且带有校验功能，当录入的值与指定的类型不同时会提示用户。正是基于此优点，Application.Inputbox 方法在工作中被大量应用。

7.1.2　基本语法

Application.Inputbox 方法的基本语法如下：

```
Application.InputBox(Prompt, Title, Default, Left, Top, HelpFile, HelpContextID,
Type)
```

表 7-1 中包括了 Application.Inputbox 方法的各参数详解。

表 7-1　Application.Inputbox方法的各参数详解

名称	必选/可选	描述
Prompt	必选	要在对话框中显示的消息。可以为字符串、数字、日期或布尔值（在显示之前，Excel 自动将其值强制转换为 String）
Title	可选	输入框的标题。如果省略该参数，默认标题将为“Input”
Default	可选	指定一个初始值，该值在对话框最初显示时出现在文本框中。如果省略该参数，文本框将为空。该值可以是 Range 对象

续表

名称	必选/可选	描述
Left	可选	指定对话框相对于屏幕左上角的 X 坐标（以磅为单位）
Top	可选	指定对话框相对于屏幕左上角的 Y 坐标（以磅为单位）
HelpFile	可选	此输入框使用的帮助文件名。如果存在 HelpFile 和 HelpContextID 参数，对话框中将出现一个帮助按钮
HelpContextID	可选	HelpFile 中帮助主题的上下文 ID 号
Type	可选	指定返回的数据类型。如果省略该参数，对话框将返回文本

Application.Inputbox 方法有 8 个参数，其中最重要的是前 3 个和最后 1 个。前 3 个参数比较简单，通过以下代码可以瞬间了解参数与输入框的对应关系：

```
Sub 输入框()
  MsgBox Application.InputBox("请输入您的姓名：", "姓名", "罗刚君")
End Sub
```

执行程序后会弹出如图 7.1 所示的输入框，3 个参数的值都已经显示在输入框中。其中第 3 参数“罗刚君”是预设的默认值，用户可以随意修改。当单击“确定”按钮后，Msgbox 函数会将用户输入的值输出到信息框中。

图 7.1　默认值为“罗刚君”的输入框

Type 参数用于指定一种或者多种数据类型，Application.Inputbox 方法会根据类型对用户录入的信息进行校验，如果不符合则会阻止程序执行。Type 参数的可选值及其含义如表 7-2 所示。

表 7-2　Application.Inputbox方法的校验参数

值	含义
0	公式
1	数字
2	文本（字符串）
4	逻辑值（True 或 False）
8	单元格引用，作为一个 Range 对象
16	错误值，如 #N/A

表 7-2 中的数值允许单个使用，也允许多个组合，0 除外。例如对 Type 参数赋值为 1 表示只能录入数值，若赋值为 3 则表示可以录入数值或者文本（1+2=3），若赋值为 10 表示允许录入文本和单元格引用。

7.1.3　案例应用

Application.Inputbox 方法的功能体现在后期指定数据或区域上，本节通过 3 个案例展示 Application.Inputbox 方法在此方面的应用。

1. 案例：指定待求和的区域

案例要求：弹出输入框让用户选择区域，然后对该区域求和，横向和纵向各求和一次。

知识要点：Application.InputBox 方法、Range.FormulaR1C1 属性、Range.Offset 属性。

实现步骤：

step 1 在工作表中录入如图 7.2 所示的成绩表。

	A	B	C	D	E	F	G
1	姓名	语文	数学	化学	政治	计算机	
2	赵光明	80	86	55	75	75	
3	赵兴望	58	76	54	77	98	
4	刘越堂	89	91	60	83	94	
5	朱文道	56	73	71	58	86	
6	刘专洪	82	67	60	69	79	
7	张彻	50	55	66	100	62	
8	钟小月	53	78	58	68	55	
9	陈星望	81	89	66	51	77	
10							

图 7.2 成绩表

step 2 按<Alt+F11>组合键打开 VBE 窗口，然后单击菜单中的“插入”→“模块”命令。

step 3 在模块中录入以下代码：

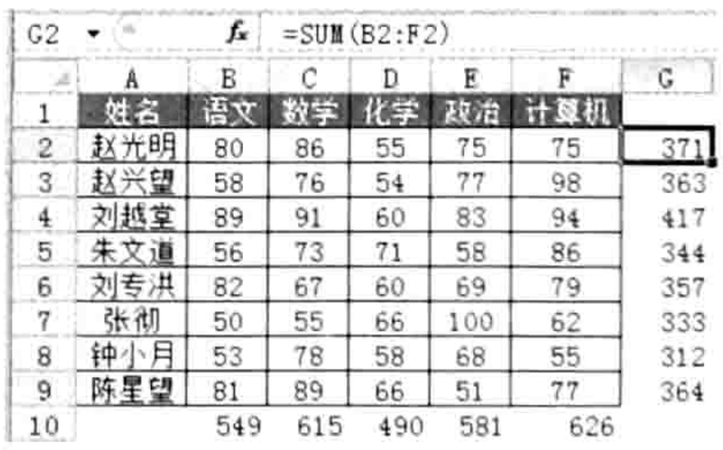
```
Sub 行列求和()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
 Dim Rng As Range
  Set Rng = Application.InputBox("请指定待求和的区域", "区域", , , , , , , 8)
Rng.Offset(,    Rng.Columns.Count).Columns(1).FormulaR1C1    =    "=SUM(RC[-"
&Rng.Columns.Count & "]:RC[-1])"
  Rng.Offset(Rng.Rows.Count).Rows(1).FormulaR1C1 = "=SUM(R[-" &Rng.Rows.Count
& "]C:R[-1]C)"
End Sub
```

step 4 按<Alt+F11>组合键返回工作表界面。

step 5 按<Alt+F8>组合键打开“宏”对话框，然后执行“行列求和”过程，当程序弹出“区域”输入框时，按下鼠标左键并拖动鼠标选择成绩区域 B2:F9，此时在输入框中会自动产生区域地址，如图 7.3 所示。

step 6 在“区域”输入框中单击“确定”按钮，程序会产生如图 7.4 所示的求和结果。

图 7.3 选择区域时在输入框中产生区域地址



	A	B	C	D	E	F	G
1	姓名	语文	数学	化学	政治	计算机	
2	赵光明	80	86	55	75	75	371
3	赵兴望	58	76	54	77	98	363
4	刘越堂	89	91	60	83	94	417
5	朱文道	56	73	71	58	86	344
6	刘专洪	82	67	60	69	79	357
7	张彻	50	55	66	100	62	333
8	钟小月	53	78	58	68	55	312
9	陈星望	81	89	66	51	77	364
10		549	615	490	581	626	

图 7.4 对指定的区域按行、列求和

思路分析：

由于在编程阶段需要计算的区域对象是未知的，因此采用变量占位，对变量所代表的区域执行求和，变量实际代表什么将在过程的执行阶段由用户指定。

根据以上思路，本例首先声明了一个 Range 型的变量，然后通过 Application.InputBox 方法弹出输入框让用户指定求和区域，并将返回值赋值给变量 Rng。由于变量 Rng 属于对象，因此对它赋值必须采用 Set 语句。

本例的难点在于最后两句——给 Rng 区域右边一列和下边一行设置求和公式。由于 Rng 代表求和区域（假设是 B2:F9），那么“Rng.Offset(, Rng.Columns.Count)”则代表 G2:K9 区域。由于只需要将公式录入到 G2:K9 区域的第一列，因此在它之后添加代码“.Columns(1)”从而引用它的第一列 G2:G9 区域。

当确定存放公式的区域后，通过 FormulaR1C1 属性向区域中录入公式即可。公式的重点在于如何取得左边的求和区域的地址。由于 R1C1 引用样式的公式采用的是相对引用地址，R 代表行、C 代表列，对 R 和 C 赋值即可得到目标区域地址。

以 G2 单元格为例，它左边的求和区域是 B2:F8，其中 A8 相对于 G2 的行偏移为 0、列偏移为 –5，即 Rng 区域的列数的相反数，因此引用 B2 应采用代码“RC[-" &Rng.Columns.Count & "]”；F2 相对于 G2 的行偏移为 0、列偏移为–1，因此引用 F2 应采用代码“RC[–1]”。将两者组合起来，引用 B2:F8 区域的相对引用代码为“RC[-" & rng.Columns.Count & "]:RC[–1]”。

知识补充：调用变量或者表达式时不能使用引号，否则变量与表达式会变成字符串。如果将一个文本与变量或者将文本与表达式串连成新的字符串应只对文本加引号，然后用连接运算符“&”将它与变量或者表达式串连起来。运算符的前后必须有一个空格，例如将变量“A”与文本“公斤”串连成字符串应用代码“A &"公斤"”，而非“"A 公斤"”。

本例实际上是调用公式对 Rng 区域求和，因此代码相当简洁。如果读者对公式比较了解，那么可以直接书写代码；如果读者对公式不太了解，可以录制宏让代码自动产生，然后手工修改代码。修改代码通常包括——加入变量、变量循环语句或判断语句，修改对象名称或者对象地址。

针对本例，先选择 G2:G9 区域，按<Alt+=>组合键，再选择 B10:F10 区域，按<Alt+=>组合键，将这两个步骤录制下来即可，代码中会自动产生公式，开发者需要做的是加入变量、修改公式的单元格地址即可。

从本节开始，所有代码不再提供每一行代码的注释，不过为了方便读者学习，在随书光盘的文件中提供了每一句代码的含义注释，还同步提供代码中涉及的某些语句的语法说明。请读者看书前将随书光盘的案例文件复制到硬盘中，然后再查看源代码及代码注释。

本例文件参见光盘：..\第七章\7-1 行列求和.xlsm

2. 案例：指定汇总方式与区域

案例要求：弹出输入框让用户选择区域，然后再弹出对话框让用户选择汇总方式，最后按行与列对选定的区域汇总。

知识要点：Application.InputBox 方法、Range.FormulaR1C1 属性、Range.Offset 属性。

实现步骤：

step 1　打开前一个案例文件，按<Alt+F11>组合键打开 VBE 窗口。

step 2　在模块中继续录入以下代码：

```
Sub 行列汇总()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Dim Rng As Range, GatherStyle As Long, GatherStr As String
  Set Rng = Application.InputBox("请指定待求和的区域", "区域", , , , , , , 8)
  GatherStyle = Application.InputBox("输入 1:求和" & Chr(13) & "输入 2:求积" &
Chr(13) & "输入 3:求平均" & Chr(13) & "输入 4:计数", "汇总方式", , , , , , , 1)
  GatherStr = Evaluate("=vlookup(" & GatherStyle & ",{1,""SUM"";2,""PRODUCT"";
3,""AVERAGE"";4,""COUNTA""},2,false)")
  Rng.Offset(, Rng.Columns.Count).Columns(1).FormulaR1C1 = "=" & GatherStr &
"(RC[-" & Rng.Columns.Count & "]:RC[-1])"
```

```
  Rng.Offset(Rng.Rows.Count).Rows(1).FormulaR1C1 = "=" & GatherStr & "(R[-" &
Rng.Rows.Count & "]C:R[-1]C)"
End Sub
```

step 3 按<Alt+F11>组合键返回工作表界面。

step 4 按<Alt+F8>组合键打开“宏”对话框，然后执行“行列汇总”过程，当程序弹出“区域”输入框时，按下鼠标左键并拖动鼠标选择成绩区域 B2:F9。

step 5 在“区域”输入框中单击“确定”按钮后，程序会产生如图 7.5 所示的输入框。

step 6 在“汇总方式”输入框中录入 3，然后单击“确定”按钮，程序会对选定的区域 B2:B9 求平均值，结果如图 7.6 所示。

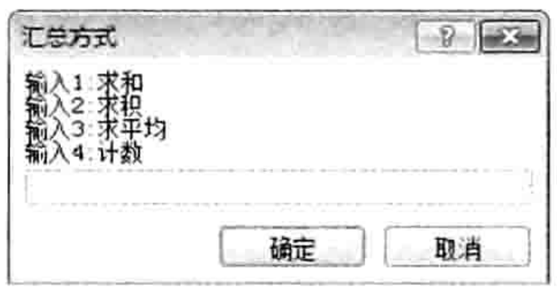
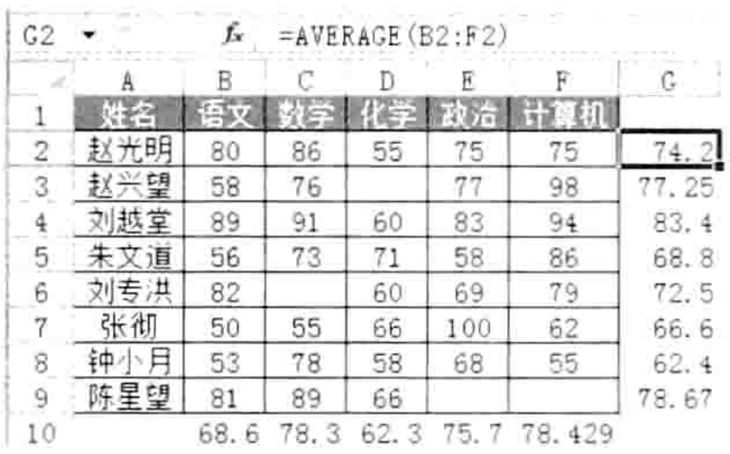


图 7.5 “汇总方式”输入框



G2 =AVERAGE(B2:F2)

	A	B	C	D	E	F	G
1	姓名	语文	数学	化学	政治	计算机	
2	赵光明	80	86	55	75	75	74.2
3	赵兴望	58	76		77	98	77.25
4	刘越堂	89	91	60	83	94	83.4
5	朱文道	56	73	71	58	86	68.8
6	刘专洪	82		60	69	79	72.5
7	张彻	50	55	66	100	62	66.6
8	钟小月	53	78	58	68	55	62.4
9	陈星望	81	89	66			78.67
10		68.6	78.3	62.3	75.7	78.429	

图 7.6 对指定的区域求平均值

思路分析：

相对于“案例 1”本例仅添加了一个确定汇总方式的输入框，不过涉及的知识点却包含 4 个。

其一：变量 GatherStyle 是数值型变量，不是对象，因此对它赋值不用 Set。

其二：为了美观，需要将 Application.Inputbox 方法的第 1 参数的字符串显示为 4 行，因此要在换行处插入换行符。VBA 中用 chr(13)或者 chr(10)表示换行，由于 chr(13)和 chr(10)属于表达式不属于文本，因此不能直接写在字符串中，而是先将文本拆分开，然后将换行符插入其中再通过连接运算符将它们串连起来。

其三：为了简化用户的操作，仅让用户在“汇总方式”输入框中录入数值，而实际产生在公式中的是函数名称，因此存在数字与函数名称的转换过程。工作表函数 Vlookup 可以实现此类转换，因此本例中创建了一个包含 Vlookup 的公式表达式，然后通过 VBA 中的 Evaluate 函数将该表达式转换成值。由于 GatherStyle 是变量，因此不能直接写入到公式中，而是通过连接运算符“&”将它与公式串连起来。

其四：要在字符串中产生双引号时必须书写两个双引号，否则会产生语法错误。例如要在信息框中显示“A"B”应使用代码“MsgBox "A""B"”，采用“MsgBox "A"B"”则会出错。

由于在 7.3 节才教授防错技术，因此本例代码并没有防错，如果在输入框中胡乱输入将可能导致程序出错。当前可以忽略此问题，待读者学到 7.3 节后再回过头来处理。

3. 案例：为新建文件夹命名

案例要求：弹出一个输入框，让用户录入文件名称，默认名称是当前日期，然后在 D 盘新建一个文件夹并且以该名称命名。

知识要点：Application.InputBox 方法、MkDir 语句。

实现步骤：

step 1 按<Alt+F11>组合键进入 VBE 窗口，然后单击菜单中的“插入”→“模块”命令。

step 2 在模块中录入以下代码：

```
'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
Sub 在输入框中指定名称然后创建文件夹()
```

```
 Dim FileName As String
 FileName = Application.InputBox("请输入文件夹名称", "文件夹名称", Format(Date,
"yyyy-mm-dd"))
 MkDir "d:\" & FileName
End Sub
```

Step 3　单击过程的任意位置，然后按<F5>键运行代码，程序会弹出如图 7.7 所示的输入框，其默认值为当前系统日期。

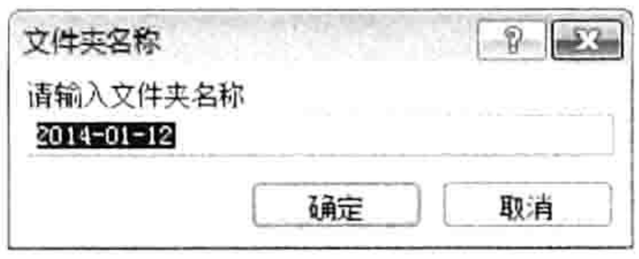


图 7.7　在输入框中指定文件夹名称

Step 4　保持默认的名称，然后单击“确定”按钮，那么程序会在 D 盘中创建一个名为“2014-01-12”的文件夹。不过如果重复此过程，多次创建相同名称的文件夹则会导致代码出错，Windows 不允许同一路径下存在多个同名的文件夹。

思路分析：

由于文件夹名称属于文本，因此首先声明一个 String 型的变量，然后通过 Application.InputBox 函数创建一个输入框等待用户输入文件夹名称。

输入框中的默认值是已经被格式化为“yyyy-mm-dd”格式的当前系统日期。

当用户输入文件夹名称后，程序会通过 MkDir 语句在 D 盘创建一个文件夹，文件夹的名称由变量 FileName 决定。

假设正常地录入文件夹名称，本例代码足以完成需求，假设用户在输入框中按下了“取消”键或者输入了“?”、“/”等禁用字符时代码必定会出错。不过本例代码的价值在于演示 Application.InputBox 方法的功能，对于条件判断将在 7.2 节中再补充。

本例文件参见光盘：..\第七章\7-2 在输入框中指定名称然后创建文件夹.xlsm

7.2　条件判断语句

条件判断语句在 VBA 中是使用率非常高的语句。由于录制宏时无法产生条件语句，必须通过 VBE 界面手工编写代码，因此了解条件判断语句的详细语法就显得尤为重要。

条件语句主要包括以下五种，本节将一一解析：

- IIF
- IF…Then…
- IF…Then…End IF
- Select Case…End Select
- Choose

7.2.1　IIF 函数的语法与应用

IIF 是 VBA 中类似于工作表函数 IF 的条件判断函数，由于 IIF 是函数，因此它会基于条件返回不同的值。

1. IIF 的基本语法

IIF 函数可以根据第一参数的值返回第 2、第 3 参数中的一个。它的基本语法为：

```
IIF(expr, truepart, falsepart)
```

IIF 的 3 个参数均为必选参数，各参数的含义如表 7-3 所示。

表 7-3　IIF的参数详解

参数	功能描述
expr	用来判断真伪的表达式
truepart	如果 expr 为 true，则返回本参数的值或表达式
falsepart	如果 expr 为 false，则返回本参数的值或表达式

如果 A1 的值大于或等于 60 分时在 B1 返回“及格”，否则返回“不及格”，那么可用以下语句：

```
Range("B1") = IIf(Range("a1") >= 60, "及格", "不及格")
```

如果第 2、第 3 参数的字符比较长，且不同的字符比较少，为了缩短代码，也可以改用以下方式：

```
Range("B1") = "您的成绩" & IIf(Range("a1") >= 60, "已", "不") & "及格"
```

即把相同部分置于 IIF 函数之外，而用 IIF 函数的第 2、第 3 参数来决定不同的部分。
A1 的值大于 B1 的值时，则 D1 返回 C1 的值，否则返回 C1 值的 50%，那么可用以下语句：

```
Range("d1") = IIf(Range("a1") > Range("b1"), Range("c1"), Range("c1") / 2)
```

也可以改用以下方式，将 C1 置于 IIF 语句之外，代码如下：

```
Range("d1") = Range("c1") / IIf(Range("a1") > Range("b1"), 1, 2)
```

2. And 运算符与 Or 运算符

当 IIF 函数使用多条件时，必须借助 And 运算符与 Or 运算符来连接条件。

如果需要同时满足多条件，可使用 And 运算符。And 运算符的主要作用是对两个表达式进行逻辑连接，其表达式如下：

```
result = expression1 And expression2
```

其中 result 与 expression1、expression2 之间的关系如表 7-4 所示。

表 7-4　And运算符的参数与结果之关系

如果 expression1 为	且 expression2 为	则 result 为
TRUE	TRUE	TRUE
TRUE	FALSE	FALSE
TRUE	Null	Null
FALSE	TRUE	FALSE
FALSE	FALSE	FALSE
FALSE	Null	FALSE
Null	TRUE	Null
Null	FALSE	FALSE
Null	Null	Null

如果有3个条件可以采用以下表达式：

```
result = expression1 And expression2 And expression3
```

如果某企业招聘时要求应聘人员的体重在50～65公斤，否则不合格，VBA表达方式如下：

```
Msgbox IIF(Range("a1")>= 50 And Range("a1")<=65, "合格", "不合格")
```

如果需要多条件中满足条件之一即可返回指定值，那么可以使用Or运算符，其主要作用是对两个表达式进行逻辑析取运算，其表达式如下：

```
result = expression1 Or expression2
```

其中result与expression1、expression2之间的关系如表7-5所示。

表7-5 Or运算符的参数与结果之关系

参数	功能描述
result	必选参数，任何数值变量
expression1	必选参数，任何表达式
expression2	必选参数，任何表达式

如果A1单元格的成绩小于1或者大于100，则提示“录入错误”，否则提示“正确”，那么VBA表达方式如下：

```
MsgBox IIf(Range("a1") > 100 Or Range("a1") < 1, "录入错误", "正确")
```

IIF函数也可以嵌套使用，即一句代码中使用多个IIF，根据两个以上的条件返回对应的值，而且每个条件参数也可以借用And或者Or运算符来连接。

当And和Or共用一个参数的时候，尽量采用括号来体现优先顺序。例如：

```
(expression1 And expression2) Or (expression3 And expression4)
```

3. IIF的应用案例

案例要求：根据录入的月份计算季度。

知识要点：Application.InputBox方法、IIF函数、Month函数。

实现步骤：

step 1 按<Alt+F11>组合键打开VBE窗口，然后单击菜单中的“插入”→“模块”命令。

step 2 在模块中继续录入以下代码：

```
Sub 根据月份判断季度()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
 Dim Months As Byte
 Months = Application.InputBox("请输入月份，只能是数字", "月份", Month(Date),
, , , , , 1)
 MsgBox IIf(Months >= 1 And Months < 4, "一季度", IIf(Months >=4 And Months <
7, "二季度", IIf(Months >=7 And Months < 10, "三季度", IIf(Months >=10 And Months
< 13, "四季度", "录入错误"))))
End Sub
```

step 3 鼠标单击过程中任意位置，然后按<F5>键，程序会弹出如图7.8所示输入框，其默认值为1。在其中录入数字7，单击“确定”按钮后将弹出如图7.9所示信息框。

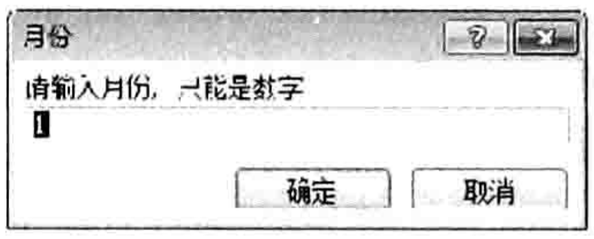


图 7.8 “月份”输入框

图 7.9 根据月份计算季度

思路分析：

本例中由于月份的下限是 1、上限是 12，因此声明变量 Months 时采用 Byte 型。

然后通过 Application.InputBox 函数创建一个输入框让用户输入月份值，由于月份是数值，Application.InputBox 的第 8 参数使用 1，从而强制用户只能录入数值。

在输入框中显示的默认值是当前月份，其中 Date 语句可以生成当前系统日期，将它配合 Month 函数使用可以提取本月的月份。

在执行判断时，由于季度是按范围来界定的，而不是单纯地等于某个值或者大于某个值，因此需要使用 And 运算符，而且 IIF 函数需要嵌套应用。

如果用户录入的值小于 1 或者大于 12，那么 IIF 会返回最后一个参数“录入错误”。

如果用户单击了“取消”按钮，那么其返回值为 0，同样会在信息框显示“录入错误”。

本例文件参见光盘：..\第七章\ 7-3 根据月份判断季度.xlsm

为了促进读者对 IIF 的理解，再提供一个案例。

案例要求：计算当前所用 Excel 的年份版本。

知识要点：Application.Version 属性、IIF 函数。

实现步骤：

step 1 按<Alt+F11>组合键打开 VBE 窗口，然后单击菜单中的“插入”→“模块”命令。

step 2 在模块中继续录入以下代码：

```
Sub Excel 版本()'代码存放位置：模块中
  MsgBox "Excel 20" & IIf(Application.Version = 11, "03", IIf(Application.
Version = 12, "07", IIf(Application.Version = 14, "10", IIf(Application.Version
= 15, "13", ""))))
End Sub
```

step 3 单击过程中任意位置，然后按<F5>键，程序会弹出当前 Excel 的年份版本号。假设在 Excel 2010 中执行以上过程，那么信息框中将显示“Excel 2010”。

思路分析：

Application.Version 属性代表 Excel 的版本号，它是一个文本形式的数字编号。Excel 2003 的编号是“11.0”，Excel 2007 的编号是“12.0”，Excel 2010 的编号是“14.0”，Excel 2013 的编号是“15.0”。当使用 Application.Version 与数值进行比较时，VBA 会将它自动转换成数值再比较。例如“MsgBox "11.0" = 11”的返回值是 True。

本例中采用了 4 个 IIF 函数嵌套使用，如果改用工作表函数 Vlookup 则可以直接完成，不需要嵌套，代码如下：

```
Sub Excel 版本 2()'代码存放位置：模块中
  MsgBox Application.Evaluate("=vlookup(" & Application.Version & ",{11,2003;
12,2007;14,2010;15,2010},2,false)")
End Sub
```

Evaluate 函数的功能是将字符串形式的公式转换成值，本例中 Application.Version 是表达式，不能直接插入到字符串中间，只能通过连接运算符“&”将它与原来的字符串连接起来，然后再作为 Evaluate 函数的参数参与运算。

本例文件参见光盘：..\第七章\7-4 计算当前 Excel 的年份版本号.xlsm

7.2.2　IIF 函数的限制

IIF 是一个函数，与工作表函数 IF 极其相似，但是它相对于 IF 函数在用法上稍有差异，主要体现在以下方面。

1. 第 3 参数是必选参数

IIF 的第 3 参数是必选参数，而工作表函数 IF 的第 3 参数是可选参数。

例如工作表中可以使用以下公式，它忽略了第 3 参数：

```
=IF(A1>=60,"及格")
```

但是使用 IIF 时不能用第 3 参数，否则将产生编译错误。

2. 是否检验第 3 参数

当第 1 参数的返回值为 True 时，IF 函数可以忽略第 3 参数，而 IIF 函数则会同时检验第 3 参数的值，如果第 3 参数存在错误值则会中断程序。例如以下语句：

```
IIF([a1] > [b1], [b1] / [a1], [a1] / [b1])
```

当 A1 中的值大于 0 且 B1 中的值为 0 时，程序会产生编译错误。因为 IIF 的特点是条件成立仍然检测第 3 参数；如果在单元格中使用工作表函数 IF 则不会产生任何错误。

3. 错误方式

当 IF 函数计算结果为错误值时，它会在单元格中产生对应的值，不影响其他单元格中的公式；而 IIF 的第 3 参数假设使用了 0 作为除数，那么整个过程都会中断并提示错误信息。

虽然 IF 函数比 IIF 函数更好用，但是在 VBA 中不能调用工作表函数 IF。

7.2.3　IF Then 语句的语法详解

IF Then 语句不是函数，而是条件判断语句，符合指定条件时一组指定的代码。如果有多句代码可利用冒号将它们显示在同一行中。

IF Then 语句的语法如下：

```
IF condition Then statements
```

参数 condition 和 statements 均为必选参数，缺一不可。其中条件语句 condition 是运算结果为 True 或 False 的表达式，如果 condition 为 Null，则 VBA 将其视为 False。

例如变量 A 大于 60，则变量 B 等于“及格”，其表达式为：

```
IF A > 60 Then B = "及格"
```

如果 A1 单元格中的字符数超过 3 个，那么将 A1 单元格的字符加粗再倾斜，代码如下：

```
If Len(Range("a1")) >= 3 Then Range("a1").Font.Bold = True: Range("a1").
Font.Italic = True
```

“Range("a1").Font.Bold = True”和“ Range("a1").Font.Italic = True”原本是两句代码，必须将它们写在同一行中，中间使用冒号分隔开。

如果将第二句代码放在第二行，那么第二句代码会在不符合条件时也执行。

7.2.4 IF Then 应用案例

IF Then 条件语句在工作中的应用比较广，现通过两个案例展示它的用法。

1. 案例：禁止打印“总表”以外的工作表

案例要求：禁止打印“总表”以外的工作表。

知识要点：Workbook_BeforePrint 事件、IF Then 条件语句。

实现步骤：

step 1 按<Alt+F11>组合键打开 VBE 窗口。

step 2 如果未显示工程资源管理器则按<Ctrl+R>组合键显示工程资源管理器，然后双击 ThisWorkbook 进入工作簿事件代码窗口。

step 3 在窗口中录入以下事件过程代码：

```
Private Sub Workbook_BeforePrint(Cancel As Boolean)'代码存放位置：ThisWorkbook
  If ActiveSheet.Name <> "总表" Then MsgBox "禁止打印": Cancel = True
End Sub
```

step 4 返回工作表界面，选择“总表”以外的任意工作表，然后单击“打印”按钮，程序将弹出如图 7.10 所示的提示信息，同时禁止工作表打印。

图 7.10　打印总表以外的工作表时的提示信息

思路分析：

Workbook_BeforePrint 事件是一个工作簿事件，代码必须放在 ThisWorkbook 窗口中，它会在发送打印命令之后、打印机响应打印命令之前触发。

在 Workbook_BeforePrint 事件中，参数 Cancel 用于控制是否允许打印，对当前工作簿中的所有工作表都生效。本例中的代码使用了条件语句作为限制，因此只有在工作表名称不等于“总表”时才生效。

CD-ROM 本例文件参见光盘：..\第七章\ 7-5 禁止打印总表以外的工作表.xlsm

2. 案例：允许 8 点到 18 点可以打开活动工作簿

案例要求：允许 8 点到 18 点可以打开活动工作簿，其他时段不允许打开。

知识要点：Workbook_Open 事件、IF Then 条件语句、Application.Quit 方法、Hour 函数。

实现步骤：

step 1 按<Alt+F11>组合键打开 VBE 窗口。

step 2　如果未显示工程资源管理器则按<Ctrl+R>组合键显示工程资源管理器，然后双击 ThisWorkbook 进入工作簿事件代码窗口。

step 3　在窗口中录入以下事件过程代码：

```
Private Sub Workbook_Open()'代码存放位置：ThisWorkbook
  If Hour(Now) < 8 Or Hour(Now) > 18 Then Application.Quit
End Sub
```

step 4　保存工作簿后再重新开启，如果当前时间小于 8 点或者大于 18 点，那么工作簿会自动关闭。

思路分析：

Excel 本身没有办法禁止打开工作簿，不过可以采用其他的办法变通一下——打开工作簿后检查条件，如果符合条件就自动关闭工作簿，从而变相达成目的。

基于以上分析，只能采用 Workbook_Open 事件来实现。本例在 Workbook_Open 事件中采用 If Then 语句判断当前的小时数是否小于 8 或者大于 18，当满足条件时采用 Application.Quit 方法退出 Excel。

工作簿事件过程的代码必须放在 Thisworkbook 代码窗口中，否则无法自动执行。

本例文件参见光盘：..\第七章\ 7-6 允许 8 点到 18 点开启工作簿.xlsm

7.2.5　IF Then Else 语句的语法与应用

IF Then 语句表示满足条件时执行指定的语句，而 IF Then Else 语句则表示满足条件时执行指定的语句，如果不满足条件要执行另一组语句。

1. IF Then Else 语句的语法详解

IF Then Else 语句有两种用法，包含单行模式和块形式。

单行模式的 IF Then Else 语句语法如下：

```
IF condition Then [statements][Else elsestatements]
```

其中 condition 代表条件，通常是值为 True 或者 False 的表达式，statements 和 elsestatements 分别代表满足条件时执行的语句和不满足条件时执行的语句。

块形式的 IF Then Else 语句的语法如下：

```
IF Condition Then
  [statements]
[Else
  [elsestatements]]
End IF
```

单行模式的条件语句不需要 End If 语句，它的所有代码都在同一行中；块形式的条件语句必须以 End If 结束，由于 statements、Else 和 elsestatements 都是可选参数，因此代码可能占据 2 行、3 行也可能是 5 行。

当代码少时应尽量使用单行模式的 IF Then Else 语句，反之采用块形式的 IF Then Else 语句。

2. IF Then Else 语句的应用案例

案例要求：利用代码打开“D:\生产表.xlsm”文件，然后将它的第一个工作表中的所有数据复制到活动工作簿的活动工作表已用数据区域的下方。如果不存在“D:\生产表.xlsm”或者该工作簿

的第一个工作表没有数据，要给予提示。

知识要点：IF Then Else 语句、Exit Sub 语句、InStr 函数、MkDir 语句。

实现步骤：

step 1 按<Alt+F11>组合键打开 VBE 窗口，然后单击菜单中的“插入”→“模块”命令。

step 2 在模块中录入以下代码：

```
Sub 打开文件且复制已用区域()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Const FileName As String = "D:\生产表.xls"
  Dim sht As Worksheet
  If Len(Dir(FileName)) = 0 Then
    MsgBox "未找到 " & FileName, vbInformation, "友情提示"
  Else
    Set sht = ActiveSheet
    With Workbooks.Open(FileName)
      If IsEmpty(.Worksheets(1).UsedRange) Then
          MsgBox FileName & "的第一个工作表是空表", vbInformation, "友情提示"
      Else
          .Worksheets(1).UsedRange.Copy sht.Cells(Rows.Count, 1).End(xlUp).Offset(1)
          .Close (False)
      End If
    End With
  End If
End Sub
```

step 3 鼠标单击过程中任意位置，然后按<F5>键，假设 D 盘不存在“生产表.xlsm”，那么程序会弹出如图 7.11 所示的信息框；假设工作簿存在，但是该工作簿的第一个工作表未使用过，那么会弹出如图 7.12 所示的信息框；如果以上两者都不是，那么程序会打开“生产表.xlsm”，然后将它的第一个工作表的已用区域的值复制到活动工作簿的活动工作表的已用区域下方。

图 7.11　未找到文件的提示

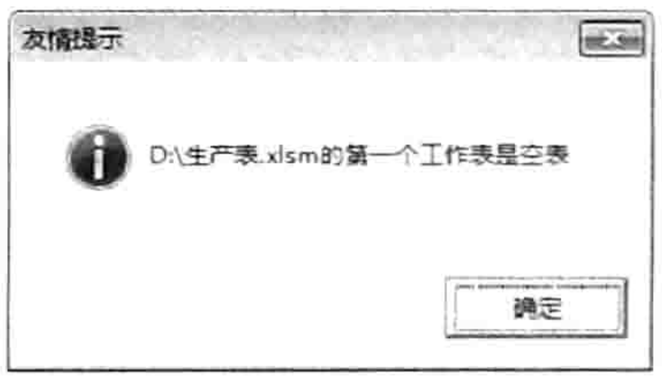


图 7.12　工作表是空白时的提示

思路分析：

如果只是打开文件并复制数据，代码会相当简单，不过编程需要注重代码的通用性，要防止某些意外情况的发生，包括指定的文件不存在或者要复制的数据也不存在，否则代码就可能在执行过程中出错。一段优秀的代码必须有足够的判断语句，对意外情况加以防范。

本例中首先采用了 Len+Dir 函数组合判断文件是否存在，其原理是 Dir 函数提取文件名称时遇到不存在的文件会返回一个空文本，因此利用 Len 函数计算 Dir 函数的返回值的长度即可判断指定的文件是否存在。

当已经存在文件时，本例采用了 Workbooks.Open 方法打开该文件，然后使用 With 语句引用该文件，使用 With 语句的目的是简化代码。Workbooks.Open 方法用于打开并返回一个工作簿对象，该对象会在后面出现 3 次，而使用 With 语句后仅需引用对象一次。

当打开工作簿后，不能立即复制工作表数据，而是采用 IsEmpty 函数判断工作表中是否有数据，对于空表没有必要执行复制操作。IsEmpty 函数的原本功能是判断变量是否初始化，本例用它来判

断工作表是否为空表。

当复制完数据后，前面所打开的工作簿必须关闭。Workbook.Close 方法用于关闭单个工作簿，将它的参数 SaveChanges 赋值为 False 可以加快关闭速度。

语法补充：

（1）Workbooks.Open 方法表示打开一个工作簿，其语法如下：

```
Workbooks.Open(FileName, UpdateLinks, ReadOnly, Format, Password,
WriteResPassword, IgnoreReadOnlyRecommended, Origin, Delimiter, Editable,
Notify, Converter, AddToMru, Local, CorruptLoad)
```

其中 FileName 参数表示要打开的工作簿名称，包含完整路径。

（2）IsEmpty 的功能是指出变量是否已经初始化，计算结果为 Boolean 值，它的语法如下：

```
IsEmpty (expression)
```

IsEmpty 用于指出变量是否已经初始化，在实际工作中极少使用 IsEmpty 函数。

（3）Workbook.Close 方法用于关闭单个工作簿，它的语法如下：

```
表达式.Close(SaveChanges, Filename, RouteWorkbook)
```

参数 SaveChanges 代表是否保存修改。当修改了工作簿时，将此参数赋值为 True 则表示保存修改，否则不保存。参数 Filename 代表文件名称（包含路径），相当于另存为一个新的文件。

本例文件参见光盘：..\第七章\7-7 打开文件且复制其已用区域的值.xlsm

知识补充：当代码比较多时，块形式的条件语句比单行模式的条件语句更利于阅读，也更美观。例如当单元格 A1 的值大于 60 时，则将 A1 的字体加粗倾斜并添加红色背景，通过两种形式书写代码，它们的效果比较如图 7.13 所示。

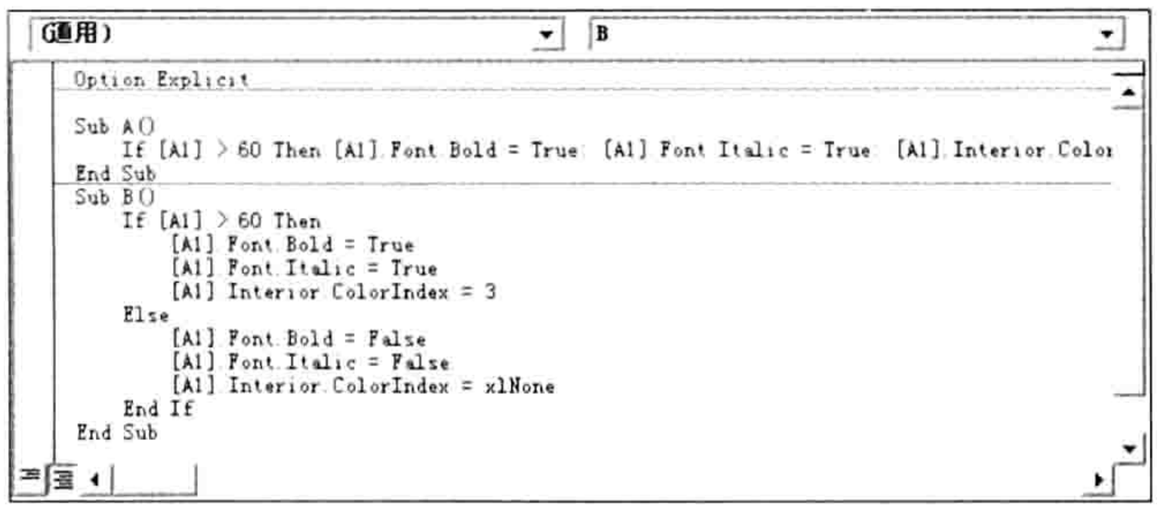


图 7.13　两种条件语句的书写方式比较

7.2.6　多条件嵌套的条件判断语句

条件语句不局限于 IF Then Else 语句这种双条件判断，其允许设置几个或者几十个条件。本节主要展示多条件嵌套的条件语句的语法，并提供相关的案例应用。

条件语句的嵌套分两种方式，方式一的语法如下：

```
IF Condition Then
  IF Condition Then
    [statements]
  [Else
    [elsestatements]]
  End IF
End IF
```

本嵌套方式表示在 IF 与 End IF 之间置入若干个块形式的 IF Then Else 条件语句，那么有多少个 IF 就会有多少个 End IF。

使用此类语句通常是指满足某个条件后继续执行条件判断，或者不满足某个条件时继续检测是否满足更多的条件。在文件“7-7 打开文件且复制其已用区域的值.xlsm”中已经使用过此类条件嵌套的判断方式，为了加深读者理解下面再提供一个相关的案例。

1. 案例：根据输入框的值创建文件夹

案例要求：弹出一个输入框，让用户输入文件名称，默认名称是当前日期，然后在 D 盘新建一个文件夹并且以该名称命名，如果用户输入了“\”、“/”、“:”、“*”、“?”、“"”、“<”、“>”、“|”这类非法字符时要及时提示用户，如果用户单击“消取”按钮则立即结束过程，不能创建名称为“False”的文件夹。

知识要点：IF Then Else 语句、Exit Sub 语句、InStr 函数、MkDir 语句。

实现步骤：

step 1 按<Alt+F11>组合键打开 VBE 窗口，然后单击菜单中的“插入”→“模块”命令。

step 2 在模块中录入以下代码：

```
'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
Sub 在输入框中指定名称然后创建文件夹()
  Dim FileName As String
   FileName = Application.InputBox("请输入文件夹名称", "文件夹名称", Format(Date,
"yyyy-mm-dd"))
   If FileName = "False" Then
      Exit Sub
   Else
  If InStr(FileName, ":") Or InStr(FileName, "\") Or InStr(FileName, "/") Or
InStr(FileName, "?") Or InStr(FileName, "*") Or InStr(FileName, """") Or
InStr(FileName, "|") Then
      MsgBox "文件夹名称不能包含（\/:*?""<>|）中的任意一个字符", vbInformation, "
友情提示"
    Else
      MkDir "d:\" & FileName
    End If
  End If
End Sub
```

step 3 单击过程中任意位置，然后按<F5>键，程序会弹出如图 7.14 所示的输入框，其默认值为当前日期。

step 4 单击“确定”按钮，程序会在 D 盘中创建一个名为“2014-01-13”的文件夹。

step 5 继续执行过程，在“文件夹名称”的输入框中输入“五月?报表”，单击“确定”按钮后程序会弹出如图 7.15 所示信息框。

图 7.14 “文件夹名称”对话框

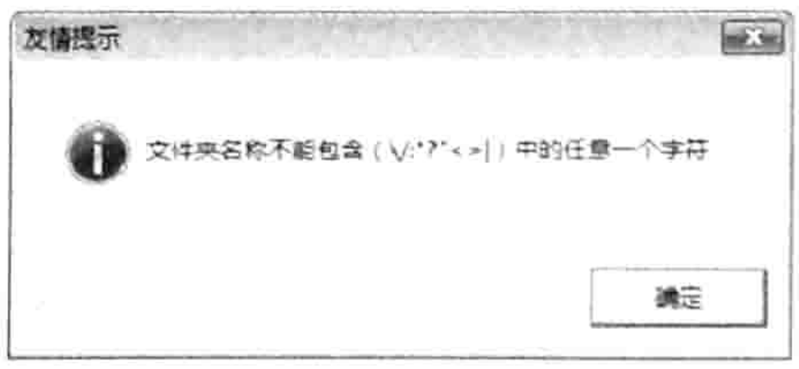


图 7.15 输入非法字符时自动提示

step 6 继续执行过程，在弹出“文件夹名称”输入框时单击“取消”按钮，程序会自动结束。

思路分析：

由于变量 FileName 的类型是 String，当在 Application.InputBox 创建的输入框中单击“取消”按钮时，赋值给变量 FileName 的值是“Flase”，为了避免用户单击“取消”按钮时创建一个名为“False”的文件夹，程序中使用了条件语句进行判断，符合条件时则通过“Exit Sub”语句结束过程，如果不符合条件则再嵌套一个条件语句判断用户是否录入了非法字符。

在判断用户是否录入非法字符时，本例采用了 Instr 函数计算“\”等 9 个字符在 FileName 中的位置，如果位置大于 0，那么说明 FileName 中包含非法字符。

由于执行逻辑判断时一切不等于 0 的数值都被当作 True 处理，因此代码中省略了“>0”。换而言之，代码“InStr(FileName, ":")”的完整书写方式是“InStr(FileName, ":")> 0”。

语法补充：

InStr 函数用于计算一个字符串在另一个字符串中最先出现的位置，它的语法如下：

```
InStr([start, ]string1, string2[, compare])
```

其中第 1 参数和第 4 参数是可选参数，通常忽略不用。第 2 参数代表接受搜索的字符串表达式，第 3 参数表示被搜索的字符。简单而言就是在第 2 参数中查询第 3 参数，返回第 3 参数在第 2 参数的起始位置。例如：

MsgBox InStr("ABCDB", "B")——返回值为 2，表示字母“B”在“ABCDB”中出现在第 2 位，按从左向右顺序取第一个目标。如果要按从右向左的顺序取第一个目标，应用以下代码：

MsgBox InStrRev("ABCDB", "B")——返回值为 5，要注意不是 1。

本例文件参见光盘：..\第七章\7-8 在输入框中指定名称然后创建文件夹.xlsm

事实上本例的代码并不完善，如果用户需要创建一个名为“False”的文件夹，在输入框中录入“False”后单击“确定”按钮，程序会自动结束。可按以下方式修改代码：

```
'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
Sub 在输入框中指定名称然后创建文件夹 2()
  Dim FileName As Variant
  FileName = Application.InputBox("请输入文件夹名称", "文件夹名称", Format(Date,
"yyyy-mm-dd"))
  If TypeName(FileName) = "Boolean" Then
    Exit Sub
  Else
    If InStr(FileName, ":") Or InStr(FileName, "\") Or InStr(FileName, "/") Or
InStr(FileName, "?") Or InStr(FileName, "*") Or InStr(FileName, """") Or
InStr(FileName, "|") Then
      MsgBox "文件夹名称不能包含（\/:*?""<>|）中的任意一个字符", vbInformation, "友
情提示"
    Else
      MkDir "d:\" & FileName
    End If
  End If
End Sub
```

思路分析：

本例中首先将变量FileName定义为变体型，当在Application.InputBox创建的输入框中单击“取消”按钮时，赋值给变量 FileName 的值是布尔值“False”，它的类别名称是“Boolean”。如果在输入框中输入了任意字符，赋值给变量 FileName 的值是用户输入的文字，其类别名称是“String”。

基于此原因，在过程中通过 TypeName 函数计算 Application.InputBox 的返回值的类别名称即可判断用户是单击了“取消”按钮还是输入了字符串“False”。

多条件嵌套的条件语句方式二的语法如下：

```
IF Condition Then
  [statements]
[ElseIF condition-n Then
  [elseifstatements] ...
[Else
  [elsestatements]]
End IF
```

语法列表中省略号代表可以继续添加更多的条件和对应的执行语句，方括号中的部分表示这是可选参数，允许忽略。

2. 案例：模仿复选框控制单元格中的钩与叉

案例要求：复选框可以实现单击时会产生一个“√”符号、再次单击会变成一个“×”符号，但在打印后却有一个方框，不太美观，而且当工作表中有大量的复选框时会导致工作簿体积增大，同时降低文件的开启速度。现要求在如图 7.16 所示的工作表的 B2:B20 区域中双击产生“√”符号，再次双击产生一个“×”符号。

知识要点：Intersect 方法、Len 函数、If Then Else 嵌套应用。

实现步骤：

step 1 在工作表标签处单击鼠标右键，在弹出的快捷菜单中选择“查看代码”命令从而进入工作表代码窗口。

step 2 在窗口中录入以下代码：

```
'①代码存放位置：工作表事件代码窗口②随书光盘中有每一句代码的含义注释
Private Sub Worksheet_BeforeDoubleClick(ByVal Target As Range, Cancel As Boolean)
  If NotIntersect(Target, Range("b2:b20")) Is Nothing Then
    Cancel = True
    If Len(Target) = 0 Or Target = "×" Then
      Target = "√"
    Else
      Target = "×"
    End If
   Else
    Cancel = False
  End If
End Sub
```

step 3 按<Alt+F11>组合键返回工作表界面，双击 B3 单元格，B3 单元格将会产生一个“√”符号，效果如图 7.17 所示。

	A	B	C
1	产名	合格	
2	A		
3	B		
4	C		
5	D		
6	E		
7	F		
8	G		
9	H		
10	I		

图 7.16　产品检验表

	A	B	C
1	产名	合格	
2	A		
3	B	√	
4	C		
5	D		
6	E		
7	F		
8	G		
9	H		
10	I		

图 7.17　双击录入“√”符号

step 4　再次双击 B3 单元格，B3 单元格的值将变成一个“×”符号。

step 5　双击 B2:B20 以外的任意单元格，将不会触发工作表的 BeforeDoubleClick 事件。

思路分析：

Worksheet_BeforeDoubleClick 事件属于工作表事件，仅对代码所在工作表生效，代码必须存放在工作表事件代码窗口中。

事件过程的参数 Target 代表被双击的单元格，Cancel 代表是否让单元格进入编辑状态。本例中 B2:B20 区域需要双击录入字符，因此将 Cancel 赋值为 True，从而禁止进入编辑状态，当双击的单元格处于 B2:B20 区域以外时才允许进入编辑状态。

在事件过程中，外层的条件判断语句用于判断当前双击的区域是否处于 B2:B20 之中，里层的条件语句用于判断何时应该输入“√”符号、何时应该输入“×”符号。由于需求是单元格中没有数据或者值为“×”时产生“√”符号，因此 If 后面必须书写两个条件，使用 Or 运算符将它们连接起来。

判断单元格是否为空白单元格有两个办法，其一是使用“Len(Range("a1")) = 0”，其二是使用“Range("a1") = ""”。

本例如果采用 IIF 函数替代里层的 IF Then Else 语句可以简化代码，代码如下：

```
'①代码存放位置：工作表事件代码窗口②随书光盘中有每一句代码的含义注释
Private Sub Worksheet_BeforeDoubleClick(ByVal Target As Range, Cancel As
Boolean)
  If Not Intersect(Target, Range("b2:b20")) Is Nothing Then
    Cancel = True
    Target = IIf(Len(Target) = 0 Or Target = "×", "√", "×")
  Else
    Cancel = False
  End If
End Sub
```

语法补充：

（1）Intersect 方法用于获取多个区域的交集，如果不存在交集则返回 Nothing。判断一个对象是否为 Nothing 时不用等号，而是用 IS 运算符。IS 运算符的语法如下：

```
result = object1 Is object2
```

其中 object1 和 object2 是两个对象，不能是数据或者数据变量。

（2）Not 也是一个运算符，它表示取反运算。例如将 True 变成 False，将 False 变成 True。它的语法如下：

```
Not expression
```

参数 expression 是一个逻辑值或者运算结果是逻辑值的表达式。

本例文件参见光盘：..\第七章\7-9 双击录入勾与叉.xlsm

7.2.7　Select Case 语法详解

Select Case 语句也是条件语句之一，当需要判断的条件比较多时，它比 IF Then Else 语句更最强大，书写方式也更简单。

1. 语法详解

Select Case 语句的语法如下：

```
Select Case testexpression
   [Case expressionlist-n
 [statements-n]] ...
   [Case Else
 [elsestatements]]
End Select
```

Select Case 语句包括四部分，每部分详细含义如表 7-6 所示。

表 7-6　Select Case语句各部分含义

部分	描述
testexpression	必要参数。任何数值表达式或字符串表达式
expressionlist-n	如果有 Case 出现，则为必要参数。其形式为 expression，expression To expression，Is comparisonoperator。expression 的一个或多个组成的分界列表。To 关键字可用来指定一个数值范围。如果使用 To 关键字，则比较小的数值要出现在 To 之前。使用 Is 关键字时，则可以配合比较运算符（除 Is 和 Like 之外）来指定一个数值范围。如果没有提供，则 Is 关键字会被自动插入
statements-n	可选参数。一条或多条语句，当 testexpression 匹配 expressionlist-n 中的任何部分时执行
elsestatements	可选参数。一条或多条语句，当 testexpression 不匹配 Case 子句的任何部分时执行

在以上语法列表中，省略号代表可以使用多个条件。只要有一个 Case 就需要有一个 statements-n，Case 和 statements-n 分别表示条件及符合条件时要执行的语句。

其中 elsestatements 表示不符合指定条件时的执行语句，是可选参数。

在 Select Case 的多个参数中最复杂的是 expressionlist-n 部分，它有多种表达形式，包括：

expression——直接声明一个条件值，例如 5；

expression To expression——声明一个条件的范围，例如 5～10；

Is comparisonoperator——声明一种比较方式，例如 is>5。

下面的实例可以展示参数中 expressionlist-n 部分的多种表达形式。

2. 案例：多条件时间判断

案例要求：根据当前的时间判断现在是上午、中午、下午、晚上还是午夜。其中 7～10 点算上午、11～12 点算中午、13～17 点算下午、18～23 点算晚上、24～6 点算午夜。

知识要点：Select Case 语句、Now 函数、Hour 函数。

实现步骤：

step 1　在工作表标签处单击鼠标右键，在弹出的快捷菜单中选择“查看代码”命令从而进入工作表代码窗口。

step 2　在窗口中录入以下代码：

```
Sub 时间()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Dim Tim As Byte, msg As String
  Tim = Hour(Now)
  Select Case Tim
  Case 7 To 10
    msg = "上午"
  Case 11, 12
    msg = "中午"
```

```
  Case 13 To 17
    msg = "下午"
  Case 18 To 23
    msg = "晚上"
  Case 24, Is < 7
    msg = "午夜"
  End Select
  MsgBox "现在是: " & msg
End Sub
```

step 3　鼠标单击过程中任意位置，然后按<F5>键，程序会在信息框中显示当前时间状态。

思路分析：

在以上代码中，“Case 7 To 10”表示当前时间在 7 点到 10 点，用于限定一个范围；“Case 11，12”表示当前时间为 11 点或者 12 点，可用于限定具体的数值；“Case 24, Is < 7”代表当前时间为 24 点或者在 7 点之前，两者中符合一个条件即可。

本例中不存在例外的情况，所以忽略了“Case Else”语句。

语法补充：

（1）Hour 函数用于从时间中提取小数时，它的参数只能是时间。代码“Hour(#13:50:50#)”的返回值为 13。

（2）Now 函数返回当前的日期和时间值，同时包含日期和时间。Now 函数没有参数，是否使用括号不影响函数的计算结果。

CD-ROM　本例文件参见光盘：..\第七章\ 7-10 多条件时间判断.xlsm

3. 案例：以指定格式的今日日期显示 Excel 标题

案例要求：Excel 的标题默认显示为“Microsoft Excel”，现要求显示为今日日期，并且日期需要提供数字格式、中文小写和中文大写三种方式供用户选择。

知识要点：Select Case 语句、工作表函数 Text、标签、Goto 语句、Application.Caption。

实现步骤：

step 1　在工作表标签处单击鼠标右键，在弹出的快捷菜单中选择“查看代码”命令从而进入工作表代码窗口。

step 2　在窗口中录入以下代码：

```
Sub 以今日日期显示标题()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Dim DateStr As String
Restart:
  Select Case Application.InputBox("请指定日期显示方式: " & Chr(10) & "1: 数字日
期" & Chr(10) & "2: 中文小写" & Chr(10) & "3: 中文大写", "日期显示方式", 1, , , , , , 1)
  Case 1
    DateStr = "yyyy-mm-dd"
  Case 2
    DateStr = "[DBNum1] yyyy年m月d日"
  Case 3
    DateStr = "[DBNum2] yyyy年m月d日"
  Case Else
    MsgBox "录入错误，请重新录入", vbInformation
```

```
    GoTo Restart
  End Select
  Application.Caption = WorksheetFunction.Text(Date, DateStr)
End Sub
```

step 3 用鼠标单击过程中任意位置，然后按<F5>键，程序会弹出如图 7.18 所示的输入框。

step 4 在输入框中输入“2”然后单击“确定”按钮，程序会将 Excel 的原有标题“Microsoft Excel”修改为中文小写格式的当前日期，效果如图 7.19 所示。

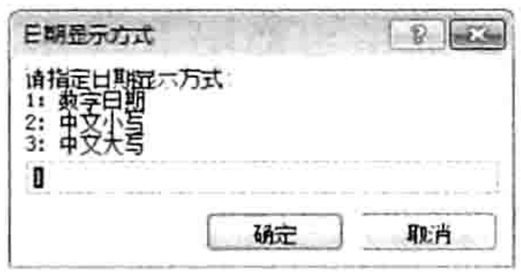


图 7.18 “日期显示方式”输入框

图 7.19 修改后的 Excel 标题栏

思路分析：

由于需求是将日期格式化为三种样式供用户选择，因此首先通过 Application.InputBox 方法创建一个输入框，让用户在 1、2、3 这三个选项之间选择，然后使用 Select Case 语句根据用户的录入值设置格式信息。

本例的条件应该是四种，包括为用户提供的 1、2、3 这三个选项，以及用户输入错误时的其他选项。对于前三种选项，分别对变量 DateStr 赋值为“yyyy-mm-dd”、“[DBNum1] yyyy 年 m 月 d 日”和“[DBNum2] yyyy 年 m 月 d 日”即可，对于第四种条件本例采用的办法是借用标签和 Goto 语句让用户重新输入，直到输入正确时再执行后续的代码。

标签的功能是标示一个位置，通常配合 Goto 语句使用，而 Goto 语句又通常配合条件语句使用，因此标签与 Goto 搭配使用后可以指定一个位置，然后在符合某条件时让程序跳转到该位置处继续执行。

在本例中，当用户输入 1、2、3 以外的任何数字时，过程中的 Goto 语句会改变程序的执行流程，让程序跳转到标签 Restart 处，从 Restart 标签的下一行代码继续执行。如果用户继续输入 1、2、3 以外的数值，程序会继续跳转到标签 Restart 处让用户重新输入，直到输入正确范围内的值时停止。

当用户指定了正确的日期格式时，调用工作表函数 Text 将日期 Date 进行格式化即可。尽管 VBA 的 Format 函数也用于格式化数字，但是它不支持“[DBNum1]”和“[DBNum2]”。

Text 属于工作表函数，前面要加“WorksheetFunction.”，它的功能和 VBA 的 Format 函数相近，不过比 Format 更强大。

Application.Caption 表示 Excel 界面的标题，它是一个可读、可写的属性。

语法补充：

工作表函数 Text 的功能是将一个字符串转换成指定格式，通常也称它为格式化函数。其语法如下：

```
TEXT(value, format_text)
```

其中参数 Value 是待格式化的字符串，可以是数值也可以是文本。参数 format_text 表示格式信息。例如“0”表示只显示整数部分，“0.00”表示只显示整数和两位小数，“yyyy 年 m 月 d 日”则表示显示为日期格式，“[DBNum2]0”表示将整数部分显示为中文大写。

本例文件参见光盘：..\第七章\7-11 按格式显示今日日期标题.xlsm

7.2.8　Select Case 与 IF Then Else 之比较

Select Case 与 IF Then Else 两者都可以实现多条件判断，但在使用中各有优势。

IF Then Else 的优势在于可以配合 And 与 Or 运算符实现多条件判断，而 Select Case 语句执行条件判断时只能实现 Or 运算符的类似功能，不能实现 And 运算符的类似功能。IF Then Else 还允许设置多个比较对象，而 Select Case 语句只能设置单个比较对象。

例如 A2 单元格存放性别，B2 单元格存放身高，要求女性的身高大于 1.55 米并且小于等于 1.70 米才算合格，男性的身高则大于 1.7 米并且小于等于 1.80 米才算合格。那么采用 IF Then Else 语句可以实现此条件判断，完整代码如下：

```
If Range("a2") = "女" And Range("b2") > 1.55 And Range("b2") <= 1.7 Then
  MsgBox "合格"
ElseIf Range("a2") = "男" And Range("b2") > 1.7 And Range("b2") <= 1.8 Then
  MsgBox "合格"
Else
  MsgBox "不合格"
End If
```

如果改用 Select Case 语句则无法实现。

Select Case 语句的优势在于对同一对象设置多个条件时比 IF Then Else 语句更简洁，读者可以根据自己的需求决定使用何种条件语句。

7.2.9　借用 Choose 函数简化条件选择

Choose 函数的作用是从参数列表中选择一个值，它只能返回值，不能根据条件执行指定的过程或者语句。Choose 函数类似于 IIF 函数，而有别于 IF Then Else 语句和 Select Case 语句。

Choose 函数的基本语法如下：

```
Choose(index, choice-1[, choice-2, ... [, choice-n]])
```

其中 index 为必选参数，可用数值表达式或数字做参数，参数的值必介于 1 和可选择的项目数量之间。如果超过可选项目个数将产生错误结果。

除 index 以外的所有参数都是可选项目，可选项目中第一个参数为必选参数其余为可选参数。

Choose(3, 1, 2, 3, 4)——返回 3，表示从 1、2、3、4 这 4 个项目中选择第 3 个值，

Choose((3 + 2) / 0.5, "A", "C", "D", , , , , , "F", "G", , , "I", "J")——返回 G。index 参数的计算结果为 10，因此返回项目表中第 10 个值，即字母 G。

根据条件返回值时，Choose 函数往往比 IIF 嵌套、IF Then Else 语句和 Select Case 语句都简单。

例如对单元格设置颜色，由用户在 5 个可选项目之间选择。如果用户输入 1 则为红色，输入 2 则为蓝色，输入 3 则为灰色，输入 4 则为棕色，输入 5 则为绿色。如果使用 Choose、IIF、IF…Then…和 Select Case 语句来实现，则四段代码如下：

```
'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
Sub 设置单元格颜色 1()'Choose 法
  Dim 颜色 As Byte
  颜色 = Application.InputBox("请选择颜色：" & Chr(10) & "1:红色" & "   2:蓝色" &
Chr(10) & "3:灰色" & "   4:棕色" & Chr(10) & "5:绿色", "指定颜色", 1, , , , , , 1)
  Range("A1").Interior.ColorIndex = Choose(颜色, 7, 5, 15, 40, 4)
End Sub
Sub 设置单元格颜色 2()  'IIF 法
```

```
  Dim 颜色 As Byte
  颜色 = Application.InputBox("请选择颜色: " & Chr(10) & "1:红色" & "   2:蓝色" &
Chr(10) & "3:灰色" & "   4:棕色" & Chr(10) & "5:绿色", "指定颜色", 1, , , , , , 1)
  Range("A1").Interior.ColorIndex = IIf(颜色 = 1, 7, IIf(颜色 = 2, 5, IIf(颜色
= 3, 15, IIf(颜色 = 4, 40, IIf(颜色 = 5, 4, xlNone)))))
End Sub
Sub 设置单元格颜色3()  'IF...Then...Else法
  Dim 颜色 As Byte
  颜色 = Application.InputBox("请选择颜色: " & Chr(10) & "1:红色" & "   2:蓝色" &
Chr(10) & "3:灰色" & "   4:棕色" & Chr(10) & "5:绿色", "指定颜色", 1, , , , , , 1)
  If 颜色 = 1 Then
     Range("A1").Interior.ColorIndex = 7
  ElseIf 颜色 = 2 Then
     Range("A1").Interior.ColorIndex = 5
  ElseIf 颜色 = 3 Then
     Range("A1").Interior.ColorIndex = 15
  ElseIf 颜色 = 4 Then
     Range("A1").Interior.ColorIndex = 40
  ElseIf 颜色 = 5 Then
     Range("A1").Interior.ColorIndex = 4
  End If
End Sub
Sub 设置单元格颜色4()  'Select Case法
  Dim 颜色 As Byte
  颜色 = Application.InputBox("请选择颜色: " & Chr(10) & "1:红色" & "   2:蓝色" &
Chr(10) & "3:灰色" & "   4:棕色" & Chr(10) & "5:绿色", "指定颜色", 1, , , , , , 1)
  Select Case 颜色
     Case 1
     Range("A1").Interior.ColorIndex = 7
  Case 2
     Range("A1").Interior.ColorIndex = 5
 Case 3
     Range("A1").Interior.ColorIndex = 15
  Case 4
     Range("A1").Interior.ColorIndex = 40
  Case 5
     Range("A1").Interior.ColorIndex = 4
  End Select
End Sub
```

执行以上任意程序时，会弹出如图 7.20 所示对话框，如果录入 1～5 中的任意数值，那么单元格 A1 会产生对应的颜色。

以上四个过程可以产生同样的效果，其中使用 Choose 函数最简捷。

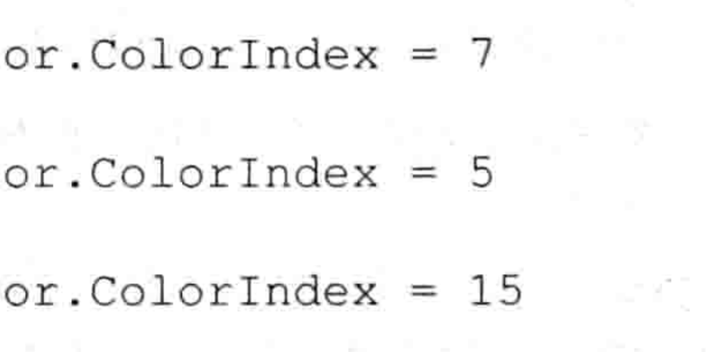


图 7.20　要求用户指定 A1 需要显示的颜色

本例文件参见光盘：..\第七章\ 7-12 按指定值设置单元格背景色.xlsm

不过 Choose 函数也有它的限制。

使用 Choose 函数进行条件选择时，虽然它只返回一个值，但 Choose 函数会计算列表中的每个选项，IIF 函数也有类似的副作用。

通过以下实例可以证实这个副作用的存在：

```
Choose(2, 12 + 2, 356, 12 / 0)
```

代码中 Choose 函数的第 1 参数是 2，因此其结果是列表中的第 2 个值，但是 Choose 函数会将每个选项分别计算一次，任意一个选项出错都会中断整个过程，使程序无法继续执行。

如果改用 Select Case 语句则不存在此副作用，代码如下：

```
Sub 选择性取值()
  a = 2
  Select Case a
  Case 1
    MsgBox 12 + 2
  Case 2
    MsgBox 256
  Case 3
    MsgBox 12 / 0
  End Select
End Sub
```

7.3　循环语句

循环语句用于重复执行一系列代码，从而批量地执行任务。循环语句在实际工作中应用面极广，而且因为循环语句不可能通过录制宏产生，所以必须潜心掌握它的语法与结构。

循环语句主要包括以下几类：

- For Next；
- For Each Next；
- Do Loop；
- Do While Loop。

7.3.1　For Next 语句

在工作中，我们可以使用 For Next 语句去重复一组语句，它的循环次数可以自由指定，循环执行的代码也可以自由指定。

1. 法语详解

For Next 循环语句的基本语法如下：

```
For counter = start To end [Step step]
    [statements]
  [Exit For]
    [statements]
Next [counter]
```

语法列表中包括六个重要的部分，其参数详解如表 7-7 所示。

表 7-7　For Next语法详解

参数	功能描述
counter	必要参数，用作循环计数器的数值变量。这个变量不能是 Boolean 或数组元素
start	必要参数，counter 的初值
End	必要参数，counter 的终值
step	可选参数，counter 的步长。如果没有指定，则 step 的默认值为 1
statements	可选参数，放在 For 和 Next 之间的一条或多条语句，它们将被执行指定的次数
Exit For	可选参数，终止循环
Next counter	Next 是必选的，counter 是可选的。Next 表示当前循环结束，即将执行入一轮循环

其中 counter 是计数器变量，由用户声明；start 和 end 表示计数器的起止范围，用户可以根据需求定义这个范围；step 表示步长值，即计数器累加的单位，它可以是正数，也可以是负数，但是不能为 0，而且不能大于 End、不能小于 start。

当循环开始后，计数器逐步累加，累加值由步长值决定；statements 则是循环语句的核心，虽然它是可选参数，然而如果忽略此参数，则所有循环都失去意义。

以下简单代码可以作为循环语句的一个通用模板，可以通过它理解循环的执行流程和循环的意义。

需求：利用 VBA 的循环语句统计从 1 累加到 100 的值，代码如下：

```
Sub 累加1到100()'代码存放位置：模块中
  Dim Item As Integer, SumValue As Integer '声明变量
  For Item = 1 To 100 Step 1     '指定循环的起止范围和步长值
    SumValue = SumValue + Item   '累加计数器(其中 SumValue 的初始值为 0)
  Next Item                '执行下一个
  MsgBox SumValue     '报告累加结果
End Sub
```

在该过程中，循环的范围是 1～100，循环的步长为 1。由于步长值的默认值即为 1，所以本例中的 Step 参数也可以忽略不写。

为了获取 1～100 的累加值，需要使用一个中间变量 SumValue，该变量在累加初期的值为 0，当进入循环语句之后，每循环一次它的值会累加一次 Item，直到循环结束。

如果只累加 1～100 的偶数，那么代码如下：

```
Sub 累加1到100之间的偶数()'代码存放位置：模块中
  Dim Item As Integer, SumValue As Integer '声明变量
  For Item = 2 To 100 Step 2 '指定循环的起止范围和步长值。由于只能取偶数，因此从 2 开始，步长值为 2
    SumValue = SumValue + Item '累加计数器(其中 SumValue 的初始值为 0)
  Next Item                '执行下一个
  MsgBox SumValue     '报告累加结果
End Sub
```

本例文件参见光盘：..\第七章\7-13 累加 1 到 100 之间的自然数和偶数.xlsm

2. 循环的方向对循环结果的影响

在多数情况下从大向小循环和从小到大循环能取得一样的结果，但是在删除或者插入 Range 对象时，循环的方向不对会导致无法取得预期的结果。

例如通过循环删除工作表中前 20 行（仅演示用），那么使用不同的循环方法会得到不同的结果。以下提供两种采用不同循环方式的过程作为对比，操作步骤如下。

step 1　在 A1 和 A2 单元格中分别录入 1 和 2，然后将它向下填充到 A20 单元格。

step 2　单击菜单中的“插入”→“模块”命令，并且在模块中录入以下代码：

```
Sub 循环删除前 20 行 A()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
    Dim Item As Integer
    For Item = 1 To 20
        Rows(Item).Delete
    Next Item
End Sub
Sub 循环删除前 20 行 B()
    Dim Item As Integer
    For Item = 20 To 1 Step -1
        Rows(Item).Delete
    Next Item
End Sub
```

step 3　返回工作表界面，按<Alt+F8>组合键打开“宏”对话框，从中选择过程“循环删除前 20 行 B”，然后单击“执行”按钮，结果如图 7.21 所示。

step 4　重复前一个步骤执行第二个过程“循环删除前 20 行 A”，其执行结果如图 7.22 所示。

	A	B
1	2	
2	4	
3	6	
4	8	
5	10	
6	12	
7	14	
8	16	
9	18	
10	20	

图 7.21　步长为正数的执行效果

	A	B
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		

图 7.22　步长为负数的执行效果

过程“循环删除前 20 行 B”完全符合预期的效果，而过程“循环删除前 20 行 A”则仅删除了前 20 行的一半，显然不符合需求。产生这种差异的原因如下：

删除行或者插入行这类操作会破坏对象的结构，例如删除第 1 行后原来的第 2 行会变成第 1 行，原来的第 3 行变则变成第 2 行。此时继续删除第 2 行其实就是删除原来的第 3 行。当循环结束后会发现刚好遗漏了一半。如果从后向前循环则不会破坏这 20 行的结构，可以达到预期的结果。

本例文件参见光盘：..\第七章\ 7-14 使用循环语句删除前 20 行.xlsm

3. 不确定的起止范围

循环语句的起止范围并非都是明确的，也可能需要通过计算才知道范围。

例如罗列今年所有星期天的日期，代码如下：

```
Sub 罗列今年所有周日的日期()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Dim Item As Long, StartLng As Long, EndLng As Long, i As Byte
```

```
  StartLng = DateSerial(Year(Date), 1, 1)
  EndLng = DateSerial(Year(Date) + 1, 1, 0)
  For Item = StartLng To EndLng
    If Weekday(Item, 2) = 7 Then
      i = i + 1
      Cells(i, 1) = Format(Item, "yyyy-mm-dd")
    End If
  Next Item
End Sub
```

思路分析：

以上过程中循环语句的起始值是本年度的第一天的日期值，终止值是本年度最后一天的日期值。由于每一年的第一天和最后一天都是不相同的，因此两个数值都只能通过计算来得到。

计算第一个周日的日期采用代码 DateSerial(Year(Date), 1, 1)，其中 Year(Date)代表今年的年份。

计算今年最后一天的日期值采用代码 DateSerial(Year(Date) + 1, 1, 0)，其中 Year(Date) + 1 代表明年的年份，整句代码的含义是明年的第 0 天，也就是今年的最后一天。

当确定好循环语句的起止范围后，重点在于提取周日所对应的日期值。Weekday 函数用于计算指定日期属于星期几，当它的第 2 参数为 2 时表示采用中国式星期制度——星期一作为一周的第一天。当 Weekday 的返回值为 7 时则表示该日期属于一周的第七天，即星期日。

找出星期日后，需要将该日期值从 A1 开始排列在 A 列中，因此首先对变量 i 累加 1，然后将日期值通过 Format 函数格式化为星期样式，再输出到第 i 行的第 1 列中。由于代码“i = i + 1”处于循环体中，因此每找到一个符合条件的日期值变量 i 就会累加 1，从而使 Cells(i,1)所引用的单元格也相应地变化，从 A1 变为 A2、A3、A4……

语法补充：

（1）DateSerial 函数用于将代表年、月、日的 3 个参数转换成相应的日期值，其语法如下：

```
DateSerial(year, month, day)
```

3 个参数分别代表年、月、日，其中 year 参数只能是 100～9999 中的整数。

DateSerial 函数具有智能纠错功能，当第 2 参数的值小于 1 或者大于 12 时，它会将月份自动调整为 1～12 中的值，同时对代表年份的第 1 参数相应地增减；当第 3 参数小于 1 或者大于 31 时也会同样地智能调整。例如：

MsgBox DateSerial(2012, 8, 70)——2012 年 8 月的第 70 天也就是 10 月的第 9 天，DateSerial 函数会自动将日期调整为有效的日期“2012-10-9”；

MsgBox DateSerial(2012, 2, 31)——2012 年 2 月只有 29 天，因此 2 月 31 日会自动转换成 3 月 2 日。

（2）Weekday 函数用于计算某个日期是星期几，它的语法如下：

```
Weekday(date, [firstdayofweek])
```

第 1 参数 date 代表日期，第 2 参数是可选参数，代表将星期几算作为一周的第一天。参数赋值为 0 时表示星期日作为一周的第一天，赋值为 2 时表示星期一作为一周的第一天，赋值为 3 时代表星期二作为一周的第一天……

本例文件参见光盘：..\第七章\ 7-15 罗列本年度所有周日的日期.xlsm

4. 根据需求中途退出循环

在一个比较大的范围中循环时会消耗比较多的内存，而根据条件适时地退出循环则可以避免不必要的消耗。下面举一个实例说明如何在需要时退出循环。

Excel 中有很多操作都会受限于合并单元格，例如排序、筛选、分列等。在进行此类操作前，最好检查一下活动工作表的已用区域中是否存在合并单元格，如果有则提示用户。以下代码即用于判断活动工作表的已用区域中是否存在合并单元格：

```
'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
Sub 判断活动工作表的已用区域是否存在合并单元格()
  With ActiveSheet.UsedRange
    Dim i As Long
    For i = 1 To .Count
      If .Cells(i).MergeArea.Address <> .Cells(i).Address Then Exit For
    Next i
    MsgBox .Address & IIf(i < .Count, "", "不") & "存在合并单元格"
  End With
End Sub
```

思路分析：

由于操作对象是活动工作表的已用区域，因此首先通过 With 语句引用 ActiveSheet.UsedRange 对象。然后利用 For Next 循环语句遍历已用区域的每一个单元格，循环的起始值是 1，终止值是已用区域的单元格数量——Range.Count。

判断单元格是否为合并单元格的办法是提取单元格的地址（Range.Address 属性），以及该单元格的合并区域的地址（Range.MergeArea.Address），然后将两者做比较，如果相同则表示不是合并单元格。因此本例的循环语句中以“.Cells(i).MergeArea.Address <> .Cells(i).Address”作为是否结束循环的条件，当找到一个合并单元格后立即通过 Exit For 语句结束循环。

本例中使用 Exit Sub 语句的目的是避免浪费程序的执行时间。当已用区域中有多个合并单元格时，只要发现第一个合并单元格后就已经可以下结论：活动工作表中存在合并单元格，不必继续检查其他单元格，因此此时使用 Exit Sub 语句结束循环可以缩短程序执行时间。

在过程的最后，判断已用区域中是否存在合并单元格时以计数器 i 的值为判断依据，如果发现合并单元格并且中途结束循环，那么计数器的值必定小于已用区域的单元格数量。

其实 VBA 中有一个专用函数 MergeCells 用于判断区域中是否存在合并单元格，本例仅用于演示循环语句的用法。

语法补充：

（1）Range.MergeArea 是一个 Range 对象，该对象代表包含指定单元格的合并区域。如果指定的单元格不在合并区域内则返回单元格本身。

MsgBox Range("a2").MergeArea.Address——如果 A2 不是合并单元格，那么返回值为“$A$2”，如果将 A2:B4 区域合并，那么以上代码将返回“$A$2:$B$4”。

（2）Range.Count 属性代表区域中的单元格数量，它是只读属性，不允许被修改。

（3）Exit For 语句代表结束 For Next 循环语句，它只能存放在 For 与 Next 语句之间，通常配合条件语句使用，表示符合某条件时结束循环语句。

本例文件参见光盘：..\第七章\7-16 判断区域中是否存在合并单元格.xlsm

5. For Next 循环的嵌套应用

循环也可以像条件语句一样多层嵌套使用，在每一层循环中可以按需求随时中断循环。本例是双层循环的应用。

假设在工作簿的第一个工作表中有本期 9 个班的三好学生名单，如图 7.23 所示。需要将这些名单分置于 9 个工作表中，并且将学生姓名纵向存放，每个表必须以班名进行命名。

实现以上需求可以使用 For next 循环语句的双层循环来完成，完整代码如下：

```
Sub 分班()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Dim 班 As Byte, 学生 As Byte
  For 班 = 2 To Cells(Rows.Count, 1).End(xlUp).Row
    With Sheets.Add(after:=Sheets(Sheets.Count))
      .Name = WorkSheets(1).Cells(班, 1).Value
      For 学生 = 2 To WorkSheets(1).Cells(班, 1).End(xlToRight).Column
        .Cells(学生 - 1, 1) = WorkSheets(1).Cells(班, 学生)
      Next 学生
    End With
  Next 班
End Sub
```

思路分析：

由于班级数量和每个班的学生数量都不足 100，因此本例中“班”和“学生”两个变量都声明为 Byte 型。

本例使用了双层循环来分班，外层循环的功能是遍历班所在区域，以班级作为工作表名称逐一新建工作表；里层循环的功能是遍历每个班级所对应的学生姓名，将它们逐一赋值到班级对应的工作表中。

外层循环的起始值为 2、终止值由 A 列最后一个非空单元格的行号决定，代码“Cells(Rows.Count, 1).End(xlUp).Row”即为 A 列最后一个非空单元格的行号。

Sheets.Add 方法用于创建新表，本例中对它的 after 参数赋值为 Sheets(Sheets.Count)表示新建的工作表存放在最后一个表的右边，当循环结束后所有新表的顺序会保持与 A 列的班级顺序一致。

里层循环的起始值仍然是 2，终止值由该行的最后一个非空列的列号决定，代码“WorkSheets(1).Cells(班, 1).End(xlToRight).Column”即代表最后一个非空列的列号。代码中使用了 WorkSheets(1)是因为通过 Sheets.Add 方法创建工作表时会改变活动工作表，如果不加 WorkSheets(1)，那么代码“Cells(班, 1).End(xlToRight).Column”只能引用新工作表中的值，而不再是针对“三好学生”工作表。

本例过程的执行结果如图 7.24 所示。

	A	B	C	D	E	F	G	H	I	J
1	班级	三好学生								
2	A班	赵	钱	孙	李	周	吴	郑		
3	B班	王	冯							
4	C班	陈	褚	卫	蒋	沈	韩	杨	朱	秦
5	D班	尤	许	何						
6	E班	吕	施	张	孔	曹				
7	F班	严	华	金	魏	陶	姜	戚		
8	G班	谢	邹	喻	柏	水	窦	章		
9	H班	云	苏	潘	葛					
10	I班	奚	范	彭						

三好学生

图 7.23　三好学生名单

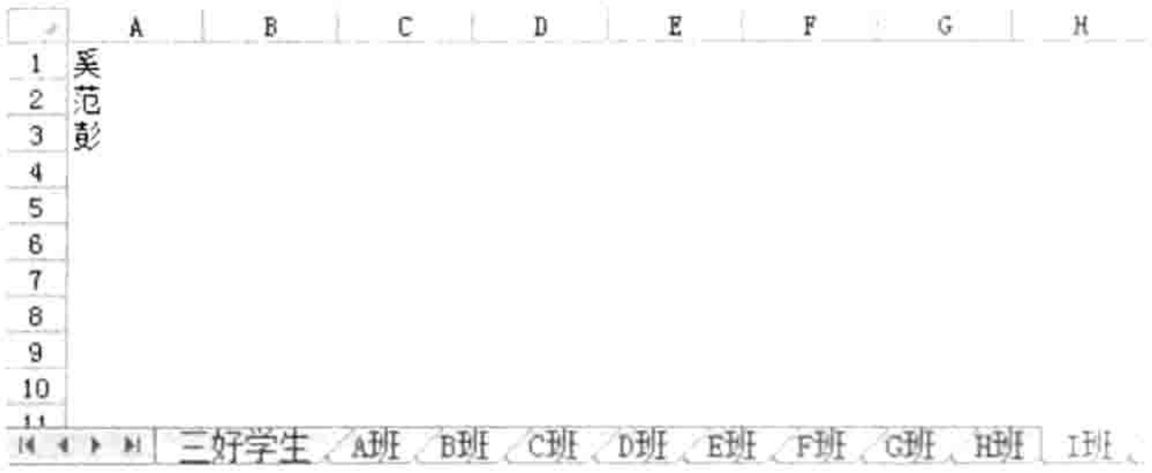


图 7.24　将三好学生分置于 9 个工作表中

语法补充：

（1）Sheets.Add 方法用于新建工作表、图表或宏表，使用最多的是新建工作表。

Sheets.Add 方法的语法如下：

```
Sheets.Add(Before, After, Count, Type)
```

其中参数 Before 用于指定工作表的对象，新建的工作表将置于此工作表之前；After 参数用于指定工作表的对象，新建的工作表将置于此工作表之后，它不能与 Before 参数同时使用；参数 Count

表示要添加的表的数量，默认值为 1，最大值不能超过 255；参数 Type 用于指定表的类型，默认值为 xlWorksheet。

Sheets.Add 方法和 WorkSheets.Add 方法的功能一致。

（2）Worksheet.Name 属性代表工作表的名称，它既可读也可写。在本例中对此属性赋值表示重命名工作表。在命名时要注意不能包含“\”、“/”、“?”、“*”、“!”、“:”等字符。

本例文件参见光盘：..\第七章\7-17 分班.xlsm

6. For Next 循环的综合应用：创建工作表目录

案例要求：对工作簿中的所有工作表创建目录，单击目录中任意单元格时可以进入对应的工作表中。

知识要点：For Next 循环、WorkSheets.Add 方法、Hyperlinks.Add 方法。

实现步骤：

step 1　按<Alt+F11>组合键进入 VBE 窗口，然后单击菜单中的“插入”→“模块”命令。

step 2　在模块中录入以下代码：

```
Sub 建立目录()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Dim i As Integer
  For i = 1 To sheets.Count
    If Sheets(i).Name = "工作表目录" Then GoTo Mulu
  Next
  Worksheets.Add Worksheets(1)
  ActiveSheet.Name = "工作表目录"
Mulu:
  Worksheets ("工作表目录").Range("A:B").Clear
  For i = 2 To Worksheets.Count
      Worksheets ("工作表目录").Cells(i - 1, 1).Value = i - 1
      Worksheets ("工作表目录").Hyperlinks.Add Anchor:= Worksheets("工作表目录").
Cells(i - 1, 2), Address:="", SubAddress:="'" & Worksheets(i).Name & "'!A1",
TextToDisplay:=Worksheets(i).Name, ScreenTip:="单击打开：" & Worksheets(i).Name
  Next
End Sub
```

step 3　单击过程中的任意位置，然后按<F5>键执行过程，程序会在工作簿中创建一个新工作表，并命名为“工作表目录”，在该工作表的 A、B 列创建所有工作表的目录，效果如图 7.25 所示。

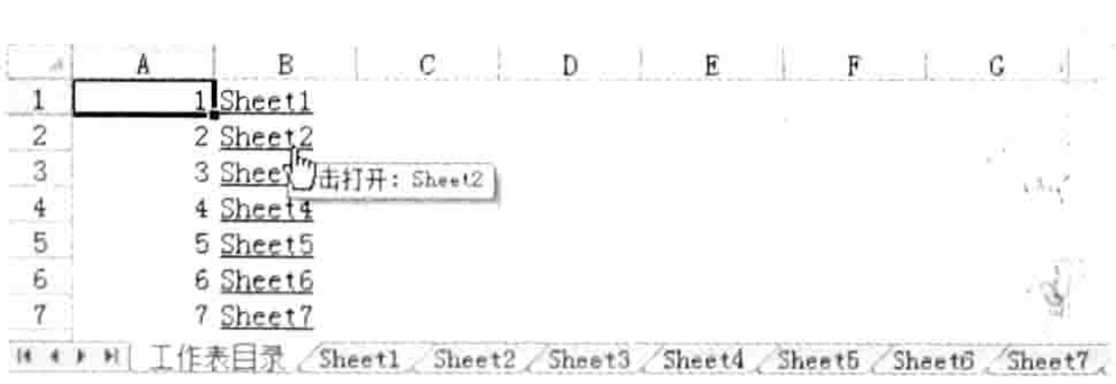


图 7.25　工作表目录

step 4　单击 B2 单元格，由于 B2 单元格的值是 Sheet2，因此会跳转到 Sheet2 工作表中。

思路分析：

创建工作表目录时，为了不破坏当前数据需要新建一个工作表，并命名为“工作表目录”，然

后将目录创建在该工作表中。但是为了程序的通用性，确保工作簿中已有名为“工作表目录”的工作表时执行当前代码仍然不出错，应首先通过代码检查工作簿中是否存在“工作表目录”，如果有则直接创建目录，如果没有则先新建工作表，然后创建目录。

本例的办法是使用 For Next 循环语句遍历活动工作簿中的所有工作表，逐一判断每个工作表的名称是否等于“工作表目录”，如果没有则通过 Worksheets.Add 方法创建工作表，且命名为“工作表目录”；如果有则通过 Goto 语句跳转到指定标签处，从而忽略新建工作表且命名的操作。

在创建工作表目录时，本例使用的是 Hyperlinks.Add 方法加 For Next 循环，它们搭配使用既可在单元格中写入所有工作表的名称又能实现单击单元格时打开对应的工作表。

语法补充：

（1）Hyperlinks.Add 方法用于对图形对象或者 Range 对象创建超级链接，它的语法如下：

```
Hyperlinks.Add(Anchor, Address, SubAddress, ScreenTip, TextToDisplay)
```

Hyperlinks.Add 方法的每个参数的功能介绍见表 7-8。

表 7-8　Hyperlinks.Add方法之参数列表

参数名称	必选/可选	数据类型	说明
Anchor	必选	Object	超链接的位置。可为 Range 或 Shape 对象
Address	必选	String	超链接的地址
SubAddress	可选	Variant	超链接的子地址
ScreenTip	可选	Variant	当鼠标指针停留在超链接上时所显示的屏幕提示
TextToDisplay	可选	Variant	要显示的超链接的文本

超链接的链接对象是文件或者网址时，应将文件或者网页的地址赋值给 Address 参数，同时忽略 SubAddress 参数；如果链接对象是单元格，则应对 Address 参数赋值为空文本，将单元格地址赋值给 SubAddress 参数。

（2）Range.Clear 方法用于清除单元格的值和格式信息，和它相近的还有 Range.ClearComments（清除批注）、Range.ClearContents（清除内容）、Range.ClearFormats（清除格式）和 Range.Delete（删除单元格）。

本例文件参见光盘：..\第七章\7-18 利用 For Next 循环语句创建工作表目录.xlsm

7.3.2 For Each Next 语句

For Each Next 循环语句是针对一个数组或集合中的每个元素重复执行一组语句。

For Each Next 循环与 For Next 循环在语法上极其相似，在功能上也极其相似，而且绝大多数时候可以用 For Next 语句来完成 For Each Next 的工作，但在处理对象集合时 For Each Next 循环具有比较多的优势。

在实际工作中，For Each Next 循环语句主要用于遍历对象集合。

1. 语法详解

For Each Next 语句的语法如下：

```
For Each element In Group
    [statements]
  [Exit For]
```

```
    [statements]
Next [element]
```

For Each Next 循环语句主要包括四部分，各部分详细说明如表 7-9 所示。

表 7-9　For Each Next语法详解

参数名称	功能描述
element	必要参数，用来遍历集合或数组中所有元素的变量。对于集合来说，element 可能是一个 Variant 变量、一个通用对象变量，对于数组而言，element 只能是一个 Variant 变量
group	必要参数，对象集合或数组的名称（用户定义类型的数组除外）
statements	可选参数，针对 group 中的每一项执行的一条或多条语句
Exit For	可选参数，表示中途退出循环，通常配合条件语句使用

在 For Next 循环中可以自由设定循环的范围，而 For Each Next 循环则无法设定范围，而是由对象的数量来决定的。例如在 WorkSheets 集合中循环，那么范围就是所有工作表。

当 group 是对象时，参数 element 可以是对象变量也可以是变体型变量，但是声明为对象变量对于编写代码而言更有利，它可以产生属性与方法列表；当 group 是数组或者集合时，参数 element 必须用变体型变量。

例如在 WorkSheets 对象集合中循环时，变量 element 应该声明为 Worksheet 型。

如果在图形对象集合 Shapes 中循环时，变量 element 应该声明为 Shape 型。

当把握不准对象类型时也可以采用 Object 或者 Variant 型。

For Next 循环语句有一个代表计数器的变量，而 For Each Next 循环语句没有计数器，当需要用到计数器时需要额外声明一个变量，并在循环体中累加变量的值。

2. 案例：使用 For Each Next 语句创建工作表目录

案例要求：利用 For Each Next 循环创建工作表目录。

知识要点：For Each Next 循环、WorkSheets.Add 方法、Hyperlinks.Add 方法。

实现步骤：

step 1　按<Alt+F11>组合键进入 VBE 窗口，然后单击菜单中的“插入”→“模块”命令。

step 2　在模块中录入以下代码：

```
Sub 建立目录()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Dim Sht As Worksheet, i As Integer
  For Each Sht In Sheets
    If Sht.Name = "工作表目录" Then GoTo Mulu
  Next
  Worksheets.Add Worksheets(1)
  ActiveSheet.Name = "工作表目录"
Mulu:
  Worksheets("工作表目录").Range("A:B").Clear
  For Each Sht In Worksheets
    If Sht.Name <> "工作表目录" Then
      i = i + 1
      Worksheets("工作表目录").Cells(i, 1).Value = i
      Worksheets("工作表目录").Hyperlinks.Add Anchor:=Worksheets("工作表目录").
Cells(i, 2), Address:="", SubAddress:="'" & Sht.Name & "'!A1", TextToDisplay:=
Sht.Name, ScreenTip:="单击打开：" & Sht.Name
```

```
    End If
  Next
End Sub
```

step 3 单击过程中的任意位置，然后按<F5>键执行过程，程序会在工作簿中创建一个新工作表，并命名为“工作表目录”，在该工作表的 A 列和 B 列创建所有工作表的目录，效果如图 7.26 所示。

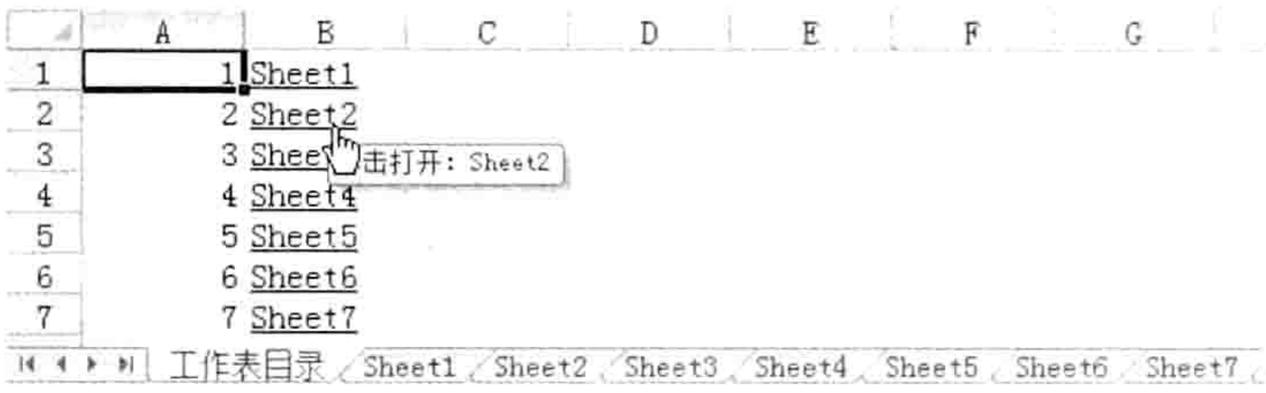


图 7.26 工作表目录

思路分析：

本例代码是基于“7-18 利用 For Next 循环语句创建工作表目录.xlsm”的代码修改而来的，由于 For Each Next 循环不使用计数器和步长值，而是通过对象变量去逐一访问对象集合中的每一个对象，因此变量 sht 即代表每一个需要创建目录的工作表，不再使用“Worksheets(i)”或者“sheets(i)”。

由于 For Each Next 循环没有计数器，而将工作表名称写入单元格中时需要计数器，因此将 For Next 循环改为 For Each Next 循环时需要多使用一个变量。

For Next 循环可以自由指定范围，而 For Each Next 循环必须遍历对象集合中的一切子集。本例中创建工作表目录时需要排除名为“工作表目录”的工作表，因此需要在循环语句中使用 If Then 语句加以限制。

本例文件参见光盘：..\第七章\7-19 利用 For Each Next 循环语句创建工作表目录.xlsm

3. 案例：利用循环选择区域中所有负数

案例要求：选中 B 列和 E 列中的所有负数。

知识要点：For Each Next 循环、Intersect 方法、Union 方法、Range.Select 方法。

实现步骤：

step 1 按<Alt+F11>组合键进入 VBE 窗口，然后单击菜单中的“插入”→“模块”命令。

step 2 在模块中录入以下代码：

```
Sub 选择进出库记录中的负数()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Dim rng As Range, TargetRng As Range
  If Intersect(ActiveSheet.UsedRange, Union(Range("b:b"), Range("e:e"))) Is
Nothing Then Exit Sub
  For Each rng In Intersect(ActiveSheet.UsedRange, Union(Range("b:b"),
Range("e:e")))
    If rng < 0 Then
      If TargetRng Is Nothing Then
        Set TargetRng = rng
      Else
        Set TargetRng = Union(rng, TargetRng)
      End If
    End If
  Next rng
```

```
If TargetRng Is Nothing Then
    MsgBox "没有小于 0 的单元格"
    Exit Sub
  Else
    TargetRng.Select
    TargetRng.Interior.ColorIndex = 3
  End If
End Sub
```

step 3　按<Alt+F11>组合键返回工作表界面，然后按<Alt+F8>组合键打开“宏”对话框。

step 4　从“宏”列表中选择“选择进出库记录中的负数”然后单击“执行”按钮，如果活动工作表的 B 列和 E 列中存在负数，那么程序会瞬间选择所有负数所在单元格，效果如图 7.27 所示。

	A	B	C	D	E
1	A仓库			B仓库	
2	时间	进出库		时间	进出库
3	8:00	450		9:20	-345
4	9:00	744		9:40	740
5	10:00	-345		10:00	-250
6	11:20	260		10:20	450
7	11:45	-450		10:45	744
8	13:50	740		13:50	100
9	14:00	-250		14:40	-480
10	15:15	100		15:00	260
11	16:52	-480		16:20	-150
12	17:45	20			

图 7.27　选中出库中的负数所在单元格

思路分析：

A 组和 B 组的进出库数据存放在 B 列和 E 列，为了提升代码的执行效率，本例首先利用 Intersect 方法提取 B 列、E 列与已用区域的交集，如果不存在交集那么直接结束过程，如果存在交集则通过 For Each Next 循环语句遍历交集。此思路比遍历已用区域的效率会高几倍，它排除了 A 列、C 列、D 列的数据区域。

虽然直接将 B2:B12 和 E2:E12 作为操作对象也可以选中当前数据中的所有负数，不过代码的通用性将大打折扣，当新增数据时必须修改代码，否则会遗漏新增的单元格。

当启动循环语句之后，由于变量 rng 代表操作对象中的每一个单元格，因此直接用“rng < 0”作为判断条件，如果符合条件则将 rng 代表的区域合并到 TargetRng 对象中去。当循环语句执行完成后，TargetRng 即代表了所有负数所在单元格，使用“TargetRng.Select”即可选中所有负数。

在使用 Union 方法合并多个区域时，其每一个参数必须是单元格（即 Range 对象），其中任何一个参数是变体型变量或者 Nothing 则无法合并成功。本例中的变量 TargetRng 在初始化之前正是 Nothing，因此需要使用条件语句判断变量是否初始化，如果没有初始化就使用 Set 语句赋值，使其由 Nothing 变成 Range 对象，其后再次发现负数所在单元格时就可以直接使用代码 Union(rng, TargetRng)将 TargetRng 与 Rng 合并了。

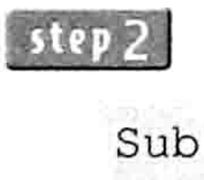
语法补充：

（1）Intersect 方法用于提取多个 Range 对象的交集，本例中 ActiveSheet.UsedRange 与 Range("b:b")有交集，与 Range("e:e")也有交集，但是 ActiveSheet.UsedRange 与 Range("b:b")、Range("e:e")不存在交集，也就是三者作为三个对象是没有重叠部分的。但是将 Range("b:b")与 Range("e:e")当作一个对象，它与 ActiveSheet.UsedRange 之间就产生交集了，因此本例中先用 Union 方法合并两个区域为单个 Range 对象，再用 Intersect 方法取它与 ActiveSheet.UsedRange 的交集。

（2）Range.Select 方法表示选择单元格或者区域，如果表达式中 Range 部分使用的是对象变量而且它尚未初始化时，执行 Range.Select 方法必定失败，因此需要先用条件语句判断，未初始

化时就提示用户并且结束过程，已经初始化就执行 Range.Select 方法选择区域。

（3）Range.Interior.ColorIndex 表示单元格的内部颜色，其中 Interior 代表内部，ColorIndex 代表颜色编码，取值范围是 0～56。本例将它赋值为 3 表示红色。



本例文件参见光盘：..\第七章\7-20 选择 B 和 E 列所有负数所在单元格.xlsm

4. 案例：利用循环统一所有图片的高度并对齐单元格

案例要求：在工作表中批量插入图形对象时，其大小与左边距、上边距不会与所在单元格自动统一，Excel 也没有提供批量统一图片尺寸与边距的工具。现要求通过 VBA 让图片高度及边距与图片所在单元格统一。

知识要点：For Each Next 循环、Shapes 对象集合、Shape.TopLeftCell 属性、shape.Top 属性、shape.Left 属性、Range.Top 属性、Range.Left 属性。

实现步骤：

step 1 假设工作表中有如图 7.28 所示数据与图片，按<Alt+F11>组合键进入 VBE 窗口，然后单击菜单中的“插入”→“模块”命令。

step 2 在模块中录入以下代码：

```
Sub 统一高度与左边距()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Dim shp As Shape
  For Each shp In ActiveSheet.Shapes
    shp.Height = shp.TopLeftCell.Height
    shp.Left = shp.TopLeftCell.Left
    shp.Top = shp.TopLeftCell.Top
  Next shp
End Sub
```

step 3 单击过程的任意位置，并按<F5>键执行程序，工作表中所有图形对象将会瞬间自整调整位置和大小。调整后的效果如图 7.29 所示。

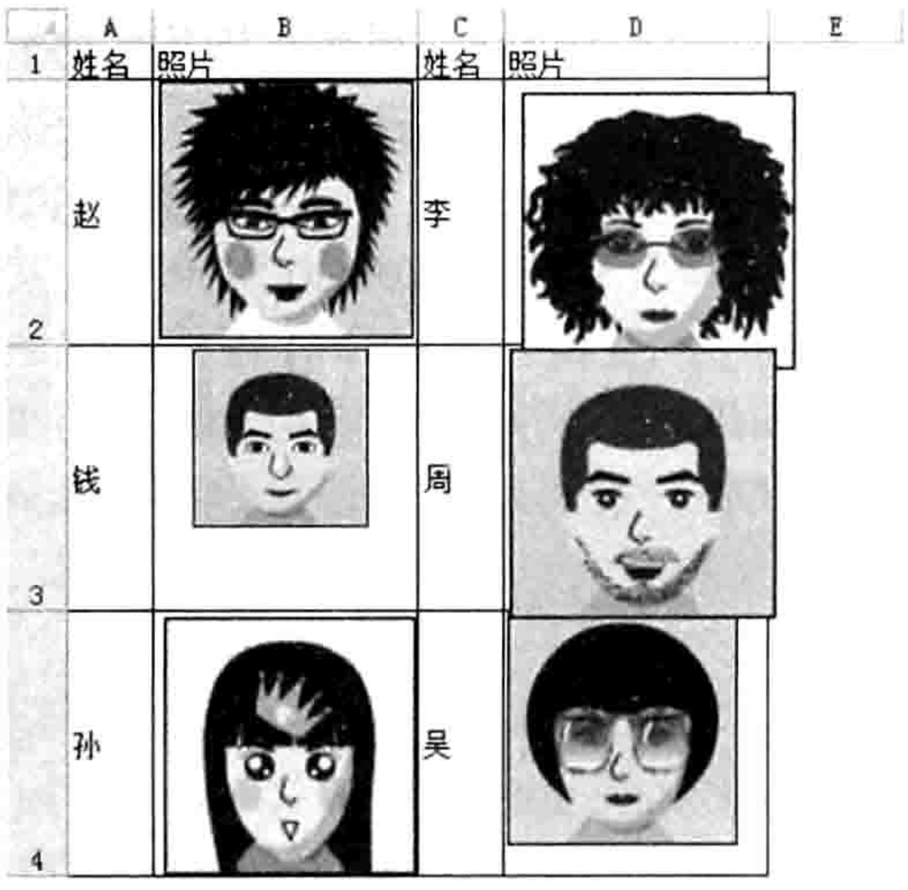


图 7.28　混乱的图形对象

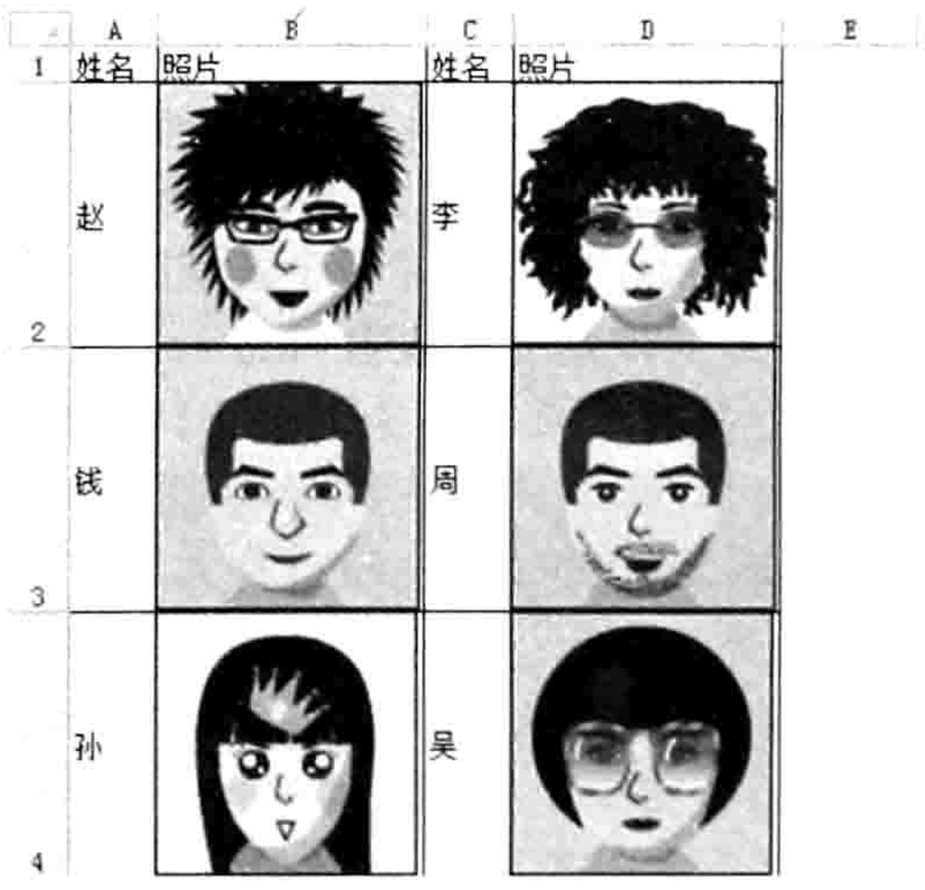


图 7.29　利用循环对齐图片并统一高度

思路分析：

本例中需要遍历的对象是所有图形对象 Shapes，因此用于循环语句的变量 shp 需要声明为 Shape 对象类型。尽管将它声明为 Object 或者变体型也不影响程序的功能，但是在书写“shp.”时不会产生属性与方法列表，不利于快速编写代码。

当程序进入循环语句之后，变量 shp 代表 shapes 集合中的每一个子集，修改 shp 的高度（shape.Height 属性）、左边距（Shape.Left 属性）和上边距（Shape.Top 属性）即等于统一所有图形对象的高度与边距。

如果工作表中除了图片以外还有图表、艺术字、文本框，则还可以修改代码限制图形对象的类型，使代码仅仅调整图片的高度与边距，代码如下：

```
Sub 统一高度与左边距 2()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Dim shp As Shape
  For Each shp In ActiveSheet.Shapes
    If shp.Type = msoPicture Then
      shp.Height = shp.TopLeftCell.Height
      shp.Left = shp.TopLeftCell.Left
      shp.Top = shp.TopLeftCell.Top
    End If
  Next shp
End Sub
```

语法补充：

（1）Shape.Height 属性代表图形对象的高度，以磅为单位。Range.Height 属性表示单元格的高度，其单位也是磅，因此可以将单元格的 Height 属性赋值给 Shape 对象的 Height 属性，从而统一两者的高度。

（2）Shape.Left 属性代表图形对象的左边距，即图形对象与 A 列左边框的距离。它的参照位置由 A 列决定，而不是屏幕的左边框，与单元格的边距算法一致。

（3）Shape.TopLeftCell 属性可以返回图形对象左上角下方的单元格，换而言之它是一个 Range 对象。与 Shape.TopLeftCell 相对的还有 Shape.BottomRightCell 属性。

（4）Shape.Type 属性代表图形对象的类型，可以通过此属性值区分开图片、文本框、艺术字、图表等图形对象。Shape.Type 属性的取值范围由表 7-10 中的 MsoShapeType 常数决定。

表 7-10　MsoShapeType常数列表

名称	值	描述
msoShapeTypeMixed	–2	混合形状类型
msoAutoShape	1	自选图形
msoCallout	2	标注
msoChart	3	图
msoComment	4	批注
msoFreeform	5	任意多边形
msoGroup	6	组合
msoEmbeddedOLEObject	7	嵌入的 OLE 对象
msoFormControl	8	窗体控件
msoLine	9	线条
msoLinkedOLEObject	10	链接 OLE 对象
msoLinkedPicture	11	链接图片
msoOLEControlObject	12	OLE 控件对象
msoPicture	13	图片
msoPlaceholder	14	占位符

续表

名称	值	描述
msoTextEffect	15	文本效果
msoMedia	16	媒体
msoTextBox	17	文本框
msoScriptAnchor	18	脚本定位标记
msoTable	19	表
msoCanvas	20	画布
msoDiagram	21	图表
msoInk	22	墨迹
msoInkComment	23	墨迹批注
msoIgxGraphic	24	IGX 图形

本例文件参见光盘：..\第七章\7-21 统一图片高度及边距.xlsm

7.3.3 Do Loop 语法详解

Do Loop 是一种循环语句，表示当条件成立或者不成立时重复执行一组命令，也可以是首先执行一组命令，直到条件成立或者不成立时停止循环。

Do Loop 循环语句没有明确的循环次数，只要符合指定条件或者不符合指定件就会一直循环下去，直到用户手工中断程序或者关闭 Excel 程序。

1. 语法详解

Do Loop 循环总共包含五种书写形式，它们的书写方式和功能都大同小异。

形式一：一直循环执行代码。语法如下：

```
Do
    [statements]
[Exit Do]
    [statements]
Loop
```

其中 statements 代表需要执行的一句或者多句代码，Exit Do 代表结束循环，通常配合条件语句 If Then 使用，表示符合某条件时结束循环。

形式二：只要符合某条件，那么一直循环执行代码。语法如下：

```
Do While condition
    [statements]
[Exit Do]
    [statements]
Loop
```

其中 condition 代表条件，While condition 代表只要符合条件就一直循环下去。形式二的 Do Loop 循环也支持中途通过 Exit Do 结束循环。

形式三：只要不符合条件，那么一直循环执行代码。语法如下：

```
Do Until condition
    [statements]
```

```
[Exit Do]
    [statements]
Loop
```

Until condition 表示不符合条件时就一直循环下去，允许中途结束循环。

形式四：一直循环执行代码，直到符合条件时停止循环。语法如下：

```
Do
    [statements]
[Exit Do]
    [statements]
LoopWhile condition
```

形式五：一直循环执行代码，直到不符合条件时停止循环，其语法如下：

```
Do
    [statements]
[Exit Do]
    [statements]
Loop Until condition
```

五种循环方式中最常用的是第一种，只要熟练掌握第一种用法就能实现其他四种的功能。

2. 案例：判断哪一日产量正常

案例要求：图 7.30 的产量表中包含日期和产量，公司要求日产量高于 800 才算合格。请计算第一次产量合格是哪一天。

知识要点：Do Loop 循环语句。

实现步骤：

step 1　按<Alt+F11>组合键进入 VBE 窗口，然后单击菜单中的“插入”→“模块”命令。

step 2　在模块中录入以下代码：

```
Sub 哪一日产量正常()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Dim i As Byte
  i = 2
  Do
    If Cells(i, 2) > 800 Then Exit Do
    i = i + 1
  Loop
  MsgBox Cells(i, 1) & "达到目标产量", vbInformation + vbOKOnly
End Sub
```

step 3　单击过程的任意位置，并按<F5>键执行程序，程序会弹出如图 7.31 所示的结果。

	A	B
1	日期	产量
2	5月2日	764
3	5月3日	748
4	5月4日	752
5	5月5日	726
6	5月7日	727
7	5月8日	770
8	5月9日	805
9	5月10日	711
10	5月11日	762
11	5月12日	821

图 7.30　产量表

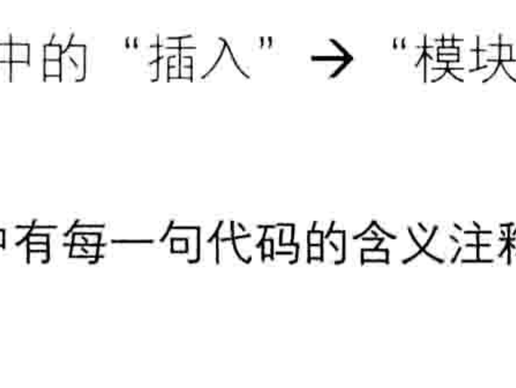


图 7.31　报告产量首次达标的日期

思路分析：

Do Loop 循环表示循环执行一组代码，遇到 Exit Do 时停止循环。在本例中循环执行的语句是

“i = i + 1”，表示让计数器在 2 的基础上逐一累加，直到条件“Cells(i, 2) > 800”成立时停止循环。最后根据计数器 i 的值引用对应的单元格的日期作为最终结果。

事实上，改用 Do While Loop 或者 Do Until Loop 形式的循环语句可以实现同等效果。以下提供另外两种形式的循环语句，加深读者对 Do While Loop 或者 Do Until Loop 循环语句的了解。

```
Sub 第几日产量正常 2()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Dim i As Byte
  i = 2
  Do While Cells(i, 2) <= 800
    i = i + 1
  Loop
  MsgBox Cells(i, 1) & "达到目标产量", vbInformation + vbOKOnly
End Sub
```

相对于 Do Loop 循环，Do While Loop 循环相当于将结束循环的条件放到了 Do 语句所在行，因此可以忽略“Exit Do”。

```
Sub 第几日产量正常 3()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Dim i As Byte
  i = 2
  Do Until Cells(i, 2) > 800
    i = i + 1
  Loop
  MsgBox Cells(i, 1) & "达到目标产量", vbInformation + vbOKOnly
End Sub
```

Do Until Loop 循环的功能是不符合条件时就一直循环下去，因此相对于 Do While Loop 循环，它的条件必须相反，由小于等于 800 改为大于 800。

本例文件参见光盘：..\第七章\ 7-22 计算第几日产量正常.xlsm

3. 案例：按格式查找

案例要求：如图 7.32 所示的是成绩表，其中部分单元格的字体是宋体、部分单元格的字体是 Impact，还有部分单元格的背景颜色是红色。现要求利用代码选中字体名称为 Impact 并且背景是红色的所有单元格。

	A	B	C	D	E	F	G	H
1	姓名	语文	数学	化学	政治	计算机	物理	法律
2	朱真光	73	75	80		66	85	98
3	柳红英	51	71	89	92	65	55	
4	曹值军		50	68	88		56	72
5	钟正国	79		60	81	97	56	57
6	陈览月	68	63	84	50	78	85	78
7	陈真亮	56	59	57	75		52	97
8	梁兴	58	92	99	75	87	76	
9	简文明	73	99	84	53	92	70	56
10	周语怀		78		55	79	65	72
11	柳龙云	57	95	72	54	59		66

图 7.32　成绩表

知识要点：Do Loop 循环语句、FindFormat.Clear 方法、Range.Find 方法、。

实现步骤：

step 1　按<Alt+F11>组合键进入 VBE 窗口，然后单击菜单中的“插入”→“模块”命令。

step 2　在模块中录入以下代码：

```
Sub 按格式查找()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
```

```
Dim Rng As Range, TargetRng As Range, FirstAddRess As String
Application.FindFormat.Clear
With Application.FindFormat
  .Interior.ColorIndex = 3
  .Font.Name = "Impact"
End With
With ActiveSheet.UsedRange
  Set Rng = .Find(what:="", SearchFormat:=True)
  If Rng Is Nothing Then MsgBox "没有找到此类单元格": Exit Sub
  Set TargetRng = Rng
  FirstAddRess = Rng.Address
  Do
    Set Rng = .Find(what:="", after:=Rng, SearchFormat:=True)
    If FirstAddRess = Rng.Address Then Exit Do
    Set TargetRng = Union(TargetRng, Rng)
  Loop
  TargetRng.Select
 End With
End Sub
```

step 3　单击过程的任意位置，并按<F5>键执行程序，程序会瞬间选中工作表中所有字体名称为 Impact 而且背景是红色的单元格。

思路分析：

Excel 提供了按格式查找的功能，不过只能录制清除查找格式、设置查找格式和查找第一个符合条件的单元格这三类操作，不能将查找全部也录制下来。基于此，必须在录制宏的基础上，将 Do Loop 循环语句写入宏代码中，配合 Range.Find 方法实现批量查找。

本例代码首先使用“Application.FindFormat.Clear”方法清除以前的查找格式，避免它影响本次查找结果。然后通过 FindFormat 对象设置本次查找的格式，包括“Interior.ColorIndex”——单元格背景色和“Font.Name”——字体名称。

设置好查找格式后，使用 Range.Find 方法查找第一个符合条件的单元格，并将查找结果赋值给变量 Rng。由于是按格式查找，因此 What 参数赋值为空文本，而 SearchFormat 参数则赋值为 True。

如果 Range.Find 方法查找成功可以取得一个单元格对象，如果查找不成功则返回 Nothing。当 Range.Find 方法的返回值是 Nothing 时必须终止过程，不必执行其他语句，因此本例中将条件语句与 Exit Sub 语句搭配使用，根据查找结果决定是否允许程序继续执行。

当 Range.Find 方法的返回值是单元格时，应首先记录它的地址，然后在 Do Loop 循环语句中以该地址作为结束循环的条件，否则 Do Loop 循环与 Range.Find 方法搭配使用会永远查找下去。

在 Do Loop 循环的查找过程中，每一次查找只能返回一个符合条件的单元格，由于可能有多个单元格符合条件，因此应使用 Union 方法将所有符合条件的目标单元格合并为单个 Range 对象，当循环结束后才能使用 TargetRng.Select 语句选中这些符合条件的所有单元格。

在第一次使用 Range.Find 方法执行查找时可以不对 After 参数赋值，其他每一次查找都必须以上一次的目标单元格作为 After 参数的值，其用意是在上一次找到的目标之后开始查找，则只能找到一个符合条件的单元格。

语法补充：

（1）FindFormat.Clear 方法表示清除以往所设置的查找格式。Excel 的查找对话框具有记忆功能，每次设置的查找条件都会自动保存下来，如果不清除将会影响本次的查找结果。假设上一次查找的条件是红色字体，本次所设置的条件是字号为 12 号，那么查找条件是红色字体、字号为 12 号的单元格。

（2）Range.Find 可以按值查找也可以按格式查找，这取决于它的参数赋值情况。Range.Find 方法的语法如下：

```
表达式.Find(What, After, LookIn, LookAt, SearchOrder, SearchDirection, MatchCase,
MatchByte, SearchFormat)
```

它有 9 个参数，每个参数的含义如表 7-11 所示。

表 7-11　Range.Find方法的参数一览

参数名称	功能描述
What	要搜索的数据，可以是字符串或任意 Excel 数据类型
After	此参数是一个单元格，Range.Find 方法将从该单元格之后开始搜索。此单元格对应于从用户界面搜索时的活动单元格的位置
LookIn	赋值为 xlComments 时表示在批注中查找，赋值为 xlFormulas 时表示在公式中查找，赋值为 xlValues 时表示在值中查找，默认是第 2 项。
LookAt	赋值为 xlWhole 时表示完全匹配，赋值为 xlPart 时表示部分匹配，例如“A”可以匹配“AB”
SearchOrder	赋值为 xlByRows 时表示按行查找，赋值为 xlByColumns 时表示按列查找
SearchDirection	搜索的方向。赋值为 xlNext 时表示搜索下一个匹配值，赋值为 xlPrevious 时表示搜索上一个匹配值。当区域中有多个符合条件的值时此参数才有意义
MatchCase	赋值为 True 时表示搜索时区分大小写，否则不区分，默认值为 False
MatchByte	赋值为 True 时表示搜索区分全/半角，否则不区分
SearchFormat	赋值为 True 时表示按格式查找，否则不按格式查找。默认值为 False

在工作表界面按<Ctrl+F>组合键可以打开“查找”对话框，该对话框中有 7 个地方对应 Range.Find 方法的 7 个参数，具体见图 7.33。

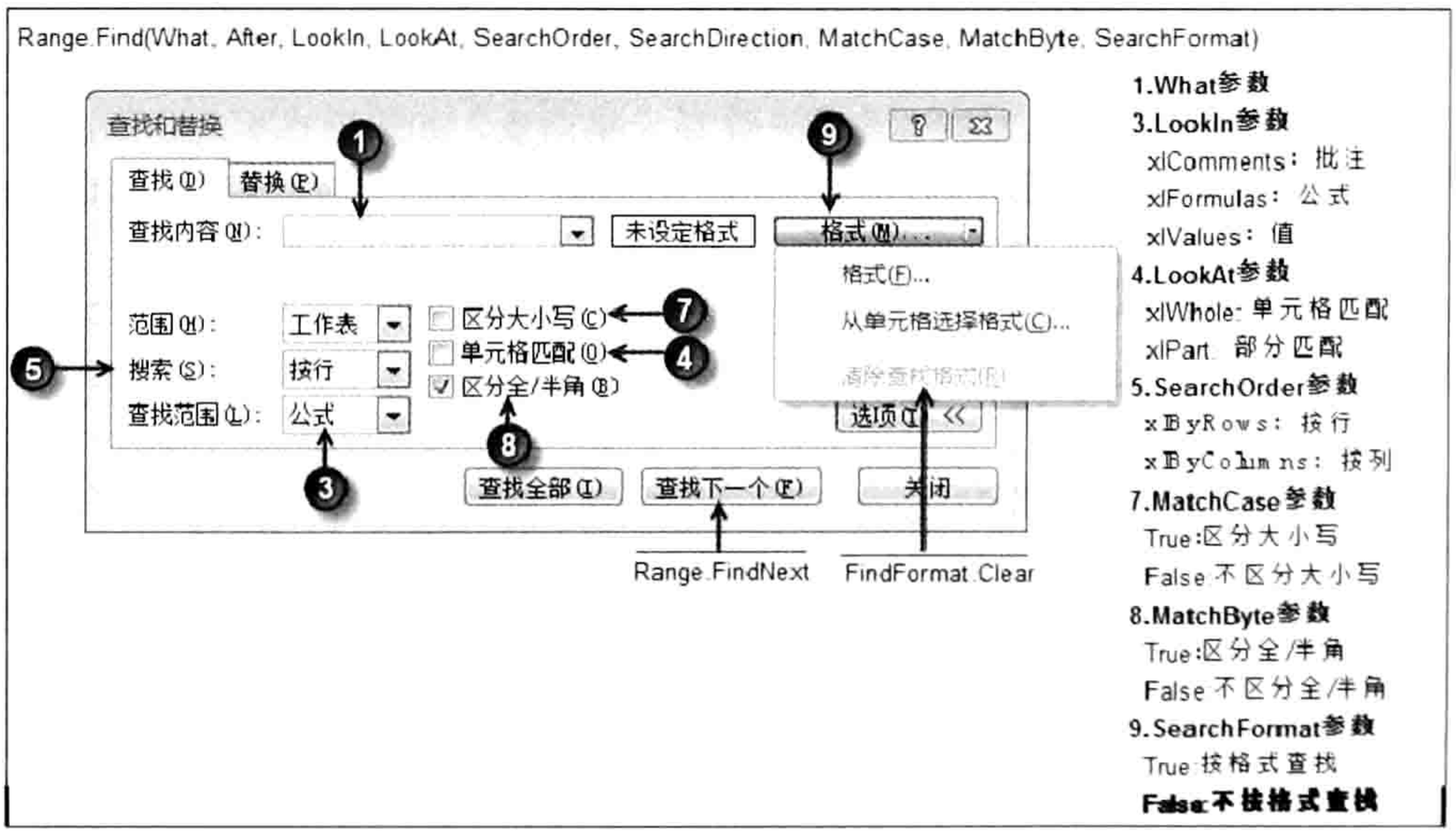


图 7.33　“查找”对话框与 Range.Find 方法的对应关系

Range.Find 方法配合 Do Loop 循环查找目标时会周而复始查找，例如在图 7.34 中查找字母 A，它的查找规则是没有指定 After 参数时默认从左上角第一个单元格之后开始查找，因此第一次找到

的单元格是 B2，第二次找到的单元格是 C3，第三次找到的单元格是 A1，第四次找到的单元格又是 B2。

为了确保程序找完一轮后自动停止，通用的办法是首次查找时记录下它的地址，然后每查找一次都与它做比较，如果比较结果为 True 则结束循环。

	A	B	C	D
1	A	77	79	
2	64	A	89	
3	50	88	A	
4				

图 7.34　待查找的数据

本例文件参见光盘：..\第七章\7-23 按格式查找.xlsm

4. 案例：计算得分累加到 1000 时的月份

案例要求：某球星参加过 100 场篮球比赛，在工作表中罗列了 100 场比赛的得分，如图 7.35 所示。现需要计算他在哪一场的累计得分达到 1000 分。

	A	B	C
1	场次	得分	
2	第1场	16	
3	第2场	18	
4	第3场	30	
5	第4场	16	
6	第5场	29	
7	第6场	38	
8	第7场	19	
9	第8场	28	
10	第9场	23	
11	第10场	28	

篮球比赛得分统计

图 7.35　比赛得分统计表

知识要点：Do Loop 循环语句、Exit Sub 语句、If Then Else 语句。

实现步骤：

step 1　按<Alt+F11>组合键进入 VBE 窗口，然后单击菜单中的“插入”→“模块”命令。

step 2　在模块中录入以下代码：

```
Sub 得分累积到1000的场次()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Dim SumValue As Integer, i As Integer
  i = 2
  Do
    SumValue = SumValue + Cells(i, 2)
    If SumValue >= 1000 Then
      MsgBox "第" & i - 1 & "场达到1000分", vbInformation, "提示"
      Exit Sub
    Else
      i = i + 1
    End If
  Loop
End Sub
```

step 3　将光标定位于过程并按<F5>键执行程序，程序会弹出累加得分达到 1000 的比赛场次。

思路分析：

由于得分累加到 1000 后就会终止过程，因此用于存放累计得分的变量 SumValue 应声明为 Integer 型，如果改为 Byte 型会产生溢出错误，如果改用 Long 型会耗费更多的资源。

在循环语句中，计数器 i 的值会逐一累加下去，Cells(i, 2)所代表的单元格也会不断地变化。变量 SumValue 的初始值是 0，将它与 Cells(i, 2)相加后可以得到每一场的得分累计值。

在循环语句中，本例设置的终止循环的条件是 SumValue 的值大于等于 1000。终止循环的语句是 Exit Sub 而不是 Exit Do，两者的功能原本不同，不过在本例中是一样的。Exit Do 只能终止循环，而 Exit Sub 既可以终止循环又可以终止过程。

CD-ROM 本例文件参见光盘：..\第七章\ 7-24 计算得分累积到 1000 的场次.xlsm

5. 案例：利用循环产生文字动画

案例要求：在 Flash 或者网页中可以实现让文字滚动，有一种炫目的感觉。是否可以利用 VBA 实现同等功能呢?

知识要点：Do Loop 循环语句、DoEvents 函数、If Then 语句。

实现步骤：

step 1 按<Alt+F11>组合键进入 VBE 窗口，然后单击菜单中的“插入”→“模块”命令。

step 2 在模块中录入以下代码：

```
Dim 停 As Boolean
Sub 字符滚动()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Dim j As Integer
  停 = False
  Range("a1") = "四维实业公司人事报表  "
  Do
    For j = 1 To 3000
      DoEvents
    Next j
    Range("a1") = Mid(Range("a1"), 2, Len(Range("a1")) - 1) & Left(Range("a1"), 1)
    If 停 = True Then Exit Do
  Loop
  Range("a1").ClearContents
End Sub
Sub 停止滚动()
  停 = True
End Sub
```

step 3 按<Alt+F11>组合键返回工作表界面，单击功能区中的“开发工具”→“插入”→“按钮（表单控件中的按钮）”命令。

step 4 在工作表中按住鼠标左键向右下拖动，从而绘制出一个按钮。

step 5 当 Excel 弹出如图 7.36 所示的“指定宏”对话框时，在宏名列表中选择“字符滚动”，然后单击“确定”按钮关闭窗口。

step 6 将按钮的标题由“按钮 1”修改为“开始”。

step 7 重复步骤 3、4、5，在工作表中添加第二个按钮，将其关联到过程“停止滚动”，然后再将它的标题由“按钮 2”改为“停止”。

step 8 单击“开始”按钮，A1 单元格中将产生“四维实业公司人事报表”，而且字符串会向左滚动，直到单击“停止”按钮时停止滚动。如图 7.37 所示即为滚动过程中的一个截图。

图 7.36　为按钮指定宏

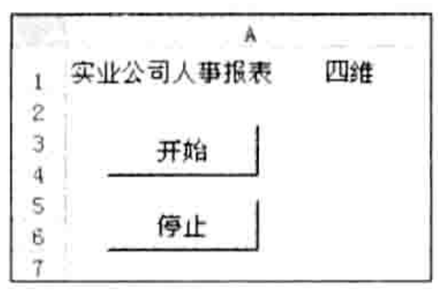


图 7.37　用按钮控件控制字符串滚动

思路分析：

本例中两个 Sub 过程都会为变量“停”赋值，因此变量必须声明为公共变量。

在“字符滚动”过程中，首先为变量“停”和 A1 单元格赋予初始值，然后在 Do Loop 循环语句中移动 A1 单元格的首字符的位置，在快速并且反复移动字符位置时，给人的感受就是字符在滚动。

由于在循环语句中，代码“Range("a1") = Mid(Range("a1"), 2, Len(Range("a1")) - 1) & Left(Range("a1"), 1)”的执行速度相当快，以致用户会完全感觉不到字符有滚动效果。为了解决这个问题，本例使用 DoEvents 函数来拖延时间，让用户有足够的时间看清楚滚动的字符。由于 DoEvents 函数执行一次所延迟的时间相当短，因此将它放在循环语句，反复执行 3000 次。读者可以将“3000”修改为更大或者更小的数值，然后感觉字符的滚动速度变化。

Do Loop 循环语句在执行期间会耗用大量的内存，为了方便随时停止字符滚动，所以在工作表中插入了命令按钮，单击按钮时将变量“停”赋值为 True，而过程“字符滚动”中的条件语句检测到变量的值为 True 后会自动结束循环。

语法补充：

（1）DoEvents 函数的功能是转移控制权，在实际工作中更多的是通过它来实现延时效果。

（2）Mid 函数用于从字符串中的指定位置提取指定长度的字符。它的语法如下：

```
Mid(string, start[, length])
```

其中 string 参数是一个字符串表达式，start 参数代表取值的起始位置，length 参数代表取值的长度。其整体的含义是从 string 字符串的第 start 位开始提取 length 位字符。length 参数是可选参数，它的默认值是 string 的长度。

例如代码“mid(12345,2)”相当于“mid(12345,2,len(12345))”，其返回值为 2345。

（3）Len 函数用于计算字符串的长度，亦称字符数量。例如代码“Len(12)”等于 2，代码“len("你好")”也等于 2。

（4）Left 函数用于提取字符串左边若干位字符，其语法如下：

```
Left(string, length)
```

第 1 参数代表字符串，第 2 参数代表长度，两者都是必选参数，与工作表函数 left 不同。

CD-ROM　本例文件参见光盘：..\第七章\ 7-25 利用循环产生文字动画.xlsm

7.4　错误处理语句

任何人在开发程序过程中必定会遇到代码错误，其中包括可以预料的错误、意外的错误，以及开发者故意设置的错误。在执行过程中有必要对这些错误进行处理或者预防，避免程序中断或者无

法达成预期结果。

本节对程序出错的原因、含义、错误捕捉设置，以及防错的方法进行一一解说。

7.4.1 错误类型与原因

VBA 内部对程序出错的原因有近百种解释，即表示错误类型有近百种。但若以大类进行划分，则通常可以分为三类：环境问题、开发者“笔误”和用户错误使用。

1. 环境问题

此处的环境问题是指 Office 应用程序的环境，而这个环境又包括两方面：版本和 Excel 对象。

版本问题是指使用者的 Office 版本与开发者的应用程序版本不同，造成代码不兼容。例如在 Excel 2010 中录制一个排序的宏，将此代码应用到 Excel 2003 中一定会出错。因为 Excel 2010 所用的排序代码只能在 Excel 2007 及以上的版本中运行，不支持 Excel 2003，但是 Excel 2003 中录制的排序宏却可以在 Excel 2010 中正常执行。

对象问题主要是指过程中涉及的对象不存在，读取一个不存在的对象的任意属性都会出错。一方面可能是开发者在代码的容错性上未下足功夫，另一方面也可能是用户使用不当，导致当前的运行环境与预设的环境不一致。例如在空白工作表中执行“调整图片大小”之类过程，由于根本不存在图片，因此操作图片的代码就无法正常执行。

2. 开发者“笔误”

很多大中型插件或者系统都有上万行 VBA 代码，难免会出现拼写失误，包括标点符号的多写或者漏写，以及单词拼写错误等，因此编程过程中测试代码就显得极为重要。

3. 用户错误使用

操作错误包含无意和有意两种。

在工作表保护状态下执行了修改数据的程序，在工作表命名的程序中录入了非法字符，以及在选择图片状态下执行了对选区进行查找、汇总之类的程序等都会导致程序出错，这些属于无意的失误操作。

有意的错误操作主要是基于测试的心态，想看一看程序如何反应：

```
Sub 加解密()
  Dim ans As Byte
  ans = Application.InputBox("输入1：加密" & Chr(10) & "输入2：解密", , , , , , , , 1)
  '…更多代码…
End Sub
```

在以上过程中，因为正常情况下变量 ans 只需在 1～2 这个范围内变化，所以编程时将其数据类型声明为 Byte。而用户可能会故意胡乱地输入 256 以上或者 0 以下的数据，此时程序必定会中断，同时产生“溢出”错误。

基于以上各种原因，程序在使用过程中出错很难避免，那么在代码中进行防错就显得尤为重要。

7.4.2 Err 对象及其属性、方法

VBA 中提供了一个 Err 对象，Err 对象拥有若干个属性和方法，借助这个对象及其属性、方法可以获取代码的错误原因，以及提前防错。

Err 是一个对象，因此在模块中录入“Err.”后可以弹出其属性与方法列表，如图 7.38 所示。

图 7.38　Err 对象的成员列表

Err 的属性参见表 7-12，Err 的方法参见表 7-13。

表 7-12　Err属性详解

属性名称	含义描述
Number	返回或设置表示错误的数值。Number 是 Err 对象的默认属性。可读也可写
HelpContext	返回或设置一个字符串表达式，包含 Windows 帮助文件中的主题的上下文 ID。可读也可写
HelpFile	返回或设置一个字符串表达式，表示帮助文件的完整限定路径。可读也可写
Source	返回或设置一个字符串表达式，指明最初生成错误的对象或应用程序的名称。可读也可写
LastDLLError	返回因调用动态链接库(DLL)而产生的系统错误号，只读。在 Macintosh 中，LastDLLError 总是返回零
Descriptio	返回或设置一个字符串表达式，包含与对象相关联的描述性字符串。可读也可写

表 7-13　Err方法详解

方法名称	含义描述
Raise	产生运行时错误
Clear	清除 Err 对象的所有属性设置

Err 的属性中使用最频繁的是 Err.Number 属性，而方法中使用最频繁的是 Err.Clear 方法。本书后面的章节会有大量的关于 Err.Number 属性和 Err.Clear 方法的应用。

7.4.3　认识 Error 函数

Error 函数用于获取对应于已知错误编码的错误信息或者创建一个错误。例如知道错误编码是 5，那么代码“Msgbox Error(5)”可以返回错误编码 5 对应的信息——无效的过程调用或参数。而代码“Error5”则可以产生一个编码为 5 的错误。

Error 函数的语法如下：

```
Error [(errornumber)]
```

其参数代表错误编码，例如编码 6 表示“溢出”错误。Error 函数的功能是获取错误信息，以及创建一个错误，在实际工作中通过它获取错误的应用更为普遍。

当 Error 有数字参数时，表示获取指定编码的错误信息；当 Error 无参数时表示获取当前过程中的错误信息，如果当前过程没有错误则返回空文本。

Err.Raise 方法也可以产生运行时错误，因此以下两句代码的功能一致：

```
Error 6
Err.Raise 6
```

7.4.4 On Error GoTo line

为了处理错误（包括代码编写有误或者代码正确但使用者使用不当造成的错误），VBA 提供了两个专用的防错语句。在编程过程中可以借助防错语句处理代码中的错误，包括让程序出错时忽略错误、继续执行下一句和程序出错时更改程序执行流程。

与 On Error 相关的语句总共包括三句，其中两句用于处理错误，第三句用于禁止前两句的功能，具体书写方式和含义见表 7-14。

表 7-14 VBA的错误处理语句

语句	含义描述
On Error GoTo line	当程序出错时跳转到 Line 标签处继续执行。 Line 是一个由开发者指定的标签，当过程中任意代码出错时，程序就会激活此错误处理程序，并将程序的执行流程转向 line 处。On Error GoTo line 语句必须与标签 line 在同一个过程中
On Error Resume Next	当运行出错时继续执行下一句，但是同在过程中记录当前的错误编号
On Error GoTo 0	禁止当前过程中任何已启动的错误处理程序

On Error GoTo line 语句表示如果程序在执行时出错就跳转到指定的标签处继续运行，此语句必须配合标签使用。

在两种情况下需要使用 On Error GoTo line，以下通过两个案例分别演示。

1. 重新描述错误提示

案例要求：将当前选择的图片扩大到两倍

知识要点：On Error GoTo Line、ShapeRange.ScaleHeight、ShapeRange.ScaleWidth、Application.Version、If Then 语句。

实现步骤：

step 1 按<Alt+F11>组合键进入 VBE 窗口，然后单击菜单中的“插入”→“模块”命令。

step 2 在模块中录入以下代码：

```
Sub 将选择的图片放大到两倍A()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Selection.ShapeRange.ScaleHeight 2, msoFalse, msoScaleFromTopLeft
  Selection.ShapeRange.ScaleWidth 2, msoFalse, msoScaleFromTopLeft
End Sub
```

以上代码仅用于 Excel 2003，它可以将当前选中的图片的高度和宽度都扩大到两倍。但是此代码在 Excel 2007 或者 Excel 2010 中执行时则会扩大到 4 倍。第一句代码原本的功能是将高度扩大到两倍，但在 Excel 2003 以上的版本中执行代码时它会保持图片的纵横比例，使图片的宽度和高度同时扩大到两倍。如果此时再执行第二句代码，图片的高度和宽度就会扩大到 4 倍。

如果要让代码通用于所有版本，任何时候都只放大到两倍，那么应按以下方式编写：

```
Sub 将选择的图片放大到两倍B()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Selection.ShapeRange.ScaleHeight 2, msoFalse, msoScaleFromTopLeft
  If Application.Version < 12 Then)
    Selection.ShapeRange.ScaleWidth 2, msoFalse, msoScaleFromTopLeft
  End If
End Sub
```

代码的含义是先将高度放大到两倍，然后使用 If Then 语句判断当前 Excel 版本号，如果小于 12 则将宽度也放大到两倍，否则直接结束过程。

step 3　返回工作表界面，选中工作表中已经插入的图片，然后执行过程“将选择的图片放大到两倍 B”，程序会将图片的高度和宽度都放大到当前值的两倍。

step 4　选择任意单元格，然后再次执行过程“将选择的图片放大到两倍 B”，程序会弹出如图 7.39 所示的错误提示。

step 5　很显然，Excel 用户对于如图 7.39 所示的错误提示很难明白是什么原因导致了代码出错。为了让用户看到错误提示就明白如何纠错，应将代码按如下形式修改：

```
Sub 将选择的图片放大到两倍C()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  On Error GoTo ErrLine
  Selection.ShapeRange.ScaleHeight 2, msoFalse, msoScaleFromTopLeft
  If Application.Version < 12 Then
    Selection.ShapeRange.ScaleWidth 2, msoFalse, msoScaleFromTopLeft
  End If
  Exit Sub
ErrLine:
  MsgBox "请选择图片后再执行本过程", vbInformation, "错误提示"
End Sub
```

step 6　选中图片执行以上过程，程序会将图片的高度和宽度都扩大到两倍。如果选中单元格再执行以上过程，程序会弹出如图 7.40 所示的提示信息。

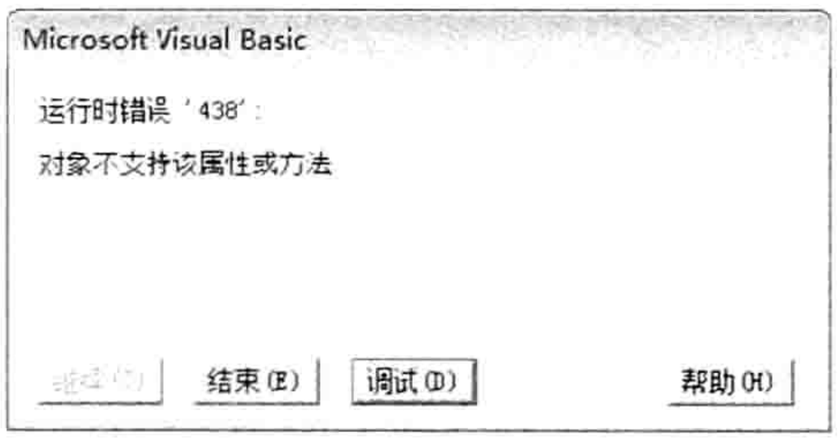


图 7.39　选择单元格执行过程时的错误提示

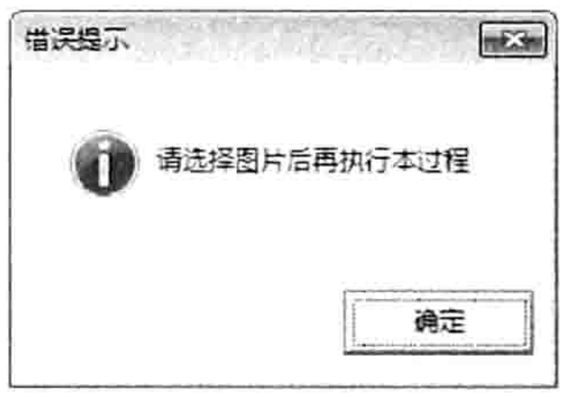


图 7.40　修改后的错误提示

如图 7.40 所示的提示和如图 7.39 所示的提示都是遇到同样错误时产生的，但是显然如图 7.40 所示的提示更人性化，可以根据提示明白错在哪里，如何纠错。

思路分析：

过程“将选择的图片放大到两倍 C”的重点在于“On Error GoTo ErrLine”，它表示程序出错时就执行标签“ErrLine”后面的代码，而“ErrLine”后面的代码可以随意定制，可以使用任何人都能看懂的提示，而非只有程序员才能懂得的“对象不支持该属性或方法”。

在本例中，标签 ErrLine 位于过程末尾，为了避免程序没有错误时也执行标签 ErrLine 后的代码，特在标签之前插入了一句“Exit Sub”。

语法补充：

（1）On Error GoTo Line 语句中的 Line 代表标签，标签名称可以自由定义，允许使用数字开头，也允许使用汉字或者英文，但不能使用标点符号开头。

（2）ShapeRange.ScaleHeight 方法代表调整图形对象的高度，由于工作表中插入的图片默认是锁定纵横比的，因此在 Excel 2003 以上的任意版本中都会同时修改图片的高度与宽度，但在 Excel 2003 中只能修改图片的高度。

本例文件参见光盘：..\第七章\ 7-26 重新描述错误信息.xlsm

2. 指定错误的处理方式

案例要求：创建新工作表并且命名为“总表”，如果工作簿中已经存在“总表”则不再新建工作表。

知识要点：Sheets.Add 方法、On Error GoTo ErrLine 语句、Application.DisplayAlerts 属性、WorkSheet.Delete 方法。

实现步骤：

step 1 新建工作簿，按<Alt+F11>组合键进入 VBE 窗口，然后单击菜单中的“插入”→“模块”命令。

step 2 在模块中录入以下代码：

```
Sub 新建总表A()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Sheets.Add after:=Sheets(Sheets.Count)
  Sheets(Sheets.Count).Name = "总表"
End Sub
```

step 3 单击过程的任意位置，并按<F5>键执行程序，程序会新建一个“总表”。

step 4 再次执行过程“新建总表 A”，由于工作簿中已经存在过“总表”，因此会弹出如图 7.41 所示的提示信息，同时在工作簿中留下一个新建的工作表。

很显然，以上代码有两个问题：一是提示信息不够人性化，二是程序有后遗症——留下一个空白工作表，需要手工删除。为了解决这两个问题，请继续第 5 步操作。

step 5 进入模块中，删除前面的过程，重新录入以下代码：

```
Sub 新建总表B()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  On Error GoTo ErrLine
  Sheets.Add after:=Sheets(Sheets.Count)
  Sheets(Sheets.Count).Name = "总表"
  Exit Sub
ErrLine:
  Application.DisplayAlerts = False
  Sheets(Sheets.Count).Delete
  MsgBox "工作簿中已经存在名为总表的工作表", vbInformation, "提示"
End Sub
```

step 6 执行过程“新建总表 B”，如果工作簿中没有“总表”，那么它会新建一个工作表，并且重命名为“总表”，如果已经存在“总表”则会产生如图 7.42 所示的提示信息。

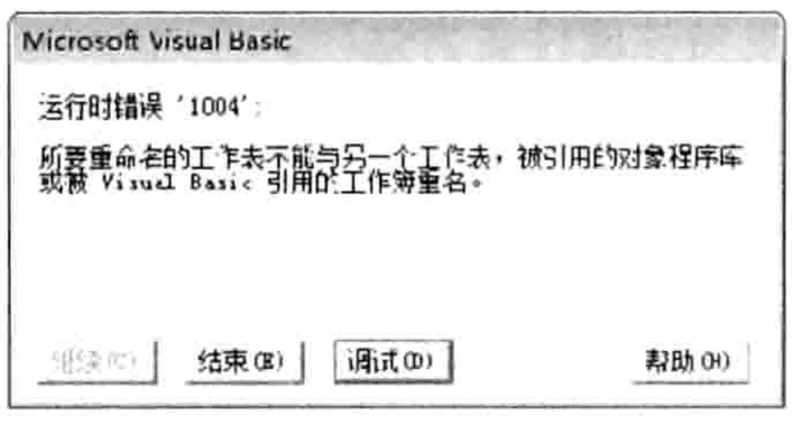


图 7.41 命名失败时的错误提示

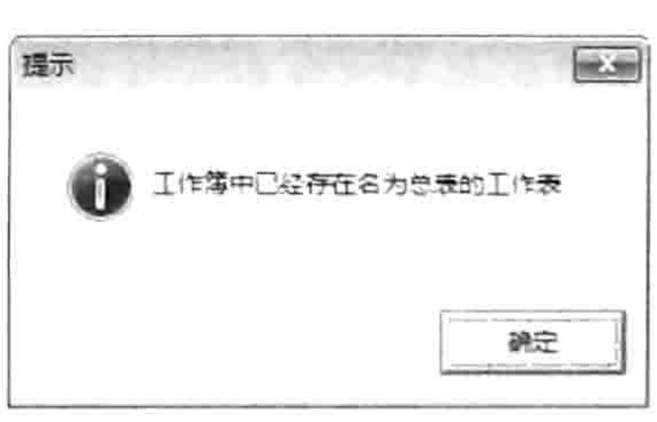


图 7.42 新的提示信息

思路分析：

过程“新建总表 B”中通过“On Error GoTo ErrLine”为程序创建了防错机制，如果程序在执行过程中出错，那么它会跳转到标签“ErrLine”之后继续执行，从而忽略出错的语句。改进的代码相对于改进前更人性化，改进后的代码屏蔽了原先的错误提示，提供了新的提示信息“工作簿中已

经存在名为总表的工作表”，同时删除了新建的工作表。

和前一个案例一样，在标签 ErrLine 之前添加“Exit Sub”语句是防止程序无错误时也执行标签 ErrLine 之后的代码。

语法补充：

（1）Worksheet.Delete 方法用于删除工作表。为了安全，在删除工作表时 Excel 会弹出一个对话框，当用户单击“删除”按钮后才能删除工作表。使用代码删除工作表时是不需要这个确认对话框的，它会降低程序的执行效率，因此 Worksheet.Delete 方法通常配合 Application.DisplayAlerts 属性一起使用。

（2）Application.DisplayAlert 属性用于控制 Excel 在执行宏时是否弹出提示信息，当赋值为 True 时允许弹出提示信息，赋值为 False 时表示禁止弹出提示信息。

Application.DisplayAlert 属性并不能控制所有提示信息，工作中通常用它控制三类提示信息——合并单元格时的提示信息、删除工作表时的提示信息、关闭未保存的工作簿时弹出的提示信息。在本例中，代码“Application.DisplayAlerts = False”用于关闭删除工作表时所弹出的提示信息。

本例文件参见光盘：..\第七章\7-27 指定错误的处理方式.xlsm

7.4.5　On Error Resume Next

On Error Resume Next 表示当程序出错时仍然继续执行后面的语句，禁止弹出错误提示及中断程序。

在工作中，On Error Resume Next 语句的应用频率非常高，因为在实际工作中不可预料的错误很多，利用这种防错机制可以确保程序顺利执行完毕，不会中途停止。

On Error Resume Next 语句可以置于过程中的任意位置，不过建议放在过程的最前端。

当然，On Error Resume Next 语句其实也是一把双刃剑，它可让程序不因某句代码出错而中断，但同时也会将某些意料之外的错误信息一并屏蔽掉，导致开发者可能错过纠错的机会。

在实际工作中，将编好的程序应用到工作中去之前应尽量多做测试，对于可以预料的错误提前防范。当确认代码没有问题后再在过程中使用 On Error Resume Next 语句。

On Error Resume Next 语句的应用相当广泛，不仅用它防止程序出错，很多时候也用它搭配 Err.Number 属性使用从而对某些特殊情况做判断。下面通过 3 个实例展示 On Error Resume Next 语句在实际工作中的应用。

1. 案例：计算人均预拨款

案例要求：如图 7.43 所示的是某企业的预拨款数目表，要求根据预拨款和部门的人数计算人均预拨款。

知识要点：On Error Resume Next、For Next 循环语句。

实现步骤：

step 1　新建工作簿，按<Alt+F11>组合键进入 VBE 窗口，然后单击菜单中的“插入”→“模块”命令。

step 2　在模块中录入以下代码：

```
Sub 人均预拨款()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  On Error Resume Next
  Dim Item As Byte
  For Item = 2 To Cells(Rows.Count, 1).End(xlUp).Row
```

```
    Cells(Item, 4) = Cells(Item, 1) / Cells(Item, 3)
  Next Item
End Sub
```

step 3 将光标定位于过程中任意位置，然后按<F5>键执行程序，程序会将各部门的人均预拨款计算出来存放在 D 列中，效果如图 7.44 所示。

	A	B	C	D
1	财务预拨款	部门	人数	人均
2	10000	人事部	1	
3	80000	业务部	4	
4	120000	生产部		
5	35000	后勤部	8	
6	35000	仓库	4	
7	28000	企划部		
8	15000	保安部	4	
9	38000	财务部	3	

图 7.43 预算表

	A	B	C	D
1	财务预拨款	部门	人数	人均
2	10000	人事部	1	10000
3	80000	业务部	4	20000
4	120000	生产部		
5	35000	后勤部	8	4375
6	35000	仓库	4	8750
7	28000	企划部		
8	15000	保安部	4	3750
9	38000	财务部	3	12667

图 7.44 计算平均预算

思路分析：

代码“Cells(Item, 1) / Cells(Item, 3)”表示用 A 列的预拨款除以 C 列的人数从而得到人均预拨款。由于 C 列部分单元格空白，空白单元格参与数学运算时当作 0 处理，同时又由于 0 不能做除数，因此计算到第 4 行时会出错，从而导致程序被中断，此后的所有单元格不再产生运算结果。基于此原因，本例在过程中添加了 On Error Resume Next 语句，它能让程序出错时继续执行下一句，完全无视错误的存在，从而将后面的已经填写人数的部门的人均预拨款项计算完成。

读者可以试着删除 On Error Resume Next 语句再执行代码，从而了解该代码的功用。

本例文件参见光盘：..\第七章\ 7-28 防错处理一.xlsm

2. 案例：错误三次则结束程序

案例要求：利用代码打开“生产表.xlsx”，如果输入密码错误时继续弹出输入框，连续错误三次则关闭程序。

知识要点：On Error Resume Next 语句、Do Loop 循环语句、Application.InputBox 方法、Workbooks.Open 方法、Err.Number 属性、Err.Clear 方法、ThisWorkbook.Path 属性。

实现步骤：

step 1 按<Alt+F11>组合键进入 VBE 窗口，然后单击菜单中的“插入”→“模块”命令。

step 2 在模块中录入以下代码：

```
Sub 打开工作簿()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  On Error Resume Next
  Dim i As Byte, PsdStr As String
  Do
    Err.Clear
    i = i + 1
    PsdStr = Application.InputBox("请输入密码：" & Chr(10) & "您还有" & 4 - i &
"次机会", "第" & i & "次输入密码", , , , , , 3)
    Workbooks.Open ThisWorkbook.Path & "\生产表.xlsx", , , , PsdStr
    If i = 3 Then
      MsgBox "对不起,你已错误三次,程序即将关闭"
      Exit Sub
    Else
      If Err.Number = 0 Then Exit Do
    End If
  Loop
```

```
    Application.StatusBar = "打开工作簿成功..."
End Sub
```

step 3　保存工作簿。

step 4　在本工作簿的相同路径下存放一个名为“生产表.xlsx”的工作簿，并且将其打开密码设置为“789”。

step 5　用鼠标单击过程中任意位置，按<F5>键执行程序，程序立即弹出如图 7.45 所示的对话框，在对话框中提示了用户当前是第几次输入密码，以及还剩下几次机会。

step 6　如果在对话框中输入密码 456，并单击“确定”按钮将弹出如图 7.46 所示的对话框。

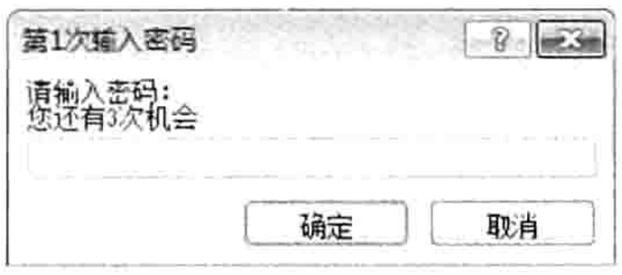


图 7.45　第一次输入密码

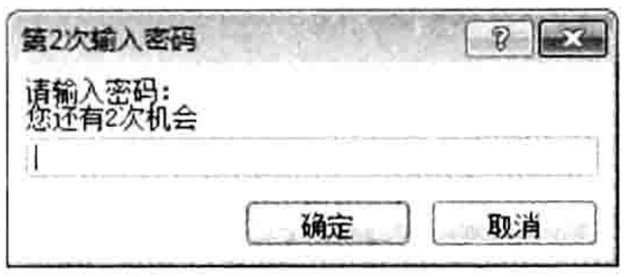


图 7.46　第二次输入密码

如果第三次输入错误的密码将弹出如图 7.47 所示对话框，同时表示这是最后一次机会。如果第三次输入错误密码则弹出如图 7.48 所示的对话框。

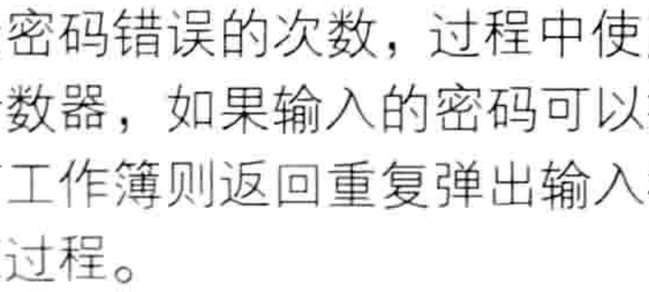


图 7.47　第三次输入密码

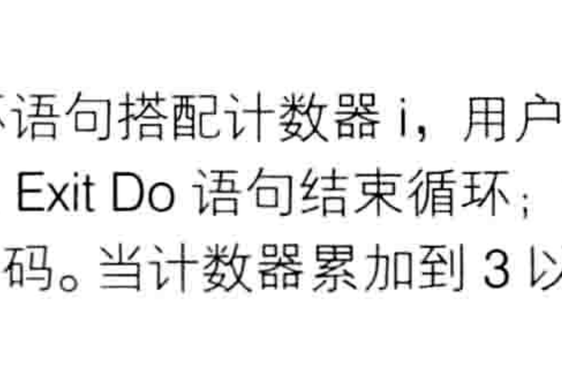


图 7.48　输入错误密码三次后的提示信息

如果在前三次中任意一次输入的密码是 789，那么程序会打开“生产表.xlsx”，同时在状态栏显示“打开工作簿成功...”。

思路分析：

Workbooks.Open 方法可以打开带密码的工作簿，它的第 5 参数 Password 用于指定密码。如果指定的密码有误并且未使用 On Error Resume Next 语句防错，则程序会弹出编号为 1004 的错误提示信息，然后中断程序。为了解决这个问题，在过程的前端加入 On Error Resume Next 语句，使程序忽略错误继续执行。

为了记录密码错误的次数，过程中使用了 Do Loop 循环语句搭配计数器 i，用户每输入一次密码累加一次计数器，如果输入的密码可以打开工作簿就通过 Exit Do 语句结束循环；如果用户输入的密码无法打工作簿则返回重复弹出输入框等待用户输入密码。当计数器累加到 3 以后就通过 Exit Sub 语句结束过程。

在判断密码是否错误时，本例的思路是先用该密码去打开工作簿，如果打开过程出错，那么 Err.Number 属性值必然不等于 0，此时通过 Do Loop 循环语句反复弹出输入框让用户重新输入密码即可。假设第一次输入密码有误，Err.Number 属性值等于 1004，等到第二次输入密码时，尽管密码输入正确，Err.Number 属性值仍然等于 1004，这会导致条件语句判断失误。为了解决此问题，在 Do Loop 循环语句中的 Workbooks.Open 之前需要使用 Err.Clear 方法清除错误。

语法补充：

（1）Workbooks.Open 方法用于打开工作簿，其语法如下：

```
Workbooks.Open(FileName, UpdateLinks, ReadOnly, Format, Password, WriteRes
Password, IgnoreReadOnlyRecommended, Origin, Delimiter, Editable, Notify,
```

```
Converter, AddToMru, Local, CorruptLoad)
```

其中 Password 参数代表工作簿的密码，如果指定的密码错误将产生编号为 1004 的错误提示，如果密码正确则会打开工作簿，并且引用该工作簿。

（2）Err.Clear 方法表示清除错误，也可以理解为将错误编码归零。

（3）Err.Number 属性代表错误编码，当执行代码过程中没有错误时此属性值为 0，因此通常用代码“If Err.Number <> 0 Then”来判断程序中是否有错误。

本例文件参见光盘：..\第七章\ 7-29 防错处理二.xlsm

3. 案例：隐藏所有公式

案例要求：一键隐藏工作表中的所有公式，即只能查看计算结果，禁止查看公式。

知识要点：On Error Resume Next 语句、Do Loop 循环语句、Application.InputBox 方法、Workbooks.Open 方法、Err.Number 属性、Err.Clear 方法、ThisWorkbook.Path 属性。

实现步骤：

step 1 按<Alt+F11>组合键进入 VBE 窗口，然后单击菜单中的“插入”→“模块”命令。

step 2 在模块中录入以下代码：

```
Sub 一键隐藏公式()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Dim rng As Range
  If ActiveSheet.ProtectContents Then
    MsgBox "工作表已保护,本程序拒绝执行！", vbInformation, "友情提示"
    Exit Sub
  Else
    On Error Resume Next
    Set rng = Cells.SpecialCells(xlCellTypeFormulas)
    If Err.Number <> 0 Then
      MsgBox "活动工作表中不存在公式", vbInformation, "友情提示"
    Else
Cells.FormulaHidden = False
      rng.FormulaHidden = True
ActiveSheet.Protect 123, False, True, False, False, True, True, True, True, True,
True, True, True, True, True, True
    End If
 End If
End Sub
```

step 3 按<Alt+F11>组合键返回工作表界面，假设工作表中有如图 7.49 所示的成绩表，表中部分区域有公式、部分区域无公式，按<Alt+F8>组合键打开“宏”对话框，然后执行过程“一键隐藏公式”命令，程序会隐藏工作表中的所有公式，效果如图 7.50 所示。

C3 =RANK(B3,B$3:B$11)

	A	B	C	D	E	F	G
1	A班				A班		
2	姓名	成绩	排名		姓名	成绩	排名
3	钟正国	86	4		王文今	59	6
4	陈丽丽	56	8		古山忠	51	9
5	胡不群	98	2		朱千文	57	7
6	朱华青	77	6		张大中	61	5
7	朱真光	99	1		朱未来	65	4
8	赵冰冰	87	3		张世后	55	8
9	周光辉	60	7		赵前门	93	1
10	周锦	84	5		洪文强	75	3
11	陈新年	53	9		陈秀雯	77	2

图 7.49 成绩表

C3

	A	B	C	D	E	F	G
1	A班				A班		
2	姓名	成绩	排名		姓名	成绩	排名
3	钟正国	86	4		王文今	59	6
4	陈丽丽	56	8		古山忠	51	9
5	胡不群	98	2		朱千文	57	7
6	朱华青	77	6		张大中	61	5
7	朱真光	99	1		朱未来	65	4
8	赵冰冰	87	3		张世后	55	8
9	周光辉	60	7		赵前门	93	1
10	周锦	84	5		洪文强	75	3
11	陈新年	53	9		陈秀雯	77	2

图 7.50 隐藏公式后的成绩表

如果工作表中没有公式，执行程序后会弹出“活动工作表中不存在公式”的信息框；如果工作表处于保护状态，执行程序后会弹出“工作表已保护，本程序拒绝执行！”的信息框。

思路分析：

隐藏公式需要修改单元格的 FormulaHidden 属性，如果工作表处于保护状态则无法修改，因此在隐藏公式之前有必要判断工作表是否处于保护状态。ProtectContents 属性值为 True 时表示工作表处于保护状态。

当工作表未保护时，使用 SpecialCells (xlCellTypeFormulas) 引用公式所在的单元格，并将它赋值给变量 rng。假设工作表中没有公式，SpecialCells 方法引用目标单元格时会出错，为了避免程序因出错而中断，在 SpecialCells 方法之前插入“On Error Resume Next”，然后再用条件语句判断 Err.Number 属性值是否不等于 0，如果不等于 0 则表示工作表没有公式，此时提示用户并且结束程序即可；如果 Err.Number 属性值等于 0，表示工作表中有公式，那么先将所有单元格的 FormulaHidden 属性赋值为 False，后将 Rng 区域的 FormulaHidden 属性赋值为 True，最后对工作表加密，Rng 变量所代表的区域就会自动隐藏公式。隐藏公式后，只能在单元格中查看公式的值，无法在编辑栏中看到公式本身。

程序的重点有两个：其一是引用所有公式所在的区域，使用 Range.SpecialCells 方法即可；其二是保护公式的同时不影响其他单元格，正确的操作思路是将所有单元格的 FormulaHidden 属性赋值为 False，而公式所在单元格的 FormulaHidden 属性赋值为 True，当使用 WorkSheet.Protect 方法保护工作表后 Rng 区域处于隐藏状态，其他单元格处于显示状态，并且可以正常编辑。

语法补充：

（1）WorkSheet.ProtectContents 属性代表工作表内容的保护状态，属性值为 True 表示已被保护，否则处于未保护状态。

（2）Range.SpecialCells 方法用于定位符合条件的目标对象，本例中将它的第 1 参数赋值为 xlCellTypeFormulas 表示只定位公式所在单元格。

（3）Range.FormulaHidden 属性代表单元格是否隐藏公式，将它赋值为 True 时表示隐藏公式，否则显示公式。此功能必须搭配 WorkSheet.Protect 方法使用，只有工作表处于保护状态时，FormulaHidden 属性为 True 的单元格才会真的被隐藏。

（4）WorkSheet.Protect 方法用于保护工作表。保护工作表对话框有很多选项，因此 WorkSheet.Protect 方法也有相当多的参数。其具体的语法如下：

```
WorkSheet.Protect(Password, DrawingObjects, Contents, Scenarios, UserInter
faceOnly, AllowFormattingCells, AllowFormattingColumns, AllowFormattingRows,
AllowInsertingColumns,    AllowInsertingRows,    AllowInsertingHyperlinks,
AllowDeletingColumns,  AllowDeletingRows,  AllowSorting,  AllowFiltering,
AllowUsingPivotTables)
```

对于每一个参数的含义，读者可以到 VBA 的帮助中查询，查询帮助的关键字为“Worksheet.Protect 方法”。

本例文件参见光盘：..\第七章\7-30 防错处理三.xlsm

7.4.6　On Error GoTo 0

On Error GoTo 0 语句表示禁止当前过程中任何已启动的错误处理程序。

假设在过程中有 On Error GoTo line 和 On Error Resume Next 语句，那么本语句可以令 On Error GoTo line 和 On Error Resume Next 失效，所以 On Error GoTo 0 语句不会单独使用。

如果在过程中使用了 On Error GoTo line 或者 On Error Resume Next 语句，限制了程序出错时的处理办法，而当某个条件下不再需要这些错误处理规则时，则可以插入 On Error GoTo 0 语句，在此语句之后如果产生了错误将会弹出错误提示，同时中断过程。

7.5 选择文件与文件夹

编写代码时需要处处注重代码的通用性，所谓的通用是指代码中尽可能少用 C5:D10、“C:\5月出货表.xlsm”或者“D:\生产表\”这类硬编码，而是让终端用户自己来选择区域、文件或者文件夹路径，否则程序的功能就会过于单一，灵活性、通用性大打折扣。

弹出对话框让用户选择区域使用 Application.Inputbox 方法可以实现，在前面的章节中有过不少案例。本节主要展示弹出一个浏览对话框让用户选择文件或者文件夹的技巧。

7.5.1 认识 FileDialog 对象

FileDialog 对象用于创建对话框，包括“打开”对话框、“另存为”对话框、“文件选取器”对话框和“文件夹选取器”对话框。

FileDialog 对象有 2 个方法和 13 个属性，读者可以从帮助中查询到这些方法与属性的详细介绍及案例演示，查询关键字为“FileDialog 对象成员”。

在此有必要罗列其中最常用的一个方法和两个属性。

（1）FileDialog.Show 方法

FileDialog.Show 方法用于显示 FileDialog 对象创建的对话框，没有参数，有一个返回值。当单击 FileDialog.Show 方法创建的对话框中的“打开”按钮时其返回值为-1，如果单击“取消”按钮则返回值为 0。

（2）FileDialog.AllowMultiSelect 属性

FileDialog.AllowMultiSelect 属性代表是否允许多选，将它赋值为 True 时表示允许多选，赋值为 False 时表示只能单选。

FileDialog.Show 方法可以创建四类对话框：“打开”对话框、“另存为”对话框、“文件选取器”对话框和“文件夹选取器”对话框，其中“另存为”对话框和“文件夹选取器”对话框不支持多选，即使将 FileDialog.AllowMultiSelect 属性赋值为 True 也只能单选。

（3）FileDialog.SelectedItems 属性

FileDialog.SelectedItems 属性代表用户所选择对象的集合，对象的类型由对话框类型而定。当 FileDialog.Show 方法创建的对话框是“打开”对话框时，那么 SelectedItems 代表用户选择的所有文件的集合；当 FileDialog.Show 方法创建的对话框是“文件夹选取器”对话框时，那么 SelectedItems 代表用户选择的所有文件夹的集合。

需要特别说明的是，尽管“另存为”对话框和“文件夹选取器”对话框不支持多选，但是它的 SelectedItems 属性仍然属于集合，需要使用 SelectedItems(1)才能引用文件名称或者文件夹名称。

FileDialog 对象可以创建 4 种对话框，对话框的类型由参数决定，具体语法如下：

```
Application.FileDialog(fileDialogType)
```

其中参数 FileDialogType 代表对话框类型，VBA 为此参数指定了一组 MsoFileDialogType 常量，FileDialogType 参数的取值范围受限于此常量。常量的值与含义见表 7-15。

表 7-15　MsoFileDialogType常量一览

名称	值	说明
msoFileDialogOpen	1	“打开”对话框
msoFileDialogSaveAs	2	“另存为”对话框
msoFileDialogFilePicker	3	“文件选取器”对话框
msoFileDialogFolderPicker	4	“文件夹选取器”对话框

从表 7-15 中可以得知：使用 msoFileDialogOpen 或者 1 作为 FileDialog 的参数都能创建一个打开文件的对话框，而创建浏览文件夹的对话框则应用 msoFileDialogFolderPicker 或者 4 作参数……

7.5.2　选择路径

要统计某路径下有多少文件，或者要求删除某路径下的所有文件、合并某路径下的所有工作簿数据等，都需要一个路径。在编程时不能手工指定这个路径名称，而是让用户自己根据实际需求选择路径，否则程序的实用性将大打折扣。

通过 FileDialog 对象创建一个选择路径的对话框可使用以下模板：

```
Sub 浏览并获取指定路径名称()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Dim PathSht As String
  With Application.FileDialog(msoFileDialogFolderPicker)
     If .Show = -1 Then PathSht = .SelectedItems(1) Else Exit Sub
  End With
  PathSht = PathSht & IIf(Right(PathSht, 1) = " \ ", "", " \ ")
  MsgBox PathSht
  '...更多代码...
End Sub
```

在过程中首先声明一个 String 型的变量，然后用 msoFileDialogFolderPicker 或者 4 作为 FileDialog 对象的参数创建对话框，接着利用 FileDialog.Show 方法显示这个对话框。

由于对话框中有“打开”和“取消”两个按钮，因此需要通过条件语句判断用户单击了哪一个按钮。代码“If .Show = -1 Then PathSht = .SelectedItems(1) Else Exit Sub”的含义是如果用户单击了“打开”按钮，则将用户选择的路径名称赋值给变量 PathSht，否则结束过程。

由于用户选择不同路径时对话框的返回值也不同，例如选择磁盘的根目录时得到的路径的最后一个字符是“\”，而选择了根目录下方的子文件夹时得到的路径最后一个字符不是“\”，为了统一路径，使用 IIF 函数判断最后一个字符是否为“\”，如果不是就添加一个“\”。

程序执行结束后，变量 PathSht 的值就代表用户选择的路径名称，以“\”结尾。

事实上也可以对对话框指定一个标题，更清晰地告知用户如何操作。例如在 FileDialog.Show 方法之前添加一句代码：

```
.Title = "请选择一个文件夹"
```

加入以上代码后，对话框的标题将由原来的“浏览”变为“请选择一个文件夹”，如图 7.51 所示。

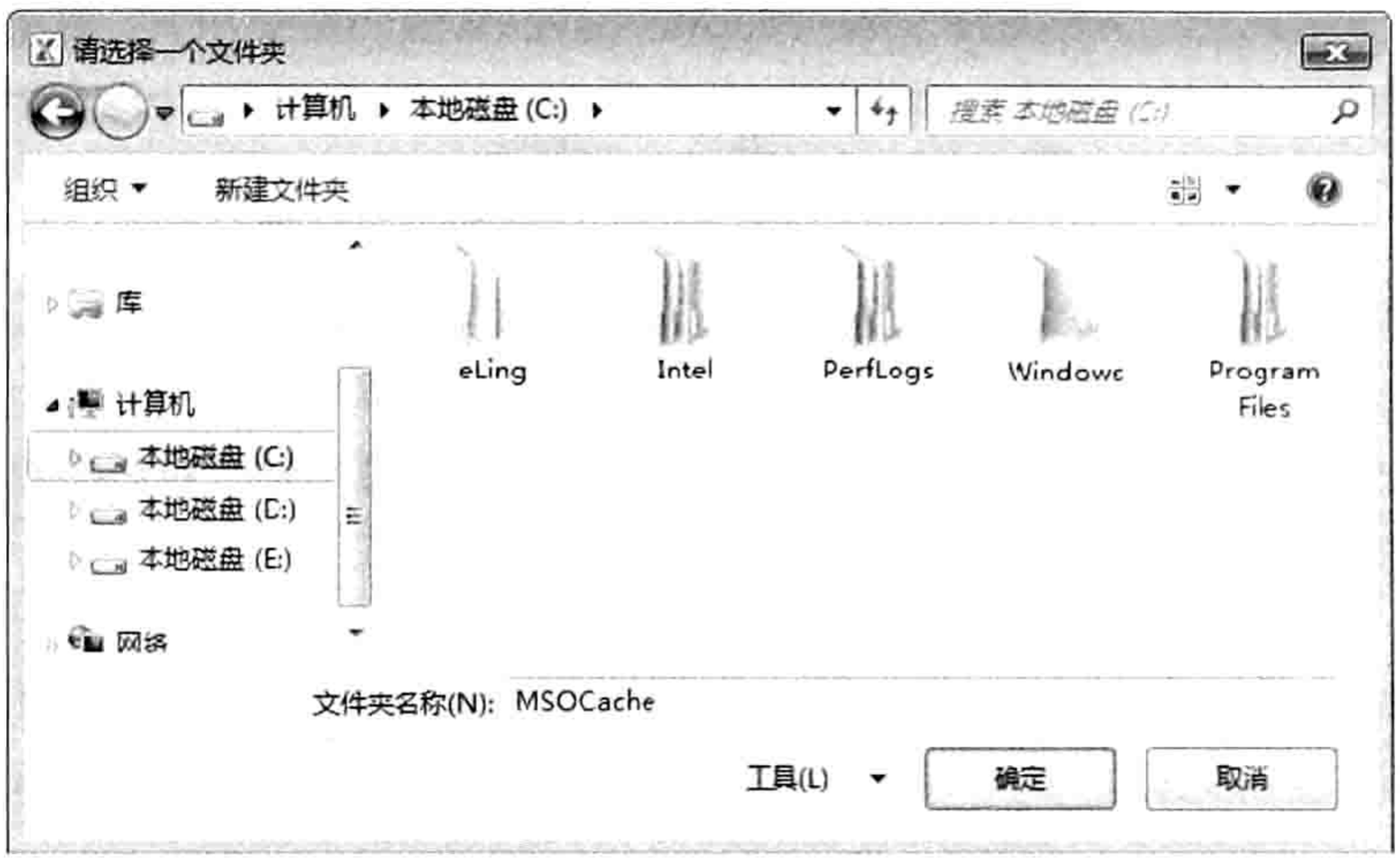


图 7.51　自定义对话框的标题

CD-ROM　本例文件参见光盘：..\第七章\7-31 浏览并获取指定路径名称.xlsm

7.5.3　选择文件

批量打开文件或者批量插入图片等都需要让用户从对话框中选择文件。

创建一个批量选择文件的对话框仍然使用 FileDialog 对象实现。

为了便于读者快速应用到工作中去，在此提供一个通用的模板：

```
Sub 浏览并报告所有文件名称()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
Dim Item As Integer
With Application.FileDialog(msoFileDialogFilePicker)
  If .Show =-1 Then
    For Item = 1 To .SelectedItems.Count
      MsgBox .SelectedItems(Item)
    Next Item
  Else
   Exit Sub
  End If
End With
  '...更多代码...
End Sub
```

以上过程通过 FileDialog 属性创建一个选择文件的对话框，然后利用条件语句判断用户单击了哪一个按钮，如果是单击“取消”按钮则结束过程，如果是单击“打开”按钮，那么逐一将用户选择的文件的完整名称显示在信息框中。

由于 FileDialog 对象的 AllowMultiSelect 属性默认值为 True，因此在代码中忽略了 AllowMultiSelect 属性仍然可以多选文件。

事实上，可以随意修改以上模板，将 Msgbox 函数替换为赋值或者打开文件。假设要将用户选择的所有文件的名称罗列在工作表中，可以按以下方式修改：

```
Sub 创建文件目录()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
Dim Item As Integer
With Application.FileDialog(msoFileDialogFilePicker)
  If .Show = -1 Then
```

```
    For Item = 1 To .SelectedItems.Count
     Cells(Item, 1) = .SelectedItems(Item)
    Next Item
  End If
End With
End Sub
```

本例文件参见光盘：..\第七章\7-31 浏览并获取指定路径名称.xlsm

7.5.4　按类型选择文件

Windows 采用扩展名来区分文件的类型，浏览文件的对话框也用扩展名作为条件筛选文件。Excel 自带的“打开”对话框就使用了筛选功能，在对话框中只能看到与 Excel 相关的文件，其他类型的文件已被自动隐藏起来。

创建具有筛选功能的对话框可用 GetOpenFilename 方法实现，其语法如下：

```
Application.GetOpenFilename(FileFilter, FilterIndex, Title, ButtonText,
MultiSelect)
```

GetOpenFilename 方法的 5 个参数都是可选参数，具体含义如表 7-16 所示。

表 7-16　GetOpenFilenam方法的参数说明

参数名称	功能描述
FileFilter	用于指定文件筛选条件的字符串
FilterIndex	指定默认文件筛选条件的索引号，取值范围为 1 到由 FileFilter 所指定的筛选条件数目。如果省略该参数，或者该参数的值大于可用筛选条件数，则使用第一个文件筛选条件
Title	指定对话框的标题。如果省略该参数，则标题为“打开”
ButtonText	仅限 Macintosh。Windows 系统中忽略此参数即可
MultiSelect	如果赋值为 True，则允许选择多个文件名。如果赋值为 False，则只能单选文件名，默认为 False

其中 FileFilter 参数比较复杂，它的功能是通过指定文件扩展名来设置筛选条件，由文件类型描述和包含 DOS 通配符的扩展名组成，中间以逗号分隔。

例如“"文本文件 (*.txt),*.txt"”或者“"文本文件,*.txt"”代表只筛选出文本文件。其中逗号前面部分可以随意书写，它用于说明当前的筛选条件，改用“ABC”也不影响功能，只是无法帮助用户快速理解罢了；逗号后面的部分用于指定文件的扩展名称，每一个字都不允许错误。可以通过以下步骤验证 FileFilter 参数为“"文本文件,*.txt"”时的筛选功能。

step 1　在 D 盘中新建一个文件夹，并命名为“文件”。

step 2　在“D:\文件\”路径下新建两个文本文件和两个 Excel 文件，再复制两个 jpg 图片和两个 png 图片到该路径下。

step 3　在 Excel 中按<Alt+F11>组合键进入 VBE 界面，然后单击菜单中的“插入”→“模块”命令。

step 4　在模块中录入以下代码：

```
Sub 浏览指定类型的文件名称()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  On Error Resume Next
```

```
  Dim FileName,i As Integer
FileName = Application.GetOpenFilename("文本文件,*.txt", , "请选择文本文件", ,
True)
  If Err.Number > 0 Then Exit Sub
  For i = 1 To UBound(FileName)
    MsgBox FileName(i)
  Next i
End Sub
```

step 5 单击过程中的任意位置，然后按<F5>键执行过程。

step 6 在对话框中选择路径“D:\文件”，此时在对话框中只能看到两个文本文件，效果如图 7.52 所示。

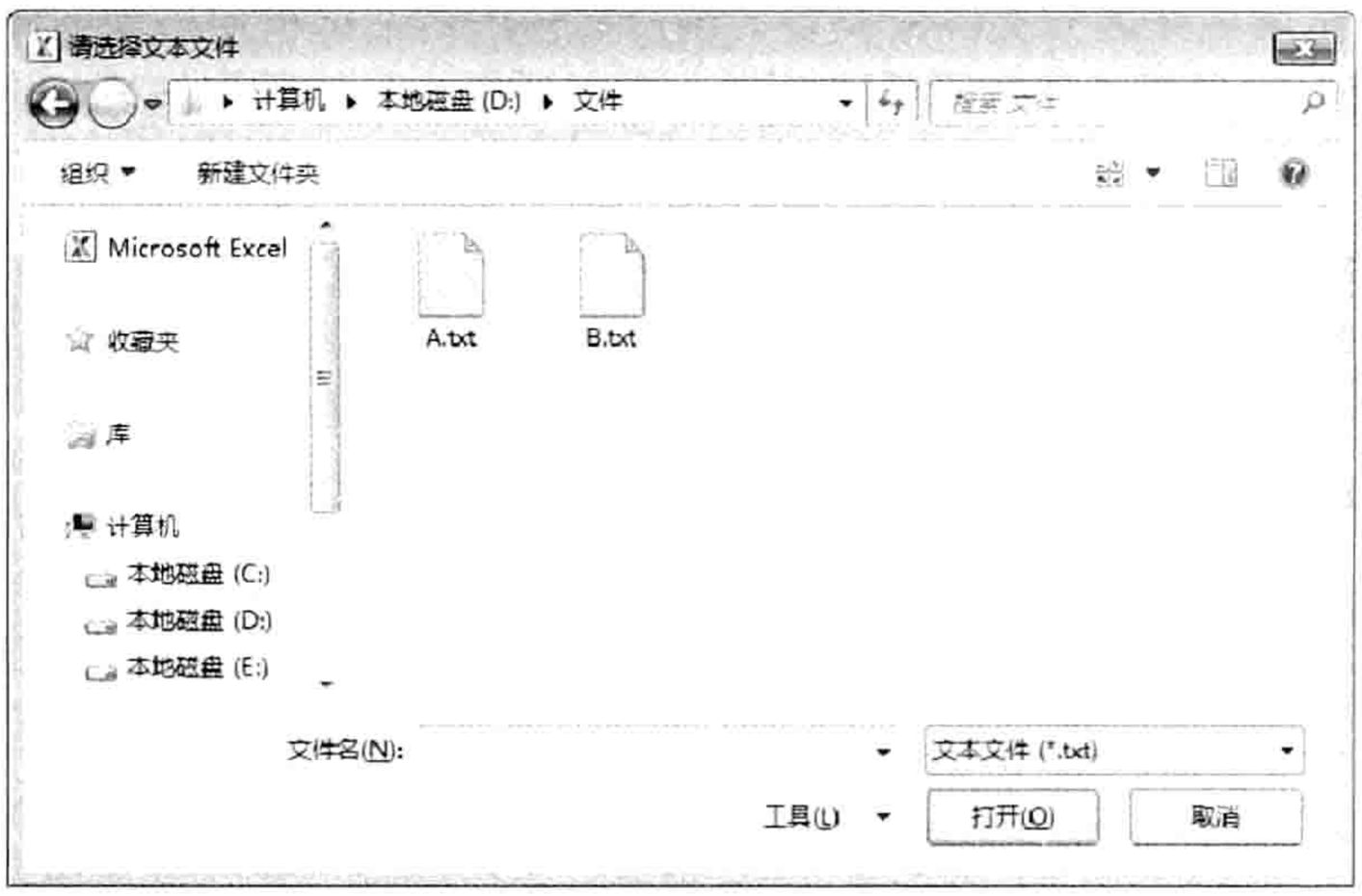


图 7.52 选择文本文件

很显然，代码中的参数“"文本文件,*.txt"”只能筛选出文本文件，将图片和 Excel 文件隐藏了，这刚好符合需求。

事实上，工作需求是变化的，而非单一的。有时要求在对话框中显示出多种类型的文件，GetOpenFilename 方法是否仍然可以满足需求呢？答案仍然是通过 FileFilter 参数实现。

GetOpenFilename 方法的 FileFilter 参数支持多种扩展名，不同扩展名之间使用分号隔开即可。例如“D:\文件”中 png 和 jpg 两种格式的图片文件，将它们的名称导入到工作表中可用以下代码：

```
Sub 批量导入图片文件名称()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  On Error Resume Next
  Dim FileName, i As Integer
  FileName = Application.GetOpenFilename("图片文件,*.jpg;*.png", , "请选择图片
文件", , True)
  If Err.Number > 0 Then Exit Sub
  For i = 1 To UBound(FileName)
    Cells(i, 1) = FileName(i)
  Next i
End Sub
```

以上代码的重点在于“"图片文件,*.jpg;*.png"”，它定义了 jpg 和 png 两种文件格式，如果还需要更多类型的文件，可以一并罗列出来，使用分号隔开即可。

执行以上过程后将弹出如图 7.53 所示的对话框。

图 7.53　选择图片文件

本例文件参见光盘：..\第七章\7-33 按类型浏览文件.xlsm

◆　◆　◆

第 8 章　让代码自动执行

Excel VBA 的主要存在价值是提升工作效率，以往需要 10 分钟的工作通过 VBA 来实现往往仅需单击一次按钮即可。同时，由于 Excel VBA 允许程序自动化执行，这无疑连按键的工夫也省了，从而将 VBA 的潜力发挥得更彻底了。

VBA 代码自动化执行主要通过自动宏和事件来实现，本章的重点在于介绍工作簿事件和工作表事件。尽管应用程序事件也属于事件的范畴，但由于应用程序事件涉及类的知识，比较复杂、晦涩，因此将它放在本书第 19 章中进行详解。

8.1　让宏自动执行

20 年前，早期版本的 Excel 就已经支持宏代码自动运行了，但是当时的自动宏功能极其有限。为了方便工作，后来在 Excel 中引入了“事件”这一新概念，事件比早期的自动宏强大得多。不过微软为了确保程序的兼容性还是保留了早期的自动宏，因此如今在 Excel 2010 或者 Excel 2013 中既可使用早期的自动宏又可使用事件。

8.1.1　Auto 自动宏

在 VBA 中，只要将宏命名为“Auto_Open”，并且保存在模块中，那么开启工作簿时就可以自动执行该程序。

对应地，如果宏程序的名称命名为“Auto_Close”，并且代码保存在模块中，那么在关闭工作簿时也可以自动执行该宏程序。

例如每次开启工作簿时让工作簿在状态栏中显示“四维实业公司人事报表”，而关闭工作簿时自动保存工作簿，那么可以使用以下代码实现：

```
Sub auto_open()'①代码存放位置：模块中
  Application.StatusBar = "四维实业公司人事报表"
End Sub
Sub auto_close()
  ThisWorkbook.Save
End Sub
```

在以上两段代码中，第一段的功能是打开工作簿时在状态栏显示“四维实业公司人事报表”，执行效果如图 8.1 所示。第二段则可以让工作簿关闭时自动保存，而不需要用户手工单击“保存”按钮。

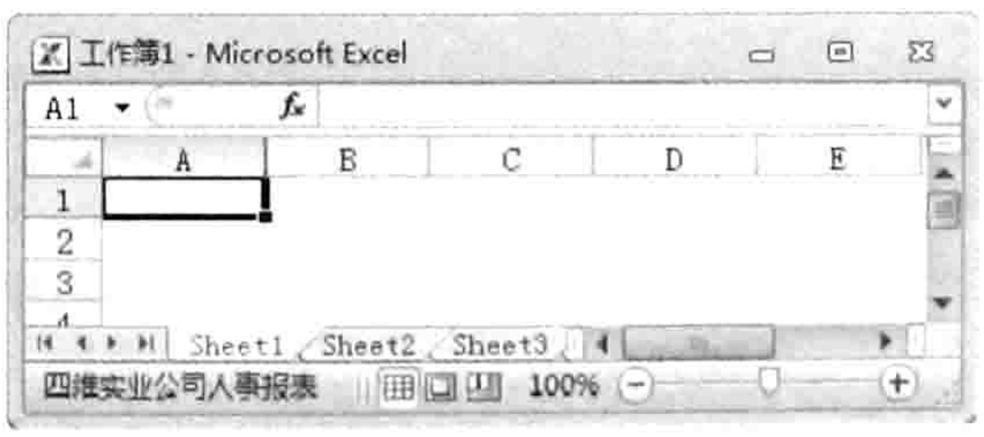


图 8.1　修改报表状态

本例文件参见光盘：..\第八章\ 8-1 自动宏.xlsm

8.1.2　升级版自动宏：事件

早期的 Excel 仅仅支持“Auto_Open”和“Auto_Close”两个自动宏，功能过于单一。为了满足日益繁复的工作需求，微软公司在 Excel 中引入了事件这个概念，而且每升级一次 Office 就会增加若干个事件。例如 Excel 2010 相对于 Excel 2007 就新增了 24 个事件，其中工作表事件 5 个，工作簿事件 7 个，详情请参考随书光盘中的“附录 C Excel 2010 的新增事件.pdf”。

事件的种类有很多，关于窗体与窗体控件的事件在本书第 17 章中会详细讲述，应用程序事件涉及类模块知识，在本书第 19 章中讲述，本章的重点在于工作簿事件和工作表事件。

1. 工作表事件

Excel 2010 提供了 14 种工作表事件，每一个工作表事件都有一个独一无二的触发条件，例如工作表的 Activate 事件在激活工作表时触发，Calculate 事件在重算公式时触发，Change 事件在改变单元格的值时触发，SelectionChange 事件在选区变化时触发……

工作表事件仅作用于代码所在的工作表，例如代码存放在 Sheet2 工作表中，那么对 Sheet2 工作表执行某个操作时可以触发该事件，在其他任意工作表中都无法触发此事件。

部分工作表事件拥有一个或者两个参数，部分工作表事件没有参数。当参数为 Target 时表示可以通过此参数获取当前的操作区域，例如 SelectionChange 事件中的参数 Target 表示当前选中的区域，Change 事件中的参数 Target 表示当前正在修改其值的区域……

当参数为 Cancel 时表示可以通过此参数控制原有功能的启用与禁用状态，例如 BeforeDoubleClick 事件是一个双击事件，原本双击单元格后可以进入编辑状态，若将此事件的 Cancel 参数赋值为 True，那么可以禁止单元格进入编辑状态；BeforeRightClick 事件属于单击右键时触发的事件，原本右击单元格后可以弹出右键菜单，若将此事件的 Cancel 参数赋值为 True 则可以禁止右击单元格时弹出右键菜单。

2. 工作簿事件

Excel 2010 提供了 36 种工作簿事件，每一个事件对应于一个独有的触发条件。

工作簿事件高于工作表事件，同时也包含工作表事件，因此任何一个工作表事件触发时都会同步触发工作簿事件。

工作表事件仅作用于代码所在的工作表，工作簿事件则作用于代码所在的工作簿或者工作簿中的所有工作表，因此一切工作表事件都可以用对应的工作簿事件代替。

工作簿事件也和工作表事件一样——部分事件无参数，部分事件有一到多个参数。当参数为 Sh 时，表示可以通过此参数引用触发事件的工作表对象。

工作簿的 SheetSelectionChange 事件的书写方式如下：

```
Private Sub Workbook_SheetSelectionChange(ByVal Sh As Object, ByVal Target As
Range)
End Sub
```

其中参数 Sh 代表触发当前事件的工作表。

当前正在操作哪一个工作表，那么参数 Sh 就代表哪一个工作表。

如果参数是 Target 或者 Cancel，那么其含义与工作表事件中的同名参数一致。

每个工作表事件都对应一个工作簿事件，例如 Worksheet_SelectionChange 事件对应 Workbook_SheetSelectionChange 事件，前者仅当代码所在工作表的选区变化时触发事件，后者则

在代码所在工作簿的任意工作表中选区变化时都触发事件。

如果要用工作簿的 SheetSelectionChange 事件替换工作表的 SelectionChange 事件，那么在 Workbook_SheetSelectionChange 事件过程中使用条件语句限制工作表名称即可。

可以通过以下步骤可以证明工作簿事件足以取代工作表事件：

step 1 新建一个工作簿，按<Alt+F11>组合键进入 VBE 界面。

step 2 双击工程资源管理器中的“Sheet2”，然后在代码窗口中录入以下事件过程代码：

```
'①代码存放位置：工作表事件代码窗口②随书光盘中有每一句代码的含义注释
Private Sub Worksheet_SelectionChange(ByVal Target As Range)
  If WorksheetFunction.Count(Target) > 0 Then
    Application.StatusBar = Target.Address(False, False) & "的平均值为:" &
WorksheetFunction.Average(Target)
  Else
    Application.StatusBar = ""
  End If
End Sub
```

代码的含义是：如果 Target 区域中的数值个数大于 0，那么在状态栏显示选区的地址和选区的平均值，否则让状态恢复原状。由于以上工作表事件过程的代码存放在 Sheet2 工作表中，因此代码的作用对象是 Sheet2 工作表，在其他工作表中选择区域时不触发事件。

如图 8.2 所示是在 Sheet2 工作表中选择 A2:C4 区域时的操作结果，状态栏显示了选区中的平均值。

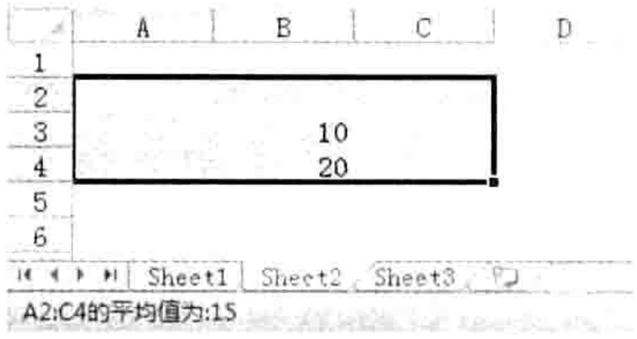


图 8.2 在状态栏显示区域的平均值

step 3 关闭工作簿，然后新建一个工作簿，再按<Alt+F11>组合键进入 VBE 界面。

step 4 双击工程资源管理器中的 ThisWorkbook，然后在代码窗口中录入以下事件过程代码：

```
'①代码存放位置：ThisWorkbook②随书光盘中有每一句代码的含义注释
Private Sub Workbook_SheetSelectionChange(ByVal Sh As Object, ByVal Target As
Range)
  If Sh.Name = Worksheets("Sheet2") Then
    If WorksheetFunction.Count(Target) > 0 Then
      Application.StatusBar = Target.Address(False, False) & "的平均值为:" &
WorksheetFunction.Average(Target)
    Else
      Application.StatusBar = ""
    End If
  End If
End Sub
```

相对于工作表事件的代码，以上工作簿事件代码添加了一句条件语句，从而将事件过程的触发条件限制在 Sheet2 工作表中，因此两组代码的功能完全一致。

☞**知识补充：**Count 和 Average 都属于工作表函数，不是 VBA 的函数，因此不能直接调用，必须在函数名称前添加父对象 WorkSheetfunction 才能调用。有 90%以上的工作表函数都可

以通过这种方式调用，而 Sqrt、if、Abs、Char、Code、Indirect、Address、Cell、Date、Datedif、Year、Day、Month、NOW、Today、Value、N、T、Mod、Maxa、Mina、Mimute、Second、Hour、Sin、Rmb、Info、Int、Isblank、Na、Not、Row、Rows、Column、Columns、Tan、True、False、Type、Upper、Lower、Left、Right、Mid 等函数则不允许，因为 VBA 已经提供了相同功能的函数。例如 VBA 函数 Sqr 等同于工作表函数 Sqrt、VBA 函数 Left 等同于工作表函数 Left、VBA 函数 IIf 等同于工作表函数 If、VBA 函数 Chr 等同于工作表函数 Char……

step 5 返回工作表界面，在 Sheet2 工作表中随意录入数值，然后选中区域，在状态栏将显示选区的平均值。

step 6 在其他工作表中录入任意数值并选中区域，状态栏不会显示平均值，这证明工作簿事件可以替代工作表事件。

8.1.3 事件的禁用与启用

Excel 的事件可以通过代码启用或者禁用，当 Application 对象的 EnableEvents 属性被赋值为 True 时表示启用事件，将它赋值为 False 时则禁用事件，完整的代码如下：

```
Application.EnableEvents = True
Application.EnableEvents = False
```

通常在以下两种情况时需要禁用事件。

1. 临时关闭事件

当工作簿中存在工作表事件或者工作簿事件的代码时，只要符合条件代码就会自动执行。而某些时候不需要触发这些事件，那么就有必要使用代码禁止事件。

以下案例用于演示禁用事件和启用事件。

案例要求： C 盘中的“生产报表.xlsm”工作簿中设置了一个名为 Workbook_Open 的工作簿事件，现要求用代码打开该工作簿，并且禁止打开工作簿时执行 Workbook_Open 事件。

知识要点： Application.EnableEvents 属性、Workbooks.Open 方法。

实现步骤：

step 1 按<Alt+F11>组合键进入 VBE 窗口，然后单击菜单中的“插入”→“模块”命令。

step 2 在模块中录入以下代码：

```
Sub 打开文件()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Application.EnableEvents = False
  Workbooks.Open "C:\生产报表.xlsm "
  Application.EnableEvents = True
End Sub
```

step 3 光标定位于过程“打开文件”中的任意位置，然后按<F5>键执行代码，程序会自动打开 C 盘中的“生产报表.xlsm”工作簿，而且工作簿中的 Workbook_Open 事件不会执行。

思路分析：

过程“打开文件”首先利用代码“Application.EnableEvents = False”禁用事件，然后使用 Workbooks.Open 方法打开工作簿，在此期间不管工作簿中有多少个事件过程都会一概被禁止执行。特别是批量打开工作簿时禁用事件更有现实意义——防止事件过程干扰代码执行。

例如某个工作簿的 Workbook_Open 事件使用了 Msgbox 函数，那么程序执行过程中必须人工关闭 Msgbox 函数产生的信息框，然后批量打开工作簿的代码才得以继续执行。

2. 防止事件的连锁反应

部分事件可能会导致连锁反应——事件过程的代码导致事件再次执行。例如在工作表的SelectionChange事件中使用了Range.Select方法，而Range.Select方法又会触发SelectionChange事件，两者相互作用下就产生了连锁反应。

例如以下事件过程：

```
'代码存放位置：工作表事件代码窗口
Private Sub Worksheet_SelectionChange(ByVal Target As Range)
  Target.Offset(1, 0).Select
End Sub
```

事件的触发条件是选区变化，而事件过程中的代码“Target.Offset(1, 0).Select”会导致选区变化，从而再次触发条件，从而导致SelectionChange事件连锁反应。例如选择单元格A1时代码“Target.Offset(1, 0).Select”会选择单元格A2，此句代码会触发事件从而选择单元格A3，然后继续选择单元格A4、A5……

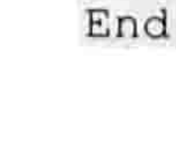
再例如以下事件过程：

```
'代码存放位置：工作表事件代码窗口
Private Sub Worksheet_Change(ByVal Target As Range)
 Target = Target + 1
End Sub
```

事件过程的本意是在单元格中录入数据时自动累加1，然而事件的连锁反应会让事件的执行结果与原本需求背道而驰。假设在单元格A1中录入1，事件过程的结果并不是2，由于事件的连锁反应会导致单元格A1的值反复累加下去。

将Application.EnableEvents属性赋值为False可以关闭事件的这种连锁反应。对于上述Worksheet_SelectionChange事件按以下方式修改如下：

```
'代码存放位置：工作表事件代码窗口
Private Sub Worksheet_SelectionChange(ByVal Target As Range)
  Application.EnableEvents = False
  Target.Offset(1, 0).Select
  Application.EnableEvents = True
End Sub
```



本例文件参见光盘：..\第八章\ 8-2 启用与禁用事件.xlsm

8.1.4 事件的特例

所谓特例，是指按照常规思路大家认为可能触发事件的动作它却没有触发事件，而大家认为可能不会触发事件的动作事实上却触发了事件。以下列举6个特例。

1. 删除空白单元格时触发Worksheet_Change事件

当单元格是空白状态时，按<Delete>键会触发Worksheet_Change事件。例如：

```
'①代码存放位置：工作表事件代码窗口②随书光盘中有每一句代码的含义注释
Private Sub Worksheet_Change(ByVal Target As Range)
  MsgBox "您在修改：" & Target.Address(0, 0)
End Sub
```

将以上代码放在 Sheet1 的代码窗口中，然后在工作表中删除空白单元格 A1 时会弹出如图 8.3 所示的提示框。

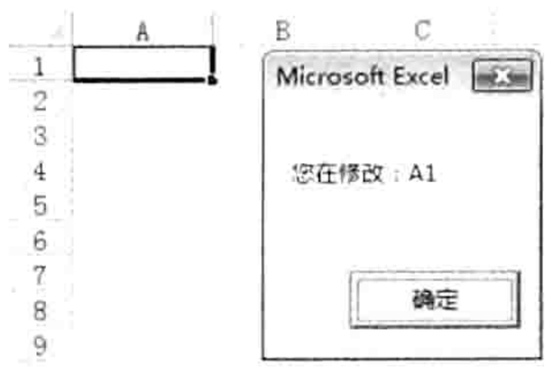


图 8.3　删除空单元格触发 Worksheet_Change 事件

事实上，如果将 Worksheet_Change 事件看作单元格的值变化时触发的事件，那么将难以理解上述结果。如果将它看作修改单元格的值时触发的事件就很容易理解了，因为“变化”是一个形容词，表明一个状态，当 A1 单元格的值为空白时，按<Delete>键后 A1 的状态并未产生变化；“改变”是一个动作，强调的是动作而非结果，因此尽管删除空白单元格时单元格的值尚未变化，但是改变 A1 的值这一个动作已经发生了，那么就可以触发事件。

2. 插入批注时不触发任何事件

往任何单元格插入批注时都不触发任何事件。

可以将批注理解为是一个悬浮于单元格之上的图形对象，导入图形的目的和结果都不是改变单元格的值。

3. 修改单元格格式不触发事件

不管对单元格设置任何格式，包括背景色、填充图案、对齐方式及边框、定义数字格式等都不触发任何事件。

虽然定义数字格式时单元格的显示内容发生了改变，然而单元格的内容在本质上是不变化的，改变的是单元格的外观、状态，而不是单元格的内容。

4. 清除格式会触发 Worksheet_Change 事件

清除单元格格式时会触发 Worksheet_Change 事件。

我们不清楚微软当初为什么设计清除格式这种不修改单元格的内容的操作会触发工作表事件，只要记住这个特例即可。

5. 数据分列时不触发事件

假设 A1 单元格中的文本为“中国/广东/广州”，以“/”为分隔符将它进行分列，效果如图 8.4 所示。在分列时不会触发任何事件，包括 Worksheet_Change 事件和 Worksheet_SelectionChange 事件。

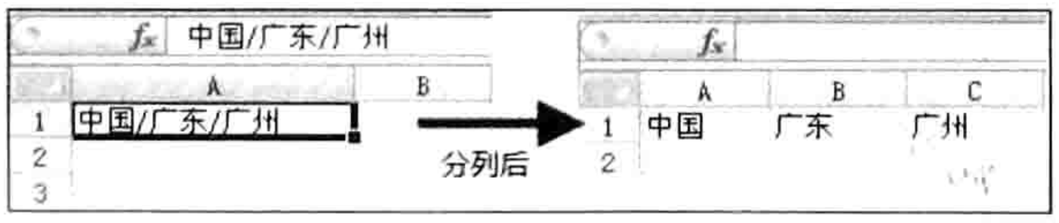


图 8.4　数据分列

6. 表单控件修改单元格不触发事件

表单控件中的复选框、列表框和组合框等都可以修改单元格的值，但是经过多次测试这个操作过程不会触发任何工作表或者工作簿事件。

但是 ActiveX 控件中的复选框、列表框和组合框却可以触发工作表事件、工作簿事件和应用程序事件。

8.2 工作表事件应用案例

工作表事件仅用于代码所在的工作表，本节对部分工作表事件提供详细的案例，促进读者对工作表事件有深入的认知。

8.2.1 在状态栏提示最大值的单元格地址

案例要求：选择“生产表”中任何区域时在状态栏中会提示选区中最大值的单元格地址。

实现步骤：

step 1 打开如图 8.5 所示的工作簿，按<Alt+F11>组合键进入 VBE 界面。

step 2 双击工程资源管理器中的“生产表”，并在“生产表”代码窗口中录入以下代码：

```
'①代码存放位置：工作表事件代码窗口②随书光盘中有每一句代码的含义注释
Private Sub Worksheet_SelectionChange(ByVal Target As Range)
  If WorksheetFunction.Count(Target) = 0 Then
    Application.StatusBar = ""
  Else '否则
    Application.StatusBar = Target.Find(WorksheetFunction.Max(Target), , ,
xlWhole).Address
  End If
End Sub
```

step 3 按<Alt+F11>组合键返回工作表界面，选择任意区域，假设选区中没有数值，那么状态栏将没有任何变化；假设选区中有数值，那么在状态栏中会显示选区中最大值的地址，效果如图 8.5 所示。

图 8.5 在状态栏显示选区最大值地址

思路分析：

工作表的 SelectionChange 事件是工作表中的选区发生变化时触发的事件，参数 Target 代表当前的选区。事件过程的代码必须放在“生产表”对应的代码窗口中，否则代码不生效。

本例要求计算选区中最大值的单元格地址，由于当选区没有数值时 Max 函数计算出来的最大值是 0，而选区中事实上没有 0，因此利用 Range.Find 方法查找 0 时将会出错。为了解决此问题，本例采用工作表函数 Count 计算 Target 区域是否有数值，如果没有数值则恢复状态栏信息，则在状态栏中显示最大值的单元格地址。

语法补充：

（1）Count 函数用于计算一个或者多个区域中的数值个数，它是工作表函数，必须使用“WorksheetFunction.Count”形式才可以调用。对于 Left、Right、Mid、Instr 这类 VBA 函数则可以直接书写函数名称，忽略 WorksheetFunction。

（2）Range.Find 方法表示在指定区域中查找字符串，返回值是一个 Range 对象。本例要求在选区中查找，而事件过程的参数 Target 即代表选区，因此本例代码使用的是 Target.Find 而非 Cells.Find。

本例文件参见光盘：..\第八章\ 8-3 在状态栏提示最大值的单元格地址.xlsm

8.2.2　快速录入出勤表

案例要求：在如图 8.6 所示的出勤表的 B2:H11 区域中单击产生字符“迟到”，双击则产生字符“早退”，而右击单元格时则产生字符“请假”。

I2　=SUMPRODUCT(--($B2:$H2=I$1))

	A	B	C	D	E	F	G	H	I	J	K
1	姓名	1日	2日	3日	4日	5日	6日	7日	迟到合计	早退合计	请假合计
2	赵								0	0	0
3	钱								0	0	0
4	孙								0	0	0
5	李								0	0	0
6	周								0	0	0
7	吴								0	0	0
8	郑								0	0	0
9	王								0	0	0
10	冯								0	0	0
11	陈								0	0	0

图 8.6　出勤表

实现步骤：

step 1　在工作表标签栏单击鼠标右键，在弹出的菜单中选择“查看代码”命令。

step 2　在代码窗口中录入以下工作表事件过程代码：

```
'①代码存放位置：工作表事件代码窗口②随书光盘中有每一句代码的含义注释
Private Sub Worksheet_SelectionChange(ByVal Target As Range)
  If Not Intersect(Target, Range("B2:H11")) Is Nothing Then
    Intersect(Target, Range("B2:H11")) = "迟到"
  End If
End Sub
```

以上事件过程表示选择 B2:H11 区域的单元格时自动产生字符“迟到”。

step 3　继续录入以下事件过程代码：

```
'①代码存放位置：工作表事件代码窗口②随书光盘中有每一句代码的含义注释
Private Sub Worksheet_BeforeDoubleClick(ByVal Target As Range, Cancel As
Boolean)
  If Not Intersect(Target, Range("B2:H11")) Is Nothing Then
      Cancel = True
      Intersect(Target, Range("B2:H11")) = "早退"
  End If
End Sub
```

以上事件过程表示双击 B2:H11 区域的任意单元格后禁止单元格进入编辑状态，然后在单元格中自动产生字符“早退”。

step 4　继续输入以下事件过程代码：

```
'①代码存放位置：工作表事件代码窗口②随书光盘中有每一句代码的含义注释
Private Sub Worksheet_BeforeRightClick(ByVal Target As Range, Cancel As
Boolean)
  If Not Intersect(Target, Range("B2:H11")) Is Nothing Then
    Cancel = True
    Target(1) = "请假"
```

```
    End If
End Sub
```

以上事件过程表示右击 B2:H11 区域中任意单元格后禁止单元格弹出右键菜单，然后在单元格中自动产生字符“请假”。

step 5 返回工作表界面，单击 B2:H11 区域中任意单元格即可产生字符“迟到”，若在 B2:H11 区域中双击单元格则可产生字符“早退”，右击单元格可产生字符“请假”。如果在 B2:H11 以外的区域单击、双击或者右击则不会触发事件，或者准确地说是虽然触发了事件，但由于不符合条件因此不会执行条件语句中的代码。

思路分析：

本例需要单击、双击和右击单元格时产生不同的字符，而这 3 个操作刚好对应 3 个工作表事件，因此本例使用了 3 个事件来完成需求。

由于本例要求只在 B2:H11 区域中产生“请假”、“迟到”和“早退”，忽略其他区域，因此 3 个事件过程中都使用了条件语句 If Then 判断 Target 对象与 B2:H11 是否存在交集，只有存在交集时才向 Target 中输入数据。

语法补充：

（1）Not 运算符表示否定运算，例如将 True 变成 False 或将 False 变成 True。当两个区域没有交集时 Intersect 方法的返回值是 Nothing，可以使用“Is Nothing”来判断。两个区域有交集时返回值是一个 Range 对象，但不能使用“Is Range”来判断，因此采用 Not 求反即可。

（2）BeforeRightClick 事件过程中的 Cancel 参数用于控制右击单元格时是否弹出右键菜单，由于在 B2:H11 区域中右击时只需要产生字符“请假”，不需要弹出右键菜单，因此当 Target 与 B2:H11 区域存在交集时将 Cancel 参数赋值为 True，屏弊原本的右键菜单。

本例文件参见光盘：..\第八章\ 8-4 快速录入出勤表.xlsm

8.2.3 在状态栏显示选区的字母、数字、汉字个数

案例要求：选择任意区域时，在状态栏中会显示选区中的字符个数，以及字母个数、数字个数和汉字个数。

实现步骤：

step 1 在工作表标签栏单击鼠标右键，在弹出的右键菜单中选择“查看代码”命令。

step 2 在代码窗口中录入以下工作表事件过程代码：

```
'①代码存放位置：工作表事件代码窗口②随书光盘中有每一句代码的含义注释
Private Sub Worksheet_SelectionChange(ByVal Target As Range)
 Dim 字符 As String, 字符数 As Long, 汉字 As Long, 字母 As Long, 数字 As Long, 其他符号
 Dim rng As Range, Item As Integer, rng As Range
 If WorksheetFunction.CountA(Target) = 0 Then
  Application.StatusBar = ""
 Else
  For Each rng In Intersect(Target, ActiveSheet.UsedRange)
   字符数 = 字符数 + Len(rng)
   For i = 1 To Len(rng)
    字符 = Mid(rng, i, 1)
    If 字符 Like "[一-隣]" = True Then
```

```
        汉字 = 汉字 + 1
      ElseIf 字符 Like "[a-zA-Z]" = True Then
        字母 = 字母 + 1
      ElseIf 字符 Like "[0-9]" = True Then
        数字 = 数字 + 1
      Else
        其他符号 = 其他符号 + 1
      End If
    Next
  Next
    Application.StatusBar = "字数：" &字符数& "个，其中：" & "汉字：" &汉字& "个，
" & "字母：" &字母& "个，" & "数字：" &数字& "个，" & "其他符号：" &其他符号& "个"
  End If
End Sub
```

step 3 返回工作表界面，选择空白区域，状态栏无任何反应；选择已用区域中的非空单元格，在状态栏中将会提示字符总数，以及汉字个数、字母个数、数字个数和其他符号个数。其他符号是指标点、空格、片假名等，效果如图 8.7 所示。

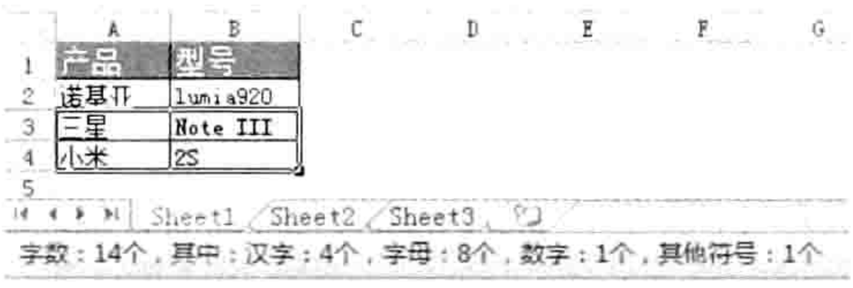


图 8.7　在状态栏显示选区字数

思路分析：

Worksheet_SelectionChange 事件是选区变化时触发的事件，它的参数 Target 代表当前选区。为了提升代码的执行效率，本例首先在条件语句中通过工作表函数 CountA 判断 Target 区域是否为空白区域，如果是空白区域则恢复状态栏的显示内容，然后直接结束过程；如果选区不是空白区域则使用 Intersect 方法提取选区与已用区域的交集，并用循环语句遍历该交集，在循环体中逐一统计每个单元格的字符数量。

在过程中判断选区是否为空白区域，以及提取选区与已用区域的交集两个步骤极为重要，它可以避免浪费系统资源，提升程序的执行效率。

在统计选区的字符数量时，首先使用 For Each Next 循环语句遍历选区与已用区域的交集，然后使用 Len 函数计算每个单元格的字符数量，将它们累加后即为选区的字符总数。而统计汉字个数、字母个数和数字个数时则需要在 For Each Next 循环语句中嵌套 For Next 循环，通过它遍历单元格的每一个字符，然后配合 Like 运算符逐一判断每个字符是汉字、字母还是数字，最后再根据字符的类型累加对应的变量。

当循环完成后，根据变量的值可以获得汉字、字母、数字和其他字符的总数量。

本例代码仅用于演示工作表事件和 Like 运算符的用法，代码的执行效率并不是最优的。如果在实际工作中有此类需求，应该使用数组的技术优化代码。不过数组技术在本书的第 13 章才会涉及，因此在本例中不宜提供与数组相关的优化代码，读者可以学完数组知识后再返回此处改写代码。

语法补充：

（1）Mid 是 VBA 的函数，用于从字符串中提取指定长度的字符。尽管它与工作表函数 Mid 的功能一致，每个参数的功能也一致，但两者在使用上还是有所区别的。工作表函数 Mid 有 3 个必选参数，分别是字符串、起始位置和长度，VBA 的 MID 函数有两个必选参数和一个可选参数，第

3 参数表示长度，如果忽略第 3 参数则表示长度等于第 1 参数的总长度。具体语法如下：

```
Mid(string, start[, length])
```

（2）Like 是 VBA 的运算符，用于比较两个字符或者字符串，返回值是 True 或者 False。Like 运算的语法如下：

```
result = string Like pattern
```

其中 result 是返回值，string 代表用来比较的字符串，pattern 代表比较条件。Like 在执行字符比较时支持通配符和字符区间。判断 A1 单元格的值是否包含“中国”可用以下代码：

```
MsgBox Range("a1") Like "*中国*"
```

代码中“*”代表任意长度的任意字符，因此比较条件“*中国*”的含义是以任意长度的任意字符开始、中间是“中国”且以任意长度的任意字符结尾。如果 A1 单元格包含“中国”二字，那么信息框中将显示 True。

再例如判断 A1 单元格是否包含 3 个字符，那么应使用以下代码：

```
MsgBox Range("a1") Like "???"
```

通配符“?”代表单个任意字符，因此“???”代表长度为 3 的任意字符。

Like 运算符最强大的地方莫过于它允许使用字符区间作为比较条件，字符区间以“[”开始，以“]”结尾，中间插入字符范围，表示一个区间。例如“[a-z]”代表所有小写字母、“[A-Z]”代表所有大写字母、“[0-9]”代表所有数字、“[一-陽]”代表所有汉字……

CD-ROM 本例文件参见光盘：..\第八章\ 8-5 选区字符统计.xlsm

8.2.4 实时监控单元格每一次编辑的数据与时间

案例要求：保存 B 列每个单元格的修改记录，包括修改后的数据和修改时间。

实现步骤：

step 1 在工作表标签栏中单击鼠标右键，在弹出的菜单中选择“查看代码”命令。

step 2 在代码窗口中录入以下工作表事件过程代码：

```
'①代码存放位置：工作表事件代码窗口②随书光盘中有每一句代码的含义注释
Private Sub Worksheet_Change(ByVal Target As Range)
    If Target.Column <> 2 Or Target.Rows.Count = Rows.Count Or Target.Columns.
Count = Columns.Count Then
      Exit Sub
    Else
     Dim rng As Range, TimeStr As String
     TimeStr = Format(Now, "m月d日hh:mm:ss")
     Application.ScreenUpdating = False
     For Each rng In Intersect(Target, Columns(2))
      If Not rng.Comment Is Nothing Then
        rng.Comment.Text rng.Comment.Text & Chr(10) & TimeStr & ":  " & IIf(rng
= "", "【清空】", rng)
      Else
        rng.AddComment TimeStr & ":  " & IIf(rng = "", "【清空】" ", rng)
      End If
```

```
rng.Comment.Shape.TextFrame.AutoSize = True
      Next
      Application.ScreenUpdating = True
    End If
End Sub
```

step 3　按<Alt+F11>组合键返回工作表界面，在 A 列输入收支项目，在 B 列输入金额，那么 B 列的任意单元格的编辑记录都会在批注中体现出来，如图 8.8 所示。

step 4　删除 B3 单元格中的值，然后再查看 B3 单元格中的批注。批注中会保留第一次输入的值 -1250，同时添加一条新的记录，包含时间及“【清空】”，从而让用户随时可以查询到该单元格的所有修改记录，效果如图 8.9 所示。

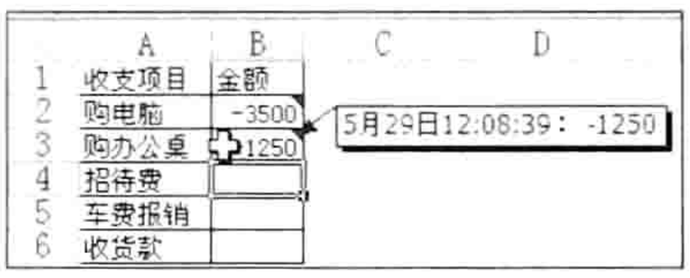


图 8.8　记录当前输入的数据及时间

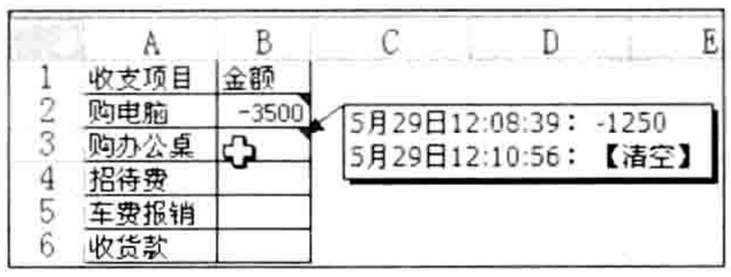


图 8.9　清空单元格时记录时间及保留以前的记录

step 5　选择 B4:C6 区域，输入数字“400”后按<Ctrl+Enter>组合键表示在多单元格同时输入数据，然后查看批注，可以发现仅 B 列的单元格会按要求记录时间和内容，其他列则自动被忽略，效果如图 8.10 所示。

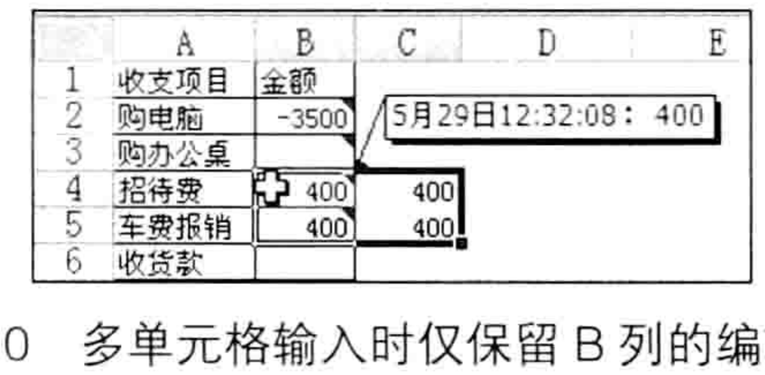


图 8.10　多单元格输入时仅保留 B 列的编辑记录

思路分析：

由于需求是在修改 B 列的值时在批注中呈现所有修改记录，因此使用工作表的 Change 事件，它的功能是修改单元格的值时自动执行代码。

在过程中首先利用条件语句判断 Target 区域的列号是否等于 2（表示当前正在 B 列录入数据），不等于 2 则结束过程。如果 Target 区域的列号为 2 还需要继续判断 Target 是否为整行或者整列，如果是也直接结束过程，因为对整行或者整列添加批注需要消耗太多的时间，影响代码的效率，而且在正常的情况下用户并不会整列输入数据，因此 Target 代表整行或者整列时不记录修改值与时间。

当确定 Target 符合条件以后，首先利用 Now 函数生成当前时间，然后通过 For Each Next 循环语句遍历 Traget 与 B 列的交集，逐一向单元格中添加批注，在批注中生成修改内容，以及修改时间。由于单元格中可能已经有批注也可能没有批注，因此生成批注前有必要判断单元格中是否存在批注，如果没有批注则使用 AddComment 方法生成批注，如果有则在原来的批注内容之后追加新的内容。

在生成批注后，为了方便用户查看批注内容，应该让批注框的高度和宽度随批注内容的长短自动变化，因此本例将批注框的 AutoSize 属性赋值为 True，其中“Comment.Shape.TextFrame”代表批注的文本框。

由于批量添加批注并且设置其 AutoSize 属性时屏幕会反复闪动，既不美观又影响代码的执行效率，因此在循环语句之前通过代码“Application.ScreenUpdating = False”关闭屏幕更新。

语法补充：

（1）Range.Column 属性代表单元格或者区域的列号，当 Range 是一个区域时只取左上角单

个单元格的列号，而不是生成一个数组，这一点与工作表函数 column 大大不同。

（2）Application.ScreenUpdating 属性赋值为 True 时表示执行代码过程屏幕可以即时更新，赋值为 False 时表示禁止更新。通常在频繁插入与删除单元格、图形对象，或者批量打开工作簿时有必要关闭屏幕更新，一切操作完成后再开启更新，从而加快代码的执行速度。

在默认状态下屏幕更新处于开启状态，Excel 会将每一个步骤的操作结果呈现给用户，而这个过程是需要时间的，如果在执行操作前关闭更新，待一切操作完成后再一次性地更新则可以节约时间，提升工作效率。

（3）Range.AddComment 方法表示向单元格中插入批注，其语法如下：

```
Range.AddComment(Text)
```

参数 Text 代表批注的内容，必须是文本，如果被添加到批注中的内容是数值应该先使用 Cstr 函数将它转换成文本，例如：

```
Range("a1").AddComment (CStr(12))
```

（4）AutoSize 属性用于控制对象是否自动调整大小，自动调整大小才能显示完整的文本内容。文本框才有 AutoSize 属性，Comment 对象和 Shape 对象都没有，因此使用“Comment.Shape.TextFrame”引用批注的文本框，然后再对 AutoSize 属性赋值。

本例文件参见光盘：..\第八章\ 8-6 在批注中存放所有修改记录.xlsm

8.2.5 利用数字简化公司名输入

案例要求：某公司有 5 个客户，分别为“广东长风汽车有限公司”、“湖南衡大塑胶生产厂”、“江苏天信集团”、“珠海电信公司”和“北京天虹印刷厂”，在工作表 A 列中希望可以通过数字 1～5 来完成 5 个常用公司名称的录入工作。

实现步骤：

step 1 在工作表标签栏单击鼠标右键，在弹出的菜单中选择“查看代码”命令。

step 2 在代码窗口中录入以下工作表事件过程代码：

```
'①代码存放位置：工作表事件代码窗口②随书光盘中有每一句代码的含义注释
Private Sub Worksheet_Change(ByVal Target As Range)
  If Target.Column = 1 Then
    If Target.Count > 1 Then Exit Sub
    Select Case Target.Value
    Case 1
      Target = "广东长风汽车有限公司"
    Case 2
      Target = "湖南衡大塑胶生产厂"
    Case 3
      Target = "江苏天信集团"
    Case 4
      Target = "珠海电信公司"
    Case 5
      Target = "北京天虹印刷厂"
    End Select
  End If
End Sub
```

step 3　返回工作表界面，在 A2 单元格中输入 1，如图 8.11 所示。当按<Enter>键后，单元格中的 1 将转换成“广东长风汽车有限公司”，如图 8.12 所示。

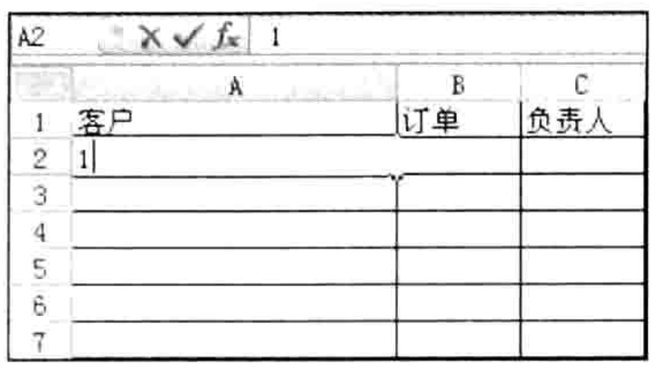


图 8.11　在 A 列单个单元格中输入 1

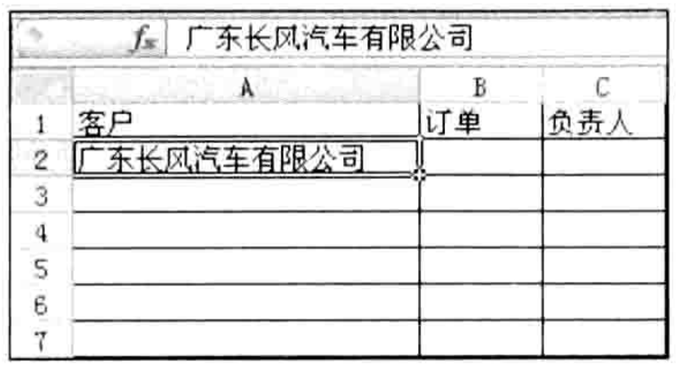


图 8.12　将 1 替换成“广东长风汽车有限公司”

思路分析：

输入数字后自动转换成长地名有很多办法可以实现，包括自定义单元格的数字格式、自动替换和本例所用的工作表事件。其中自定义格式仅适用于公司名称字符不超过 4 个的情况，自动替换功能对所有工作簿都生效，而本例要求仅对 A 列生效，因此采用工作表事件是最理想的办法。

本例的事件过程使用了两个 If Then 语句嵌套，从而只有在 A 列的单个单元格中输入字符时才触发条件。当然，也可以使用 or 运算符将两个条件连接起来，从而只用单个 If Then 语句限定条件。

在执行数字转公司名称过程时本例采用的是 Select Case 语句，其含义是在 Target 中输入 1～5 时分别替换成对应的公司名称。如果不使用 Select Case 语句可以改用工作表函数 Vlookup 实现，代码如下：

```
'①代码存放位置：工作表事件代码窗口②随书光盘中有每一句代码的含义注释
Private Sub Worksheet_Change(ByVal Target As Range)
  If Target.Column = 1 Then
    If Target.Count > 1 Then Exit Sub
    On Error Resume Next
    Application.EnableEvents = False
    Target = WorksheetFunction.VLookup(Target.Value, [{1,"广东长风汽车有限公司
";2,"湖南衡大塑胶生产厂";3,"江苏天信集团";4,"珠海电信公司";5,"北京天虹印刷厂"}], 2,
False)
    Application.EnableEvents = False
  End If
End Sub
```

代码中“{1,"广东长风汽车有限公司";2,"湖南衡大塑胶生产厂";3,"江苏天信集团";4,"珠海电信公司";5,"北京天虹印刷厂"}”是一个用于工作表函数的常量数组，在 VBA 中不支持这种形式的常量数组，但是可以使用“[]”将它们转换成 VBA 可以接受的数组。

过程中“Application.EnableEvents = False”的功能是避免 Worksheet_Change 事件的连锁反应。

语法补充：

VLookup 是工作表函数，需要通过 WorksheetFunction 对象才能调用。不过 Vlookup 函数找不到目标时不会在单元格中产生错误值，而是中断过程，这是在 VBA 中调用函数与在单元格中使用工作表函数的区别所在。代码中的 On Error Resume Next 语句的存在价值就在于避免用户录入 1～5 以外的字符时导致代码出错，从而中断过程。

事实上，本例还可以采用 Choose 函数替代 Vlookup 函数，代码如下：

```
Private Sub Worksheet_Change(ByVal Target As Range)'①代码存放位置：工作表事件代
码窗口
  If Target.Column = 1 Then
    If Target.Count > 1 Then Exit Sub
    If Target = 1 Or Target = 2 Or Target = 3 Or Target = 4 Or Target = 5 Then
```

```
Target = Choose(Target, "广东长风汽车有限公司", "湖南衡大塑胶生产厂", "江苏天信集团
", "珠海电信公司", "北京天虹印刷厂")
  End If
End Sub
```

本例文件参见光盘：..\第八章\ 8-7 利用数字简化公司名称输入.xlsm

8.2.6 录入数据时自动跳过带公式的单元格

案例要求：如图 8.13 所示是一个产量统计表，其中 D 列、F 列和 G 列分别包含计算良品数、不良率和良品率的公式。现要求录入数据并按<Enter>键后总是自动跳过公式所在的单元格，定位于下一个待输入数据的单元格。例如在 C2 单元格输入数据并按<Enter>键后将自动选择 E2 单元格；在 E2 单元格中输入数据并按<Enter>键后可自动选择 H2 单元格；在 H2 单元格中输入数据并按<Enter>键将自动选择 A3 单元格……

D2 =IF(B2="","",B2-C2)

	A	B	C	D	E	F	G	H
1	机台号	产量	废品	良品	返修品	不良率	良品率	操作员
2								
3								
4								
5								

图 8.13 含有公式的产量统计表

实现步骤：

step 1 在工作表标签栏中单击鼠标右键，在弹出的菜单中选择“查看代码”命令。

step 2 在代码窗口中录入以下工作表事件过程代码：

```
'①代码存放位置：工作表事件代码窗口②随书光盘中有每一句代码的含义注释
Private Sub Worksheet_SelectionChange(ByVal Target As Range)
  If Target.Count = 1 Then
    Application.MoveAfterReturnDirection = xlToRight
    If Target.HasFormula Then Target.Offset(0, 1).Select
    If Target.Column = 9 Then Target.Offset(1, -8).Select
End If
End Sub
```

step 3 返回工作表界面，在 B2 单元格输入产量并按<Enter>键，活动单元格将成为 C2 单元格。当在 C2 单元格中输入废品数量并按<Enter>键后，活动单元格成为 E2 单元格而不是 D2 单元格，因为 D2 单元格中有公式。当在 E2 单元格按<Enter>键后活动单元格成为 H2 单元格。当在 H2 单元格中输入操作员姓名后，活动单元格将变成 A3 单元格，而非 I2 单元格。

思路分析：

尽管本例的需求是录入数据后自动定位下一个不包含公式的单元格，但不宜使用 Worksheet_Change 事件，而应该采用 Worksheet_SelectionChange 事件。

在 Worksheet_SelectionChange 事件中，将 MoveAfterReturnDirection 属性赋值为 xlToRight 表示按<Enter>键后光标向右移动，这符合输入数据时的日常习惯。然后利用 If Then 语句判断 Target 中是否存在公式，如果有公式则选中它右边的一个单元格。由于 Range.Select 方法会触发 Worksheet_SelectionChange 事件的连锁反应，因此当多个连续的单元格中有公式时会一并跳过，直到遇到第一个不含公式的单元格。

为了提升录入数据的效率，本例代码还加入了 “If Target.Column = 9 Then Target.Offset(1,

-8).Select”，它表示在第 8 列（H 列）录入数据并按<Enter>键后可以自动定位下一行的首个单元格，从而让代码代替手工定位，在数据录入过程中不再需要运用鼠标，大大节约了操作时间。

语法补充：

（1）Application.MoveAfterReturnDirection 属性用于控制按<Enter>键后活动单元格的移动方向，它的可选范围受 XlDirection 常量限制，XlDirection 常量的名称与说明见表 8-1。

表 8-1　XlDirection常量一览

名称	值	说明
xlDown	-4121	向下
xlToLeft	-4159	向左
xlToRight	-4161	向右
xlUp	-4162	向上

（2）Range.HasFormula 属性代表单元格是否有公式，当值为 True 时表示有公式，值为 False 时表示没有公式。

本例文件参见光盘：..\第八章\ 8-8 自动跳过公式区.xlsm

8.2.7　对选择区域进行背景着色

案例要求：Excel 2010 的选区与非选区的区别极不明显，这给输入和查看数据带来不便。现要求选择单元格时突出显示指定的区域，如果选择单个单元格，那么突出单元格所在的行与列，如果选择区域则突出区域本身。

实现步骤：

step 1　在工作表标签栏中单击鼠标右键，在弹出的菜单中选择“查看代码”命令。

step 2　在代码窗口中录入以下工作表事件过程代码：

```
'①代码存放位置：工作表事件代码窗口②随书光盘中有每一句代码的含义注释
Private Sub Worksheet_SelectionChange(ByVal Target As Range)
  Cells.FormatConditions.Delete
  If Target.Count = 1 Then
    With Target.EntireRow.FormatConditions
      .Add xlExpression, , True
      .Item(1).Interior.ColorIndex = 14
    End With
    With Target.EntireColumn.FormatConditions
      .Add xlExpression, , True
      .Item(2).Interior.ColorIndex = 14
    End With
  Else
    With Target.FormatConditions
      .Add xlExpression, , True
      .Item(1).Interior.ColorIndex = 14
    End With
  End If
End Sub
```

step 3　返回工作表界面，单击 D5 单元格，那么 D 列和第 5 行将会显示指定的背景颜色，效果如图 8.14 所示。

step 4 选择 C3:E7 区域，那么 C3:E7 区域将显示指定的背景色，效果如图 8.15 所示。

	A	B	C	D	E	F	G
1	姓名	语文	数学	化学	地理	政治	计算机
2	赵雯	85	77	79	64	65	89
3	钱三强	50	88	91	86	52	71
4	林文强	93	90	69	99	94	52
5	于政美	98	68	76	89	52	80
6	陈丽如	73	65	81	83	63	64
7	张政光	92	92	80	100	96	61
8	黄美珠	85	99	62	77	55	100
9	陈嫣然	84	50	79	55	55	90

图 8.14 突出当前行与当前列

	A	B	C	D	E	F	G
1	姓名	语文	数学	化学	地理	政治	计算机
2	赵雯	85	77	79	64	65	89
3	钱三强	50	88	91	86	52	71
4	林文强	93	90	69	99	94	52
5	于政美	98	68	76	89	52	80
6	陈丽如	73	65	81	83	63	64
7	张政光	92	92	80	100	96	61
8	黄美珠	85	99	62	77	55	100
9	陈嫣然	84	50	79	55	55	90

图 8.15 突出当前选区

思路分析：

突出显示选区有很多方法，其中代码比较简单破坏性又小的办法是使用条件格式为区域添加背景颜色。本例突出显示选区或者活动单元格所在行与列的思路如下：

首先删除活动工作表中的所有条件格式。然后利用 If Then 语句判断 Target 区域中的单元格数量是否等于 1，如果是 1 则分别对它所在的行与所在列添加条件格式，让单元格显示为编码 14 的背景颜色；如果不是 1 则直接对 Target 所在区域添加条件格式，让单元格显示为编码 14 的背景颜色。

为了避免反复添加条件格式，在过程的前端必须使用代码“Cells.FormatConditions.Delete”删除工作表中的所有条件格式，不过这也会带来一个副作用——当工作表中有条件格式时会被工作表事件的代码清除。因此本案例的代码仅适用于不需要使用条件格式的工作簿。

语法补充：

（1）FormatConditions.Delete 方法表示删除所有条件格式，如果使用 FormatConditions(2).Delete 则表示只删除第 2 个条件格式。

（2）FormatConditions.Add 方法表示创建一个新的条件格式，其语法如下：

```
FormatConditions.Add(Type, Operator, Formula1, Formula2)
```

其中必选参数 Type 表示条件格式的类型，参数 Operator 代表运算符，Formula1 和 Formula2 分别代表与条件格式相关联的第一个、第二个表达式或值。假设要让选区中大于等于 70 并且小于等于 90 的单元格显示为红色背景，应使用以下代码：

```
Sub 将值为 70～90 的单元格显示为红色背景()'代码存放位置：模块中
  Selection.FormatConditions.Add xlCellValue, xlBetween, "70", "90"
  Selection.FormatConditions(1).Interior.ColorIndex = 3
End Sub
```

条件格式的类型比较多，没有可能将它们都记下来，好在可以通过录制宏记录与条件格式相关的一切操作，在实际工作中需要用到条件格式时仅需录制宏即可，不必手工书写代码。

CD-ROM 本例文件参见光盘：..\第八章\8-9 背景着色.xlsm

8.3 工作簿事件应用案例

工作簿级别的事件高于工作表级的事件，它对 ThisWorkbook 中所有工作表和单元格都生效。可以使用工作簿事件完成一切工作表事件的同等功能。善用工作簿事件可以让程序自动化，甚至做到让工作无人值守，一切按预设的步骤自动完成。

8.3.1　新建工作表时自动设置页眉

案例要求：在工作簿中有一个名为“样本”的工作表，已经设置好页眉与页脚，为了让所有工作表都保持相同的页眉与页脚，而且节约手动设置的时间，现要求在任意时候新建的工作表都自动复制“样本”工作表中的页眉与页脚信息。

实现步骤：

step 1　新建一个工作簿，并且将第一个工作表命名为“样本”。

step 2　对“样本”工作表设置好页眉与页脚。

step 3　按<Alt+F11>组合键打开 VBE 界面。

step 4　如果此时没有显示工程资源管理器则通过按<Ctrl+R>组合键调出工程资源管理器，然后双击 ThisWorkbook 进入工作簿事件代码窗口，并在窗口中录入以下事件过程代码：

```
'①代码存放位置：ThisWorkbook②随书光盘中有每一句代码的含义注释
Private Sub Workbook_NewSheet(ByVal Sh As Object)
  With Sh.PageSetup
    .LeftHeader = Worksheets("样本").PageSetup.LeftHeader
    .CenterHeader = Worksheets("样本").PageSetup.CenterHeader
    .RightHeader = Worksheets("样本").PageSetup.RightHeader
    .LeftFooter = Worksheets("样本").PageSetup.LeftFooter
    .CenterFooter = Worksheets("样本").PageSetup.CenterFooter
    .RightFooter = Worksheets("样本").PageSetup.RightFooter
  End With
End Sub
```

step 5　按<Alt+F11>组合键返回工作表界面，按<Shift+F11>组合键新建一个工作表，然后在工作表中随意输入字符。

step 6　按<Ctrl+P>组合键进入预览状态，在预览界面可以看到新工作表中的页眉和页脚与“样本”工作表完全一致。

思路分析：

Workbook_NewSheet 是工作簿事件，事件过程的代码必须存放在 ThisWorkbook 代码窗口才生效。本事件的触发条件是在创建新表时触发，事件的应用范围只限于代码所在的工作簿。事件过程的参数 Sh 代表新建的表，它可能是工作表也可能是图表，还可能是 4.0 宏表。

在本例过程中，由于需要引用 6 次“Sh.PageSetup”，因此采用 With 语句引用该对象，从而提高代码的书写速度和执行速度。然后对该对象的 LeftHeader、CenterHeader、RightHeader、LeftFooter、CenterFooter、RightFooter 共 6 个属性赋值，这 6 个属性分别代表左边页眉、中间页眉、右边页眉、左边页脚、中间页脚和右边页脚，它们都是可读也可写的属性，因此读取“样本”的 6 个属性然后赋予 Sh 对象的相同属性即完成题目的需求。

由于页眉与页脚相关的一切操作都可以通过录制宏获得代码，因此不需要花时间记忆这些代码的书写方式，更不需要学习它们的语法。

本例文件参见光盘：..\第八章\ 8-10 自动设置工作表页眉.xlsm

8.3.2 未汇总则禁止打印与关闭工作簿

案例要求：如图 8.16 所示的工作簿中有若干个产量表和一个汇总表，每个表中都可能有一个“合计”单元格，在它右方的单元格用于存放产量合计。现要求每次打印和关闭工作簿之前都检查是否存在未汇总的工作表，如果有任何一个工作表未汇总则禁止打印和关闭工作簿。

	A	B	C	D	E
1	机台	产量			
2	1#	7875			
3	2#	7901			
4	3#	7448			
5	4#	7994			
6	5#	7019			
7	6#	5966			
8	7#	5456			
9	8#	5781			
10	9#	7365			
11	合计	62805			

一车间 二车间 三车间 汇总表

图 8.16 产量表

实现步骤：

step 1 打开如图 8.16 所示的工作簿，按<Alt+F11>组合键打开 VBE 界面。

step 2 如果没有显示工程资源管理器则按<Ctrl+R>组合键调出工程资源管理器，然后双击 ThisWorkbook 进入工作簿事件代码窗口，并且在窗口中录入以下两个事件过程代码：

```
'①代码存放位置：ThisWorkbook②随书光盘中有每一句代码的含义注释
Private Sub Workbook_BeforeClose(Cancel As Boolean)
  Dim sht As Worksheet, ShtName As String, rng As Range
  For Each sht In Worksheets
    Set rng = sht.UsedRange.Find("合计", , , xlPart)
    If Not rng Is Nothing Then
      If Len(rng.Offset(0, 1)) = 0 Then
        ShtName = ShtName & Chr(13) & sht.Name
      End If
    End If
  Next sht
  If Len(ShtName) > 0 Then
    Cancel = True
    MsgBox "以下工作表尚未汇总，禁止关闭工作簿" & ShtName, vbInformation, "提示"
  End If
End Sub
Private Sub Workbook_BeforePrint(Cancel As Boolean)
  Dim sht As Worksheet, ShtName As String, rng As Range
  For Each sht In Worksheets
    Set rng = sht.UsedRange.Find("合计", , , xlPart)
    If Not rng Is Nothing Then
      If Len(rng.Offset(0, 1)) = 0 Then
        ShtName = ShtName & Chr(13) & sht.Name
      End If
    End If
  Next sht
  If Len(ShtName) > 0 Then
    Cancel = True
    MsgBox "以下工作表尚未汇总，禁止打印工作表" & ShtName, vbInformation, "提示"
  End If
End Sub
```

step 3　按<Alt+F11>组合键返回工作表界面，然后单击“打印”按钮，假设工作簿中有任意工作表未汇总，在打印数据之前会弹出如图 8.17 所示的信息框，然后禁止用户打印。

step 4　按<Alt+F4>组合键关闭工作簿，会弹出如图 8.18 所示的信息框。

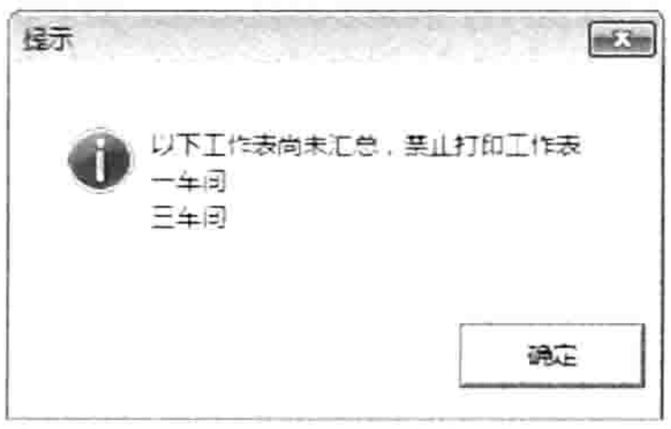


图 8.17　未汇总则禁止打印工作表

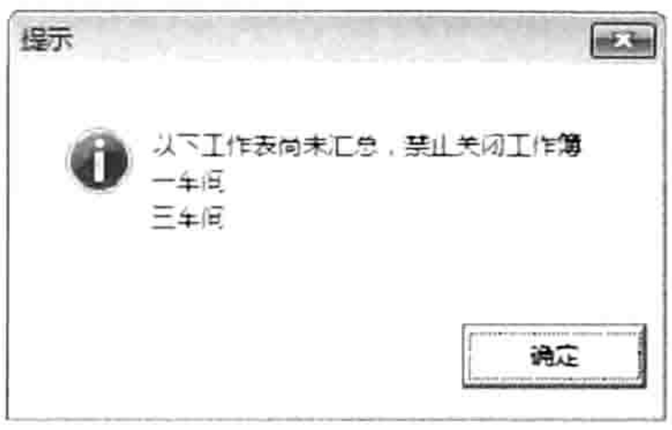


图 8.18　未汇总则禁止关闭工作簿

如果此时在“一车间”和“三车间”工作表的合计单元格中录入合计结果，再次关闭工作簿或者打印工作表则一切正常。

思路分析：

本例要求未汇总时禁止打印工作表，以及禁止关闭工作簿，因此需要使用两个事件。

Workbook_BeforeClose 事件是工作簿事件，在单击“关闭”按钮或者按<Alt+F4>组合键之后、实际关闭工作簿之前触发，它的参数 Cancel 用于控制是否允许关闭工作簿，赋值为 True 时表示禁止取消关闭工作簿，赋值为 False 时表示禁止关闭工作簿。

在 Workbook_BeforeClose 事件中，首先使用 For Each Next 循环遍历所有工作表，在循环体中使用 Range.Find 方法查找“合计”，如果未找到则继续查找下一个工作表，如果找到则记录工作表的名称，将它追加到变量 ShtName 中。当循环结束之后使用 Len 函数配合条件语句 If Then 判断变量 ShtName 的长度是否大于 0，如果大于 0 表示有一个或者多个表未汇总，则将参数 Canel 赋值为 True，表示禁止关闭工作簿，然后提示用户。

Workbook_BeforePrint 事件是工作簿事件，在打印工作表之前触发，参数 Cancel 用于控制是否允许打印工作表。本例中 Workbook_BeforePrint 事件的代码与 Workbook_BeforeClose 事件的路径完全一致，因此不再详解其思路。

本例文件参见光盘：..\第八章\8-11 未汇总则禁止关闭工作簿.xlsm

8.3.3　为所有工作表设计一个阅读模式

案例要求：当工作表中数据的行数与列数比较多时，活动单元格离行号与列标比较远，从而阅读工作表数据时不太方便。现要求创建一个阅读模式，在编辑单元格数据时保持 Excel 的默认状态即可，而需要查看数据时则进入阅读模式，使活动单元格所在行与列都突出显示，从而方便查看，避免看错行与列。

不能使用单元格的背景色或者条件格式来实现，这两种思路都会破坏工作表的原有格式。

实现步骤：

step 1　按<Alt+F11>组合键打开 VBE 界面，并单击菜单中的“插入”→“模块”命令。

step 2　在模块中录入以下过程代码：

```
Public YueDu As Boolean
Sub 阅读模式()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  YueDu = Not YueDu
End Sub
```

step 3 双击工程资源管理器中的 ThisWorkbook 进入工作簿事件代码窗口，然后分别录入以下两段事件过程代码：

```
'①代码存放位置：ThisWorkbook②随书光盘中有每一句代码的含义注释
Private Sub Workbook_Open()
  Application.OnKey "^Y", "阅读模式"
End Sub
Private Sub Workbook_SheetSelectionChange(ByVal Sh As Object, ByVal Target As
Range)
  If YueDu Then
    If Target.CountLarge = 1 Then
      Union(Target.EntireRow, Target.EntireColumn).Select
      Target.Activate
    End If
  End If
End Sub
```

step 4 保存并重启工作簿，然后任意选择单元格，可以发现工作表没有任意异常，和默认状态一致。

step 5 按<Ctrl+Shift+Y>组合键，然后再选择单个单元格，由于此时已经进入阅读模式，因此活动单元格所在的行与列都会采用更暗的背景色突出显示，从而便于阅读数据，效果如图 8.19 所示。

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	姓名	语文	数学	地理	化学	政治	法律	计算机	体育	几何	微积分	生物	哲学	历史
2	朱丽华	70	76	54	56	75	91	84	59	59	76	84	55	87
3	张坦然	99	53	95	94	76	98	98	89	90	73	87	85	70
4	张三月	90	74	78	93	78	56	100	81	67	87	73	79	79
5	范亚桥	95	88	84	77	75	86	88	88	91	56	86	77	85
6	陈新年	99	97	93	56	94	95	66	100	88	51	81	52	95
7	程前和	99	53	62	100	90	68	60	88	90	89	50	75	65
8	管语明	68	61	57	90	84	54	73	71	72	56	71	73	90
9	刘佩佩	96	100	69	72	98	51	94	69	57	77	62	85	58
10	罗新华	54	58	76	60	76	70	78	57	94	54	74	83	84
11	张世后	51	96	90	63	71	68	88	83	85	83	96	65	80

图 8.19 阅读模式效果

step 6 再次按<Ctrl+Shift+Y>组合键退出阅读模式，此时工作表将恢复常态。

思路分析：

本例所创建的阅读模式不是通过标示颜色来实现的，而是让活动单元格所在行与列变成选区，从而让它的背景色与其他区域形式反差来实现，因此开启阅读模式后就再也不能选择单个单元格。为了解决这个问题，本例采使用了一个公共变量 YueDu 来启用或者禁用阅读模式，当对它赋值为 True 时可进入阅读模式，当将它赋值为 False 时则退出阅读模式。

本例首先在模块中声明一个公共变量 YueDu，再在模块中使用一个“阅读模式”过程来改变该变量的值，从而控制阅读模式的启用与禁用。

为了更方便地调用过程“阅读模式”，本例通过 Workbook_Open 事件为该过程自动分配一个快捷键<Ctrl+Shift+Y>，从而在用户按下快捷键时可以让变量 YueDu 的值变成 True，再次按下快捷键时可以让变量 YueDu 的值变成 False，再次按下快捷键时又让变量的值变成 True……此设计可以让用户快速切换阅读模式。

当确定公共变量的值及修改变量的方法后，下一个重点就是通过事件让工作表进入阅读模式。本例采用的是 Workbook_SheetSelectionChange 事件，在事件中首先判断变量 YueDu 的值是否为 True，然后判断用户是否只选择了单个单元格，如果两个条件都符合，那么使用 Union 方法将活动单元格所在行与列合并为一个对象，然后使用 Range.Select 方法选中这个对象，此时工作表就进

入了阅读模式。

代码中“Target.Activate”的功能是纠正活动单元格对象，在事件过程中执行 Range.Select 方法不只是改变选区，活动单元格也被修改了，因此需要使用“Target.Activate”重置活动单元格。

语法补充：

（1）Application.OnKey 方法用于为过程指定快捷键，它的语法如下：

```
Application.OnKey(Key, Procedure)
```

其中 Key 参数代表快捷键名称，Procedure 参数代表过程名称，过程代码必须存放在模块中，否则按下指定的快捷键时无法调用该过程。

Key 参数是 Shift、Ctrl 和 Alt 这 3 个基本功能键与键盘上的其他键的组合，其中 3 个功能键的书写方式如表 8-2 所示。

表 8-2　功能键

功能键名称	VBA 符号
Shift	+
Ctrl	^
Alt	%

当快捷键中使用了字母键时要区分大小写，当 Key 参数了使用大写字母时，在调用过程时需要用 Shift 加字母键。例如本例中的 Key 参数使用了“^Y”，调用过程时不能用<Ctrl+Y>而是用<Ctrl+Shift+Y>。若对 Key 参数赋值为“^y”，那么调用过程时应使用<Ctrl+Y>。

如果需要取消已经为过程指定的快捷键，那么将 Application.OnKey 方法的第 2 参数赋值为空文本即可。例如要取消本例中过程“阅读模式”的快捷键应使用以下代码：

```
Application.OnKey "^Y", ""
```

（2）Range.Activate 方法用于激活一个单元格，使其成为活动单元格。当被激活的单元格处于选区之中时，激活单元格不会改变选区地址。在本例中，进入阅读模式之后，选区是 Target 所在行与列，而 Target 处于选区之间，因此代码“Target.Activate”可以激活行与列的重叠处，又不会改变选区地址。

本例文件参见光盘：..\第八章\ 8-12 阅读模式.xlsm

8.3.4　设计未启用宏就无法打开的工作簿

案例要求：当工作簿中有 VBA 代码特别是事件过程代码时，如果用户未启用宏则代码无法运行，无法实现预先设置的功能。现要求设计一个未启用宏就不能查看工作簿中任意数据、启用宏后数据就自动呈现出来的工作簿。当用户未启用宏时需要提示用户。

实现步骤：

step 1　新建一个空白工作簿，再新建若干个工作表，确保工作簿中不少于 3 个工作表，然后将第一个工作表重命名为“提示”。

step 2　将 B2:G7 单元格合并，然后在其中写入“请启用宏再开启本工作簿”。

step 3　将前 8 行及第 8 列以外的区域隐藏起来，然后将 A1:H8 区域设置背景色、边框，使其呈现出立体感。此步骤不是必要的，读者也可以跳过此步骤。

step 4 按<Alt+F11>组合键打开 VBE 窗口，然后双击 ThisWorkbook 事件。

step 5 在工作簿事件代码窗口中录入以下事件过程代码：

```
'①代码存放位置：ThisWorkbook②随书光盘中有每一句代码的含义注释
Private Sub Workbook_Open()
  Dim sht As Worksheet
  For Each sht In Worksheets
    If sht.Name <> "提示" Then sht.Visible = xlSheetVisible
  Next sht
  Worksheets("提示").Visible = xlSheetVeryHidden
End Sub
Private Sub Workbook_BeforeClose(Cancel As Boolean)
  Dim sht As Worksheet
  For Each sht In Worksheets
    If sht.Name = "提示" Then
      sht.Visible = xlSheetVisible
    Else
      sht.Visible = xlSheetVeryHidden
    End If
  Next sht
  ThisWorkbook.Save
End Sub
```

step 6 单击菜单中的“工具”→“VBAProject 属性”命令，然后进入“保护”选项卡，选中“查看时锁定工程”复选框，并且在下方输入密码，效果如图 8.20 所示（本案例文件的密码是 123456）。

图 8.20 为工程加密

step 7 单击菜单中的“开发工具”→“宏安全性”命令，然后在对话框中将“宏设置”选项设置为“禁用所有宏，并发出通知”。

step 8 保存工作簿，然后重启工作簿，此时在工作表上方将会出现“安全警告”，同时显示“提示”工作表，其他工作表全处于隐藏状态，效果如图 8.21 所示。

step 9 再次进入“宏设置”选项，将其设置为“启用所有宏”。

step 10 重启工作簿，会看到“提示”工作表处于隐藏状态，其他工作表则处于显示状态，效果如图 8.22 所示。

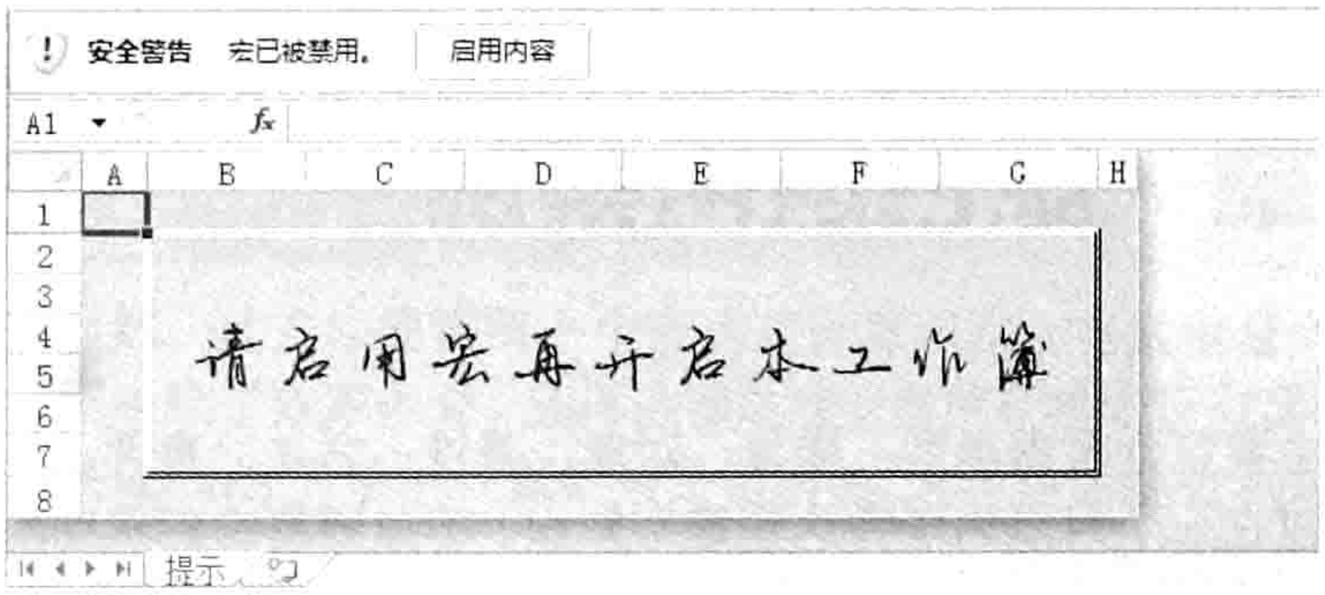


图 8.21　禁用宏后打开作簿时显示的页面

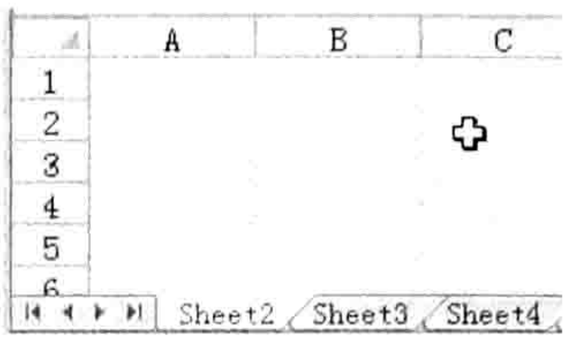


图 8.22　启用宏后显示的页面

思路分析：

当在工作簿中使用了宏代码而用户又未启用宏的前提下打开工作簿时，由于宏代码无法执行，这必定会影响制表的效率，以及预设的功能。为了强制用户启用宏，本例主要通过 3 个步骤实现。

其一是将第一个工作表命名为“提示”，并且在工作表中通过文字提示用户要启用宏。

其二是关闭工作簿时通过 Workbook_BeforeClose 事件将“提示”工作表显示出来，将其他工作表深度隐藏，同时对工程加密，防止用户修改代码。如此设置后，如果禁用宏再打开工作簿，那么只能看到“提示”工作表，其他工作表不能查看也不能取消隐藏。

其三是打开工作簿时通过 Workbook_Open 事件将“提示”工作表隐藏起来，将其他工作表显示出来，恢复正常的工作状态。

Workbook_Open 事件过程中隐藏“提示”工作表的代码的位置很重要，该语句必须放在循环语句之后，否则执行代码时会出错。

Excel 规定任何工作簿都必须至少有一个可见工作表，在执行循环语句之前唯一可见的工作表是“提示”，因此在循环语句之前不能将“提示”工作表隐藏起来。

本例文件参见光盘：..\第八章\ 8-13 启用宏才能开启的工作簿.xlsm

工作簿事件配合数组，以及 FSO 知识可以实现更强大的功能，在后面关于 FSO 和数组的章节还会提供工作簿事件的应用案例。

◆　◆　◆

第 9 章　综合应用案例

Excel VBA 的基础知识并不多，主要包含数据类型、变量、对象、属性、方法、事件、条件语句、循环语句、防错语句等，通常可以在一个月以内学完这些基础知识。不过要将它们熟练地组合起来解决工作中的疑难问题并不容易，需要勤加练习，多看他人成熟的代码，从中借鉴思路。

9.1　Application 应用案例

Application 对象代表 Excel 2010 应用程序，它包含 52 个方法、203 个属性和 43 个事件，本节针对其中常用的属性和方法展开案例演示。

9.1.1　计算字符表达式

案例要求：将如图 9.1 所示的工作表中文字表达式转换成值，计算结果存放在 C 列。

知识要点：Application.Evaluate 方法。

程序代码：

```
Sub 将表达式转换成值()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Dim rng As Range, Result
  If Cells(Rows.Count, 2).End(xlUp).Row < 2 Then Exit Sub
  For Each rng In Range("B2:B" & Cells(Rows.Count, 2).End(xlUp).Row)
    If Len(rng) > 0 Then
     Result = Application.Evaluate(rng.Value)
     If Not IsError(Result) Then rng.Offset(0, 1) = Result
    End If
  Next rng
End Sub
```

将以上代码存放在模块中，按<F5>键执行过程，在 C 列将产生如图 9.2 所示结果。

	A	B	C
1	品名	产品规格	体积
2	A	80*40*35	
3	B	75*60*30	
4	C	68*50*40	

图 9.1　产品规格

	A	B	C
1	品名	产品规格	体积
2	A	80*40*35	112000
3	B	75*60*30	135000
4	C	68*50*40	136000

图 9.2　将表达式转换成值

思路分析：

Application.Evaluate 方法用于将文本形式的表达式转换成值，不过它不能批量转换表达式，因此需要配合循环语句逐一计算表达式的值，然后将计算结果存放在右边的单元格中。

在循环语句中，尽管直接使用 Range("B2:B4")作为循环的对象也足以实现本例需求，不过代码的通用性比较差，当增减数据时需要修改代码才能确保不遗漏。为了提升代码的通用性，本例首先使用条件语句判断 B 列最后一个非空单元格的行号是否小于 2，如果小于 2 则直接结束过程。然后用 Range("B2:B" & Cells(Rows.Count, 2).End(xlUp).Row)作为循环的对象，其重点在于代码 Cells(Rows.Count, 2).End(xlUp).Row，即利用代码计算 B 列最后一个非空行的行号，而不是直接在代码中手工指定行号 4，它的优点是自动适应数据变化，不需要每增加一次数据就修改一次代码。

此外，为了提升程序的执行效率，本例代码采用了两条措施，其一是使用条件语句排除空白单

元格，节约 Evaluate 方法转换表达式的计算时间；其二是利用变量减少一半的计算次数。假设不使用变量 Result，那么应按以下方式编写代码：

```
If Not IsError(Application.Evaluate(rng.Value)) Then rng.Offset(0, 1) =
Application.Evaluate(rng.Value)
```

很显然，此方法需要每一个表达式计算两次，当数据量大时其劣势会相当明显。

编程时除了效率以外，还得注重防错。当单元格中没有字符或者单元格中的字符不是标准的表达式时，Evaluate 方法的计算结果是错误值，为了避免在 C 列单元格中产生错误值，本例采用 Iserror 函数排除了错误值。事实上也可以使用 IIF 语句替代本例的 If Then 语句：

```
rng.Offset(0, 1) = IIf(IsError(Result), "", Result)
```

语法补充：

（1）Application.Evaluate 方法用于将表达式转换为对象或者值，它的具体语法如下：

```
Application.Evaluate(Name)
```

其中参数 Name 可以是用户定义的名称，也可以是一个标准的表达式，如果不是标准的表达式则会返回错误值。标准表达式需要符合两个条件，其一是符合 Excel 对象的命名规则，例如“A1”、“Sheet1”、“B:B”；其二是符合数学运算的规则，例如“1+2”、“Sum(A1,B10,20)”等。“ABC”和“(2+1*8”就属于不标准的表达式，无法正确地转换成结果。

（2）IsError 是 VBA 函数，用于判断表达式是否为错误值。VBA 未提供判断表达式是否不是错误值的函数，因此 IsError 函数常搭配 Not 运算符使用。

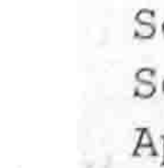
（3）Evaluate 可以单独使用，忽略其父对象 Application。

扩展应用：

本例中 B 列的产品规格是标准的表达式，如果是“长 80 宽 40 高 35”这种形式的不标准的表达式则应按以下方式修改代码：

```
Result = Application.Evaluate(Replace(Replace(Replace(rng.Value, "长", ""), "
宽", "*"), "高", "*"))
```

代码的含义是使用 Replace 函数将“长”替换成空文本，再将“宽”与“高”替换成乘号，从而转换成标准确的表达式，然后再通过 Evaluate 方法转换成计算结果。



本例文件参见光盘：..\第九章\ 9-1 计算字符表达式.xlsm

9.1.2　合并相同且相邻的单元格

案例要求：将任意列中已选中的相同并且相邻的单元格合并，合并时不能弹出提示框。

知识要点：Application.DisplayAlerts 属性、Application.Intersect 方法、Application.Selection 属性。

程序代码：

```
Sub 合并相同且相邻的单元格()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Dim rng As Range, PreviousRng As Range, TargetRng As Range
  If TypeName(Selection) <> "Range" Then Exit Sub
  If Selection.Columns.Count > 1 Then MsgBox "只能选择单列": Exit Sub
  Set TargetRng = Intersect(ActiveSheet.UsedRange, Selection)
  Set PreviousRng = TargetRng(1)
  Application.DisplayAlerts = False
  For Each rng In TargetRng.Offset(1, 0)
```

```
    If rng <> rng.Offset(-1, 0) Then
      Range(PreviousRng, rng.Offset(-1, 0)).Merge
      Set PreviousRng = rng
    End If
  Next
End Sub
```

选择如图 9.3 中所示的 A2:A13 区域，然后执行过程“合并相同且相邻的单元格”，程序会将选区中相同且相邻的单元格合并，效果如图 9.4 所示。

	A	B
1	省	市
2	福建省	彰州市
3	福建省	厦门市
4	福建省	三明市
5	云南省	昭通市
6	云南省	玉溪市
7	云南省	曲靖市
8	广东省	珠海市
9	广东省	中山市
10	广东省	肇庆市
11	江苏省	镇江市
12	江苏省	扬州市
13	江苏省	盐城市

图 9.3　省市列表

	A	B
1	省	市
2	福建省	彰州市
3		厦门市
4		三明市
5	云南省	昭通市
6		玉溪市
7		曲靖市
8	广东省	珠海市
9		中山市
10		肇庆市
11	江苏省	镇江市
12		扬州市
13		盐城市

图 9.4　合并相同且相邻的单元格

思路分析：

合并相同并且相邻的单元格主要针对单列的选区，因此在过程的开始需要先判断 Selection 是区域还是图形对象，以及选区的列数是否只有一列，不符合条件时就直接结束过程。

为了提升程序的执行效率，避免用户选择整列后再执行代码从而将时间浪费在大量的空白区域中，本例使用 Intersect 方法提取选区与已用区域的交集并赋值给变量 TargetRng，其后以 TargetRng 作为操作对象，从而可以忽略空白区域。然后将 TargetRng 向下偏移一行生成一个新的区域，通过循环语句遍历该区域，在循环语句中使用变量 rng 与它上一行单元格进行比较，如果相同则继续比较下一个单元格，如果不同则将它上方相同值的单元格合并。

如果合并非空单元格时会弹出提示框，导致程序中断，也影响代码的执行效率，因此有必要在循环语句之前关闭提示。

语法补充：

（1）Application.DisplayAlerts 属性用于控制执行宏时是否弹出警告框。当值为 True 时允许弹出警告框，否则禁止弹出警告框。在过程结束后，Application.DisplayAlerts 属性会自动恢复 True。调用 DisplayAlerts 属性时不可省略父对象 Application。

（2）Application.Intersect 方法用于提取至少两个区域的交集，只要涉及 Selection 对象的过程都应该使用 Application.Intersect 方法，避免用户选择了太大的区域时浪费执行时间。

（3）Application.Selection 属性代表用户当前选中的对象，可能是区域也可能是图形对象，因此调用之前有必要判断它的类型名称，如果 TypeName 的返回值为“Range”则说明是区域，如果返回值是“Shape”则说明是图形对象。调用 Selection 时允许忽略父对象 Application。

扩展应用：

本例的过程用于批量合并单元格，事实上也可以通过 VBA 将合并单元格批量取消合并，然后将合并时的值填充到取消合并后的所有单元格中去，代码如下：

```
Sub 取消合并单元格并填充()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Dim Rng As Range
  Application.FindFormat.Clear
  Application.FindFormat.MergeCells = True
  If TypeName(Selection) <> "Range" Then Exit Sub
  If Selection.MergeCells = False Then Exit Sub
```

```
    Do
      Set Rng = Selection.Find("", , , , , , , , , True)
      With Rng.MergeArea
        .UnMerge
        .Value = Rng.Value
        Set Rng = Selection.Find("", Rng, , , , , , , , True)
        If Rng Is Nothing Then Exit Do
      End With
    Loop
  End Sub
```

本例文件参见光盘：..\第九章\ 9-2 合并相同且相邻的单元格.xlsm

9.1.3　在指定时间提示行程安排

案例要求：在 13:30 时提示开会时间到，并在 10 分钟后关闭 Excel。

知识要点：Application.OnTime 方法、Application .Quit 方法、Application.StatusBar 属性、Application.Speech.Speak 方法。

程序代码：

```
Sub 添加计划任务()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Application.OnTime TimeValue("13:30:00"), "提示"
  Application.OnTime TimeValue("13:40:00"), "关闭工作簿"
End Sub
Sub 提示()
  Application.Speech.Speak "It's time for the meeting, please prepare"
  Application.StatusBar = "会议时间到，请准备。"
End Sub
Sub 关闭Excel()
  Application.DisplayAlerts = False
  Application.Quit
End Sub
```

执行过程“添加计划任务”，当到了 13:30 时会在 Excel 的状态栏提示“会议时间到，请准备”，同时 Excel 会朗读“It's time for the meeting, please prepare”，从而提示用户。

思路分析：

Application.OnTime 方法用于设置计划任务，指定某个时间段执行某个程序，本例中需要执行提示和关闭工作簿两个任务，因此需要使用 3 个 Sub 过程。

Application.OnTime 方法只能调用模块中的过程，因此“提示”和“关闭工作簿”不能保存在 ThisWorkbook 和工作表事件代码窗口中。

关闭所有工作簿用 Workbooks.Close 方法，而关闭应用程序应用 Application.Quit 方法。在关闭应用程序时，如果工作簿未保存则会弹出提示框，从而中断程序运行。将 Application.DisplayAlerts 赋值为 False 后再关闭应用程序则可以顺利完成。

语法补充：

（1）Application.OnTime 方法可以安排一个过程在将来的特定时间运行（既可以是具体指定的某个时间，也可以是指定的一段时间之后）。它的语法如下：

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

4 个参数的含义如表 9-1 所示。

表 9-1　OnTime参数详解

参数名称	必选/可选	参数说明
EarliestTime	必选	希望此过程运行的时间
Procedure	必选	要运行的过程名
LatestTime	可选	过程开始运行的最晚时间
Schedule	可选	如果为 true，则预定一个新的 OnTime 过程。如果为 false，则清除先前设置的过程

第 1 参数代表任务的执行时间，使用 Now + TimeValue(time)的形式可以安排程序在多少分钟、多少秒钟之后执行，若使用 TimeValue(time)形式则可安排程序在指定的时、分、秒执行；第 3 参数用于指定待运行的过程名称，在指定过程名称时必须使用引号。

如果需要中途取消已经设置的计划任务，那么可以将第 4 参数设为 False。例如以下过程可以取消本例所设置的“提示”任务：

```
Application.OnTime TimeValue("13:30:00"), "提示", False
```

（2）Application.Quit 方法用于退出 Excel，同时关闭工作簿。Workbooks.Close 方法可以关闭所有工作簿，但不关闭 Excel 程序。

（3）Application.Speech.Speak 方法用于朗读文本，执行此代码需要确保计算机的声卡驱动已经正常安装好，同时计算机已经连接好音响设备。Application.Speech.Speak 方法的语法如下：

```
Application.Speech.Speak (Text, SpeakAsync, SpeakXML, Purge)
```

其中第 1 参数 text 是需要朗读的文本，可以是中文也可以是英文，其他 3 个参数是可选参数，通常只写第 1 参数即可。

如果用户的操作系统是 Windows XP 的，那么 Application.Speech.Speak 方法只能朗读英文，需要打开以下链接地址安装中文语音引擎才能朗读中文。若是 Windows 7 或者 Windows 8 则可以直接阅读中文短句，系统自带中文语音引擎：

http://pan.baidu.com/s/1kT9Jxu3。

扩展应用：

如果要求 10 分钟后执行过程“提示”，那么应按如下方式编写代码：

```
Application.OnTime Now + TimeValue("00:10:00"), "提示"
```

本例文件参见光盘：..\第九章\ 9-3 定时执行程序.xlsm

9.1.4　模拟键盘快捷键打开高级选项

案例要求：利用代码打开 Excel 2010 的高级选项。

知识要点：Application.SendKeys 方法。

程序代码：

```
Sub 打开高级选项()'①代码存放位置：模块中
 Application.SendKeys "%to{DOWN 5}"
End Sub
```

在工作表界面按<Alt+F8>组合键打开“宏”对话框，选择“打开高级选项”命令然后执行过程，程序立即会开启 Excel 的高级选项。功能等同于<Alt+D+O+↓+↓+↓+↓+↓>组合键。

思路分析：

在 Excel 2010 中使用<Alt+T+O>组合键可以打开 Excel 选项对话框，再按 5 次<↓>键可以打开高级选项。而 Application.SendKeys 方法可以模拟键盘实现这 8 个按钮的同等效果，因此本例代码使用了 Application.SendKeys 方法向 Excel 发送“%to{DOWN 5}”，其中%符号代表按<Alt>键，而{DOWN 5}代表按 5 次<↓>键。

必须在 Excel 界面执行以上代码才生效，不能在 VBE 界面中执行过程。

语法补充：

（1）Application.SendKeys 方法可以将一个或多个按键消息发送到活动窗口，模仿键盘操作。其语法如下：

```
Application.SendKeys(Keys, Wait)
```

第 1 参数 Keys 表示要发送的按键消息，可以是键盘上的任意键，允许是组合键；第 2 参数 Wait 是可选参数，如果赋值为 True 则 Excel 会等到处理完按键后再执行后面的代码，默认值是 False，表示不等候按键是否处理完成，可以同步执行其他代码。

利用 Application.SendKeys 方法发送字母或者数字时，直接对 Keys 参数赋值为字母或者数字并加引号即可，如果是<F1>、<Alt>、<Insert>、<Enter>等键则必须按 Excel VBA 预设的代码编写才能正常调用，具体请以“Application.SendKeys 方法”为关键字查询 VBA 帮助，其中有详细介绍。

（2）Application.SendKeys 方法只能发送键盘上的键，无法发送汉字。例如向 A1 单元格中发送“中国”是无法实现的，但发送“china”则可以。

```
Sub 向A1输入china()'①代码存放位置：模块中
  Range("a1").Select
  Application.SendKeys "china~"
End Sub
```

代码中的参数 Keys 赋值为“china~”表示先输入单词 china，然后按<Enter>键。

扩展应用：

Excel 2003 提供了打开“多重合并计算数据区域”的透视表向导，而从 Excel 2007 开始不再提供该菜单，是否能用 VBA 直接启动此向导呢?答案是使用 VBA 发送<Alt+D+P>的快捷键即可，代码如下：

```
Sub 打开透视表向导()'①代码存放位置：模块中
 Application.SendKeys "%dp"
End Sub
```


本例文件参见光盘：..\第九章\9-4 模拟键盘快捷键打开高级选项.xlsm

9.1.5　使用快捷键合并与取消单元格

案例要求：设置两个快捷键，分别用于合并单元格与取消单元格合并，同时要做到合并单元格时不产生提示框，取消合并单元格后可以将合并状态下的值填充到合并区域中。

知识要点：Application.OnKey 方法、Application.FindFormat 属性、Application.DisplayAlerts 属性、Application.Selection 属性。

程序代码：

```
Sub 合并()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  If TypeName(Selection) = "Range" Then
```

```
    Application.DisplayAlerts = False
    Selection.Merge
    Selection.HorizontalAlignment = xlCenter
  End If
End Sub
Sub 取消合并()
  Dim Rng As Range
  Application.FindFormat.Clear
  Application.FindFormat.MergeCells = True
  If TypeName(Selection) <> "Range" Then Exit Sub
  If Selection.MergeCells = False Then Exit Sub
  Do
    Set Rng = Selection.Find("", , , , , , , , , True)
    With Rng.MergeArea
      .UnMerge
      .Value = Rng.Value
      Set Rng = Selection.Find("", Rng, , , , , , , , True)
      If Rng Is Nothing Then Exit Do
    End With
  Loop
End Sub
Sub Auto_open()
  Application.OnKey "^m", "合并"
  Application.OnKey "^M", "取消合并"
End Sub
```

将以上代码存放在模块中，然后保存并重启工作簿。

假设工作簿中有如图 9.5 所示的数据，需要合并 A2:A4、C2:C4 和 E2:E4 这 3 个区域，那么首先选中这 3 个区域，然后按<Ctrl+M>组合键执行合并，合并结果如图 9.6 所示。如果此时按<Ctrl+Shift+M>组合键，那么如图 9.6 所示的数据将会还原到如图 9.5 所示的状态。

	A	B	C	D	E	F
1	省	市	省	市	省	市
2	福建省	彰州市	云南省	昭通市	广东省	珠海市
3	福建省	厦门市	云南省	玉溪市	广东省	中山市
4	福建省	三明市	云南省	曲靖市	广东省	肇庆市

图 9.5　待合并数据

	A	B	C	D	E	F
1	省	市	省	市	省	市
2	福建省	彰州市	云南省	昭通市	广东省	珠海市
3		厦门市		玉溪市		中山市
4		三明市		曲靖市		肇庆市

图 9.6　合并效果

思路分析：

如果本例的需求只是区域合并与取消合并，那么采用 Selection.Merge 与 Selection.UnMerge 两句代码即可，而取消合并后要填充合并状态下的值则比较麻烦，无法一步完成，除非选区中只有一个区域。

为了提升程序的通用性，编写代码时应该考虑单个区域与多个区域的环境。本例在取消合并单元格时使用了 Do Loop 循环语句搭配 Range.Find 方法从 Selection 对象中查找合并单元格，然后逐一取消合并并且填充内容。

在合并单元格时，尽管使用 Selection.Merge 方法即可合并单个或者多个区域，然而它无法处理这 3 个问题：Selection 对象不一定是单元格，合并单元格时会弹出提示框，Range.Merge 方法只能合并单元格却不能让单元格的值居中显示。所以本例代码中加入了条件判断语句、关闭提示及居中显示的语句，从而让程序更完善。

过程 Auto_open 的功能是为“合并”与“取消合并”两个过程指定快捷键。

过程命名为 Auto_open 的目的是开启工作簿时自动执行，不用再手工指定快捷键。过程中的“^m”和“"^M”分别代表<Ctrl+M>和<Ctrl+Shift+M>两个组合键。

语法补充：

（1）Application.OnKey 用于为过程指定快捷键，其语法如下：

```
Application.OnKey(Key, Procedure)
```

第 1 参数 Key 表示按键组合，第 2 参数表示过程名称，两个参数都必须带引号。

事实上 Application.OnKey 方法还可以为过程指定内置的快捷键，例如<Ctrl+C>组合键用于复制数据，如果要将该键与过程“合并”绑定，那么可用以下代码实现：

```
Application.OnKey "^c", "合并"
```

如果要取消<Ctrl+C>组合键与过程“合并”的绑定，有两种办法实现，一是重启工作簿，二是对 Application.OnKey 方法的第 2 参数赋值为空文本，代码如下：

```
Application.OnKey "^c", ""
```

（2）Application.OnKey 方法除了可以为过程指定快捷键以外，也可以用于屏蔽内置的快捷键。例如<Ctrl+V>是粘贴数据的组合键，使用以下代码可以使<Ctrl+V>组合键失效，不过代码仅在重启工作簿之前有效，重启工作簿后该快捷键会自动恢复。

```
Application.OnKey "^v", ""
```

OnKey 方法不能单独使用，不能忽略其前置对象 Application。

扩展应用：

Application.OnKey 方法用于为过程指定快捷键，事实上也可以利它修改内置的快捷键。例如<Ctrl+C>组合键已经绑定到 Excel 内置的复制功能，如果要禁用<Ctrl+C>，改用<Ctrl+Shift+C>组合键复制数据，那么代码如下：

```
Sub 快捷键()'①代码存放位置：模块中
  Application.OnKey "^c", "快捷键"
  Application.OnKey "^C", "复制"
End Sub
Sub 复制()
  Selection.Copy
End Sub
```

第一句 Application.OnKey 可禁用<Ctrl+C>组合键，第二句 Application.OnKey 则为复制操作指定了新的组合键<Ctrl+Shift+C>。

本例文件参见光盘：..\第九章\ 9-5 使用快捷键合并与取消单元格.xlsm

9.1.6　查找至少两月未付货款的客户名称

案例要求：图 9.7 中的 B 列是发货时间，D 列是收款时间。现要求开启工作簿时自动选中超过两个月未收到款项的客户，同时将客户名称显示在状态栏。

知识要点：Application.Union 方法、Application.StatusBar 属性。

程序代码：

```
'①代码存放位置：ThisWorkbook②随书光盘中有每一句代码的含义注释
Private Sub Workbook_Open()
  Dim rng As Range, TargatRng As Range, NameStr As String
  For Each rng In Range("b2:b" & Cells(Rows.Count, 2).End(xlUp).Row)
    If DateDiff("m", rng.Value, Date) > 2 And Len(rng.Offset(0, 2)) = 0 Then
```

```
    If TargatRng Is Nothing Then
      Set TargatRng = rng
    Else
      Set TargatRng = Application.Union(rng, TargatRng)
    End If
    NameStr = NameStr & "," & rng.Offset(0, -1).Value
  End If
Next rng
If Not TargatRng Is Nothing Then
  TargatRng.Offset(0, -1).Select
  Application.StatusBar = "超过两个月未付款的客户：" & NameStr
  End If
End Sub
```

将以上代码保存在 ThisWorkbook 窗口中，然后重启工作簿，假设今天是 6 月 6 日，那么“货款工作表”中两个超过两月未付款的客户名称所在单元格将自动呈选中状态，同时会在状态栏里显示客户名称，效果如图 9.8 所示。

	A	B	C	D
1	公司	发货时间	货款	收款时间
2	正大集团	3月1日	52428	
3	天信鞋业	3月1日	8952	3月23日
4	朝明镙丝厂	3月5日	35978	4月5日
5	洪锋鞋材厂	3月25日	58480	3月30日
6	珠海新立五金厂	3月28日	10115	
7	越福塑胶公司	4月2日	36160	5月1日
8	福泰轮胎厂	5月1日	36044	5月29日
9	金六福酒厂	5月4日	59710	
10	天心礼帽公司	6月1日	16634	6月6日
11	正太服份有限公司	6月6日	58475	

图 9.7　货款表

	A	B	C	D
1	公司	发货时间	货款	收款时间
2	正大集团	3月1日	52428	
3	天信鞋业	3月1日	8952	3月23日
4	朝明镙丝厂	3月5日	35978	4月5日
5	洪锋鞋材厂	3月25日	58480	3月30日
6	珠海新立五金厂	3月28日	10115	
7	越福塑胶公司	4月2日	36160	5月1日
8	福泰轮胎厂	5月1日	36044	5月29日
9	金六福酒厂	5月4日	59710	
10	天心礼帽公司	6月1日	16634	6月6日
11	正太服份有限公司	6月6日	58475	

货款工作表

超过两个月未付款的客户：正大集团,珠海新立五金厂

图 9.8　选中超过两月未付款的客户

思路分析：

本例过程首先使用 For Each Next 循环语句遍历 B 列的所有发货时间，在循环体中使用 DateDiff 函数计算每一个发货时间与今天的间隔月数，如果间隔月数大于 2 且发货时间右边两列是空白的，那么使用 Application.Union 方法将发货时间左边一列的公司所在单元格合并为一个 Range 对象，同时将公司名称串连成一个字符串。

待循环完成后，选中符合条件的公司所在单元格，同时将公司名称显示在状态栏。

本例的循环对象采用的 Range("b2:b" & Cells(Rows.Count, 2).End(xlUp).Row)，而非 Range("b2:b11")，此代码的优点是可以适应数据的增减变化，用代码计算最后一个非空行的行号，而不是在代码中手工指定行号。

语法补充：

（1）Application.Union 方法返回两个或多个区域的合集，如果没有合集则返回 Nothing。

Application.Union 方法无法合并跨表的区域，例如执行以下代码必定会产生错误：

```
Sub 求合集()'代码存放位置：模块中
MsgBox Application.Union(Sheets(1).[a1], Sheets(2).[d1:g2]).Address
End Sub
```

Union 可以单独使用，忽略其前置对象 Application。

（2）Application.StatusBar 属性代表应用程序的状态栏，可以对它随意赋值，不过状态栏可以显示的字符长度有限制。

扩展应用：

状态栏中可以显示指定的字符，若配合循环语句则可以让状态栏显示滚动文字。例如：

```
Sub 状态栏()'代码存放位置：模块中
  Application.StatusBar = "兴趣是最好的老师"
    Do
    For i = 1 To 10000
      DoEvents
    Next
    Application.StatusBar  =  Right(Application.StatusBar,  Len(Application.
StatusBar) - 1) & Left(Application.StatusBar, 1)
  Loop
End Sub
```

代码中 10000 用于调节滚动速度，值越大滚动得越慢。

本例文件参见光盘：..\第九章\9-6 在状态栏显示超期未付款的客户名称.xlsm

9.2　Range 对象应用案例

单元格对象是数据的基本载体，是 Excel 制表工作中接触最多的一个对象，它的类别名称是 Range。本节针对 Range 对象的常用属性和方法展开案例演示。

9.2.1　合并工作表

案例要求：工作簿中有若干个工作表，现要求将所有工作表的数据合并到“总表”中，合并时需将原来的公式转换成值，而且合并后要确保格式、列宽与合并前一致，如图 9.9 所示。

H2　=AVERAGE(B2:G2)

	A	B	C	D	E	F	G	H
1	姓名	语文	数学	地址	历史	化学	生物	平均成绩
2	黄淑宝	42	79	82	60	45	81	64.8
3	潘大旺	40	100	88	93	68	72	76.8
4	刘晶扬	94	44	87	94	84	44	74.5
5	周少强	95	90	83	97	42	98	84.2
6	石开明	86	44	49	59	50	54	57.0
7	刘越堂	69	98	76	96	70	83	82.0
8	陈玲	91	43	45	50	61	49	56.5
9	胡秀文	42	67	42	58	99	96	67.3
10	钱光明	89	43	44	56	92	97	70.2

一班　二班　三班

图 9.9　成绩表

知识要点：Range.Copy 方法、Range.PasteSpecial 方法、Range.Offset 属性、Range.End 属性。

程序代码：

```
Sub 合并成绩到总表()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
Dim sht As Worksheet, i As Byte
Application.ScreenUpdating = False
On Error Resume Next
Application.DisplayAlerts = False
Worksheets("总表").Delete
Worksheets.Add.Name = "总表"
For Each sht In Worksheets
  If sht.Name <> "总表" Then
    If WorksheetFunction.CountA(sht.Range("A:A")) > 0 Then
      i = i + 1
      If i = 1 Then
        sht.UsedRange.Copy
```

```
            Range("a1").PasteSpecial xlPasteAllUsingSourceTheme
            Range("a1").PasteSpecial xlPasteValues
            Range("a1").PasteSpecial xlPasteColumnWidths
          Else
            sht.UsedRange.Offset(1, 0).Copy
            With Cells(Rows.Count, 1).End(xlUp).Offset(1, 0)
              .PasteSpecial xlPasteAllUsingSourceTheme
              .PasteSpecial xlPasteValues
              .PasteSpecial xlPasteColumnWidths
            End With
          End If
        End If
      Next sht
      Application.ScreenUpdating = True
    End Sub
```

思路分析：

合并工作表其实就是将多个工作表的数据复制到同一个工作表中，在复制过程会频繁更新屏幕，为了提升代码的执行效率，在循环语句之前将 Application.ScreenUpdating 属性赋值为 False，从而关闭屏幕更新，待循环完成后再恢复更新。

在合并工作表前，需要确保工作簿中有“总表”，否则合并会出错。本例采用的办法是先删除“总表”，然后再新建一个“总表”，此思路比先判断是否存在“总表”，然后根据判断结果决定处理方式的代码简单得多。不过当工簿中没有“总表”时代码会出错，因此需要在代码中插入防错语句。

在启动循环语句后，首先使用条件语句排除“总表”，然后再排除 A 列是空白的工作表，接着将其他工作表的已用区域复制到“总表”中。

在复制工作表的数据时，由于第一个工作表才有必要复制标题，因此需要借助一个变量 i 来判断 Sht 属于第几个工作表。对于第一个工作表，将它的已用区域复制到“总表”的 A1 单元格即可，对于其他工作表则应该只复制排除标题行之后的已用区域，由于本例标题行只有一行，所以复制对象是“UsedRange.Offset(1, 0)”。

在复制数据后，本例对每一个工作表都粘贴了 3 次，第 1 次是粘贴全部（xlPasteAllUsingSourceTheme），即数据与格式信息，第 2 次是粘贴值（xlPasteValues），第 3 次只粘贴列宽。第 2 次粘贴的目的在于将公式转换成值，避免合并前后的公式结果不一致。例如合并前的工作表的 A2 单元格中有公式“=row()”，它的计算结果 2，而合并后公式存放在“总表”的 A100 单元格中，此时公式的结果将变成 100，不再是原值 2。

第 3 次粘贴的目的在于确保合并前后单元格的列宽一致。第 1 次粘贴时尽管是粘贴全部，其实只是包含单元格的数据和格式，并不包含列宽，为了将合并后的“总表”与合并前的工作表的列宽保持一致，必须单独粘贴一次列宽。

语法补充：

（1）Range.Copy 方法表示复制单元格或者区域，它有两种用法，其一是通过参数指定粘贴时的目标地址，从而将单元格或者区域复制到指定的单元格中，例如：

Range("a1:b2").Copy Range("d1")——表示将 A1:B2 区域复制到 D1:E2 区域中。书写代码时允许只写目标区域的左上角单元格。

其二是忽略参数，直接将单元格或者区域复制到剪贴板中，然后再配合 Range.PasteSpecial 方法粘贴数值或者粘贴格式、粘贴列宽、粘贴公式等。

（2）Range.PasteSpecial 方法表示选择性粘贴，其语法如下：

```
Range.PasteSpecial(Paste, Operation, SkipBlanks, Transpose)
```

其中 Range 代表粘贴时的目标单元格，4 个参数的含义见表 9-2。

表 9-2　Range.PasteSpecial方法的参数说明

参数名称	功能描述
Paste	要粘贴的区域部分，例如粘贴格式、粘贴值、粘贴列宽等
Operation	粘贴操作，包含加、减、乘、除与无操作 5 个选项
SkipBlanks	如果赋值为 True 表示不将剪贴板上区域中的空白单元格粘贴到目标区域中，默认值为 False
Transpose	如果赋值为 True 表示在粘贴区域时转置行和列，默认值为 False

其中 Paste 参数包含 12 个选项，Operation 参数包含 5 个选项，不过对于它们的书写方式与含义不必要花时间记，在编程前可以通过录制宏产生代码，然后将宏代码复制到自己的过程中即可。

扩展应用：

Range.Copy 方法可以将一个区域复制到剪贴板中，粘贴后仍然是一个区域。如果需求将区域中的值合并成单个字符串再放入剪贴板，粘贴时只粘贴在单个单元格中，那么应改用 DataObject 对象和 PutInClipboard 方法。操作步骤如下。

step 1　在 A1:A4 区域中随意输入字符，然后按<Alt+F11>组合键进入 VBE 界面。

step 2　单击菜单中的“工具”→“引用”命令，然后在引用窗口中将“Microsoft Forms 2.0 Object Library”打钩（也可以单击菜单中的“插入”→“用户窗体”命令，从而自动添加引用，然后再删除窗体）。

step 3　插入一个模块，然后在模块中录入以下代码：

```
Sub 将A1到A4的值复制到粘贴板中()'代码存放位置：模块中
  Dim MyData As DataObject, rng As Range, Mystr As String
  Set MyData = New DataObject
  For Each rng In Range("a1:a4")
  Mystr = Mystr & rng.Value
  Next rng
  MyData.SetText Mystr
  MyData.PutInClipboard
End Sub
```

step 4　返回工作界面，执行过程“将 A1 到 A4 的值复制到粘贴板中”，此时 A1:A4 中的值已经被复制到剪贴板中，可以在任意单元格粘贴 A1:A4 中的值。

本例文件参见光盘：..\第九章\ 9-7 合并工作表.xlsm

9.2.2　合并区域且保留所有数据

案例要求：将当前选区合并再居中显示，同时保留合并前的所有数据。

知识要点：Range.Merge 方法、Range.Areas 属性、Range.HorizontalAlignment 属性、Range.ClearContents 方法、Range.Value 属性。

程序代码：

```
Sub 合并区域且保留所有值()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Dim Rng As Range, MyStr As String, i As Byte
  If TypeName(Selection) <> "Range" Then Exit Sub
```

```
For i = 1 To Selection.Areas.Count
  With Selection.Areas(i)
    MyStr = ""
    For Each Rng In .Cells
      MyStr = MyStr & Rng
    Next Rng
    .ClearContents
    .Merge
    .HorizontalAlignment = xlCenter
    .Value = MyStr
  End With
Next i
End Sub
```

假设工作表中有如图 9.10 所示的数据，同时选中 A1:C1、A3:B3 和 A5:C5 区域，然后执行过程“合并区域且保留所有值”，3 个区域将会分别合并，并且合并区域时保留合并前的所有数据，效果如图 9.11 所示。

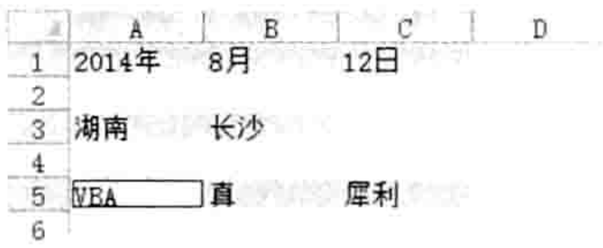


图 9.10　合并前

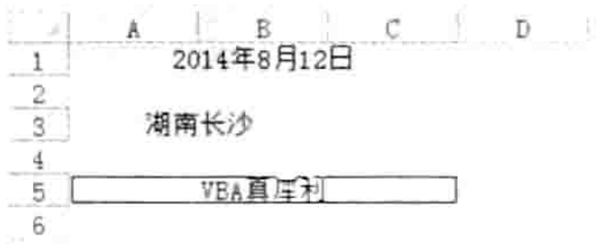


图 9.11　合并后

思路分析：

Selection 可能是区域也可能是图形对象，合并单元格仅对区域生效，因此在过程开始时需要使用条件语句排除图形对象，当前选择的对象是图形时直接结束过程即可。

当前选区可能包含多个不相邻的区域，为了提升程序的通用性，本例使用循环语句遍历所有区域，然后再嵌套一个循环语句遍历区域中的每个单元格，将它们合并成一个字符串并赋值给变量 MyStr。接着清除区域中的值、合并区域、将区域设置为居中显示、将变量 MyStr 的值赋予合并后的单元格。

当外层的循环结束后，每一个区域都会变成合并单元格，并且显示合并前的所有字符。

过程“合并区域且保留所有值”中有两个重点，其一是变量 Mystr 用于储存区域的所有字符，当里层循环结束后变量 MyStr 将包含第 i 个区域的值，为了避免合并下一个区域时变量 MyStr 中还保留了当前区域的值，从而影响下一个区域中的值，在进入里层循环之前必须清除变量 MyStr 的值。

第二个重点是合并区域时会不会弹出提示信息是由区域中是否有值决定的，本例中由于合并区域前已经将所有值记录在变量 MyStr 里，所以可以先清除区域中的值再合并，那么 Excel 不再弹出提示框，因此本例不需要使用代码 Application.DisplayAlerts = False。

语法补充：

（1）Range.Merge 方法用于合并区域，如果区域中超过一个单元格有值将会弹出提示框。Range.Merge 方法只负责合并区域，不会让单元格的值居中显示，因此它总配合 Range.HorizontalAlignment 属性使用。Range.Merge 方法的语法如下：

```
Range.Merge(Across)
```

如果参数 Across 赋值为 True，则将区域中每一行合并一次，区域中有多少行就合并成多少个区域，相当于 Excel 内置的“跨越合并”功能。Across 参数的默认值是 False，表示一个区域只合并一次，不管多少行都合并为一个区域。

（2）Range.Areas 属性代表区域集合，相邻的单元格为一个区域，图 9.10 中包含 3 个区域。

Range.Areas 属性其实也是 Range 对象，不过它的单位是区域，所以使用索引号引用子集时不是单个单元格，而是单个区域。

扩展应用：

图 9.12 中 A1:B13 区域中包含省市名称，要求将它合并为 D1:E13 所示的结果。代码如下：

```
Sub 跨越合并且保留所有值()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Dim RowCount As Integer, ColCount As Integer, i As Integer, Mystr As String
  If TypeName(Selection) <> "Range" Then Exit Sub
  For i = 1 To Selection.Areas.Count
    For RowCount = 1 To Selection.Areas(i).Rows.Count
      Mystr = ""
      For ColCount = 1 To Selection.Areas(i).Columns.Count
        Mystr = Mystr & Selection.Areas(i).Cells(RowCount, ColCount)
      Next ColCount
      Selection.Areas(i).Rows(RowCount).ClearContents
      Selection.Areas(i).Cells(RowCount, 1) = Mystr
    Next RowCount
    Selection.Areas(i).Merge True
    Selection.Areas(i).HorizontalAlignment = xlCenter
  Next i
End Sub
```

	A	B	C	D	E
1	省	市		省市	
2	福建省	彰州市		福建省彰州市	
3	福建省	厦门市		福建省厦门市	
4	福建省	三明市		福建省三明市	
5	云南省	昭通市		云南省昭通市	
6	云南省	王溪市		云南省王溪市	
7	云南省	曲靖市		云南省曲靖市	
8	广东省	珠海市		广东省珠海市	
9	广东省	中山市		广东省中山市	
10	广东省	肇庆市		广东省肇庆市	
11	江苏省	镇江市		江苏省镇江市	
12	江苏省	扬州市		江苏省扬州市	
13	江苏省	盐城市		江苏省盐城市	

图 9.12　跨越合并且保留所有值

本例文件参见光盘：..\第九章\ 9-8 合并选区且保留所有值.xlsm

9.2.3　合并计算多区域的值

案例要求：图 9.13 中包含 4 个产品组别的产量，要求对所有产品分类汇总。

知识要点：Range.Consolidate 方法、Range.CurrentRegion 属性、Range.Borders 属性。

程序代码：

```
Sub 合并计算()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
   Range("m1").Consolidate  Array("R1C1:R9C2",  "R1C4:R9C5",  "R1C7:R9C8",
"R1C10:R9C11"), xlSum, True, True, False
   Range("m1") = "产品"
   Range("m1").CurrentRegion.Borders.LineStyle = xlContinuous
End Sub
```

执行以上过程后将得到如图 9.14 所示的计算结果。

	A	B	C	D	E	F	G	H	I	J	K
1	A组产品	数量		B组产品	数量		C组产品	数量		D组产品	数量
2	测压器	82		老虎钳	77		电笔	77		直尺	77
3	主板诊断卡	23		梅花刀	63		六角螺丝	63		扳手	63
4	梅花刀	63		压线钳	66		改锥	66		圆形螺丝	66
5	一字刀	97		圆形螺丝	43		温度计	43		压线钳	43
6	电笔	32		美工刀	44		直尺	44		美工刀	44
7	六角螺丝	95		圆规	82		主板诊断卡	82		一字刀	82
8	改锥	20		扳手	21		美工刀	21		老虎钳	21
9	温度计	99		改锥	81		一字刀	81		测压器	40

图 9.13　产量表

M	N
产品	数量
测压器	122
直尺	121
主板诊断卡	105
老虎钳	98
梅花刀	126
一字刀	260
电笔	109
六角螺丝	158
压线钳	109
圆形螺丝	109
美工刀	109
圆规	82
扳手	84
改锥	167
温度计	142

图 9.14　合并计算结果

程序代码：

本例过程首先利用 Range.Consolidate 方法将 A1:B9、D1:E9、G1:F9、J1:K9 4 个区域进行合并计算，合并计算时首行与最左列作为标题行，不参与运算，计算方式是求和。由于合并计算结果没有每列列标题和边框，因此在合并计算后手工添加列标题和边框。

Range.Consolidate 方法的第 1 参数必须是数组，VBA 中使用 Array 函数生成数组。在本书的第 13 章中会详细介绍更多关于数组的知识。

语法补充：

（1）Range.Consolidate 方法可以将多个区域的数据分类汇总，汇总方式包括平均、计数、只计数数值、最大值、最小值、乘、基于样本的标准偏差、基于全体数据的标准偏差、总计、未指定任何分类汇总函数、基于样本的方差、基于全体数据的方差。它的语法如下：

```
Range.Consolidate(Sources, Function, TopRow, LeftColumn, CreateLinks)
```

5 个参数的含义如表 9-3 所示。

表 9-3　Range.Consolidate方法参数详解

参数名称	功能描述
Sources	以文本引用字符串数组的形式给出合并计算的源，该数组采用 R1C1 样式表示法。这些引用必须包含将要合并计算的工作表的完整路径
Function	用于指定合并计算的类型，包含 12 种计算方式，详见表 9-4
TopRow	如果为 true，则基于合并计算区域中首行内的列标题对数据进行合并。如果为 false，则按位置进行合并计算。默认值为 false
LeftColumn	如果为 true，则基于合并计算区域中左列内的行标题对数据进行合并计算。如果为 false，则按位置进行合并计算。默认值为 false
CreateLinks	如果为 true，则让合并计算使用工作表链接。如果为 false，则让合并计算复制数据

其中第 1 参数 Sources 必须使用 R1C1 样式的单元格地址，例如 A1:B9 区域改用 R1C1 样式后为 R1C1:R9C2，即为“第 1 行第 1 列:第 9 行第 2 列”。

其中第 2 参数 Function 包含 12 种计算方式，可选范围由表 9-4 中所示的 12 个常量决定。

表 9-4　汇总方式详解

常量名称	功能描述	参数名称	功能描述
xlAverage	平均	xlStDev	基于样本的标准偏差
xlCount	计数	xlStDevP	基于全体数据的标准偏差

续表

常量名称	功能描述	参数名称	功能描述
xlCountNums	只计数数值	xlSum	总计
xlMax	最大值	xlUnknown	未指定任何分类汇总函数
xlMin	最小值	xlVar	基于样本的方差
xlProduct	乘	xlVarP	基于全体数据的方差

Range.Consolidate 方法的语法，以及 Function 参数的常量名称都不必要记，需要时录制宏即可自动产生代码。

（2）Range.Borders 对象代表单元格的边框集合。Borders.LineStyle 属性则代表边框的线型，表 9-5 中是代表线型的常量及功能说明。

表 9-5　边框边线列表

常量名称	功能描述	常量名称	功能描述
xlContinuous	实线（单线）	xlDot	点式线
xlDash	虚线	xlDouble	双线
xlDashDot	点画相间线	xlLineStyleNone	无线条
xlDashDotDot	画线后跟两个点	xlSlantDashDot	倾斜的画线

扩展应用：

Range.Consolidate 方法也支持跨工作表或者跨工作簿合并计算。假设要对 A 组、B 组、C 组、D 组的 A1:B9 区域合并计算，将结果存在“分类汇总”工作表的 A1 单元格中，代码如下：

```
Sub 合并计算()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Worksheets("分类汇总").Range("A1").Consolidate Array("A 组!R1C1:R9C2", "B
组!R1C1:R9C2", "C 组!R1C1:R9C2", "D 组!R1C1:R9C2"), xlSum, True, True, False
  Worksheets("分类汇总").Range("A1") = "产品"
  Worksheets("分类汇总").Range("A1").CurrentRegion.Borders.LineStyle = 1
End Sub
```

本例文件参见光盘：..\第九章\9-9 合并计算.xlsm

9.2.4　模糊查找公司名称并罗列出来

案例要求：在活动工作表中精确查找名字长度为 2 并且第一个字是“天”的公司，然后将查找结果存放在 F 列和 G 列中。

知识要点：Range.Copy 方法、Range.Resize 属性、Range.Offset 属性、Range.Find 方法。

程序代码：

```
Sub 模糊查找公司名称并罗列出来()'①代码存放位置：模块中②随书光盘中有每一句代码含义注释
  Dim Rng As Range, FirstAddress As String
  Set Rng = Columns("A:D").Find("天?公司", , , xlPart)
  If Not Rng Is Nothing Then
    FirstAddress = Rng.Address
    Range("A1:B1").Copy Range("F1")
    Do
```

```
        Rng.Resize(1, 2).Copy Cells(Rows.Count, 6).End(xlUp).Offset(1, 0)
        Set Rng = Columns("A:D").FindNext(Rng)
        If Rng.Address = FirstAddress Then Exit Do
      Loop
    End If
End Sub
```

当工作表有图 9.15 中 A1:D7 区域所示的数据时，执行过程“模糊查找公司名称并罗列出来”后得到 F1:G3 区域所示的结果。

	A	B	C	D	E	F	G
1	公司	电话	公司	电话		公司	电话
2	福春公司	33776390	宏运企业	20818931		天宏公司	24002049
3	彰化公司	41204934	天宏公司	24002049		天信公司	85365201
4	柳州化工厂	28023755	福兴钢铁厂	93517381			
5	福缘制造厂	20944756	尊明企业	80774419			
6	天信公司	85365201	天龙兴公司	50004542			
7	龙锋企业	61292960	龙华胶水厂	67219661			

图 9.15　查找并罗列在 F 列和 G 列

思路分析：

Range.Find 方法支持通配符“?”和“*”，前者代表单个任意字符，后者代表任意长度的任意字符。本例要查找名字长度为 2 并且第一个字是“天”的公司，因此将“天?公司”设置为 Range.Find 方法的查找对象。

Range.Find 方法一次只能查找一个目标，因此需要配合 Do Loop 循环语句使用，每找到一个目标就使用 Range.Resize 方法引用该单元格及其右边的电话号码，然后使用 Range.Copy 方法将它们复制到 F 列。

由于 Do Loop 循环配合 Range.Find 方法执行查找时会一直循环下去，为了让它在查找完一遍后自动停止，特意在 Do Loop 循环之前将第一次找到的目标单元格地址记录在变量 FirstAddress 中，然后在 Do Loop 循环之中比较每一次查找到的目标单元格的地址是否与变量 FirstAddress 一致，如果一致则结束循环，避免在 F 列产生重复的公司名称。

语法补充：

Range.Resize 用于重置区域大小，它的两个参数分别代表重置后的区域高度与宽度。本例中 Range.Find 的查找目标是公司名称，而要复制的目标是公司加电话，因此需要将变量 Rng 重置为 1 行 2 列后再复制。

扩展应用：

假设要求在 E1 单元格中输入查找条件，按<Enter>键后自动启动查找过程，并且将结果罗列在 F 列和 G 列中，那么应用以下方式修改代码：

```
'①代码存放位置：工作表事件代码窗口②随书光盘中有每一句代码含义注释
Private Sub Worksheet_Change(ByVal Target As Range)
  Dim Rng As Range, FirstAddress As String
  If Target.Count > 1 Then Exit Sub
  If Target.Address <> "$E$1" Then Exit Sub
  Set Rng = Columns("A:D").Find(Target.Value, , , xlPart)
  If Not Rng Is Nothing Then
    FirstAddress = Rng.Address
    Range("A1:B1").Copy Range("F1")
    Do
      Rng.Resize(1, 2).Copy Cells(Rows.Count, 6).End(xlUp).Offset(1, 0)
      Set Rng = Columns("A:D").FindNext(Rng)
      If Rng.Address = FirstAddress Then Exit Do
    Loop
End If
```

```
End Sub
```

本例文件参见光盘：..\第九章\9-10 模糊查找.xlsm

9.2.5　反向选择单元格

案例要求：选择当前工作表中已用区域的反向区域，即选择未被选中的区域。

知识要点：Range.Select 方法、Range.Address 属性、Range.CountLarge 属性。

程序代码：

```
Sub 反向选择()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  If TypeName(Selection) <> "Range" Then Exit Sub
  Dim SelectionAddress As String, UsedRangeAddress As String, FanXiang As String,
rng As Range
  If Selection.CountLarge = Cells.CountLarge Then Exit Sub
  If IsEmpty(ActiveSheet.UsedRange) Then MsgBox "反向选择仅对数据区域生效": Exit
Sub
  Set rng = Intersect(Selection, ActiveSheet.UsedRange)
  If rng Is Nothing Then MsgBox "反向选择仅对数据区域生效": Exit Sub
  If rng.Address = ActiveSheet.UsedRange.Address Then ActiveSheet.UsedRange.
Select: Exit Sub
  SelectionAddress = rng.Address
  UsedRangeAddress = ActiveSheet.UsedRange.Address
  Application.ScreenUpdating = False
  With Worksheets.Add
     .Range(UsedRangeAddress) = 0
     .Range(SelectionAddress) .ClearContents
     FanXiang = .Range(UsedRangeAddress).SpecialCells(xlCellTypeConstants, 1).
Address
     Application.DisplayAlerts = False
     .Delete
  End With
  ActiveSheet.Range(FanXiang).Select
  Application.ScreenUpdating = True
End Sub
```

在图 9.16 中，当前选区为 B2:B9 和 D2:D9 区域，执行过程“反向选择”后选区会变成 A1:A9、B1、C1:C9、D1，效果如图 9.17 所示。

	A	B	C	D
1	A组产品	数量	B组产品	数量
2	测压器	82	老虎钳	77
3	主板诊断卡	23	梅花刀	63
4	梅花刀	63	压线钳	66
5	一字刀	97	圆形螺丝	43
6	电笔	32	美工刀	44
7	六角螺丝	95	圆规	82
8	改锥	20	扳手	21
9	温度计	99	改锥	81

图 9.16　选择 B3:B6 区域

	A	B	C	D
1	A组产品	数量	B组产品	数量
2	测压器	82	老虎钳	77
3	主板诊断卡	23	梅花刀	63
4	梅花刀	63	压线钳	66
5	一字刀	97	圆形螺丝	43
6	电笔	32	美工刀	44
7	六角螺丝	95	圆规	82
8	改锥	20	扳手	21
9	温度计	99	改锥	81

图 9.17　反向选择区域

思路分析：

反向选择是针对当前选择的区域而言的，而且选区必须与工作表的已用区域存在交集，而且该交集必须小于活动工作表的已用区域，否则反向选择就没有存在的意义。

本例代码使用了 5 个条件语句，将不符合条件的情况排除在外，然后再新建辅助工作表，在该表中计算反向区域的地址，最后以该地址为依据选择活动工作表中的反向区域。

5 个条件语句中需要特别说明的是判断用户是否全选了工作表，过程中的判断依据是“Selection.CountLarge = Cells.CountLarge”，其中 Range.CountLarge 属性代表单元格数量，功能上等同于 Range.Count 属性，不过 Range.Count 属性的值是 Long 型，适用于 Excel 2003，而在 Excel 2010 中单元格的总数量远远大于 Long 型的上限，因此微软为 VBA 新开发了一个属性——Range.CountLarge，此属性增大了储存空间，专用于 Excel 2007 及以上版本。在 Excel 2010 中编写代码计算单元格的数量时，如果需要强调兼容性与通用性，那么可用 Range.Count 属性，而强调代码的准确性，确保代码不会出错则应使用 Range.CountLarge 属性。

在计算反向区域时，本例的思路是：先记录下选区与已用区域的交集的地址，并且存放于变量 SelectionAddress 中，以及记录下已用区域的地址并且存放于变量 UsedRangeAddress 中，然后新建一个工作表，在新表中向地址等于 SelectionAddress 的区域写入数值 0，将新表中地址等于 UsedRangeAddress 的区域清除 0 值，那么剩下的 0 值所在的区域即为反向区域。此时使用 Range.SpecialCells 方法引用常量区域（即 0 值所在的区域），然后记录下它的地址，删除辅助工作表，返回原来的工作表中，使用 Range.Select 方法选中已经记录下的反向区域即可。

本例的重点有两个，一是使用条件语句排除不符合条件的情况，二是借助辅助工作表识别反向区域。

语法补充：

（1）Range.Select 方法用于选择 Range 对象。可以是单个单元格，也可以是单个区域或者多个区域。它有别于 Range.Activate 方法，Range.Activate 方法只能激活单个单元格。

（2）Range.Address 属性用于获取单元格的地址，它的语法如下：

```
Range.Address(RowAbsolute, ColumnAbsolute, ReferenceStyle, External, RelativeTo)
```

5 个参数皆为可选参数，当忽略所有参数时表示 A1 样式的绝对引用。各参数含义如表 9-6 所示。

表 9-6　Range.Address属性参数含义列表

名称	描述
RowAbsolute	如果为 True，则以绝对引用返回引用的行部分。默认值为 True
ColumnAbsolute	如果为 True，则以绝对引用返回引用的列部分。默认值为 True
ReferenceStyle	代表引用样式，可能是 x1A1 也可能是 xlR1C1。默认值为 xlA1
External	如果为 True，则返回外部引用。如果为 False，则返回本地引用。默认值为 False
RelativeTo	如果 RowAbsolute 和 ColumnAbsolute 为 False，并且 ReferenceStyle 为 xlR1C1，则必须包括相对引用的起始点。此参数是定义起始点的 Range 对象

（3）Range.CountLarge 属性可返回 Range 对象的单元格数量，仅用于 Excel 2007 及以上版本（Excel 的帮助中关于 Range.CountLarge 属性的含义解释是错误的）。

扩展应用：

Range.Select 仅对活动工作表有效，不能选择其他工作表的单元格。例如活动工作表是 Sheet1，那么执行以下代码必定出错：

```
Worksheets("sheet2").Range("a100").Select
```

跨表选择单元格应使用 Application.Goto 方法，代码如下：

```
Sub 跨表选择单元格()'代码存放位置：模块中
  Application.Goto Worksheets("sheet2").Range("a100")
End Sub
```

本例文件参见光盘：..\第九章\9-11 反向选择.xlsm

9.2.6　插入图片并调整为选区大小

案例要求：插入一张产品图片，并且图片刚好适应选区。

知识要点：Range.Top 属性、Range.Left 属性、Range.Height 属性、Range.Width 属性。

程序代码：

```
Sub 导入图片且等于选区大小()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  If TypeName(Selection) <> "Range" Then Exit Sub
  Dim Filenname As String
  Filenname = Application.GetOpenFilename("所有图片文件 (*.jpg;*.bmp;*.png;
*.gif),*.jpg;*.bmp;*.png;*.gif", , "请选一个图片文件", , False)
  If Filenname= "False" Then Exit Sub
  ActiveSheet.Shapes.AddPicture Filenname, False, True, Selection.Left,
Selection.Top, Selection.Width, Selection.Height
End Sub
```

执行以上过程会弹出一个选择图片的对话框，其支持 jpg、bmp、png 和 gif 共 4 种格式的图片。当选择图片并且单击“打开”按钮后，程序会将选择的图片导入到工作表中，而且图片的大小、边距刚好适应当前区域。例如选择了如图 9.18 所示的区域再执行代码，图片将显示为如图 9.19 所示的效果。

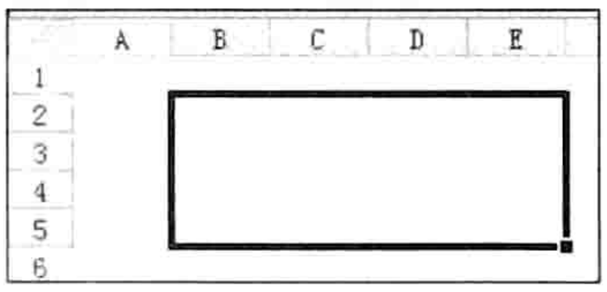


图 9.18　选择区域

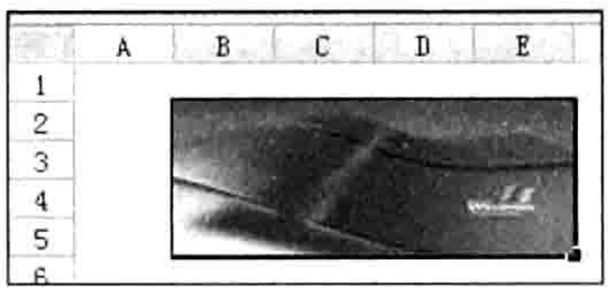


图 9.19　插入图片覆盖选区

思路分析：

插入图片并且显示为选区的大小，那么必须确保 Selection 对象是单元格，而不是图形对象，因此本例首先使用条件语句判断 Selection 对象的类型，如果不是单元格则结束过程。然后使用 GetOpenFilename 方法创建一个打开图片文件的对话框让用户选择图片，由于用户可能选择了图片也可能单击了“取消”按钮，因此有必要使用条件语句判断返回值是否为“False”，如果是则结束过程，否则调用 Shapes.AddPicture 方法插入图片，并且根据选区的左边距、上边距、宽度与高度调整图片。

过程中代表图片路径的变量 Filenname 被声明为 String 型，当用户在对话框中单击“取消”按钮时其返回值是文本“False”，如果单击的是“打开”按钮则返回值是图片路径，因此根据返回值的差异可以简单而准确地区分用户的按键行为。

语法补充：

（1）Range.Top 属性用于获取或设置单元格的上边距，以磅为单位；而 Range.Left 属性则用于获取或设置单元格的左边距，以磅为单位。当 Range 对象包含多个单元格时，以左上角单元格作为计算依据。

（2）Range.Width 属性用于获取一个区域的宽度，以磅为单位（VBA 帮助中关于 Range.Width 属性的解释是错的，它是可读属性，只能读取值，不能修改）；Range.Height 则用于获取或者设置一个区域的高度，以磅为单位。当 Range 对象包含多个区域时仅对第一个区域有效。

（3）Shapes.AddPicture 方法用插入图片，其语法如下：

```
Shapes.AddPicture(Filename, LinkToFile, SaveWithDocument, Left, Top, Width,
Height)
```

7个参数都是必选参数，它们的含义参见表9-7。

表9-7 Shapes.AddPicture方法的参数一览

参数名称	功能说明
Filename	图片路径
LinkToFile	赋值为True时表示向工作表中插入图片链接，赋值为False时表示将图片文件嵌入到文件中
SaveWithDocument	赋值为True时表示将图片与文档一起保存，赋值为False时表示不保存图片，当删除硬盘中的图片后工作表中将不再显示图片
Left	图片左上角相对于文档左上角的位置（以磅为单位）
Top	图片左上角相对于文档顶部的位置（以磅为单位）
Width	图片的宽度（以磅为单位）
Height	图片的高度（以磅为单位）

其中LinkToFile和SaveWithDocument不允许同时赋值为False。当SaveWithDocument参数为False、LinkToFile参数赋值为True时不能在工作表中嵌入图片文件，删除磁盘中的图片文件后工作表中的图片会消失；当SaveWithDocument参数为True时，不管LinkToFile参数的值是什么都可以在工作表中嵌入图片文件，删除磁盘中的图片文件后工作表中仍然会显示图片。

录制插入图片的宏时，宏代码会调用Pictures.Insert方法来插入图片，本例中不使用Pictures.Insert方法插入图片是因为它有一个显著的缺点——删除磁盘中的图片文件后，插入到工作表中的图片会自动消失。

扩展应用：

插入单个图片并且指定其边距与大小并不能体现VBA的优势。

Shapes.AddPicture方法配合循环语句可以从活动单元格开始向单元格中批量插入图片，并且让图片的位置和大小随单元格变化，完整代码如下：

```
Sub 批量导入图片且指定高度高度()'①代码存放位置：模块中②随书光盘中有每一句代码含义注释
  If TypeName(Selection) <> "Range" Then Exit Sub
  Dim Filenname, shpName, i As Integer, shp As Shape
  Filenname = Application.GetOpenFilename("所有图片文件 (*.jpg;*.bmp;*.png;
*.gif),*.jpg;*.bmp;*.png;*.gif", , "请选一个图片文件", , True)
  If TypeName(Filenname) = "Boolean" Then Exit Sub
  For Each shpName In Filenname
    Set shp = ActiveSheet.Shapes.AddPicture(shpName, msoFalse, msoTrue,
ActiveCell.Offset(i, 0).Left, ActiveCell.Offset(i, 0).Top, ActiveCell.
Offset(i, 0).Width, ActiveCell.Offset(i, 0).Height)
    shp.Placement = xlMoveAndSize
    shp.Name = Dir(shpName)
    i = i + 1
  Next shpName
End Sub
```

选择A1单元格，然后执行以上过程，当弹出“请选择所有待插入的图片文件”对话框后，在图片文件夹中按住鼠标左键不放并拖动从而选择所有图片，效果如图9.20所示。

当单击“打开”按钮后，当前选择的所有图片会插入到工作表中，从活动单元格开始向下一字

排开，每个图片的位置和大小随单元格而定，效果如图 9.21 所示。

选择 1～12 行，然后修改行高，A1:A12 区域中的图片也会相应地调整高度，效果如图 9.22 所示。

图 9.20　选择需要插入的图片

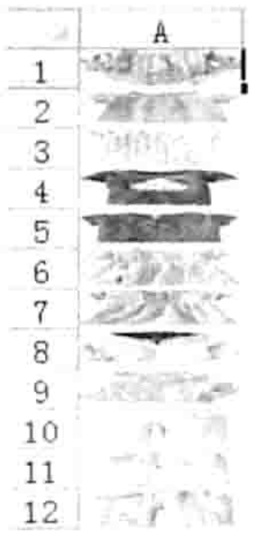


图 9.21　插入图片

图 9.22　修改行高

本例文件参见光盘：..\第九章\9-12 插入图片且调整为选区的大小.xlsm

9.2.7　提取唯一值

案例要求：图 9.23 为参赛人员名单，其中部分人员参加了多个项目的比赛。现需要提取参赛人员名单的唯一值，将结果罗列在 C 列中。

知识要点：Range.RemoveDuplicates 方法、Range.Value 属性。

程序代码：

```
Sub 提取唯一值1()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  With Range("C1:C" & Cells(Rows.Count, 1).End(xlUp).Row)
    .Value = Range("A1:A" & Cells(Rows.Count, 1).End(xlUp).Row).Value
    .RemoveDuplicates 1, xlYes
  End With
End Sub
```

执行以上过程，在 C 列将产生具有唯一性的参赛名单，效果如图 9.24 所示。

	A	B
1	姓名	参赛项目
2	赵月峨	长跑
3	范亚桥	田径
4	朱文道	跳高
5	张中正	跳水
6	张秀文	举重
7	张秀文	田径
8	范亚桥	长跑
9	张彻	射击
10	张中正	长跑
11	陈年文	跳水

图 9.23　参赛者列表

	A	B	C
1	姓名	参赛项目	姓名
2	赵月峨	长跑	赵月峨
3	范亚桥	田径	范亚桥
4	朱文道	跳高	朱文道
5	张中正	跳水	张中正
6	张秀文	举重	张秀文
7	张秀文	田径	张彻
8	范亚桥	长跑	陈年文
9	张彻	射击	
10	张中正	长跑	
11	陈年文	跳水	

图 9.24　获取唯一值

思路分析：

Range.RemoveDuplicates 方法可以从区域中提取唯一值，结果只能存放在原区域，本例需要

将结果存放在另一列中，因此只能先将 A 列的姓名复制到 C 列，然后再提取唯一值。

语法补充：

（1）Range.RemoveDuplicates 方法用于从区域中删除唯一值，允许设置多条件，即多列同时相同才算重复，也可以指定其中某一列相同就算重复。它的语法如下：

```
Range.RemoveDuplicates(Columns, Header)
```

其中两个可选参数的含义如表 9-8 所示。

表 9-8　Range.RemoveDuplicates方法参数详解

参数名称	功能描述
Columns	用于指定以哪一列或者哪几列作为判断是否重复的依据。当以单列作为判断依据时赋值为代表列数的数值即可，当以多列作为判断依据时应赋值为数组。如果忽略参数则表示每一列都参与运算，作为判断重复的依据
Header	指定第一行是否参与运算，赋值为 xlNo 时表示不参与运算，赋值为 xlYes 时表示参与运算，赋值为 xlGuess 时表示让 Excel 自己判断。默认值是 xlNo

Range.RemoveDuplicates 方法对应于功能区的“数据”选项卡中的“删除重复项”功能，可以通过录制宏得到代码，因此不必记忆语法或者参数名称，需要时录制宏即可产生代码，然后根据需求修改宏代码中的区域即可。

Range.RemoveDuplicates 方法仅用于 Excel 2007 及以上版本，不支持 Excel 2003。

（2）“Range.Value=Range.Value”形式用于将一个区域的值复制到另一个区域，它只复制数值不会复制公式和格式信息。

扩展应用：

Range.RemoveDuplicates 方法提取的唯一值只能存放在原位置，若改用高级筛选提取唯一值则可以将结果存放在任意位置，包括跨工作表或者跨工作簿存放。

仍以取 A 列的唯一值并存在 C 列为例，完整代码如下：

```
Sub 提取唯一值 2()'代码存放位置：模块中
 Range("A1:A"      &      Cells(Rows.Count,      1).End(xlUp).Row).AdvancedFilter
xlFilterCopy, , Range("C1"), True
End Sub
```

以上过程中 Range.AdvancedFilter 方法表示高级筛选，第 1 参数赋值为 xlFilterCopy 表示将结果复制到其他区域，而第 3 参数则用于指定结果存放区域，允许该区域来自其他工作表或者其他工作簿。

本例文件参见光盘：..\第九章\9-13 提取唯一值.xlsm

9.2.8　隐藏所有公式结果为错误的单元格

案例要求：图 9.25 中所示的部分公式的运算结果为错误值，现要求将所有错误隐藏起来，让单元格显示空白即可，但同时要保留公式。

知识要点：Range.Font 属性、Range.Interior 属性、Range.NumberFormatLocal 属性。

程序代码：

```
Sub 隐藏所有错误值()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Dim rng As Range
```

```
  For Each rng In ActiveSheet.UsedRange.SpecialCells(xlCellTypeFormulas, 16)
    rng.Font.Color = rng.Interior.Color
    rng.NumberFormatLocal = "[黑色]G/通用格式"
    rng.Errors(1).Ignore = True
  Next
End Sub
```

执行以上过程后，计算结果为错误值的公式将自动隐藏起来，效果如图 9.26 所示。

C3 =RANK(B3,B$2:B$11)

	A	B	C	D
1	姓名	成绩	排名	
2	赵	42	7	
3	钱	①	#N/A	
4	孙		#N/A	
5	李	94	1	
6	周	65	5	
7	吴	90	3	
8	郑	56	6	
9	王	93	2	
10	冯		#N/A	
11	陈	86	4	

图 9.25　公式中的错误值

C3 =RANK(B3,B$2:B$11)

	A	B	C	D
1	姓名	成绩	排名	
2	赵	42	7	
3	钱			
4	孙			
5	李	94	1	
6	周	65	5	
7	吴	90	3	
8	郑	56	6	
9	王	93	2	
10	冯			
11	陈	86	4	

图 9.26　隐藏错误值

思路分析：

隐藏计算结果为错误值的公式比较简单，将字体颜色设置为与单元格背景颜色一致即可。不过此操作会有后遗症——修改被公式引用的单元格的值从而使公式的计算结果不再是错误值后，公式的计算结果仍然不会显示出来。本例的办法是将单元格的数字格式自定义为“[黑色]G/通用格式”，表示单元格中的值不是错误时，字体颜色将显示为黑色。

由于不同单元格的背景颜色可能不一致，因此不能一次性设置错误值所在单元格的字体颜色和数字格式，必须使用循环语句。

语法补充：

（1）Range.NumberFormatLocal 属性可以获取或设置一个单元格的数字格式编码。修改单元格的数字格式只能修改单元格的显示值，不影响单元格的实际值。当单元格中的值是“2014-9-9”时，通过以下代码可以将它的显示值修改为“星期二”：

```
Range("A1").NumberFormatLocal = "AAAA"
```

自定义单元格的数字格式可以通过录制宏产生代码，因此不必记忆各种格式所对应的格式编码，需要编写自定义格式的代码时，录制宏即可。

（2）Range.Errors 代表单元格中的错误对象集合，Error.Ignore 属性用于设置错误检查选项的状态，将它赋值为 True 时表示禁用错误检查选项，即关闭单元格中的绿色倒三角符号。

（3）Range.Font 代表单元格的字体对象，Range.Font.Color 属性则代表字体颜色；Range.Interior 代表单元格的内部，Range.Interior.Color 属性则代表单元格的背景颜色。

扩展应用：

本例代码的功能是修改错误值的显示状态。

如果仅仅是要求打印时忽略错误值，而不修改单元格的显示状态，那么应使用以下代码修改活动工作表的页面设置：

```
Sub 不打印错误值()'代码存放位置：模块中
   ActiveSheet.PageSetup.PrintErrors = xlPrintErrorsBlank
End Sub
```

本例文件参见光盘：..\第九章\9-14 隐藏所有错误值.xlsm

9.3 Comment 对象应用案例

Comment 对象即批注，在工作表中应用极为广泛，常用于对单元格数据做补充说明，也可用于储存图片。本节展示 4 个 VBA 操作批注的综合应用案例。

9.3.1 在所有批注末尾添加指定日期

案例要求：图 9.27 中有 4 个批注，现要求在所有批注末尾添加日期。

知识要点：Comment.Text 方法、Comment.Shape.TextFrame.AutoSize 属性。

程序代码：

```
Sub 在批注中添加日期()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Dim DateStr As String, Com As Comment
  Do
    DateStr = Application.InputBox("请指定一个日期：", "日期", Date, , , , , 2)
    If IsDate(DateStr) Then Exit Do Else MsgBox "只能输入日期", vbInformation
  Loop
  For Each Com In ActiveSheet.Comments
    Com.Text Text:=Com.Text & Chr(10) & DateStr
    Com.Shape.TextFrame.AutoSize = True
  Next
End Sub
```

执行以上过程会弹出如图 9.28 所示的输入框，其中默认值为当前的系统日期。如果在其中输入了非日期值则程序会提醒用户只能输入日期，然后继续弹出输入框，直到输入日期为止。

图 9.27　成绩表

图 9.28　指定日期

如果使用默认日期，直接单击“确定”按钮即可，程序执行完成后会在活动工作表的所有批注的下一行追加日期，而且批注框会随内容的增减而自动调整大小，效果如图 9.29 所示。

如果需要将日期添加到第一行，其中修改批注内容的代码按以下方式修改即可。

```
Com.Text Text:=DateStr & Chr(10) & Com.Text
```

修改后的代码执行效果如图 9.30 所示。

图 9.29　将日期添加到原有批注后

图 9.30　将日期添加到原有批注前

思路分析：

使用 Application.InputBox 方法可以弹出输入框让用户输入日期，不过一个完善的程序还应考虑防错和便捷性，因此本例代码中加入了条件语句和循环语句。条件语句的功能是防错，用于判断用户输入的值是否为日期，当用户输入的不是日期时则提示用户；循环语句的功能是当用户输入错

误时可以再次弹出输入框等待用户重新输入，不用手工重新执行代码。

当用户输入了正确的日期后，则通过 For Each Next 循环语句遍历活动工作表中的批注集合 Comments，然后通过 Comment.Text 方法将换行符 Chr(10)与日期追加到每一个批注末端。

当对批注中添加新的内容后，应调整批注框的大小，避免批注内容显示不完整。正确的做法不是将批注框拉大，而是将它的 AutoSize 属性赋值为 True，使其自动调整大小。

语法补充：

（1）Comments 是批注集合，Comment 则是批注的类别名称，因此声明变量用于遍历批注集合时应将变量的类型声明为 Comment。

（2）Comment.Text 方法用于设置批注的文本，其语法如下：

```
Comment.Text(Text, Start, Overwrite)
```

3 个参数含义如表 9-9 所示。

表 9-9　Comment.text方法参数详解

参数名称	功能描述
Text	要添加的文本
Start	要插入文本的起始位置，只有第 3 参数赋值为 False 时才使用本参数。如果省略第 2、第 3 参数则会覆盖所有批注内容
Overwrite	省略此参数时表示覆盖批注内容，赋值为 False 时表示向批注中插入文本，不允许赋值为 True

在 VBA 帮助中关于第 3 参数 Overwrite 的解释是错的，正确的说法见表 9-9。

例如 A1 的批注内容是“我要学 VBA”，那么执行以下代码后批注内容是“Excel”：

```
Range("a1").Comment.Text "Excel"
```

如果改用以下代码，则执行过程后批注内容是“我要学 Excel”：

```
Range("a1").Comment.Text "Excel", 4
```

如果改用以下代码，则执行过程后批注内容是“我要学 ExcelVBA”：

```
Range("a1").Comment.Text "Excel", 4, False
```

（3）Shape.TextFrame.AutoSize 属性代表图形对象的大小是否随文本字符的长短自动调整大小。所有可以编辑字符的图形对象都有此属性，例如批注、文本框、形状（也称自选图形）。

扩展应用：

如果需要使用<Ctrl+Q>组合键为活动单元格的批注添加日期，代码如下：

```
Sub Auto_open()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Application.OnKey "^q", "为当前批注添加日期"
End Sub
Sub 为当前批注添加日期()
  If ActiveCell.Comment Is Nothing Then
    MsgBox "活动单元格不存在批注"
  Else
    ActiveCell.Comment.Text ActiveCell.Comment.Text & Chr(10) & Date
    ActiveCell.Comment.Shape.TextFrame.AutoSize = True
  End If
End Sub
```

本例文件参见光盘：..\第九章\9-15 批量为批注添加日期.xlsm

9.3.2 生成图片批注

案例要求：给单元格插入一个图片批注，图片由用户随意选择。

知识要点：Comment.Shape.Fill.UserPicture 方法。

程序代码：

```
Sub 插入图片标注()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Dim Pic As String, com As Comment
  Pic = Application.GetOpenFilename("图片文件,*.jpg; *.bmp;*.png", , "请选择一
张图片", , False)
  If Pic = "False" Then Exit Sub
  If ActiveCell.Comment Is Nothing Then Set com = ActiveCell.AddComment Else
Set com = ActiveCell.Comment
  com.Shape.Fill.UserPicture Pic
  com.Shape.Height = 100
  com.Shape.Width = 120
End Sub
```

执行过程“插入图片批注”，在弹出的对话框中选择一个图片，单击“打开”按钮后，如果活动单元格已经有批注则将选择的图片填充到批注中，批注的原本内容保持不变；如果活动单元格中没有批注，那么会插入一个新的批注，然后将图片填充到批注中。

代码中的 100 和 120 分别表示批注框的高度和宽度，可以根据需求随意修改。如果将 100 改为 160，那么最后产生的效果如图 9.31 所示。

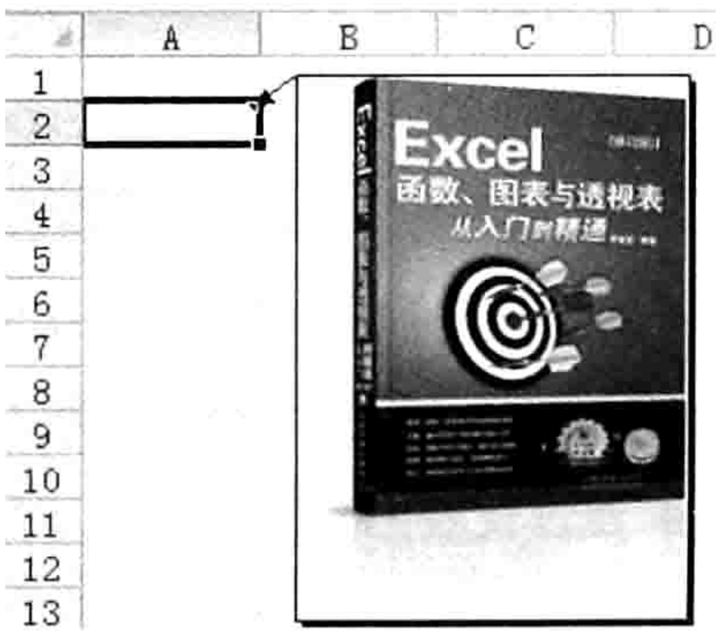


图 9.31　插入图片批注

思路分析：

本例过程首先通过 Application.GetOpenFilename 方法创建一个打开图片文件的对话框，然后通过条件语句判断用户是否选择了图片文件并选择“打开”按钮，如是不是则直接结束过程，如果是则再利用条件语句判断活动单元格是否有批注，当没有批注时会新建一个批注，最后将前面所选择的图片通过 Shape.Fill.UserPicture 方法填充到批注中去。最后为批注指定高度与宽度，使批注中的图片看起来不致于变形。

判断单元格是否存在批注和判断工作簿中是否存在某个工作表的思路大大不同，引用一个不存在的工作表时代码会出错，因此可以根据代码是否出错来判断工作表是否存在；引用批注时，如果批注不存在则返回 Nothing，因此判断批注是否存在主要通过“Is Nothing”的返回值决定，返回为 True 则表示批注不存在。

语法补充：

（1）Comment.Shape 是一个 Shape 对象，代表批注的外框。批注这个对象没有高度、宽度和背景填充这些属性，因此调整批注的大小和填充背景图片要借助 Comment.Shape 实现。

（2）Fill.UserPicture 方法的功能是将图片填充为背景，其参数是图片路径，可以是 bmp、jpg、png 或者 gif 等格式的图片文件，但将动画文件填充为背景后不会产生动画效果。

扩展应用：

如果将本例中 Application.GetOpenFilename 方法的最后一个参数赋值为 True，然后再配合循环语句就可以实现批量插入图片批注，完整代码如下：

```
Sub 批量生成图片批注()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  If TypeName(Selection) <> "Range" Then Exit Sub
  Dim Filenname, shpName, i As Integer, shp As Shape
  Filenname = Application.GetOpenFilename("所有图片文件 (*.jpg;*.bmp;*.png;
*.gif),*.jpg;*.bmp;*.png;*.gif", , "请选择所有待插入的图片文件", , True)
  If TypeName(Filenname) = "Boolean" Then Exit Sub
  Application.ScreenUpdating = False
  For Each shpName In Filenname
    If ActiveCell.Offset(i, 0).Comment Is Nothing Then Set com = ActiveCell.
Offset(i, 0).AddComment Else Set com = ActiveCell.Offset(i, 0).Comment
    com.Shape.Fill.UserPicture shpName
    com.Shape.Height = 100
    com.Shape.Width = 120
    i = i + 1
  Next shpName
  Application.ScreenUpdating = True
End Sub
```

本例文件参见光盘：..\第九章\9-16 生成图片批注.xlsm

9.3.3　添加个性化批注

案例要求：生成具有个性化外观的批注，提供多种形状可供选择。

知识要点：Range.AddComment 方法、Comment.Shape.AutoShapeType 属性、Comment.Delete 方法。

程序代码：

```
Sub 添加个性化批注()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
    Dim mystr As String, mystr2 As String, Com As Comment
    ActiveCell.ClearComments
    mystr = Application.InputBox("输入批注内容", "批注", Application.UserName, 10,
10, , , 2)
    mystr2 = Application.InputBox("输入批注外形" & Chr(10) & "1 口哨型,2 书卷型,3
箭头型,4 圆角矩形" & Chr(10) & "5 缺角矩形,6 菱形,7 五角星,8 云形标注,9 圆形,10 六边形,11
八边形,12 柱形,13 笑脸形,14 心形,15 八角星,16 横卷形,17 竖卷形,18 波形,19 双波形,20 十六角
星,21 二十四角星,22 文档.", "批注外型", 1, 10, 10, , , 1)
    With ActiveCell.AddComment(mystr)
      Select Case mystr2
      Case 1
      .Shape.AutoShapeType = msoShapeFlowchartSequentialAccessStorage
      Case 2
```

```
            .Shape.AutoShapeType = msoShapeFoldedCorner
        Case 3
            .Shape.AutoShapeType = msoShapeRightArrow
        Case 4
            .Shape.AutoShapeType = msoShapeRoundedRectangularCallout
        Case 5
            .Shape.AutoShapeType = msoShapePlaque
        Case 6
            .Shape.AutoShapeType = msoShapeDiamond
        Case 7
            .Shape.AutoShapeType = msoShape5pointStar
        Case 8
            .Shape.AutoShapeType = msoShapeCloudCallout
        Case 9
            .Shape.AutoShapeType = msoShapeOval
        Case 10
            .Shape.AutoShapeType = msoShapeHexagon
        Case 11
            .Shape.AutoShapeType = msoShapeOctagon
        Case 12
            .Shape.AutoShapeType = msoShapeCan
        Case 13
            .Shape.AutoShapeType = msoShapeSmileyFace
        Case 14
            .Shape.AutoShapeType = msoShapeHeart
        Case 15
            .Shape.AutoShapeType = msoShape8pointStar
        Case 16
            .Shape.AutoShapeType = msoShapeHorizontalScroll
        Case 17
            .Shape.AutoShapeType = msoShapeVerticalScroll
        Case 18
            .Shape.AutoShapeType = msoShapeWave
        Case 19
            .Shape.AutoShapeType = msoShapeDoubleWave
        Case 20
            .Shape.AutoShapeType = msoShape16pointStar
        Case 21
            .Shape.AutoShapeType = msoShape24pointStar
        Case 22
            .Shape.AutoShapeType = msoShapeFlowchartDocument
        Case Else
            MsgBox "只能输入1到22的自然数", vbInformation, "友情提示"
            .Delete
        End Select
    End With
End Sub
```

执行以上过程会弹出如图 9.32 所示的输入框，输入框的默认值是 Office 的用户名称，假设在其中输入“明天休息”，单击“确定”按钮后程序会继续弹出如图 9.33 所示的输入框。

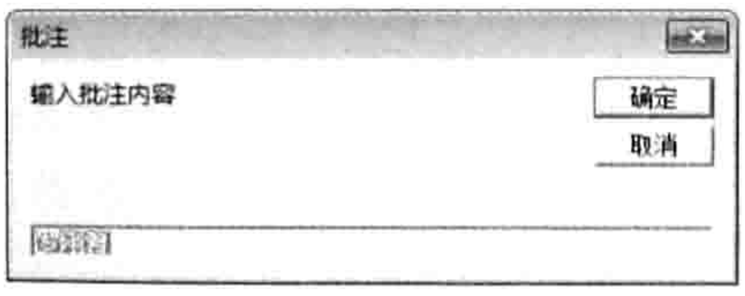


图 9.32　提示输入批注内容

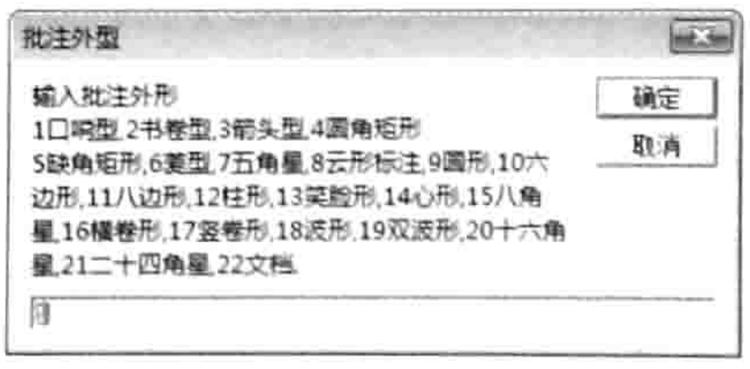


图 9.33　选择批注的形状

在第 2 个输入框中输入 16，表示设置为横卷形批注，然后单击“确定”按钮，活动单元格将产生如图 9.34 所示的批注；如果输入的是 6，表示设置为菱形批注，那么产生的结果如图 9.35 所示。

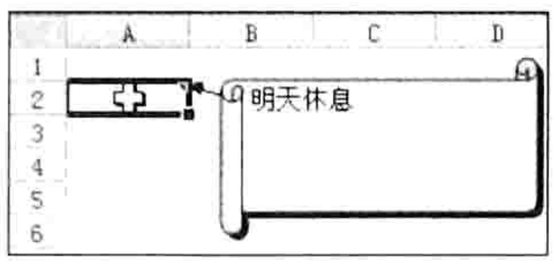


图 9.34　横卷形批注

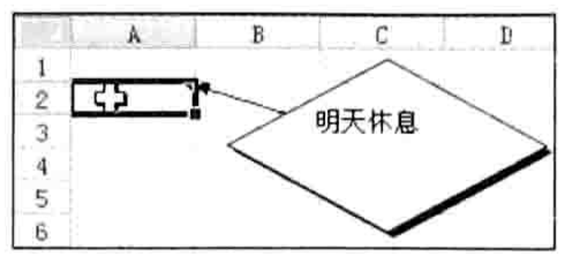


图 9.35　菱形批注

思路分析：

由于要让用户指定批注内容和批注的外观，所以在过程中先使用两个 Application.InputBox 分别创建两个输入框，前者用于录入批注内容，第 8 参数必须使用 2，后者让用户录入形状编码，因此第 8 参数使用 1，强制用户录入数值。

当确定好批注内容与外观后，使用 Range.AddComment 方法向单元格中添加批注，然后使用条件语句根据用户输入的数值修改批注的外观。换而言之，插入批注时的默认外观是不能调整的，只能插入批注后再修改它的 Shape.AutoShapeType 属性。

本例提供了 22 种批注外观，如果用户录入的值超出了 1～22 这个范围，那么程序会弹出提示框告知用户，然后结束过程。

语法补充：

（1）Range.AddComment 方法表示向单元格添加一个批注，如果单元格中已经有批注则会出错。通常采用 On error Resume Next 和 ClearComments 两种方法之一防错。

（2）Shape.AutoShapeType 属性代表图形对象的类型，Excel 提供了 139 种类型，读者可以通过“MsoAutoShapeType 枚举”关键字查询帮助找到这些内置常量的书写方式和含义。

扩展应用：

本例的代码重点在于修改批注的形状，如果要修改批注的颜色，那么可在“ActiveCell.AddComment(mystr)”语句之后添加以下代码，表示将批注填充红色：

```
.Shape.Fill.ForeColor.RGB = RGB(255, 0, 0)
```

本例文件参见光盘：..\第九章\ 9-17 生成个性化批注.xlsm

9.3.4　批量修改当前表的所有批注外观

案例要求：为当前表的所有批注修改外观。

知识要点：Comments.Count 属性、Comment.Shape.AutoShapeType 属性、Comment.Shape.TextFrame.AutoSize 属性。

程序代码：

```
Sub 批量修改批注外观()  '①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Dim Style As String, Com As Comment
  If ActiveSheet.Comments.Count = 0 Then Exit Sub
  Style = Application.InputBox("输入批注外形" & Chr(10) & "1 口哨型,2 书卷型,3 箭头型,4 圆角矩形" & Chr(10) & "5 缺角矩形,6 菱形,7 五角星,8 云形标注,9 圆形,10 六边形,11 八边形,12 柱形,13 笑脸形,14 心形,15 八角星,16 横卷形,17 竖卷形,18 波形,19 双波形,20 十六角星,21 二十四角星,22 文档.", "批注外型", 1, 10, 10, , , 1)
 If Style = "False" Then Exit Sub
  If Style < 1 Or Style > 22 Then MsgBox "只能输入 1-22 之间的整数。", vbInformation:
```

```
Exit Sub
  For Each Com In ActiveSheet.Comments
    With Com
      Select Case Style
      Case 1
        .Shape.AutoShapeType = msoShapeFlowchartSequentialAccessStorage
      Case 2
        .Shape.AutoShapeType = msoShapeFoldedCorner
      Case 3
        .Shape.AutoShapeType = msoShapeRightArrow
      Case 4
        .Shape.AutoShapeType = msoShapeRoundedRectangularCallout
      Case 5
        .Shape.AutoShapeType = msoShapePlaque
      Case 6
        .Shape.AutoShapeType = msoShapeDiamond
      Case 7
        .Shape.AutoShapeType = msoShape5pointStar
      Case 8
        .Shape.AutoShapeType = msoShapeCloudCallout
      Case 9
        .Shape.AutoShapeType = msoShapeOval
      Case 10
        .Shape.AutoShapeType = msoShapeHexagon
      Case 11
        .Shape.AutoShapeType = msoShapeOctagon
      Case 12
        .Shape.AutoShapeType = msoShapeCan
      Case 13
        .Shape.AutoShapeType = msoShapeSmileyFace
      Case 14
        .Shape.AutoShapeType = msoShapeHeart
      Case 15
        .Shape.AutoShapeType = msoShape8pointStar
      Case 16
        .Shape.AutoShapeType = msoShapeHorizontalScroll
      Case 17
        .Shape.AutoShapeType = msoShapeVerticalScroll
      Case 18
        .Shape.AutoShapeType = msoShapeWave
      Case 19
        .Shape.AutoShapeType = msoShapeDoubleWave
      Case 20
        .Shape.AutoShapeType = msoShape16pointStar
      Case 21
        .Shape.AutoShapeType = msoShape24pointStar
      Case 22
        .Shape.AutoShapeType = msoShapeFlowchartDocument
      End Select
      .Shape.TextFrame.AutoSize = True
    End With
  Next Com
End Sub
```

执行以上过程会弹出如图 9.36 所示的提示框，在其中输入 1～22 中的任意数值并单击“确定”

按钮，程序会将工作表中的所有批注框都修改为指定的外观。

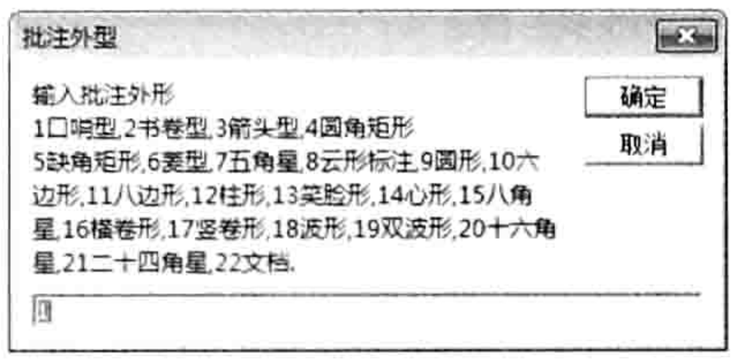


图 9.36　选择批注外观

思路分析：

本例代码首先计算活动工作表中的批注数量，如果等于 0 则直接结束过程。然后使用 Application.InputBox 方法弹出输入框让用户选择批注的新外观，为了避免用户单击“取消”按钮或者输入了 1～22 之外的数据时导致程序浪费执行循环语句的时间，在进入循环语句之前使用了两次条件判断，符合这两个条件之一就结束过程。其中“取消”按钮对应的值是文本“False”而不是 0，因为变量 Style 已经被声明为 String 型。

当用户输入的值是 1～22 中的数值时，使用 For Each Next 循环语句遍历所有批注，然后在循环体中借助条件语句 Select Case 根据用户的输入值决定批注的新外观。

由于批注的外观调整可能会影响批注的内容无法完整显示出来，因此在修改批注的外观后需要将 AutoSize 属性赋值为 True。

语法补充：

（1）Application.InputBox 的第 8 参数使用 1 时，其返回值是数值，如果在输入框中单击“取消”按钮则返回 False。而将 False 赋值给 String 型的变量 Style 后会自动转换成文本“False”；如果变量 Style 被声明为 Integer 型，那么将 False 赋值给变量 Style 则会自动转换成数值 0。因此判断用户是否单击了“取消”按钮要看变量的声明类型。

（2）Comments.Count 属性代表单个工作表中的批注集合的数量，如果要计算工作簿中的批注数量需要配合循环语句遍历所有工作表，然后逐一累加。

扩展应用：

本节 4 个案例都是在手工执行过程中插入批注或者修改批注，事实上 VBA 远不是如此简单，还可以修改 Excel 内置的功能。例如右键菜单中的“插入批注（M）”命令用于创建新批注，可以通过 VBA 修改该菜单的功能，使用户单击鼠标右键时可以默认生成口哨型的批注，完整代码如下：

```
Sub Auto_open()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Application.CommandBars("cell").Controls("插入批注(M)").OnAction = "批注"
End Sub
Sub 批注()
  If ActiveCell.Comment Is Nothing Then
    With ActiveCell.AddComment
      .Text Application.UserName
      .Shape.AutoShapeType = msoShapeFlowchartSequentialAccessStorage
    End With
  Else
    MsgBox "活动单元格已经有批注，不能再生成批注。", vbInformation, "友情提示"
  End If
End Sub
```

本例文件参见光盘：..\第九章\9-18 批量修改所有批注的外观.xlsm

9.4 WorkSheet 对象应用案例

工作表对象的类别名称是 Worksheet，而引用具体的工作表对象则应使用“Worksheets("生产表")”和“Worksheets(2)”这两种形式。

工作表对象大于单元格对象且小于工作簿对象，是每一天的工作都离不开的对象。本节主要展示工作表对象的应用。

9.4.1 新建工作表且命名为今日日期

案例要求：在工作簿中新建一个工作表，保存在所有工作表的右方，并以今日日期命名。

知识要点：Worksheets.Add 方法、Worksheet.Name 属性。

程序代码：

```
Sub 新建工作表()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Dim ShtName As String, sht As Worksheet
  ShtName = Format(Date, "yyyy-mm-dd")
  On Error Resume Next
  Set sht = Worksheets(ShtName)
  If Err.Number = 0 Then
    MsgBox "已经存在名为 " & ShtName & "的工作表", vbInformation, "友情提示"
  Else
    Worksheets.Add(, Sheets(Sheets.Count)).Name = ShtName
  End If
End Sub
```

假设今日是 2014 年 2 月 7 日，工作簿中没有以今日日期命名的工作表，执行以上过程会在工作簿的右端产生名为“2014-02-07”的工作表，效果如图 9.37 所示。

再次执行以上过程，程序会弹如出图 9.38 所示的提示信息。

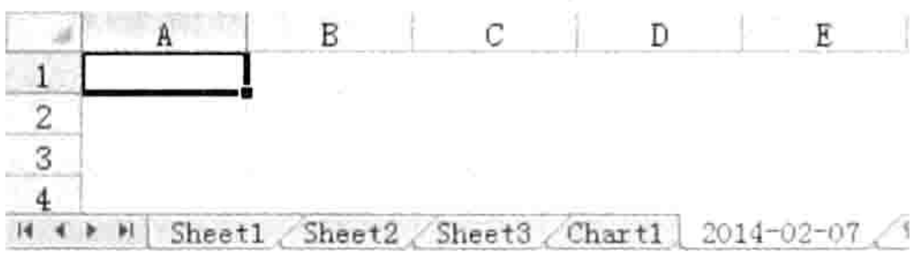


图 9.37　新建的工作表

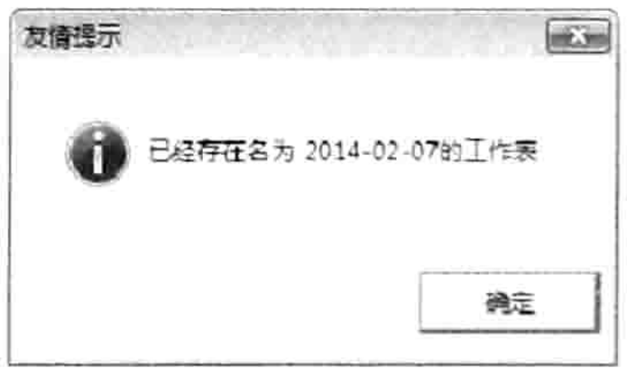


图 9.38　提示

思路分析：

Date 语句可以产生当前的系统日期，日期格式在控制面板中设置，其分隔符有可能是“-”、“/”和“.”三者中的一个。Excel 的工作表名称不支持“/”分隔符，因此在命名前应使用 Format 函数将日期强制转换成“yyyy-mm-dd”格式，采用“-”作为分隔符。

工用簿中不允许存在多个同名的工作表，因此在创建指定名称的工作表之前应先判断工作簿中是否已经存在该名称的工作表。判断方式是使用“Worksheets(Name)”形式的代码引用该工作表，如果引用过程出错则表示工作表不存在，此时 Err.Number 属性值不等于 0。因此本例首先引用工作表，然后使用条件语句根据 Err.Number 的值判定是否新建工作表。

新建指定名称的工作表不能直接使用 Worksheets.Add 方法，而应该在创建工作表前加以判断，确保代码的通用性和准确性。

语法补充：

（1）WorkSheets.Add 方法用于新建工作表、图表或者宏表，新建的表将自动成为活动表。它

的语法如下：

```
Sheets.Add(Before, After, Count, Type)
```

4 个参数都是可选参数。如果忽略所有参数，表示在当前活动工作表之前添加一个新的工作表。本例要求新工作表放在最后面，因此将第 2 参数 After 赋值为 Sheets(Sheets.Count)。

（2）Worksheet.Name 属性用于获取或者修改工作表的名称，是可读亦可写的属性。工作表名称不能超过 31 个字符，不能包含“:\/?*[]”中的任意字符。

扩展应用：

批量新建工作表，数量等于下个月的总天数，而且以下个月每一天的日期命名。完整代码如下：

```
Sub 新建下月工作表()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Dim Days As Byte, i As Byte, sht As Worksheet, ShtName As String
  On Error Resume Next
  Days = Day(DateSerial(Year(Date), Month(Date) + 2, 0))
  Application.ScreenUpdating = False
  For i = 1 To Days
    ShtName = Format(DateSerial(Year(Date), Month(Date) + 1, 0) + i,
"yyyy-mm-dd")
    Set sht = Worksheets(ShtName)
    If Err.Number = 0 Then
      sht.Move , Sheets(Sheets)
    Else
      Worksheets.Add(, Sheets(Sheets.Count)).Name = ShtName
      Err.Clear
    End If
  Next
  Application.ScreenUpdating = True
End Sub
```

假设今天是 2014 年 2 月，工作簿中已经有 3 个工作表，执行以上过程后会在工作簿中新建 31 个工作表，并以 3 月每一天的日期命名，效果如图 9.39 所示。

图 9.39　新建下个月工作表

本例文件参见光盘：..\第九章\ 9-19 新建工作表且以今日日期命名.xlsm

9.4.2　批量保护工作表与解除保护

案例要求：保护所有工作表以及解除保护。

知识要点：Worksheet.ProtectContents 属性、Worksheet.ProtectDrawingObjects 属性、Worksheet.Protect 方法、Worksheet.UnProtect 方法。

程序代码：

```
Sub 批量保护工作表()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Dim sht As Worksheet, Shtname As String
  For Each sht In Worksheets
    If sht.ProtectContents Or sht.ProtectDrawingObjects Then
      Shtname = Shtname & Chr(10) & sht.Name: Err.Clear
```

```
    Else
      sht.Protect "会当凌绝顶"
    End If
  Next sht
  If Len(Shtname) > 0 Then MsgBox "以下工作表已有密码，加密不成功" & Shtname
End Sub
Sub 批量解除工作表保护()
  On Error Resume Next
  Dim sht As Worksheet, Shtname As String
  For Each sht In Worksheets
    sht.Unprotect "会当凌绝顶"
    If Err.Number <> 0 Then Shtname = Shtname & Chr(10) & sht.Name: Err.Clear
  Next sht
  If Len(Shtname) > 0 Then MsgBox "以下工作表解密不成功" & Shtname
End Sub
```

假设工作簿中有 4 个工作表，Sheet2 和 Sheet4 工作表已经用密码 123 保护，执行过程“批量保护工作表”会弹出如图 9.40 所示的提示信息，未提示名称的工作表则表示已加密成功。

接着执行过程“批量解除工作表保护”会弹出如图 9.41 所示的提示信息，未提示名称的工作表则已经解密成功。

图 9.40　提示加密不成功的工作表名称

图 9.41　提示解密不成功的工作表名称

思路分析：

Worksheet.Protect 方法用于对工作表加密，一次只能加密一个工作表，因此本例将它配合循环语句使用。

如果工作表已经处于保护状态，那么 Worksheet.Protect 方法会执行失败，但是不会弹出错误提示，因此不能根据 Err.Number 属性值来判断工作表是否被保护，而是使用 Worksheet.ProtectContents 属性和 Worksheet.ProtectDrawingObjects 属性判断。

在使用 Worksheet.UnProtect 方法解除密码时，如果解密不成功会产生错误提示，Err.Number 的值为 1004，因此可以根据 Err.Number 的值判断是否解密成功。

语法补充：

（1）Worksheet.ProtectContents 属性和 Worksheet.ProtectDrawingObjects 属性分别表示工作表的内容和形状是否处于保护状态，两者都是可读不可写的属性。

（2）Worksheet.Protect 方法用于保护工作表，有 16 个可选参数。它的语法如下：

```
Worksheet.Protect(Password, DrawingObjects, Contents, Scenarios, UserInter
faceOnly, AllowFormattingCells, AllowFormattingColumns, AllowFormattingRows,
AllowInsertingColumns,    AllowInsertingRows,    AllowInsertingHyperlinks,
AllowDeletingColumns,  AllowDeletingRows,  AllowSorting,  AllowFiltering,
AllowUsingPivotTables)
```

其中第 1 参数是密码，可以使用字母、数字、汉字或者标点符号。

（3）Worksheet.UnProtect 方法用于解除工作表的密码，它只有一个代表密码的必选参数，如果指定的密码有误则会产生错误。

扩展应用：

假设要求关闭工作簿时自动保护所有工作表，避免忘记加密，那么可以将加密的代码放在 Workbook_BeforeClose 事件过程中，完整代码如下：

```
'①代码存放位置：ThisWorkbook②随书光盘中有每一句代码的含义注释
Private Sub Workbook_BeforeClose(Cancel As Boolean)
  Dim sht As Worksheet, Shtname As String
  For Each sht In Worksheets
    If sht.ProtectContents = False Or sht.ProtectDrawingObjects = False Then
      sht.Protect "会当凌绝顶"
    End If
  Next sht
End Sub
```

本例文件参见光盘：..\第九章\ 9-20 批量保护工作表与解除保护.xlsm

9.4.3　为所有工作表设置水印

案例要求：对活动工作簿的所有工作表设置可以打印的水印。

知识要点：Worksheet.PageSetup.LeftHeader 属性、Worksheet.PageSetup.LeftHeaderPicture.Filename 属性。

程序代码：

```
Sub 为工作表设置可打印的水印()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Dim Pic As String, rng As Range
  Pic = Application.GetOpenFilename ("图片文件 (*.jpg; *.bmp),*.jpg; *.bmp")
  If Pic = "False" Then exit sub
  ActiveSheet.PageSetup.LeftHeaderPicture.Filename = Pic
  ActiveSheet.PageSetup.LeftHeader = "&G"
End Sub
```

在 Photoshop 软件中设计一张 1557×2258 的图片，在图片中加上水印内容，效果如图 9.42 所示。然后运行以上过程，当弹出选择图片的对话框后从中选择“水印.jpg”文件，然后单击“打开”按钮，程序会将该图片文件插入到页眉中。

按<Ctrl+F2>组合键进入打印预览状态，此时可以看到指定的图片内容已经显示为水印效果，效果如图 9.43 所示。

水印.jpg

图 9.42　作为水印的图片

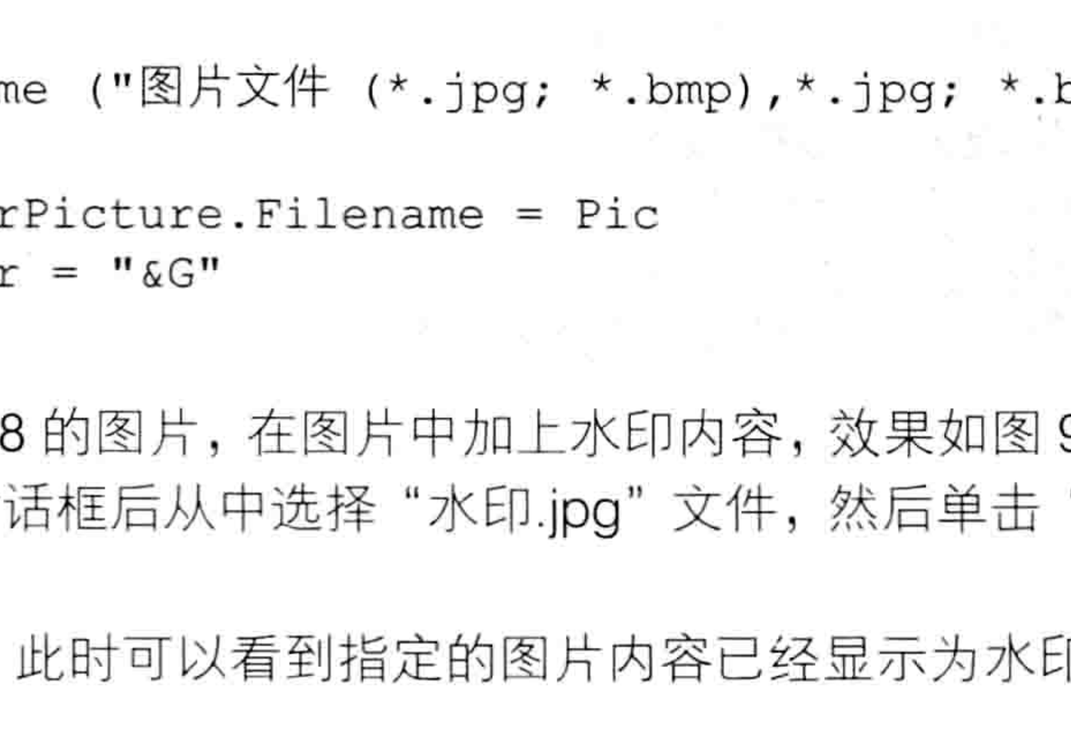


图 9.43　通过打印预览查看水印

思路分析：

在 Word 中可以设置水印，但 Excel 不具备设置水印功能，不过将图片插入到页眉中可以实现与水印一样的效果，因此本例首先使用 GetOpenFilename 方法弹出对话框让用户选择图片，然后将该图片插入到页眉中，当工作表进入预览状态后就可以产生如图 9.43 所示的水印效果。

语法补充：

（1）PageSetup.LeftHeaderPicture.Filename 属性用于指定将要显示在页眉中的图片的路径。但是并非指定图片路径后就可以在页眉中显示图片，还需要将 PageSetup.LeftHeader 属性赋值为“&G”才行。

（2）PageSetup.LeftHeader 代表左边页眉，赋值为“&G”时表示在其中显示图片。表 9-10 中罗列了常用的格式代码与含义说明。

表 9-10　用于页脚和页眉的VBA代码

编码	含义	编码	含义
&D	打印当前日期	&P+数字	打印页号加上指定数字
&T	打印当前时间	&P-数字	打印页号减去指定数字
&F	打印文档名称	&&	打印单个和号
&A	打印工作簿标签名称	&N	打印文档的总页数
&P	打印页号	&Z	打印文件路径
&G	插入图像		

扩展应用：

对活动工作簿中的所有工作表设置水印，代码如下：

```
Sub 为工作表设置可打印的水印()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Dim Pic As String, rng As Range, sht As Worksheet
  Pic = Application.GetOpenFilename("图片文件 (*.jpg; *.bmp),*.jpg; *.bmp")
  If Pic = "False" Then Exit Sub
  For Each sht In Worksheets
    sht.PageSetup.LeftHeaderPicture.Filename = Pic
    sht.PageSetup.LeftHeader = "&G"
  Next sht
End Sub
```

本例文件参见光盘：..\第九章\ 9-21 为工作表设置可打印的水印.xlsm

9.4.4　批量命名工作表

案例要求：根据单元格的值对工作表进行批量命名。

知识要点：WorkSheet.Name 属性。

程序代码：

```
Sub 批量命名所有表()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  On Error Resume Next
  Dim onlys As New Collection, i As Integer, ShtName As String
  For i = 1 To Cells(Rows.Count, 1).End(xlUp).Row
    If Len(Cells(i, 1)) > 0 Then onlys.Add Cells(i, 1),Cells(i, 1)).text
  Next i
```

```
  If Err.Number <> 0 Then
    MsgBox "数据源中存在重复值，请修正后再执行本过程,vbInformation, "友情提示"
  Else
    For i = 1 To Sheets.Count
      Sheets(i).Name = onlys.Item(i)
      If Err.Number <> 0 Then ShtName = ShtName & Chr(10) & Sheets(i).Name:
Err.Clear
    Next i
    If Len(ShtName) > 0 Then MsgBox "以下工作表命名不成功" & ShtName
  End If
End Sub
```

假设工作簿中有 7 个工作表，A1:A7 中有如图 9.44 所示的数据，执行以上过程后会将 7 个工作表命名为 A1:A7 的值，效果如图 9.45 所示。

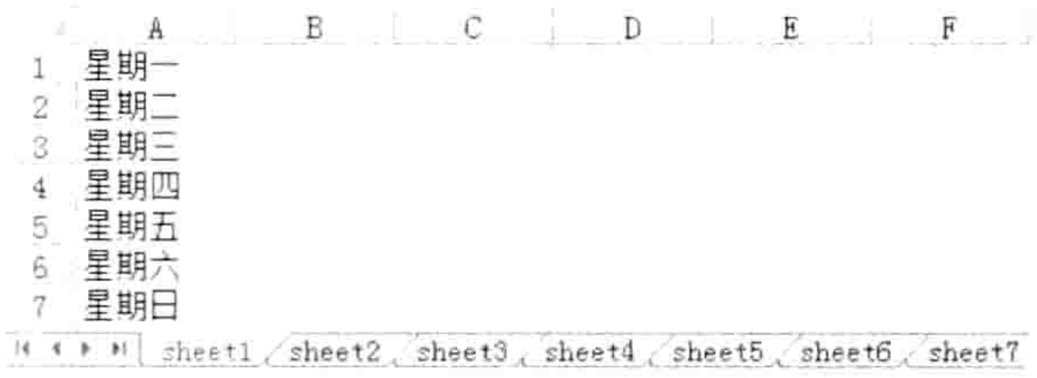


图 9.44　命名前

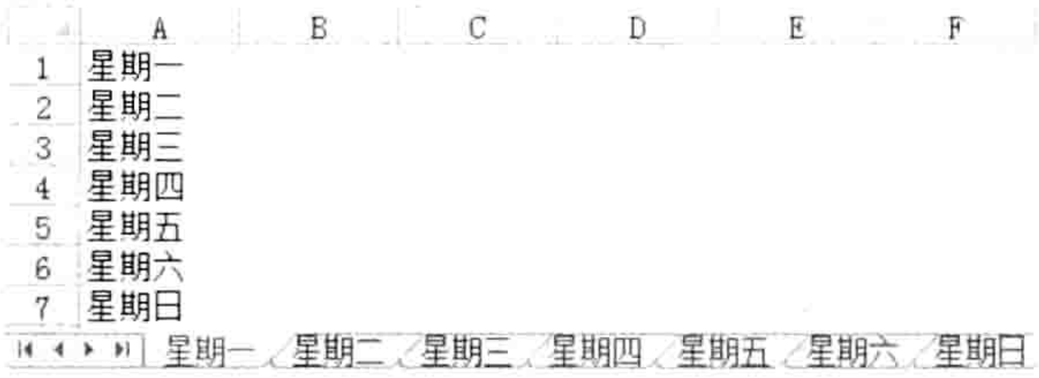


图 9.45　命名后

如果 A1:A7 区域中存在重复值，那么无法对工作表命名，而是弹出“数据源中存在重复值，请修正后再执行本过程”的信息框。如果部分单元格中包含“/”、“*”等字符，由于工作表名称中禁止使用这些字符，因此会弹出未命名成功的所有工作表名称。

思路分析：

工作簿中不允许存在多个同名的工作表，因此批量命名工作表前有必要检测数据源是否包含重复值。

集合对象 Collection 是一个容器，可以向其中添加任意字符，容器的 Key 值具有唯一性，如果向容器中的 Key 参数写入重复值会产生错误。基于此原则，先用代码“On Error Resume Next”强制代码一直执行下去，然后再通过循环语句向集合对象 Collection 的 Key 参数中逐一写入区域中的值，最后查看 Err.Number 属性的值是否大于 0 即可判断这个区域中是否存在重复值。

由于集合对象 Collection 的 Key 参数不能使用数值，因此 Collection.Add 方法的第 2 参数使用 Cells(i, 1).Text，而非 Collection 的 Cells(i, 1)或者 Cells(i, 1).Value。Range.Text 属性总是返回 String，不管单元格中是数值还是文本。

当确定 A 列不存在重复值后，使用 For Next 循环语句遍历所有表，然后用集合 Onlys 中的第 i 个值对第 i 个表命名，当循环完成后，所有工作表都已命名完成。

不过由于工作表名称不能包含“:”“\”“?”“*”“[]”中的任意字符，因此对表命名有可能不成功，所以在过程中根据 Err.Nmuber 属性值是否等于 0 来判断表命名成功与否，并且将命名失败的表名称报告给用户。

语法补充：

（1）Collection 对象属于集合对象，可以将任意字符添加到集合中，然后再像引用单元格一样将它引用出来使用。Collection 对象可以存放一行数据，通过 Collection 对象的 Add 方法将数据添加到集合中。在添加数据时允许通过 Key 参数为每一个值命名，因此实际上可以向集合中写入两行相关联的数据，其中 Item 参数所关联的数据允许重复，而 Key 参数所关联的数据不重复。

Collection.Add 的语法如下：

```
Collection.Add item, key, before, after
```

4 个参数的功能说明见表 9-11。

表 9-11　Collection.Add方法的参数说明

参数名称	功能描述
item	必选参数，这是要添加到集合中的成员，不具有唯一值，允许是文本或数值
key	可选参数，用于对对应的 Item 命名，可以通过 key 名称来访对应的 Item 值
before	可选参数，用于指定集合中的相对位置，新添加的值将存放于此位置之前。当此参数是数值时，表示以序号指定位置，序号必须介于 1 到数据个数范围中；当此参数是字符时，那么以名称指定位置。before 和 after 两个参数不能同时使用
after	可选参数，用于指定集合中的相对位置，新添加的值将存放于此位置之后。当此参数是数值时，表示以序号指定位置，序号必须介于 1 到数据个数范围中；当此参数是字符时，那么以名称指定位置。before 和 after 两个参数不能同时使用

为了让读者对集合对象有更深刻的理解，请看以下案例：

```
Sub 向集合中写入与导出数据()'代码存放位置：模块中
  Dim onlys As New Collection '声明变量
  onlys.Add "张达明", "第一届" '向集合中写入“张达明”，并将它命名为“第一届”
  onlys.Add "胡丽丽", "第二届" '向集合中写入“胡丽丽”，并将它命名为“第二届”
  onlys.Add "陈文坤", "第三届" '向集合中写入“陈文坤”，并将它命名为“第三届”
  onlys.Add "胡丽丽", "第四届" '向集合中写入“胡丽丽”，并将它命名为“第四届”
  For i = 1 To 4 '从 1 到 4
    Cells(i, 1) = onlys.Item(i) '将集合中第 i 个值写入到第 i 个单元格中
  Next
End Sub
```

过程首先创建一个名为 onlys 的集合对象，然后分 4 次向集合中写入 4 个值：张达明、胡丽丽、陈文坤、胡丽丽，并且分别将它们命名为“第一届”、“第二届”、“第三届”和“第四届”，其中 Item 参数添加的数据存在重复值，而 Key 参数添加的名称没有重复，如果 Key 有重复值必然会添加失败。在过程的最后，通过循环语句将集合中的 4 个数据分别写入 A1:A4 单元格中去。

在以上过程中，读取集合中的值时采用的是序号，例如使用代码“Msgbox onlys.Item(2)”可以获得第 2 个数据“胡丽丽”，若改用名称来引用数据也可以获取相同结果，例如代码“MsgBox onlys.Item（"第二届"）”的结果仍然是“胡丽丽”。

名称具有唯一性，否则名称就没有了存在价值。假设将以上过程中的“第四届”修改为“第二届”，然后再执行代码必定会出错。基于此原则，在工作中使用 Collection 对象的目的通常是判断一组数组是否重复，或者提取一组数据的唯一值。

图 9.46 中的代码用于提取 A 列的唯一值，然后输出到 A 列中。

图 9.46　使用 Collection 对象提取 A 列的唯一值

（2）本例中使用 Sheets 而不用 Worksheets 是因为要对工作簿中的所有表命名，而非仅针对工作表，否则当工作簿中有非嵌入式的图表和宏表时会命名不完整。

扩展应用：

通过 Worksheet.Name 属性可以为工作表指定新的名称，也可以替换工作表名称中的部分字符，或者在原来的表名前后追加新的字符。假设要将“星期一”、“星期二”、“星期三”……重命名为“周一”、“周二”、“周三”……，那么可用以下代码实现：

```
Sub 批量替换表名称()'代码存放位置：模块中
  For Each sht In Sheets '遍历所有表
    sht.Name = Replace(sht.Name, "星期", "周") '将表名中的“星期”替换成“周”
  Next sht
End Sub
```

实现效果如图 9.47 所示。

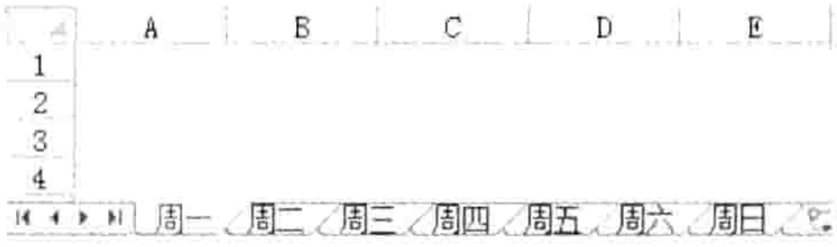


图 9.47　替换表名称

本例文件参见光盘：..\第九章\9-22 批量命名工作表.xlsm

9.4.5　判断筛选条件

案例要求：根据给出的条件筛选工作表中的数据比较简单，现要求在处于筛选状态下的工作表中快捷找出每一列的筛选条件。

知识要点：WorkSheet.FilterMode 属性、WorkSheet.AutoFilter 属性、Filters.Count 属性、Filter.On 属性、Filter.Criteria1 属性、Filter.Criteria2 属性、Filter.Operator 属性。

程序代码：

```
Sub 找出筛选条件()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Dim i As Byte, msg As String
  If ActiveSheet.FilterMode Then
    For i = 1 To ActiveSheet.AutoFilter.Filters.Count
      With ActiveSheet.AutoFilter.Filters(i)
        If .On Then
          If .Operator >= 0 And .Operator < 7 Then
            msg = msg & Chr(10) & "第" & i & "列:" & .Criteria1
            If .Operator = 1 Or .Operator = 2 Then
              msg = msg & vbTab & IIf(.Operator = 1, "而且", "或者") & vbTab
& .Criteria2
            End If
          ElseIf .Operator > 6 And .Operator < 12 Then
            msg = msg & Chr(10) & "第" & i & "列:" & WorksheetFunction.
VLookup(.Operator, {7,"按值筛选(不少于 3 个)";8,"按单元格颜色筛选";9,"按字体颜色筛
选";10,"按图标筛选";11,"高于平均值或低于平均值"}, 2, False)
          End If
        End If
      End With
```

```
    Next i
    MsgBox msg, vbInformation, "友情提示"
  End If
End Sub
```

图 9.48 中的成绩表处于筛选状态，执行过程“找出筛选条件”后弹出如图 9.49 所示的信息框，表示工作表的第 1 列没有设置筛选条件，第 2 列有一个筛选条件，第 3 列和第 4 列各有两个筛选条件，第 5 种是按颜色筛选。

	A	B	C	D	E
1	姓名	性别	成绩	级别	名次
20	胡不群	女	79	一班	22
28	蒋文明	女	77	二班	26
45	柳三秀	女	78	二班	24

图 9.48　筛选状态下的成绩表

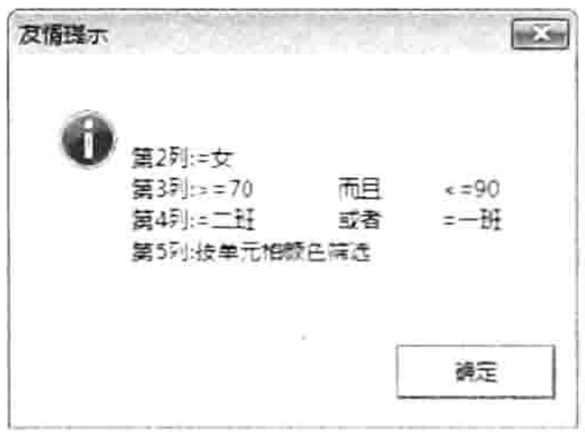


图 9.49　筛选条件

思路分析：

过程首先使用 WorkSheet.FilterMode 属性判断工作表是否处于筛选模式，如果是则使用循环语句遍历所有筛选，在循环体中通过 Filter.On 属性判断每一个选择是否指定了筛选条件，如果指定了条件则参照表 9-12 的操作符将条件分类，按类别不同采取不同的方式提取筛选条件。

Excel 的筛选操作符包含 12 种，操作符的值为 1 和 2 时表示有两个筛选条件，可以通过 Filter.Criteria1 属性和 Filter.Criteria2 属性获取这两个条件的值；当操作符的值是 0、3、4、5、6 时表示只有一个筛选条件，可以通过 Filter.Criteria1 属性获取筛选条件的值；当操作符的值为 7～11 时无法提取具体的筛选条件，只能用一句话描述筛选方式。

基于以上规则，本例代码首先使用代码“If .Operator >= 0 And .Operator < 7 Then”将所有通过 Filter.Criteria1 属性指定筛选条件的筛选列数与条件 1 提取出来，然后使用代码“If .Operator = 1 Or .Operator = 2 Then”进一步提取条件 2，如果以上都不是，则属于第三类——“只能用一句话描述筛选方式”的筛选，为了简化条件判断，本例使用了工作表函数 Vlookup 从列表中自动查找对应的筛选方式描述。

Excel 2003 自带的筛选条件比较单一，判断筛选条件也简单很多，从 Excel 2007 开始增加了按图标筛选、按颜色筛选、按最大值筛选、按高于或低于平均值筛选等，因此代码比较复杂，需要使用比较多的条件语句。

语法补充：

（1）WorkSheet.FilterMode 属性用于判断工作表是否处于筛选模式，值为 True 时表示是筛选模式，否则表示不是筛选模式。

（2）AutoFilter.Filters 对象集合代表工作表中的自动筛选集合，可以使用索引号引用其中单个筛选。工作表处于筛选模式时，有多少个筛选器（倒三角符号）那么 AutoFilter.Filters.Count 的值就等于几。

（3）Filter.Operator 属性代表筛选符，用于区分当前的筛选方式，以及筛选条件的数量。Excel 有 12 个筛选符，其值与含义见表 9-12。

表 9-12　操作符常量枚举

名称	值	说明
（无）	0	按单值筛选（由条件 1 决定）
xlAnd	1	同时满足两个条件

续表

名称	值	说明
xlOr	2	满足两个条件中的一个
xlTop10Items	3	筛选前 N 大值（由条件 1 决定）
xlBottom10Items	4	筛选前 N 小值（由条件 1 决定）
xlTop10Percent	5	筛选最大百分比（由条件 1 决定）
xlBottom10Percent	6	筛选最小百分比（由条件 1 决定）
xlFilterValues	7	按多值筛选（不少于 3 个）
xlFilterCellColor	8	按单元格颜色筛选
xlFilterFontColor	9	按字体颜色筛选
xlFilterIcon	10	按图标筛选
xlFilterDynamic	11	高于平均值或低于平均值

（4）Filter.On 属性用于说明某一个筛选是否指定了筛选条件，当至少指定一个筛选条件时则 Filter.On 属性值为 True。Filter.On 属性和 WorkSheet.FilterMode 属性的功能大大不同，WorkSheet.FilterMode 属性表示工作中是否使用了筛选功能，只要工作表中有筛选器就算处于筛选模式下，而不管每一个筛选器中是否指定了筛选条件。

（5）Filter.Criteria1 属性和 Filter.Criteria2 属性分别代表筛选条件 1 和筛选条件 2，只有操作符是 1 或者 2 时才会使用条件 2，而操作符是 0～6 中的任意一个值时都会使用条件 1。

扩展应用：

假设工作簿中有若干个工作表处于筛选模式，现要求让每个工作表显示出所有数据，避免合并工作表或者数据汇总时产生遗漏，那么有以下两种思路可以实现：

```
'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
Sub 让筛选状态的工作表显示所有数据1()
  Dim sht As Worksheet
  For Each sht In Worksheets
If sht.FilterMode Then sht.AutoFilter.Range.AutoFilter
  Next sht
End Sub
Sub 让筛选状态的工作表显示所有数据2()
  Dim sht As Worksheet
  For Each sht In Worksheets
    If sht.FilterMode Then sht.ShowAllData
  Next sht
End Sub
```

Worksheet.AutoFilter.Range 代表工作表中的自动筛选区域，即拥有倒三角符号的单元格集合。Range.AutoFilter 方法让自动筛选区域取消筛选，而 Worksheet.ShowAllData 方法不会取消筛选，只是清除筛选条件，不过两者都可以让筛选模式的工作表显示出隐藏的数据。

本例文件参见光盘：..\第九章\ 9-23 获取工作表的筛选条件.xlsm

9.5　Workbook 对象应用案例

Workbook 对象即工作簿对象。工作簿是一个单独的文档，保存完整的报表信息，不像工作表

或单元格那样附属于其他对象中。本节提供与工作簿对象相关的 6 个应用实例。

9.5.1 拆分工作簿

案例要求：将工作簿按工作表拆分，将每个工作表存为一个独立的工作簿，保存路径由用户选择。

知识要点：Workbook.SaveAs 方法、Workbook.Close 方法。

程序代码：

```
Sub 拆分工作簿()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Dim path As String, sht As Worksheet
  With Application.FileDialog(msoFileDialogFolderPicker)
    If .Show = -1 Then
      path = .SelectedItems(1) & IIf(Right(.SelectedItems(1), 1) = "\", "", "\")
    Else
      Exit Sub
    End If
  End With
  Application.ScreenUpdating = False
  For Each sht In Sheets
    sht.Copy
    ActiveWorkbook.SaveAs path & sht.Name, xlWorkbookDefault
    ActiveWorkbook.Close , False
  Next sht
  Application.ScreenUpdating = True
End Sub
```

假设活动工作簿中有如图 9.50 所示的 7 个工作表，执行以上过程后会弹出“浏览”对话框，在其中选择“C:\拆分结果”文件夹，然后单击“确定”按钮，程序会瞬间将活动工作簿中的 7 个工作表拆分成如图 9.51 所示的工作簿。

	A	B	C	D
1	姓名	机台	产量	
2	黄真真	1#	704	
3	黄淑宝	2#	882	
4	欧阳华	3#	671	
5	童怀礼	4#	711	
6	曹值军	5#	840	
7	陈胡明	6#	688	
8	张开来	7#	714	
9	张彻	8#	701	

星期一 星期二 星期三

图 9.50 拆分前的工作簿

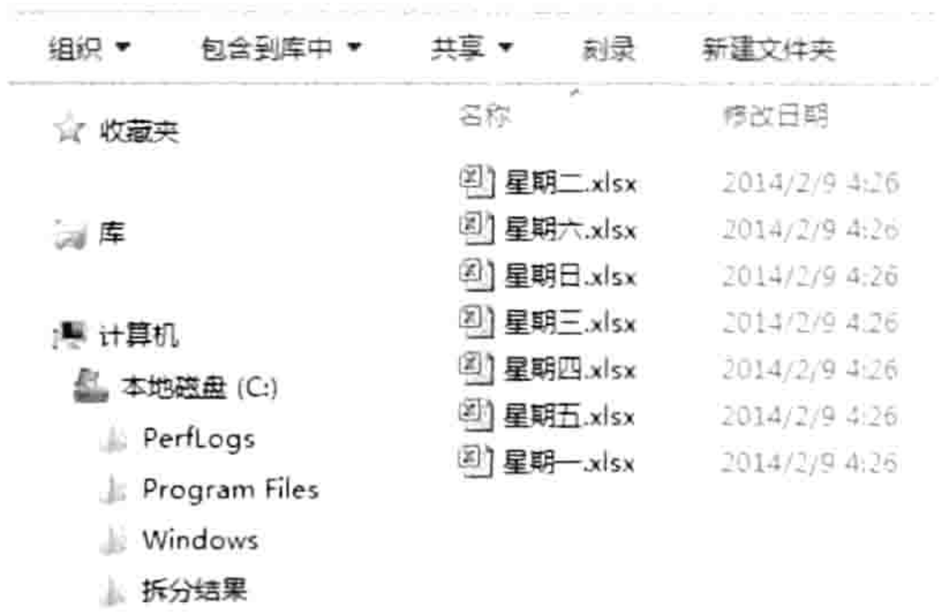


图 9.51 拆分后的工作簿

思路分析：

首先使用 Application.FileDialog 属性创建一个“浏览”对话框供用户指定路径，然后使用循环语句 For Each Next 遍历活动工作簿中的所有表，通过 Worksheet.Copy 方法将表复制到新工作簿中，再用 Workbook.SaveAs 方法将新工作簿另存到前面所指定的路径中，以工作表的名称作为新工作簿的名称，最后关闭新工作簿。当循环语句完成后工作簿即已被拆分完成。

由于拆分工作簿时屏幕会频繁闪动，因此在循环语句之前有必要通过“Application.ScreenUpdating = False”语句关闭屏幕更新，循环结束后再恢复更新。

语法补充：

（1）Worksheet.Copy 方法表示将工作表复制到工作簿的另一位置，如果不指定参数则表示将

新建一个工作簿，并且将工作表复制到新工作簿中。它的语法如下：

```
Worksheet.Copy(Before, After)
```

其中 Before 和 After 都是可选参数，表示复制工作表时的参照位置。前者表示将工作表复制到该表之前，后者表示将工作表复制到该表之后，两者不能同时使用。如果两者都忽略则将工作表复制到新建的工作簿中。

（2）Workbook.Close 方法表示关闭工作簿，它的第 1 参数 SaveChanges 表示关闭工作簿时是否保存工作簿，赋值为 False 时表示保存，否则不保存。

（3）Workbook.SaveAs 方法表示另存工作簿，其语法如下：

```
Workbook.SaveAs(FileName, FileFormat, Password, WriteResPassword,
ReadOnlyRecommended, CreateBackup, AccessMode, ConflictResolution, AddToMru,
TextCodepage, TextVisualLayout, Local)
```

Workbook.SaveAs 方法参数众多，不过常用的参数是前 3 个，第 1 参数代表文件名称，包含完整的路径；第 2 参数指定文件的格式，必须要与文件的扩展名一致；第 3 参数代表文件的打开密码。

Excel 2010 支持 54 种工作簿格式，常用的格式代码有以下 6 种。

表 9-13　常用的文件格式代码

格式名称	说明	扩展名
xlAddIn	Excel 2007 加载项	xlam
xlAddIn8	Excel 97-2003 加载项	xla
xlExcel8	Excel 8	xls
xlOpenXMLWorkbook	打开 XML 的工作簿	xlsx
xlOpenXMLWorkbookMacroEnabled	打开启用宏的 XML 工作簿	xlsm
xlWorkbookDefault	默认工作簿	由 Excel 选项设置决定

扩展应用：

假设工作表中有公式，要求拆分后的工作簿中只显示公式的值，避免公式跨工作簿引用数据失败而导致数据不完整，那么应在本例过程的“sht.Copy”语句之后插入以下代码：

```
ActiveSheet.UsedRange = ActiveSheet.UsedRange.Value
```

本例文件参见光盘：..\第九章\ 9-24 拆分工作簿.xlsm

9.5.2　每 10 分钟备份一次工作簿

案例要求：Excel 有工作簿定时保存的功能，但无法实现定时备份。现要求从打开工作簿开始每 10 分钟备份一次工作簿。第一次备份前弹出对话框让用户选择备份的路径，以后的备份操作则全自动完成。

知识要点：Workbook.SaveCopyAs 方法、Workbook.Name 属性。

程序代码：

```
Private Sub Workbook_Open()'①代码存放位置：ThisWorkbook②随书光盘中有每一句代码含义注释
  With Application.FileDialog(msoFileDialogFolderPicker)
```

```
.Title = "请选择备份文件的路径"
    If .Show Then PathStr = .SelectedItems(1) Else Exit Sub
  End With
  PathStr = PathStr & IIf(Right(PathStr, 1) = " \ ", "", " \ ")
  Call 备份
End Sub
```

以上代码存放在 ThisWorkbook 中。

```
Public PathStr As String
Sub 备份()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Application.OnTime Now + TimeValue("00:10:00"), "备份"
  ThisWorkbook.SaveCopyAs PathStr & Format(Now, "HHMMSS--") & ThisWorkbook.Name
End Sub
```

以上代码存放在模块中，然后保存并且重启工作簿，在打开工作簿时会弹出选择备份路径的对话框，假设活动工作簿名称为“2 月生产表.xlsm”，备份路径设置为“D:\备份”，当指定路径并且单击“确定”按钮后程序会马上将文件备份在“D:\备份”中。

其后每 10 分钟备份文件一次，每一次的备份文件的名称都由备份时间加“2 月生产表.xlsm”组成，效果如图 9.52 所示。

图 9.52　D 盘中的备份文件

遇到停电等意外故障时可能会导致文件被损坏，定时备份文件可以减少损失，而且当编辑文档过程中产生失误时，如果想返回前面的某一个步骤，而这个步骤超出了 Excel 的可撤销步骤，那么最好的办法是打开备份文件。

代码中的备份文件间隔时间可以根据需求随意修改，而在每天上班时可以根据需求决定是否删除头一天的所有备份文件，避免备份文件夹中有太多不需要的文件。

思路分析：

定时备份文件主要涉及三个知识点：指定路径、备份文件、定时调用过程。

对于“指定路径”，本例首先在模块中定义一个公共变量 PathStr，便于 Workbook_Open 事件和“备份”两个过程调用，然后在 Workbook_Open 事件过程中通过 Application.FileDialog 方法弹出对话框让用户指定路径。

对于“备份文件”，调用 Workbook.SaveCopyAs 方法即可，它可将工作簿备份在任意路径下，备份时可以随意指定路径文件名称，而且备份时不会改变活动工作簿的路径，这是与 Workbook.SaveAs 方法最大的区别。

对于“定时调用过程”，使用 Application.OnTime 方法每 10 分钟调用一次“备份”过程即可。尽管代码中只用 Application.OnTime 方法创建了一个计划任务，但由于它调用的是过程本身，因此在执行备份时会反复添加计划任务，从而形成每 10 分钟创建一个新的任务。

语法补充：

（1）Workbook.SaveCopyAs 方法用于备份工作簿，而在备份后不改变活动工作簿的路径。Workbook.SaveAs 方法用于另存工作簿，另存后会改变活工作簿的路径。

Workbook.SaveCopyAs 方法的语法如下：

```
Workbook.SaveCopyAs(Filename)
```

参数 Filename 用于指定备份后的文件名称，包括路径。VBA 帮助中说此参数是可选参数，应该是书写有误，忽略此参数时无法执行备份。

（2）Workbook.Name 属性代表工作簿的名称，不包含路径，而 Workbook.FullName 属性则包含路径。

扩展应用：

假设要用代码删除“D:\备份”路径中的所有 Excel，可用以下过程完成：

```
Sub 删除备份文件夹中的所有 Excel 文件()
 Kill "d:\备份\*.xls*"
End Sub
```

Kill 方法用于删除文件，支持“*”和“?”两个通配符。

本例文件参见光盘：..\第九章\ 9-25 每 10 分钟备份工作簿.xlsm

9.5.3　5 分钟未编辑工作簿则自动备份

案例要求：假设在制表过程中未保存工作簿就离开了座位，在回来之后有可能发现工作簿已被他人意外关闭，或者由于停电、意外重启等原因导致工作簿未保存，从而丢失数据。现要求利用 VBA 代码实现 5 分钟未编辑工作簿（对单元格赋值、删除、插入行等）则自动将文件备份到“D:\备份”路径中。

知识要点：Workbook.SaveCopyAs 方法、Application.OnTime 方法

程序代码：

```
'①代码存放位置：ThisWorkbook②随书光盘中有每一句代码含义注释
Private Sub Workbook_SheetChange(ByVal Sh As Object, ByVal Target As Range)
  On Error Resume Next
  Application.OnTime Time1, "备份", , False
  Time2 = Now + TimeValue("00:05:00")
  Application.OnTime Time2, "备份"
  Time1 = Time2
End Sub
Private Sub Workbook_SheetSelectionChange(ByVal Sh As Object, ByVal Target As
Range)
  On Error Resume Next
  Application.OnTime Time1, "备份", , False
  Time2 = Now + TimeValue("00:05:00")
  Application.OnTime Time2, "备份"
  Time1 = Time2
End Sub
```

以上代码存放在 ThisWorkbook 中。

```
Public Time1 As Date, Time2 As Date
```

```
Sub 备份()
  ThisWorkbook.SaveCopyAs "D:\备份\" & ThisWorkbook.Name
End Sub
```

以上代码存放在模块中，然后保存并且重启工作簿，在任意单元格中录入数据，等 5 分钟后可以发现“D:\备份”文件夹中已经产生了备份文件。如果任意两次修改单元格的间隔时间小于 5 分钟，那么“D:\备份”文件夹中不会产生备份文件。

思路分析：

要想实现“5 分钟未编辑则备份文件”，其重点有两个：其一是如何判断文件是否已结束编辑，其二是如何取得编辑结束后的 5 分钟这个时间点。

编辑工作簿主要体现在两方面，一是修改单元格的值、插入行、删除行等操作，可以通过 Workbook_SheetChange 事件捕捉到这些操作，进而获取这些操作的时间。其二是无法用 Workbook_SheetChange 事件捕捉到的操作，例如对单元格设置背景色、修改字体、生成图表等，所幸执行这些操作前都会选择区域，而选择区域的操作会触发 Workbook_SheetSelectionChange 事件，因此可以简单地认为触发 Workbook_SheetChange 事件和 Workbook_SheetSelectionChange 事件时表示工作簿正处在编辑过程中。

VBA 中的 Now 函数可以取得当前时间，因此代码“Now + TimeValue("00:05:00")”表示 5 分钟之后的时间。在 Workbook_SheetChange 事件和 Workbook_SheetSelectionChange 事件中使用 Application.OnTime 方法添加一个 5 分钟后执行备份文件的计划任务就可以实现本例的一半需求，剩下的一半需求——如果两次编辑的时间间隔不到 5 分钟则不能备份文件——仍然通过 Application.OnTime 方法实现，只不过不是使用它添加计划任务，而是将它的第 4 参数赋值为 False，从而取消前面所添加的计划任务。

简而言之，实现本例需求的思路是：在 Workbook_SheetChange 事件和 Workbook_SheetSelectionChange 事件中添加一个 5 分钟后备份文件的计划任务，同时取消上一次所设置的计划任务。假设两次触发事件的间隔时间小于 5 分钟，由于计划任务已经被取消，因此不会在闲置时间小于 5 分钟时备份文件；假设两次触发事件的间隔时间大于等于 5 分钟，由于计划任务已经执行完毕，因此取消计划任务的操作会执行失败，不会影响备份文件，因此过程的重点在于借助时间差完美实现按时备份，不到时间则不备份。

语法补充：

Application.OnTime 方法用于设置计划任务，当第 4 参数的值是 False 时则用于取消预设的计划任务，其语法如下：

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

扩展应用：

打开工作簿后每 1 分钟保存一次工作簿，代码如下：

```
Sub Auto_Open()'代码存放位置：模块中
  Application.OnTime Now + TimeValue("00:01:00"), "Auto_Open"'设置计划任务
  ThisWorkbook.Save'保存文件
End Sub
```

本例文件参见光盘：..\第九章\9-29 5 分钟不编辑则自动备份工作簿.xlsm

9.5.4 记录文件打开次数

案例要求：记录文件打开次数。

知识要点： Workbook.CustomDocumentProperties 属性、DocumentProperties.Add 方法、DocumentProperty.Value 属性。

程序代码：

```
Sub Auto_open()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  On Error Resume Next
  Dim Open_Count As Integer
  With ActiveWorkbook.CustomDocumentProperties
    Open_Count = .Item("打开次数").Value
    If Err.Number<> 0 Then
      .Add "打开次数", False, msoPropertyTypeNumber, 1
    Else
      .Item("打开次数") = .Item("打开次数").Value + 1
    End If
    MsgBox "本文件打开次数：" & .Item("打开次数").Value
  End With
End Sub
```

将以上代码存放在模块中，然后保存并且重启工作簿，Excel 会提示“本文件打开次数：1”。如果关闭后再次打开则会提示“本文件打开次数：2”。

思路分析：

本例过程取名为“Auto_open”，其目的是让过程在打开工作簿时自动运行。

在“Auto_open”过程中，首先判断是否存在名为“打开次数”的自定义属性，如果没有则使用 CustomDocumentProperties.Add 方法新建该属性，默认值为 1，如果已经有该属性，则对该属性的值累加 1，最后使用 Msgbox 函数报告“打开次数”属性的值。

没有函数或者属性用于判断某个自定义属性是否存在，只能根据读取属性的值是否出错来判断是否存在该属性。

语法补充：

（1）Workbook.CustomDocumentProperties 代表自定义属性集合，可以随意添加、删除文件的自定义属性。

（2）DocumentProperties.Add 方法表示向工作簿中创建一个自定义属性，新建的自定义属性保存在工作簿中，可以通过单击菜单中的“文件”→“信息”→“属性”→“高级属性”命令打开工作簿的属性对话框，然后在“自定义”选项卡中看到 VBA 代码新建的自定义属性。

DocumentProperties.Add 方法的语法如下：

```
DocumentProperties.Add(Name, LinkToContent, Type, Value, LinkSource)
```

表 9-14 DocumentProperties.Add方法的参数说明

参数名称	必选/可选	说明
Name	必选	新属性的名称
LinkToContent	必选	指定属性是否链接到容器文档中的内容。如果参数为 True，则必需通过 LinkSource 参数指定链接对象，如果为 False，则必为 Value 参数赋值
Type	可选	新属性的数据类型。可以是以下常量之一：msoPropertyBoolean、msoPropertyDate、msoPropertyFloat、msoPropertyNumber 或 msoPropertyString。
Value	可选	属性的值。如果 LinkToContent 属性为 True，则忽略此参数
LinkSource	可选	链接属性的来源。如果 LinkToContent 为 False，则忽略此参数

以下过程可以获取活动工作簿中所有自定义属性的名称与值：

```
Sub 获取自定义属性()'代码存放位置：模块中
  Dim Item As Byte, P As DocumentProperty '声明变量
  For Each P In ActiveWorkbook.CustomDocumentProperties '遍历所有自定义属性
    Item = Item + 1 '累加变量
    Cells(Item, 1).Value = P.Name '将自定义属性的名称输出到第一列
    Cells(Item, 2).Value = P.Value '将自定义属性的值输出到第二列
  Next
End Sub
```

扩展应用：

利用自定义属性配合工作簿事件可以让工作簿只能打开 3 次，关闭工作簿时自动销毁文件。完整代码如下：

```
Dim Open_Count As Byte
Sub Auto_Open()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  On Error Resume Next
  With ActiveWorkbook.CustomDocumentProperties
    Open_Count = .Item("打开次数").Value
    If Err <> 0 Then
      .Add "打开次数", False, msoPropertyTypeNumber, 1
    Else
      .Item("打开次数") = .Item("打开次数").Value + 1
    End If
    ThisWorkbook.Save
  End With
End Sub
Sub Auto_Close()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  If Open_Count = 3 Then
    With ThisWorkbook
      .Saved = True
      .ChangeFileAccess Mode:=xlReadOnly
      Kill .FullName
      .Close
    End With
  End If
End Sub
```

本例文件参见光盘：..\第九章\9-30 记录文件打开次数.xlsm

9.5.5 不打开工作簿而提取数据

案例要求：从关闭的工作簿中获取数据，允许用户随意指定工作簿的路径、工作表名称和区域地址。假设当前工作簿同路径下有一个文件夹名为“人事资料”，其中有一个工作簿“人事报表”，现要求在不打开工作簿的前提下获取其中任何工作表、任何区域的值。

知识要点：Workbook.Path 属性。

程序代码：

```
'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
Sub 取值(路径 As String, 文件 As String, 工作表, 单元格 As String)
```

```
  On Error Resume Next
  Dim rng As Range
  Set  rng  =  ActiveCell.Resize(Range( 单 元 格 ).Rows.Count,  Range( 单 元
格).Columns.Count)
  If Err.Number <> 0 Then
    MsgBox "请调整区域,当前区域不足以存放引用区域的值" & Chr(10) & "建议选择 A1 再执行
程序", vbInformation, "提示"
    Exit Sub
  End If
  With rng
    .FormulaArray = "='" &路径& "\[" &文件& "]" &工作表& "'!" &单元格
    .Value = .Value
  End With
End Sub
Sub 不打开工作簿而获取其值 1()
  取值 ThisWorkbook.Path & "\人事资料", "人事报表.xls", "部门人员统计", "A1:B9"
End Sub
```

在模块中录入以上代码，然后将工作簿保存在文件“人事资料”的相同路径下（由于过程“不打开工作簿而获取其值 1”是使用的相对路径，因此当前的工作簿必须在“人事资料”的相同路径下，否则无法通过代码 ThisWorkbook.Path 引用路径），最后执行过程“不打开工作簿而获取其值 1”，假设活动单元格是 XFD2，由于它右边没有单元格，不足以存放“人事报表.xls”工作簿中 A1:B9 区域的值，因此会弹出如图 9.53 所示的提示信息。

选择 B2 单元格，然后再执行过程“不打开工作簿而获取其值 1”，B2:C10 区域将会产生如图 9.54 所示的取值结果。

如果要求未保存活动工作簿时也可以调用其他工作簿的值，那么过程中的文件路径必须使用绝对路径。假设工作簿保存在“D:\人资料事”路径下，要引用其值可使用以下代码：

```
Sub 不打开工作簿而获取其值 2()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  If Len(Dir("D:\人事资料\人事报表.xls")) > 0 Then
  取值 "D:\人事资料", "人事报表.xls", "人事资料", "A1:C42"
  End If
End Sub
```

选择 A1 单元格，然后执行以上过程，在 A1:C42 区域将产生如图 9.55 所示结果。

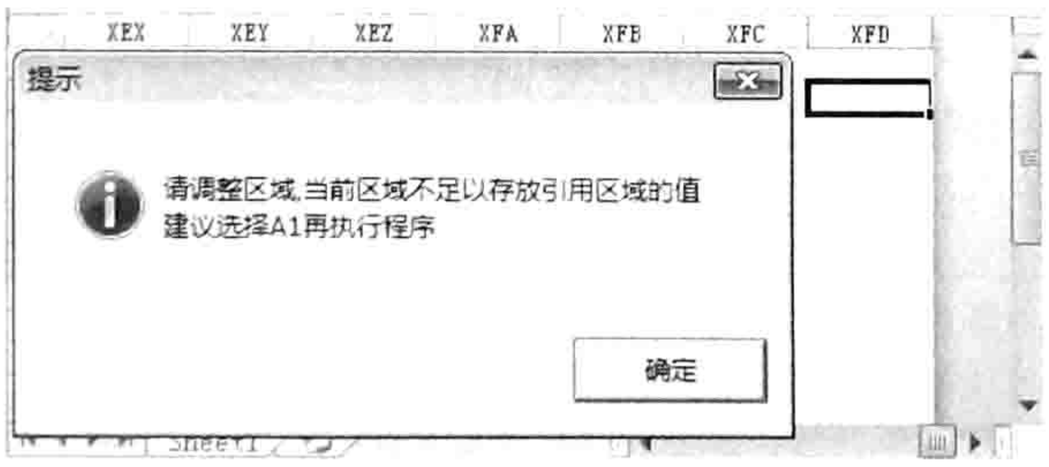


图 9.53　区域中无法存放目标数据时的提示

	A	B	C
1			
2		部门	人数
3		生管	5
4		企划	2
5		仓库	2
6		人事	1
7		总务	2
8		保安	3
9		财务	2
10		生产	50

图 9.54　引用 2 列数据

	A	B	C
1	姓名	性别	工号
2	赵	女	7922
3	钱	女	4015
4	孙	女	1836
5	李	男	3718
6	周	女	2077
7	吴	女	2197
8	郑	女	8275
9	王	女	4584
10	冯	女	7801

图 9.55 引用 3 列数据

思路分析：

Excel VBA 没有提供不打开工作簿而引用其数据的办法，但是直接在单元格中录入公式可以引用未被打开的工作簿中的值，只要公式中的路径正确即可。因此本例在过程“取值”中通过 Range.FormulaArray 属性在区域中写入公式，当公式将目标数据提取出来后，再将公式转换成值。

本例的过程“取值”是一个带参数的通用程序，其他过程可以调用此过程去提取未打开的工作簿中的数据，只要对过程“取值”的 4 个参数正确赋值即可。

带有参数的过程不允许直接执行，只能通过其他过程调用。

语法补充：

（1）Range.FormulaArray 代表单元格对象中的数组公式，本例通过它向区域中写入公式，从而引用未打开的工作簿中的数据。

（2）ThisWorkbook.Path 代表代码所在工作簿的路径，不包含文件名称。如果改用 ThisWorkbook.FullName 则同时包含路径和文件名称。

扩展应用：

本例中带参数的过程“取值”是为了其他过程反复调用或者方便多个过程调用而设计的，如果只需要取值一次，那么只用一个不带参数的 Sub 过程完成即可。假设要将过程“取值”和“不打开工作簿而获取其值 2”合并为一个过程，那么应按以下方式编写代码：

```
Sub 不打开工作簿而获取其值 3()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  On Error Resume Next
  Dim rng As Range
  Set rng = ActiveCell.Resize(42, 3)
  If Err.Number <> 0 Then
    MsgBox "请调整区域,当前区域不足以存放引用区域的值" & Chr(10) & "建议选择 A1 再执行
程序", vbInformation, "提示"
  Else
    With rng
      .FormulaArray = "='D:\人事资料\[人事报表.xls]人事资料'!A1:C42"
      .Value = .Value
    End With
  End If
End Sub
```

本例文件参见光盘：..\第九章\ 9-31 不打开工作簿而获取其数据.xlsm

9.5.6 建立指定文件夹下所有工作簿目录和工作表目录

案例要求：对指定目录下的所有工作簿，以及每个工作簿中的工作表建立目录，工作簿名称存放在 A 列，工作表名称存放在 B 列。

知识要点：Workbooks.Open 方法、ActiveWorkbook.Close 方法、Workbook.Name 属性。

程序代码：

```
'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
Sub 建立所有工作簿中的工作表目录()
  Dim path As String, Old_Name As String, RowCount As Integer
  Dim ShtItem As Integer, WbName As String
  With Application.FileDialog(msoFileDialogFolderPicker)
    If .Show = -1 Then
      path = .SelectedItems(1) & IIf(Right(.SelectedItems(1), 1) = "\", "", "\")
    Else
      Exit Sub
    End If
  End With
  Worksheets.Add
  Range("a1") = "工作簿"
  Range("b1") = "工作表"
  Old_Name = ActiveWorkbook.Name
```

```
  WbName = Dir(path & "*.xls*")
  Application.ScreenUpdating = False
  Do
    If WbName = "" Then Exit Do
    RowCount = ActiveSheet.UsedRange.Rows.Count + 1
    ActiveSheet.Hyperlinks.Add Cells(RowCount, 1), path & WbName, , WbName,
WbName
    Workbooks.Open path & WbName
    For ShtItem = 1 To Sheets.Count
      Workbooks(Old_Name).Sheets(1).Cells(RowCount + ShtItem, 2) =
Sheets(ShtItem).Name
    Next ShtItem
    ActiveWorkbook.Close False
    WbName = Dir
  Loop
  Application.ScreenUpdating = True
End Sub
```

假设在某个文件夹中有“一车间.xls”、“二车间.xls”、“三车间.xls”、“四车间.xls”4 个工作簿，执行以上过程时选择该文件夹，然后单击“确定”按钮，程序会在活动工作簿中新建一个工作表，然后在 A 列创建所选路径下的工作簿目录，在 B 列创建每个工作簿的表目录，其中工作簿目录具有超链接功能，效果如图 9.56 所示。

	A	B
1	工作簿	工作表
2	一车间.xls	
3		一月
4		二月
5		三月
6	三车间.xls	
7		三车间.xls 月
8		二月
9		三月
10	二车间.xls	
11		一月
12		二月
13		三月
14	四车间.xls	

图 9.56　工作簿与工作表目录

思路分析：

本例首先利用 Application.FileDialog 方法弹出一个对话框让用户选择路径，如果用户单击了“取消”按钮则用 Exit Sub 语句结束过程，否则记录下路径。

然后使用 Worksheets.Add 方法新建一个工作表，用于存放目录。

接着使用 Dir 函数配合 Do Loop 循环语句搜索指定目录下的所有工作簿，每找到一个工作簿就通过 Hyperlinks.Add 方法创建与工作簿的超链接。同时使用 Workbooks.Open 方法打开工作簿，并且嵌套一个 For Next 循环语句遍历其所有工作表，将工作表的名称逐一导入到 B 列中。最后关闭打开的工作簿。

获取工作簿名称和工作表名称比较简单，主要在于循环语句的嵌套应用。本案例的难点在于如何确定工作簿名称和工作表名称在活动工作表中的存放位置，本例的思路如下：

对于工作簿名称的存放位置，其列数是 1，行号由代码“ActiveSheet.UsedRange.Rows.Count + 1”决定，表示已用区域下一行的行号。每追加一个工作簿的目录，该表达式的结果会自动累加，因此不会导致多个工作簿的名称存放在同一个单元格中。

对于工作表名称的存放位置，其列数是 2，行号由工作簿名称所在行的行号加工作表的序号决定，由于工作表序号总是从 1 向上累加，因此工作表名称会存放在工作簿名称右方向下逐一延伸的区域中。

语法补充：

（1）Workbooks.Open 方法表示打开工作簿，其第 1 参数 FileName 代表需要打开的工作簿的名称，包含路径。Workbooks.Open 方法在打开工作簿时会改变活动工作簿对象，例如活动工作簿名为“A.xlsm”，使用 Workbooks.Open 方法之前代码“Sheets(1).Range("a1")”可以引用 A.xlsm 工作簿的第 1 个表的 A1 单元格，使用 Workbooks.Open 方法打开 B.xlsm 工作簿后代码“Sheets(1).Range("a1")”则只能引用 B.xlsm 工作簿中的第 1 个表的 A1 单元格。如果此时一定要引用 A.xlsm 工作簿的第 1 个表的 A1 单元格，则必须对“Sheets(1).Range("a1")”添加父对象，书写为“Workbooks ("A.xlsm") .Sheets(1).Range("a1")”，因此，在本例过程“建立所有工作簿中的工作表目录”中的循环语句之前使用以下代码记录原来的工作簿名称：

```
Old_Name = ActiveWorkbook.Name
```

然后再使用以下代码引用打开新工作簿之前的活动工作簿中的单元格：

```
Workbooks(Old_Name).Sheets(1).Cells(RowCount + ShtItem, 2)
```

（2）Workbook.Close 表示关闭工作簿，其参数 SaveChanges 代表关闭工作簿的同时是否保存工作簿，本例将参数赋值为 False 表示只关闭工作簿而不保存。

扩展应用：

本例中工作表名称是纵向摆放的，如果要横向摆放，那么应将以下代码：

```
Workbooks(Old_Name).Sheets(1).Cells(RowCount + ShtItem, 2) = Sheets(ShtItem).
Name
```

修改为：

```
Workbooks(Old_Name).Sheets(1).Cells(RowCount, ShtItem + 1) = Sheets(ShtItem).
Name
```

修改代码后执行效果如图 9.57 所示。

	A	B	C	D	E
1	工作簿	工作表			
2	一车间.xls	一月	二月	三月	
3	三车间.xls	一月	二月	三月	
4	二车间.xls	一月	二月	三月	
5	四车间.xls	一月	二月	三月	四月

图 9.57　工作簿与工作表目录 2

本例文件参见光盘：..\第九章\ 9-32 创建工作簿及工作表目录.xlsm

◆　◆　◆

第10章　编程规则与代码优化

编写代码的目的是完成工作需求，但是优秀的代码并非只要完成工作需求即可，还得利于阅读、理解、维护，以及追求执行速度。

本章向读者展示编程过程的一些规则和代码优化手法，这些规则并不是必须要遵守的，但是按规则编写的代码会更完善。

10.1　代码编写规则

由于工作中的需求经常会变化，因此 VBA 代码也会经常被修改。一段优秀的代码必须是可读性强、可维护性强，并且让人能快速看懂的代码，否则后期维护会相当困难。

本节提供若干编写代码的规则，读者在编写代码时应遵循这些规则。

10.1.1　对代码添加注释

不管程序开发人员和终端用户是否是同一个人，都完全有必要对代码进行注释。代码具有清晰的含义注释才有利于阅读和修改。

1. 添加注释的用途

添加代码注释主要有以下两个作用。

（1）描述每句代码的含义

编程时应对每一句代码添加含义描述，让阅读者能快速了解代码的功能。

（2）整段程序的介绍

对于大中型程序需要声明开发者的姓名、版权、版本号、开发日期、更新日期、更新内容及本程序功能说明、注意事项等，有利于用户了解程序的设计思路、修改记录、注意事项。

2. 添加注释的方法

对代码添加注释有 3 种方式：Rem 语句、单引号（也称撇号），以及工具按钮法。

（1）利用 Rem 添加程序注释

Rem 语句用来在程序中添加注释。其语法如下：

```
Rem comment
```

其中 comment 是注释内容，与 Rem 之间有一个空格。

VBA 会将注释行显示为绿色以示区别。

在以下代码中，过程上方一行即为注释，用于阐述当前过程的功能。

```
Rem 程序功能：获取 Excel 用户名称
Sub UserName()
    MsgBox Application.UserName
End Sub
```

在任何代码窗口录入以上代码，Rem 注释行都呈绿色显示，如图 10.1 所示。

注释可置于代码的上方或者右方，如图 10.2 所示是注释放在代码上方的效果，如图 10.3 所示

则是注释放在代码右方的效果。

图 10.1　为过程添加功能注释

图 10.2　在代码上方添加含义注释

（2）利用半角单引号添加程序注释

半角单引号后面的字符都是注释，半角单引号与注释之间不需要空格。

如图 10.4 所示是将注释放在代码上方和右方的两种效果。

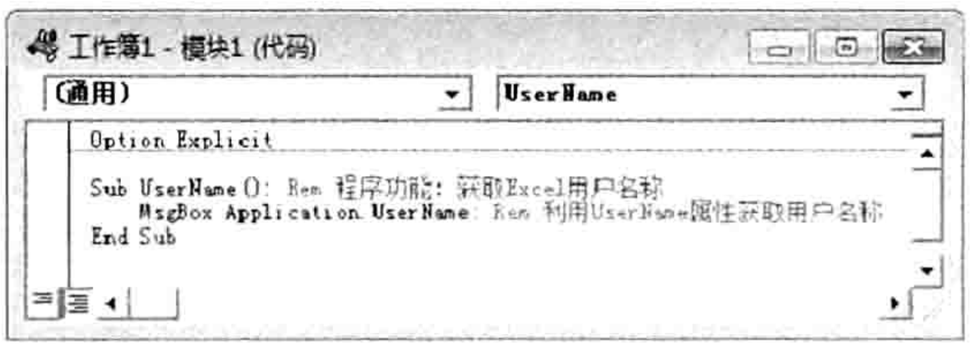


图 10.3　在代码右方添加含义注释

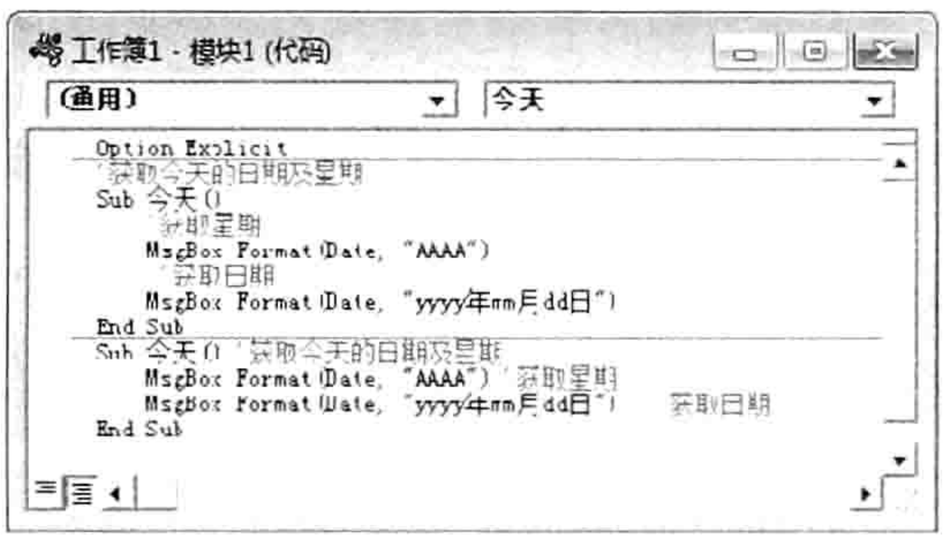


图 10.4　使用单引号添加注释

在添加注释时，如果注释与代码呈上下行关系，则注释尽量与代码对齐。如图 10.4 所示的第一个过程，过程名称未缩进，那么代码的注释也不需要缩进；过程中间的代码缩进了 4 个单位，那么注释也相应缩进 4 个单位。

如果代码与注释呈前后关系，则应在代码与注释之间应添加几个空格。

虽然以上规则并非必须执行，但遵循本规则会使代码看起来更美观，给阅读者带来便利。

（3）用工具栏按钮将代码批量转换成注释

在“编辑”工具栏中有“设置注释块”和“解除注释块”按钮，分别用于批量添加注释及批量解除注释。图 10.5 中鼠标指针指向的图标即为设置注释的专用工具，其右边第一个按钮则用于解除注释。

使用工具设置注释的步骤为：选择代码（可以是多行，甚至多个过程）→单击“设置注释块”按钮。

该工具按钮添加的注释是以行为单位的，无法从代码中间某个字符开始设置注释。所以选择代码时只需要定位于该行任意位置再单击“设置注释块”按钮即可。

例如以下过程用于获取今天的日期，代码中提供了 3 种日期格式，现暂时仅需要第 3 种，但不排除以后使用前两种格式的可能，因此不宜删除前两行代码，而是将这两行暂时不用的代码转换成注释，以后需要时再从注释转换成代码才是最佳的选择。

```
Sub 今天()
  MsgBox Format(Date, "DDDD yyyy 年 mm 月 dd 日")
  MsgBox Format(Date, "DDD yyyy 年 mm 月 dd 日")
  MsgBox Format(Date, "AAAA yyyy 年 mm 月 dd 日")
End Sub
```

用鼠标选择两行代码的任意位置，然后单击“设置注释块”按钮，效果如图 10.6 所示。

图 10.5 设置注释及解除注释的按钮

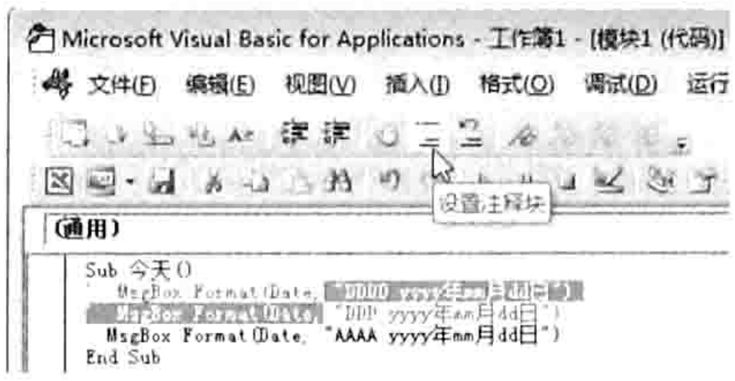


图 10.6 批量设置注释

后续需要该代码时，无须重新编写代码，选择目标代码行的任意位置，然后单击“解除注释块”按钮即可。

3. 设置注释在调试代码中的作用

调试代码是每个程序员的必然经历。

鉴于测试环境或者数据与实际工作中可能存在差异，在调试代码时部分代码不需要执行，但在调试时又不宜直接删除它，在代码前加一个单引号使其暂时不执行才是上策。

例如以下代码用于检测 B2:B10000 区域中的成绩，如果成绩单元格为空白则提示用户输入成绩，如果成绩小于 60 分则对该单元格填充红色背景。

```
Sub 标示小于60分的成绩()
  For i = 2 To 10000 '遍历B2:B10000
    IF Len(Cells(i, 2)) = 0 Then MsgBox Cells(i, 1).Address(0, 0) & "没有录入成绩"
    IF Cells(i, 2) <60 Then Cells(i, 2).Interior.ColorIndex = 3
  Next
End Sub
```

在测试时，往往习惯于随意输入几个数据，从而造成 B2:B10000 区域中存在大量的空白单元格，那么执行此代码时将弹出无数个对话框。为了避免此问题，可以选择第二行代码并将它设置为注释，从而暂时不执行该语句，调试完后正式投入工作中使用时再解除注释。

另外，在调试具有破坏性的代码时也有必要将部分代码转换成注释，从而仅测试其他代码。例如一段过程中的部分代码具有清除数据、图片或者批量添加条件格式等功能，在调试代码时应禁止这些代码执行，仅运行其他代码即可。

4. 对插件添加声明

对于大中型程序或者利用 VBA 设计 Excel 插件时，有必要对工具添加详细说明，包括功能、版本、更新项目等。通常这些说明信息置于程序顶端。

例如对于从身份证号码中获取信息的代码，可以在前面代码添加如图 10.7 和图 10.8 所示的描述信息。

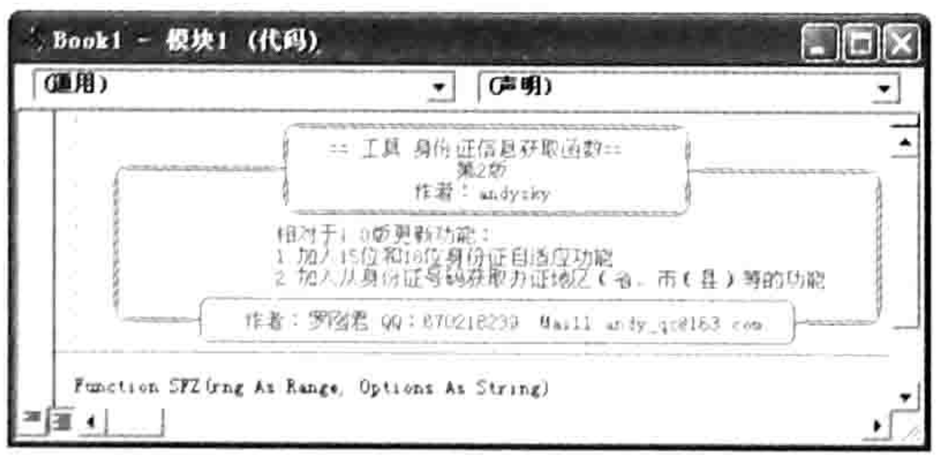


图 10.7 对身份证函数插件添加功能与版本声明 1

图 10.8 对身份证函数插件添加功能与版本声明 2

10.1.2 长代码分行

VBA 代码每行允许存放 1～1024 个字符。

然而为了提升阅读和修改的便利性，一行尽量不要超过 200 个字符，或者尽量让一行代码不超过当前屏幕的可见宽度，使用户查看代码时不需要拖动滚动条。

对代码进行分行的方法如下：

从整行代码的换行处插入空格，再输入下画线“_”，最后按<Enter>键即可。

假设某行代码为“AAAABBBB”，需要在第四个“A”之后截断为两行，那么代码分行前后的对比效果如图 10.9 所示。

图 10.9 代码换行

以下代码用于获取计算机中所有磁盘的盘符、序列号、总空间和剩余空间：

```
Sub 磁盘信息()
  Dim 盘符 As String, 类型 As String
  For i = 1 To 26
    On Error Resume Next
盘符 = Mid("ABCDEFGHIJKLMNOPQRSTUVWXYZ", i, 1)
    Select Case CreateObject("Scripting.FileSystemObject").GetDrive( 盘 符 &
":").DriveType
    Case 0: 类型 = "无法识别"
    Case 1: 类型 = "移动磁盘"
    Case 2: 类型 = "固定磁盘"
    Case 3: 类型 = "网络磁盘"
    Case 4: 类型 = "光盘 DVD"
    Case 5: 类型 = "虚拟磁盘"
    End Select
    If Err.Number = 0 Then
      msg = msg & CreateObject("Scripting.FileSystemObject").GetDrive(盘符& ":").
DriveLetter & " " &类型& " " & CreateObject("Scripting.FileSystemObject").
GetDrive(盘符& ":").SerialNumber & " " & CreateObject("Scripting.FileSystemObject").
GetDrive( 盘 符 & ":").TotalSize / 1024 & " " & CreateObject("Scripting.
FileSystemObject").GetDrive(盘符& ":").FreeSpace / 1024 & Chr(10)
    End If
  Next i
  MsgBox msg
End Sub
```

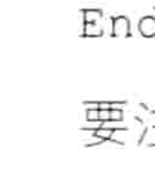
代码的具体含义本章不作详述，此处仅需要理解长代码如何分行显示即可。

以上代码中 MsgBox 语句的代码过长，超过两屏的宽度才能显示完成，这非常不利于阅读。可以通过以下方式转换成五行：

```
msg = msg & _
CreateObject("Scripting.FileSystemObject").GetDrive(盘符& ":").DriveLetter & " " &类型& " " & _
```

```
CreateObject("Scripting.FileSystemObject").GetDrive(盘符& ":").SerialNumber & " " & _
CreateObject("Scripting.FileSystemObject").GetDrive(盘符& ":").TotalSize / 1024 & " " & _
CreateObject("Scripting.FileSystemObject").GetDrive(盘符& ":").FreeSpace / 1024 & Chr(10)
```

截成五行后的代码不仅查看方便，对于用户理解代码也大有裨益。



本例文件参见光盘：..\第十章\ 10-1 代码分行.xlsm

如果需要从一个字符串中间截成两行，那么需要对截断后的两段字符串补齐双引号，并且添加连接符。例如以下代码中 MsgBox 函数的参数需要分行显示：

```
Sub test()
    MsgBox "123456789123456789"
End Sub
```

分行后的效果如下：

```
Sub test()
    MsgBox "123456789" _
& "123456789"
End Sub
```

要注意的是“-”之前有一个空格，分行后的两行代码需要补齐双引号。

10.1.3　代码缩进对齐

代码缩进是指在代码的前面按层次添加若干个空格，或者使用<Tab>键缩进若干个单位。

代码缩进也不是必需的，但是对于阅读代码却有莫大的帮助。在编程过程中应尽量对代码缩进，使其具有层次感。图 10.10 使用了代码缩进，从而使代码层次分明。

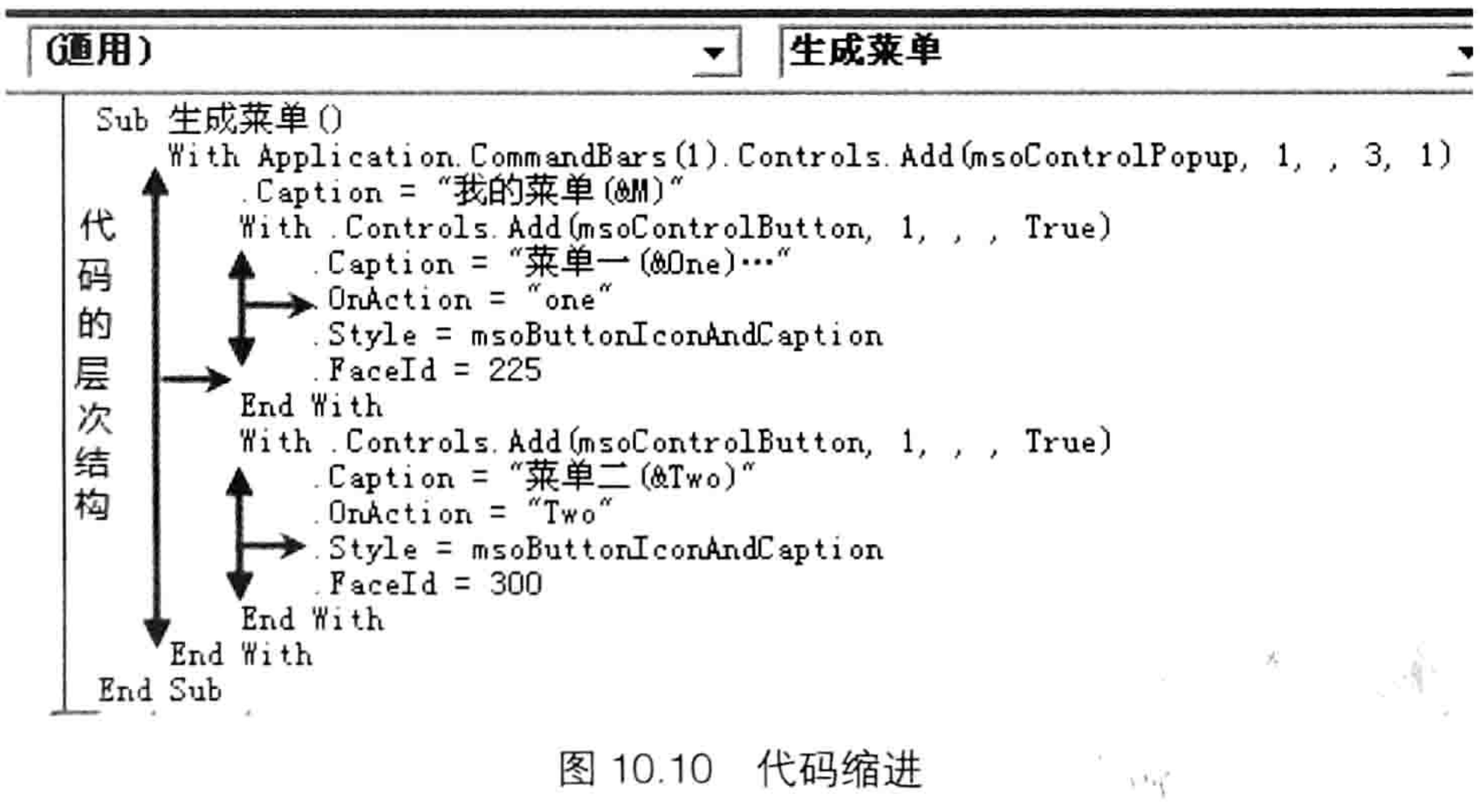


图 10.10　代码缩进

代码缩进有两种方式：单行代码手动缩进和利用工具批量缩进。

1. 单行代码手动缩进

缩进代码其实就是在代码前添加空格或者 Tab 制表符。将光标插入点移到代码的前方，按<Tab>键可以让代码一次缩 4 个单位的字符宽度，如果只需要缩进两个单位，那么可以单击菜单中的“工具”→“选项”命令，然后在选项对话框中将“Tab 宽度”由 4 修改为 2。

2. 利用工具批量缩进

当有多句代码需要缩进时，手动缩进代码的效率极差，VBA 提供了工具用于批量缩进及还原。在图 10.11 中“编辑”工具栏中、鼠标指针下方的按钮即为“缩进”菜单，单击一次产生一个制表符，它右边的一个按钮是“凸出”菜单，用于删除制表符。

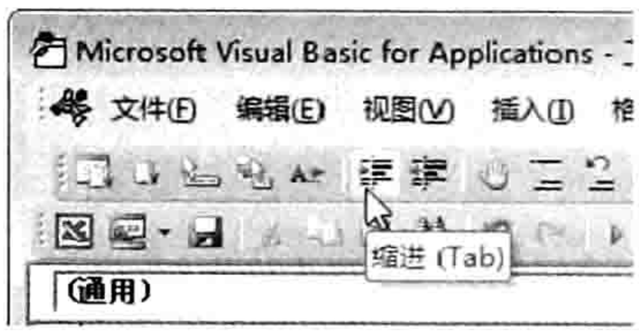


图 10.11　缩进菜单

图 10.12 是缩进前的代码，图 10.13 是缩进后的代码，显然后者更利于阅读。

(通用)　　生成菜单

```
Sub 生成菜单()
With CommandBars(1).Controls.Add(msoControlPopup, 1, , 3, 1)
.Caption = "我的菜单(&M)"
With .Controls.Add(msoControlButton, 1, , , True)
.Caption = "菜单一(&One)…"
.OnAction = "one"
.Style = msoButtonIconAndCaption
.FaceId = 225
End With
With .Controls.Add(msoControlButton, 1, , , True)
.Caption = "菜单二(&Two)"
.OnAction = "Two"
.Style = msoButtonIconAndCaption
.FaceId = 300
End With
End With
End Sub
```

图 10.12　缩进前

(通用)　　生成菜单

```
Sub 生成菜单()
    With CommandBars(1).Controls.Add(msoControlPopup, 1, , 3, 1)
        .Caption = "我的菜单(&M)"
        With .Controls.Add(msoControlButton, 1, , , True)
            .Caption = "菜单一(&One)…"
            .OnAction = "one"
            .Style = msoButtonIconAndCaption
            .FaceId = 225
        End With
        With .Controls.Add(msoControlButton, 1, , , True)
            .Caption = "菜单二(&Two)"
            .OnAction = "Two"
            .Style = msoButtonIconAndCaption
            .FaceId = 300
        End With
    End With
End Sub
```

图 10.13　缩进后

将图 10.12 的代码转换成如图 10.13 所示的代码需要使用以下步骤。

step 1　选择第 5 行到第 8 行，并单击工具栏中的“缩进”按钮。

step 2　选择第 11 行到第 14 行，并单击工具栏中的“缩进”按钮。

step 3　选择第 3 行到第 15 行，并单击工具栏中的“缩进”按钮。

step 4　选择第 2 行到第 16 行，并单击工具栏中的“缩进”按钮。

最后的效果如图 10.13 所示。

如果需要所有代码左对齐，那么可以选择所有代码后，反复单击“突出”按钮，直到所有代码左对齐为止。

CD-ROM　本例文件参见光盘：..\第十章\10-2 代码缩进.xlsm

10.1.4　声明有意义的变量名称

在本书第 5 章中已经讲述关于声明变量的数据类型的优势，事实上变量的名称也需要进行规范。虽然“ABC”或者“One”、“数量 1”等都可以作为变量名称，然而不利于用户快速阅读、理解代码。正确地命名变量可采用以下 3 种方式之一。

（1）简写的数据类型名称

用数据类型名称的简写形式作为变量名称可以让用户一目了然，根据变量名称明白变量的有效范围。例如声明一个 String 型变量可以使用以下方式：

```
Dim str as string
Dim Mystr as string
```

在过程中的任意地方看到变量 str 或者 Mystr 立即就能明白该变量是 String 型，用于储存文本字符串。

而声明一个 Integer 型的变量则可以使用以下方式：

```
Dim integ As Integer
```

声明一个工作表对象变量则用以下方式：

```
Dim sht As Worksheet
```

（2）对变量的作用进行简要描述

前一种变量命名方式注重变量类型，第 2 种命名方式则注重变量的功能。通过变量的名称描述变量的功能可大大提升代码的阅读性，例如要声明一个用于储存工作表中所有图形对象数量的变量，那么变量名称可以用“图形数量”或者“Shapes_count”。例如：

```
Dim 图形数量 As Integer
图形数量 = ActiveSheet.Shapes.Count
```

（3）两者综合

变量名称同时包含变量的功能和类型会更利于用户理解代码，不过变量名称会偏长。

假设需要声明一个用于储存图形对象数量的变量，那么可按以下方式声明：

```
Dim ShapesCount_int As Integer
```

其中“_”前面的 ShapesCount 表示当前变量的功能是储存图形对象数量，后面的 int 则表示此变量的数据数型是 Integer。本方式声明变量的优势在于可以让用户准确、迅速理解变量的含义，缺点是变量名称偏长，读者可以根据自己的喜好在以上 3 种方式中选择。

10.1.5 IF Then...End If 类配对语句的录入方式

VBA 中有很多类似于 If Then...End If 这类需要配对输入的语句，包括 With…End With、For each...Next、For...next、Do...Loop、Sub...End Sub 等。

VBA 可以对 Sub 语句自动配对，即只要输入完整的 Sub 语句后按<Enter>键，VBA 就会全自动生成 End Sub，但以 If Then...End If、With…End With 等语句只能由用户手工录入。

对于初学者，常常因为代码录入不完整而产生编译错误，例如图 10.14 和图 10.15 即为常见的因忘记录入结束语句而产生的编译错误。

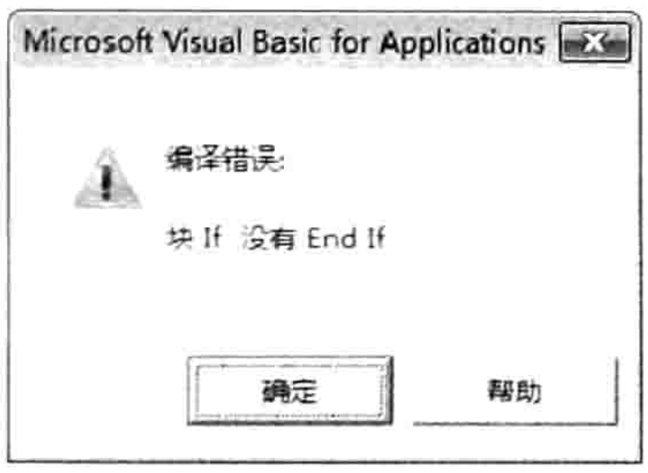


图 10.14 IF 语句录入不完整

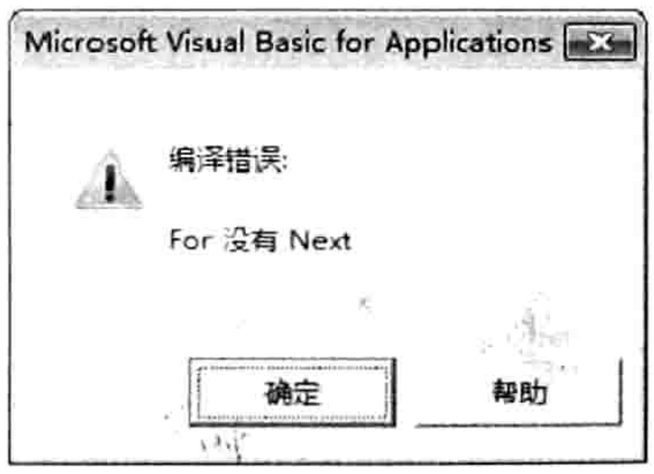


图 10.15 For 语句录入不完整

正确的录入规则是先录入起始语句，然后按两次<Enter>键，接着录入结束语句，最后再返回上一行录入中间的代码，从而避免忘记录入结束语句。

```
If Range("a1") >= 60 Then
  Range("b1") = "及格"
End If
```

基于以上规则，正确录入以上 3 句代码的步骤如下：

step 1 录入“If Range("a1") >= 60 Then”。

step 2 按两次<Enter>键，然后录入“End IF”。

step 3 按<↑>键返回第 2 行，接着录入代码“Range("b1") = "及格"”。

对于其他需要配对的代码同样应采用以上步骤录入，从而避免忘记录入结束语句。

10.1.6 录入事件代码的方式

多数事件过程拥有两个以上的参数，而且事件过程的名称偏长，为了避免用户输入错误导致程序无法执行，VBA 提供了自动产生代码的工具——对象列表与过程列表。

不管事件过程的名称有多长，只要选择对象名称后再选择过程名称即可自动产生完整代码。例如在 ThisWorkbook 代码窗口中单击对象窗口将看到名为“Workbook”的对象名称，如图 10.16 所示，而单击右方的过程列表则可以看到所有工作簿事件的过程名称，效果如图 10.17 所示。当分别单击对象名称和过程名称后，事件过程的程序外壳就会自动产生在代码窗口中。

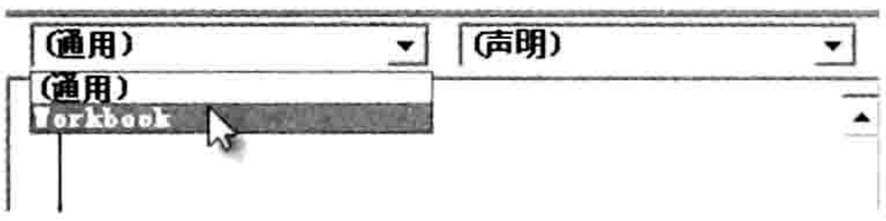


图 10.16 选择对象名称

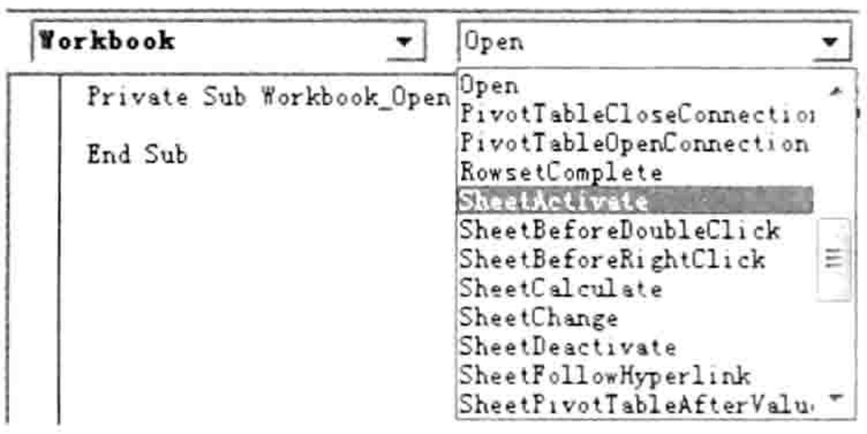


图 10.17 选择过程名称

10.1.7 录入属性与方法的技巧

VBA 有数千个属性与方法，任何人都不可能准确地记得所有单词，而手工录入代码的出错率偏高，录入速度也偏慢，在此情况下借助“属性与方法列表”录入代码才是最佳选择。

借用“属性与方法列表”录入代码的重点在于掌握库，顶层的对象即库。

Excel VBA 有 Excel、Office、Stdole、VBA、VBAProject 共 5 个库，常用的库是 Excel 和 VBA 两个，其中 Excel 可以用 Application 代替，因此只要熟记 Application 和 VBA 两个单词，常用的函数名称、对象名称、方法名称和属性名称都可以自动产生。

假设要录入 AppActivate 语句，如果不记得它的完整拼写方式或者担心记错，又或者为了避免手误，不管基于何种原因，都应该通过 VBA 的库配合“属性与方法”列表来录入单词。

例如要录入激活其他应用程序窗口专用的 AppActivate 语句，此单词比较长，不利于记忆，也不便于输入，在实际工作中可以借用“自动列出成员”来完成。具体步骤如下。

step 1 首先思考激活其他应用程序窗口专用的 AppActivate 语句应该是 Application 库还是 VBA 库，如果记得库名称就直接录入库名称加一个小圆点，然后从“属性与方法列表”中选择 AppActivate，如果不确定是哪一个库，那么只能一个个地测试，例如先输入“Application.”，然后查看其列表，结果如图 10.18 所示。

显然 Application 没有名为 AppActivate 的属性，因此再执行第 2 步。

step 2 删除“Application.”，然后录入“VBA.”，可以看到列表中有 AppActivate 存在，效果如图 10.19 所示。

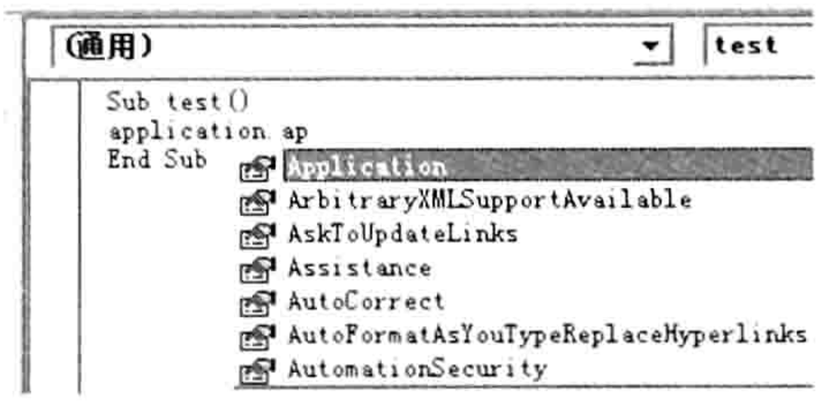


图 10.18 调用 Application 库的属性与方法

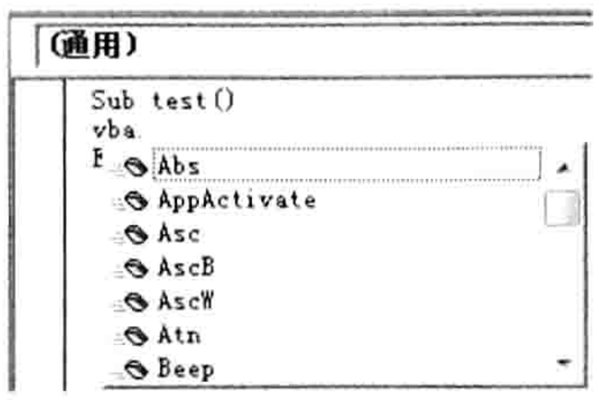


图 10.19 调用 VBA 库的属性与方法

step 3 按<↓>键，从列表中选择目标 AppActivate，然后按<Tab>键，代码将自动产生在代码窗口中。此方式录入代码比较快捷，而且不会产生拼写错误。

要引用工作簿时，不知道该用“Workbooks”、“Workbook”还是“Workbuks”，那么可以先输入“Application.Wo”，然后查看列表中的正确拼写方式。图 10.20 中已经表明了正确的书写方式是 Workbooks：

要录入工作表函数 Transpose，由于单词太长可能拼写错误，那么同样可以借助“属性与方法列表”来完成。由于函数 Transpose 的父对象是 WorksheetFunction，因此在代码窗口录入“WorksheetFunction.”，VBA 会自动产生所有工作表函数的名称，继续录入“tr”，那么完整的单词拼写 Transpose 会自动产生，效果如图 10.21 所示。

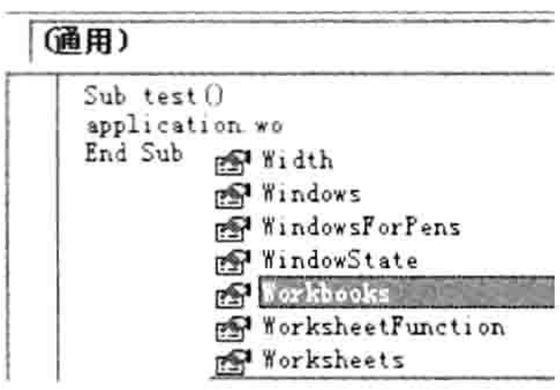


图 10.20 从列表中找查“Wo”开头的属性与方法

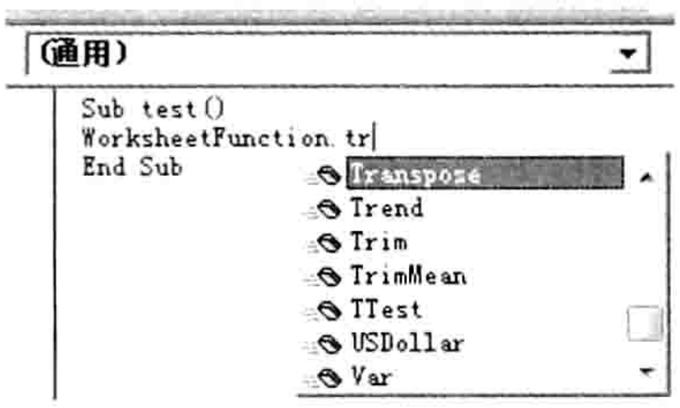


图 10.21 借助 WorksheetFunction 录入 Transpose

如果单词 WorksheetFunction 的拼写也不记得，那么可以通过 Application 库来产生 WorksheetFunction，再通过 WorksheetFunction 产生 Transpose。图 10.22 即为借助 Application 库录入单词 WorksheetFunction。

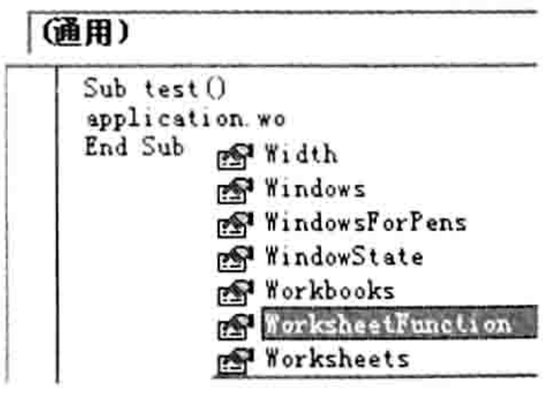


图 10.22 借助 Application 录入 WorksheetFunction

10.1.8 无提示的词组的录入技巧

Excel VBA 提供的“属性与方法列表”仅对部分对象有效，部分对象不会产生该列表。例如 Range("a1")、Sheets、Workbooks、Workbooks(10)、Cells、Columns、Application、ActiveCell 等对象能产生“属性与方法列表”，但是 Cells(1,1)、Worksheets(1)、Columns(2)、ActiveSheet、Shapes(1)、Charts(1)、ChartObjects(1) 等对象没有“属性与方法列表”。

对于没有“属性与方法列表”的对象，要录入它们的属性与方法时可以借助变量产生“属性与方法列表”。例如忘记了 Worksheet.Delete 方法的具体书写方式，输入“activesheet.”后又无法调用“属性与方法列表”供参考，此时可以按如图 10.23 所示的方式借助变量调用“属性与方法列表”。

当变量太长时，要正确输入变量也是一件不容易的事情，不过 Excel VBA 提供了一个快捷键帮助用户快捷录入变量，具体操作步骤如下。

step 1 声明一个较长的变量：

```
Dim ShapesCount_Inte As Integer
```

step 2 调用变量时，只需要输入前两个字母，然后按<Ctrl+J>组合键，VBA 会自动弹出“sh”开头的所有变量名称、对象和常量名称，效果如图 10.24 所示。

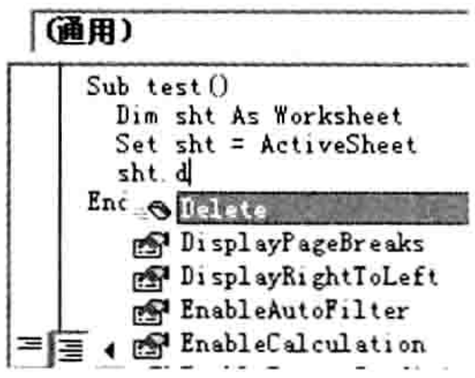


图 10.23 通过变量调用“属性与方法列表”

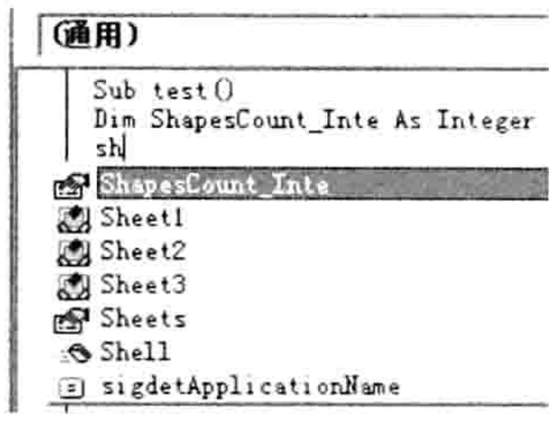


图 10.24 按<Ctrl+J>组合键调用变量名称

10.1.9 善用公共变量

如果过程中的某个表达式的运算结果需要在“过程二”中调用，那么可以将“过程一”的运算结果写入某个辅助单元格或者写入注册表中，“过程二”中需要使用该值时直接读取这个辅助单元格的值或者读取注册表即可。

尽管以上方式可以达成需求，但是代码的执行效率并不太理想，而且辅助区与注册表还可能会被意外删除，而最佳方式是借用公共变量作为容器，将值传递给所有模块中的过程。

```
Public Sums As Long
Sub 过程一() '赋值
  '汇总第一个工作表的所有数值
  Sums = WorksheetFunction.Sum(Worksheets(1).UsedRange)
End Sub
Sub 过程二() '取值
  MsgBox Sums '提取变量的值
End Sub
```

在以上代码中，执行“过程一”后，变量 Sums 即已被赋值，只要工作簿不关闭，那么任何模块都可以直接调用变量 Sums 的值，相比在“过程一”中将汇总值存入单元格，在“过程二”中读取单元格的值要快得多，而且避免了意外删除的可能。

10.1.10 将比较大的过程分为多个再调用

VBA 允许一个过程存入上千条代码，然而过于庞大的过程不利于阅读和维护。当一个过程的代码超过一屏时，应将过程按作用分为多个子过程，再在主过程中分别调用。

```
Sub 主程序()
  MsgBox "1"
  MsgBox "2"
  MsgBox "3"
End Sub
```

假设以上过程需要分为多个子过程，那么可以按以下方式进行：

```
Sub 主程序()
  Call 过程一
  Call 过程二
  Call 过程三
End Sub
```

```
Sub 过程一()
  MsgBox "1"
End Sub
Sub 过程二()
  MsgBox "2"
End Sub
Sub 过程三()
  MsgBox "3"
End Sub
```

但是 With...End With 语句或者 If Then...End If 语句跨两屏显示时则不宜截断，例如“With”在第一屏中，而“End with”在第二屏中。

10.1.11 减少过程参数

VBA 中的 Sub 过程和 Function 过程都支持 200 个以上的参数，但为了便于维护和理解，过程尽量不要超过 5 个必选参数。

在本书的第 11 章中会讲述过程中与参数相关的知识。

10.1.12 使用 DoEvents 转移控制权

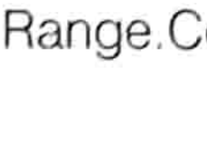
VBA 过程在执行期间不能执行其他操作，否则极有可能会导致程序崩溃。假设一段 VBA 代码的执行时间超过 3 秒钟，那么应在过程中添加 DoEvents 转移控制权，允许用户执行其他操作。此外，DoEvents 还可以避免代码进入死循环而导致 Excel 崩溃。

10.1.13 使用常量名称替代常数

VBA 每一个内置的常量名称都对应一个数值，调用常量时尽量写常量的名称而不调用它对应的数值，从而便于识别和理解。

例如 Msgbox 函数的第 2 参数使用 vbYesNo 和使用数值 4 的功能一致，但是使用 vbYesNo 能让人瞬间明白它的含义是产生“是”和“否”两个按钮，而数值 4 则无法让人理解，只能查询帮助之后才能理解代码。

再例如 Range.End 属性的参数使用 xlDown 和使用-4121 时含义是一致的，但是前者可以让人顾名思义，瞬间明白它是表示“向下”，而数值-4121 则让人不明就理。



本例文件参见光盘：..\第十章\ 10-3 使用内置常量名称代替数值.xlsm

10.1.14 尽可能兼容 Excel 2003、2010 和 2013 版本

在目前阶段，Excel 多个版本并存，编写代码时应该尽可能考虑代码的兼容性。

兼容性主要体现在 3 个方面：对象、方法、属性和函数的数量变化，工作表行数与列数变化，用功能区替代传统菜单。

1. 对象、方法、属性与函数的数量变化

Excel 2010 相对于 Excel 2003 增加了很多对象、方法、属性及函数，如果代码中涉及这些新的对象、方法、属性及函数，那么使用低版本 Excel 的用户将无法正常执行代码。例如 Excel 2003 以上的版本才有的 ColorScaleCriteria 对象（色阶条件格式）、Sumifs（工作表函数）、Range.CountLarge 属性（计算单元格数量）等都无法在 Excel 2003 中运行。再例如排序功能，Excel

2007 开始对它做了强化，可以添加 64 个条件，Excel 2003 只能使用 3 个条件。

对于以上情况，假设编写代码时在 Excel 2010 或者 Excel 2013 中完成，同时又要确保代码能在 Excel 2003、Excel 2010 和 Excel 2013 都能使用，那么应尽量只用通用的部分，对于新增的对象、方法、属性或函数都不用。

Office 升级时也并非都是添加新功能，同时还会删除部分功能，例如删除了 FileSearch 对象。因此为了提升代码的兼容性，搜索文件时应一律使用 Dir 函数，不再通过 FileSearch 对象搜索文件。

2. 工作表行列数差异

Excel 2003 的最大行数为 65 536，最大列数为 256，而新版本的 Excel 支持 1 048 576 行、16 384 列。为了提升代码的兼容性，引用 A 列最后一行时不能采用以下两种形式：

Range("A65536")——不适用于 Excel 2007 和 Excel 2010；

Range("A1048576")——不适用于 Excel 2003。

以下代码才是引用 A 列最后一个单元格的最通用的办法：

```
Cells(Rows.Count, 1)
```

其中行数既不是强制使用 65 536 也不是强制使用 1 048 576，而是通过计算得来的，因此在任何情况下使用都准确无误。

3. 用功能区替代传统菜单

Excel 从 2007 版本开始推行功能区来替代传统菜单，但是没有完全删除 CommandBars 对象，因此当需要代码在 Excel 2003 和高版本中通用时应采用传统菜单，不能编写功能区菜单来调用 Sub 过程；如果确定程序的终端用户 100%都是使用高版本的 Excel（Excel 2007、Excel 2010 或 2013），那么才能使用功能区。

10.2 优化代码

对于程序开发者的最基本要求是运算结果准确，但若想成为一名优秀的程序员，自我要求不能止于准确，还需要快捷。可以高效而准确地完成任务的代码才算是优秀的代码。

本节主要介绍优化代码的一些基本思路。

10.2.1 强制声明变量

VBA 并不要求用户必须声明每个变量，VBA 会自动为每个变量分配数据类型，这是相对于其他程序软件的一个优点，即兼容性好。然而它同时也是一个缺点，不声明变量的数据类型会让程序消耗更多的内存，相当于以牺牲效率来换取兼容性。

为了提升程序的效率，在编写代码时应尽量显式声明变量并指定其数据类型。

10.2.2 善用常量

如果某个数值或者字符串在程序中反复出现，那么应尽量声明一个常量取代该值，以后在代码中直接调用常量参与运算，而非每次都手工输入这个值。

当目标字符偏长时，使用常量可以提升输入代码的速度，也可以避免拼写错误。

10.2.3 关闭屏幕更新

在单元格中大量写入数据、删除数据、插入行或者批量插入图形对象时，每执行一句代码都会更新屏幕一次，而更新屏幕需要时间。为了提升代码的执行效率，可以在执行这些操作之前关闭屏

幕更新，执行完成后再恢复更新。

以下两句代码分别表示开启更新和关闭更新：

```
Application.ScreenUpdating=True
Application.ScreenUpdating=False
```

以下两个过程的功能一致，都用于隐藏工作表中的所有偶数列：

```
Sub 隐藏偶数列()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Dim tim1 As Date, tim2 As Date
  tim1 = Timer
  For Col = 1 To Columns.Count
    If Col Mod 2 = 0 Then Cells(1, Col).EntireColumn.Hidden = True
  Next
  tim2 = Timer
  MsgBox Format(tim2 - tim1, "程序执行时间为：0.00秒"), 64, "时间统计"
End Sub
Sub 隐藏偶数列2()
  Dim tim1 As Date, tim2 As Date
  tim1 = Timer
  Application.ScreenUpdating = False
  For Col = 1 To Columns.Count
    If Col Mod 2 = 0 Then Cells(1, Col).EntireColumn.Hidden = True
  Next
  Application.ScreenUpdating = True
  tim2 = Timer
  MsgBox Format(tim2 - tim1, "程序执行时间为：0.00秒"), 64, "时间统计"
  End Sub
```

过程“隐藏偶数列”未并关闭更新，因此代码的执行时间远远超过“隐藏偶数列2”，读者可以在任意工作簿中测试两个代码，然后比较两者的执行时间差异。

本例文件参见光盘：..\第十章\ 10-4 并闭屏幕更新提升工作效率.xlsm

10.2.4 利用 With 减少对象读取次数

在 VBA 中引用对象需要花费一定的时间，如果引用多级对象（同时列出父对象与子对象）则需要使用更多时间。例如引用 ThisWorkbook.Sheets(1). Range ("a1")的时间大于引用“Range ("a1")的时间。如果在循环语句中反复引用同一个对象，那么程序会浪费不少时间。

With 语句可以在一定程度上减少这种浪费，让多次引用同一个对象时既减少书写时间又减时代码的执行时间。例如以下代码中引用了 5 次 A1 单元格的字体对象：

```
Sub 设置字体()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
 Range("A1").Font.Name = "黑体"
  Range("A1").Font.FontStyle = "加粗倾斜"
  Range("A1").Font.Size = 11
  Range("A1").Font.Underline = xlUnderlineStyleNone
  Range("A1").Font.Color = 192
End Sub
```

借用 With 语句可以简化代码的书写方式，同时也提高代码的执行效率：

```
Sub 设置字体()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  With Range("A1").Font
    .Name = "黑体"
    .FontStyle = "加粗倾斜"
    .Size = 11
    .Underline = xlUnderlineStyleNone
    .Color = 192
  End With
End Sub
```

以上两段代码的书写时间明显有比较大的变化，但是单独执行两段代码无法感受两者在效率上的差异，如果将代码放在循环语句中，反复执行上万次后就能感受两者的差异了。

本例文件参见光盘：..\第十章\10-5 使用 With 提升代码的书写效率和执行效率.xlsm

10.2.5 利用变量减少对象读取次数

如果某个对象在一个过程中多次出现，应考虑用一个变量来替代该对象，因为变量存于内存中，VBA 读取内存中的值远远快过读取对象的值。

```
'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
Sub 对小于 B1 的单元格填充背景 1()
  Dim tim1 As Date, tim2 As Date, rng As Range
  tim1 = Timer
  For Each rng In Range("A1:A20000")
    If rng > [b1] Then rng.Interior.ColorIndex = 3
  Next
  tim2 = Timer
  Range("b2") = Format(tim2 - tim1, "0.00 秒")
End Sub
Sub 对小于 B1 的单元格填充背景 2()
  Dim tim1 As Date, tim2 As Date, rng As Range, MyValue As Byte
  tim1 = Timer
  MyValue = [b1].Value
  For Each rng In Range("A1:A20000")
    If rng > MyValue Then rng.Interior.ColorIndex = 3
  Next
  tim2 = Timer
  Range("b3") = Format(tim2 - tim1, "0.00 秒")
End Sub
```

以上两个过程的功能一致，由于第 2 个过程使用了变量替代对象，它的执行效率提高了数倍。

本例文件参见光盘：..\第九章\10-6 对小于 B1 的单元格填充背景.xlsm

10.2.6 善用带$的字符串处理函数

在 VBA 中，有两套字符串处理函数，包括带“$”和不带“$”的函数，例如 Mid 和 Mid$，Left 和 Left$，Right 和 Right$。

如果使用不带“$”符号的函数计算字符串，那么 VBA 将字符串作为变体型数据进行计算，而

使用带“$”的函数时则将字符串当作 String 类型进行计算。显然变体型数据在计算时需要更多的内存空间。

例如以下两句代码，第 2 句在执行效率上会稍占优势。

```
Reault = Mid("中华人民共和国", 3)
Reault = Mid$("中华人民共和国", 3)
```

随书光盘中有比较两者效率的代码。

本例文件参见光盘：..\第十章\10-7 使用带$的函数优化程序.xlsm

10.2.7 利用数组代替单元格对象

VBA 从数组中读取或者写入的速度远远快于读取和写入对象的速度，数组是提升程序效率的常用手段。关于数组的概念和具体应用在本书的第 13 章中有详细阐述。

10.2.8 不使用 Select 和 Activate 直接操作对象

手工操作表格时，不管任何操作都是先选中对象，然后操作对象，包括删除、写入、设置格式等，因为手工操作时总是只对 Selection 生效，对其他对象无效。

VBA 操作对象时不需要 Select 和 Activate 动作，直接操作对象即可，不管目标对象是否处于选中状态。因此以下代码：

```
Range("B1").Select'选中 B1 单元格
Selection.Font.Name = "宋体"'设置选区的字体名称
```

可以修改为：

```
Range("B1").Font.Name = "宋体"'对 B1 单元格设置字体名称
```

后者的效率将高出一倍以上。

10.2.9 将与循环无关的语句放到循环语句外

循环语句中间的语句是需要反复运行的，如果某些语句与循环无关，或者说它只需要执行一次的代码，那么应该将它们放在循环体以外，从而减少代码的执行时间。

例如声明变量及以下所有语句都不宜放在循环体中：

```
If TypeName(Selection) <> "Range" Then Exit Sub
On Error Resume Next
Application.Calculation = xlManual
Application.EnableEvents = False
Application.DisplayAlerts = False
Application.ScreenUpdating = False
If ActiveSheet.ProtectContents Or ActiveSheet.ProtectDrawingObjects Then
MsgBox "工作表已保护,本程序拒绝执行！", 64, "友情提示": Exit Sub
```

10.2.10 利用 Instr 函数简化字符串判断

当需要批量执行字符串比较时，借用 Instr 函数可以大大简化代码。

```
Sub 判断输入的省份是否正确()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
```

```
  Dim MyVal As String
  MyVal = Application.InputBox("请输入省份名称")
  If MyVal <> "河北省" And MyVal <> "山西省 " And MyVal <> "辽宁省" And MyVal <>
"吉林省" And MyVal <> "黑龙江省" And MyVal <> "江苏省" And MyVal <> "浙江省" And
MyVal <> "安徽省" And MyVal <> "福建省" And MyVal <> "江西省" And MyVal <> "青
海省" And MyVal <> "甘肃省" And MyVal <> "陕西省" And MyVal <> "西藏自治区" And MyVal
<> "云南省" And MyVal <> "山东省" And MyVal <> "河南省" And MyVal <> "湖北省" And
MyVal <> "湖南省" And MyVal <> "贵州省" And MyVal <> "广东省" And MyVal <> "海
南省" And MyVal <> "四川省" And MyVal <> "重庆市" Then
    MsgBox "录入有误，请重录入"
  Else
    MsgBox MyVal
  End If
End Sub
```

以上代码执行了24次判断，效率低而且代码长。以下代码能实现同样的功能，只执行了单次运算，而且代码短小精悍：

```
Sub 判断输入的省份是否正确2()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Dim MyVal As String
  MyVal = Application.InputBox("请输入省份名称")
  If InStr("-河北省-山西省-辽宁省-吉林省-黑龙江省-江苏省-浙江省-安徽省-福建省-江西省-
山东省-河南省-湖北省-湖南省-广东省-海南省-重庆市-四川省-贵州省-云南省-西藏自治区-陕西省-
甘肃省-青海省-", "-" & MyVal & "-") = 0 Then
    MsgBox "录入有误，请重录入"
  Else
    MsgBox MyVal
  End If
End Sub
```

本例文件参见光盘：..\第十章\10-8 用 Instr 函数简化字符串判断.xlsm

10.2.11 使用 Replace 函数简化字符串连接

如果某字符串中需多次插入一个固定的字符串，在代码中多次插入字符串，不如直接录入一个不常用的生僻字来替代，最后使用 Replace 函数将该字符替换为目标字符串，此举可以简化代码，也可以减少代码的运算量。

```
Sub test1()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Reault = Application.InputBox("输入1:小学" & Chr(13) & "输入2:初中" & Chr(13)
& "输入3:高中" & Chr(13) & "输入4:大学" & Chr(13) & "输入5:博士" & Chr(13) & "
输入6:教授", "请指定代表您学历的数字", , , , , , , 1)
  If Reault > 0 And Reault < 7 Then MsgBox WorksheetFunction.VLookup(Int(Reault),
[{1,"小学";2,"初中";3,"高中";4,"大学";5,"博士";6,"教授"}], 2, False)
End Sub
```

以上过程表示弹出输入框让用户选择代表自己学历的数字，代码会根据该数字返回对应的学历。为了让信息框显示得更美观，在字符中使用了5个 chr(13)换行，效果如图 10.25 所示。

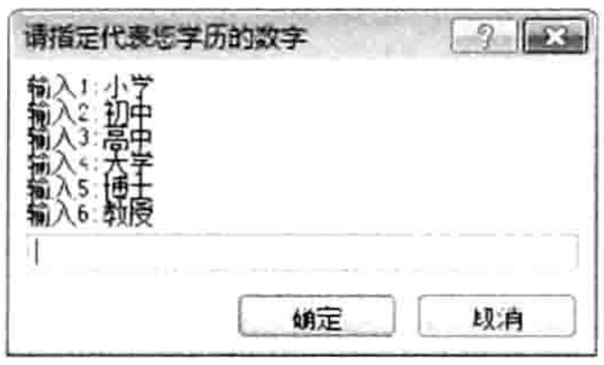


图 10.25　在字符中插入 Chr(13)的效果

以下过程实现了同等的功能，不过代码更简短，效率也更高：

```
Sub test2()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Reault = Application.InputBox(Replace("输入 1:小学@输入 2:初中@输入 3:高中@输入
4:大学@输入 5:博士@输入 6:教授", "@", Chr(13)), "请指定代表您学历的数字", , , , , , 1)
  If Reault > 0 And Reault < 7 Then MsgBox WorksheetFunction.VLookup(Int(Reault),
[{1,"小学";2,"初中";3,"高中";4,"大学";5,"博士";6,"教授"}], 2, False)
End Sub
```

本例文件参见光盘：..\第十章\10-9 使用 Replace 简化字符串连接.xlsm

◆　◆　◆

第11章 利用参数强化过程

在过程中使用参数相当于给软件添加选项，从而使软件的功能更强大，适应面更广。带参数的过程可以根据用户的实际需求决定代码的执行方式。

Excel VBA 的绝大多数函数和方法都有参数，本章的重点在于讲解过程中的参数。

11.1 什么是参数

Excel VBA 的绝大多数函数和方法都有参数，可能有一个参数也可能有多个参数，还可能有不确定数量的参数，最少为 0 个。最多为 255 个。使用 VBA 开发 Sub 过程时也可以使用参数，用好参数可以强化过程，提升过程的灵活性。

11.1.1 参数的概念与用途

参数是传递给一个过程的常数、变量或表达式，从而使过程按参数的值执行。换而言之，通过对参数赋值来控制过程的执行方式，同一个过程可以根据参数的赋值不同而产生不同的结果。

参数相当于软件中的选项，为用户提供自定义空间，从而提升其灵活性，同时简化代码。

例如在 Excel 选项中设置“默认文件位置”为常用文件夹“D:\生产表”，当单击 Excel 的“打开”按钮或者另存文件时总是自动进入“D:\生产表”文件夹，避免每次都要手工切换路径的麻烦。如果在 VBA 的过程中使用了参数，那么在调用该过程时就可以根据实际需求限定过程的参数值，从而让过程按需求执行。

在过程中使用参数的另一个优势在于简化代码。例如在 10 个 Sub 过程中都需要使用到某个功能，不过这 10 个过程的需求并非完全一致，在细节处稍有变化。假设实现这个功能需要使用 10 行代码，如果每个过程都不使用参数，那么只能在这 10 个过程中都插入这 10 行代码，即需要 100 行代码才能满足需求。事实上更便捷的做法是将这 10 行代码单独提取出来加工成一个带参数的过程，然后在这 10 个过程中各使用一句代码调用这个过程。换而言之，仅需 20 行代码可以实现原本 100 行代码才能实现的同等功能。

11.1.2 参数的语法结构

Sub 过程的语法如下：

```
[Private | Public | Friend] [Static] Sub name [(arglist)]
```

其中 arglist 代表 Sub 过程的参数。arglist 比较复杂，因为参数包含按值传递的参数、按地址传递的参数、必选参数、可选参数、指定数量的参数、不确定数量的参数。

arglist 的语法如下：

```
[Optional] [ByVal | ByRef] [ParamArray] varname[( )] [As type] [= defaultvalue]
```

其中 varname 代表参数的名称，可以是 1～255 个，As type 部分表示为参数指定类型，它和声明变量的类型时采用相同的规则。

Optional、ParamArray、（ ）和 defaultvalue 四者比较复杂，现补充介绍如下。

（1）过程的参数由 7 个部分组成，只有 varname 部分是必需的。假设在编写 Sub 过程时只指

定过程参数的 varname 部分，忽略其他的部分，那么这个过程的参数将是必选参数；如果在 varname 前面加上 Optional 关键字，那么参数将被转换成可选参数。

声明可选参数时应对参数指定默认值，格式如下：

```
Optionalvarname As type= defaultvalue
```

对于可选参数，调用过程时可以忽略参数的值，VBA 会自动取它的默认值参与运算。

（2）Optional 关键字可以将一个参数转换成可选参数，而 ParamArray 关键字可产生不确定数量的参数，其下限为 1、上限为 255。工作表函数 Sum 就有 1～255 个参数。

当过程有多个参数时，ParamArray 关键字只能用于限制最末尾的参数，而且它不能与 ByVal、ByRef 或 Optional 一起使用。ParamArray 仅对数组参数有效。

（3）参数名称 varname 右边的括号代表当前参数是数组，如果不想使用数组参数则不需要在参数后面添加括号。

（4）“= defaultvalue”部分表示对可选参数赋予默认值，它只能与 Optional 关键字同时出现，但是并非使用了 Optional 关键字就必须指定默认值。

11.2 设计带有参数的 Sub 过程

11.1 节中讲述了关于设计参数的一些语法和规则，本节通过多个案例演示必选参数、可选参数和不确定数量的参数的设计过程，从而加深读者对参数的理解。

11.2.1 必选参数

假设在有很多过程中都需要“定位 A 列最后一个非空单元格”这个功能，那么根据本书前 10 章的知识可以得到以下过程代码：

```
Sub 定位最后一个非空单元格()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  If WorksheetFunction.CountA(Range("A:A")) > 0 Then
    Dim EndRng As Range
    Set EndRng = Cells(Rows.Count, "A")
    If Len(EndRng) = 0 Then
      EndRng.End(xlUp).Select
    Else
      EndRng.Select
    End If
  Else
    MsgBox "A 列不存在非空单元格"
  End If
End Sub
```

以上过程用于定位 A 列最后一个非空单元格，可以在其他任意过程中使用 Call 语句调用这个过程。

```
Sub test()
  Call 定位最后一个非空单元格
End Sub
```

当多个过程都需要定位 A 列最后一个非空单元格时，应编写一个单独的过程来定位，其他过程中通过 Call 语句调用此过程才是上策，否则只能在每个过程中都插入过程“定位最后一个非空单元格”的代码，那么编程的效率会相当低下。

然而过程“定位最后一个非空单元格”包含了 11 句代码，却功能单一，只能定位 A 列最后一

个非空单元格。假设 100 个过程中需要定位 100 个不同列的最后一个非空单元格，那么需要编写 100 个 11 行的代码来实现吗？答案是:如果不使用带参数的过程，那么真的需要 100 个 11 行的代码，若对过程“定位最后一个非空单元格”添加一个参数，那么只需要以下 11 行代码实现定位，其他 100 个过程中各用一句代码调用过程即可。

```
'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
Sub 定位最后一个非空单元格(Col As String)
  If WorksheetFunction.CountA(Range(Col & ":" & Col)) > 0 Then
    Dim EndRng As Range
    Set EndRng = Cells(Rows.Count, Col)
    If Len(EndRng) = 0 Then
      EndRng.End(xlUp).Select
    Else
      EndRng.Select
    End If
  Else
    MsgBox Col & "列不存在非空单元格"
  End If
End Sub
```

以上过程带有一个参数，参数代表列标，当调用过程时对参数赋值为字母 C 则定位到 C 列最后一个非空单元格，如果对参数赋值为字母 ZZ 则定位到 ZZ 列最后一个非空单元格。例如以下两个过程：

```
Sub test1()
  '...更多代码...
  Call 定位最后一个非空单元格("C")
End Sub
Sub test2()
  '...更多代码...
  Call 定位最后一个非空单元格("ZZ")
End Sub
```

通过以上 3 个过程可以证明在过程中使用参数后，过程瞬间变得灵活起来。

知识补充：调用过程时可以使用 Call 语句，也可以不使用 Call 语句。当过程有参数时，如果使用了 Call 语句，则参数前后必须添加括号；如果不使用 Call 语句，则添加前后不能添加括号。例如本例中“Call 定位最后一个非空单元格("C")”可以改为“定位最后一个非空单元格 "C"”。

事实上，任何过程中都可以使用多个参数。例如前面的过程“定位最后一个非空单元格”只能定位活动工作表的单元格，如果对工作表没有限制，那么可以再追加一个参数，代码如下：

```
'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
Sub 定位最后一个非空单元格(ShtName As String, Col As String)
  On Error Resume Next
  Dim sht As Worksheet
  Set sht = Worksheets(ShtName)
  If Err.Number <> 0 Then
    MsgBox "您的 ShtName 参数赋值有误，不存在名为" & ShtName & "工作表", vbInformation
  Else
    If WorksheetFunction.CountA(sht.Range(Col & ":" & Col)) > 0 Then
      Dim EndRng As Range
```

```
    Set EndRng = sht.Cells(Rows.Count, Col)
    If Len(EndRng) = 0 Then
      EndRng.End(xlUp).Select
    Else
      EndRng.Select
    End If
  Else
    MsgBox ShtName & " 工作表的" & Col & "列不存在非空单元格"
  End If
 End If
End Sub
```

以上过程使用了 ShtName 和 Col 两个参数，分别代表工作表名称和列标，调用过程时只要对过程的两个参数正确赋值即可定位到目标工作表中指定列的最后一个非空单元格。

以下两个过程都调用了同一个过程“定位最后一个非空单元格”，但是由于对参数赋予了不同的值，因此定位的目标也不相同。

```
Sub test1()
  '...更多代码...
  Call 定位最后一个非空单元格("Sheet1", "C")
End Sub
Sub test2()
  '...更多代码...
  Call 定位最后一个非空单元格("Sheet2", "B")
End Sub
```

本例文件参见光盘：..\第十一章\11-1 必选参数.xlsm

11.2.2 可选参数

可选参数的优点是简化代码。

任何可选参数的赋值范围都必然不止一个值，而在赋值范围中往往又有一个最常用的值。可选参数的价值就在于将参数的默认值设置为那个最常用的值。假设调用过程时刚好要用到最常用的这个值，那么可以忽略该可选参数，VBA 会直接调用默认值去参与运算。

例如工作表函数 Row 用于获取指定单元格的行号，而在实际工作中用得最多的是通过 Row 函数获取公式所在单元格的行号，因此 Excel 将 Row 函数的参数的默认值设置为公式所在的单元格，从而在使用公式时允许忽略，让公式更简短。

在 VBA 中编写 Sub 过程时，在参数前添加 Optional 关键字即可让参数变成可选参数，然后再对该参数指定默认值。不过对参数指定默认值仅限于数据型参数，对象型参数只能在过程赋值，不能在声明函数时赋值。

以 11.2.1 节的过程“定位最后一个非空单元格”为例，假设将两个参数都修改为可选参数，当忽略两个参数时默认定位于活动工作表 A 列的最后一个非空单元格，那么完整代码如下：

```
'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
Sub 定位最后一个非空单元格(Optional ShtName As String, Optional Col As String =
"A")
  On Error Resume Next
  Dim sht As Worksheet
  If Len(ShtName) = 0 Then
    Set sht = ActiveSheet
  Else
```

```
    Set sht = Worksheets(ShtName)
    If Err.Number <> 0 Then
      MsgBox "您的 ShtName 参数赋值有误，不存在名为" & ShtName & "工作表",
vbInformation
      Exit Sub
    End If
  End If
  If WorksheetFunction.CountA(sht.Range(Col & ":" & Col)) > 0 Then
    Dim EndRng As Range
    Set EndRng = sht.Cells(Rows.Count, Col)
    If Len(EndRng) = 0 Then
      EndRng.End(xlUp).Select
    Else
      EndRng.Select
    End If
  Else
    MsgBox ShtName & " 工作表的" & Col & "列不存在非空单元格"
  End If
End Sub
```

按前面的分析，参数的默认值是数据时可在声明参数名称时直接赋值，参数的默认值涉及对象时则只能在过程中赋值，而上面的过程中参数 Col 的默认值是字母 A，不涉及对象，因此直接赋值；参数 ShtName 的默认值由活动工作表的名称决定，涉及对象，因此只能在过程中赋值。对 ShtName 指定默认值时，先使用 If Then 语句判断该参数的长度是否等于 0（String 类型的参数未赋值时其值为零长度的空文本），如果是 0 则使用活动工作表对象 Activesheet 去参与运算，否则以用户对变量 ShtName 所指定的值去参与运算。

假设要定位活动工作表的 A 列的最后一个非空单元格，那么调用过程时可以忽略两个参数，其代码如下：

```
Sub test1()
  '...更多代码...
  Call 定位最后一个非空单元格
End Sub
```

如果要定位 Sheet2 工作表的 B 列最后一个非空单元格，那么代码如下：

```
Sub test2()
  Call 定位最后一个非空单元格("Sheet2", "B")
End Sub
```

本例文件参见光盘：..\第十一章\11-2 可选参数.xlsm

11.2.3 不确定数量的参数

在过程的参数前面使用 ParamArray 关键字可以将参数转换为不确定数量的参数。不过 ParamArray 不能与 ByVal、ByRef 和 Optional 一同使用，而且必须将参数声明为数组参数。

假设要开发一个带有不确定数量的参数，用于加密工作表的过程，那么完整代码如下：

```
'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
Sub 加密工作表(PassWord As String, ParamArray ShtName())
  On Error Resume Next
  Dim Item, sht As Worksheet, ErrName As String
```

```
If UBound(ShtName) >= 0 Then
  For Each Item In ShtName
    Set sht = Sheets(Item)
    If Err.Number <> 0 Then
      ErrName = ErrName & Chr(10) & Item
      Err.Clear
    Else
      sht.Protect PassWord
    End If
  Next
  If Len(ErrName) > 0 Then MsgBox "以下工作表名称书写有误:" & ErrName
  End If
End Sub
```

以上过程拥有 0～255 个参数。假设只对活动工作表加密，那么可以按以下方式调用：

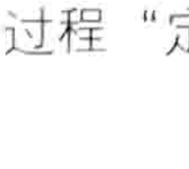
```
Sub test1()
Call 加密工作表("andysky", ActiveSheet.Name)
End Sub
```

过程“加密工作表”的参数是工作表名称而非工作表对象，因此使用“ActiveSheet.Name”。

```
Sub test2()
Call 加密工作表(123456, "Sheet1", "Sheet2", "Sheet3", "Sheet4", "Sheets")
End Sub
```

以上过程使用了 5 个参数，表示对 5 个工作表加密。假设工作簿中只有 Sheet1、Sheet2、Sheet3 这 3 个工作表，那么执行过程后这 3 个工作表将被加密为 123456，然后弹出如图 11.1 所示的信息框。

图 11.1　提示赋值错误的参数值



本例文件参见光盘：..\第十一章\11-3 不确定数量的参数.xlsm

11.3　参数的赋值方式

参数包含按位置赋值和按名称赋值两种形式，本节介绍这两种形式的特点与差异。

11.3.1　按位置赋值

当过程中有多个参数时，调用过程时可以按参数的位置对参数赋值也可以按参数的名称赋值。按位置赋值时可以忽略参数的名称，直接将值写在对应的位置即可；按参数名称赋值时则可以不在意参数的顺序，只要名称正确即可。

在 11.2.2 节中，代码“Call 定位最后一个非空单元格("Sheet2", "B")”就是按位置对参数赋值。过程“定位最后一个非空单元格”的第 1 参数是 ShtName，第 2 参数是 Col，调用该过程时只要

在对应的位置赋值即可，不用指定参数的名称。

如果某参数是可选参数，需要调用其默认值参与运算，那么可以仅使用逗号占位，忽略参数的值。例如定位活动工作表的 C 列最后一个非空单元格，可用以下三句代码之一：

```
Call 定位最后一个非空单元格(, "C")
Call 定位最后一个非空单元格(ActiveSheet.Name, "C")
定位最后一个非空单元格 , "C"
```

假设将字母 C 写在前面，然后忽略第 2 参数，VBA 会将 C 看作是工作表的名称，从而定位名为 C 的工作表的 A 列最后一个非空单元格。

如果调用过程时采用按位置赋值，那么应该在输入左括号后一边书写代码一边查看参数提示，VBA 的提示中包含了参数的名称、位置、数据类型和可选参数的默认值，效果如图 11.2 所示。

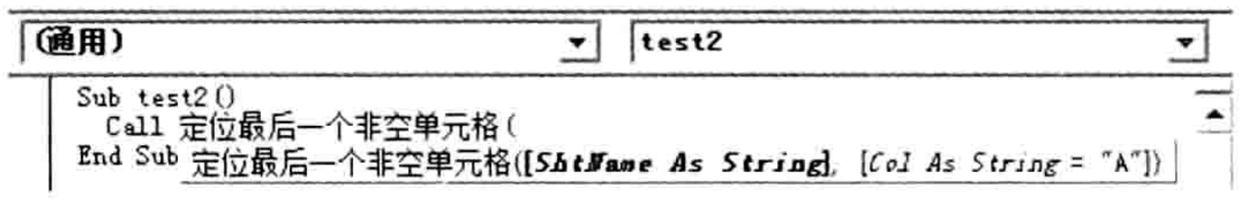


图 11.2　调用过程时提示参数名称与位置

11.3.2　按名称赋值

如果调用带多个参数的过程时采用按名称赋值，那么书写代码时可以不注重参数的顺序，只要确保参数名称正确即可。按名称对参数赋值时须遵循以下格式：

```
参数 1 名称:=值,参数 2 名称:=值……
```

仍以 11.2.2 节中的过程“定位最后一个非空单元格”为例，它有 ShtName 和 Col 两个参数，调用过程时若按名称对参数赋值可以有以下两种书写方式：

```
Call 定位最后一个非空单元格(ShtName:=ActiveSheet.Name, Col:="C")
Call 定位最后一个非空单元格(Col:="C", ShtName:=ActiveSheet.Name)
```

由于是按名称赋值，所以顺序随意打乱也没有问题，而且对于可选参数也不需要使用逗号占位。例如以上两句代码中“ShtName:=ActiveSheet.Name”是多余的，可以简化为：

```
Call 定位最后一个非空单元格(Col:="C")
```

要注意的是，对参数赋值时等号前面必须有冒号。

11.3.3　方法的参数

对象的方法大多有参数，其参数的赋值方式和对过程的参数赋值方式完全一致。

例如 Worksheets.Add 方法包含 4 个可选参数，其语法如下：

```
Worksheets.Add([Before],[After], [Count], [Type])
```

如果要求新建两个工作表，放在所有表的后面，采用按位置赋值应使用以下代码：

```
Sub 新建工作表()
  Worksheets.Add , Sheets(Sheets.Count), 2, xlWorksheet
End Sub
```

由于 xlWorksheet 是默认值，因此可以忽略，代码可简化为：

```
Worksheets.Add , Sheets(Sheets.Count), 2
```

如果采用按参数名称赋值，那么新建两个工作表并且放在所有表的后面应采用以下代码：

```
Sub 新建工作表 2()
  Worksheets.Add after:=Sheets(Sheets.Count), Count:=2
End Sub
```

由于按名称赋值时可以不在意参数的顺序，因此以上代码也可以写为：

```
Worksheets.Add Count:=2,after:=Sheets(Sheets.Count)
```

◆ ◆ ◆

第12章 编程的捷径

编程是一项比较复杂的工作，不过实际上也存在捷径，可以让编程工作变得轻松简单。

Excel VBA 的编程捷径主要有四条：录制宏、查看提示、调用笔记和使用模板，本章将一一介绍这些捷径。

12.1 录制宏

录制宏对于学习 VBA 而言是极其重要的，录制宏可以自动产生代码，既减少编写代码的时间，又为程序员提供参考，在忘记代码拼写方式的前提下仍然可以编程，完成既定的工作。

12.1.1 录制宏的目的

什么是录制宏呢？录制宏是指利用宏代码记录自己的所有操作过程。录制宏时 VBA 会将用户的每个操作记录下来，以代码的形式保存在模块中。执行模块中的代码可以实现与手工操作一致的功能，这与录音机、摄像机的工作原理相近。

不过，录制宏的目的与录音机和摄像机的目的不一致，录音机和摄像机的目的是为了再次播放，重现当时的声音或画面；录制宏的目的不是直接执行宏代码去实现相同的功能，而是仅仅作为参考，替代手工作业而已。

所谓的参考主要体现在两个方面——缩短学习时间和确保代码的准确性。

1. 缩短学习时间

学习 VBA 的前提是懂得排序、筛选、条件格式、自定义单元格格式、数据有效性、插入行、新建工作表、加密工作表、插入图形对象、查找与替换、页面设置等制表基本功能。

如果能熟练掌握这些基本操作，那么就可以通过录制宏自动产生相应的代码，不需再花费大量的时间来学习这些操作所对应的语法。

Excel VBA 的范围极广，其中能通过录制宏产生的代码约占 60%。变量、常量、循环语句、条件语句、防错语句、对话框、窗体、数组、Dictionary 对象、设计功能区菜单等必须手工书写，无法通过录制宏产生代码。

换而言之，如果对 Excel 的手工操作比较熟练，那么只需要学习 Excel VBA 40%的知识点，其他 60%可以略过，需要使用时录制宏即可得到代码。

由此可见，学会录制宏对于 VBA 爱好者而言是极其重要的。

2. 确保代码的准确性

人的记忆能力是有限的，或者说每个人能发挥出来的记忆能力是有限的，Excel VBA 中有数百个对象、数千个方法和数千个属性，没有人能将它们准确地记在大脑中。在编写 VBA 程序时，如果是纯手工编写出错率将会偏高，若采用录制宏产生代码，然后再根据实际需求修改地址、添加循环语句、判断语句则可以避免拼写失误，或者避免记错单词。

例如活动工作簿中有一个名为“财务”的工作表，要用代码将这个工作表移到最前面去，不确定使用 Workbooks（"财务"）、Worksheets（"财务"）、Sheets（"财务"）、Worksheet（"财务"）中的哪一种书写方式，那么只要录制宏马上就可以得到正确的拼写方式。宏代码如下：

```
Sub 移动工作表()
  Sheets("财务").Select
  Sheets("财务").Move Before:=Sheets(1)
End Sub
```

根据以上代码可以判断引用名为“财务”的表应书写为“Sheets（"财务"）”。宏代码的第一句表示选择名为“财务”的表，第二句表示将该表移到第一个表之前。

很显然，录制宏既能避免书写错误，又能在忘记代码时提供参考，这对于编程而言是极其重要的，是每一个 VBA 爱读者必须掌握的利器。

12.1.2　录制宏的方法

录制宏比较简单，主要包含以下 3 个步骤。

step 1　单击“开发工具”按钮或者单击左下角状态栏中的“录制宏”按钮，启动“录制新宏”对话框。

step 2　宏名保持不变，将“保存在”设置为“当前工作簿”，然后单击“确定”按钮。

step 3　手工操作要录制的步骤。

其中第 3 步相当重要。由于 VBA 会如实地录制所有的操作步骤，因此在录制宏期间应尽可能避免不必要的操作，避免产生太多冗余代码。

知识补充：在日常操作中有部分操作是不能录制的，对于这部分操作只能自己书写代码，并且将代码保存起来供日后调用，避免每次都手工编写代码。例如在工作表中查找某个值，Excel 只能录制“查找下一个”操作，不能录制“查找全部”操作，因此对于批量查找只能录制“查找下一个”过程，然后手工编写 Do Loop 循环语句，从而实现批量查找。

在此再次提示：录制宏的目的不是直接使用宏，而是从宏代码中提取有用的信息。

下面以在 B1:B10 区域创建 10 个复选框为例，通过录制宏获得代码，然后完善代码。具体操作步骤如下。

step 1　单击开发工具中的“录制宏”命令，然后单击“确定”按钮。

step 2　单击功能区中的“开发工具”→“插入”→“复选框（窗体控件）”命令，然后在 B2 单元格中按住鼠标左键不放并向右下方拖动，从而在 B2 单元格中产生复选框，如图 12.1 所示。

step 3　单击“开发工具”中的“停止录制”按钮，然后按<Alt+F11>组合键进入 VBE 界面。

step 4　双击工程资源管理器的“模块 1”，在代码窗口中将会看到名为“宏 1”的宏代码。图 12.2 中包含了插入的复选框和录制宏所产生的代码。

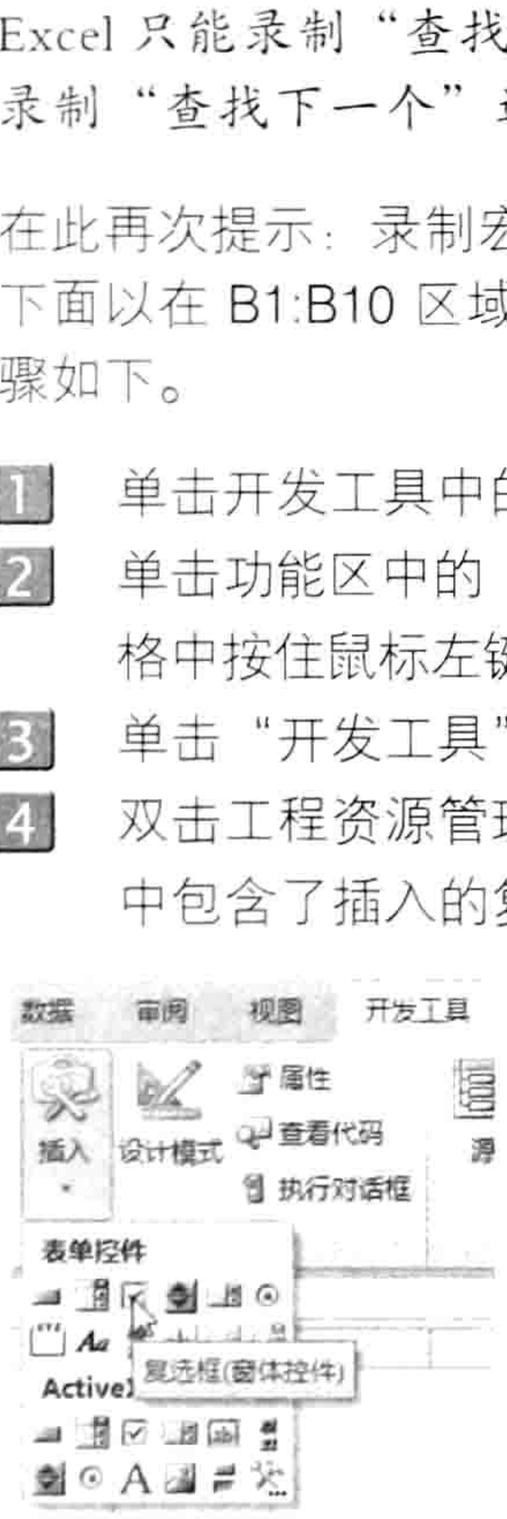


图 12.1　复选框菜单

图 12.2　复选框位置与宏代码

根据代码得知 CheckBoxes.Add 方法用于创建复选框，它的 4 个参数通常可以在帮助中找到，

不过比较特殊的是 CheckBoxes，它是一个隐藏对象（DrawingObjects、Pictures 也是隐藏的对象），VBA 的帮助中不提供与隐藏对象相关的方法和属性介绍，因此此处只能通过测试来确定它的 4 个参数的含义。例如单独修改其中的某个值，然后重新执行代码，查看生成的新复选框与第一个复框的差异，从而确定参数的含义。

step 5 将 CheckBoxes.Add 方法的第 1 参数修改为 154，然后重新执行代码，可以发现新的复选框位于第 1 个复选框的右边，其他的设置不变，这表示第 1 参数是左边距。

step 6 重复步骤 4，逐一修改其他参数，可以确定第 2 参数代表上边距，第 3 参数代表宽度，第 4 参数代表高度。

step 7 由于要求在 B1:B10 区域产生复选框，因此先将 B1 单元格的左边距、上边距、宽度与高度值替代进去，从而得到以下代码：

```
Sub 宏 2()
  ActiveSheet.CheckBoxes.Add(Range("B1").Left, Range("B1").Top, Range("B1").
Width, Range("B1").Height).Select
End Sub
```

执行以上代码后程序会新建一个复选框，复选框刚好位于 B1 单元格中，效果如图 12.3 所示，这说明前面的思路完全正确。

step 8 由于目的是在 B1:B10 区域生成 10 个复选框，因此应在上面的过程中添加变量和循环语句，并将 Range("B1")替换成变量即可。完整代码如下：

```
Sub 宏 3()
  Dim rng As Range
  For Each rng In Range("B1:B10")
    ActiveSheet.CheckBoxes.Add(rng.Left, rng.Top, rng.Width, rng.Height).Select
  Next rng
End Sub
```

step 9 执行以上代码，在工作表中将产生图 12.4 所示的 10 个复选框。

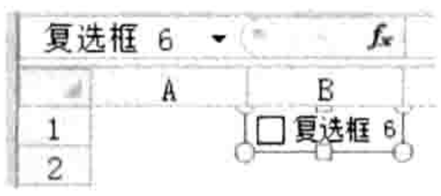


图 12.3 在 B1 单元格创建复选框

图 12.4 批量生成复选框

复选框的默认标题是由“复选框”加编号组成的，在实际工作中往往显示的是“合格”、“迟到”、“OK”之类，那么应该如何修改代码才能使所有的复选框都显示为“合格”——继续录制宏。

step 10 单击“开发工具”中的“录制宏”命令，并单击“确定”按钮启宏动录制器。

step 11 选择工作表中第一个复选框“复选框 7”，将它的标题修改为“合格”，然后单击 C1 单元格使修改生效。

step 12 单击“开发工具”中的“停止录制”命令，然后进入模块中查看代码，宏代码如下：

```
Sub 宏 4()
    ActiveSheet.Shapes.Range(Array("Check Box 7")).Select
    Selection.Characters.Text = "合格"
```

```
    Range("C1").Select
End Sub
```

即使从来没有用过复选框，但是对于懂 VBA 的基础者而言也足以明白以上 3 句代码的含义分别是选中复选框、修改选中的对象的文字、选择 A1 单元格。

根据以上 3 句代码，可以猜测修改复选框的标题需要的代码是其中的第 2 句。

step 13 将代码“Selection.Characters.Text = "合格"”复制到前面所编写的 Sub 过程中，得到以下最终代码：

```
Sub 宏 5()
  Dim rng As Range
  For Each rng In Range("B1:B10") '遍历 B1:B10
    ActiveSheet.CheckBoxes.Add(rng.Left, rng.Top, rng.Width, rng.Height).Select
    Selection.Characters.Text = "合格"
  Next rng
End Sub
```

step 14 删除工作表中的所有复选框，然后执行过程“宏 5”，结果如图 12.5 所示。

	A	B
1		☐合格
2		☐合格
3		☐合格
4		☐合格
5		☐合格
6		☐合格
7		☐合格
8		☐合格
9		☐合格
10		☐合格

图 12.5　批量生成合为“合格”的复选框

以上过程说明了录制宏的手法和重要性，尽管本书没有介绍过用代码插入复选框，但是要编写批量插入复选框同时将它们存放在指定区域的代码却可以轻松完成。同时这也说明了另一个问题——学习 VBA 的重点是不能录宏的这一部分知识点，包括变量、循环语句、防错语句、条件语句等，对于可以录制的操作则可以一概略过。

本例文件参见光盘：..\第十二章\12-1 录制宏并修改代码批量创建复选框.xlsm

12.2　查看提示

VBA 提供了大量的提示信息帮助程序员录入代码，通过提示可以加快录入代码的速度，同时也可以提高代码的准确度。

12.2.1　属性与方法列表

所有对象都有若干个属性与方法，要记住这些属性与方法是不可能的，事实上也是没有必要的，因为 VBA 提供了详细的属性与方法列表，录入代码时仅需要从该列表中选择即可。

例如要录入与 Range 对象相关的属性或者方法，那么输入 Range 对象的完整名称后加一个小圆点即可自动弹出 Range 对象的属性与方法列表，效果如图 12.6 所示。

对于 Sheets（"汇总"）这类不会弹出属性与方法列表的对象，只能通过变量来调用。

如图 12.7 所示即为通过变量调用 Sheets（"汇总"）对象的属性与方法列表。

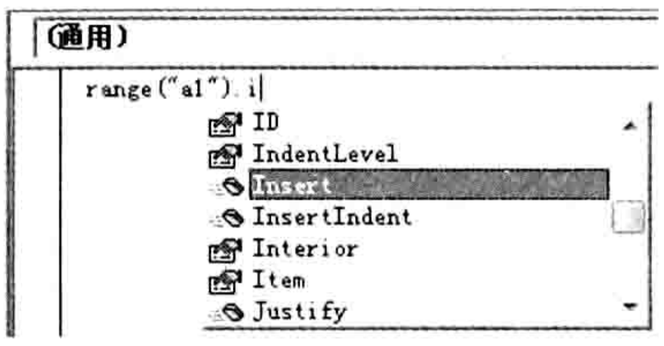


图 12.6　通过对象名称调用属性与方法列表

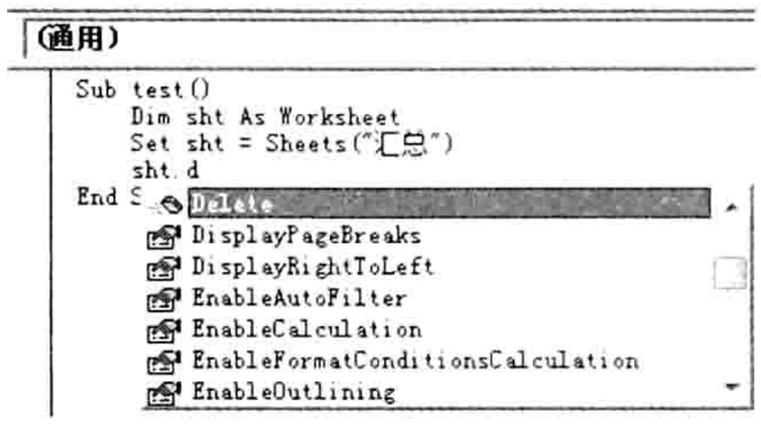


图 12.7　通过变量调用属性与方法列表

12.2.2　参数提示

VBA 对于所有参数都提供了参数提示，包括参数名称、参数位置、参数类型等，对于部分可选参数还提供了默认值。

如图 12.8 所示是 Worksheets.Add 方法的参数提示，从提示中可以得知 Worksheets.Add 方法包含 4 个参数，所有参数都是可选参数，同时还可以明确每个参数的位置。

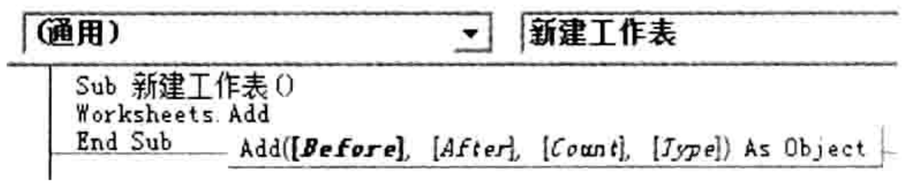


图 12.8　Worksheets.Add 方法的参数提示

如图 12.9 所示是 Range.AutoFill 方法的参数提示，从提示中可以得知 Range.AutoFill 方法有两个参数，第 1 参数是必选参数，第 2 参数是可选参数，它的默认值是 xlFillDefault。

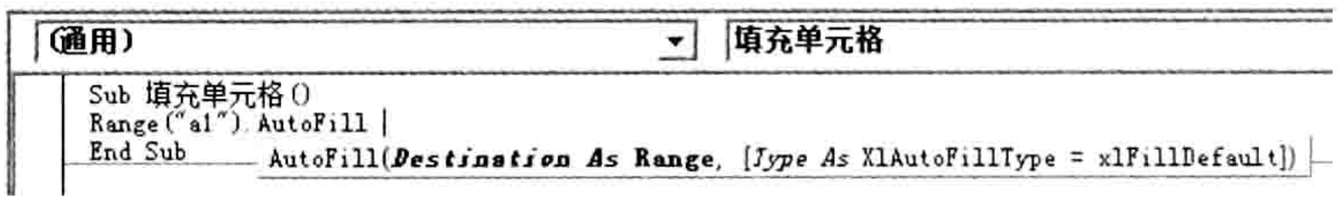


图 12.9　Range.AutoFill 方法的参数提示

12.3　调用笔记

VBA 的知识点有很多，几乎没有可能将这些知识点都记在大脑中。即使能全记住也实在没有必要，不值得耗费太多的精力去记忆这些代码。对于无法录制同时又比较常用的代码应该做好笔记，在需要时从笔记中找出来使用即可。

12.3.1　笔记的对象

学习 VBA 时，对于有用的知识点和常用代码都应该做笔记以备查找，善于做笔记者在编程时能大大提升效率。做笔记的对象主要包括综合性知识点与带有详细注释的成品代码。

1. 综合性知识点

对于综合性知识点应该做好笔记，不能过于信任自己的记忆力。

代码提速有哪些手法？防止代码错误有哪些思路？条件语句 If Then 和 Select Case 有哪些区别？Range、Cells(1,1)和[a1]这 3 种单元格引用方式的优缺点是什么？什么时候宜用公共变量，什么时候宜用私有变量？For Next 循环和 Do Loop 循环的区别和适用环境是什么？……做笔记应该针对这类综合性、帮助中没有、无法录制宏得到的知识点，对于 Instr 函数是干什么的或者 Range.Find 方法的语法是什么之类简单问题不宜做笔记，随时都可以按<F1>键瞬间找到答案。

2. 带有注释的成品代码

单句代码没有放入笔记中的价值，应该将带有完善注释的成品代码放在笔记中，当在后续的工作需要时可以直接将复制代码出来，再根据实际需求稍加修改即可。

例如制作一个二级菜单需要 30 句以上的代码，手工编写这 30 句代码需要 10～20 分钟，包含查询帮助或者翻书的时间（完全记得每个单词的拼写方式的几率很小）。如果将完整的过程“生成二级菜单”记录在笔记中，需要时仅需从笔记中复制代码到模块中，然后修改一下菜单名称即可，既能在 1 分钟完成工作又避免了拼写错误。

基于以上分析，为了提升编程的效率和准确性，在学习阶段应该对每一类应用编写一段成品代码，追加完善的注释后保存在笔记中备用。

12.3.2　笔记的记录方式

这里所讲的记录方式包含两层含义，其一是用什么记录，其二是如何完善笔记。

1. 用什么记录

做 VBA 的笔记时不能使用笔记本，应该使用 Word、记事本之类的软件，当需要使用笔记中的代码时才能瞬间复制出来。

假设记录的是成品代码，那么也可以放置在 Excel 工作簿的模块中，比将代码放在 Word 中更好理解，因为可以在工作簿中同步放置与代码相关的数据，代码配合数据能更直观地呈现代码的效果，从而便于用户理解代码和修改代码。

2. 如何完善笔记

在做笔记时，对于综合性知识点的笔记，应该尽可能全面、详细，最好是总结性、列表式的内容；对于成段的成品代码，应该有详尽的代码注释和思路注释，以及标注哪些地方是需要修改的。例如关于单元格查找的笔记，可以按如下方式记录：

```
Rem 功能：批量查找且选中目标单元格
Rem 思路：先使用 Range.Find 方法在工作表中以完全匹配的方式查找 0 值，然后利用条件语句判断是否查找成功，如果不成功则弹出提示信息，如果成功则将该单元格的地址记录在变量 FirstAdd 中，同时将该单元格赋值给变量 TargetRng，接着使用 Do Loop 循环语句配合 Range.Find 方法继续查找，每找到一个就使用 Union 方法将它合并到变量 TargetRng 中。此时，为了避免程序进入死循环，使用条件语句判断当前找到的目标单元格地址是否等于第一个目标单元格的地址，如果相等则选中变量 TargetRng 所代表的区域，最后结束过程，如果不相等则继续查找下一个目标
Rem 工作中调用时需要修改的地方：只需要修改 Range.Find 方法的第 1 参数和第 4 参数，其中第 1 参数代表查找的目标，第 4 参数代表匹配方式，使用 xlWhole 表示完全匹配，使用 xlPart 表示部分匹配
Sub 查找并选中所有0值()
  Dim rng As Range, TargetRng As Range, FirstAdd As String '声明变量
  Set rng = Cells.Find(0, , , xlWhole) '查找第一个 0 值，按完全匹配方式查找
  If rng Is Nothing Then '如果未找到(Range.Find 方法查找失败时返回 Nothing)
    MsgBox "不存在0值", vbOKOnly '提示用户
  Else  '否则
    FirstAdd = rng.Address '记录第一个目标单元格的地址
    Set TargetRng = rng    '将第一个目标单元格赋予变量 TargetRng
    Do ' 启动循环
      Set rng = Cells.Find(0, rng, , xlWhole) '查找其他所有符合条件的目标
```

```
        '将找到的目标与变量 TargetRng 合并为一个 Range 对象
        Set TargetRng = Union(TargetRng, rng)
        '如果当前目标等于第一个目标，那么选中 TargetRng，然后束过程
        If rng.Address = FirstAdd Then TargetRng.Select: Exit Sub
    Loop
  End If
End Sub
```

以上笔记是一段完整的关于批量查找的代码，笔记中有程序的源代码、思路分析、每句代码的含义注释，以及工作中要调用代码时应该修改哪些地方。这是一个比较完善的笔记，对于工作有极大的帮助，所有成品代码都应该遵循这种方式记录。

此外，为了完善笔记，还应该为笔记添加标题并分类。例如笔记中的内容分为自定义函数类、窗体类、数组类、查找与替换类、合并与拆分类、文件与文件夹处理类等。

12.4 使用工具模板

在编程过程中总有很多代码是经常使用的，基于效率和准确性原则，常用代码不应该手工录入，而应借助工具自动生成代码，这样既快捷又不至于拼写出错。

12.4.1 代码百宝箱

代码百宝箱是一款可以自动生成 VBA 代码的 VBA 插件，安装后可以在 VBE 界面创建二级菜单，单击子菜单即可生成代码。代码百宝箱中涉及数十组常用代码，包括常用的单句代码、整段代码和 API 代码。代码百宝箱的界面如图 12.10 所示。

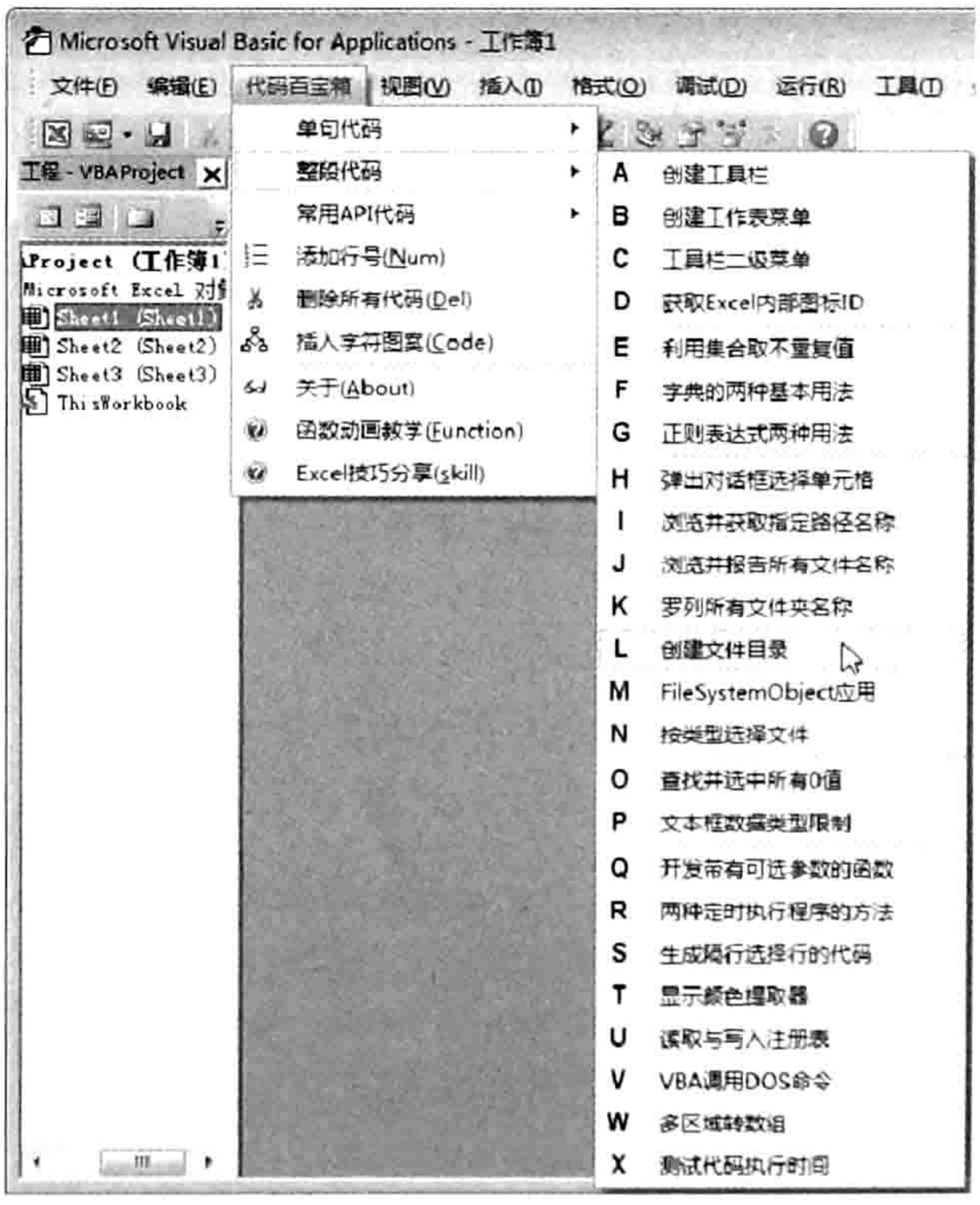


图 12.10 代码百宝箱的界面

工具的下载地址如下所示（随书光盘中也提供了安装包）：

http://pan.baidu.com/s/1hqHXnwg

在论坛中的以下帖子中有关于本工具的详细介绍：
http://club.excelhome.net/thread-1080867-1-1.html

12.4.2　开发 VBA 插件

前面所言的代码百宝箱属于 VBA 插件，开发 VBA 插件是比较复杂的工作。

本书不涉及 VBA 插件的开发教学，如果读者有兴趣可以翻看笔者的另外两本著作——《Excel VBA 程序开发自学宝典（第 2 版）》和《Excel 2010 VBA 编程与实践》，这两本书中有关于设计 VBA 插件的相关知识。

第13章 利用数组提升程序效率

数组的存在价值就是让代码提速。

数组和非数组的差异只在于数据的保存方式与读取方式不同，而操作这些数据的方法并没有不同。但是保存与读取方式上的差异却给 VBA 代码在处理数据时带来了质的飞跃，在完成相同的工作时，使用数组比非数组的效率提升几倍乃至几十倍都完全可能，数组对于 VBA 而言有着举足轻重的作用。

读者可以在掌握本章的知识后利用数组的知识去优化前面几个章节中的部分案例，可以发现很多程序都有提速的空间，掌握数组后，你会觉得完全进入了一片新的天地。

13.1 基本概念

对于提高程序效率而言，数组有着神奇的功效，不过关于数组的基础理论也不少，明白了这些理论才能在使用数组时得心应手。

13.1.1 何谓数组

数组就是连续可索引的具有相同内在数据类型的元素的集合，数组中的每一个元素具有唯一索引号。简单而言，数组就是一组相同类型的数据集合。

可以通过索引号访问数组中的每个元素，也可以随时修改数组中的任意元素。

数组支持一维到六十维，不过常用的是一维数组和二维数组。

一维数组、二维数组和区域有太多的相似点，所以通常都借助区域来理解数组，因为数组存在于内存中，有虚无缥缈之感，而区域则比较形象化。

事实上，数组和区域正是相互依存的关系，在工作中会时常将区域中的数据导出到数组中，当在数组中处理完毕后又需要将数据从数组导出到区域中。本章及后面的章节都会有大量的关于区域与数组相互转换的应用。

13.1.2 数组的特点

数组有以下 3 个特点。

1. 包含多个元素

顾名思义，数组包含一组数据，单个数据无法构成数组，也没有必要使用数组。

事实上，由于数组的功能是提速，而数据越多越能体现数组的优势。所以在工作中使用数组时往往涉及大量的待运算的数据。

2. 读取速度快

计算机在读取数据时，从软盘中读取的速度最慢，其次是光盘、U 盘、硬盘，最快的是内存。而读取 Excel 工作表单元格中的数据时相当于硬盘级速度，读取数组中的数据则是内存级速度，所以数组的运算速度快于区域的运算速度。

在使用数组时，通常将区域中的值读取到内存中，然后再针对数组执行各种运算，运算完毕后再根据需求将结果写入相应的区域或者单元格。

换而言之，使用数组就是尽量减少读取单元格的次数，替之以读取内存，从而加快代码执行效率。

3. 不能常驻内存

内存中的数据生命周期不长，不像工作表那样可以长期保留数据。

VBA 中的数组的载体其实是指数组变量或者集合（Collection），它们都会在结束过程或者关闭工作簿后自动消失，所以数组仅用于临时保存数据，在内存中处理完后需要再将数据导出到区域中。数据无法长期驻留于数组中。

13.1.3　一维数组

VBA 中的数组包含一维、二维、三维……直到六十维，常用的是一维数组和二维数组。

在理解一维数组之前，先了解一下一维区域。

1. 一维区域

一维区域是指单列多行或者单行多列的区域。如图 13.1 所示的是一维纵向区域，包含 5 个单元格；如图 13.2 所示的是一维横向区域，包含 3 个单元格。

可以利用转置函数 Transpose 将横向区域的值转换到纵向区域中，也可以将纵向区域的值转换到横向区域中。如图 13.3 所示的即为 Transpose 函数转置区域的应用结果。

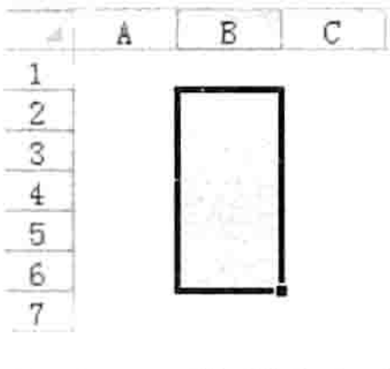


图 13.1　一维纵向区域

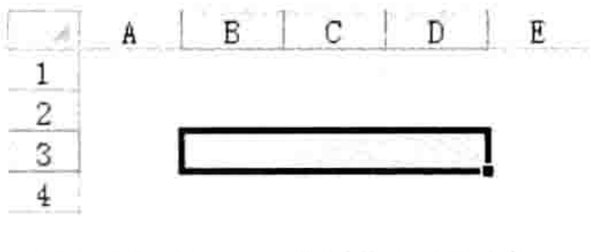


图 13.2　一维横向区域

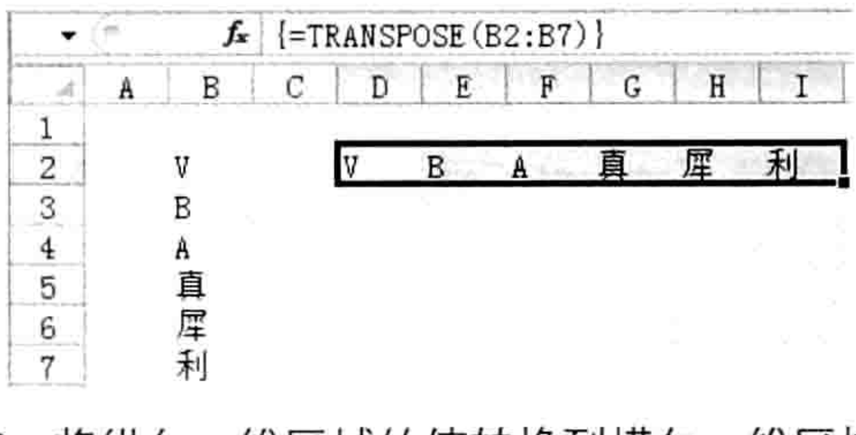


图 13.3　将纵向一维区域的值转换到横向一维区域中

图 13.4 中包含一个可以产生一维横向数组的公式，输入数组公式的方式是选择 B2:F2 区域，然后录入以下公式，最后按<Ctrl+Shift+Enter>组合键即可：

```
={1,3,5,7,10}
```

图 13.5 中包含一个可以产生一维纵向数组的公式，公式如下：

```
={"语文";"数学";"地址";"化学"}
```

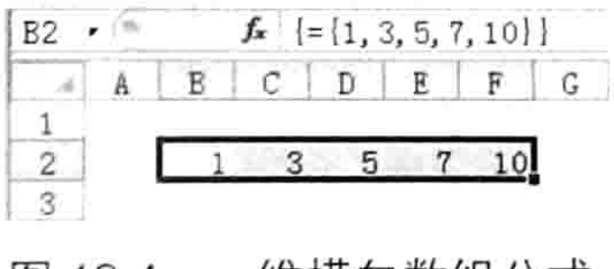


图 13.4　一维横向数组公式

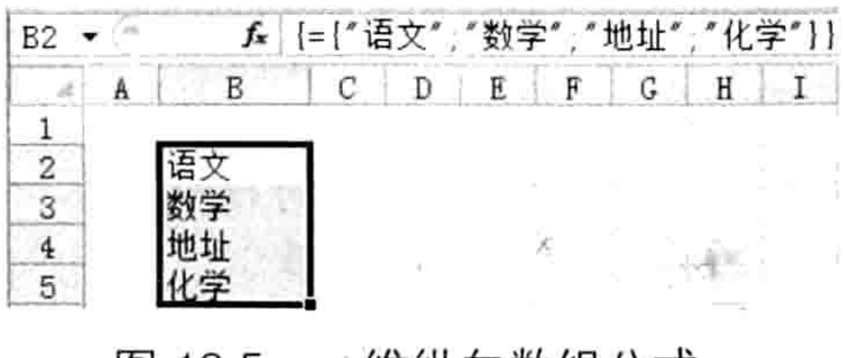


图 13.5　一维纵向数组公式

一维数组公式的特点是公式同时产生多个结果，必须同时选择多行一列或者多列一行的区域再输入公式，并且按<Ctrl+Shift+Enter>组合键结束。其中横向数组的元素与元素之间采用逗号作为分隔符，而纵向数组的元素与元素之间采用分号作为分隔符。

2. 一维数组

一维数组和前面的一维区域具有类似的特性，不过数组中只有一维横向数组，没有一维纵向数组。VBA 将具有多行单列的数组定义为二维数组，不过使用 Transpose 函数将它转换成横向数组

后就成了一维数组。

公式中通过花括号产生的一维常量数组可以应用在 VBA 中，VBA 利用方括号或者 Evaluate 函数将它转换成 VBA 的数组。

如果改用 VBA 实现如图 13.4 和图 13.5 所示的同等效果可以分别采用以下代码：

```
Range("B2:F2") = [{1,3,5,7,10}]
Range("B2:B5") = [{"语文";"数学";"地址";"化学"}]
```

第一句代码使用的是一维横向数组，第二句代码使用的是二维纵向数组，只不过这个二维数组只有一列。

利用数组在区域中批量输入数据时有比较大的优势，不需要逐个向单元格中写入值，而是同时在区域中的每个单元格产生数据。

3. 产生数组的方式

产生数组的方式比较多，主要体现为以下 5 种。

（1）手工指定数组元素

前面所讲的 "[{"语文";"数学";"地址";"化学"}]" 就是手工指定数组中的每个元素，从而生成常量数组。

也可以用 Evaluate 函数代替方括号生成数组，不过和方括号的写法稍有不同。

如果采用 Evaluate 函数代替方括号，由于其参数是文本，所以需要添加引号，代码如下：

```
Range("B2:F2") = Evaluate("{1,3,5,7,10}")
Range("B2:B5") = Evaluate("{""语文"";""数学"";""地址"";""化学""}")
```

使用 Evaluate 函数时要注意两点，其一是花括号前后需要使用半角的双引号；其二是当花括号里面的数组元素是文本时，它原本自带的单个双引号要改写成两个双引号。例如数组中的 ""语文"" 改写成 """语文""" 。

手工指定数组元素还可以使用 Array 函数，例如：

```
Range("B2:F2") = Array(1, 3, 5, 7, 10)
```

Array 函数只能创建一维横向数组，不能使用分号创建纵向数组，也不能生成二维数组。

（2）引用区域

区域中包含多个数据，将区域中的值读取到内存中就形成了数组。以下代码先声明一个变体型的变量，然后将区域的值赋予变量，此时变量 arr 就变成了数组。

```
Dim arr As Variant
arr = Range("A1:A10").Value
```

引用区域所产生的数组都是二维数组，不管区域是横向的还是纵向的，包含多少行、多少列，这和区域中的一维与二维的概念稍有分别。

（3）使用转置函数 Transpose

工作表函数 Transpose 可以将区域转换成数组，不过 Transpose 不是 VBA 函数，调用 Transpose 函数时需要加上其父对象名称 WorksheetFunction。

以下过程可将一维纵向的区域 A1:A10 转换成一维横向的数组：

```
Dim arr As Variant
arr = WorksheetFunction.Transpose(Range("A1:A10"))
```

13.2.4 节中有更多关于 Transpose 函数的详细说明。

（4）使用字典 dictionary

字典对象 dictionary 提供了一个 Add 方法，它可以将多个数据逐个写入到字典对象的关键字中

或者条目中，从而形成数组，同时也提供 Items 方法和 Keys 方法导出数组。

（5）使用 Split 函数

Split 函数可以将字符串按指定的分隔符转换成下标为 0 的一维数组。例如：

Split("四川,重庆,朝天门", ",")——生成包括四川、重庆和朝天门 3 个元素的一维数组；

Split("12 元 88 元 0.55 元", "元")——生成包括 12、88、0.55 和空文本 4 个元素的一维数组。

在 13.2 节和 13.3 节中会详细讲述 Split 函数的语法和应用案例。

同一个数组中每个元素的生成方式必须一致，不能部分元素是引用、部分元素是常量。例如以下代码只能产生错误值，因为数组加 4 个元素的生成方式不统一：

```
Range("a1:d1") = [{1,2,range("a1"),"VBA"}]
```

13.1.4　二维数组

二维数组是指多行多列的数组，其行数和列数大于 1，没有上限。不过数组是为工作而服务的，而工作中的数据受工作表的区域大小限制，所以在实际工作中使用数组时，数组的行数与列数上限都小于工作表的最大行数与列数。

A1:B5 是一个二维的区域，以下公式是一个可以产生二维数组的公式：

```
={"序号","目录";1,"Sheet1";2,"Sheet2";3,"Sheet3";4,"Sheet4"}
```

选择 A1:B5 后输入如图 13.6 所示的公式，然后按<Ctrl+Shift+Enter>组合键可以产生以下效果。

如果用 VBA 中的数组输入图 13.6 中的数据，可使用以下代码：

```
Range("A1:B5")  =  [{" 序 号 "," 目 录 ";1,"Sheet1";2,"Sheet2";3,"Sheet3";4,
"Sheet4"}]
```

对比二维数组和一维数组可以发现二者的差异在于行列数的不同。

二维数组有以下特点：

其一是在同一行中元素与元素之间的分隔符是逗号，换行时则使用分号。

其二是每一行的元素个数必须一致，同一列中的元素个数也必须一致。如果某行或者某列缺少一个元素那么整个数组中所有元素都会失效。例如在前面的代码中删除“Sheet4”后代码将只能产生如图 13.7 所示的错误值。

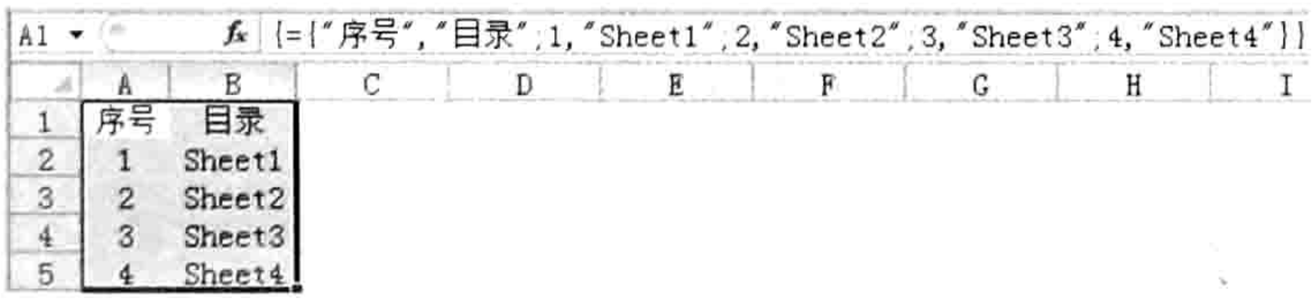


图 13.6　能产生二维数组的公式

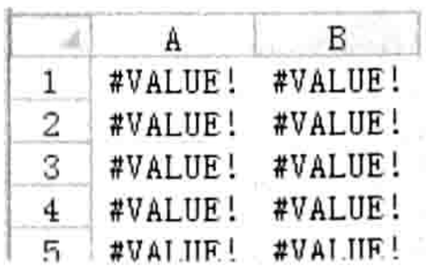


图 13.7　缺少元素时产生的错误值

解决办法是对该元素赋值为空文本占位。执行以下代码后将产生如图 13.8 所示效果。

Range("A1:B5") = [{"序号","目录";1,"Sheet1";2,"Sheet2";3,"Sheet3";4}]——错误代码；

Range("A1:B5") = [{"序号","目录";1,"Sheet1";2,"Sheet2";3,"Sheet3";4,""}]——正确代码。

其三是可以借助 Transpose 函数将二维数组的行与列转置。例如以下代码可以让 2 列 5 行的数组转换成 2 行 5 列，执行效果如图 13.9 所示。

```
Range("A1:E2") = WorksheetFunction.Transpose( _
[{"序号","目录";1,"Sheet1";2,"Sheet2";3,"Sheet3";4,"Sheet4"}])
```

	A	B
1	序号	目录
2	1	Sheet1
3	2	Sheet2
4	3	Sheet3
5	4	

图 13.8　使用空文本占位

	A	B	C	D	E
1	序号	1	2	3	4
2	目录	Sheet1	Sheet2	Sheet3	Sheet4
3					

图 13.9　转置二维数组

13.1.5　数组的参数

可以把一维数组和二维数组看作区域引用，这有助于理解数组和数组的参数。

Range ("B2:B5")是一个一维区域，使用索引号可以获取区域中单个单元格的值，例如以下代码可以获取 Range ("B2:B5")区域中的第 2 个值：

```
MsgBox Range("B2:B5")(2)
```

也可以采用双索引号引用一维区域中的第 2 个值，代码如下：

```
MsgBox Range("B2:B5")(2,1)
```

二维区域和一维区域一样，既可以使用单索引号也可以使用双索引号引用其中单个单元格的值。

Range ("A1:C5")是一个二维区域，以下代码可以获取该区域中第 3 行中第 2 列的值：

```
MsgBox Range("A1:C5")(3, 2)
```

以下代码可引用 Range ("A1:C5")中第 6 个单元格的值：

```
MsgBox Range("A1:C5")(6)
```

如果能理解以上从 Range 对象中获取单个值的思路，那么理解数组的参数就容易多了。

提取数组中的单个元素时也有单索引号和双索引号之分。不过与区域引用的索引号相比，数组中的索引号数量有更严格的规定。

在 VBA 中，引用一维数组中的单个元素时只能使用单索引号，例如：

```
Sub test()'代码存放位置：模块中
  '声明三个变体型变量，当给它们赋值数组后，变体型变量会自动变成数组变量
  Dim arr1, arr2, arr3
  arr1 = Evaluate("{1,2,5,8}")   '将 Evaluate 函数产生的一维数组赋值给变量
  arr2 = [{1,2,5,8}]             '将方括号产生的数组赋予变量
  arr3 = Array(1, 2, 5, 8)       '将 Array 产生的数组赋予变量
  MsgBox arr1(1)                 '获取数组 arr1 中第 1 个值
  MsgBox arr2(1)                 '获取数组 arr2 中第 1 个值
  '获取数组 arr2 中第 1 个值，不过 array 产生的数组下标为 0，所以第 1 个值是 2，第 0 个值为 1
  MsgBox arr3(1)
End Sub
```

引用二维数组中的单个元素时必须使用双索引号，例如：

```
Sub test2()'代码存放位置：模块中
    Dim arr                          '声明变量
    arr = Range("A1:C5").Value       '将区域的值写入变体型变量，此后变量变成数组
    MsgBox arr(3, 2)                 '获取数组中第 3 行中第 2 列的值
End Sub
```

以下代码是从 Evaluate 函数创建的二维数组中取值，也必须使用双索引号：

```
Sub test3()'代码存放位置：模块中
  MsgBox Evaluate("{""序号"",""目录"";1,""Sheet1"";2,""Sheet2"";3,""Sheet3"";
4,""Sheet4""}")(2, 2)
End Sub
```

13.1.6　声明数组变量

数组变量的声明方式和其他任何变量的声明方式都有区别，其中涉及比较多的规则。

由于数组总是包含多个元素，所以在赋值时就有一次性赋值所有元素和分次逐个赋值两种方法，而这两种方法决定了数组的声明方式也不同。

如果是一次性对数组赋值，那么声明变量只能使用变体型变量，而不能使用数组变量（指定了维数和上标、下标的数组变量），当将组数据赋值给变体型变量后，变量自然会转变成数组变量。

```
Sub test()'代码存放位置：模块中
    Dim Arr1, Arr2                          '声明两个变体型变量
Arr1 = Evaluate("{1,2;3,4;5,6}")            '将函数产生的数组赋值给变量，从而转换成数组
Arr2 = Range("a1:b10").Value                '将区域的值赋予变量，从而转换成数组
End Sub
```

以上过程中声明了两个变体型变量，当将数组赋值给两个变量后，变量都成了数组。

第一个变量的赋值方式是利用 Evaluate 函数产生一个二维数组，然后赋值给变量，此时变量将成为二维数组。

第二个变量的赋值方式是引用区域中的值，然后生成数组。代码“Arr2 = Range("a1:b10").Value”也可以简写为“Arr2 = Range("a1:b10")”，其实含义一样，因为 Range 对象的默认属性是 Value，省略属性名称时仍然调用它的 Value 属性值。

如果需要对数组变量逐个赋值，那么声明变量时应直接声明为指定维数和上标、下标的数组变量。

数组变量允许使用变体型以外的数据类型，例如声明带有 5 个元素的一维数组，那么应采用以下两种方式：

```
Dim arr1(1 To 5)
Dim arr2(4)
```

第一种声明方式指定了数组变量 arr1 是一个一维数组，其下标是 1、上标是 5。

第二种声明方式指定了数组变量 arr2 也是一个一维数组，其下标采用默认值 0，上标是 4，同样包含 5 个元素。

也可以在声明数组变量时指定其数据类型：

```
Dim arr1(1 To 5) as String
Dim arr2(4) as Long
```

“arr1(1 To 5)”和“arr2(4)”两种方式声明的数组变量在本质上是没有分别的，仅仅调用数组中的元素时采用的索引号不同。

在默认设置下，变量 arr1(1 To 5)所代表的数组的下标为 1，而变量 arr2(4)所代表的数组的下标为 0。

调用下标为 1 的数组中第 1 个值时，其索引号用 1；而调用下标为 0 的数组中第 1 个值时，其索引号用 0。

以下过程包含对下标为 0 和 1 的两个数组的声明、赋值和取值过程，可以通过此过程了解下标不同的数组在赋值及取值时的差异：

```
Sub test()'代码存放位置：模块中
  Dim arr1(4), arr2(1 To 5), i As Byte
  For i = 1 To 5
    arr1(i - 1) = i
    arr2(i) = i
  Next i
  MsgBox arr1(0)
  MsgBox arr2(1)
End Sub
```

在以上过程中，数组变量 Arr1 和 Arr2 的数组维数和数组的值完全一样，唯一的区别在于引用数组时的索引号不同。

如果声明二维的数组变量，那么分别在小括号中指定每一维的上标与下标即可。以下是声明二维数组变量的几种方式：

```
Dim arr1(4, 2)
```

以上代码声明了一个二维数组变量，第一维的下标为 0、上标为 4，第二维的下标为 0、上标为 2，数组中有 15 个元素。

```
Dim arr2(1 To 5, 1 To 3)
```

以上代码声明了一个二维数组变量，第一维的下标为 1、上标为 5，第二维的下标为 1、上标为 3，数组中也是 15 个元素。

```
Dim arr1(1 To 10, 6 To 10)
```

以上代码声明了一个二维数组变量，第一维的下标为 1、上标为 10，第二维的下标为 6、上标为 10，数组中有 50 个元素。不过在工作中极少人会用这种方式声明变量，不利于理解。

下面提供一个数组的应用案例，读者可以从中看到数组的价值，以及在不同需求下采取的数组声明方式与赋值方式。

案例要求：图 13.10 中包含 3000 个学生的成绩，要求根据 B 列的成绩判断是否“及格”，如果不及格则在“成绩”单元格的右侧单元格中返回“不及格”三个字。

	A	B
1	姓名	成绩
2	赵	50
3	钱	88
4	孙	72
5	李	50
6	周	89
7	吴	75
8	郑	74
9	王	54
10	冯	61

Sheet1

图 13.10　成绩表

知识要点：不同需求下的数组变量的声明方式。

程序代码：

为了展示数组应用对程序的裨益，这里特地采用两种方式实现案例需求，数组方法的代码如下：

```
Sub 对小于 60 分成绩进行注释()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Dim i As Integer, tim As Date
  Dim arr1, arr2(1 To 3000, 1 To 1)
  tim = Timer
  arr1 = Range("B2:B3001")
  For i = 1 To 3000
    If arr1(i, 1) < 60 Then arr2(i, 1) = "不及格"
  Next i
  Range("C2:C3001") = arr2
  MsgBox Format(Timer - tim, "0.00") &秒
End Sub
```

思路分析：

（1）以上过程中使用了 arr1 和 arr2 两个数组变量，第一个变量是直接调用区域的值，数组的

维数和每一维的上标、下标由区域决定，所以声明变量时不需要小括号及指定上标与下标。

第二个数组变量需要根据成绩值来决定赋值方式，不可能一次性完成赋值，所以声明变量时指定数组的维数和上标、下标。

（2）本例中数组的应用主要体现在两个方面，一是将成绩区域转换成数组，然后再判断成绩是否小于 60 时改用读取数组中的值，而不是读取单元格中的值，因为读取数组中的值远快于读取单元格中的值；其二是将原本需要逐个写入到单元中的数据写入到数组变量 arr2 中，最后再一次性将数组中的值导出到单元格中。换而言之，以前需要多次向单元格中写入值的操作改为只写入一次，从而提升操作速度。

（3）本例中数据变量 arr1 用于提升读取数据的速度，将读取单元格 3000 次修改为只读取一次，以后的读取操作都针对数组 arr，从而提升效率；数据变量 arr2 的作用则是提升写入的速度，将原本需要写入 900 多次的操作转换成只向单元格中写入一次。

为了方便读者比较，接下来提供不使用数组的过程代码：

```
Sub 对小于60分成绩进行注释2()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Dim i As Integer, tim As Date
  tim = Timer
  For i = 2 To 30001
    If Cells(i, 2) < 60 Then Cells(i, 3) = "不及格"
  Next i
  MsgBox Format(Timer - tim, "0.00秒")
End Sub
```

读者可以反复执行以上两个过程，比较它们的执行时间差异。

本例文件参见光盘：..\第十三章\13-1 数组与非数组的效率比较.xlsm

13.1.7 动态数组与静态数组的分别

在 13.1.6 节中提到声明其中一种数组变量时需要指定数组的维数和每一维的元素个数，其实那是理想状态下的用法，在实际工作中往往在声明变量时不确定每一维的元素个数，而是在声明变量之后通过代码计算得来，那么在声明变量时就无法指定其每一维的元素个数，这就引申出另一个知识点——动态数组与静态数组的区别。

所谓的静态数组是指维数一定、每一维的元素个数一定的数组，其书面解释是“在代码执行期间不能修改数组的上界（最后一个元素的索引号）的数组”。在前面的案例中数组 arr2(1 To 3000, 1 To 1)在声明时和使用时都固定了数组的维数和元素的个数，所以它是一个静态数组。

动态数组在工作中应用比较频繁，例如将前面案例的要求修改为罗列出所有不及格人员的姓名和成绩，那么由于声明变量时无法获知不及格的人数，所以需要采用动态数组。

1. 声明动态数组变量的方法

声明动态数组比较特别，需要分多个步骤完成。先用 Dim 语句声明一个带空括号（没有维数和下标）的数组变量，然后采用 ReDim 语句重置数组的维数和上标、下标（查询帮助的关键字：ReDim 语句）。

通常 ReDim 语句都会配合变量使用，而变量的值通过计算得来，这也正是动态数组的存在价值，即根据实际情况决定数组中最末维的上标，而不是预先固定上标。

以下代码是最典型的动态数组声明方法：

```
Dim arr(), i As Integer
```

```
i = Cells(Rows.Count, 1).End(xlUp).Row
ReDim arr(1 To i, 1 To 1)
```

第一句代码声明了一个没有维数和下标的数组变量；第二句代码对变量 i 赋值为 A 列最后一个非空行的行号；第三句代码利用 ReDim 语句重新指定数组变量的维数和每一维的下标、上标。

2. 重置数组时不保留原值

前面所讲的 ReDim 语句在重置数组变量时有一个特点——当数组变量已经赋值时会清除数组中的原有数据。

通过以下代码可以印证 ReDim 语句的这个特性：

```
Sub test()                              '请对 A1:B5 区域写入数据后再测试代码
    Dim arr()                           '声明数组变量
    arr = Range("a1:b5").Value          '对数组变量赋值
    MsgBox arr(2, 2)                    '获取变量中第 2 行第 2 列的值
    ReDim arr(1 To 3, 1 To 4)           '重新指定数组的维数和下标
    MsgBox arr(2, 2)                    '再次获取数组中的第 2 行第 2 列的值
End Sub
```

代码中第一次获取数组中的值时可以得到对应于单元格 B2 中的值，当使用 Redim 语句重置变量后将清除数组中的一切数据，所以第二个 Msgbox 函数只能取得空值。

通过以下案例可以更深入地了解动态数组的应用。

案例要求：图 13.10 中包含 3000 个学生的成绩，现在要求罗列出所有不及格的学生姓名、成绩，将结果存放在以 D1 开始的区域中。

知识要点：使用 ReDim 语句重置数组变量。

程序代码：

由于符合条件的人数需要通过计算得来，因此本例宜用动态数组，代码如下：

```
Sub 罗列不及格人员姓名与成绩()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Dim i As Integer, j As Integer, rng As Range, arr1, arr2()
  Set rng = Range([A2], Cells(Rows.Count, 2).End(xlUp))
  ReDim arr2(1 To WorksheetFunction.CountIf(rng, "<60"), 1 To 2)
  arr1 = rng.Value
  For i = 1 To UBound(arr1)
    If arr1(i, 2) < 60 Then
      j = j + 1
      arr2(j, 1) = arr1(i, 1)
      arr2(j, 2) = arr1(i, 2)
    End If
  Next i
  Range("D1:E" & j) = arr2
End Sub
```

思路分析：

（1）在以上过程中声明了两个数组变量，其中 Arr1 用于保存成绩区域的值，一次性赋值即可，所以只需要使用“Dim arr1”语句。

数组变量 arr2 用于保存不及格的成绩信息，在声明变量时无法获知不及格人数，所以先用 Dim 声明一个不指定维数的上标、下标的数组，然后重起一行利用 ReDim 语句重新指定数组的维数、上标及下标。

也就是说使用 Dim 语句声明数组变量时，上标和下标只能是数值，而使用 ReDim 语句重置数组的上标和下标时，既可以使用数值也可以使用变量，还可以是包含函数的表达式等。

（2）本例中 For Next 循环的初始值是 1，终止值使用了 UBound(arr1)，它表示数组 arr1 的第一维的元素个数，可以解理为二维数组的行数，也就是本例中的成绩个数。在 13.2 节中会详细介绍 Ubound 函数的语法和应用。

（3）本过程中变量 j 的作用是计算符合条件的人数，即在循环语句中每找到一个不及格的成绩就对变量 j 累加一次，变量 j 的值代表当前需要处理的不及格的成绩的序号，它刚好对应于数组 arr2 中需要写入的值的顺序，所以赋值时分别采用“arr2(j, 1) = arr1(i, 1)”和“arr2(j, 2) = arr1(i, 2)”。

（4）ReDim 语句可以反复地改变数组的元素，以及维数的数目，但是不能使用 ReDim 语句修改数组的数据类型，除非是该数组变量声明为变体型，并且没有使用括号。

（5）将数组的值一次性输出到工作表中时，直接使用 arr2 即可，不需要使用索引号。本例中代码“Range("D1:E" & j) = arr2”表示将 arr2 的值导出到以 D1 开始，到 E 列变量 j 行结束的区域中。其中变量 j 的值等于数组 arr2 的行数。图 13.11 中 D 列和 E 列中的值是程序的执行结果。

	A	B	C	D	E
1	姓名	成绩		赵	50
2	赵	50		李	50
3	钱	88		王	54
4	孙	72		陈	47
5	李	50		褚	53
6	周	89		卫	53
7	吴	75		沈	51
8	郑	74		韩	51
9	王	54		许	42
10	冯	61		吕	43
11	陈	47		张	56

图 13.11　罗列不及格人员信息

本例文件参见光盘：..\第十三章\13-2 罗列不及格人员姓名与成绩.xlsm

3. 重置数组时保留原值

在上面的案例中，在循环语句之前使用了工作表函数 Countif 计算符合条件的成绩数量，所以可以通过 ReDim 语句直接重置数组的上标。事实上在有些情况下是无法用函数或者其他任何办法计算出最终符合条件的数据个数的，只能在循环过程中逐一判断，反复计算符合条件的数据个数，并同步更新数组的上标，以确保数组足以存放所有符合条件的值。这就引申出 VBA 数组中的另一个概念——重置数组变量时保留原值。

在 VBA 中重置数组变量的上标并且能保留原值的语句是 ReDim Preserve。

（1）当声明数组变量时使用了空括号

当声明数组变量时使用了空括号时，例如“Dim arr()”，那么 ReDim Preserve 语句有以下特点：

- 可以修改一次数组变量的维数；
- 可以修改一次任意维数的下标；
- 可以反复修改最末一维的上标。

（2）当声明数组变量时指定了维数

当声明数组变量时指定了维数，例如“Dim arr(1 to 2,1 to 5)”或者“Dim arr2(5,6)”，那么 ReDim Preserve 语句有以下特点：

- 不可以修改数组变量的维数；
- 不可以修改任意维数的下标；
- 可以反复修改最末一维的上标。

以上特点表明 ReDim Preserve 在使用中限制比较多。

不过，尽管 ReDim Preserve 语句并不完美，但是在工作中仍然有很多需求依赖它，它能给工作带来比较大的帮助。

先看一看 ReDim Preserve 的工作模式：

```
Sub test()   '代码存放位置：模块中
  Dim arr()        '声明数组变量
  '重新指定数组的维数和上标
  'ReDim Preserve 能多次修改最后一维的上标，但只能改变一次数组的维数
  ReDim Preserve arr(1 To 2, 1 To 2)
  arr(1, 1) = 1   '数组赋值，方便后面的测试
  arr(1, 2) = 2
  arr(2, 1) = 3
  arr(2, 2) = 4
  '重新指定数组最末一维的上标(第一维的上标不可以修改)
  ReDim Preserve arr(1 To 2, 1 To 3)
  '获取数组中的第 2 行第 2 列的值(可以发现重置最末一维的上标后，原数据还在)
  MsgBox arr(2, 2)
End Sub
```

以上过程中先利用 Dim 语句声明一个空的数组变量，然后利用 ReDim Preserve 语句重置数组的维数和上标、下标。ReDim Preserve 语句第一次重置数组变量时可以修改其维数和第一维的下标、上标，其后只能修改最末一维的上标。

过程中第一次重置数组变量后，数组包含两维，每一维有两个元素，然后分别对数组的 4 个元素赋值。接着使用 ReDim Preserve 语句第二次重置数组，这一次只能修改最末一维的上标，如果采用代码“ReDim Preserve arr(1 To 3, 1 To 2)”重置数组则会产生“下标越界”错误。

在过程的最后，使用 MsgBox 函数获取二维数组中第 2 行第 2 列的值，结果是 2，表示 ReDim Preserve 语句重置数组时会保留重置前的数据。

ReDim Preserve 语句的这个特性极其有用，下面的案例就借用了它的独特优势。

案例要求：图 13.12 的工作簿中包含 3 个工作表，每个工作表中的学生人数不确定，要求罗列出所有班级中不及格的学生信息，包括姓名和成绩，结果显示在“不及格”工作表中。

	A	B	C
1	姓名	成绩	
2	朱通	81	
3	周锦	67	
4	曹值军	57	
5	吴国庆	74	
6	蒋有国	61	
7	严西山	100	
8	陈文民	56	
9	陈冲	51	
10	郑君	67	

A班 B组 C组

图 13.12　三个班级的成绩表

知识要点：使用 ReDim Preserve 重置数组、利用 Transpose 转置数组方向。

程序代码：

```
'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
Sub 罗列多个班级的不及格人员姓名与成绩()
  Dim sht As Worksheet, Item As Integer, TargetCount As Integer, arr, arr2()
  For Each sht In Worksheets
    arr = sht.UsedRange.Value
    For Item = 2 To UBound(arr)
      If arr(Item, 2) < 60 Then
```

```
        TargetCount = TargetCount + 1
        ReDim Preserve arr2(1 To 2, 1 To TargetCount)
        arr2(1, TargetCount) = arr(Item, 1)
        arr2(2, TargetCount) = arr(Item, 2)
      End If
    Next Item
Next
On Error Resume Next
If TargetCount > 0 Then
    Application.DisplayAlerts = False
    Worksheets("不及格").Delete
    Application.DisplayAlerts = True
    Worksheets.Add(after:=Worksheets(Worksheets.Count)).Name = "不及格"
    Worksheets("不及格").Range("a1:b1") = [{"姓名","成绩"}]
    Worksheets("不及格").Range("A2").Resize(TargetCount, 2) = WorksheetFunction.
Transpose(arr2)
    Worksheets("不及格").UsedRange.Borders.LineStyle = xlContinuous
  End If
End Sub
```

执行以上代码后得到如图 13.13 所示结果。

	A	B	C
1	姓名	成绩	
2	计尚云	55	
3	罗至贵	56	
4	徐大鹏	50	
5	梁兴	54	
6	陈随机	59	
7	曲华国	57	
8	刘文喜	54	
9	魏郑	54	
10	柳星华	58	

A班 / B组 / C组 / 不及格

图 13.13 不及格人员信息

思路分析：

（1）本例中使用了两个数组，一个用于提升读取数据的速度，另一个用于提升写入数据的速度。其中数组 arr 用于存放工作表中的学生信息，当区域中的值导入到数组中后，读取数组替代了读取单元格，从而大幅缩短了读取时间。而数组 arr2 则用于保存所有符合条件的数据，借用此数组可以实现一次性将数据写入到新工作表“不及格”中，从而缩短写入数据的时间。

（2）由于在执行循环语句之前无法获得所有班级中不及格的人数，所以声明数组变量时无法指定其每一维的上标，而是首先声明一个不指定维数的数组变量，然后在循环语句中使用 ReDim Preserve 语句为数组变量重新分配维数和上标、下标。

在第一次使用 ReDim Preserve 时允许修改数组变量的维数和上标、下标，但是第二次使用 ReDim Preserve 时则只能修改数组最末维的上标。所以本例采用“ReDim Preserve arr2(1 To 2, 1 To TargetCount)”而不是“ReDim Preserve arr2(1 ToTargetCount, 1 To 2)”语句。

（3）按照需求，动态数组变量 arr2 应该是包含两列（第一列存放姓名，第二列存放成绩），行数则由不及格人数决定，即数组的第一维是动态的，第二维的上标固定为 2。但是鉴于 VBA 的规定，第一维不允许反复修改上标，所以本例采用了变通的方式，将数组变量声明为行数为 2、列数由不及格成绩的数量决定，当向数组中写入数据后将变成横向数组。

为了避免最后将数组的值导出到工作表时方向与要求不一致，所以本例采用了工作表函数 Transpose 将其转置方向，然后再导出到工作表中。

（4）本例中的“On Error Resume Next”和“Worksheets("不及格").Delete”语句都是为了防错

而存在的，虽然图 13.12 所示的案例文件中并没有工作表“不及格”，但是编程时仍然有必要通过代码处理一些意外情况，先假设可能存在同名的表，然后再通过代码来处理该表。在此思想指导下产生的代码才可能完善，读者在编程过程中有必要养成这种习惯。

（5）过程中最后一个条件语句的作用同样是防错，避免表中没有不及格成绩时也创建一个名为“不及格”的工作表，而且执行一些不必要的操作。在编程时要考虑所有的可能情况，不能一切按正常或者最常出现的情况处理，否则代码将留下太多漏洞，也浪费执行时间。

本例文件参见光盘：..\第十三章\13-3 罗列多个班级的不及格人员姓名与成绩.xlsm

13.1.8 释放动态数组的存储空间

数组变量中保存了大量的数据，当确定不再使用时需要释放变量的值。

数组变量和普通变量的生命周期是一致的。

过程级变量的生命周期最短，当过程结束时变量的值就自动释放，所以不需要手动释放变量的值。模块级变量和工程级变量属于公有变量，需要关闭 Excel 或者手动释放变量，变量所占用的空间才会释放出来。

手动释放数组变量的值有两种方法，其一是使用 Erase 语句，其二是使用 End 语句。

1. Erase 语句

Erase 语句的语法如下：

```
Erase arraylist
```

参数 arraylist 代表数组，即一次只能释放一个数组变量的值。假设模块中有 3 个公有数组变量 arr1、arr2 和 arr3，那么可采用以下 3 句代码释放所有数组变量的值：

```
Erase arr1
Erase arr2
Erase arr3
```

2. End 语句

当需要释放的数组变量太多时，End 语句可以一次性释放所有变量的值，所以 End 语句的便利性和高效性超过 Erase 语句。

不过 End 语句有可能会误伤，因为它是释放所有变量的值，包括一切公有变量和私有变量，而且不仅仅是针对数组变量。所以在使用 End 语句前需要仔细斟酌。

13.2 数组函数

VBA 提供了诸多内置的数组函数，可以方便地操作数组。在工作中应用数组前有必要了解这些函数的功能和语法。

13.2.1 用函数创建数组

VBA 提供了专用的创建数组的函数 Array 和 Evaluate，它们都能创建数组。

1. Array 函数创建一维横向常量数组

Array 函数只能创建一维常量数组，其语法如下：

```
Array(arglist)
```

参数 arglist 是一个用逗号隔开的数据表，这些值分别用于给数组的各个元素赋值。例如 Array(1,2,3)表示创建一个包含 3 个元素的一维数组，第一个元素的值为 1，第二个元素的值为 2，第三个元素的值为 3。

假设需要向工作表中 A1、B1、C1、D1 单元格中输入标题“姓名”、“工号”、“产量”和“不良率”，那么不需要分 4 次赋值，而是利用 Array 函数产生一个包含 4 个元素的一维数组，然后将数组赋值给区域，代码如下：

```
Range("A1:D1") = Array("姓名", "工号", "产量", "不良率")
```

使用数组对区域进行赋值比逐个单元格赋值的效率更高，不过 Array 函数的缺点是只能创建一维横向数组，如果需要创建纵向的数组，那么可以利用转置函数 Transpose 将 Array 产生的数组转换方向，例如在 A1:A4 区域产生“一月”、“二月”、“三月”和“四月”，那么可采用以下代码：

```
Range("A1:A4") = WorksheetFunction.Transpose(Array("一月", "二月", "三月", "
四月"))
```

2. Evaluate 函数创建横向与纵向的常量数组

Evaluate 函数比 Array 函数强大得多，既可以创建一维常量数组，也可以创建二维常量数组，还可以通过引用区域的值创建内存数组。不过“Range("A1:A10").Value”这种模式也可以将区域的值转换成数组，所以在工作中一般只用 Evaluate 函数创建常量数组。

使用 Evaluate 函数创建一维常量数组时，其参数是文本字符串，而文本字符串中需要使用“{}”，在“{}”中包含创建数组的数据列表。如果创建只有单行的一维横向常量数组，那么元素与元素之间采用逗号作为分隔符，如果创建单列的纵向常量数组，那么元素与元素之间采用分号作为分隔符。例如：

Evaluate("{1,3,5,7,9}")——创建包括 5 个元素的一维横向数组；

Evaluate("{2;4;6;8}")——创建包括 4 个元素的二维纵向数组（只有一列但仍是二维数组）。

图 13.14 和图 13.15 中比较形象地展现了以上两个数组的区别。

	A	B	C	D	E
1	1	3	5	7	9
2					

图 13.14　一维横向数组

	A	B
1	2	
2	4	
3	6	
4	8	

图 13.15　二维纵向数组

如果数组中的元素是文本，那么需要对文本的左右添加双引号。但是由于“{}”外面已经有双引号，所以数组元素的每个双引号都要改为两个双引号，VBA 才能正常识别。例如：

Evaluate("{2;""VBA"";6;8}")——表示创建一维纵向数组，其中第 2 元素是字符串“VBA”，它的前后各采用了两个半角的双引号，如果采用单个双引号将会产生编译错误。

Evaluate 函数创建的一维横向常量数组和 Array 函数创建的一维横向常量数组稍有不同——Evaluate 函数创建的一维横向常量数组下标为 1，Array 函数创建的一维横向常量数组下标为 0。以下过程创建了两个值相同的一维横向常量数组，不过在访问其中的数据时需要采用不同的索引号，因为它们的下标不同。

```
Sub test()'代码存放位置：模块中
    Dim arr, arr2
    arr = Array(1, 2, 3, 4, 5, 6)
    arr2 = Evaluate("{1,2,3,4,5,6}")
    MsgBox arr(2)     '取索引号为 2 的值，结果为 3(下标为 0)
    MsgBox arr2(2)    '取索引号为 2 的值，结果为 2(下标为 1)
End Sub
```

以上代码采用两个函数创建了具有相同元素的数组，不过由于下标不同，所以同样是访问索引号为 2 的元素，得到的答案并不相同。

Array 函数创建的数组下标总是 0，索引号为 0 的元素才是数组中的第一个值。

3. Evaluate 函数创建多行多列的二维常量数组

Evaluate 函数创建二维常量数组和一维常量数组的方式相似，只是在需要换行的地方添加分号即可，例如将一维常量数组 Evaluate ("{1,2,3,4,5,6}")修改为 2 行 3 列的二维数组，只要将代码修改为 “Evaluate ("{1,2,3;4,5,6}")” 即可，在 10.1.4 节也有关于二维常量数组的介绍。

需要特别强调的是，代码 “Evaluate("=Sheet2!A1:A5")” 创建的其实是 Range 对象，而不是数组，“Evaluate("=Sheet2!A1:A5").value” 才能创建数组。

13.2.2 获取数组元素

VBA 允许调用工作表函数 Index 从数组中获取单个值，语法如下：

```
WorksheetFunction.Index(arraylist, Row_num,Column_num)
```

其中参数 arraylist 表示数组，参数 Row_num 表示行数，Column_num 表示列数。此处的行数、列数和数组的索引号稍有不同。

行数与列数的最小值是 1，而数组的索引号（下标）可能是 0，可能是 1，也可能是 100 或者 555 等，这取决于声明数组变量的方式。

```
Sub test()     '区分不同数组的下标，以及了解 Index 函数从数组中取值的特性（放置位置：模块中）
    Dim arr, arr2, arr3(5 To 10)       '声明 3 个数组变量
    arr = Array(1, 2, 3, 4, 5, 6)      '使用 Array 函数对第一个数组赋值(其下标为 0)
    arr2 = Evaluate("{1,2,3,4,5,6}") '使用 Evaluate 函数对第二个数组赋值(其下标为 1)
    arr3(5) = 1                        '对第 3 个数组的元素逐个赋值(其下标为 5)
    arr3(6) = 2
    arr3(7) = 3
    arr3(8) = 4
    arr3(9) = 5
    arr3(10) = 6
    MsgBox WorksheetFunction.Index(arr, 1, 3)    '获取第 1 行、第 3 列的值
    MsgBox WorksheetFunction.Index(arr2, 1, 3)   '获取第 1 行、第 3 列的值
    MsgBox WorksheetFunction.Index(arr3, 1, 3)   '获取第 1 行、第 3 列的值
End Sub
```

以上过程中采用 3 种方式创建了 3 个数组，它们的下标分别是 0、1、5，不过使用 Index 函数读取数组中的元素时全按行、列数计算，不理会上标与下标的值。

13.2.3 判断变量是否为数组

VBA 提供了判断变量是否为数组的函数——IsArray。

当函数 IsArray 的返值是 True 时，表示参数是数组，否则不是数组。

在实际工作中 IsArray 函数的价值极小，几乎不用。

13.2.4 转置数组

工作表函数 Transpose 在数组应用中是一个很有用的函数，不过它不是 VBA 内置的函数，而是属于工作表函数，所以调用此函数需要指定父对象 WorksheetFunction。

转置函数 Transpose 主要有 3 个功能：转换数组方向、将区域引用转换成数组、将任意的只有单行或者单列的二维数组转换成一维数组。

1. 转换数组方向

VBA 的数组应用中经常需要转置数组的方向，特别是动态数组的应用。

```
Sub test()'代码存放位置：模块中
  Dim arr As Variant
  arr = Evaluate("{""V"", ""B"", ""A""; ""It"", ""Is"", ""Good""}")
  Range("a1:c2") = arr
  Range("E1:F3") = WorksheetFunction.Transpose(arr)'转置方向后写入另一个区域
End Sub
```

以上过程中利用 Evaluate 函数创建了一个二维常量数组，它的每个元素的排序方式为图 13.16 中 A1:C2 区域所示的效果；当使用 Transpose 函数转置数组后则得到图中 E1:F3 区域所示的效果。

	A	B	C	D	E	F
1	V	B	A		V	It
2	It	Is	Good		B	Is
3					A	Good

图 13.16　转置数组前后效果比较

2. 将区域引用转换成数组

使用区域引用作为 Transpose 函数的参数时将会生成数组，不过数组的方向与区域本身的行列方向相反。

例如：Range("A1:E1")是横向的区域引用，其类型为“Range”，而 WorksheetFunction.Transpose (Range("A1:E1"))的类型则是“Variant()”，它此时是一个纵向的数组。

3. 将单行或者单列的二维数组转换成一维数组

Transpose 在处理只有单行或者单列的数组时比较特别，它在改变数组的方向时还可以改变数组的维数。

当一维的区域引用赋值给变体型变量后，此变量所代表的数组是一个二维数组，其第一维和第二维的下标都是 1，需要使用双索引号才能引用数组中的元素。使用 Transpose 函数能将它们转换成一维数组。

以下过程中使用了两个变体型变量，分别将一维横向区域和一维纵向区域赋值给两个变量，从而使两个变量变成二维数组，然后再通过 Transpose 函数将它们都转换成一维数组：

```
Sub 将二维区域转置成横向的一维数组()'代码存放位置：模块中
  Dim arr1, arr2                                    '声明变体型变量
  arr1 = Range("A1:A5").Value                       '此时 arr1 是 5 行 1 列的二维数组
  arr1 = WorksheetFunction.Transpose(arr1)          '此时 arr1 是一维数组
  arr2 = Range("A1:E1").Value                       '此时 arr2 是 1 行 5 列的二维数组
  '此时 arr2 是一维数组
  arr2 = WorksheetFunction.Transpose(WorksheetFunction.Transpose(arr2))
End Sub
```

此外，Transpose 函数还有一个比较神奇的功能——将单行或者单列的区域生成的二维数组转换成一维数组。

在 VBA 中，引用区域的值可以生成数组，不过 VBA 与工作表函数对于数组维数的理解是不同

的。在 VBA 中不管是引用单行还是引用单列的值所生成的数组都是二维数组，而利用 Transpose 函数可以将它们转换成一维数组。

VBA 中的 Join 函数和 Filter 函数都只支持一维数组作参数，因此要引用单行或者单列的值作为 Join 函数和 Filter 函数的参数去参与运算时应配合 Transpose 函数使用。

例如将 A1:A10 区域中的值合并到 B1 单元格中，以下代码无法执行：

```
Sub 合并区域的值()'代码存放位置：模块中
  Range("b1") = Join(Range("a1:a10").Value, "")
End Sub
```

而使用 Transpose 函数后，区域中的值可以合并成功，如图 13.17 所示即为合并后的效果。

```
Sub 合并区域的值2()'代码存放位置：模块中
  Range("b1") = Join(WorksheetFunction.Transpose(Range("a1:a10").Value), "")
End Sub
```

	A	B
1	A	ABCDEFGHIJ
2	B	
3	C	
4	D	
5	E	

图 13.17　将区域中的值合并到单元格中

在 13.2.6 和 13.2.7 节中会详细讲述 Join 和 Filter 函数的功能与语法。

Transpose 函数也有它的局限性——参数的元素个数不超过 65 536 个才能转置，否则会产生“类型不匹配”错误。所以当确定数组中的元素个数超过 65 536 时，应采用循环语句逐个赋值的方式转置方向。以下过程即为参考标准：

```
Sub 转置超过65536个元素的数组()          '代码存放位置：模块中
  Dim arr1, arr2(1 To 65537)             '声明两个变量，其中arr2是一维横向数组
  arr = Range("a1:a65537").Value         '对变量arr赋值，arr变成一维纵向数组
  For i = 1 To 65537                     '从1到65537
    arr2(i) = arr(i, 1)                  '将数组arr1中的值逐个输出到arr2中
  Next
End Sub
```

如果是小于等于 65 536 个元素，那么直接用 Transpose 函数进行转置即可：

```
Sub 转置不超过65536个元素的数组()        '代码存放位置：模块中
  Dim arr1, arr2
  Arr1 = Range("a1:a65536").Value
  arr2 = WorksheetFunction.Transpose(arr1)
End Sub
```

13.2.5　获取数组的上标与下标

获取数组的上标、下标可以知道数组中的元素个数。对数组执行循环时也需要调用数组的下标与上标。

VBA 提供了 UBound 和 LBound 函数分别用于获取数组的上标和下标。

```
Sub test()'代码存放位置：模块中
  Dim arr
  arr = Array(1, 2, 3, 4, 5, 6)                '
```

```
    MsgBox "上标为：" & UBound(arr) & Chr(10) & "下标为：" & LBound(arr)
End Sub
```

此过程中使用 Array 函数创建了一个一维数组，如图 13.18 所示为通过 UBound 和 LBound 函数获得的该数组的上标与下标。

Array 函数创建的数组默认下标为 0，而 Evaluate 函数创建的数组默认下标为 1，这种差异往往给初学者带来麻烦。VBA 提供了修改默认下标的方法——Option Base 语句。

Option Base 语句的语法如下：

```
Option Base {0 | 1}
```

即可选项包含“Option Base 0”和“Option Base 1”两项。

默认的数组下标是 0，所以在任何情况下都不需要使用“Option Base 0”，当需要统一所有数组的下标为 1 时在模块顶部输入“Option Base 1”即可。

图 13.19 中所示的代码声明了一个变体型变量 arr 和一个数组变量 arr3，当变体型变量 arr 赋值为区域的值后就成为了真正的数组。

由于在模块顶部使用了“Option Base 1”，所以通过 LBound 函数获取两个数组的下标时都能得到数值 1。

图 13.18 获取数组的上标与下标

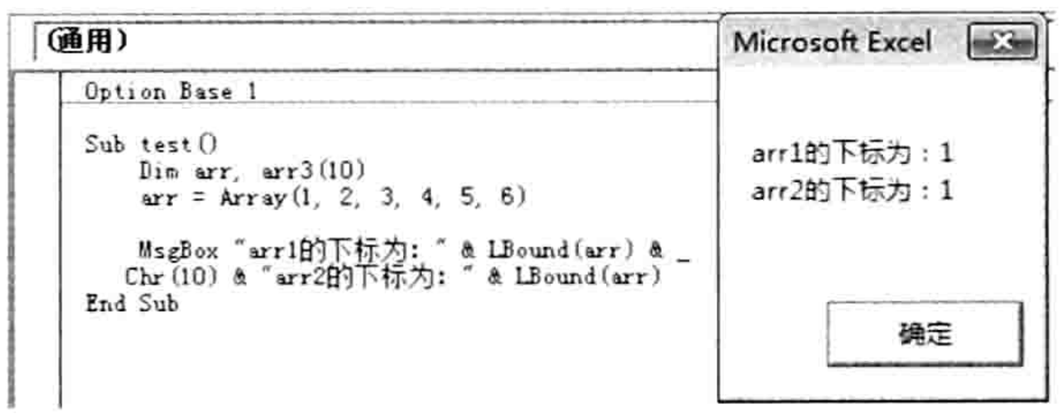


图 13.19 通过 Option Bas 语句将数组的默认下标修改为 1

当然，UBound 和 LBound 函数并非仅限于计算数组第一维的上标和下标，它们还可以通过第 2 参数指定维数从而计算任意维的上标与下标。

以下过程将分别得到一维、二维、三维的上标，值分别为 1、5 和 20：

```
Sub 分别计算一二三维的上标()  '代码存放位置：模块中
  Dim arr(1, 5, 20)
  MsgBox UBound(arr, 1) & Chr(10) & UBound(arr, 2) & Chr(10) & UBound(arr, 3)
End Sub
```

知识补充：Excel 帮助中关于 UBound 函数的解释有错误，关于 Option Base 的解释也有错误。具体错在何处，以及证明错误的存在请参阅随书光盘中“帮助纠错.txt”。

Option Base 语句可以修改 Array 函数所创建的数组的下标，但是对于 Split 函数创建的数组无效。

13.2.6 转换文本与数组

Split 函数可将字符串转换成数组，而 Join 函数可以数组转换成字符串。

1. Split 函数

Split 函数可以将字符串按指定的分隔符转换成下标为 0 的一维数组。它的语法如下：

```
Split(expression[, delimiter[, limit[, compare]]])
```

其中 4 个参数的含义如表 13-1 所示。

表 13-1　Split函数的参数说明

参数	是否必选	描述
expression	必选参数	包含子字符串和分隔符的字符串表达式。如果是一个长度为零的字符串("")，Split 则返回一个空数组
delimiter	可选参数	用于标识子字符串边界的字符串字符。如果忽略，则使用空格字符(" ")作为分隔符，返回的数组仅包含 expression 一个元素
limit	可选参数	要返回的子字符串数量，使用–1 或者忽略参数时表示返回所有子字符串
compare	可选参数	数字值，表示判别子字符串时使用的比较方式。有 4 种比较方式，其中 vbTextCompare 表示执行文字比较，不区分大小写；vbBinaryCompare 表示执行二进制比较，要区分大小写。忽略参数时表示采用 vbTextCompare

通常在字符串中有明显的分隔符时才有必要使用 Split 函数，例如将“1，2，3”以逗号为分隔符转换成包含 3 个元素的数组，再例如将“湖南-张家界-天门山-天门洞”以“-”为分隔符转换成包含 4 个元素的数组……

以下过程可以展示 Split 函数的工作模式，以及它生成的数组的特点。

```
Sub Split 函数的应用()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Dim i As Byte, arr As Variant
  arr = Split("湖南-张家界-天门山-天门洞", "-")
  i = UBound(arr)
  Range("A1").Resize(1, i + 1) = arr
End Sub
```

如图 13.20 所示是执行以上过程的结果。

	A	B	C	D
1	湖南	张家界	天门山	天门洞
2				

图 13.20　将字符串转换成数组并导出到区域中

由于 Split 函数生成的数组总是下标为零，不受 Option Base 语句的影响，所以它所生成的数组的元素个数一定等于数组的上标加 1。所以以上过程中 Range.Resize 属性的第 2 参数使用了“i+1”。

本例文件参见光盘：..\第十三章\13-4 Split 函数的应用.xlsm

其实，Split 函数最常见的应用是根据路径获取根目录或者文件名称。

例如某文件的路径是“D:\5 月\生产表\总表.xlsx”，要求计算该文件名称和所在的根目录，采用数组函数 Split 和 UBound 可以处理此问题，代码如下：

```
'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
Sub 从路径中获取文件名称与根目录()
  Dim Mystr As String
  Mystr = "D:\5 月\生产表\总表.xlsx"
  MsgBox "路径" & Split(Mystr, "\")(UBound(Split(Mystr, "\"))) & Chr(10) &
Split(Mystr, "\")(0)
End Sub
```

过程中 Split 函数对路径以“\”为分隔符转换成一个一维数组，然后利用 UBound 函数计算该

数组的上标，以上标作为数组的索引号可以提取数组中最后一个元素的值，即文件名称。而以 0 作为数组的索引号时则可以提取数组中第一个元素的值，即路径的根目录名称，如图 13.21 所示。

本例文件参见光盘：..\第十三章\13-5 从路径中取文件名与根目录.xlsm

图 13.21 文件名称与根目录

2. Join 函数

Join 函数可用于连接数组中的所有子字符串从而创建一个新的字符串，可随意指定分隔符，其语法如下：

```
Join(sourcearray[, delimiter])
```

其中参数 sourcearray 代表数组，参数 delimiter 代表分隔符，如果忽略分隔符，则默认采用空格作为分隔符。

Join(Array("D:", "生产表", "5 月"), "\")——转换结果为“D:\生产表\5 月”；

Join(Array("2013", "09", "28"), "-")——转换结果为“2013-9-28”；

Join(Array("VBA", "果然", "神奇"), "")——转换结果为“VBA 果然神奇”。

可以将 Join 函数理解为合并数组的值，并在适当的位置添加分隔符。

事实上也可以使用 Join 函数合并区域的值，不过必须配合 Transpose 函数来实现，在 13.2.4 节中有此类案例应用。

13.2.7 筛选数组

Filter 函数用于返回一个下标从零开始的数组，该数组包含基于指定筛选条件的一个字符串数组的子集。通俗地讲，Filter 函数就是对一个一维数组执行筛选，产生一个新的数组。其语法如下：

```
Filter(sourcesrray, match[, include[, compare]])
```

各参数的含义见表 13-2。

表 13-2 Filter函数的参数提示

参数	含义
sourcearray	必需的。要执行搜索的一维字符串数组
match	必需的。要搜索的字符串
include	可选的。Boolean 值，表示新数组是否包含 match 字符串。如果值为 True，则返回包含 match 子字符串的数组子集，否则返回不包含 match 子字符串的数组子集
compare	可选的。数字值，表示所使用的字符串比较类型，通常用于控制是否区分大小写。

Filter 函数的第 3 参数比较重要，它决定筛选时采用包含或者不包含的方式。

Filter 函数只能筛选一维数组，对二维、三维数组无效。

以下案例对数组执行了两种筛选方式，生成两个新数组，然后再用 Join 函数合并数组中的每个元素，并显示在信息框中，结果如图 13.22 所示。

```
Sub 找出姓罗与不姓罗的所有姓名()'①代码存放位置：模块中②随书光盘中有每一句代码含义注释
  Dim arr1, arr2, arr3
  arr1 = Array("罗成", "骆宾王", "柳如是", "曲非烟", "罗贯中", "罗通")
  arr2 = Filter(arr1, "罗", True)
  arr3 = Filter(arr1, "罗", False)
```

```
  MsgBox "所有姓罗的姓名:" & Chr(10) & Join(arr2, ",") & Chr(10) & _
    "所有不姓罗的姓名: " & Chr(10) & Join(arr3, ",")
End Sub
```

除了从数组中筛选出子集生成一个新数组外，也常使用Filter函数判断一个数组中是否包含某个元素。例如判断数组中是否包含“曲飞烟”，那么可用以下代码：

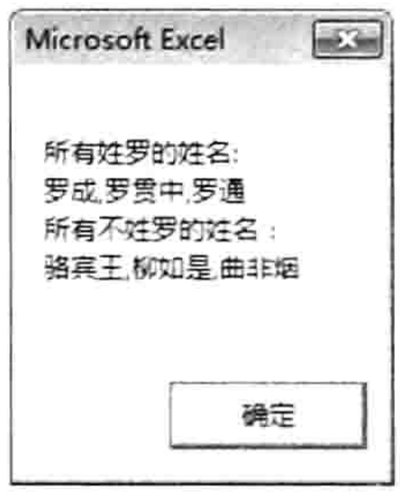


图 13.22　两种数组筛选方式结果

```
Sub 判断是否包含曲飞烟()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Dim arr1
  arr1 = Array("罗成", "骆宾王", "柳如是", "曲非烟
", "罗贯中", "罗通")
  MsgBox IIf(UBound(Filter(arr1, "曲飞烟", True))
< 0, "不包含", "包含")
End Sub
```

过程的执行结果为“不包含”。过程的原理是：

Filter函数的筛选结果是一维数组，并且下标为0，上标大于等于0，如果上标小于0，那么说明未成功生成数组，从而表示不包含筛选条件。所以本例利用UBound函数计算Filter函数生成的数组的上标，并配合IIF函数即可达成需求。

本例文件参见光盘：..\第十三章\13-6 Filter函数应用.xlsm

13.3　案例分析

VBA在数组方面的理论不多，掌握数组的应用也较简单。

在实际工作中通常使用数组替代区域，从而提升数据的读写速度。本节中通过7个案例展示数组的应用，加深读者对数组的理解。

13.3.1　将指定区域的单词统一为首字母大写

案例要求：将指定区域中的英文单词全部修改为首字母大写。

知识要点：Application.InputBox方法、Intersect方法、UBound函数、StrConv函数。

程序代码：

```
Sub 将单词转换成首字母大写()'①代码存放位置：模块中②随书光盘中有每一句代码有含义注释
  Dim arr, rng As Range
  Set rng = Application.InputBox("请选择操作区域", "确定区域", , , , , , , 8)
  If rng Is Nothing Then Exit Sub
  arr = Intersect(ActiveSheet.UsedRange, rng).Value
  Dim RowCount As Integer, ColumnCount As Integer
  For RowCount = 1 To UBound(arr)
    For ColumnCount = 1 To UBound(arr, 2)
      arr(RowCount, ColumnCount) = StrConv(arr(RowCount, ColumnCount), vbProperCase)
    Next ColumnCount
  Next RowCount
  Intersect(ActiveSheet.UsedRange, rng) = arr
End Sub
```

执行以上过程时将弹出“确定区域”的输入框，此时用鼠标选择 A:C 区域即可，输入框中将自动产生选区的地址，如图 13.23 所示。当单击输入框中的“确定”按钮后，程序瞬间将选区中的所有单词转换成首字母大写，执行效果如图 13.24 所示。

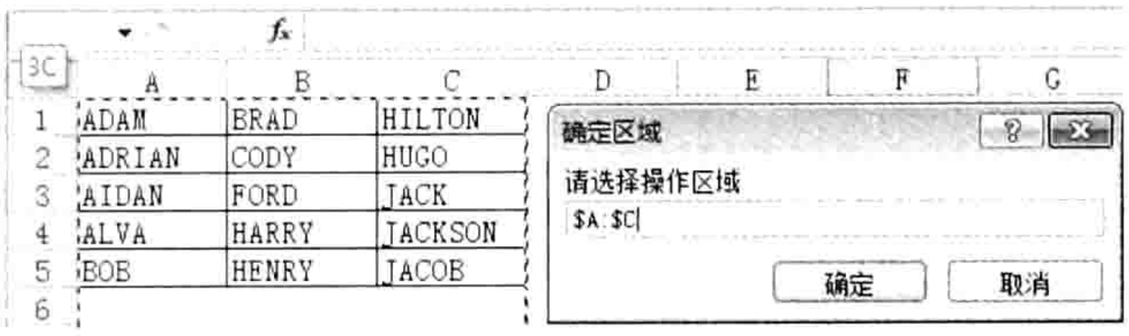


图 13.23　选择区域

	A	B	C
1	Adam	Brad	Hilton
2	Adrian	Cody	Hugo
3	Aidan	Ford	Jack
4	Alva	Harry	Jackson
5	Bob	Henry	Jacob

图 13.24　转换结果

思路分析：

（1）Application.InputBox 的最后一个参数必须使用 8 才能实现鼠标单击区域时自动在输入框中产生地址并且返回 range 对象。同时，由于 Application.InputBox 的返回值是 range 对象，所以对 rng 变量赋值时要采用 Set 语句。

（2）尽管工作表中需要转换的区域是 A1:C5，但由于程序中使用了 Intersect 方法获取变量 rng 与活动工作表的已用区域的交集，所以选择区域时可以更自由一些，直接选择 A:C 区域即可，程序会自动忽略空白区域。

（3）本例中对变量 arr 一次性赋值，从而生成数组，所以定义变量时一定要采用变体型，不能在变量名称后面指定数组的维数，以及上标、下标。

（4）数组变量中的每一个元素都允许随意写入、读取或者修改，所以本例中先将区域的值赋予变量，然后在双循环语句中逐一修改数组中每个元素的值，将它们转换成首字母大写状态。待所有字符转换成功后一次性导出到区域中。

（5）StrConv 函数是一个功能强大的转换函数，其第 2 参数使用 vbUpperCase、vbLowerCase、vbProperCase 时可分别实现将英文单词转换成全部大写、全部小写和首字母大写形式。

（6）由于本例需要大量读取和写入单元格，所以极有必要使用数组。

本例文件参见光盘：..\第十三章\13-7 将指定区域的单词统一为首字母大写.xlsm

13.3.2　罗列不及格学生的姓名、科目和成绩

案例要求：图 13.25 中包含若干名学生的成绩，要求将所有不及格学生的姓名、科目与成绩一并罗列出来，排为三列存放在 J 列到 L 列中。

知识要点：ReDim Preserve 语句、UBound 函数、Transpose 函数、Array 函数。

程序代码：

```
Sub 罗列不及格人员信息()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
Dim arr, arr2(), i As Integer, j As Integer, TargetCount As Integer
arr = ActiveSheet.UsedRange.Value
For i = 2 To UBound(arr)
  For j = 2 To UBound(arr, 2)
    If arr(i, j) < 60 Then
      TargetCount = TargetCount + 1
      ReDim Preserve arr2(1 To 3, 1 To TargetCount)
      arr2(1, TargetCount) = arr(i, 1)
      arr2(2, TargetCount) = arr(1, j)
      arr2(3, TargetCount) = arr(i, j)
    End If
  Next j
```

```
  Next i
  Range("J1:L1") = Array("姓名", "科目", "成绩")
  Range("j2").Resize(TargetCount, 3) = WorksheetFunction.Transpose(arr2)
  Range("j1").CurrentRegion.Borders.LineStyle = xlContinuous
End Sub
```

当执行过程后得到如图 13.26 所示的结果。

	A	B	C	D	E	F	G	H
1	姓名	语文	数学	化学	政治	计算机	英语	法律
2	计尚云	76	89	52	80	73	65	81
3	赵国	83	63	64	92	92	80	100
4	罗至贵	96	61	85	99	62	77	55
5	徐大鹏	100	84	50	79	55	55	90
6	张志坚	64	52	65	69	65	98	99
7	朱千文	70	64	58	58	82	70	71
8	赵秀文	86	66	82	60	59	79	54
9	梁爱国	73	96	63	90	69	64	96
10	梁兴	82	82	71	54	78	85	96
11	陈随机	92	51	77	96	71	84	75

图 13.25　成绩表

	G	H	I	J	K	L
1	英语	法律		姓名	科目	成绩
2	65	81		计尚云	化学	52
3	80	100		罗至贵	法律	55
4	77	55		徐大鹏	化学	50
5	55	90		徐大鹏	计算机	55
6	98	99		徐大鹏	英语	55
7	70	71		张志坚	数学	52
8	79	54		朱千文	化学	58
9	64	96		朱千文	政治	58
10	85	96		赵秀文	计算机	59
11	84	75		赵秀文	法律	54

图 13.26　不及格学生信息

思路分析：

（1）由于编写代码时无法获知符合条件的成绩个数，所以声明数组变量 arr2 时不指明数组的维数和每一维的下标、上标，而是在循环语句中通过 ReDim Preserve 语句为数组重置维数和上标。不过需要强调的是 ReDim Preserve 语句并非可以随意修改数组的维数和下标、上标，它只能修改一次数组的维数和每一维的下标，最末一维的上标才可以反复地修改。

（2）本例中所有符合条件的数据所形成的数组包含 3 列，即姓名、科目和成绩，其行数则由符合条件的成绩个数决定。换而言之，数组的第二维的下标和上标可以固定为 1 和 3，但第一维的下标不确定，而 VBA 规定 ReDim Preserve 语句只能反复修改最末一维的下标，所以指定变量的维数时不能使用“ReDim Preserve arr2(1 To TargetCount，1 To 3)”，而是使用“ReDim Preserve arr2(1 To 3, 1 To TargetCount)”，当对数组赋值完成后再使用转置函数 Transpose 转置方向。

（3）本例中对变量 arr2 赋值的方式比较重要，源代码如下：

```
arr2(1, TargetCount) = arr(i, 1)
arr2(2, TargetCount) = arr(1, j)
arr2(3, TargetCount) = arr(i, j)
```

由于数组 arr2 包含 3 行，列数由变量 TargetCount 决定，即第 TargetCount 列是最后一列，所以将符合条件的姓名、科目和成绩追加到数组中时分别采用 arr2（1，TargetCount）、arr2（2，TargetCount）和 arr2（3，TargetCount）。

而 arr(i, 1)的来历是：由于姓名处于数组 arr 的第 1 列第 i 行；科目处于数组 arr 的第 1 行第 j 列，所以引用科目时使用(1, j)；成绩处于第 i 行第 j 列，所以引用成绩时采用 arr(i, j)。

（4）将数组 arr2 导出到工作表中时，不能像 Range.Copy 那样只指定目标区域的左上角单元格，而是必须指定与数组相同高度与宽度的区域，所以本例对 Range("j2")单元格的 Resize 属性指高度为 TargetCount，宽度为 3。其中 TargetCount 表示符合条件的成绩个数，3 代表姓名、科目和成绩三项标题所占用的列宽。

本例如果不使用数组，那么代码执行时间将延长到两倍以上。

本例文件参见光盘：..\第十三章\13-8 罗列不及格人员姓名、科目和成绩.xlsm

13.3.3　跨表搜索学员信息

案例要求：如图 13.27 所示的工作簿中有若干个成绩表，每个成绩表中包含学生姓名、班级、

学号与成绩。现要求利用 VBA 代码在所有成绩表中按姓名查找学员信息，找到后逐一罗列在“成绩查询”工作表中。程序必须支持糊模匹配，例如查找“黄”则将所有姓名中包含“黄”字的学生信息罗列出来。

	A	B	C	D	E	F
1	姓名	班级	学号	成绩		
2	张松	一班	86613	69		
3	刘大年	一班	51957	94		
4	陈云	一班	93332	74		
5	张铁戈	一班	39138	66		
6	刘五	一班	88766	87		
7	李新	一班	03512	90		
8	王六	一班	83821	93		
9	周维同	一班	43243	83		
10	罗通	一班	80235	56		

一班成绩表 二班成绩表 三班成绩表 成绩查询

图 13.27 成绩表

知识要点：Application.InputBox 方法、Range.Find 方法、ReDim Preserve 语句、Transpose 函数、Array 函数。

程序代码：

```
Sub 跨表搜索学员信息()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Dim Tim As Date, arr(), TargetCount As Long, 姓名 As String, firstAddress As
String, cell As Range
  Range("A:F").Clear
  姓名 = Application.InputBox("您想查找谁的成绩？可以输入一个或者多字", "查找目标
", , , , , , , 2)
  If 姓名 = "False" Then Exit Sub
  Tim = Timer
For i = 1 To Worksheets.Count - 1
  Set cell = Worksheets(i).Range("A:A").Find(what:=姓名, LookAt:=xlPart)
  If Not cell Is Nothing Then
    firstAddress = cell.Address(0, 0)
    Do
      TargetCount = TargetCount + 1
      ReDim Preserve arr(1 To 6, 1 To TargetCount)
      arr(1, TargetCount) = WorkSheets(i).Name
      arr(2, TargetCount) = cell.Address(0, 0)
      arr(3, TargetCount) = cell.Value
      arr(4, TargetCount) = cell.Offset(0, 1).Text
      arr(5, TargetCount) = cell.Offset(0, 2).Text
      arr(6, TargetCount) = cell.Offset(0, 3).Text
      Set cell = Sheets(i).Range("A:A").FindNext(cell)
      If cell.Address(0, 0) = firstAddress Then Exit Do
    Loop
    End If
  Next
  If TargetCount > 0 Then
  Worksheets("成绩查询").Range("A2:F" & TargetCount) = WorksheetFunction.
Transpose(arr)
  Worksheets("成绩查询").Range("A1:F1") = Array("工作表", "地址", "姓名", "班级
", "学号", "成绩")
  Worksheets("成绩查询").Range("A1:F" & TargetCount).Borders.LineStyle =
xlContinuous
  End If
  MsgBox Format(Timer - Tim, "0.00秒")
End Sub
```

执行以上过程后会弹出如图 13.28 所示的输入框，在其中输入“刘”然后单击“确定”按钮，程序会在所有成绩表的 A 列中搜索包含“刘”的姓名，然后将与该学生相关的一切信息罗列在“成绩查询”表中，同时将目标所在的工作表和单元格地址一并罗列出来，效果如图 13.29 所示。

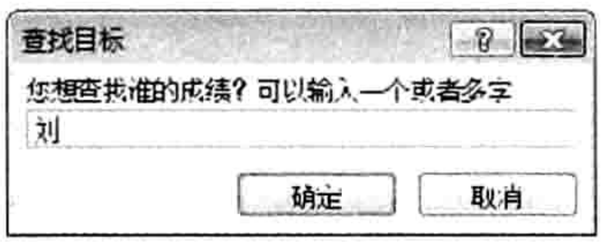


图 13.28　指定查找目标

	A	B	C	D	E	F
1	工作表	地址	姓名	班级	学号	成绩
2	一班成绩表	A3	刘大年	一班	51957	94
3	一班成绩表	A6	刘五	一班	88766	87
4	一班成绩表	A11	刘丽	一班	50118	62
5	二班成绩表	A4	刘林年	二班	3258	74
6	二班成绩表	A11	刘稳	二班	75822	85
7	三班成绩表	A3	刘重山	三班	69162	56
8	三班成绩表	A5	刘新政	三班	32145	51

图 13.29　查找结果

思路分析：

（1）本例的 Sub 过程在执行搜索之前首先清除 Range("A:F")区域的值，其目的在于避免上一次的搜索结果影响本次搜索结果。例如上一次搜索时有 8 个符合条件的目标，本次只有 5 个符合条件的目标，由于本次搜索结果没有完全覆盖上一次的结果，因此会多出 3 行数据。

（2）由于在编写代码时无法确定符合搜索条件的目标的具体数量，因此声明数组变量 arr 时没有指定维数和下标、上标，而是在循环语句 Do Loop 配合 Range.Find 方法执行批量查找时累加计数器，然后以计数器的值为上标重置数组，从而确保数组的上标随目标个数相应变化。

（3）在执行查找时，由于允许单字查找，因此应该使用模糊匹配方式，将 Range.Find 方法的 LookAt 参数赋值为 xlPart。

（4）由于搜索条件由用户随意指定，那么就存在没有符合条件的目标的可能性，因此在将数组 arr 的值输出到工作表中之前应该判断是否存在符合条件的值，否则代码会赋值出错。

（5）数组 arr 是二维数组，第一维可以看作数组的行，第二维看作数组的列，由于 VBA 只允许反复修改最末一维的上标，而数组 arr 在 Do Loop 循环语句中第二维是保持不变的，只累加第一维的上标，因此在对变量 arr 赋值时只能将它当作横向数组处理，最后导出数组的值时再将数组 arr 转置为纵向数组。

本例文件参见光盘：..\第十三章\13-9 跨表搜索学员信息.xlsm

13.3.4　将职员表按学历拆分成多个工作表

案例要求：将图 13.30 中所示的职工信息表按学历拆分成多个工作表，每个工作表的名字以学历命名。学历包含小学、初中、中专和大学 4 类。

知识要点：Array 函数、UBound 函数、ReDim Preserve 语句、Transpose 函数、Erase 语句、Range.CurrentRegion 属性。

程序代码：

	A	B	C	D
1	姓名	学历	年龄	性别
2	计尚云	大学	80	男
3	赵国	中专	100	男
4	罗至贵	中专	96	男
5	徐大鹏	中专	61	男
6	张志坚	大学	85	男
7	朱千文	大学	99	女
8	赵秀文	初中	62	女
9	梁爱国	大学	77	男
10	梁兴	中专	55	男
11	陈随机	大学	100	男

职工信息表

图 13.30　职工信息表

```
'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
Sub 将职员表按学历拆分成多工作表()
  Dim 学历, data, arr()
  Dim i As Byte, RowCount As Integer, TargetCount As Integer
  On Error Resume Next
  学历 = Array("小学", "初中", "中专", "大学")
  data = Range("a1").CurrentRegion.Value
  Application.DisplayAlerts = False
  For i = 0 To UBound(学历)
    TargetCount = 0
```

```
For RowCount = 2 To UBound(data)
  If data(RowCount, 2) = 学历(i) Then
    TargetCount = TargetCount + 1
    ReDim Preserve arr(1 To 4, 1 To TargetCount)
    arr(1, TargetCount) = data(RowCount, 1)
    arr(2, TargetCount) = data(RowCount, 2)
    arr(3, TargetCount) = data(RowCount, 3)
    arr(4, TargetCount) = data(RowCount, 4)
  End If
Next RowCount
If TargetCount > 0 Then
      Worksheets(学历(i)).Delete
      Worksheets.Add(after:=Worksheets(Worksheets.Count)).Name = 学历(i)
      With Worksheets(学历(i))
        .Range("a1:d1") = Array("姓名", "学历", "年龄", "性别")
        .Range("A2").Resize(TargetCount, 4) = WorksheetFunction.Transpose(arr)
        .UsedRange.Borders.LineStyle = xlContinuous
      End With
      Erase arr
    End If
  Next i
End Sub
```

执行以上过程后将会把职工信息表拆分成“小学”、“初中”、“中专”和“大专”4个工作表，每个表中保存具有同一类学历的职工信息，结果如图13.31所示。

	A	B	C	D	E	F
1	姓名	学历	年龄	性别		
2	计尚云	大学	80	男		
3	张志坚	大学	85	男		
4	朱千文	大学	99	女		
5	梁爱国	大学	77	男		
6	陈随机	大学	100	男		
7	刘子中	大学	84	男		
8	诸有光	大学	50	男		
9	李文新	大学	55	女		
10	梁今明	大学	65	男		
11	柳三秀	大学	69	女		

职工信息表 小学 初中 中专 大学

图13.31 拆分结果

思路分析：

（1）本例中使用了两个数组变量，第一个用于一次性读取职工信息表中的数据，所以声明变量时采用变体型变量；第二个数组用于逐个写入符合条件的职工信息，所以声明变量时采用数组变量但不指定数组维数，当进入For Next循环语句后利用ReDim Preserve语句重置数组的维数和上标。不过由于ReDim Preserve只能更改数组最末维的上标，所以定义数组的维数时不能定义为列数等于4、行数等于符合条件的数据个数，而是定义为行数等于4、列数等于符合条件的数据个数，当循环完成后需要再利用Transpose函数将数组转置方向并导出到工作表中。

（2）数组arr需要多次使用，所以每一次使用完后都需要通过Erase语句释放变量的值，避免影响第二轮赋值。

（3）将数组data的值赋予数组arr时，由于两者的方向相反，所以索引号的顺序也相反：

```
arr(1, TargetCount) = data(RowCount, 1)
arr(2, TargetCount) = data(RowCount, 2)
arr(3, TargetCount) = data(RowCount, 3)
arr(4, TargetCount) = data(RowCount, 4)
```

（4）Erase语句释放数组时会将数组的值全部清空，同时将数组的维数、上标等一切信息都清

除，所以在下次引用该动态数组之前必须使用 ReDim 语句重新定义该数组变量的维数，否则无法对该数组写入新值。

（5）使用代码删除工作表时，不管工作表是否是空白都会弹出一个提示框，从而影响操作，所以在使用 Worksheets.Delete 方法之前必须通过“Application.DisplayAlerts = False”语句关闭提示。不过 Worksheets.Delete 方法处于循环语句之中，而“Application.DisplayAlerts = False”语句则不适宜放在循环语句中，否则该语句将反复执行，浪费资源。

本例文件参见光盘：..\第十三章\13-10 将职员表整理为学历分类表.xlsm

13.3.5 将选区中的数据在文本与数值之间互换

案例要求：将选区的数据在文本与数值之间互换，由用户选择转换方式和区域。

知识要点：UBound 函数、Intersect 方法、Range.NumberFormatLocal 属性、Msgbox 函数、IF Then 条件语句、For Next 循环语句。

程序代码：

```
Sub 文本与数值互换()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Dim 转换方式 As VbMsgBoxResult, arr(), row As Long, column As Long
  转换方式 = MsgBox("选择是：将文本转换成数字；" + Chr(10) + "选择否：将数字转换成文本。
", vbYesNo, "操作方式")
  With Intersect(Selection, ActiveSheet.UsedRange)
    If 转换方式 = vbYes Then
      .NumberFormatLocal = "G/通用格式"
      .Value = .Value
    Else
      arr = .Value
      For row = 1 To UBound(arr)
        For column = 1 To UBound(arr, 2)
          If Len(arr(row, column)) > 0 Then arr(row, column) = "'" & arr(row, column)
        Next
      Next
      .Value = arr
    End If
  End With
End Sub
```

图 13.32 中所示的成绩区域有部分单元格是数值格式，有部分单元格是文本格式，所以公式“=AVERAGE(B2:B11)”的计算结果可能不准确。对于这种情况有必要通过程序将它们统一格式。

选择 B2:B11 区域后，执行过程“文本与数值互换”弹出如图 13.33 所示的信息框。

B12 =AVERAGE(B2:B11)

	A	B	C	D
1	姓名	成绩		
2	计尚云	52		
3	赵国	71		
4	罗至贵	93		
5	徐大鹏	90		
6	张志坚	69		
7	朱千文	99		
8	赵秀文	94		
9	梁爱国	52		
10	梁兴	98		
11	陈随机	68		
12	平均	74.875		

图 13.32 成绩区域有文本导致计算不正确

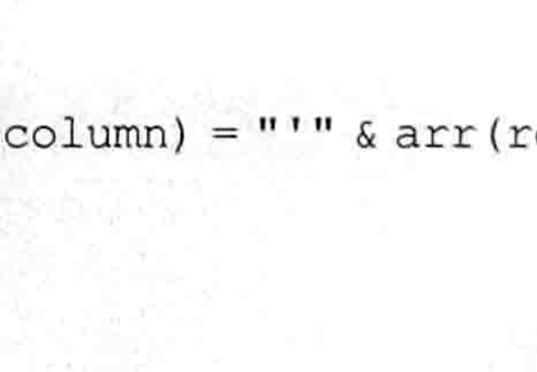


图 13.33 让用户选择转换方式

单击信息框中的“是”按钮把选区中的所有单元格统一为数值格式，公式“=AVERAGE(B2:B11)”才能获得正确结果，结果如图 13.34 所示。

B12　=AVERAGE(B2:B11)

	A	B	C	D
1	姓名	成绩		
2	计尚云	52		
3	赵国	71		
4	罗至贵	93		
5	徐大鹏	90		
6	张志坚	69		
7	朱千文	99		
8	赵秀文	94		
9	梁爱国	52		
10	梁兴	98		
11	陈随机	68		
12	平均	78.6		

图 13.34　文本转数值后的公式计算结果

如果需要将单元格批量修改为文本格式，执行本例的过程后，在图 13.33 所示信息框中单击“否”按钮即可。

思路分析：

（1）让文本格式的数字显示为数值需要分两个步骤，先将单元格的数字格式修改为“G/通用格式”，然后重新对单元格赋值，这一步可以批量操作。

让数值显示为文本则在数字前添加半角状态的单引号即可，不过这一步无法批量操作，所以本例将选区的值导入到数组中，然后在数组中所有非空元素前添加“'”，最后一次性将数组的值导出到区域中。

（2）Range.NumberFormatLocal 属性表示单元格的数字格式，相当于“设置单元格格式”对话框中“数字”选项卡中的各项设置功能。可以通过该对话框获取各种数字格式，然后应用于 VBA 中。图 13.35 中显示了 Excel 自带的单元格数字格式代码，这些代码都可以直接用于 VBA 中对 Range.NumberFormatLocal 属性赋值，从而调整单元格的显示状态。

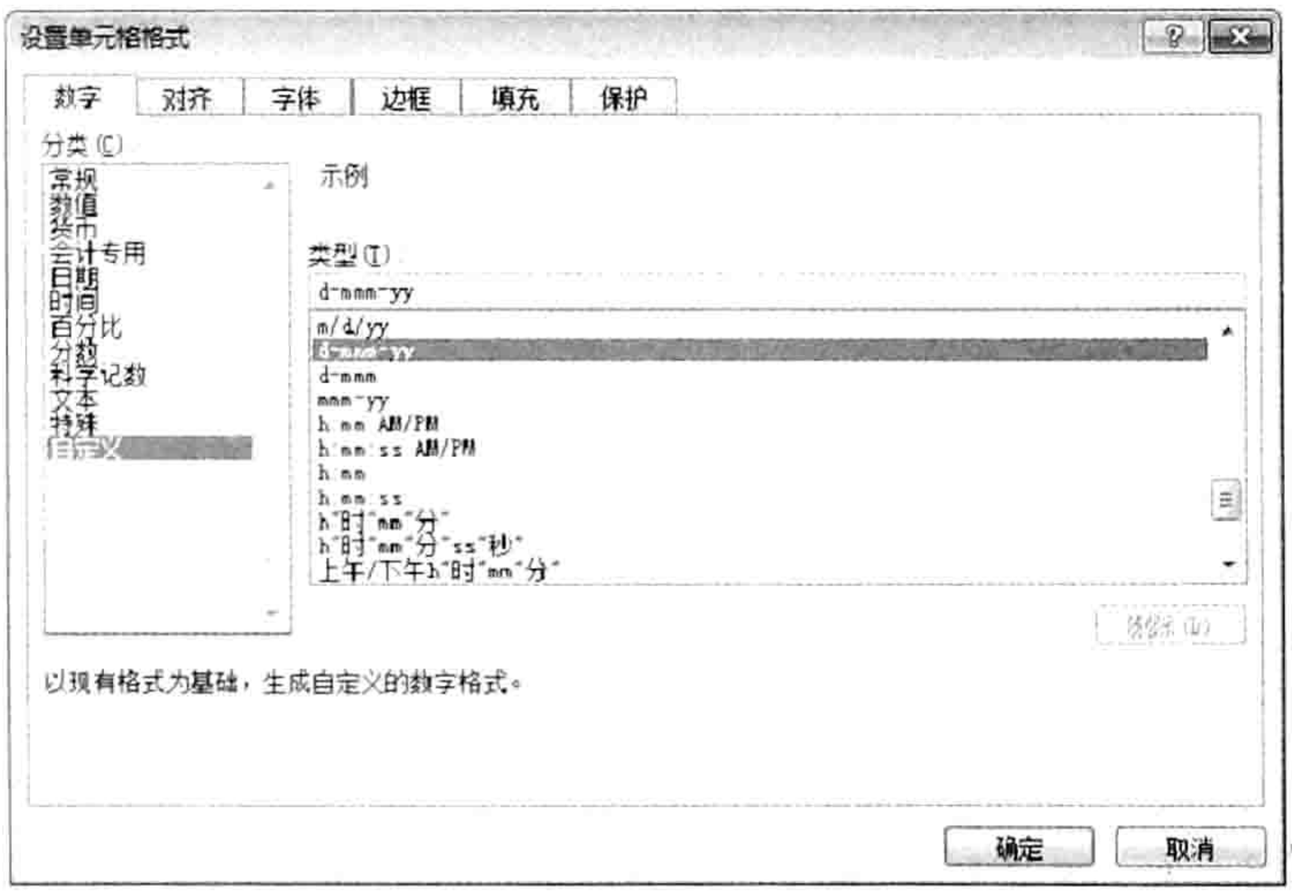


图 13.35　Excel 自带的数字格式代码

（3）本例代码仅对单个区域进行转换，在本书第 22 章中还会提供多区域转换的代码。

CD-ROM　本例文件参见光盘：..\第十三章\13-11 将选区的数据在文本与数值间互换.xlsm

13.3.6　获取两列数据的相同项

案例要求：图 13.36 中的 A、B 两列包含两届参赛队员的信息，要求比较两列数据的异同，并

分别罗列出来，保存在 D 列和 E 列。

知识要点：Transpose 函数、Filter 函数、UBound 函数、ReDim Preserve 语句、For Next 语句、Range.Resize 属性。

程序代码：

```
Sub 获取两列的相同项与不同项()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Dim arr1, arr2, arr3(), arr4(), Item As Integer, Count1 As Integer, count2
As Integer
  arr1   =   WorksheetFunction.Transpose(Range([A2],   Cells(Rows.Count,   1).
End(xlUp)).Value)
  arr2   =   WorksheetFunction.Transpose(Range([B2],   Cells(Rows.Count,   2).
End(xlUp)).Value)
  For Item = 1 To UBound(arr1)
    If UBound(Filter(arr2, arr1(Item), True)) >= 0 Then
      Counter1 = Counter1 + 1
      ReDim Preserve arr3(1 To Counter1)
      arr3(Counter1) = arr1(Item)
    Else
      Counter2 = Counter2 + 1
      ReDim Preserve arr4(1 To Counter2)
      arr4(Counter2) = arr1(Item)
    End If
  Next Item
  Range("d1:e1") = Array("相同项", "不同项")
  Range("D2").Resize(Counter1, 1) = WorksheetFunction.Transpose(arr3)
  Range("E2").Resize(Counter2, 1) = WorksheetFunction.Transpose(arr4)
End Sub
```

执行以上过程后，程序会瞬间找出两届参赛队员的相同项与不同项，并将它们分别罗列在 D 列和 E 列中，如图 13.37 所示。

	A	B
1	2011届	2012届
2	计尚云	朱千文
3	赵国	诸有光
4	罗至贵	周华章
5	徐大鹏	曲华国
6	张志坚	李文新
7	朱千文	陈随机
8	赵秀文	陈强生
9	梁爱国	刘文喜
10	梁兴	赵国
11	陈随机	柳三秀

参赛队员信息

图 13.36　两届参赛队员的信息表

	A	B	C	D	E
1	2011届	2012届		相同项	不同项
2	计尚云	朱千文		赵国	计尚云
3	赵国	诸有光		朱千文	罗至贵
4	罗至贵	周华章		陈随机	徐大鹏
5	徐大鹏	曲华国			张志坚
6	张志坚	李文新			赵秀文
7	朱千文	陈随机			梁爱国
8	赵秀文	陈强生			梁兴
9	梁爱国	刘文喜			
10	梁兴	赵国			
11	陈随机	柳三秀			

参赛队员信息

图 13.37　提取两列姓名的相同项与不同项

思路分析：

（1）首先将两列数据通过 Transpose 函数转换成一维数组，并赋值给两个变体型变量，此时两个变量被转换成数组。由于区域引用通过 Transpose 函数转换后是横向的一维数组，所以可以作为 Filter 函数的参数参与运算。

如果要求比较的对象是两行数据，那么必须使用 Transpose 函数转置两次才能将区域引用转换成一维数组，否则不支持 Filter 函数。

例如判断 A1:E1 区域中是否包含“上海”二字，如果采用数组思路实现，则应在 Range("A1:E1") 对象外套两层 Transpose，从而使一维横向的区域引用转换成一维数组，然后通过 Filter 函数以“上海”作为筛选条件生成新数组，并根据数组的上标判断 A1:E1 区域中是否包含“上海”二字。完整代码如下：

```
Sub 判断 A1 到 E1 中是否包含上海()'代码存放位置：模块中
```

```
    Dim arr
    arr = WorksheetFunction.Transpose(WorksheetFunction.Transpose(Range("A1:E1")))
    MsgBox IIf(UBound(Filter(arr, "上海", True)) <= 0, "不包含", "包含")
End Sub
```

如果直接将 Range("A1:E1")赋值给变量 arr，或者将转置后的 WorksheetFunction.Transpose (Range("A1:E1"))赋值给 arr，数组 arr 都不能作为 Filter 函数的参数参与运算，因为这两种方式产生的数组都不是一维数组。

（2）判断一个数组中是否包含某个字符也可以采用 Instr 函数+Join 函数组合实现。其中 Join 函数用于将数组转换成字符串，Instr 函数则用于计算查找的目标字符在此字符串中的首次出现位置，如果位置大于 0 则表示包含。本例代码若改用 Instr 函数+Join 函数组合实现，那么代码如下：

```
Sub 判断 A1 到 E1 是否包含上海()'代码存放位置：模块中
    Dim arr
    arr = WorksheetFunction.Transpose(WorksheetFunction.Transpose(Range("A1:E1")))
    MsgBox IIf(InStr("@" & Join(arr, "@") & "@", "@上海@") > 0, "包含", "不包含")
End Sub
```

以上代码中的“@”的功能可以理解为匹配方式，不使用“@”时表示模糊匹配，使用“@”时表示精确匹配。

例如当 A1:E1 区域中的值分别是北京、上海滩、香港、重庆、成都时，那么使用“@”后代码的执行结果是“不包含”。因为 Join 函数的合并结果是“@北京@上海滩@香港@重庆@成都@”，在其中查找“@上海@”时将返回 0，因为不存在“@上海@”。但是如果不使用“@”，那么 Join 函数的合并结果是“北京上海滩香港重庆成都”，在此字符串中查找“上海”，其结果为 3。

（3）本例为了代码的通用性，对变量 arr1 和变量 arr2 赋值时采用了 Range.End 属性来定位最后一个单元格，从而使 A、B 列的数据在增减时代码可以自动识别数据区域。

本例文件参见光盘：..\第十三章\13-12 获取两列数据的相同项.xlsm

13.3.7　无人值守的多工作簿自动汇总

案例要求：在某个文件夹中有若干个车间产量表，多个工作簿之间的工作表数量不同、多个工作表的数据行数也不同，但是所有工作表的格式一致，都包含姓名、产品、产量和不良品 4 项，图 13.38 和图 13.39 分别是待汇总的文件夹和待汇总的产品信息。

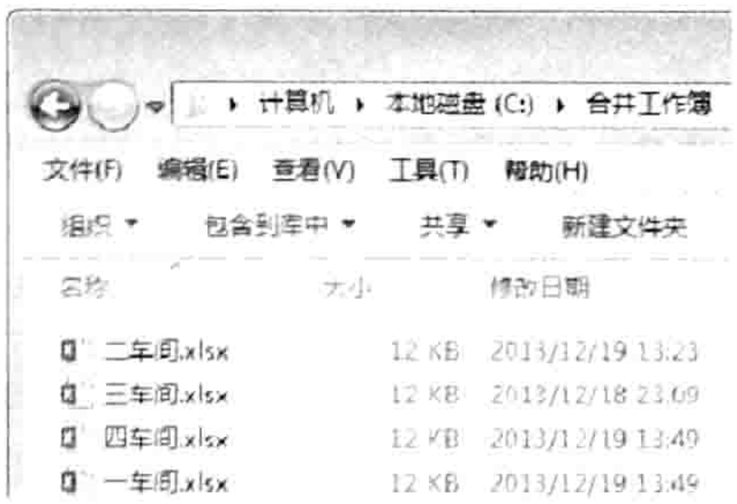


图 13.38　待汇总的文件夹

	A	B	C	D
1	姓名	产品	产量	不良品
2	刘玲玲	异形螺帽	97	4
3	刘兴宏	钢轴	118	13
4	陈深渊	外六角螺丝	110	10
5	黄山贵	旋转螺丝	84	10
6	张秀文	十字盘头螺钉	116	3
7	陈星望	双头螺丝	118	6
8	黄明秀	钢轴	110	12
9	廖工庆	方螺帽	97	8
10	肖小月	扳手螺丝	119	5
11	赵月峨	钢轴	80	13

图 13.39　待汇总的产品信息

现要求设计一个用于汇总的工作簿，将工作簿放在待汇总的文件夹中，打开工作簿时能全自动汇总。

知识要点：Workbook_Open 事件、Workbooks.Open 方法、For Each Next 循环、ReDim Preserve 语句、Workbook.Close 方法、Range.Consolidate 方法、Range.CurrentRegion 属性。

程序代码：

```
'①代码存放位置：ThisWorkbook②随书光盘中有每一句代码的含义注释
Private Sub Workbook_Open()
  Dim FilePath As String, FileName As String, sht As Worksheet, Item As Integer,
arr()
  FilePath = ThisWorkbook.Path
  FilePath = FilePath & IIf(Right(FilePath, 1) = "\", "", "\")
  Application.ScreenUpdating = False
  FileName = Dir(FilePath & "*.xls*")
  Do
    If Len(FileName) = 0 Then Exit Do
    If FileName <> ThisWorkbook.Name Then
      With Workbooks.Open(FilePath & FileName)
        For Each sht In .Worksheets
          Item = Item + 1
          ReDim Preserve arr(1 To Item)
          arr(Item) = "'" & FilePath & "[" & FileName & "]" & sht.Name & "'!R1C2:R"
& sht.UsedRange.Rows.Count & "C4"
        Next sht
      End With
      Workbooks(FileName).Close , False
    End If
    FileName = Dir
  Loop
  Range("a1").Consolidate   Sources:=arr,   Function:=xlSum,   TopRow:=True,
LeftColumn:=True, CreateLinks:=False
  Range("a1") = "产品名称"
  Range("a1").CurrentRegion.EntireColumn.AutoFit
  Range("a1").CurrentRegion.Borders.LineStyle = xlContinuous
  Application.ScreenUpdating = True
End Sub
```

新建一个工作簿，将以上代码放在 ThisWorkbook 中，然后将工作簿保存在待汇总的文件夹中，命名为“汇总表.xlsm”，最后关闭工作簿。

由于工作簿中的代码使用了 Workbook_Open 事件，它能在打开工作簿时自动汇总文件所在路径下的所有工作簿中的数据，因此尽管“汇总表.xlsm”是空白工作簿，但是当双击打开该工作簿后可以发现“汇总表.xlsm”的活动工作表中已经汇总了图 13.38 中所示的文件夹中的所有数据，效果如图 13.40 所示。

如果在图 13.38 所示的文件夹中放置更多的工作簿，或者在工作簿中加入更多的工作表，再或者在工作表中加入更多的数据，不需要修改“汇总表.xlsm”中的代码，直接打开工作簿就能刷新汇总结果。

如果将“汇总表.xlsm”复制到其他任意文件夹中再打开，工作簿会自动汇总该文件夹中的数据。不过由于代码中指定了汇总列是 B 列到 D 列，当格式与如图 13.39 所示的格式不同时需要修改区域范围。

	A	B	C
1	产品名称	产量	不良品
2	异形螺帽	416	22
3	扳手螺丝	1535	91
4	铝连杆	634	40
5	方螺帽	1497	72
6	连接螺丝	911	87
7	压花螺帽	516	31
8	方管连杆	1033	56
9	钢铀	1675	133
10	双头螺丝	1183	83
11	外六角螺丝	651	36

图 13.40　汇总结果

思路分析：

（1）由于要求全自动汇总数据，因此 Sub 过程采用 Workbook_Open 事件，代码必须存放在 ThisWorkbook 窗口中。

（2）由于是汇总代码所在工作簿路径下的其他工作簿，因此文件路径设置为 ThisWorkbook.Path，不需要用户手工选择路径。

（3）Excel 文件的常用扩展名包含“*.xls”、“*.xlsx”和“*.xlsm”，为了让代码对这 3 种格

式的工作簿都有效，使用 Dir 函数搜索文件时用“*.xls*”作为搜索条件，通配符“*”代表任意长度的任意字符。

（4）本例中 Do Loop 循环语句配合 Dir 函数从文件夹中搜索文件时，由于“汇总表.xlsm”本身也符合搜索条件，但是“汇总表.xlsm”不能参与汇总，因此在循环语句中需要使用条件语句排除“汇总表.xlsm”。

（5）Range.Consolidate 方法的功能对应于“数据”选项卡中的“合并计算”菜单，它可以跨工作表或者跨工作簿汇总数据，比数据透视表简单，但不如数据透视表功能强大。

使用 Range.Consolidate 方法汇总其他工作簿中的数据时可以不用打开要汇总的工作簿，将工作簿的路径和要汇总的区域地址放在 Range.Consolidate 方法的参数中即可。不过由于本例中不同工作簿中的工作表数量不一致、工作表中的数据行数也不一致，在未打开工作簿的前提下无法获取工作表名称和已用区域的地址，因此本例仍然在 Do Loop 循环中使用了 Workbooks.Open 方法逐一打开工作簿，其目的不是汇总数据，而是为了获取所有工作表名称和已用区域地址。

（6）Range.Consolidate 方法的语法如下：

```
Range.Consolidate(Sources, Function, TopRow, LeftColumn, CreateLinks)
```

Range.Consolidate 方法拥有 5 个可选参数，每个参数的含义见表 13-3。

表 13-3　Range.Consolidate方法的参数说明

参数名称	功能说明
Sources	以文本引用字符串数组的形式给出合并计算的数据源地址，该数组采用 R1C1 样式表示法。这些引用必须包含将要合并计算的工作表的完整路径
Function	用于指定合并计算的类型，可选项包含 xlAverage（平均）、xlCount（计数）、xlCountNums（数值个数）、xlMax（最大值）、xlMin（最小值）、xlProduct（乘）、xlStDev（基于样本的标准偏差）、xlStDevP（基于全体数据的标准偏差）、xlSum（总计）、xlUnknown（未指定任何分类汇总函数）、xlVar（基于样本的方差）、xlVarP（基于全体数据的方差）
TopRow	如果为 True，则基于合并计算区域中首行内的列标题对数据进行合并。如果为 False，则按位置进行合并计算。默认值为 False。
LeftColumn	如果为 True 则基于合并计算区域中左列内的行标题对数据进行合并计算。如果为 False，则按位置进行合并计算。默认值为 False
CreateLinks	如果为 True，则让合并计算使用工作表链接。如果为 False，则让合并计算复制数据。默认值为 False

（7）Range.Consolidate 方法在汇总数据方面不如数据透视表强大、灵活，如果要跨工作簿合并数据而且数据比较凌乱时应用多重合并计算数据区域的数据透视表。

本例中的代码限制了合并计算的列数，如果读者的实际需求与本例不一致，仅需要修改代码中的区域范围即可。代码中的“C2”和“C4”代表参与合并计算的区域限制在第 2 至第 4 列中，忽略其他数据。

本例文件参见光盘：..\第十三章\合并工作簿\汇总表.xlsm

◆　◆　◆

第 14 章　正则表达式与 VBA

正则表达式是一门极其强大的字符搜索技术，几乎所有程序语言都可以调用正则表达式来处理字符串，从而强化程序自身的功能，Excel VBA 亦不例外。在 VBA 中可以通过添加引用获取正则表达式的方法与属性，实现比 VBA 自身更强大的字符处理功能。本章阐述正则表达式的语法与属性、方法、匹配原则等知识，同时展示在 VBA 中调用正则表达式的方法与案例。

14.1　何谓正则表达式

使用正则表达式前，有必要了解正则表达式的起源、特点，以及 VBA 采用何种方式调用正则表达式的资源。

14.1.1　概念

正则表达式英文名为 Regular Expression，通常缩写成“regex”。它是一种描述字符串结构模式的形式化表达方法，通常用于搜索和替换符合某个模式的文本内容。

正则表达式起源于 UNIX 系统中，微软公司于 2002 年推出.net 平台后才大量推广正则表达式，并开始在各主流平台中部署正则表达式的搜索引擎。不过尽管正则表达式在 Office 软件中应用比较晚，但引进正则表达式后经过了一些改进，使它逐步完善，比 UNIX 系统中早期版本强大得多。

在 VBA 中应用正则表达式主要用于处理文本字符串，通常应用于处理由某些 ERP 系统或者网页导入到 Excel 中的杂乱语句。

从杂乱语句中取出符合某些规则的字符串正是正则表达式的强项。

14.1.2　特点

正则表达式和 VBA 一样不是独立存在的，不可以通过它开发程序。但它可以依附在某个应用程序中，强化该主体程序。

很多软件都支持正则表达式，例如 Microsoft Office、Microsoft Visual Basic 6 或 Microsoft VBScript、.NET Framework、PHP、C#、Java、C++、VB、Javascript、Ruby 等。各软件开发者在引进正则表达式时大多进行了强化、改进，使目前的正则表达式趋于多元化发展，表现出不同的流派。

所以 Excel VBA 所能应用的正则表达式与其他软件的正则表达式也会大同小异，在细节处会有所不同，包括语法和所支持的元字符不同。

例如正则表达式有一个反向预查，采用“?<=”实现，但 VBA 仅支持正向预查，采用“?=”实现。

如果你熟悉其他语言的正则表达式，仍然有必要了解 VBA 对正则表达式的驾驭方式。

正则表达式在 VBA 中的应用是否必要，这要看用户的工作复杂程度。正则表达式的特点是处理字符串，如果工作中出现的字符串简单，或者规律性很强，那么直接用 VBA 足以胜任工作。对于复杂的工作才有必要交给正则表达式来实现。

例如对字符串“美元：123 元人民币：44 元英镑：100 元美元：44 元人民币：300.06 元，日元：55 元、人民币-22.8、人民币 8 毛”中的人民币进行汇总，使用 VBA 处理这种工作显然力不从心，而调用正则表达式的资源进行处理则得心应手，使用简单的几句代码就可以完成。

如果你对DOS还算了解，那么应该知道查找文件名时使用通配符“*”和“?”对提升效率会带来多大的帮助。而正则表达式和DOS的通配符运算思路相近，但支持的通配符（在正则表达式中称之为元字符）更多，匹配方式也更灵活。学习本章后读者会了解正则搜索方式比DOS何止强大千百倍。

14.1.3 调用方式

正则表达式可以为VBA所用，但它并不集成在Excel的内部，必须通过注册、引用才可以调用正则表达式的资源。

在Windows系统中，正则表达式的资源存在于名为“vbscript.dll”的动态链接库文件中。VBA要调用该资源则需要注册此文件，然后在VBA中引用。引用方式包括前期绑定和后期绑定。

注册态动链接库的方法是在“运行”对话框中执行以下语句：

```
regsvr32 vbscript.dll
```

如果成功会弹出如图14.1所示的信息。

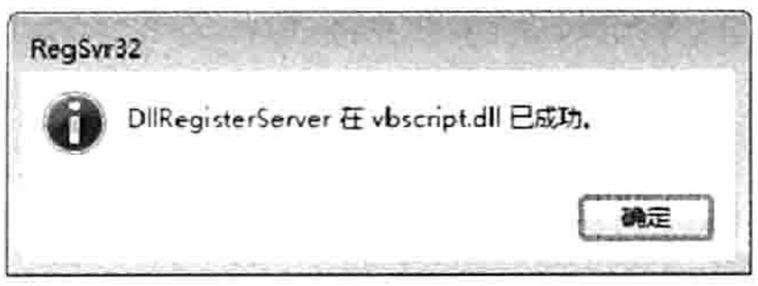


图14.1 注册动态链接库

注册后即可在VBA中引用正则表达式的资源，对文本字符串进行处理。

1. 前期绑定

前期绑定的操作步骤如下。

step 1 按<Alt+F11>组合键打开VBE界面。

step 2 单击菜单中的“工具”→“引用”命令，打开“引用—VBAProject”对话框。

step 3 在引用列表中查找“Microsoft VBScript Regular Expressions 5.5”，并将其打钩，如图14.2所示。

当返回VBE的代码窗口后，即可直接调用正则资源。如图14.3所示，因为已执行前期绑定，所以可以将变量声明为“New VBScript_RegExp_55.RegExp”类型，同时录入变量名称后会弹出属性与方法列表。

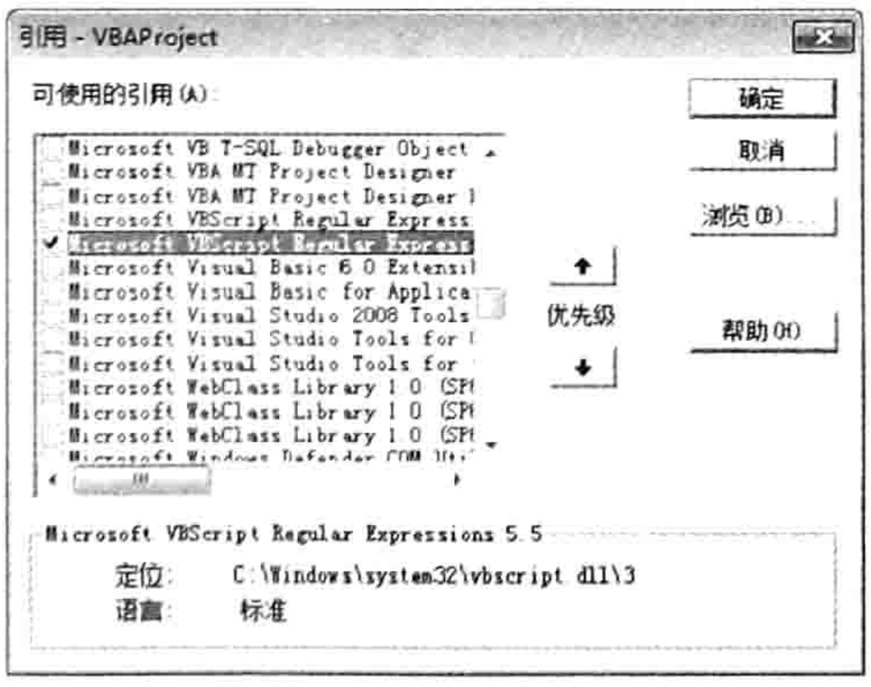


图14.2 引用正则

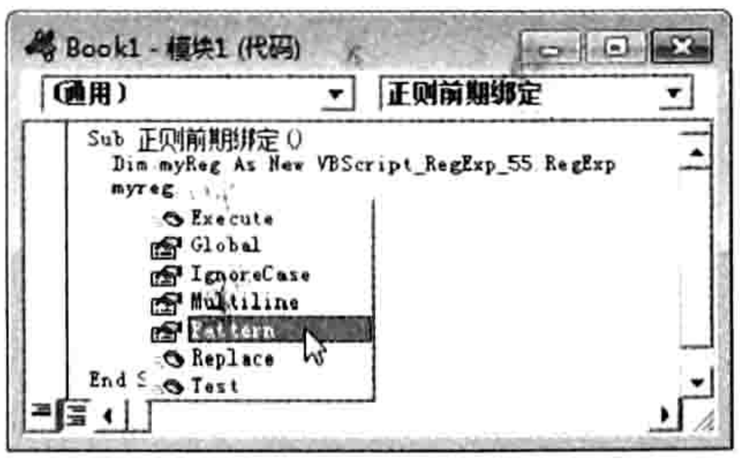


图14.3 自动成员列表

可以通过以下代码测试是否绑定成功：

```
'功能：从字符串中取出所有字母以外的字符。'结果是：我喜欢？！
```

```
Sub 正则前期绑定() '代码存放位置：模块中
  Dim myReg As New VBScript_RegExp_55.RegExp '声明变量
  With myReg                    '引用字典对象
    .Pattern = "[a-z]"          '指定匹配条件为所有字母
    .IgnoreCase = True          '忽略大小写
    .Global = True              '全局匹配
    MsgBox .Replace("我喜欢 VBA?Of course!", "") '将字母替换掉，仅取剩下的字符
  End With
End Sub
```

前期绑定的优点是为编写代码提供便利，可以自动弹出成员列表，缺点是把代码发给客户后，需要客户手工添加引用，否则代码将拒绝执行。

2. 后期绑定

后期绑定是通过代码创建引用，不需要手工操作。即通过 CreateObject 函数创建引用。根据以下案例代码，读者可以明白后期绑定的思路。后期绑定的优点是通用性好，缺点是写代码时没有提示，要求开发者熟悉正则表达式的所有属性与方法。

```
'功能：从字符串中取出所有字母以外的字符。'结果是：我喜欢？！
Sub 正则后期绑定()
  With CreateObject("VBSCRIPT.REGEXP") '创建正则对象并引用该对象
    .Pattern = "[a-z]"          '指定匹配条件为所有字母
    .IgnoreCase = True          '忽略大小写
    .Global = True              '全局匹配
    MsgBox .Replace("我喜欢 VBA?Of course!", "") '将字母替换掉，仅取剩下的字符
  End With
End Sub
```

本例文件参见光盘：..\第十四章\14-1 前期绑定与后期绑定.xlsm

14.2 语法基础

正则表达式的语法比较复杂，本节将一一解析正则的基本语法，14.3 节展示正则表达式的应用。

14.2.1 调用正则表达式的基本格式

正则表达式处理文本有两种方式，其一是从字符串中替换符合条件的字符，其二是从字符串中提取符合条件的字符。为了便于使用和理解，在此为读者提供两个模板，后续使用时将代码套进去就行了。

从字符串中替换符合条件的字符，可用以下格式的模板：

```
Sub 正则表达式格式一()
  Dim MyStr As String                          '声明字符变量，表示储存待替换的字符串
  Dim ResultStr As String                      '声明字符变量，用于获取返回值
  MyStr=                                       '指定待替换的字符串
  With CreateObject("VBSCRIPT.REGEXP")         '创建并引用正则表达式对象
    .Pattern =                                 '指定匹配模式(即搜索条件)
    .IgnoreCase =                              '指定匹配时是否忽略大小写（默认为 false）
```

```
    .Global =                          '指定全局匹配对象，默认为 false
  If .Test(Mystr) Then                 '如果有符合条件的字符
     ResultStr = .Replace(MyStr, "")   '将符合条件的字符替换掉,将剩下的字符赋予变量
    End if
  End With
End Sub
```

从字符串中提取符合条件的字符，可用以下格式的模板：

```
Sub 取出 a 开头的单词()
  Dim MyStr As String, ResultStr As String, Item '声明变量
  MyStr =                                        '指定待搜索的字符串
  With CreateObject("VBSCRIPT.REGEXP")           '创建并引用正则对象
    .Pattern =                                   '指定即搜索条件
    .IgnoreCase =                                '指定匹配时是否忽略大小写（默认为 false）
    .Global =                                    '指定匹配方式，默认为 false 表示只匹配第一个对象
    If .Test(MyStr) Then                         '如果有符合条件的字符
      For Each Item In .Execute(MyStr)           '遍历所有符合条件的对象
        ResultStr = ResultStr & Chr(10) & Item ' 将所有符合条件的对象串连起来
      Next Item
      MsgBox ResultStr                           '报告结果
    End If
  End With
End Sub
```

当然，以上格式并非必需的，可以使用其他的书写方式，不过统一格式便于理解。

为了证明以上两个模板的正确性，随书光盘中的“14-2 两个正则表达式模板.xlsm”工作簿中提供了两段完整的代码，读者可以打开该文件测试代码。

以上两个模板中使用了正则表达式相关的对象、属性和方法，在 14.2.2 节中会详细阐述这些对象、属性和方法。

14.2.2　正则表达式的对象、属性和方法

正则表达式的对象、属性和方法的概念既少也简单，唯一复杂的是匹配规则，也称之为搜索条件。本节主要介绍正则表达式的对象、属性与方法。

1. 正则表达式的对象

正则表达式的对象是用于保存有关正则表达式模式匹配信息的固有全局对象，VBA 通过该对象调用正则表达式的所有资源。

在后期绑定方式下，采用以下语句可以让 VBA 与正则表达式的动态链接库建立链接：

```
Set myReg = CreateObject("VBSCRIPT.REGEXP")
```

当然，变量 myReg 需要声明为 Object 或者 Variant。如果不用变量则可以使用 With 语句直接引用对象。

正则表达式的对象有 3 个属性，包括 Global、IgnoreCase 和 Pattern。其中前两个分别代表是否匹配所有符合条件的目标，以及搜索时是否区分大小写，两者的默认值都是 False，因此使用正则表达式时允许忽略这两个属性值；Pattern 属性代表搜索条件，尽管可以被忽略，但是忽略此属性值后代码将无法正常工作。

2. Global 属性

Global 属性用于指定全局匹配模式，如果值为 True，表示在被查找的字符串中搜索所有符合条件的字符串；如果值为 False，表示找到第一个目标后就停止搜索，默认值为 False。

也就是说，当在字符串“广州市,广州体育学院,广场,广州影剧院”中查找“广州”时，Global 属性可以控制匹配对象是 3 个还是 1 个。当 Global 属性赋值为 True 时，匹配对象有 3 个——“广州市,广州体育学院,广场,广州影剧院；当 Global 赋值为 False 时匹配第一个对象。

3. IgnoreCase 属性

IgnoreCase 属性用于控制查找对象时是否区分大小写，值为 True 表示不区分大小写。

例如在字符串“About an apple”中查找字母 a，如果 IgnoreCase 属性赋值 True，将会有 3 个符合条件的目标，若赋值为 False 则只有两个。

4. Pattern 属性

本属性代表正则表达式模式，通俗地讲就指定搜索条件。正则表达式的重点和难点都在于此属性，其余皆简单明了。

Pattern 属性的赋值决定了搜索结果，它的赋值方式千变万化，极其复杂。可以说学习正则表达式，就是学习 Pattern 属性的用法。

5. Test 与 Replace 方法

Test 方法用于测试是否匹配成功，如果成功则返回 True，否则返回 False。

Replace 方法用于替换符合条件的字符串，它和 VBA 内部的 Replace 函数的用法不同。Replace 方法的语法如下：

```
RegEx.Replace（待替换的字符串，新字符串）
```

例如删除“诺基亚 8310 手机桑塔纳 2000 汽车步步高 KD009 影碟机”中的所有数字：

```
Sub 删除数字()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Dim Mystr As String
  Mystr = "诺基亚 8310 手机桑塔纳 2000 汽车步步高 KD009 影碟机"
  With CreateObject("VBSCRIPT.REGEXP")
    .Pattern = "[0-9]+"
    .Global = True
    If .Test(Mystr) Then MsgBox .Replace(Mystr, "") Else MsgBox "没有数字"
  End With
End Sub
```

执行以上代码可以得到字符串“诺基亚手机桑塔纳汽车步步高 KD 影碟机”。

当然，你也可以将找到的数字替换成任意文本，而不是空文本“”。

代码中的搜索条件“[0-9]+”的含义在 14.2.3 节中会介绍。

本例文件参见光盘：..\第十四章\14-3 删除字符串的所有数字.xlsm

6. Execute 方法

假设上面的案例的需求不是删除字符串中的数字，而是提取所有数字，那么需要使用 Execute 方法。Execute 方法用于执行搜索并返回一个对象集合，该集合中包含了所有符合条件的字符串，可以通过 For Each Next 循环语句逐一提取其中的每个元素。

例如提取“赵大 500 元、钱二 250 元、孙三 580 元、李四 488 元”中所有数字：

```
Sub 提取所有数字()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Dim MyStr As String, ResultStr As String, Item
  MyStr = "赵大 500 元、钱二 250 元、孙三 580 元、李四 488 元"
  With CreateObject("VBSCRIPT.REGEXP")
    .Pattern = "[0-9]+"
    .Global = True
    If .Test(MyStr) Then
      For Each Item In .Execute(MyStr)
        ResultStr = ResultStr & Chr(10) & Item
      Next Item
      MsgBox "字符串中的数字包含：" & ResultStr
    Else
      MsgBox "没有数字"
    End If
  End With
End Sub
```

执行以上过程可以获得如图 14.4 所示的结果。

事实上，能提取所有数字就可以对这些数字执行任何运算。以下过程就可以取得字符串“赵大 500 元、钱二 250 元、孙三 580 元、李四 488 元”的金额之和：

```
Sub 累加所有数字()
  Dim MyStr As String, ResultStr As Long, Item
  MyStr = "赵大 500 元、钱二 250 元、孙三 580 元、李四 488 元"
  With CreateObject("VBSCRIPT.REGEXP")
    .Pattern = "[0-9]+"
    .Global = True
    If .Test(MyStr) Then
      For Each Item In .Execute(MyStr)
        ResultStr = ResultStr + Item
      Next Item
      MsgBox "金额之和为：" & ResultStr
    Else
      MsgBox "没有数字"
    End If
  End With
End Sub
```

执行以上过程可以得到如图 14.5 所示结果。

图 14.4　提取所有数字

图 14.5　累加所有数字

本例文件参见光盘：..\第十四章\14-4 提取字符串的所有数字.xlsm

本例过程中的代码没有考虑小数点的问题，处理小数点的问题稍微复杂一些，涉及的知识点也比较多，在 14.2.4 节中会有详细说明。

14.2.3 匹配的优先顺序

正则表达式的匹配规则主要包括 3 个。

1. 从左到右

当指定匹配条件后，正则表达式会从字符串的左边开始搜索匹配对象。例如在“i mandy you andy”中搜索“andy”，第一个匹配结果为“i mandy you andy”，而不是“i mandy you andy”。

2. 贪婪匹配

正则表达式中有贪婪匹配与惰性匹配两种匹配规则，前者用加号（+）表示，后者用问号（?）表示。当采用贪婪匹配时，正则表达式会匹配尽可能多的字符，即已经搜索到符合条件的对象时仍然继续向右搜索，直到不符合条件时为止。当然也可以理解为它以右边的整个字符串作为匹配对象，如果不成功就向左缩小范围，直到找到第一个完全匹配的对象为止。因此贪婪匹配总是只有一个符合条件的结果。

可以通过下面的字符底纹变化过程展示贪婪匹配的匹配过程。

在字符串“走遍中国中宣部中国人民中文中”中搜索“中.+ ”，其中“中”代表第一搜索条件是“中”，后面的小圆点表示长度为 1 的任意字符（类似 DOS 命令中的通配符“? ”），小圆点后面的“+”表示贪婪匹配，最后有一个空格。整个条件“中.+ ”表示以“中”开头、空格结尾，中间有至少一个字符。它的匹配过程比较复杂，首先从第一个字“走”开始比较，并记录下比较结果，然后用“走遍”、“走遍中”、“走遍中国”、“走遍中国”、“走遍中国中”……去比较，接着再用“遍”、“遍中”、“遍中国”、“遍中国 ”……去比较，每次的比较结果都记录下来，最终用最长的那一个字符串作为结果。本例中有 4 个符合条件的结果：

走遍中国中宣部中国人民中文中
走遍中国中宣部中国人民中文中
走遍中国中宣部中国人民中文中
走遍中国中宣部中国人民中文中

以上 4 行中用灰色底纹标示的字符串皆符合“中.+ ”条件，最终结果为最长的字符串。

搜索条件中的“+”仅作用于小圆点，不作用于“中”，因此“.+”是一个整体。此处“+”的作用是将“长度为 1 的任意字符”这个条件扩展为“长度不少于 1 的任意字符”，允许是无数位。当在字符串中发现多个符合条件的字符串时，取其中最长的那一个作为最终结果。

3. 惰性匹配

当采用惰性匹配时，正则表达式只要找到一个符合条件的对象时就停止本轮搜索，不管其他的字符是否也符合条件，这正是“惰性”的由来。

如果在字符串“走遍中国中宣部中国人民中文中”中搜索“中.+?”，条件中的“? ”代表惰性匹配，即只要找到一个符合条件的字符串就停止本轮搜索。

在“走遍中国中宣部中国人民中文中”中搜索“中.+?”时，仍然从“走”字开始，然后用“走遍”、“走遍中”……去逐一比较，这点和贪婪匹配是一样的，区别在于用第 3 个字“中”开始比较时，比较到第 3 次“中国”刚好符合条件，此时正则会终止本轮比较，然后从第 4 个字“国”开始进入下一轮比较，而采用贪婪匹配方式的话，不会终止本轮比较，还会继续用“中国 中”、“中国 中宣”、“中国 中宣部”……去比较。

因此，在“走遍中国中宣部中国人民中文中”中搜索“中.+?”时最终会有 4 个符合条件的目

标字符串，结果如下：

走遍中国中宣部中国人民中文中
走遍中国中宣部中国人民中文中
走遍中国中宣部中国人民中文中
走遍中国中宣部中国人民中文中

以上标注的4个目标全都是匹配结果，不像贪婪匹配那样仅取其中最长的作为结果。

本例文件参见光盘：..\第十四章\14-5 贪婪匹配与惰性匹配.xlsm

在后面的内容会有关于贪婪匹配与惰性匹配案例和分析。

14.2.4 借用元字符强化搜索功能

在DOS系统中，“*”和“?”是通配符，有特殊的含义。而在正则表达式中将一些具有特殊意义的字符统称为元字符，元字符的数量有很多，本节会一一介绍，并列举应用方法与思路。

1. 可选匹配

如果正则表达式的 Pattern 属性赋值为“AB”，那么搜索文本时就会以“AB”为条件进行查找，即A在前、B在后时才符合条件。例如在“ABS ABOUT FAIR BEER BASS”中搜索，结果为“ABS ABOUT FAIR BEER BASS”，其中灰色底纹者表示匹配结果。

当字母A和B单独出现或者B在前、A在后时皆无法匹配。那么如何才能让A和B在任意情况下皆可匹配呢?

正则表达式提供3种方式。

- 方括号：[]
- 问号：?
- 分隔号：|

以下通过5个案例演示这3种匹方式。

```
Sub 可选匹配() '替换掉所有A和B  (代码存放位置：模块中)
  Dim Mystr As String                          '声明变量
  Mystr = "ABS ABOUT FAIR BEER BASS"           '指定待搜索的文本
  With CreateObject("VBSCRIPT.REGEXP")         '创建正则表达式引用
    .Pattern = "[AB]"                          '匹配A或者B，无先后顺序
    .Global = True                             '全局匹配
    '如果匹配成功，将找到的所有对象替换成空文本
   If .Test(Mystr) Then MsgBox .Replace(Mystr, "")
    End With
End Sub
```

以上代码中 Pattern 属性采用“[AB]”，表示A和B皆为可选的匹配对象，单独的A或者单独的B都可以匹配成功。以上过程结果为“S OUT FIR EER SS”。

条件“[AB]”的完整解释是：A或者B，或者AB都不是，即空文本。

如果不用方括号，改用元字符“?”能否实现同等结果呢?

将以上过程中 Pattern 属性的值修改为“A?B?”即可，表示匹配A或者匹配B，两者都是可选条件。

```
Sub 可选匹配2()  '替换掉所有A和B
```

```
  Dim Mystr As String                                '声明变量
  Mystr = "ABS ABOUT FAIR BEER BASS"                 '待搜索的文本
  With CreateObject("VBSCRIPT.REGEXP")               '创建正则表达式引用
    .Pattern = "A?B?"                                '匹配 A 或者 B，无先后顺序
    .Global = True                                   '全局匹配
  '如果匹配成功，将找到的所有对象替换成空文本
   If .Test(Mystr) Then MsgBox .Replace(Mystr, "")
  End With
End Sub
```

那是否表明“[]”和“?”的功能完全一致呢？其实不然，可以通过以下改进突显两者的差异：

```
Sub 可选匹配 3() '替换掉所有 AB,AB 是一个整体
  Dim Mystr As String                                '声明变量
  Mystr = "ABS ABOUT FAIR BEER BASS"                 '待搜索的文本
  With CreateObject("VBSCRIPT.REGEXP")               '创建正则表达式引用
    .Pattern = "(AB)?"                               '匹配 AB，但 AB 组合是可选的
    .Global = True                                   '全局匹配
    '如果匹配成功，将找到的所有对象替换成空文本
    If .Test(Mystr) Then MsgBox .Replace(Mystr, "")
  End With
End Sub
```

执行以上过程，结果为“S OUT FAIR BEER BASS”，这表明条件 A 和 B 是一个整体，它对于单独的 A 和 B 无法匹配成功，而且有顺序限制。元字符“?”可以使一个字符组成为可选的条件，而元字符“[]”无法实现。

或许以上 3 段代码仍不够直观，再看一看以下过程：

```
Sub 可选匹配 4() '替换掉“中人”和“中国文人”
  Dim Mystr As String                                '声明变量
  Mystr = "中国人中间人中国文人中介人中人"           '待搜索的文本
  With CreateObject("VBSCRIPT.REGEXP")               '创建正则表达式引用
    .Pattern = "中(国文)?人"                         '本条件可以匹配“中人”、“中国文人”
    '.Pattern = "中[国文]人"                         '本条件可以匹配“中国人”、“中文人”
    .Global = True                                   '全局匹配
    '如果匹配成功，将找到的所有对象替换成空文本
    If .Test(Mystr) Then MsgBox .Replace(Mystr, "")
  End With
End Sub
```

以上过程中分别采用“[]”和“()?”指定了两个条件，分别执行两个条件并比较结果，你就会明白其中的奥秘。

本过程中的小括号“()”也是正则表达式中的元字符，用于将多个字符转换成一个整体。

“[]”还有一个特殊的作用，用于确定字符区间。例如 0～9 除了用“[0123456789]”表示外，也可以用“[0-9]”这种简化形式表示；而 26 个字母也可以写作“[a-zA-Z]”，这个特性在处理数字与字母时极其有用，在后面有大量的相关案例。

元字符“|”的用法与“[]”和“?”大同小异，只要利用“|”将多个条件隔开即可。例如“A|B|C”表示 3 个条件中符合任意一个皆可。当然也可以是字符串，例如“Is|It's|It is|It”，表示“Is”、“It's”和“It is”、“It”4 个条件任选其一，它们是并列关系。

通过案例或许能更快了解可选匹配的用处，字符串 ""WIN98:1555 WIN2000:2860 VISTA:1880 WIN95:1500 WIN7:2000 WIN2003:2800 WIN2008:3200" 中包括多个系统的名称和价格，现需要提取其中属于服务器版本的系统名称与价格，即 WIN2000、WIN2003 和 WIN2008。代码如下：

```
Sub 可选匹配5()'代码存放位置：模块中
  Dim Mystr As String, ResultStr As String, Item '声明变量
  Mystr = "WIN98:1555 WIN2000:2860 VISTA:1880 WIN95:1500 WIN7:2000 WIN2003:2800
WIN2008:3200"  '待搜索的文本
  With CreateObject("VBSCRIPT.REGEXP")              '创建正则表达式引用
    '匹配 WIN2000 和 WIN2003、2008,后面的“[0-9]+”表示至少一位数字
    .Pattern = "WIN200(0|3|8):[0-9]+"
    .Global = True                                  '全局匹配
    For Each Item In .Execute(Mystr)                '遍历搜索结果
      ResultStr = ResultStr & Chr(10) & Item        '将所有符合条件的目标串连起来
    Next
    MsgBox ResultStr                                '报告结果
  End With
End Sub
```

条件“WIN200(0|3|8)”表示“WIN2000”、“WIN2003”和“WIN2008”，当然，你也可以采用“(WIN2000|WIN2003|WIN2008)”这种写法，其含义相同。

本例文件参见光盘：..\第十四章\14-6 认识可选匹配.xlsm

2. 排除型匹配

在工作中有时也会使用排除型条件搜索对象。例如要在 10 个人中选定 8 个人，那么你一定是采用以“某两人以外的人”为条件，而不会逐一罗列 8 个符合条件的姓名，这其实就是排除法。在正则表达式中，使用元字符“^”配合“[]”来排除条件。

例如要获取一句英语中的所有元音字母，那么可以将非元音字母替换掉，剩下的字符即为元音字母，匹搜索条件应使用“[^a^i^o^e^u]”，完整代码如下：

```
Sub 获取元音字母()'代码存放位置：模块中
  Dim Mystr As String, ResultStr As String, Item '声明变量
  Mystr = "I Love English"                      '指定待搜索的字符串
  With CreateObject("VBSCRIPT.REGEXP")          '创建正则表达式引用
    .Pattern = "[^a^i^o^e^u]"                   '匹配五个元音以外的字母
    .IgnoreCase = True                          '不区分大小写
    .Global = True                              '全局匹配
    MsgBox .Replace(Mystr, "")                  '将符合条件的字母替换成空白，留下元音字母
  End With
End Sub
```

在以上代码中“[^a^i^o^e^u]”条件表示排除 5 个元音字母，过程的执行结果是“loeEi”。

当然，也可以通过使用小括号进一步简化代码，即采用“[^(aioeu)]”为条件，仅仅需要使用一次“^”即可。VBA 不会将“(aioeu)”看作为一个整体，它们之间仍然是并列关系，因为它们处于元字符“[]”之间。

知识补充：元字符“^”必须配合“[]”时才具有排除条件的功能，如果在“[]”之外，它有另外的含义，在后面的内容将会提到。

3. 限量匹配

限量匹配是指对条件指定数量。例如字母 A 重复 3 次时才符合条件，或者数字 12 重复 0～10 次皆符合条件。在正则表达式中关于限制重复次数的方法比较多，包括以下 4 种。

- 星号：*
- 加号：+
- 问号：?
- 上下限：{下限,上限}

在以上 4 种方法中，前 3 种元字符称为元字符量词。它们都用于限制某字符或者字符串的出现次数。表 14-1 中为对 3 个元字符进行比较。

表 14-1　元字符量词比较

元字符	匹配下限	匹配上限	备注
?	无	1	单次可选
*	无	无	任意次数均可
+	1	无	至少一次

从表 14-1 中可以看出，元字符“?”表示字符重复 0～1 次，也就是条件是可选的。元字符“*”表示字符重复任意次，无上下限。由于“*”具有的这种特性，使得任意情况下它都可以匹配成功，所以它总是和其他符号套用，否则单独使用没有任何意义。而“+”和“*”的功能大致相同，只不过“+”有下限，最少重复一次，否则不符合条件。通过以下案例可以比较三者之间的异同。

以下案例用于在字符串“黄黄张明明黄松林”中查找“黄”，并分别以“黄*”、“黄+”和“黄?”作为条件，然后比较三者的结果。

```
Sub 元字符量词的区别()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Dim Mystr As String, ResultStr As String, Item
  Mystr = "黄黄张明明黄松林"
  With CreateObject("VBSCRIPT.REGEXP")
  '.Pattern = "黄*"
    '.Pattern = "黄+"
    '.Pattern = "黄?"
    .Global = True
    For Each Item In .Execute(Mystr)
      ResultStr = ResultStr & Chr(10) & Item
    Next
    MsgBox ResultStr
  End With
End Sub
```

由于代码中 3 个 Pattern 参数的赋值语句都已转换成注释，读者需要分别取消其中一句注释，并分 3 次执行。可以看到 3 次执行的结果如图 14.6 至图 14.8 所示。

采用“黄*”作为匹配条件时表示“黄”字出现任意次（包含 0 次，即空文本），所以能区配成功 10 次，不过其中 8 次是空文本；采用“黄+”作为匹配条件时，由于“+”限制至少有一个匹配，所以字符串“黄黄张明明黄松林”中只有两个符合条件，其中“黄黄”由于并列出现，只算一次匹配；而“黄?”的匹配条件是 0 个或者 1 个“黄”，因此匹配成功次数也是 10 次，只不过有 7 次的匹配结果是空文本。

图 14.6　使用“*”匹配结果

图 14.7　使用“+”匹配结果

图 14.8　使用“？”匹配结果

通过以上案例可以区分三者的差异，但仅从此案例中似乎看不到它们在实际工作中的用途，其实三者极其有用，仅仅因为本案例所举的字符过于随意罢了。在后面的案例中，会有这 3 个元字符的大量应用，从而见证它们的强大功能。

本例文件参见光盘：..\第十四章\ 14-7 限量匹配.xlsm

“{下限,上限}”的形式也可以指定字符的出现次数。例如：

“G{1,5}”——表示字母 G 连续出现 1～5 次；

“G{5}”——表示字母 G 连续出现 5 次；

“G{2,}”——表示字母 G 连续出现至少 2 次，没有上限；

“G{,5}”——这种写法无法匹配成功；

“G{0,5}”——表示字母 G 连续出现 0～5 次。

实际工作中，“{下限,上限}”的形式应用相当普遍。

```
Sub 取出大于等于10000的数字()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Dim Mystr As String, ResultStr As String, Item
  Mystr = "一月5690 二月5533 三月67890 四月12345 五月4999 六月18890..9"
  With CreateObject("VBSCRIPT.REGEXP")
    .Pattern = "[1-9][0-9.]{4,}"
    .Global = True
    For Each Item In .Execute(Mystr)
      ResultStr = ResultStr & Chr(10) & Item
    Next
  End With
  MsgBox "取出大于等于10000的数值:" & ResultStr
End Sub
```

以上过程用于取出字符串中所有大于等于 10000 的数值，条件“[1-9][0-9.]{4,}”的含义如下。

首先，条件“[1-9]”限制了第一个搜索字符为 1～9，因为如果等于 0 则会小于 10000。

其次，“[0-9.]”表示数字 0～9，以及小数点。这 11 个字符中任取其一皆可。

最后，采用“{4,}”限制前面的条件“[0-9.]”，表示“[0-9.]”最少出现 4 次，上不封顶。

3 个条件联合起来获取的数据即为大于等于 10000 的数值，结果如图 14.9 所示。

当然，细心的读者可能会认为“[0-9.]{4,}”会导致“六月 18890...4”中的“18890...4”也匹配成功，实际上它已不符合“数值”这个条件，无法参与数学运算。

不错，编写正则表达式的匹配条件时，总是需要在“全面”与“实用”两者之间做出选择：要

么代码精简、速度快捷，但是代码可能只能在当前数据中使用，数据不规范时就会遗漏或者出错；要么代码完善，适用于所有数据，但是代码复杂、难懂，或者执行效率低。采用哪一种方式，完全由用户自行决定。如果你认为只要完美解决当前数据即可，不需要考虑例外的项目，那么可以采用实用型的思路，不需要尽善尽美。当然，如果代码是自己使用，可以确保数据都很规范，那是无可厚非的，如果开发的是商业插件，给其他用户使用，则建议尽量完善才好，当效率和正确性发生冲突时，宁愿放弃效率。

图 14.9　取大于等于 10000 的数值

对于本例，如果希望代码完善，以便获取的数值可以参与后期的运算，那么可以按以下方式修改代码：

```
Sub 取出大于等于10000的数字2()'①代码存放位置：模块中②随书光盘中有每一句代码含义注释
  Dim Mystr As String, ResultStr As String, Item
  Mystr = "一月 5690 二月 5533 三月 67890 四月 12345 五月 4999 六月 18890....4, 七月
888888.8"
  With CreateObject("VBSCRIPT.REGEXP")
    .Pattern = "[1-9][0-9]{4,}(\.[0-9]*)?"
    .Global = True
    For Each Item In .Execute(Mystr)
      ResultStr = ResultStr & Chr(10) & Item
    Next
  End With
  MsgBox "取出大于等于10000的数值:" & ResultStr
End Sub
```

执行以上代码，后会返回如图 14.10 所示的结果。条件“[1-9][0-9]{4,}(\.[0-9]*)?”将“18890...4”中的后三位排除在外，避免得到无法参与计算的数据。

其中“\.”表示小数点，为什么要采用这种方式表示小数点在后面的章节会有说明。

将改进后的条件“[1-9][0-9]{4,}(\.[0-9]*)?”与“[1-9][0-9.]{4,}”进行比较，可以发现它们的区别在于如何处理小数点。条件“[1-9][0-9.]{4,}”将小数点直接放进“[]”中，然后利用“{4,}”限制数据出现次数不少于 4 次时，产生了副作用，使小数点重复出现也可匹配成功。另一个条件“[1-9][0-9]{4,}(\.[0-9]*)?”将小数点从“[0-9]{4,}”中分离出来，使小数点前面只有不少于 4 位的数字，小数点后面也只有数字，数字的出现次数受元字符“*”限制，表示任意位。最后的“?”符号用于限制小数点和小数点后面的数字，表示它是可选的，有没有皆不重要。如果不使用“?”，那么必须含有小数的数值才会匹配成功。

图 14.10　取大于等于 10000 的值修正结果

为了加深读者对“{下限,上限}”的印象，再举一个例子。将图 14.11 中“(###)########”格式的电话号码替换为“###-#########”格式，如果不用正则表达式很难处理这种数据，它的规律性不够强，需要加入很多判断条件，并多次循环才能实现。然而对于正则表达式的超强能力来说，这只是小试牛刀。

```
Sub 取出电话号码并转换格式()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Dim Str As String, Rng As Range, Item
  For Each Rng In Range("a1:a4")
    Str = Rng.Text
```

```
    With CreateObject("VBSCRIPT.REGEXP")
      .Global = True
      .Pattern = "\(([0-9]{3})\)([0-9]{8})"
      For Each Item In .Execute(Str)
        Str = Replace(Str, Item, Replace(Replace(Item, "(", ""), ")", "-"))
      Next
      Rng = Str
    End With
  Next Rng
End Sub
```

以上代码执行结果如图 14.12 所示。

	A
1	厂长：(办公室一)(202)82514769(办公室二)(020)82515869
2	经理：(办公室一)(769)83457924(办公室二)(769)88734567
3	主任：(办公室一)(020)86341122(手机)15912345678
4	董事：（760）22675432

图 14.11　电话簿

	A
1	厂长：(办公室一)202-82514769(办公室二)020-82515869
2	经理：(办公室一)769-83457924(办公室二)769-88734567
3	主任：(办公室一)020-86341122(手机)15912345678
4	董事：760-22675432

图 14.12　替换结果

代码中的条件“"\(([0-9]{3})\)([0-9]{8})"”可以分 3 段理解：

[0-9]{3}：表示连续出现的 3 位数值。

[0-9]{8}：表示连续出现的 8 位数值。

“\(”和“\)”：分别表示“（”和“）”，元字符“\”的功能是将元字符转换成文本，换而言之就是消除特殊符号的特殊功能，还原为文本本身。

以上 3 个条件组合后，表示在字符串中搜索以“（”开头，接着是 3 位数字，再接着是“）”，最后是 8 位数字的字符串。“（）”虽然多次出现，但表达式的取值规则以“（）”配合数字个数，从而避过了“（办公室一）”，不会错将其他小括号也替换成“-”。

本例文件参见光盘：..\第十四章\14-8 转换电话簿格式

4. 位置匹配

位置匹配包括行首与行尾，及单词界限。VBA 提供 4 个方法对应行首、行尾及单词界限与非界限，它们包括：

- 匹配行首——^
- 匹配行尾——$
- 匹配单词界限——\b
- 匹配非单词界限——\B

位置匹配分文本的行首与行尾，以及单词的分界线。

确定起始位置有时对于搜索相当有用，例如某个姓名多次出现时仅取位于该语句第一位的姓名，或者提取独立成为一个单词的字母组合（例如 an），而不是包含在一个单词中的部分字母（例如 banana）。

在正则表达式中，文本的行首与行尾分别用“^”和“$”来表示，但前提是位于元字符“[]”之外。如果在“[]”内，则没有匹配位置之功能。

例如：

“^Q”——表示匹配位于行首的字母 Q，如果在中间或者位于行尾则略过，例如“QQQ”；

“[^^]Q[^Q]{1,}”——表示字母不在行首与行尾，但是包含 Q。例如“QA BQDE QEQ”；

“[^^]Q”——表示字母 Q 不在行首，其中两个“^”表示行首，第一个“^”表示排除；

“[^Q]{1,}Q$”——表示不能以 Q 开头，但是以 Q 结束。例如“QABQDEQ”；

“.{2}$”——最后两位字符。其中“.{2}”表示任意两位，“$”表示行尾。例如“ABCD”。

以上添加灰色底纹者表示匹配成功的对象。

匹配行首时，元字符“^”需要写在条件之首，不能放在“[]”中。在下面的案例中，获取以“江”开头，但不能以“江”结尾的字符串：

```
Sub 匹配行首()                                          '代码存放位置：模块中
  Dim Mystr As String                                   '声明变量
  Mystr = "江南大江南北"                                '待搜索的文本
  With CreateObject("VBSCRIPT.REGEXP")                  '创建正则表达式引用
    .Pattern = "^江.+(?=江)"                            '搜索以“江”开头但结尾不包含“江”的字符串
    .Global = False                                     '仅匹配第一个
    MsgBox .Execute(Mystr)(0)                           '报告匹配结果
  End With
End Sub
```

执行以上代码后，结果为“江南大”。因为“江”位于行首，符合条件“^江”；而“大”字之后是“江”，所以取“大”，符合条件“(?=江)”，而中间有“南”字则符合条件“.+”。

关于元字符“.”和“?=”的功能将在稍后的内容中有详述。

在实际工作中，英文单词的界限匹配应用要广得多。例如从一句话或者一篇文章中提取符合条件的单词，可以根据单词与单词间有分隔符这个特点设置正则表达式的条件。

在通常情况下，以空格、段落首行、段落末尾、逗号、句号等符号作为单词的边界，当然，在特殊情况下书写单词会用到“-”，“-”也作为单词的界限。

例如，将一句话中的所有单词列一一罗列出来，可以采用以下过程：

```
Sub 分别取出所有单词()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Dim Mystr As String, ResultStr As String, Item
  Mystr = "abs bus av-aya i love"
  With CreateObject("VBSCRIPT.REGEXP")
    .Pattern = "\b[a-zA-Z]+\b"
    .Global = True
    If .test(Mystr) Then
      For Each Item In .Execute(Mystr)
        ResultStr = ResultStr & Chr(10) & Item
      Next
      MsgBox "符合条件的对象有:" & ResultStr
    End If
  End With
End Sub
```

过程中用“b[a-zA-Z]+\b”作为条件，两个“\b”分别表示单词首尾界限，中间的[a-zA-Z]+表示不少于一个字母。如果要求只提取长长度为 3 的单词，那么可以将搜索条件修改为“\b[a-zA-Z]{3}\b”。如图 14.13 所示是所有单词，如图 14.14 所示是所有长度为 3 的单词。

本例文件参见光盘：..\第十四章\ 14-9 位置匹配.xlsm

以下过程中包括 3 个条件，读者可以分别比较三者的差异：

```
Sub 匹配单词界限() '①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Dim Mystr As String, ResultStr As String, Item
```

```
Mystr = "abs bus pass avaya about moon spar ada abandon a334"
With CreateObject("VBSCRIPT.REGEXP")
  '.Pattern = "\ba.*?\b"
  '.Pattern = "\ba[a-z]*a\b"
  '.Pattern = "\ba[^a]{1,}?[b-z]{1}\b"
  .IgnoreCase = True
  .Global = True
  If .test(Mystr) Then
    For Each Item In .Execute(Mystr)
      ResultStr = ResultStr & Chr(10) & Item
    Next
      MsgBox "符合条件的对象有:" & ResultStr
    End If
  End With
End Sub
```

在以上过程中，“\ba.*?\b”条件表示首尾匹配单词的界限，中间以字母“a”开头的字符串，也就是匹配所有以“a”开头的单词。

图 14.13　逐一取单词

图 14.14　提取三个字母组成的单词

“\ba[a-z]*a\b”条件表示匹配首尾皆为字母“a”的单词。

“\ba[^a]{1,}?[b-z]{1}\b"”条件表示匹配以“a”开头，其他任何字符不等于“a”的单词。条件中的“[^a]{1,}?”表示不等于“a”字符至少一个，条件最后的“[b-z]{1}”表示单词的末尾是“a”以外的字母。为什么不用“[^a]”呢？因为它仅仅排除字母了“a”，却不能排除汉字或者数字。

5. 任意字符匹配

在 DOS 系统中，“*”可以匹配任意字符，并且数量不限；“?”可以匹配一个任意字符。而在正则表达式中，“*”和“?”都具有了不同的含义。差别在于当“*”和“?”不在最前面时，它用于指定前一个字符的出现次数，即“*”代表次数，而不是代表字符。例如“大*”表示“大”字出现任意次，而在 DOS 系统中，“大*”表示以“大”开头，任意字符结尾的字符串。“大?”在正则表达式中表示“大”出现次数为 0 或者 1 次，即长度为 0 或者 1；在 DOS 中表示“大”字后跟随一个任意字符，长度为两位。

在正则表达式中，也有类似于 DOS 系统中通配符的符号，即元字符“.”。它代表任意单个字符，即“大.”表示“大”字后跟随一个任意字符，长度为两位。如果配合另一个元字符“+”，或者“?”可以产生很多变化。

“大.+”——表示第一位为“大”，后接长度至少为 1 的任意字符；

“大.?”——表示第一位为“大”，后接长度为 0 或者 1 的任意字符，即“大”和“大小”两者都可以匹配成功；

“.?”——表示 0 位或者 1 位任意字符，那么不管什么字符都可以匹配成功；

“.*”——表示匹配任意位字符，不管是什么字符都可以匹配成功。

“.?”和“.*”都是可以匹配任意字符，但两者差异很大。在后面介绍贪婪匹配和惰性匹配时将会进行详细比较。

在以下实例中，“紫竹一期13 光大二期2014 紫竹二期2013 宏远四期2014 松山紫竹2014”表示建筑公司的工程记录。如果需要单独取出关于“紫竹”工程的信息，那么可以使用正则表达式中的元字符“.”作为搜索条件，迅速找出所有匹配对象：

```
Sub 提取所有紫竹工程()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Dim Mystr As String, ResultStr As String, Item
  Mystr = "紫竹一期13 光大二期2014 紫竹二期2013 宏远四期2014 松山紫竹2014" & " "
  With CreateObject("VBSCRIPT.REGEXP")
    '.Pattern = "紫竹(....|......) "
    '.Pattern = "紫竹.{4,6} "
    .Global = True
    If .test(Mystr) Then
      For Each Item In .Execute(Mystr)
        ResultStr = ResultStr & Chr(10) & Item
  Next
      End if
    End With
  MsgBox "符合条件的对象有:" & ResultStr
End Sub
```

执行以上过程后可以得到如图14.15所示的结果。

以上过程中有5点需要注意：

（1）被查找对象为“紫竹一期13 光大二期2014 紫竹二期2013 宏远四期2014 松山紫竹2014”，为了让它更具有规律性，特意在后加一个辅助字符“”，即空格，从而使所有工程名称后面都有一个空格，便于匹配。

（2）过程分别采用“紫竹(....|......) ”和“紫竹.{4,6}? ”为条件，并且已将两者都转换成注释。读者在使用代码时需要分别将其中一句还原为代码，否则只会得到空文本。

（3）两个条件的末尾都各有一个空格，表示匹配对象必须以空格结尾。

（4）两个条件可以得到相同结果。其中“紫竹(....|......) ”表示以“紫竹”开头，后跟4个或者6个字符，每一个元字符“.”代表一个任意字符。

图14.15 搜索结果

（5）如果不使用辅助字符创造条件是否可以完成呢？也就是说各期工程的结尾有所不同。答案是肯定的，元字符“|”提供多项选择功能，可以解决此问题，改用以下条件即可：

“紫竹(....|......)(|$)”——重点在于“(|$)”部分，它表示空格结尾和行尾皆符合条件。

CD-ROM 本例文件参见光盘：..\第十四章\ 14-10 按指定长度匹配任意字符.xlsm

6. 转义字符

“[]”、“.”、“|”、“{}”、“()”、“?”、“^”、“$”等都属于正则表达式的元字符，它们有着特殊的作用。那如果需要使用这些元字符自身时该如何表达呢？例如在字符串中查找“(”或者“?”，使用前面的知识将无法操作成功。

正则表达式中提供一个转义字符“\”用于将元字符还原为普通文本。只要将转义字符置于元字符之前即可，例如“*”表示文本“*”，而不是“\”和“*”。“\”也是一个元字符。

如果要匹配“\”自身呢？可采用“\\”。第一个“\”是转义字符，它可将第二个元字符还原为文本“\”。

例如将“电话：(020)87654321”的区号提取出来，那么可以采用以下过程：

```
Sub 取区号()'代码存放位置：模块中
  Dim Mystr As String                              '声明变量
  Mystr = "电话：(020)87654321"                    '待搜索的文本
  With CreateObject("VBSCRIPT.REGEXP")             '创建正则表达式引用
    .Pattern = "\(.+\)"                            '以括号作为首尾条件
    .Global = False                                '仅匹配第一个
    If .Test(Mystr) Then MsgBox .Execute(Mystr)(0) '报告第一个匹配结果
  End With
End Sub
```

以上过程中的条件“"\(.+\)”可以分3段理解：

（1）“\(”表示第一个字符是“(”，“\”是元字符，不参与匹配，它的功能是还原“(”；

（2）“.+”表示不少于一位的任意字符。本例中也可以采用“...”替代，表示3个字符；

（3）“\)”表示最后一个字符是“)”。

以上代码执行后结果为“(020)”。

以上代码针对于字符串有一个括号时有效，如果字符串是“电话：(020)87654321(769)12345678”则无法按需求提取字符了，它会将“(020)87654321(769)”一并提取出来，而不是只提取“(020)”。如果将搜索条件“"\(.+\)"”修改为“"\(.+?\)"”就可以，这就是贪婪匹配与惰性匹配的区别。

7. 非获取匹配

非获取匹配是正则表达式中的一种新概念，为搜索工作提供了一个全新的搜索模式，它可使某些字符参与搜索，但不参与匹配。

非获取匹配本身包括正向预查、负正向预查、反向预查和负反向预查。然而Excel 2010 VBA仅仅支持正向预查和负正向预查。本书讲述正向预查和负正向预查的语法与案例。

先看一个案例——要求将图14.16中成绩为100分的科目标示为红色。

	A
1	语文68高等数学100政治89化学实验76英语44物理100体育70音乐89几何100
2	

图14.16 成绩表

根据前面的知识可以得到以下代码：

```
Sub 提取成绩100的科目()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Dim Mystr As String, ResultStr As String, Item
  With CreateObject("VBSCRIPT.REGEXP")
    .Pattern = "([^0-9])+100"
    .Global = True
    If .test(Range("a1")) Then
      For Each Item In .Execute(Range("a1"))
        Range("a1").Characters(Start:=InStr(Range("a1"),    Item),    Length:=
Len(Item)).Font.ColorIndex = 3
      Next
    End If
  End With
End Sub
```

以上代码中的搜索条件为“([^0-9])+100”，表示以非数字任意位数开始、以数字 100 结尾，其搜索结果如图 14.17 所示。

	A
1	语文68高等数学100政治89化学实验76英语44物理100体育70音乐89几何100
2	

图 14.17　红色标示 100 分成绩

如图 14.17 所示的结果显然不符合要求，此代码虽然正确地找出了所有成绩为 100 分对应的科目，然而在标示颜色时将 100 本身也当作了目标。为了杜绝这种“意外”，必须采用正则表达式中的正向预查，即搜索时将 100 作为搜索条件，但实际取值时将它排除在外。

正向预查的格式为“(?=条件)”。本例中要求 100 可以参与搜索，但不参与颜色标示，那么代码中的条件“([^0-9])+100”修改为“([^0-9])+(?=100)”即可，搜索结果如图 14.18 所示。

	A
1	语文68高等数学100政治89化学实验76英语44物理100体育70音乐89几何100
2	

图 14.18　利用正向预查标示成绩为 100 分的科目

本例文件参见光盘：..\第十四章\14-11 正向预查.xlsm

正向预查还可以实现更强大的功能，例如与其他元字符组合，产生多项可选匹配。以下案例是正向预查的又一应用。

图 14.19 中 B 列是学校门卫对每日迟到学生的记录。记录表是按迟到学生的入校时间顺序记录的，现需要将它们转换成初中学生在前，高中学生在后，并用“|”区分开，方便查看。

	A	B
1	时间	迟到人员
2	4月3日	初一王宏伟 高二李宽 初三张兰 初三彭大年 高一吴伟
3	4月4日	高一吴文政 初三张霞 高三李文高 初二高民高
4	4月5日	高二张光明 高三欧阳文文 初三李孟华

图 14.19　迟到人员统计表

观察图 14.19 可以发现数据的规律——每一个学生的记录是以“初”或者“高”开始，以空格结束，末尾者例外。根据这个规律可以采用以下代码实现：

```
Sub 整理出勤表()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Dim ResultStr As String, Item, rng As Range, Mystr As String
  For Each rng In Range("B2:B4")
    Mystr = " " & rng.Text & " "
    With CreateObject("VBSCRIPT.REGEXP")
      .Pattern = "(^| )初.*?(?=( |$))"
      .Global = True
    If .Test(Mystr) Then
      For Each Item In .Execute(Mystr)
        ResultStr = ResultStr & Item
      Next
    End If
    '前面处理好了初中的学员信息，现在开始处理高中的学员信息
    .Pattern = "(^| )高.*?(?=( |$))"
    .Global = True
    If .Test(Mystr) Then
      ResultStr = ResultStr & " | "
      For Each Item In .Execute(Mystr)
```

```
        ResultStr = ResultStr & Item
      Next
    End If
  End With
  rng = ResultStr
  ResultStr = ""
 Next rng
End Sub
```

执行过程后结果如图 14.20 所示。

	A	B
1	时间	迟到人员
2	4月3日	初一王宏伟 初三张兰 初三彭大年 | 高二李宽 高一吴伟
3	4月4日	初三张霞 初二高民高 | 高一吴文政 高三李文高
4	4月5日	初三李孟华 | 高二张光明 高三欧阳文文

图 14.20 按初中、高中排序

本例中使用“(^|)初.*?(?=(|$))”为条件，可以分 3 段来理解：

（1）“(^|)初”：表示以行首或者空格作为第一条件，然后是字符“初”。

（2）“.*?”：代表任意长度的任意字符，采用的是惰性匹配。

（3）“(?=(|$))”：表示最后一个搜索条件为空格或者行尾，它参与搜索，但不参与取值。

本例中“高”字作为搜索条件的第一位，却并不总出现在第一位，例如“高三李文高”。对于这些特例，需要适量添加搜索条件，避免取值时错位。

本例文件参见光盘：..\第十四章\14-12 正向预查应用 2

负正向预查表示正向预查的相反效果，它表示将某个条件以外的字符参与搜索，但实际取值时将它排除在外，其格式为“(?!条件)”。

例如“A(?!B)”，表示搜索条件是以“A”开头，以“B”以外的字符结尾，而“B”以外的字符仅参与搜索，不参与取值。以下是关于负正向匹配的应用案例。

图 14.21 中用“盈”、“亏”、“平”来表示财务收支状况，现要求将其中未亏损的月份用红色标示出来，但不能将“盈”、“亏”、“平”3 个字本身进行标示。

根据图 14.21 中字符的规律，可用以下代码实现：

```
Sub 红色标注不亏的月份()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Dim Item
  With CreateObject("VBSCRIPT.REGEXP")
    .Pattern = "([一二三四五六七八九]|[十][一二]?)月(?!亏)"
    .Global = True
    If .Test(Range("a1")) Then
      For Each Item In .Execute(Range("a1"))
        Range("a1").Characters(Start:=InStr(Range("a1"), Item), Length:=
Len(Item)).Font.ColorIndex = 3
      Next
    End If
  End With
End Sub
```

执行以上过程后，结果如图 14.22 所示。

	A
1	一月盈二月亏三月盈四月盈五月亏六月平七月平八月亏九月平十月亏十一月盈十二月盈

图 14.21 收支盈亏统计表

	A
1	一月盈二月亏三月盈四月盈五月亏六月平七月平八月亏九月平十月亏十一月盈十二月盈

图 14.22 标示未亏损的月份

以上过程采用“([一二三四五六七八九]|[十][一二]?)月(?!亏)”作为条件，可以分3段理解：

（1）“[一二三四五六七八九]”——用于匹配一到九月。因为只有一到九月是单字符；

（2）“[十][一二]?”——表示十或者十一、十二月。后面的“?”让“一”和“二”变成可选匹配；

（3）“(?!亏)”——表示亏字以外的字符作为最后一位搜索条件，但是它不参与匹配。

本例文件参见光盘：..\第十四章\14-13 负正向预查.xlsm

8. 贪婪匹配与惰性匹配

贪婪匹配与惰性匹配其实在前面已经通过案例展示过。

贪婪匹配表示尽可能匹配多的字符。例如2个、3个、4个字符皆符合条件时，它会匹配4个字符。

惰性匹配与贪婪匹配刚好相反，它匹配尽可能少的字符。例如2个、3个、4个字符皆符合条件时，它会匹配2个字符。

从字面理解，两者岂非背道而驰？事实上正因为存在这种区别，才给工作带来了极大的便利，在不同需求下可以选择不同的匹配方式。

例如从“电话一(020)87654321、电话二(0769)88654328”字符串中提取区号，只需要匹配括号即可。先看一看以下过程的匹配结果：

```
Sub 取区号()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Dim Mystr As String
  Mystr = "电话一(020)87654321、电话二(0769)88654328"
  With CreateObject("VBSCRIPT.REGEXP")
    .Pattern = "\(.+\)"
    .Global = False
    If .Test(Mystr) Then MsgBox .Execute(Mystr)(0)
  End With
End Sub
```

在以上过程中“\(.+\)”表示获取首尾分别为“(”和“)”的字符串。在“电话一(020)87654321、电话二(0769)88654328”中，“(020)”符合条件，“(020)87654321、电话二(0769)”也符合条件，而贪婪匹配的规则是匹配尽可能多的字符，所以它取后者，结果如图14.23所示。

再看下面的代码：

```
Sub 取区号2()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Dim Mystr As String
  Mystr = "电话一(020)87654321、电话二(0769)88654328"
  With CreateObject("VBSCRIPT.REGEXP")
    .Pattern = "\(.+?\)"
    .Global = True
    If .Test(Mystr) Then MsgBox .Execute(Mystr)(0)
'    如果执行以下代码可以得到所有区号
'    Dim Item, ResultStr As String
'    For Each Item In .Execute(Mystr)
'      ResultStr = ResultStr & Chr(10) & Item   ' 将所有符合条件的对象串连起来
'    Next
'    MsgBox "区号为：" & ResultStr
  End With
End Sub
```

执行以上代码的结果为“(020)”，这就是惰性匹配的特点——当多个字符串皆符合要求时，仅取最少的字符作为最终结果。

如果将代码中被注释掉的几句代码解除注释，那么可以分别提取所有符合条件的区号，效果如图 14.24 所示。

图 14.23　贪婪匹配

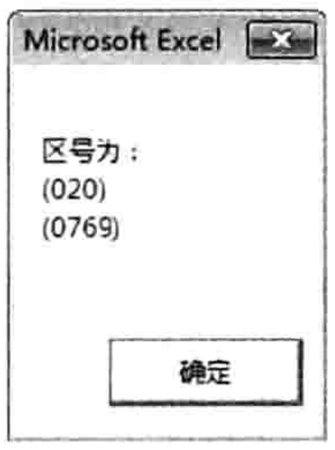


图 14.24　惰性匹配

在以上两个案例中可以看出贪婪匹配和惰性匹配的代码编写差异和匹配结果差异。

表 14-2 中罗列了关于贪婪匹配与惰性匹配的使用方法与差异。

表 14-2　贪婪匹配与惰性匹配

贪婪匹配	惰性匹配	匹配描述
?	??	匹配 0 个或 1 个
+	+?	匹配 1 个或多个
*	*?	匹配 0 个或多个
{n}	{n}?	匹配 n 个
{n,m}	{n,m}?	匹配 n 个或 m 个
{n,}	{n,}?	匹配 n 个或多个

再看一个案例——图 14.25 中 B 列的人个资料显示在一个单元格中不利于查看，要求利用 VBA 将这些资料分列，显示在 4 列中。

	A	B
1	姓名	个人资料
2	赵国	出生1990-5-3小学入学1996-9-1参加工作2010-11-2结婚2011-5-14
3	刘昂扬	出生1985-7-7小学入学1982-9-1参加工作2005-10-28结婚2010-7-9
4	赵云秀	出生1992-5-19小学入学1994-9-1参加工作2012-11-28结婚2014-6-7
5	李范文	出生1988-4-23小学入学1985-9-1参加工作2007-10-1结婚2010-4-12

图 14.25　个人资料

根据前面所介绍的知识，解决本例问题应用以下代码：

```
Sub 分列()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Dim I As Byte, Mystr As String, arr(0, 3), rng As Range
With CreateObject("VBSCRIPT.REGEXP")
  For Each rng In Range("b2:b5")
    Mystr = rng
    .Pattern = "[^0-9].+?[0-9]{4}(\-[0-9]{1,2}){2}"
    .Global = True
    For I = 0 To .Execute(Mystr).Count - 1
      arr(0, I) = .Execute(Mystr)(I)
    Next
    rng.Resize(1, 4) = arr
  Next rng
End With
Range("a1").CurrentRegion.EntireColumn.AutoFit
```

```
  Range("a1").CurrentRegion.Borders.LineStyle = 1
End Sub
```

执行以上过程的结果如图 14.26 所示。

	A	B	C	D	E
1	姓名	个人资料			
2	赵国	出生1990-5-3	小学入学1996-9-1	参加工作2010-11-2	结婚2011-5-14
3	刘昂扬	出生1985-7-7	小学入学1982-9-1	参加工作2005-10-28	结婚2010-7-9
4	赵云秀	出生1992-5-19	小学入学1994-9-1	参加工作2012-11-28	结婚2014-6-7
5	李范文	出生1988-4-23	小学入学1985-9-1	参加工作2007-10-1	结婚2010-4-12

图 14.26　分列结果

```
"[^0-9\-].+?[0-9]{4}(\-[0-9]{1,2}){2}"
```

在以上过程中，搜索条件为“[^0-9].+?[0-9]{4}(\-[0-9]{1,2}){2}”，其中“[^0-9].+?”表示以不少于 1 位的数字以外的字符，采用惰性匹配。如果删除问号则是贪婪匹配，无法分列。

“[0-9]{4}”则表示 4 位数字，匹配字符串中的年份。

“(\-[0-9]{1,2}){2}”则表示以“-”开头、以 1～2 位数字结尾，并且重复出现两次，此条件刚好匹配月和日。

值得注意的是惰性匹配是由字符“+”、“*”搭配“?”使用而产生的，而“{2}”或者“..”这类固定字符长度的元字符在任意情况下都无法产生惰性匹配。例如在“ABCDEFG”中搜索“[A-Z]+”为贪婪匹配，可以匹配“ABCDEFG”；条件为“[A-Z]+?”时表示惰性匹配，它只能匹配一个字母“A”；而“[A-Z]{3}”和“[A-Z]{3}?”都匹配同样的字符串“ABC”，在此元字符“?”对“{3}”无效。

本例文件参见光盘：..\第十四章\14-14 贪婪匹配与惰性匹配

9. 一些特殊的元字符

当条件复杂时，正则表达式的匹配条件可能会很长，不利于理解和记忆，为此正则表达式提供一种简单的元字符替代过长的字符组合。

（1）匹配所有单个数字：\d

也就是说“[0-9]”可以简写为“\d”。有了这个组合，前面的很多案例代码都可以改写，例如“[^0-9].+?[0-9]{4}(\-[0-9]{2}){2}”可以简化成“[^\d].+?\d{4}(\-\d{2}){2}”，读者可以将此组合替换后再进行测试。

（2）匹配单个数字以外的字符：\D

也就是说“[^0-9]”可以简写为“\D”，表示匹配数字以外的单字符。

（3）匹配单个字母或数字或下画线:\w

即“\w”组合相当于“[a-zA-Z_0-9]”，例如电子邮箱地址就采用了数字、字母和下画线，所以从字符串获取电子邮箱地址时通常采用“\w”组合。

（4）匹配任意非单个字母，非单个数字或下画线:\W

（5）匹配空白字符，包括空格、制表符、换页符：\s

其相当于“[\f\n\r\t\v]”组合的功能。

（6）匹配非空白字符，包括空格、制表符、换页符以外的字符：\S

10. 一些常见的组合

^\d{5}：区配 5 个数值数字，例如美国邮政编码。

^[+-]?\d+(\.\d+)?$：匹配任意有可选符号的实数。

^[+-]?\d*\.?\d*$：匹配任意有可选符号的实数，但同时也匹配空字符串。

^(20|21|22|23|[01]\d):[0-5]\d$：匹配 24 小时制时间值，可用它判断时间值格式是否规范。

/*.**/：匹配 C 语言风格的注释“/* ... */”。
[\u4e00-\u9fa5]：匹配所有单个中文字符。
\n\s*\r：匹配空行。
<(\S*?)[^>]*>.*?</\1>|<.*? />：匹配 HTML 网页标记。
\w+([-+.]\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*：匹配电子邮箱地址。
[a-zA-z]+://[^\s]*：匹配网址 URL。
d{3}-\d{8}|\d{4}-\d{7}：匹配国内电话号码。
[1-9][0-9]{4,}：匹配腾讯 QQ 号。
^(\d{15}|(\d{17}(\d|X)))$：匹配身份证。
\d+\.\d+\.\d+\.\d+：匹配 IP 地址。

^[A-Za-z]+$：匹配整行皆为字母的字符串，而“[A-Za-z]+”则用于匹配一行中的第一次出现的字母组成的字符串，该行允许有其他字符。

通过以上组合，可以实现很多工作中常用的搜索。当然，你也可以将这些字符任意组合，产生更多变化无穷的匹配条件。

14.3 正则表达式应用

14.3.1 乱序字符串取值并汇总

案例要求：图 14.27 中 B 列为采购记录，由于前期录入数据不规范，金额无法汇总。现要求用 VBA 从中取出所有金额并汇总。

	A	B
1	日期	采购记录
2	2014/7/1	白菜25.5元西红柿18元猪肉36.0元
3	2014/7/2	食盐55元包菜22元鸡蛋18无
4	2017/7/3	土豆14白菜8元牛肉(凉拌)33元
5	2014/7/4	B级火锅调料33元鱼55元
6	合计	

图 14.27 采购记录表

过程代码：

```
Sub 金额汇总()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Dim Mystr As String, Item, Result As Double, rng As Range
  With CreateObject("VBSCRIPT.REGEXP")
    For Each rng In Range("B2:B5")
      Mystr = rng.Text
      .Pattern = "[0-9]+(\.[0-9]*)?"
      .Global = True
      If .test(Mystr) Then
        For Each Item In .Execute(Mystr)
          Result = Result + Item
        Next
      End If
    Next rng
  End With
  Range("b6").Value = Result
End Sub
```

思路分析：

本例中使用了两个循环，里层循环用于获取每个单元格中的所有数值，并逐个汇总。外层循环

用于遍历 B2:B5 区域中的每个单元格。

真正的重点在于正则表达式的 Pattern 参数，本例中采用“[0-9]+(\.[0-9]*)?”为搜索条件，可以分 3 段理解其含义。

（1）“[0-9]+”表示至少一位数字，元字符“+”的作用是强调数字的重复次数。

（2）“\.[0-9]*”表示小数点后面跟任意位数字。元字符“*”表示小数点和数字的位数不限；可以是 0 位、1 位，也可以是 100 位。

（3）最后的“?”用于将表达式转换为惰性匹配，促使正则表达式在搜索时可以将多段数字一起提取出来，否则只能提取第一个值。

由于“[0-9]”等同于“\d”，因此搜索条件也可以使用“\d+(\.\d*)?”。

本例文件参见光盘：..\第十四章\14-15 汇总采购金额.xlsm

14.3.2 计算建筑面积

案例要求：计算字符串“长 4.2 宽 3.8[2 号大厅]长 4.5 宽 1.2[阳台]长 3.8 宽 1[左侧楼梯]”中所指定的区域的建筑面积。注意“[]”中的数字不参与计算。

过程代码：

```
Sub 计算面积1()  '①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Dim ResultStr As String, Item
  With CreateObject("VBSCRIPT.REGEXP")
    .Pattern = "长.*?(?=\[)"
    .Global = True
    For Each Item In .Execute("长4.2宽3.8[2号大厅]长4.5宽1.2[阳台]长3.8宽1[左
侧楼梯]")
      ResultStr = ResultStr & Item
    Next
    MsgBox Evaluate(Replace(Replace(ResultStr, "长", "+"), "宽", "*"))
  End With
End Sub
```

执行以上过程后计算结果为 25.16。

思路分析：

观察字符串可以发现字符分为 3 段，每一段的规律为以“长”开头，以“]”结尾。由于“[]”中间的字符需要忽略，那么在搜索时以“[”结尾即可，但需要确保取值时忽略“[”本身。所以本例过程采用“长.*?(?=\[]”作为匹配条件。可以分 3 段理解这个匹配条件。

（1）“长”表示搜索条件以“长”开头。

（2）“.*?”表示任意长度的任意字符，采用惰性匹配。

（3）“(?=\[)”表示以符号“[”结尾，但符号本身不参与匹配。由于“[”是元字符，因此要在它前面添加转义符“\”。

本例中正则表达式匹配的结果为“长 4.2 宽 3.8 长 4.5 宽 1.2 长 3.8 宽 1”，由于要计算面积，需要将数字两两相乘，所以采用 Replace 函数将“长”替换成运算符“+”，将“宽”替换成运算符“*”，替换结果为“+4.2*3.8+4.5*1.2+3.8*1”。最后再用 Evaluate 方法将它转换成值即可。

需要注意的是，此处的 Replace 是 VBA 内置的函数，而不是正则表达式中的 Replace 方法，所以不需要前面加“.”。

当然，鉴于本例的特殊性，即字符顺序比较规范，也可以将条件简化。

```
Sub 计算面积 2()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  With CreateObject("VBSCRIPT.REGEXP")
    .Pattern = "\[.+?\]"
    .Global = True
    MsgBox Evaluate(Replace(Replace(.Replace("长 4.2 宽 3.8[2 号大厅]长 4.5 宽
1.2[阳台]长 3.8 宽 1[左侧楼梯]", ""), "长", "+"), "宽", "*"))
  End With
End Sub
```

本过程和前一个过程的思路刚刚相反。前一个过程是取出长与宽的记录，然后将字符串替换成可以计算的表达式，本过程以“[”和“]”为搜索的起止条件，然后将匹配成功的字符串替换掉，从而一步实现前一段代码中多步循环的结果。

另外需要注意一个问题，Evaluate 方法虽然可以将表达式转换成值，但它有长度限制，参数超过 256 之后就无法再执行计算，所以本例中待计算的字符不多时，可以采用 Evaluate 方法，如果字符串过长，必须改用以下思路实现：

```
Sub 计算面积 3()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Dim item, Result As Double
  With CreateObject("VBSCRIPT.REGEXP")
    .Pattern = "长.*?(?=\[)"
    .Global = True
     For Each item In .Execute("长 4.2 宽 3.8[2 号大厅]长 4.5 宽 1.2[阳台]长 3.8 宽
1[左侧楼梯]")
       Result = Result + Evaluate(Replace(Replace(item, "长", "+"), "宽", "*"))
     Next
     MsgBox Result
  End With
End Sub
```

本例文件参见光盘：..\第十四章\14-16 计算建筑面积.xlsm

14.3.3　取括号中的数字

案例要求：图 14.28 为某 ERP 系统导出到 Excel 工作表的数据表，现需要计算括号中的产品数量合计。

	A	B
1	产品代号(数量)	合计
2	A43(58)、A32(13)、C1(100)、DE220(500)	
3	A22(77)、D22(130)、C11(500)、DE220(204)	
4	A43(8)、A12(33)、C1(100)	
5	DE43(33)、D33(33)、A05(70)、D12(34)	

图 14.28　由 ERP 系统导出到 Excle 工作表的数据表

过程代码：

```
Sub 汇总括号中的数字()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Dim Item, Result As String, Rng As Range
  With CreateObject("VBSCRIPT.REGEXP")
    For Each Rng In Range("a2:a5")
      .Pattern = "\(.+?(?=\))"
      .Global = True
      For Each Item In .Execute(Rng)
        Result = Result & Item
```

```
      Next
      Rng.Offset(, 1) = Evaluate(Replace(Result, "(", "+"))
    Next Rng
  End With
End Sub
```

执行以上过程后，结果如图 14.29 所示。

	A	B
1	产品代号(数量)	合计
2	A43(58)、A32(13)、C1(100)、DE220(500)	671
3	A22(77)、D22(130)、C11(500)、DE220(204)	1582
4	A43(8)、A12(33)、C1(100)	1723
5	DE43(33)、D33(33)、A05(70)、D12(34)	1893

图 14.29　计算结果

思路分析：

本例中“\(.+?(?=\))"”条件表示以左括号“(”开头、以右括号“)”结尾，中间为任意字符，其中“)”参与搜索，但不参与匹配。

本例也和上一个案例一样，会面临 Evaluate 的参数长度限制问题，读者可以根据前面的思路修改本例代码，使其可以适应 A 列字符任意增减之变化。

本例文件参见光盘：..\第十四章\ 14-17 汇总括号中的数值.xlsm

14.3.4　去除字符串首尾的空白字符

案例要求：去除字符串首尾的空白字符，对中间出现的空白字符则忽略。空白字符包括空格、换行符和制表符。

过程代码：

```
Sub 去除首尾空白符()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Dim MyStr As String
  MyStr = VBA.vbCrLf & " 你好我也好   " & Chr(10)
  With CreateObject("VBSCRIPT.REGEXP")
    .Global = True
    .Pattern = "^\s*|\s*$"
    MsgBox "替换前:" & Chr(10) & "|" & MyStr & "|" & Chr(10) & "替换后: " & Chr(10)
& "|" & .Replace(MyStr, "") & "|"
  End With
End Sub
```

执行以上过程后可以得到如图 14.30 所示的结果。

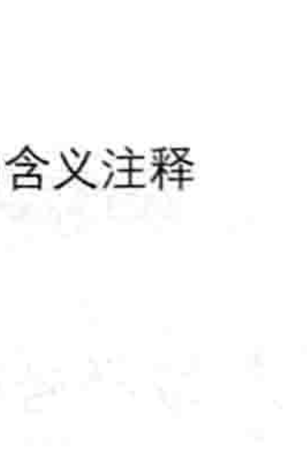


图 14.30　删除首尾空白符

思路分析：

空白符包括空格、换行符、制表符等，通常在通过网页导出数据时容易产生。单元格数据首尾存在空白符时不利于后期计算。例如 Vlookup 函数引用数据时，会查找不到匹配对象。

本例为了让读者清晰地看到替换后的效果，特在字符串前后加上“|”，可以看到空白符号是否保存下来，也可以方便对替换前后的数据进行比较。

本例的搜索条件“^\s*|\s*$”使用了“|”，表示符合两个条件之一就能匹配成功。由于采用了“^”和“$”两个元字符，所以替换时会略过字符串中间出现的空白符号。

本例文件参见光盘：..\第十四章\14-18 去除首尾空白字符.xlsm

14.3.5　将字符串中的多段数字分列

案例要求：图 14.31 中的字符中包含多段数字，因需要执行后期计算，要求将它们分别提取出来，逐一罗列在右边的单元格中。

过程代码：

```
Sub 数字分列()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Dim Item, Result As String, Rng As Range, i As Byte
  With CreateObject("VBSCRIPT.REGEXP")
    For Each Rng In Range("a1:a3")
      Mystr = Rng.Text
      .Pattern = "[0-9]+(\.[0-9]*)?"
      .Global = True
      For Each Item In .Execute(Mystr)
        i = i + 1
        Rng.Offset(0, i) = Item
      Next
      i = 0
    Next Rng
  End With
End Sub
```

执行以上代码后，分列结果如图 14.32 所示。

	A
1	白菜12公斤，大米8公斤
2	猪肉7.9公斤、油10.0公斤
3	长6米\宽4.5米/高5.8米

图 14.31　明细表

	A	B	C	D
1	白菜12公斤，大米8公斤	12	8	
2	猪肉7.9公斤、油10.0公斤	7.9	10	
3	长6米\宽4.5米/高5.8米	6	4.5	5.8

图 14.32　分列结果

思路分析：

本例的搜索条件“[0-9]+(\.[0-9]*)?”表示提取带小数的正整，小数部分为可选项。由于小数点和小数是一个整体，因此必须添加括号，然后再使用问号转换成可选项。

变量 i 是计数器，用于计算 A 列字符串中匹配成功的次数。在对单元格赋值时，用它确定单元格的位置。里层的循环结束时需要将计数器归零，否则只有第一个单元格的分列结果存放正确，后面单元格中的数据会错位。

本例文件参见光盘：..\第十四章\14-19 数字分列.xlsm

14.3.6　获取 E-mail 地址

案例要求：由 ERP 系统中导出个人信息时会包括很多项信息，姓名、个人主页、年龄、E-mail 地址、爱好、身高和体重。现要求从中提取每个人的 E-mail 地址置于右列单元格中。如果不存在 E-mail 地址则显示“信息不完整”，如图 14.33 所示。

	A
1	张文男http://zhannwen.com25岁ZW@163.com游泳1.7米55公斤
2	李高梅女http://hanShen.cn33岁Gaomei.Li@126.cn音乐1.55米43公斤
3	杨子江男44岁棒球69公斤
4	朱仝男http://ShuiHu.cn33岁Shuihu_168@yahoo.com电子游戏59公斤

图 14.33　个人信息表

过程代码：

```
Sub 获取E-mail地址()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Dim rng As Range
  With CreateObject("VBSCRIPT.REGEXP")
    For Each rng In Range("a1:a4")
      .Pattern = "\w+([-+.]\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*"
      .IgnoreCase = True
      If .test(rng.Text) Then
        rng.Offset(0, 1) = .Execute(rng.Text)(0)
      Else
        rng.Offset(0, 1) = "资料不完整"
      End If
    Next rng
  End With
End Sub
```

执行以上代码后，E-mail 地址提取结果如图 14.34 所示。

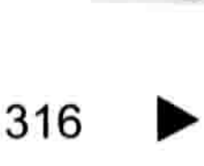
	A	B
1	张文男http://zhannwen.com25岁ZW@163.com游泳1.7米55公斤	ZW@163.com
2	李高梅女http://hanShen.cn33岁Gaomei.Li@126.cn音乐1.55米43公斤	Gaomei.Li@126.cn
3	杨子江男44岁棒球69公斤	资料不完整
4	朱仝男http://ShuiHu.cn33岁Shuihu_168@yahoo.com电子游戏59公斤	Shuihu_168@yahoo.com

图 14.34　获取 E-mail 地址

思路分析：

E-mail 地址的格式为“用户名@域名”，其中用户名包括字母、数字、正负号、小数点和下画线，域名包括字母、数字、负号、小数点和下画线，基于此特点，采用“\w”加“[-+.]”组合可以完整匹配用户名部分，而域名部分则采用用户名部分的条件稍加变化即可。其中“\w”等于“[a-zA-Z_0-9]”。

由于邮件地址只有一个，所以本例不需要循环，直接用“.Execute(rng.Text)(0)”获取目标即可。其中参数 0 表示数组的下限，数组在默认状态下下限为 0。



本例文件参见光盘：..\第十四章\14-20 获取 E-mail 地址.xlsm

14.3.7　提取文件的路径与文件名

案例要求：从文件名称中提取文件的路径和文件名称。

过程代码：

```
'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
Sub 分别从全名中提取文件的路径和名称()
  Dim Mystr As String, FileName As String
  With Application.FileDialog(msoFileDialogOpen)
    If .Show = -1 Then FileName = .SelectedItems(1) Else Exit Sub
  End With
  With CreateObject("VBSCRIPT.REGEXP")
```

```
    .Pattern = "[^\\]*$"
    Mystr = .Replace(FileName, "")
    .Pattern = ".*\\"
    Mystr = Mystr & Chr(10) & .Replace(FileName, "")
    MsgBox Mystr
  End With
End Sub
```

执行以上过程，在弹出的对话框中任意选择一个文件，在单击“打开”按钮后，程序会在信息框中显示指定文件的路径和文件名称，效果如图14.35所示。

图14.35 分别提取文件的路径和名称

思路分析：

文件完整名称是“d:\文件夹\文件名.后缀名”这种格式，文件名称是最后一个“\”右边的字符串，而文件名以外的所有字符即为路径名称。基于此特点，设置搜索条件时找出最后一个“\”的位置即可。

搜索条件“.*\\”表示以任意字符开头，长度为任意位，并且以字符“\”结尾。这正是文件路径的特点；搜索条件“[^\\]*$”表示以除“\”以外的任意字符开头、以任意长度的任意字符任意结尾，而且必须位于行尾。其中元字符“^”用于排除“\”符号。

本例文件参见光盘：..\第十四章\14-21 返回文件名与路径.xlsm

14.3.8 汇总人民币

案例要求：“美元:123元人民币:44元英镑:100元美元:44元人民币:300.06元。”字符串中包括多种货币，现要求仅对其中人民币求和。

过程代码：

```
Sub 汇总人民币()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Dim Mystr As String, Result As Double, Item
  Mystr = "美元:123元人民币:44元英镑:100元美元:44元人民币:300.06元"
  With CreateObject("VBSCRIPT.REGEXP")
    .Global = True
    .Pattern = "人民币:(\d+.(\d+)?)(?=元)"
    For Each Item In .Execute(Mystr)
      Result = Result + Replace(Item, "人民币:", "")
    Next
    MsgBox Result
  End With
End Sub
```

执行以上过程后，将得到结果344.06。

过程思路：

从字符串中找规律，可以发现待汇总的对象总是以“人民币：”开头、以“元”字结束。利用此特点可以设置匹配条件——“人民币:(\d+.(\d+)?)(?=元)”，它表示获取“人民币：”和“元”中间的数字，可以包括小数点，而“元”字本身不参与匹配。

由于Excel不支持反向预查，所以“人民币：”4个字会显示在匹配结果中，需要使用VBA的Replace函数将它替换掉。

本例文件参见光盘：..\第十四章\14-22 仅汇总人民币.xlsm

14.3.9 开发分列函数

案例要求：如图 14.36 所示的是购物记录表，由于多种物品的购买记录都放在同一个单元格中不利于查看和统计，因此要求使用 VBA 对它们分列，产品、数量和单位分别放在不同的单元格中。

	A
1	花生15公斤,黄豆234斤,大米20袋
2	白菜2公斤,黄花菜45.1KG
3	土豆2袋,米3公斤,豆234.45公斤
4	珍珠米88.3公斤
5	食盐12袋,味精8包

图 14.36　待分列的购物记录

过程代码：

```
Sub 增强型分列()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Dim Mystr As String, i As Byte, arr, LenRng As Byte
  With CreateObject("VBSCRIPT.REGEXP")
    Application.ScreenUpdating = False
    For Each rng In Range("a1:a5")
      If Len(rng) > 0 Then
        LenRng = (Len(rng) - Len(Replace(rng, ",", ""))) + 1) * 3
        arr = Array("[\u4E00-\u9FA5]+|[a-zA-Z]+", "\d+(\.\d*)?", "[\u4E00-\
u9FA5]+|[a-zA-Z]+")
        Mystr = rng.Text
        For i = 1 To LenRng
          .Pattern = arr(((i - 1) Mod 3))
          rng.Offset(0, i) = .Execute(Mystr)(0)
          Mystr = Mid(Mystr, Len(.Execute(Mystr)(0)) + 1, 99)
          If Left(Mystr, 1) = "," Then Mystr = Mid(Mystr, 2, 99)
        Next i
      End If
    Next rng
    Application.ScreenUpdating = True
     Range("a1").CurrentRegion.Borders.LineStyle = 1
  End With
End Sub
```

执行以上过程后可得到如图 14.37 所示的结果。

	A	B	C	D	E	F	G	H	I	J
1	花生15公斤,黄豆234斤,大米20袋	花生	15	公斤	黄豆	234	斤	大米	20	袋
2	白菜2公斤,黄花菜45.1KG	白菜	2	公斤	黄花菜	45.1	KG			
3	土豆2袋,米3公斤,豆234.45公斤	土豆	2	袋	米	3	公斤	豆	234.45	公斤
4	珍珠米88.3公斤	珍珠米	88.3	公斤						
5	食盐12袋,味精8包	食盐	12	袋	味精	8	包			

图 14.37　分列效果

思路分析：

本例待分列的字符串包含品名、数量和单位，由于 3 段字符串的位置都有明显的规律，因此可以通过正则表达式将它们逐一分离出来。

本例的数组变量 arr 中存放了 3 个字符串，它们分别用于提取品名、数量和单位。

由于在循环语句中正则表达式的匹配条件是逐一使用 arr 中第 1 个、2 个、3 个、1 个、2 个、3 个……元素，因此本例采用 MOD 运算符从 arr 中按顺序获取对应的值并赋值为 Pattern 属性。MOD 运算符配合循环语句刚好可以得到 1、2、3、1、2、3……这类有规律的序列。

数据源中的逗号不参与匹配，但是要搜索单位和品名时需要以它作为分界点，因此在使用正则表达式取值之前不能将它替换掉，而是在逗号左边的值已经完成匹配后再替换掉。

搜索条件“[\u4E00-\u9FA5]+|[a-zA-Z]+”中的“[\u4E00-\u9FA5]”部分代表任意汉字，“[a-zA-Z]”部分代表任意字母，两者是并列关系，因此中间使用“|”分隔开。在条件中添加“[a-zA-Z]”的目的在于可以匹配字母组成的品名，但是实际上本例中的品名全是汉字，因此也可以删除这一段代码。不过为了程序的通用性最好保留。

本例文件参见光盘：..\第十四章\14-23 分列的高级应用.xlsm

14.3.10 删除重复字词

案例要求：在写文章时绝大多数情况下字或者词组不会连续多次出现，如果连续出现则有可能是错误的。本例要求用正则表达式检查单元格中的文章中是否有重复出现的单字与词组。

过程代码：

```
Sub 标示重复出现的单字符()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Dim Item, rng As Range
  With CreateObject("VBSCRIPT.REGEXP")
    .Pattern = "(.)\1+"
    .Global = True
    For Each rng In Range("a1:a2")
      If .Test(rng) Then
        For Each Item In .Execute(rng)
          rng.Characters(Start:=InStr(rng,   Item),   Length:=Len(Item)).Font.
ColorIndex = 3
        Next
      End If
    Next rng
  End With
End Sub
```

以上代码适用于在单个字符重复出现时进行添加标示，如图14.38为添加标示后的效果。当然重复出现的字符可能有错，也可能无错，这需要人工进行判断。而程序代码只需要做到具有标示重复字符的功能，对于工作的帮助就已不容小觑了。

	A	B
1	城上风光莺语乱，城下烟波春拍拍。绿杨芳草几时休？泪眼愁肠先已断。	
2	情怀渐渐成哀晚，鸾镜朱颜惊暗换换换。昔年多病厌芳尊，今日芳尊惟恐浅。	

图14.38 标示重复出现的单个字符

以下代码用于检查重复出现的词组，即至少两个字重复出现才标示。例如“一大一小”、“高高兴兴”都不算重复，“中国中国”和“ExcelExcel”才算重复。

```
Sub 标示重复出现的词组()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Dim Item, rng As Range
  With CreateObject("VBSCRIPT.REGEXP")
    .Pattern = "(.{2,})\1+"
    .Global = True
      For Each rng In Range("a1:a2")
      If .Test(rng) Then
        For Each Item In .Execute(rng)
          rng.Characters(Start:=InStr(rng,   Item),   Length:=Len(Item)).Font.
ColorIndex = 3
        Next
      End If
```

```
    Next rng
  End With
End Sub
```

假设工作表中有如图 14.39 所示的文章——《荷塘月色》，当执行以上过程后，如果文章中现了重复词组，则会全部标示出来。然后人工核对这些重复词语是否需要修改。

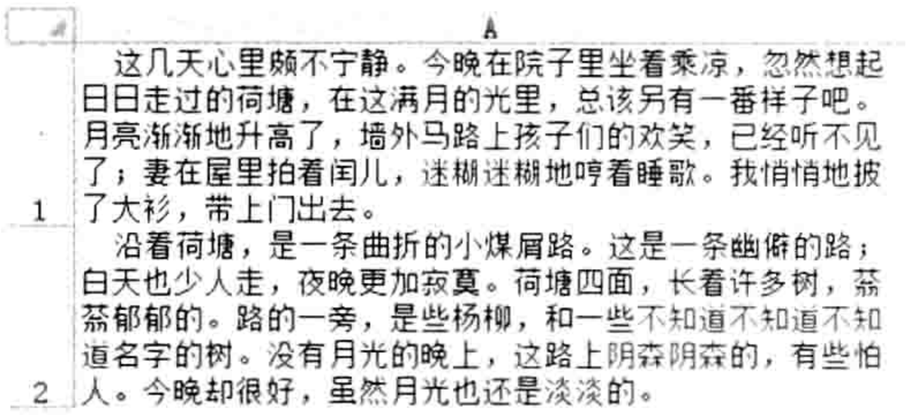


图 14.39　标示重复出现的词组

过程思路：

第一个过程“(.)\1+”条件可以分 3 段理解：

（1）“(.)”表示任意一个单字符，为了便于配合后面的“\1”引用，才使用括号。

（2）“\1”组合不是代表数字 1，而是后向引用的匹配模式。数字表示匹配前面的第 1 个子模式。例如“(A)(B)\2”等价于“ABB”，即“\2”表示引用第 2 个子模式；“(A)(B)\1{1,3}”等价于“ABA{1,3}”。

（3）元字符“+”表示重复出现至少一次。本例中的匹配条件对标点符号也有用。

“(.)\1+”所匹配的对象是单个字符至少重复一次的字符串，例如“高高”。

“(.{2,})\1+”所匹配的对象是至少两个字符重复不少于一次的字符串。例如“高兴高兴”或者“狼来了狼来了”。“(.{2,})”部分表示至少两个字符，“\1+”表示将前面的子模式重复至少一次，两者组合起来表示重复出现的词组。

如果只需要标示出现 3 次的词组，可以改条件为“(.{2,})\1{2}”。后者“{2}”表示重复两次，加上前面的一次则为出现 3 次的词组。

本例文件参见光盘：..\第十四章\14-24 标示重复字词.xlsm

学正则表达式可以在 VBA 中逐步调试，当然网上也有便利的工具可用。读者可以在以下网址下载免费版本的正则表达式工具：

http://www.cnblogs.com/Files/JimmyZhang/RegexTester.zip

◆　◆　◆

第15章 详解字典应用

字典的学名是 Dictionary，与 collection 集合功能相近，但比它更强大，在工作中可以发挥极大的效用。

熟用字典可以提升代码执行效率，也可以促进对数组的掌控能力，因为字典的条目对是以数组形式存在的。本章就字典的语法、属性、方法与应用详加解析。

15.1 Dictionary 对象基础

Dictionary 对象俗称字典，Dictionary 对象有它独有的方法和属性，而且 Dictionary 对象不是 Excel 内部集成的功能，需要从外部引用，因此在学习字典的应用前有必要了解 Dictionary 对象的基础知识。

15.1.1 Dictionary 对象的调用

Dictionary 对象集成在动态链接库文件 Scrrun.dll 中，非 Excel 自带功能，注册 Scrrun.dll 文件后才可以调用 Dictionary 对象的属性与方法。

注册 Scrrun.dll 的方法是从 Windows 的“开始”→“运行”菜单中运行以下代码：

```
Regsvr32 Scrrun.Dll
```

运行代码后，如果提示运行成功，那么可以在模块中通过代码调用字典对象的资源，否则表明缺少文件，应在其他计算机中将 Scrrun.Dll 文件复制过来，然后再次运行以上代码。

注册 Scrrun.Dll 文件后，使用 VBA 代码调用 Dictionary 对象还区分前期绑定和后期绑定，两种绑定方式各有优点和缺点。

1. 前期绑定

前期绑定是指在“引用”对话框中引用动态链接库文件，前期绑定的优点是录入代码时可以调用字典对象的属性与方法列表，从而提升录入代码的速度，同时也避免拼写错误。

添加引用的步骤如下。

step 1 在 VBE 窗口中单击菜单中的“工具”→“引用”命令，打开“引用”对话框。

step 2 在引用列表中找到“Microsoft Scripting Runtime”，将它打钩，并单击“确定”按钮。

Windows 分为 32 位和 64 位两种版本，在 32 位的 Windows 中 Scrrun.dll 文件的路径是 C:\Windows\ system32\scrrun.dll，而在 64 位 Windows 中 Scrrun.dll 文件的路径是 C:\Windows\sysWOW64\ scrrun.dll。

如图 15.1 所示为引用“Microsoft Scripting Runtime”的窗口。

添加引用后，可以声明一个名为“New Dictionary”的对象变量，然后通过该变量可调用字典对象的一切资源，包括属性与方法。如图 15.2 所示即为声明字典对象的变量，并在录入变量后自动弹出属性与方法列表。

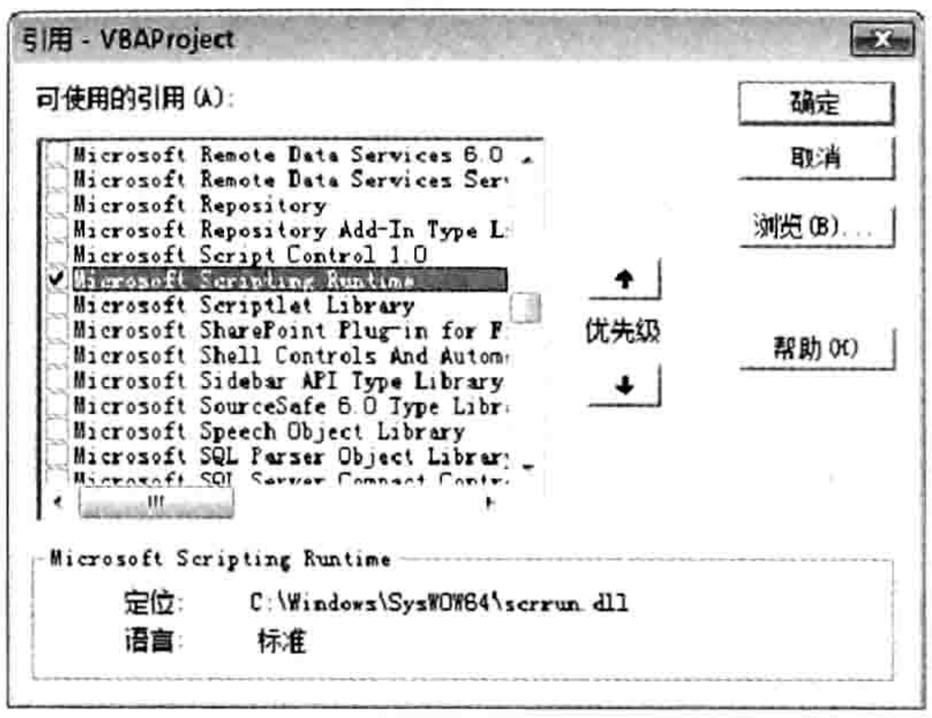


图 15.1　引用“Microsoft Scripting Runtime”

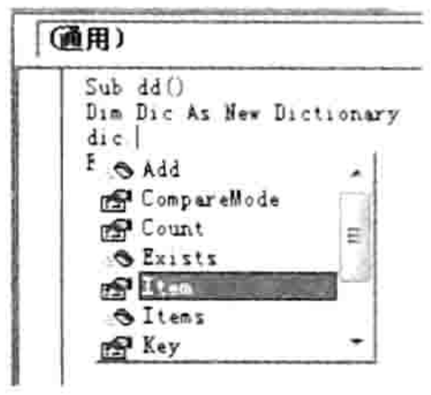


图 15.2　字典的常数列表

2. 后期绑定

前期绑定方式有利于程序开发者调用属性与方法列表，但是不利于使用者。当把代码复制给他人后，他人的 Excel 工作簿中没有手工添加引用就无法执行代码。基于此，Excel VBA 允许用户使用后期绑定。

所谓后期绑定是指定通过代码创建对象引用，它的优点是将代码复制给用户后可以直接使用，不管“引用”对话框中是否已经引用“Microsoft Scripting Runtime”。

后期绑定字典对象的代码是：

```
CreateObject("scripting.dictionary")
```

当通过 CreateObject 函数创建引用后，就可以调用字典的所有资源了。

后期绑定的缺点是录入代码时不弹出属性与方法列表，用户必须记得字典对象的属性和方法名称。不过读者可以采用本书第 12 章的办法解决：要么直接复制笔记中的代码，要么使用工具自动产生与字典相关的代码，而不用通过记忆录入代码。

3. 用代码实现全自动绑定

既然动态链接库文件的存放路径总是固定的，可以通过手工添加引用实现前期绑定，那么理论上也可以利用代码自动添加引用。

为了实现这个目标，先来了解一些相关知识。

为了防止宏病毒，微软禁止在默认设置下用代码去操作 VBE 界面中的任何对象，不过可以选中“信任中心”的“信任对 VBA 工程对象模型的访问”复选框来突破这道防线。

如图 15.3 所示为设置界面，按此方法设置后就可以利用代码实现前期绑定字典对象。

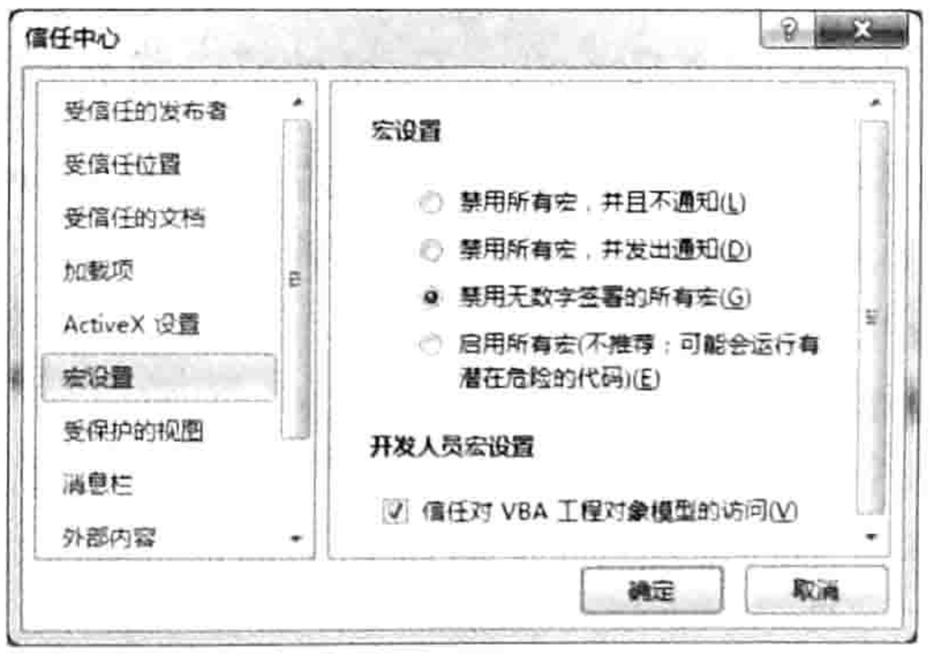


图 15.3　信任中心

利用代码引用字典对象，其实就是将 Scrrun.Dll 文件添加到引用列表。前面已经讲过 32 位和

64 位的 Windows 的系统文件的路径是不同的，不过微软特意设置了纠错功能，假设用户使用的是 64 位的 Windows，引用“C:\Windows\system32”路径下的文件时会自动跳转到“C:\Windows\sysWOW64”中去，因此书写代码时可以不用考虑 Windows 版本问题，直接使用以下代码：

```
Sub 添加字典引用()'代码存放位置：模块中
 Application.VBE.ActiveVBProject.References.AddFromFile "C:\Windows\system32\
scrrun.dll"
End Sub
```

如果需要删除已引用的字典对象，那么可以采用以下代码：

```
Sub 删除字典的引用()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  For Each 引用 In ThisWorkbook.VBProject.References
    If 引用.Description = "Microsoft Scripting Runtime" Then
      Application.VBE.ActiveVBProject.References.Remove 引用
      Exit For '终止循环
    End If
  Next 引用
End Sub
```

本例文件参见光盘：..\第十五章\15-1 用代码引用字典对象或者删除引用.xlsm

15.1.2 Dictionary 的特点

字典对象用于储存两个相关联的一维数组，包括条目和关键字，所以也可以将字典看作是特殊的数组。当对数组知识有足够的了解后，学习字典对象会相当轻松。

Dictionary 对象用于存放关键字与条目组成的条目对，图 15.4 可以展示字典对象的内部情况。其中第一行不允许有重复的关键字，第二行允许有重复的条目，每个关键字与每个条目之间一一对应。

图 15.4 字典的关键字与条目示意图

由于字典对象的关键字 Key 具有唯一性，因此可以通过它提取一组数据的唯一值。

前面已经提到过集合对象 collection 也可以提取一组数据的唯一值，不过提取成功后要将这些唯一值输出到工作表中时只能一个个地输出，而使用字典对象可以借用 Keys 方法一次性输出。

15.1.3 Dictionary 对象的属性与方法

Dictionary 对象有 Add、Items、Keys、Remove、RemoveAll 和 Exists 六个方法和 Item、Key、Count 和 CompareMode 四个属性。

1. Add 方法

功能：添加一对相对应的关键字和条目到 Dictionary 对象，其语法如下：

```
object.Add Key, Item
```

其中 Key 代表关键字，同一个 Dictionary 对象中不允许出现重复的关键字。关键字宜采用文本形式，所以通常书写代码时通过 Cstr 函数将参数转换成 String 型，类似于集合对象 Collection 中

Add 方法的第 2 参数。

Item 参数代表条目，多个条目之间允许重复，条目的数据类型不限。

Key 和 Item 两个参数都是必选参数，所以使用 Add 方法向 Dictionary 对象添加的关键字和条目总是成对的，可通过 Item 方法来访问这一对条目。

以下过程可向 Dictionary 对象中添加一个条目对，关键字是“及时雨”，条目是“宋江”。

```
Sub 添加条目对()'代码存放位置：模块中
  With CreateObject("scripting.dictionary") '创建字典对象并引用
    .Add "及时雨", "宋江"                    '添加一个条目对，关键字是"及时雨"，条目是"宋江"
  End With
End Sub
```

当然有时只需要关键字，不需要条目，那么可以采用以下两种方法处理：

Object.Add "B", Nothing——向条目中添加空对象；

Object.Add "B", ""——向条目中添加空文本。

由于 Key 关键字具有唯一性，所以不能多次添加相同值到 Key 参数中，否则会出现如图 15.5 所示的错误提示。

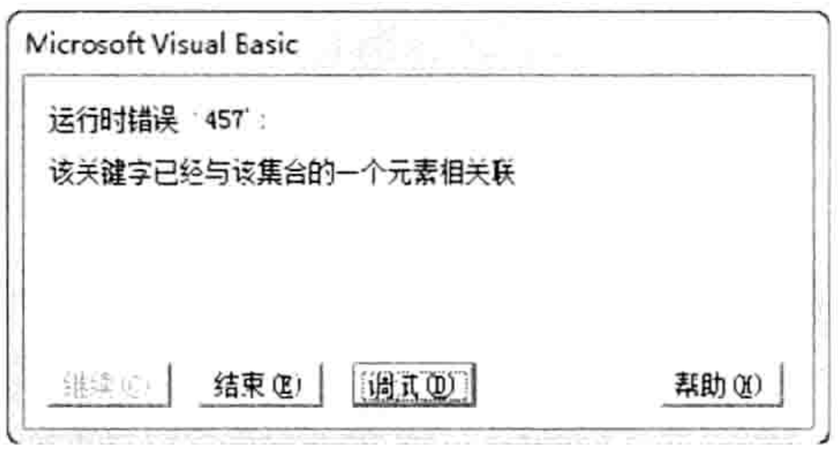


图 15.5　错误提示

本例文件参见光盘：..\第十五章\ 15-2 字典的方法和属性.xlsm

2. Key 属性与 Item 属性

Key 属性的功能是设置字典对象的关键字，可写不可读。其语法如下：

```
object.Key(key) = newkey
```

Item 属性的功能是返回或者设置字典对象的条目，可写亦可读。其语法如下：

```
object.Item(key) [= newitem]
```

其中“[= newitem]”部分是可选的，如果忽略该部分则表示读取关键字对应的条目。

集合对象 Collection 中添加的成员是不可以修改的，而通过字典对象的 Key 属性与 Item 属性可以修改关键字与条目，所以字典与集合相比有着更多的优越性。

以下过程包括使用 Add 方法创建条目对、使用 Item 属性修改与获取关键字所对应的条目、利用 Key 属性修改关键字。可以在选择过程后按<F8>键逐步运行代码，观察每句代码的执行结果，从而理解 Item 属性与 Key 属性的关系与区别：

```
'使用 Itemn 属性返回关键字对应的条目，或者设置条目，使用 Key 属性修改关键字名称
Sub Item 属性与 Key 属性()
 With CreateObject("scripting.dictionary")  '创建字典对象
'添加一个条目对，关键字是“及时雨”，条目是“宋江”
   .Add "及时雨", "宋江"
```

```
    '使用 Item 属性获取关键字所对应的条目，验证刚才添加的条目对
      MsgBox "关键字：及时雨" & Chr(10) & "条目：" & .Item("及时雨"), vbOKOnly, ".Add
""及时雨"", ""宋江"""
    '使用 Item 属性修改关键字所对应的条目
      .Item("及时雨") = "呼保义"
    '使用 Item 属性获取关键字所对应的条目，验证刚才修改后的条目对
      MsgBox "关键字：及时雨" & Chr(10) & "条目：" & .Item("及时雨"), vbOKOnly,
" .Item(""及时雨"") = ""呼保义"""
    '利用 Key 属性修改关键字，Key 属性只有设置作用，没有读取作用，即不能通过 Key 获取关键字的
    名称
      .Key("及时雨") = "宋公明"
    '使用 Item 属性获取关键字所对应的条目，验证刚才修改后的条目对
      MsgBox "关键字：宋公明" & Chr(10) & "条目：" & .Item("宋公明"), vbOKOnly, ".Key(""
及时雨"") = ""宋公明"""
  End With
End Sub
```

执行过程后将分别弹出三个信息框，每个信息框的标题是代码，信息框的内容是该代码对应的执行结果，如图 15.6 至图 15.8 所示。

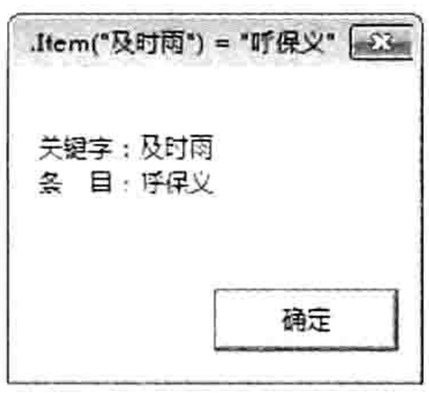


图 15.6　创建条目对的结果　　图 15.7　修改条目的结果　　图 15.8　修改关键字的结果

假设通过 Item 属性修改关键字对应的条目时没有发现参数所指定的关键字，那么代码将会创建一个新的条目对。

例如通过 Add 方法创建了一个关键字为“及时雨”、条目是“宋江”的条目对，然后再通过 Item 属性修改关键字“呼保义”对应的条目，由于字典对象中本不存在名为“呼保义”的关键字，那么它会创建一个新的条目对。代码如下：

```
'使用 Item 属性添加条目对，和 Add 方法功能相近，用法不同
'Add 方法只能添加新的关键字与条目，而 Item 属性既可以新建也可以修改原有关键字所对应的条目
Sub 利用 Item 属性创建条目对()
  With CreateObject("scripting.dictionary") '创建字典对象
    .Add "及时雨","宋江"                    '添加一个条目对，关键字是“及时雨”，条目是“宋江”
    .Item("呼保义") = "宋江"                '添加一个条目对，关键字是“呼保义”，条目是“宋江”
    '分别验证关键字“及时雨”和“宋江”及其对应的条目
    MsgBox "关键字：及时雨" & Chr(10) & "条目：" & .Item("及时雨")
    MsgBox "关键字：呼保义" & Chr(10) & "条目：" & .Item("呼保义")
  End With
End Sub
```

执行以上代码后，字典对象中将存在两个条目对，分别为关键字“及时雨”对应条件“宋江”，关键字“呼保义”对应条目“宋江”。

基于此特性，也可以采用 Item 属性创建字典的条目对，不再局限于 Add 方法，而且还可以将

Item 属性配合循环语句使用，反复修改（累加或者递减）关键字对应的条目的值。

```
'利用 Item 属性累加关键字对应的条目，可用此技术对数据分类汇总，实现数据透视表相近的功能
Sub Item 方法创建条目对和累加条目()
  With CreateObject("scripting.dictionary")   '创建字典对象
    .Item("一车间") = 80                      '添加一个条目对，关键字是"一车间"，条目是 80
    .Item("二车间") = 90                      '添加一个条目对，关键字是"呼保义"，条目是"宋江"
    .Item("一车间") = .Item("一车间") + 100 '累加关键字“一车间”的值
    .Item("二车间") = .Item("二车间") + 120 '累加关键字“二车间”的值
  '分别验证关键字“一车间”和“二车间”及其对应的条目
    MsgBox "关键字：一车间" & Chr(10) & "条目：" & .Item("一车间")
    MsgBox "关键字：二车间" & Chr(10) & "条目：" & .Item("二车间")
  End With
End Sub
```

以上过程通过 Item 属性创建了两个条目对，然后又通过 Item 属性累加条目的值，最后的结果是关键字“一车间”对应的值为 180，关键字“二车间”对应的值为 210。

使用此思路可以对数据分类汇总，在后面的案例中将有详细展示。

3. Keys 方法与 Items 方法

Keys 方法可返回一个没有重复值的一组数据，该数组包含一个 Dictionary 对象中的全部关键字，即从字典中获取第一行数据。其语法如下：

```
object.Keys
```

Keys 方法的返回值可导出到区域中，可赋值给列表框，也可以直接赋值给数组变量，或者通过 Join 函数将它合并为一个字符串。所以在取唯一值方面，字典对象比高级筛选、删除重复项、数据透视表和集合对象都更强大。

Items 方法也能返回一个一维数组，该数组包含一个 Dictionary 对象中的全部条目，即从字典中获取第二行数据。其语法如下：

```
object.Items
```

以下过程先用 Add 方法向字典中创建 4 个条目对，然后分别通过 Keys 方法和 Items 方法将两个一维数组输入到 A、B 列中：

```
'利用 keys 方法和 Items 方法输出所有关键字和所有条目
Sub Keys 与 Items 方法输入数组()
  With CreateObject("scripting.dictionary") '创建字典对象
    .Add "及时雨", "宋江"                    '添加第一个条目对
    .Add "黑旋风", "李逵"                    '添加第二个条目对
    .Add "拼命三郎", "石秀"                  '添加第三个条目对
    .Add "呼保义", "宋江"                    '添加第四个条目对
    Range("A1:A4") = WorksheetFunction.Transpose(.Keys)'将 4 个关键字导出到 A1:A4
区域中
    Range("B1:B4") = WorksheetFunction.Transpose(.Items) '将 4 个条目导出到 B1:B4
区域中
  End With
End Sub
```

由于 Keys 和 Items 的返回值都是一维数组，横向排列，所以导出到 A1:A4 和 B1:B4 区域前需

要利用工作表函数 Transpose 转置方向。输出结果如图 15.9 所示。

事实上也可以通过 Join 函数将关键字与条目显示在信息框中，过程中最后两句修改为以下语句即可，输入结果如图 15.10 所示。

```
MsgBox "关键字: " & Join(.Keys, ", ") & Chr(10) & "条目: " & Join(.Items, ", ")
```

	A	B
1	及时雨	宋江
2	黑旋风	李逵
3	拼命三郎	石秀
4	呼保义	宋江
5		

图 15.9 输出关键字与条目到工作表中

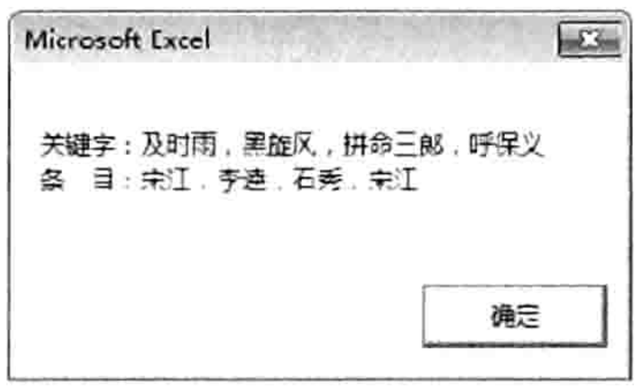


图 15.10 输出关键字与条目到信息框中

Items 和 Keys 都是字典对象的方法而不是属性，不能通过索引号引用其中单个元素。如果一定要获取单个元素，应借助数组变量实现。

例如获取索引号为 2 的关键字和条目，可在过程中插入以下代码：

```
'将关键字和条目输出到数组中，再通过数组的索引号获取其中单个元素的值
Sub 获取索引号为 2 的关键字与条目()
  With CreateObject("scripting.dictionary")      '创建字典对象
    .Add "及时雨", "宋江"                        '添加第一个条目对
    .Add "黑旋风", "李逵"                        '添加第二个条目对
    .Add "拼命三郎", "石秀"                      '添加第三个条目对
    .Add "呼保义", "宋江"                        '添加第四个条目对
    Dim arr1, arr2                               '声明两个变体量变量
    arr1 = .Keys                                 '将关键字赋予 arr1
    arr2 = .Items                                '将条目赋予 arr2
    '将数组中的索引号为 2 的值显示在信息框中
    MsgBox "关键字: " & arr1(2) & Chr(10) & "条目: " & arr2(2)
  End With
End Sub
```

4. Remove 方法与 RemoveAll 方法

Remove 方法功能用于从一个 Dictionary 对象中删除一个关键字和条目对。其语法如下：

```
object.Remove(key)
```

Remove 方法一次只能删除一个关键字和条目，而 RemoveAll 方法可以删除所有条目对。其语法如下：

```
object.RemoveAll
```

5. Exists 方法

Exists 方法用于判断 Dictionary 对象中是否包含某个关键字，如果返回值为 True 表示该关键字存在，否则不存在。其语法如下：

```
object.Exists(key)
```

6. Count 属性

Count 属性可返回 Dictionary 对象中的条目数量。语法如下：

```
object.Count
```

通常在导出字典对象的关键字或者条目之前需要计算条目数量，然后根据此数量重置区域的高度或者宽度，最后将关键字或者条目导出到区域中。

前面的 Sub 过程“Keys 与 Items 方法输入数组”中的代码“ Range("A1:A4") = WorksheetFunction.Transpose(.Keys)”就有必要修改区域地址 A1:A4，从而提升代码的通用性。修改的思路是先计算关键字的数量，然后根据此数量重置 Range("A1")的高度产生一个新的区域，此区域的单元格个数对应字典中关键字的个数。完整代码如下：

```
Range("A1").Resize(.count,1) = WorksheetFunction.Transpose(.Keys)
```

7. CompareMode 属性

CompareMode 属性用于设置或返回某个 Dictionary 对象中的比较字符串的比较模式。简单而言是指比较关键字时是否需要区分大小写，默认值为 0，表示区分大小写。

修改或者返回 CompareMode 属性的语法如下：

```
object.CompareMode[ = compare]
```

对参数 compare 赋值为 vbBinaryCompare 时表示区分大小写，赋值为 vbTextCompare 时表示不区分大小写。

15.2 Dictionary 对象的应用技巧

由于字典的 Key 关键字具有唯一性，无法添加多个同名的 Key，所以在工作中常用它获取唯一值。同时由于字典属于内存数组，可以在内存中执行数据处理，通常又使用字典对象来对代码提速。本节从取唯一值和提速方面展示字典的 5 个应用案例。

15.2.1 利用字典创建三级选单

案例要求：图 15.11 中包含了省、市、县名称，现要求在“三级选单”工作表的 A 列、B 列和 C 列中创建相关联的三级选单，例如 A 列的选单列表中只出现省份名称，B 列的选单列表中只出现隶属于 A 列指定省的市名，而 C 列的选单列表中则只出现隶属于 A 列、B 列指定的省市的县名。

	A	B	C	D
1	省份	市	县	
2	广东	梅州市	大埔县	
3	广东	梅州市	五华县	
4	广东	梅州市	梅县	
5	广东	肇庆市	封开县	
6	广东	肇庆市	广宁县	
7	广东	肇庆市	德庆县	
8	广东	韶关市	曲江县	
9	广东	韶关市	新丰县	
10	广东	韶关市	始兴县	
11	四川	内江	资中县	
12	四川	内江	隆昌县	
13	四川	内江	威远县	
14	四川	成都	金堂县	

三级选单 数据源

图 15.11 包含省、市、县名称的数据源

过程代码：

```
'①代码存放位置：Sheet1 的代码窗口中②随书光盘中有每一句代码的含义注释
Private Sub Worksheet_Change(ByVal Target As Range)
  If Target.Column = 1 Then Target(1).Offset(0, 1).Resize(1, 2).ClearContents
  If Target.Column = 2 Then Target(1).Offset(0, 1).ClearContents
End Sub

Private Sub Worksheet_SelectionChange(ByVal Target As Range)
If Target.Column > 3 Then Exit Sub
```

```
  Dim i As Integer, arr(), Mystr As String
  On Error Resume Next
  With CreateObject("Scripting.Dictionary")
     arr = Worksheets("数据源").Range("A2").Resize(Worksheets("数据源").
UsedRange.Rows.Count - 1, 3).Value
     Select Case Target.Column
     Case 1
       For i = 1 To UBound(arr, 1)
       .Add CStr(arr(i, 1)), ""
       Next
       Mystr = Join(.keys, ",")
       With Target(1).Validation
       .Delete
       .Add Type:=xlValidateList, AlertStyle:=xlValidAlertStop, Formula1:=
Mystr
       End With
     Case 2
       For i = 1 To UBound(arr, 1)
         If Target(1).Offset(0, -1) = arr(i, 1) Then
         .Add CStr(arr(i, 2)), ""
         End If
       Next
       Mystr = Join(.keys, ",")
       With Target(1).Validation
       .Delete
       .Add Type:=xlValidateList, AlertStyle:=xlValidAlertStop, Formula1:=
Mystr
       End With
     Case 3
       For i = 1 To UBound(arr, 1)
         If Target(1).Offset(0, -2) = arr(i, 1) And Target(1).Offset(0, -1) =
arr(i, 2) Then
         .Add CStr(arr(i, 3)), ""
         End If
       Next
       Mystr = Join(.keys, ",")
       With Target(1).Validation
       .Delete
       .Add Type:=xlValidateList, AlertStyle:=xlValidAlertStop, Formula1:=
Mystr
       End With
     End Select
    .RemoveAll
  End With
End Sub
```

将以下代码存放在“三级选单”工作表的代码窗口中，然后返回工作表界面，进入“三级”选单的A列，单击A列的任意单元格后会弹出如图15.12所示的一级选单，列表中包含了“数据源”工作中的所有省份名称。

当从列表中选择“四川”后，再选择它右边的单元格，单元格中会产生如图15.13所示的二级选单，其列表中包含了“数据源”工作表中对应于四川省的所有市名。

当从列表中选择“成都”后，再选择它右边的单元格，单元格中会产生如图15.14所示的三级选单，其列表中包含了“数据源”工作表中对应于四川省、成都市的所有县名。

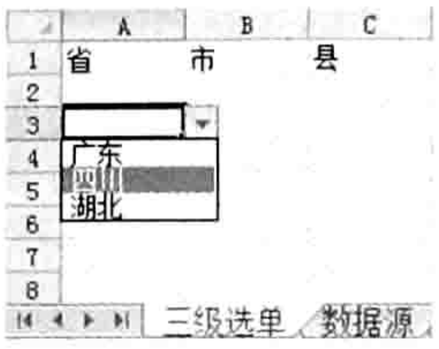


图 15.12 一级选单

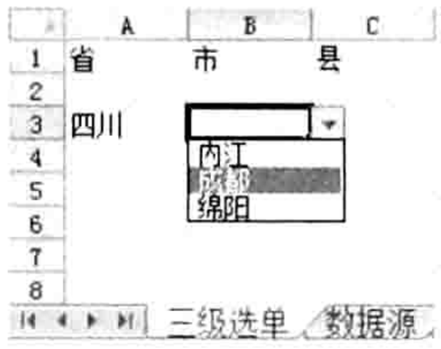


图 15.13 二级选单

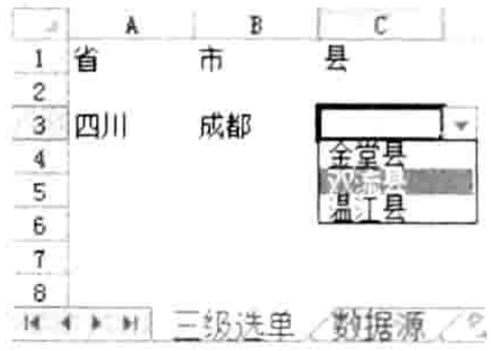


图 15.14 三级选单

思路分析：

三级选单是在选择单元格时产生的，因此创建三级选单的代码应放在 Worksheet_SelectionChange 事件过程中。同时由于 3 个选单只出现在 A 列、B 列和 C 列，因此在过程中首先使用条件语句判断 Target 的列号是否大于 3，大于 3 时直接结束过程。

接下来的工作是根据活动单元格的列号创建对应的选单。首先将“数据源”工作表中的省、市、县名称导入到数组 Arr 中，后续的数据读取操作都基于数组，而非单元格，从而提升代码的执行速度。

然后使用 Select Case 语句判断当前选区的列号是几，如果是 1，那么使用 For Next 循环语句遍历数组 arr 的第一列，使用字典的 Add 方法将所有省份添加到字典的关键字中，从而取出不重复的省份名称，然后通过 Join 函数将字典的关键字转换成字符串写入数据有效性的序列来源中，从而让单元格产生下拉选单，选单中包含不重复的省份名称。

如果 Select Case 语句的判断结果是 2，那么仍然遍历数组的第一列中的省份名称，使用它与当前选区左边的单元格进行比较，如果相同，那么将数组中的这些省份名称右边的市名导入到字典对象的关键字中，从而提取不重复的市名。最后用 Join 函数将字典的关键字转换成字符串写入数据有效性的序列来源中，从而让单元格产生第二级下拉选单，选单中包含不重复的市名。

如果 Select Case 语句的判断结果是 3，那么参照结果为 2 的思路处理，仅仅是比较时多了一个条件，需要省、市名称同时一致时才将数组 arr 中第 3 列的值导入到字典对象的关键字中。

本例的 Worksheet_Change 事件过程的功能是：修改 A 列的值时清除右边两个单元格，修改 B 列的值时清空右边一个单元格。清空单元格的目的是避免右边的市或者县的名与当前选择的省或者市的名没有隶属关系，从而强制用户重新选择正确的市名或者县名。

本例文件参见光盘：..\第十五章\ 15-3 创建三级选单.xlsm

15.2.2 分类汇总

案例要求：图 15.15 中 A1:C12 为产品出库表，要求对各产品进行分类合计，将结果存放在 E 列和 F 列中。

	A	B	C	D	E	F
1	日期	产品	出库数		产品	出库合计
2	2014/7/2	梅花刀	54		梅花刀	91
3	2014/7/2	一字刀	20		一字刀	62
4	2014/7/2	六角螺丝	28		六角螺丝	28
5	2014/7/3	钳子	23		钳子	23
6	2014/7/4	一字刀	13		2号钉	80
7	2014/7/4	梅花刀	14		改锥	35
8	2014/7/4	2号钉	38			
9	2014/7/5	梅花刀	23			
10	2014/7/6	2号钉	42			
11	2014/7/6	改锥	35			
12	2014/7/6	一字刀	29			

图 15.15 对出库明细表分类汇总

过程代码：

```
Sub 分类汇总()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
```

```
  On Error Resume Next
  With CreateObject("scripting.dictionary")
    For Item = 2 To Cells(Rows.Count, 2).End(xlUp).Row
      .Item(CStr(Cells(Item, 2))) = .Item(CStr(Cells(Item, 2))) + Cells(Item,
3).Value
    Next
    Range("E1:F1") = Array("产品", "出库合计")
    Range("e2").Resize(.Count, 1) = WorksheetFunction.Transpose(.keys)
    Range("F2").Resize(.Count, 1) = WorksheetFunction.Transpose(.items)
    Range("e1").CurrentRegion.Borders.LineStyle = 1
  End With
End Sub
```

执行以上代码后可以分类汇总效果，类似于数据透视表的汇总功能。

思路分析：

对数据分类汇总主要包含两项内容，其一是提取产品名称的唯一值，而字典对象的关键字刚好具备唯一性，因此宜用字典对象实现本例需求。

其二是逐一累加每个产品名称所对应于 C 列的出库数，由于字典对象的条目可以随意修改，因此在循环语句中修改字典对象的 Item 属性即可实现累加产品的出库数量。

由于字典对象的 Keys 方法和 Items 方法只能输出一维的横向数组，本例需要将结果纵向存放，所以需要使用工作表函数 Transpose 将它转置方向后再输出到单元格中。

事实上本例也可以使用数据透视表实现分类汇总，完整代码如下：

```
Sub 生成透视表()'代码存放位置：模块中
  '创建透视表，使用 R1C1 样式，其中“R1C2:R12C3”即 B1:C12 区域，表示数据源
  '而“R1C5”即 E1 单元格，表示存放透视表单元格(仅指定目标区域左上角的单元格)
  With     ActiveWorkbook.PivotCaches.Create(xlDatabase,     "R1C2:R12C3").
CreatePivotTable("R1C5")
  '将“产品”字段的位置设置为“行标签”
  ActiveSheet.PivotTables(1).PivotFields("产品").Orientation = xlRowField
  With ActiveSheet.PivotTables(1).PivotFields("出库数") '引用“出库数”字段
    .Orientation = xlDataField  '将该字段的位置设置为“数值”字段，即汇总对象
    .Function = xlSum '将“数值”字段的汇总方式设置为求和，如果改用 xlAverage 则是求平均值
    End With
  End With
End Sub
```

透视表的汇总功能很强大，只不过结果只能存放在区域中。字典对象产生的汇总结果在数组中，其优点是可以在内存中执行下一步运算，运算完后再输出到单元格中。

CD-ROM 本例文件参见光盘：..\第十五章\15-4 分类汇总.xlsm

15.2.3 对多列数据相同者应用背景色

案例要求：当工作簿中数据太多时，对整个数据区域添加条件格式将造成工作簿占用空间偏大，开启速度降低。现要求使用字典对象的技术对图 15.16 中姓名、班级、学号与性别数据皆相同者进行颜色标示，不允许使用条件格式实现。

过程代码：

```
Sub 四列皆相同者添加背景色()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
```

```
  Dim rng As Range, i As Integer
  On Error Resume Next
  With CreateObject("scripting.dictionary")
    For Item = 2 To Cells(Rows.Count, 1).End(xlUp).Row
      .Add Cells(Item, 1).Value & "@" & Cells(Item, 2).Value & "@" & Cells(Item,
4).Value & "@" & Cells(Item, 5).Value, ""
      If Err.Number <> 0 Then
        If rng Is Nothing Then
          Set rng = Cells(Item, 1).Resize(1, 5)
        Else
          Set rng = Union(rng, Cells(Item, 1).Resize(1, 5))
        End If
        Err.Clear
      Next
    End With
    If Not rng Is Nothing Then rng.Interior.ColorIndex = 3
End Sub
```

执行以上过程，工作表中 B、C、D、E 四列的值完全相同时将会自动添加红色背景，如图 15.16 和图 15.17 所示分别为着色前后的成绩表。

	A	B	C	D	E
1	姓名	班级	成绩	学号	性别
2	钱	一班	75	101	男
3	孙	一班	68	471	女
4	李	二班	73	308	男
5	周	一班	75	120	男
6	吴	一班	77	395	男
7	郑	一班	61	696	女
8	王	二班	99	201	女
9	李	二班	74	308	男
10	陈	三班	82	676	男
11	褚	三班	68	492	男

图 15.16　着色前的成绩表

	A	B	C	D	E
1	姓名	班级	成绩	学号	性别
2	钱	一班	75	101	男
3	孙	一班	68	471	女
4	李	二班	73	308	男
5	周	一班	75	120	男
6	吴	一班	77	395	男
7	郑	一班	61	696	女
8	王	二班	99	201	女
9					
10	陈	三班	82	676	男
11	褚	三班	68	492	男

图 15.17　着色后的成绩表

思路分析：

本例中要求四列数据都完全相同才算重复，因此将四列的值串连起来写入字典对象的关键字中，然后根据 Add 方法在执行期间是否出错判断数据是否重复，当数据重复时 Err.Number 属性的值必定不等于 0。因此本例的重点在于字典对象的 Add 方法及计算 Err.Number。

在串连四列数据时，为了提升代码的通用性，在其中加入了间隔符“@”。例如第 2 行 B 列的值是 1，C 列的值是 23，两者串连后的结果为 123，假设第 3 行 B 列的值为 12，C 列的值是 3，两者串连后的结果仍然是 123。当代码中加入了分隔符“@”后不会判断失误。

本例代码中使用了 Union 方法将多个符合条件的区域合并为一个 Range 对象，待循环完成后再一次性对 Rng 变量所代表的区域进行着色，此举可以大大提升工作效率。

CD-ROM　本例文件参见光盘：..\第十五章\15-5 标示重复值.xlsm

15.2.4　按姓名计数与求产量平均值

案例要求：图 15.18 中 B 列和 C 列包含了产量冠军的姓名和该月的产量，现要求在 E 列到 G 列中罗列出不重复的产量冠军、每人夺冠次数和平均产量。

	A	B	C	D	E	F	G
1	月份	产量冠军	产量		产量冠军	夺冠次数	平均产量
2	一月	朱华青	998		朱华青	3	906.6666667
3	二月	陈真亮	977		陈真亮	4	972
4	三月	朱华青	905		黄花秀	3	900.3333333
5	四月	黄花秀	956		陈秀敏	1	985
6	五月	陈秀敏	985				
7	六月	黄花秀	895				
8	七月	陈真亮	960				
9	八月	朱华青	817				
10	九月	陈真亮	974				
11	十月	黄花秀	850				
12	十一月	陈真亮	977				

图 15.18 月产量冠军信息表

过程代码：

```
Sub 分类计数与求平均()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Dim i As Integer, Dic1 As Object, Dic2 As Object, NameArr(), CountArr()
  On Error Resume Next
  Set Dic1 = CreateObject("scripting.dictionary")
  Set Dic2 = CreateObject("scripting.dictionary")
  For i = 2 To Cells(Rows.Count, 2).End(xlUp).Row
    Dic1.Item(CStr(Cells(i, 2))) = Dic1.Item(CStr(Cells(i, 2))) + 1
    Dic2.Item(CStr(Cells(i, 2))) = Dic2.Item(CStr(Cells(i, 2))) + Cells(i,
3).Value
  Next
  NameArr = Dic1.keys
  CountArr = Dic1.items
  For i = 0 To UBound(CountArr)
    Dic2.Item(NameArr(i)) = Dic2.Item(NameArr(i)) / CountArr(i)
  Next i
  Range("E1:G1") = Array("产量冠军", "夺冠次数", "平均产量")
  Range("E2").Resize(Dic1.Count, 1) = WorksheetFunction.Transpose(NameArr)
  Range("F2").Resize(Dic1.Count, 1) = WorksheetFunction.Transpose(CountArr)
  Range("G2").Resize(Dic2.Count, 1) =WorksheetFunction.Transpose(Dic2.items)
  Range("E1").CurrentRegion.Borders.LineStyle = 1
End Sub
```

执行以上代码，将返回图 15.18 中 E1:G5 区域所示的结果。

思路分析：

字典对象包含两个一维数组，本例中要产生 3 个一维数据，因此需要创建两个字典对象，其中第一个字典对象 Dic1 用于存放不重复的产量冠军和对应的夺冠次数，第二个字典对象用于存放不重复的产量冠军和对应的产量总和——在循环语句执行期间无法取得产量的平均值，只能取得累加值，因此采用先累加后转换成平均值的思路。

当通过循环取得每个产量冠军的夺冠次数和对应的产量合计后，再一次通过循环语句将产量合计除以夺冠次数得到平均产量，最后将它们输出到工作表的 E 列、F 列和 G 列中。

当过程中需要使用多个字典对象时，不宜使用 With 语句引用 CreateObject("scripting.dictionary")，而是将 CreateObject("scripting.dictionary")创建的对象赋予变量，然后通过变量去调用字典对象的属性与方法。

本例和前一例一样可以使用数据透视表来完成同等需求，随书光盘中提供了透视表代码。

本例文件参见光盘：..\第十五章\15-6 计算产量冠军的夺冠次数和平均产量.xlsm

15.2.5 按品名统计半年内的产量合计

案例要求：图 15.19 中 A 列到 H 列包含了公司半年内的产量明细，现要求按产品名称合计所有产量，结果如图 15.19 中 J1:K5 区域所示。

	A	B	C	D	E	F	G	H	I	J	K
1	车间	产品	一月	二月	三月	四月	五月	六月		品名	合计
2	一车间	镙丝	10822	10134	10318	9158	9208	11099		镙丝	121746
3	一车间	扳手	8056	11043	11258	10836	8181	9656		扳手	117351
4	一车间	梅花刀	11451	11162	9494	11848	11486	8225		梅花刀	186214
5	二车间	镙丝	11799	9456	10099	11069	8214	10370		老虎钳	119265
6	二车间	扳手	9875	9192	10491	10591	9055	9117			
7	二车间	梅花刀	11320	11299	10357	11945	11644	8907			
8	二车间	老虎钳	10781	11920	8975	10136	8425	11998			
9	三车间	梅花刀	10705	8062	10301	8400	8412	11196			
10	三车间	老虎钳	9138	8182	9183	9528	9204	11795			

图 15.19　按品名汇总 6 个月的产量

过程代码：

```
Sub 使用字典对产品的产量汇总()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Dim i As Integer
  On Error Resume Next
  With CreateObject("scripting.dictionary")
    For i = 2 To ActiveSheet.UsedRange.Rows.Count
      .Item(CStr(Cells(i, 2))) = .Item(CStr(Cells(i, 2))) + WorksheetFunction.
Sum(Range("C" & i & ":H" & i))
    Next i
    Range("j1:k1") = Array("品名", "合计")
    Range("j2").Resize(.Count, 1) = WorksheetFunction.Transpose(.keys)
    Range("k2").Resize(.Count, 1) = WorksheetFunction.Transpose(.items)
    Range("j1").CurrentRegion.Borders.LineStyle = 1
  End With
End Sub
```

思路分析：

本例的数据源分布在 6 列中，使用数据透视表和字典对象都无法直接汇总。

本例的思路是利用工作表函数 Sum 合计每一行的产量，从而将 6 列数据转换成单列数据，然后再将它们写入到字典对象的条目中去。

本例通过循环语句将产品名称写入字典对象的关键字中，从而去除重复值，在写入关键字的同时将每一个关键字所代表的产品的产量合计累加到条目中。当循环完成后，字典对象的关键字将会包含所有不重复的产品名称，而条目中则包含了每一个产品名称的累计产量。

最后将字典对象的关键字和条目转置方向，并通过 Keys 方法和 Items 方法导出到工作表中。

CD-ROM　本例文件参见光盘：..\第十五章\ 15-7 按品名统计半年内的产量合计.xlsm

◆　◆　◆

第16章 开发自定义函数

Excel 2010 提供了 400 多个函数，不过与每个用户工作相关的函数并不多。对于报表数据量大或者需求比较高的用户而言，内置的函数不足以满足日常需求。

Excel VBA 可以开发函数，而且新开发的函数和内置的函数具有同等的易用性。

本章首先介绍自定义函数的语法，然后会提供 10 多个自定义函数的源代码和思路说明，从而帮助读者掌握开发函数的基本技巧，以及了解 Function 过程与 Sub 过程的区别。

16.1 自定义函数的功能和语法

Function 过程即自定义函数，它和 Sub 过程有着比较显著的区别，包括功能上的区别和开发过程的区别。

Function 过程的功能偏重于运算，执行函数过程的目的通常是获取函数的运算结果。

如果需要将运算结果直接显示在单元格中，而且该结果需要实时更新，或者需要将运算结果传递给另一个过程，那么宜用 Function 过程，在其他情况下宜用 Sub 过程。

16.1.1 Function 过程与 Sub 过程的区别

Function 过程与 Sub 过程的差异比较大，主要体现在以下 3 个方面。

1. 调用方式

Function 过程可以通过代码调用，也可以在单元格中输入函数名称来调用，就和使用内置的工作表函数一样。但是 Function 过程不能使用按钮或者菜单调用，也不会出现在使用<Alt+F8>组合键打开的“宏”对话框中。

2. 是否返回值

Function 过程有返回值，不管在单元格中调用 Function 过程还是使用 VBA 调用 Function 过程都可直接获得其返回值，因此可以利用等号去获取 Function 过程的值。

Sub 过程没有返回值。

3. 放置位置

Sub 过程可以放在模块中，也可以放在工作表事件代码窗口中和 ThisWorkbook 中，但是 Function 过程只能放在模块中，否则无法在单元格中调用该过程。

如果在另一个过程中调用 Function 过程，那么 Function 过程的代码可以不放在模块中。

绝大多数 Function 过程都有参数，而绝大多数 Sub 过程都不带参数。

16.1.2 Function 过程的语法

Function 过程的语法如下：

```
[Public | Private | Friend] [Static] Function name [(arglist)] [As type]
```

仅从语法上讲，Function 过程仅比 Sub 过程的语法多一个“[As type]”。

“[As type]”的功能是为函数的返回值指定数据类型或者对象类型。

Sub 过程没有返回值，因此不需要指定其类型。

Function 过程的语法中比较重要的部分是 arglist，即自定义函数的参数。

arglist 部分的具体语法如下：

```
[Optional] [ByVal | ByRef] [ParamArray] varname[( )] [As type] [= defaultvalue]
```

在本书第 11 章已经讲述过如何在 Sub 过程中编写 Arglist，而 Function 过程中关于 Arglist 的用法完全一致，因此本节不再详细 Arglist 的规则和注意事件，仅通过一个简单的案例展示 Function 过程，以及 Function 过程的 Arglist 参数的设计思路。

某公司规定业务员的业绩计算规则如下：

如果业绩为大于等于 20 万元，那么奖金为 2000 元，如果业绩为 15 万元到 20 万元则奖金为 1800 元，如果业绩为 10 万元到 15 万元则奖金为 1500 元，如果业绩为 8 万元到 10 万元则奖金为 1000 元，如果业绩为 5 万元到 8 万元则奖金为 500 元，如果业绩为小于 5 万元则没有奖金。现要求开发一个函数来计算业务员的奖金。

根据本书前面 16 章的知识，以及本章的 Function 过程语法表可以得到以下代码：

```
'函数名称:Bonus,有一个必选参数 rng，函数的返回值为 Integer 型数值
'参数 rng 采用 long 型而不用 Range 型是为了避免只能使用单元格做参数
'用 Long 型表示可以直接作数字参数
Function Bonus(rng As Long) As Integer
'假设参数 rng 的值大于等于 200000,那么函数返回值为 2000,如果参数 rng 的值大于等于 150000,
那么函数返回值为 1800……
Bonus = IIf(rng >= 200000, 2000, IIf(rng >= 150000, 1800, IIf(rng >= 100000,
1500, IIf(rng >= 80000, 1000, IIf(rng >= 50000, 500, 0)))))
End Function
```

将以上代码放置在模块中，然后返回工作表界面。假设工作表中有如图 16.1 中 B 列所示的业绩数据，要计算每个业绩对应的奖金仅需在 C2 单元格中录入以下公式并填充即可：

```
=Bonus(B2)
```

C2 =Bonus(B2)

	A	B	C	D
1	姓名	业绩	奖金	
2	范亚桥	97923	1000	
3	陈强生	204309	2000	
4	姚达开	108735	1500	
5	钟正国	73458	500	
6	张珍华	99032	1000	
7	游有之	161826	1800	
8	曲华国	218649	2000	
9	陈越	201320	2000	
10	柳红英	187643	1800	
11	严西山	144418	1500	

图 16.1　根据业绩计算奖金

以上过程中函数名称是 Bonus，其返回值为 Integer 型。假设奖金包含小数，那么应改用 Currency 型。

如果使用内置函数，那么公式如下：

```
=IF(B2>=200000,2000,IF(B2>=150000,1800,IF(B2>=100000,1500,IF(B2>=80000,1000
,IF(B2>=50000,500,0)))))
```

很显然，公式“=Bonus(B2)”更简捷，而且因为简捷所以书写时能降低失误的可能性。

事实上，Function 过程并非只能在单元格中调用，还可在其他 Sub 过程中调用。例如：

```
Sub 计算奖金()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Dim rng As Range
  For Each rng In Range("B2:B" & Cells(Rows.Count, 2).End(xlUp).Row)
    rng.Offset(0, 1) = Bonus(rng.Value)
  Next rng
End Sub
```

本例文件参见光盘：..\第十六章\16-1 根据业绩计算奖金.xlsm

16.1.3 自定义函数的命名规则

自定义函数的名称和变量名称一样，需要遵循诸多规则，否则可能出错，也可能和其他过程产生冲突。定义函数的命名规则如下。

（1）不能以数字开头。

（2）不能包含标点符号。

（3）长度不能大于 255。

（4）多个函数不能同名，不管函数过程是否保存在同一模块内。

（5）不能与函数的参数同名。

（6）不能与同一模块内的变量和常量同名。

（7）不能使用 VBA 内部的保留字，例如 Function、Sub、Dim 等。

（8）不宜与工作表函数重名，也不宜与模块同名。

知识补充：工作表函数只能返回值，不能实现返回值以外的任何功能，例如修改单元格地址、关闭工作簿等。自定义函数可以实现返回值以外的功能，但是与返回值无关的功能更适合使用 Sub 过程。

16.2 开发不带参数的 Function 过程

Now、Today、Pi 等工作表函数都没有参数，利用 VBA 也可以开发没有参数的自定义函数。本节展示两个没有参数的自定义函数的开发与使用过程。

16.2.1 判断活动工作簿是否存在图形对象

案例要求：工作表中的图形对象可以隐藏起来，因此要人工判断活动工作簿的所有工作表中是否有图形对象是比较困难的。现要求利用自定义函数判断活动工作簿是否存在图形对象。

函数代码：

```
Function HasShapes() As Boolean'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Dim sht As Worksheet
  For Each sht In Worksheets
    HasShapes = sht.Shapes.Count
    If HasShapes = True Then Exit For
  Next sht
End Function
```

函数测试：

在任意单元格中录入公式：

```
=HasShapes()
```

公式的返回值是 True 表示工作簿中存在图形对象，反之则表示不存在图形对象，如图 16.2 所示。

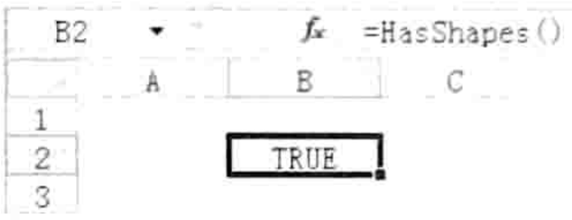


图 16.2 判断工作簿中是否有图形对象

思路分析：

本例的需求是判断有没有图形对象，因此返回值应该用布尔值，不宜用“有”和“没有”。

由于工作簿中可能有多个工作表，因此本例采用循环语句逐一将工作表中的图形对象数量赋值给函数 HasShapes，如果图形对象的数量大于 0，布尔型函数的返回值为 True，而不是具体的数值。同时又由于只要有一个工作表中存在图形对象就代表活动工作簿中有图形对象，所以对函数每赋值一次就要判断一次函数的值是否等于 True，如果是 True 则直接结束循环，不需要再计算其他工作表的图形对象数量，从而节约程序的执行时间。

点评：当工作簿中的图形对象过多时会影响工作簿的开启速度，以及增大文件体积，有时隐藏的图形对象很难让人察觉到它的存在，使用 VBA 代码计算图形对象数量时不会忽略隐藏的图形对象，因此本例的自定义函数是有存在价值的。

思考：请读者修改本例代码使其可以计算活动工作簿中的所有图形对象数量。

本例文件参见光盘：..\第十六章\ 16-2 判断活动工作簿中是否有图形对象.xlsm

16.2.2 计算公式所在单元格的页数

案例要求：在页眉与页脚中可以生成“第 X 页”的字样，现要求开发一个自定义函数，从而在任意单元格中都能计算当前页的页数。

函数代码：

```
Function Page()As Integer'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Dim 水平分页符 As Integer, 垂直分页符 As Integer, 当前页 As Integer, m As Integer,
n As Integer
水平分页符 = ActiveSheet.HPageBreaks.Count
垂直分页符 = ActiveSheet.VPageBreaks.Count
  For n = 1 To 水平分页符
    If ActiveSheet.HPageBreaks(n).Location.Row > Application.ThisCell.Row Then
Exit For
  Next
  For m = 1 To 垂直分页符
    If  ActiveSheet.VPageBreaks(m).Location.Column  >  Application.ThisCell.
Column Then Exit For
  Next
  If ActiveSheet.PageSetup.Order = xlOverThenDown Then
当前页 = (n - 1) * (垂直分页符 + 1) + m
  Else
当前页 = (m - 1) * (水平分页符 + 1) + n
  End If
  Page = 当前页
End Function
```

函数测试：

在工作表的任意单元格录入以下公式：

```
=Page()
```

公式的结果是公式所在单元格的页数，即打印工作表时该单元格出现在第几页中。

图 16.3 中有 4 个公式，由于 4 个单元格处在不同的页面中，因此计算结果为 4 个不同的值。

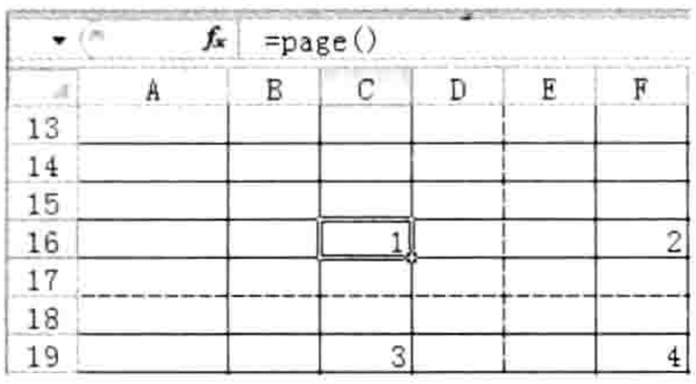


图 16.3 利用公式计算页数

思路分析：

HPageBreaks.Count 代表水平分页符的数量，HPageBreak.Location.Row 代表水平分页符的行号，Application.ThisCell.Row 代表公式在单元格中的行号，根据这 3 个已知条件足以计算公式所在单元格处于纵向第几页中。计算思路是使用循环语句遍历所有纵向分页符，然后逐一判断每个分页符的行号是否大于 ThisCell 单元格的行号，如果是则表示 ThisCell 刚好处于该页中，例如纵向第 4 个分页符的行号大于 ThisCell 的行号，那么说明 ThisCell 处于纵向第 4 页中。

利用相同的原理，由于 VPageBreaks.Count 则代表垂直分页符的数量，PageBreak.Location.Column 代表垂直分页符的列号，Application.ThisCell.Column 代表公式在单元格的列号，因此利用它们三者的关系可以判断公式所在单元格处于横向第几页中。

当横向、纵向的位置都计算出来后，公式所在单元格处于工作表中的第几页就可以顺理成章地获取到。不过需要注意一点，打印时有先行后列和先列后行之分，两种顺序对于页数的计算方式不同，因此需要在过程中加以判断。

知识补充： ThisCell 对象代表公式所在的单元格，只能用于 Function 过程中，而且只能在单元格中调用这个函数时才能正常工作。如果在 Sub 过程中利用代码调用该函数将会在执行过程中出错。

点评： 尽管进入分页预览模式后 Excel 会用灰色字体标示每一页的页数，但是不能将它打出来。使用自定义函数的优势在于灵活性好，公式放在哪个单元格就会在哪个单元格产生页数。

思考： 修改函数，使其可以计算公式所在工作表的总页数。

CD-ROM 本例文件参见光盘：..\第十六章\16-3 计算当前页数.xlsm

16.3 开发带有一个参数的 Function 过程

没有参数的函数功能相当单一，灵活性也极差。函数的参数越多则意味着函数的功能越强大，本节向读者展示带有一个参数的自定义函数的开发思路。

16.3.1 在不规则的合并单元格中执行合计

案例需求： 如图 16.4 所示的工作表是职工的销量统计表，由于不同销售人员之间销售的产品

数量不同，因此存放销售额的合并单元格也不规则，从而无法填充公式。现要求开发自定义函数突破这个限制，一次性将每个售货员的销售额统计出来。

函数代码：

```
'①代码存放位置：模块中②随书光盘中有代码含义注释
Function gather(rng As Range) As Double
  gather = WorksheetFunction.Sum(rng.Resize(Application.ThisCell.MergeArea.
Rows.Count, 1))
End Function
```

函数测试：

选择 D2:D12 区域，然后录入以下公式，并按<Ctrl+Enter>组合键，公式会瞬间将所有售货员的销售额统计出来，统计结果如图 16.5 所示。

```
=gather(C2)
```

	A	B	C	D
1	姓名	产品	销售额	合计
2	罗生荣	镙丝	4884	
3		扳手	3774	
4		梅花刀	5550	
5	胡华	梅花刀	9324	
6		扳手	4440	
7	钟小月	镙丝	14652	
8	梁爱国	扳手	7548	
9		改锥	8658	
10	周长传	电笔	4662	
11		老虎钳	1998	
12		一字刀	888	

图 16.4　销量统计表

D2　=gather(C2)

	A	B	C	D
1	姓名	产品	销售额	合计
2	罗生荣	镙丝	4884	14208
3		扳手	3774	
4		梅花刀	5550	
5	胡华	梅花刀	9324	13764
6		扳手	4440	
7	钟小月	镙丝	14652	14652
8	梁爱国	扳手	7548	16206
9		改锥	8658	
10	周长传	电笔	4662	7548
11		老虎钳	1998	
12		一字刀	888	

图 16.5　在不规则的合并单元格中合计销量

思路分析：

公式所在的单元格处于合并状态，它的行数决定了左边被合计的区域的行数，因此代码中使用 Range.Resize 属性将参数 rng 重置为与公式所在区域的相同行高，然后再使用工作表函数 Sum 对它求和。

点评：如果不使用自定义函数，仅使用 Excel 的内置函数很难完成本例需求，其难点有两个——不能填充公式、待求和的区域的单元格数量不同。本例代码的突破方式是只引用单列待求和的单元格，然后在代码中使用 Range.Resize 属性重置区域的高度。

思考：将本例的求和区域由单列修改为两列，用如图 16.6 所示的方式存放数据，而且必须先将它们求积再汇总，那么该如何修改代码呢？

E2　=gather(C2,D2)

	A	B	C	D	E
1	姓名	产品	数量	单价	合计
2	罗生荣	镙丝	4884	0.7	106427
3		扳手	3774	17	
4		梅花刀	5550	7	
5	胡华	梅花刀	9324	7	140748
6		扳手	4440	17	
7	钟小月	镙丝	14652	0.7	10256.4
8	梁爱国	扳手	7548	17	232212
9		改锥	8658	12	
10	周长传	电笔	4662	8	73260
11		老虎钳	1998	14	
12		一字刀	888	9	

图 16.6　双条件合计

本例文件参见光盘：..\第十六章\16-4 根据合并单元格合计数据.xlsm

16.3.2 建立活动工作簿的表目录

案例要求：为活动工作簿创建表目录，并且当单击表目录中的名称时可以跳转到该表中。

函数代码：

```
'①代码存放位置：模块中②随书光盘中有代码含义注释
Function 工作表(Optional 序号 As Integer) As String
  If IsMissing(序号) Then 序号 = ActiveSheet.Index
  If 序号> Sheets.Count or 序号<1 Then
    工作表 = ""
  Else
    工作表 = Sheets(序号).Name
  End If
End Function
```

函数测试：

在工作表的A1单元格中录入以下公式，然后向下填充即可获取工作簿中所有工作表的名称，效果如图16.7所示。

```
=工作表(ROW(A1))
```

由于函数“工作表”只能提取工作表名称，不能产生链接功能，因此在公式中套入HYPERLINK函数后才算是创建了真正的工作表目录。以下公式可以显示如图16.8所示的结果。

```
=HYPERLINK("#"&工作表(ROW(A1))&"!A1",工作表(ROW(A1)))
```

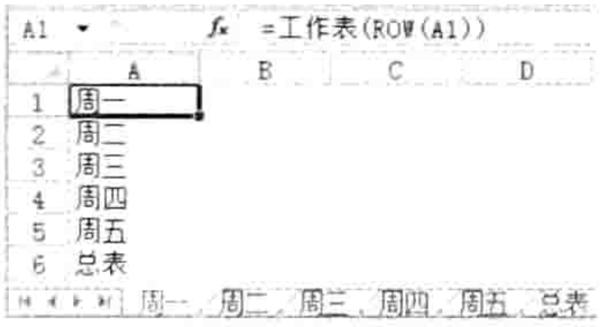


图16.7 提取所有工作表名称

图16.8 创建工作表目录

思路分析：

本例函数参数“序号”是一个可选参数，当忽略参数时其默认值为活动工作表的序号。由于“ActiveSheet.Index”涉及对象，按照VBA的规则对参数指定默认值时只能使用常数，因此本例声明函数名称与参数名称时没有指定其默认值，而是在过程中用IsMissing函数判断参数“序号”是否已经被赋值，如果返回值为False则将ActiveSheet.Index赋值给参数。

参数“序号”是一个数值，用于指定函数的返回值是第几个工作表的名称，因此参数的值不能小于0、不能大于工作簿的所有表的总量。

点评：Excel本身没有获取工作表名称的函数，虽然调用宏表函数并借助名称可以完成，但公式相当长，并且必须要借助名称，公式无法在单元格中直接套用。本自定义函数以“Row(a1)”作为参数可以逐一提取所有表名，再配合HYPERLINK建立链接，在使用上相当方便。

当工作簿中有数十个或上百个表时，创建目录能大大方便工作，提升定位工作表的速度。

思考：本例的自定义函数可以取得所有表的名称，请读者修改代码使其只能获取工作表的名称。

本例文件参见光盘：..\第十六章\16-5 创建工作表目录.xlsm

16.4 开发带有两个参数的 Function 过程

带有两个参数的函数通常是其中一个用于引用区域，另一个用于指定操作方式。本节通过两个案例演示带有两个参数的函数的开发过程，以及思路分析。

16.4.1 分段提取数值

案例需求：图 16.9 中 B 列是每个售货员的销售产品与销售额记录，由于多个数值在同一个单元格中无法执行合计，因此要求利用自定义函数将所有数字分段提取出来。

	A	B
1	姓名	销售额
2	孙二兴	镙丝455扳手228.5梅花刀427.8
3	赵兴文	一字刀220.5六角镙丝78
4	朱丽华	梅花刀200扳手78压线钳200

图 16.9 销量统计表

函数代码：

```
'①代码存放位置：模块中②随书光盘中有代码含义注释
Function Separate(rng As Range, Optional Count As Byte = 1)
  If rng.Count > 1 Then Separate = "": Exit Function
  With CreateObject("VBSCRIPT.REGEXP")
    .Global = True
    .Pattern = "\d+\.?\d+?"
    If Count < 1 Or Count > .Execute(rng).Count Then
      Separate = ""
    Else
      Separate = .Execute(rng)(Count - 1)
    End If
  End With
End Function
```

函数测试：

在 B2 单元格录入以下公式，然后将公式向右填充到 F 列，再向下填充到第 4 行。如图 16.10 所示是公式的计算结果。

```
=Separate($B2,COLUMN(A1))
```

C2 =Separate($B2,COLUMN(A1))

	A	B	C	D	E
1	姓名	销售额			
2	孙二兴	镙丝455扳手228.5梅花刀427.8	455	228.5	427.8
3	赵兴文	一字刀220.5六角镙丝78	220.5	78	
4	朱丽华	梅花刀200扳手78压线钳200	200	78	200

图 16.10 分列结果

思路分析：

从杂乱的字符串中识别数字，这是正则表达式所擅长的工作。本例中正则表达式的搜索条件是“\d+\.?\d+?”，其中“\d”代位单位数字，而“\d+”则代表至少一位数字。“\.?”表示一个或者 0 个小数点，此处的点不代表重复前一个条件，因为它前面有转义符“\”。最后的“d+?”表示以任意个数字结尾，采用惰性匹配。

函数的 Count 参数表示从所有匹配结果中提取第几个值作为函数的最终结果，它的值不能小于 0 或者大于区配结果的总数量，否则计算结果将是错误值“#VALUE!”。对于自定义函数的防错不再是使用 On Error Resume Next 语句，而是使用条件语句 If Then。

关于正则表达式的更多规则请参考本书第 14 章。

点评：Excel 最擅长的是数值运算，最不擅长的是处理复杂的字符串，不过由于 VBA 可以调用正则表达式的资源，因此用 VBA 开发的自定义函数就可以完成很多 Excel 原本无法完成的工作。

思考：修改本例函数，使其具有对字符串中的所有数值求和的功能。

本例文件参见光盘：..\第十六章\16-6 分段提取数值.xlsm

16.4.2 获取最大值、最小值或众数的地址

案例需求：Excel 内置函数可以计算出一个区域的最大值、最小值和众数，但是不能计算出它们的地址，并且不能单击定位这些单元格。现要求通过自定义函数找出指定区域中的最大值、最小值和众数的地址，而且配合 HYPERLINK 函数定位单元格。

函数代码：

```
'①代码存放位置：模块中②随书光盘中有代码含义注释
Function 极值(rngg As Range, Optional Style As String = "大") As String
  Dim i As Integer, rng As Range, TargetRng As Range, FirstAdd As String,
TargetVal As Double
  If Style = "大" Then
    TargetVal = WorksheetFunction.Max(rngg)
  ElseIf Style = "小" Then
    TargetVal = WorksheetFunction.Min(rngg)
  ElseIf Style = "众" Then
    TargetVal = WorksheetFunction.Mode(rngg)
  Else
    极值 = ""
    Exit Function
  End If
  Set rng = rngg.Find(TargetVal, , , xlWhole)
  If Not rng Is Nothing Then
    FirstAdd = rng.Address
    Do
      i = i + 1
      Set rng = rngg.Find(TargetVal, rng, , xlWhole)
      If i = 1 Then Set TargetRng = rng Else Set TargetRng = Union(TargetRng,
rng)
      If rng.Address = FirstAdd Then 极值 = TargetRng.Address(0, 0): Exit
Function
    Loop
  Else
    极值 = ""
  End If
End Function
```

函数测试：

在 B2 单元格中录入以下公式，将得到 B2:C10 区域中所有最大值的地址，效果如图 16.11 所示。此公式省略了第 2 参数，表示提取所有最大值的地址。B2:C10 区域中最大值是 100。

```
=极值(B2:C10)
```

继续在 E5 单元格中录入以下公式，公式会在单元格中显示“定位众数”，而单击公式则可以定位 B2:C10 区域中所有众数所在的单元格，效果如图 16.12 所示。

```
=HYPERLINK("#"&极值(B2:C10,"众"),"定位众数")
```

E2 =极值(B2:C10)

	A	B	C	D	E
1	姓名	语文	数学		最大值地址
2	李文新	64	68		C6, B8, B5
3	朱华青	80	80		
4	周文明	56	82		
5	徐金明	100	53		
6	陈丽丽	59	100		
7	钱单文	72	80		
8	鲁华美	100	75		
9	梁兴	49	80		
10	曲华国	69	49		

图 16.11　获取最大值的地址

	A	B	C	D	E
1	姓名	语文	数学		最大值地址
2	李文新	64	68		C6, B8, B5
3	朱华青	80	80		
4	周文明	56	82		定位极值
5	徐金明	100	53		定位众数
6	陈丽丽	59	100		
7	钱单文	72	80		
8	鲁华美	100	75		
9	梁兴	49	80		
10	曲华国	69	49		

图 16.12　单击公式定位所有众数

知识补充：所谓众数是指出现次数最多的数。在 B2:C10 区域中数值 80 出现了 4 次，它的出现次数最“众”，因此单击第二个公式所在的单元格可以定位所有值为 80 的单元格。

思路分析：

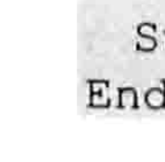
本例的函数有两个参数，其中第 1 参数用于指定区域，第 2 参数用于指定取值标准，它的可选项包含“大”、“中”和“众”三个值。通常在工作中计算最大值的需求偏多，因此通过 Optional 语句将第 2 参数转换成可选参数，默认值为“大”。

获取某个值的地址重点在于 Do Loop 循环和 Range.Find 方法的搭配使用，找到目标后使用 Range.Address 属性获取单元格的地址，在本书前面的章节中有许多类似的案例，与本例的差异仅仅在于少了一个可选参数而已。

本例中的函数只能取得目标单元格的地址，无法生成该地址对应的单元格的超链接，因此在使用函数“极值”时还需要配合使用 HYPERLINK 函数。

点评：通过 Excel 的查找与定位工具可以选中符合某些条件的单元格，这些条件中不包含最大值、最小值和众数，利用 VBA 可以弥补这个缺陷。

思考：修改本例代码，使函数“极值”可以获取所有等于区域内的平均值的单元格地址，如果没有单元格中的值等于平均值，那么取最接近的那一个值所在单元格地址。



本例文件参见光盘：..\第十六章\16-7 获取极值的地址.xlsm

16.4.3　汇总前 N 大值

案例需求：利用函数统计区域中前 N 大值的合计，默认取前 3 大值的合计。

函数代码：

```
'①代码存放位置：模块中②随书光盘中有代码含义注释
Function SumTop3(Rng As Range, Optional Count As Integer = 3)
  Dim ValCount As Long
  ValCount = WorksheetFunction.Count(Rng)
  If ValCount = 0 Then SumTop3 = 0: Exit Function
  If Count < 0 Then SumTop3 = 0: Exit Function
  If Count > ValCount Then Count = ValCount
  SumTop3 = Evaluate("Sum(large(" & Rng.Address & ",row(1:" & Count & ")))")
End Function
```

函数测试：

假设要计算 B2:C10 区域中前 3 大值的合计，那么可在 F1 单元格录入以下公式：

```
=SumTop3(B2:C10)
```

如果由 E1 单元格录入的数值决定计算求和范围，那么应在 F1 单元格中录入以下公式：

```
=SumTop3(B2:C10,E1)
```

如图 16.13 和图 16.14 所示的分别是计算前 3 大值和前 5 大值的统计结果。

=SumTop3(B2:C10)

	A	B	C	D	E	F
1	姓名	语文	数学		前3大值合计	300
2	李文新	64	68			
3	朱华青	80	80			
4	周文明	56	82			
5	徐金明	100	53			
6	陈丽丽	59	100			
7	钱单文	72	80			
8	鲁华美	100	75			
9	梁兴	49	80			
10	曲华国	69	49			

图 16.13 统计前 3 大值的合计

F1 =SumTop3(B2:C10,E1)

	A	B	C	D	E	F
1	姓名	语文	数学		5	462
2	李文新	64	68			
3	朱华青	80	80			
4	周文明	56	82			
5	徐金明	100	53			
6	陈丽丽	59	100			
7	钱单文	72	80			
8	鲁华美	100	75			
9	梁兴	49	80			
10	曲华国	69	49			

图 16.14 统计前 5 大值的合计

思路分析：

在本例过程代码中最重要也是最难的地方是作为 Evaluate 方法的参数的那个表达式，此表达式的功能是生成一个用于汇总前 N 大值的公式，然后 Evaluate 方法将该公式转换成值并赋值给函数。此表达式的计算过程如下：

",row(1:" & Count & ")"——表达式生成的公式用于创建 1 到 count 的数组；

Large(" & Rng.Address & ",row(1:" & Count & "))——表达式生成的公式能分别计算 rng 区域中第 1 大值、第 2 大值……第 Count 大值，从而生成一个数组；

Sum(large(" & Rng.Address & ",row(1:" & Count & ")))——利用 Sum 函数统计 Large 函数生成的数组，从而得到前 N 大值的合计。

点评：如果不使用本例的自定义函数，那么只能通过以下数组公式完成：

```
=SUM(LARGE(B2:C10,ROW(INDIRECT("1:"&E1))))
```

很显然，自定义函数可以让公式变得简单，降低学习成本（针对终端用户而言）。

思考：

修改本例代码，使函数可以计算前 N 个最小值。

CD-ROM 本例文件参见光盘：..\第十六章\16-8 对前 N 大值求和.xlsm

16.5 开发复杂的 Function 过程

开发自定义函数的重点在于如何设计参数，参数越多则函数过程的设计越复杂。本节展示 4 个复杂函数的开发过程。

16.5.1 合并区域的值或者数组

案例要求：将区域中的所有数据或者数组中的所有数据合并成一个字符串。

函数代码：

```
'①代码存放位置：模块中②随书光盘中有代码含义注释
Function Connect(Delimiter As String, ParamArray Rng() As Variant) As String
    Dim cell As Range, i As Integer, item As Variant
    For i = 0 To UBound(Rng)
      If Not IsMissing(Rng(i)) Then
        Select Case TypeName(Rng(i))
        Case "Range"
          For Each cell In Intersect(Rng(i), Rng(i).Parent.UsedRange)
            Connect = Connect & cell & Delimiter
          Next cell
        Case "Variant()"
          For Each item In Rng(i)
            If item <> False Then Connect = Connect & item & Delimiter
          Next item
        Case Else
          Connect = Connect & Rng(i) & Delimiter
        End Select
      End If
    Next i
    If Len(Connect) > 2 Then Connect = Left(Connect, Len(Connect) -
Len(Delimiter))
End Function
```

函数测试：

Connect 函数可以合并三类数据：一是区域，二是数组，三是直接在参数中指定的字符串。假设工作表中有如图 16.15 中 A1:C1 区域所示的数据，要将它们与“广场”二字合并为一个字符串，并且用“，”作分隔符，那么可在 D1 单元格录入以下公式。

公式结果是“湖北,武汉,东西湖区,广场”。

```
=Connect(",",A1:C1,"广场")
```

对于如图 16.16 所示的成绩表，假设要合并其中成绩大于 90 的学生姓名，那么可用以下数组公式：

```
=Connect(",",IF(B2:B11>90,A2:A11))
```

由于此公式是数组公式，必须在录入公式后按<Ctrl+Shift+Enter>组合键，最终效果如图 16.16 所示。

D1 =Connect(",",A1:C1,"广场")

	A	B	C	D
1	湖北	武汉	东西湖区	湖北、武汉、东西湖区、广场

图 16.15　合并区域的值

D2 {=Connect(",",IF(B2:B11>90,A2:A11))}

	A	B	C	D	E
1	姓名	成绩		90分以上的学生姓名	
2	周少强	97		周少强,姚达开,陈真亮	
3	黄真真	70			
4	仇正风	69			
5	周辉煌	86			
6	姚达开	100			
7	谢有金	64			
8	陈金来	50			
9	徐大鹏	80			
10	陈真亮	96			
11	古至龙	81			

图 16.16　合并内存数组

思路分析：

本例的函数 Connect 支持 1～255 个参数，其中第 2 至第 255 个参数是可选参数，因此在声明函数的参数时需要在第 2 参数前添加 ParamArray，该语句的功能是将必选参数转换成不确定数量的可选参数。参数的具体数量由 255 减去其他参数的个数决定。

本函数的第 2 参数 Rng 是变体型，因此它可能是数组，可能是区域，也可能是直接录入到参数中的文本，在执行数据合并前有必要使用 Typaname 函数判断它的类型，其中数组的类别名称是“Variant()”，区域的类别名称是“Range”，其他的都属于直接录入在参数的文本或数值。

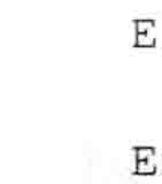
对于数组和区域，需要使用循环语句逐一合并数据，对于数值和文本型的参数则直接合并，在后面追加分隔符即可。

当合并完成后，最右边的分隔符需要删除。删除最后一个分隔符的思路是利用 Len 函数计算 Connect 的长度，再计算分隔符的长度，两者的差值即为要保留的字符。

代码中的“Intersect(Rng(i), Rng(i).Parent.UsedRange)”表示只取 Rng(i)区域与它所在工作表的已用区域的交集，避免用户使用“A:Z”这类区域作参数时浪费循环语句的执行时间，也可以理解为使用该语句排除 Rng(i)中的空白区域。

点评： Excel 自带一个连接文本的函数——CONCATENATE 和一个运算符&，然而它们共同的缺点都是不能批量合并单元格和数组，而且合并时不能随意指定分隔符，而自定义函数可以突破这两个限制从而完成更复杂的工作，这正是本函数的亮点。

思考： 本例的代码并没有防范空值所带来的分隔符连续出现的问题，例如合并 A1:A3 单元格时，只有 A3 单元格有值并且值为 3，那么公式“=Connect(",",A1:A3)”的结果是“，，3”。请读者修改代码避免此问题。



本例文件参见光盘：..\第十六章\16-10 合并区域或数组的值.xlsm

16.5.2 按单元格背景颜色进行条件求和

案例需求： SumIF 函数可以按数据的值或者按范围进行条件求和，现要求参照单元格的颜色对区域进行条件求和。

函数代码：

```
'①代码存放位置：模块中②随书光盘中有代码含义注释
Function SumIFColor(条件区 As Range, 颜色单元格 As Range, Optional 统计区)
  Dim arr(), Item, i As Long
  If IsMissing(统计区) Then
    arr = Intersect(条件区, 条件区.Parent.UsedRange).Value
  Else
    arr = 统计区(1).Resize(条件区.Rows.Count, 条件区.Columns.Count).Value
  End If
  For Each Item In arr
    i = i + 1
    If 条件区.Cells(i).Interior.Color = 颜色单元格(1).Interior.Color Then
      SumIFColor = SumIFColor + Item
    End If
  Next
End Function
```

函数测试：

图 16.17 中 A 列是姓名、B 列是产量，B 列的部分单元格有颜色标示，如果要对黄色单元格中的数值进行求和，那么可用以下公式：

```
=SumIFColor(B2:B10,B3)
```

由于条件区和统计区是重叠的，因此以上公式忽略了第3参数。

如果条件区与统计区不重叠，如图16.18所示，那么统计黄色的姓名所对应的产量应用以下公式：

```
=SumIFColor(A2:A10,A3,B2)
```

C2 =SumIFColor(B2:B10,B3)

	A	B	C	D
1	姓名	产里	黄色单元格数据之和	
2	柳龙云	85	345	
3	朱华青	77		
4	赵光文	79		
5	张居正	64		
6	朱未来	65		
7	计尚云	89		
8	张开来	50		
9	古至龙	88		
10	曹值军	91		

图16.17　对黄色的产量求和

C2 =SumIFColor(A2:A10,A3,B2)

	A	B	C	D
1	姓名	产里	黄色单元格对应的产里求和	
2	柳龙云	85	345	
3	朱华青	77		
4	赵光文	79		
5	张居正	64		
6	朱未来	65		
7	计尚云	89		
8	张开来	50		
9	古至龙	88		
10	曹值军	91		

图16.18　对黄色的姓名对应的产量求和

思路分析：

本例代码的目的是以第2参数的单元格的背景色作为参照颜色，然后去第1参数所指定的区域中一一比较，如果相同则将第3参数中对应的单元格的值累加起来，整个过程相当简单，难点在于将第3参数设计成类似SumIF函数中的可选参数。

SumIF函数的第3参数有两个特点，一是忽略参数时则对第1参求和，二是第3参数允许只引用单个单元格，函数自动将它扩展为第1参数的高度和宽度后再计算。本例的函数完全效仿了SumIF函数的这个特点。

实现第一个特点的步骤有两个，其一是声明函数的参数时使用Optional语句将它转换成可选参数，其二是利用IsMissing函数判断用户是否已对参数赋值，如果没有赋值则将第1参数“条件区”当作第3参数“统计区”去参与运算。

实现第二个特点的方法是利用Range.Resize方法重置参数“统计区”的高度和宽度，使其与参数“条件区”一致。

点评：Excel的400多个内置函数都无法实现按颜色求和，本例的SumIFColor函数不仅可以对区域按颜色求和还可以使作为颜色参考的条件区与实际参与求和的统计区脱离，从而使函数的应用面更广，统计方式更灵活。

思考：

修改本例的代码，使函数可以按字体颜色求和。

本例文件参见光盘：..\第十六章\16-11 按颜色对区域进行条件求和.xlsm

16.5.3　按颜色查找并返回数组

案例需求：在区域的最左列中查找指定的颜色，然后返回其右边若干列的所有数据，公式的结果必须是数组。

函数代码：

```
'①代码存放位置：模块中②随书光盘中有代码含义注释
Function VlookupCol(查找值 As Range, 查找区域 As Range, Optional 列数 As Byte = 2)
  Dim Col As Long, cell As Range, arr(), i As Byte
  Col = 查找值.Interior.Color
  For Each cell In Intersect(查找区域.Columns(1), 查找区域.Parent.UsedRange)
```

```
    If cell.Interior.Color = Col Then
      i = i + 1
      ReDim Preserve arr(1 To i)
      arr(i) = cell.Offset(0, 列数 - 1)
    End If
  Next cell
  VlookupCol = WorksheetFunction.Transpose(arr)
End Function
```

函数测试：

在如图 16.19 所示的工作表中，A 列的姓名以不同背景颜色进行区分，在 E1 单元格中有参考颜色，在 E2 单元格中录入以下公式可以获得第 2 个符合条件的目标数值 79。

```
=INDEX(VlookupCol(E1,A2:B11),2)
```

由于 VlookupCol 函数的第 3 参数默认值是 2，因此以上公式中忽略了第 3 参数的值。

如果成绩在第 3 列，而且要求一次性将所有结果都罗列出来，那么应该选择 B2:B11 区域后再录入以下数组公式，接着按<Ctrl+Shift+Enter>组合键结束，公式计算结果如图 16.20 所示。

```
=VlookupCol(F1,A2:C11,3)
```

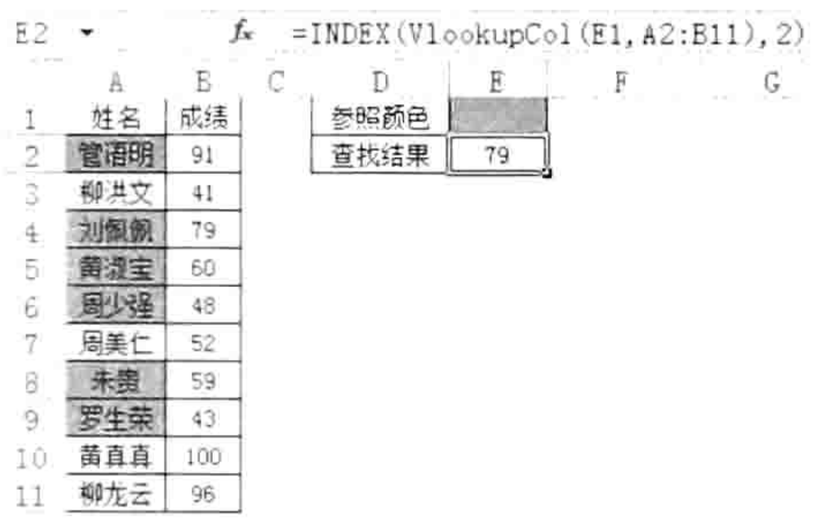


图 16.19 按颜色获取第 2 个目标值

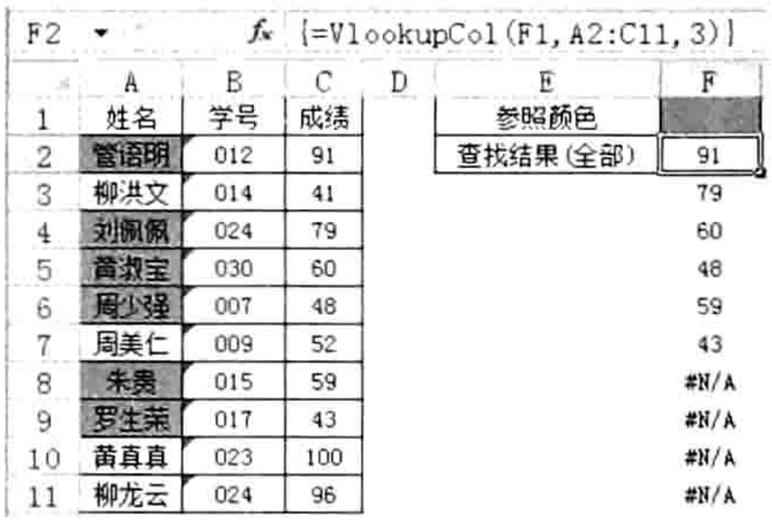


图 16.20 按颜色获取所有目标值

事实上本例还可以有很多延伸用法，例如按颜色查找并求和、按颜色查找并求平均值或者按颜色查找并计数等。

用以下公式可以找到所有符合颜色条件的数据，然后计算它们的平均值，结果如图 16.21 所示。

```
=AVERAGE(VlookupCol(查找全部!F1,查找全部!A1:C11,3))
```

图 16.21 利用普通公式返回所有结果并排除错值

以下公式可以找到所有符合颜色条件的数据，然后计算它们的合计：

```
=SUM(VlookupCol(查找全部!F1,查找全部!A1:C11,3))
```

以下公式则表示 A1:A11 区域中符合指定颜色的单元格数量：

```
=COUNTA(VlookupCol(查找全部!F1,查找全部!A1:C11,3))
```

思路分析：

工作表函数 Vlookup 可以在单元格左边一列查找，返回第一个符合条件的右边某列中的值，它的查找条件是数值、只能返回第一个符合条件的值。VBA 的功能极其强大，利用 VBA 开发函数时可以不再受这些功能限制，因此本例将查找条件设置为单元格的背景颜色，从而弥补内置函数的不足之处，同时将返回结果设置为数组，从而将所有符合条件的结果都罗列出来。

要让自定义函数的结果是一个数组需要两个步骤，其一是声明函数时将它声明为变体，由于函数的默认类型就是变体型，因此不声明类型即可。其二是在函数过程中声明一个数组变量，然后通过循环语句将所有目标值写入数组变量中，最后将这个变量赋值给函数即可。

点评：本函数可以实现按颜色进行查找，是 Vlookup 函数的补充。另外对于会用 Vlookup 函数的读者一定有一个心得：Vlookup 只能返回一个符合条件的目标值，尽管使用其他函数嵌套也可以查找出所有目标值，但公式比较长，也不利于理解和学习。

本例的函数可以弥补 Vlookup 函数的不足，同时也告诉我们一个道理——自己开发函数时应尽量考虑全面，为函数提供更多的可选项，使函数更灵活，适应更多的需求。

思考：修改函数过程的代码，使其可以在最上面一列中查找，返回查找结果对应的下方某一列的值——本例的函数是在单元格左边一列中查找，返回查找结果对应的右方单元格中的某一列的值。

本例文件参见光盘：..\第十六章\16-12 按颜色从左向右查找所有数据.xlsm

16.5.4 合计分隔符左边的所有数值

案例需求：字符串中有多串数值，现要求将某个字符之前的所有数值求和，忽略其他数值。例如“3 套教材 200 元”，仅统计“元”之前的 200，忽略数值 3。

函数代码：

```
'①代码存放位置：模块中②随书光盘中有代码含义注释
Function 合计(区域 As Range, 分隔符 As String) As Double
  Dim cell As Range, Item
  With CreateObject("VBSCRIPT.REGEXP")
    For Each cell In 区域
      .Pattern = "\-?\d+(\.\d+)?(?=" &分隔符& ")"
      .Global = True
      If .test(cell) Then
        For Each Item In .Execute(cell)
合计 = 合计 + Item
        Next
      End If
    Next cell
  End With
End Function
```

函数测试：

图 16.22 中的 A1:A2 区域是购物记录，包含产品名称、数量和金额，如果只计算其中的金额合计，则应用以下公式：

```
=合计(A1:A2,"元")
```

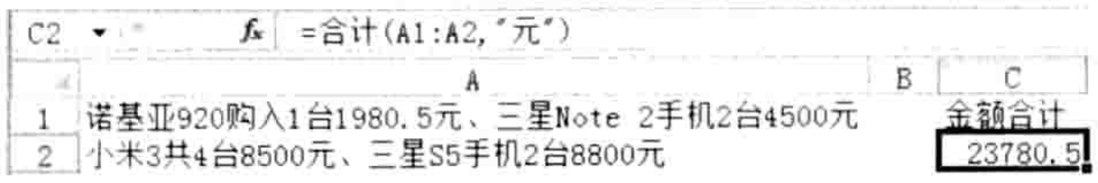


图 16.22 金额合计

如果要计算台数合计则应用以下公式，效果如图 16.23 所示。

```
=合计(A1:A2,"台")
```

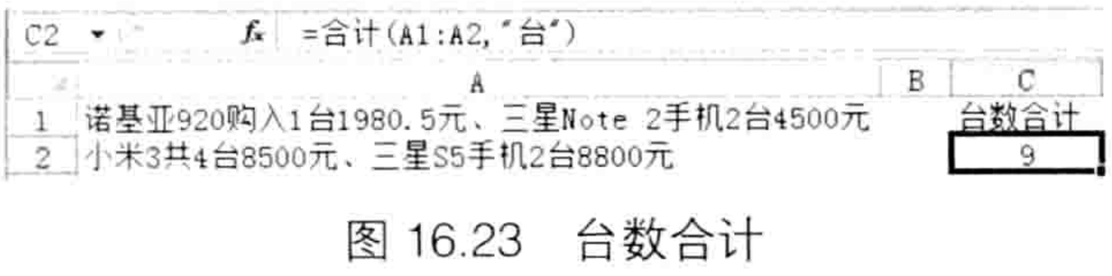


图 16.23 台数合计

思路分析：

本例使用正则表达式逐一提取单元格中以参数“分隔符”结尾的所有数值，通过循环语句将它汇总，接着在外面套一层循环语句对区域汇总。其中唯一的难点在于正则表达式的搜索条件。

本例代码中的“\-?\d+(\.\d+)?(?=" &分隔符& ")”是搜索条件，可以将它分成五段理解：

“\-?”——整体的含义是可选的负号。由于“-”是元字符，因此在前面添加转义符“\”。而问号则表示“-”是可选的，不管是否有“-”都可以匹配成功。

“\d”——代表 0～9 的任意数字。

“(\.\d+)?”——代表小数点加 0～9 的任意数字，而且是可选的。其中问号用于限制小数点和数字这个整体，因此要在问号前面添加小括号。

“(?=" &分隔符& ")”——代表匹配对象是以变量“分隔符”结尾的字符串，但是“分隔符”所指定的字符并不参与匹配，仅仅参与搜索，这是正则表达式中正向预查的一种手法。

关于正则表达式的更多规则请参考本书第 14 章。

点评：正则表达式的文字处理能力极其强大，在处理杂乱的字符串时正则表达式可以让工作更高效、快捷，不过对于制表而言，前期的设计更重要，不要将太多的精力放在后期的调整上。例如本例中的字符串在前期录入时就应该分放在不同的单元格中。

思考：修改本例的代码，使函数可以计算分隔符右边的数值。

本例文件参见光盘：..\第十六章\16-13 合计分隔符左边的所有数值.xlsm

16.6 编写函数帮助

用户自定义的函数不管是给自己使用还是给其他用户使用，都有必要对函数的功能和参数添加说明，使用户在使用时更加便捷。不仅如此，还有必要对函数进行分类，例如小写转大写函数应该划入财务函数类，按颜色求和函数应该划入统计函数类……

16.6.1 MacroOptions 方法的语法

当工作簿中有自定义函数时，打开“插入函数”向导后可以发现所有自定义函数的类别都是“用户定义”，而且没有函数的功能介绍也没有参数说明，这无疑增大了学习成本。Excel 2010 的 Application.MacroOptions 方法可以解决这个问题，它的语法如下：

```
Application.MacroOptions(Macro, Description, HasMenu, MenuText, HasShortcutKey,
ShortcutKey, Category, StatusBar, HelpContextID, HelpFile)
```

各参数的介绍如表 16-1 所示。

表 16-1　MacroOptions参数说明

参数名称	功能说明
Macro	宏的名称或用户定义函数的名称
Description	功能描述
HasMenu	忽略该参数
MenuText	忽略该参数
HasShortcutKey	如果赋值为 True，则为宏指定一个快捷键（还必须指定 ShortcutKey）。如果该参数为 False，则不为宏指定快捷键，默认值为 False
ShortcutKey	如果 HasShortcutKey 为 True，则本参数为必选参数，否则本参数为可选参数。参数的功能是指定快捷键
Category	为自定义函数分配函数类别
StatusBar	宏的状态栏文本
HelpContextID	指定宏的帮助主题的 ID，是一个整数
HelpFile	包含 HelpContextId 参数定义的帮助主题的帮助文件名称
ArgumentDescriptions	指定函数的参数的功能说明，可以同时为多个参数添加说明

Application.MacroOptions 方法的所有参数都是可选参数，它的 11 个参数中只有 4 个参数比较重要，包括 Macro、Description、Category 和 ArgumentDescriptions。其中 Macro 代表函数名称，Description 代表函数的功能描述，Category 用于指定函数的类型，ArgumentDescriptions 用于为函数的参数添加说明。

16.6.2　为函数分类及添加说明

每一个自定义函数都应该分类、添加功能说明，以及添加参数说明，从而提升函数的易用性，以及节约学习成本。

以本章 16.5 节中的自定义函数 SumIFColor 为例，将它分配在“数学与三角函数”类别中，而且指定函数的功能与参数说明，完整代码如下：

```
Sub auto_open()'①代码存放位置：模块中②随书光盘中有代码含义注释
  Application.MacroOptions Macro:="SumIFColor", Description:="在第 1 参数指定的区域中查找第 2 参的背景颜色，找到后将对应于第 3 参数相同位置的数值汇总。" + Chr(10) + "如果忽略了第3参数则用第1参数参与汇总。", Category:="数学与三角函数", ArgumentDescriptions:=Array("在此区域中查找，查找条件是单元格的背景颜色", "要查找的颜色来自此函数所指定的单元格", "要统计数值的区域，如果忽略此参数则用第 1 参数参与统计")
End Sub
```

将以上过程与函数 SumIFColor 的代码放在同一个模块中，然后执行过程“Auto_open”，并打开“插入函数”向导，在“数学与三角函数”类别中可以找到自定义函数 SumIFColor，效果如图 16.24 所示。

打开“函数参数”对话框后可以看到函数的功能描述，以及每个参数的功能描述。图 16.25 中光标定位于第 1 参数的“条件区”中，因此在参数下方显示了参数“条件区”的描述信息。

以上过程中有两点需要特别说明，其一是为 Category 参数赋值时可以用内置的函数类别名称也可以使用类别编号，表 16-2 中罗列了所有内置的类别编号。

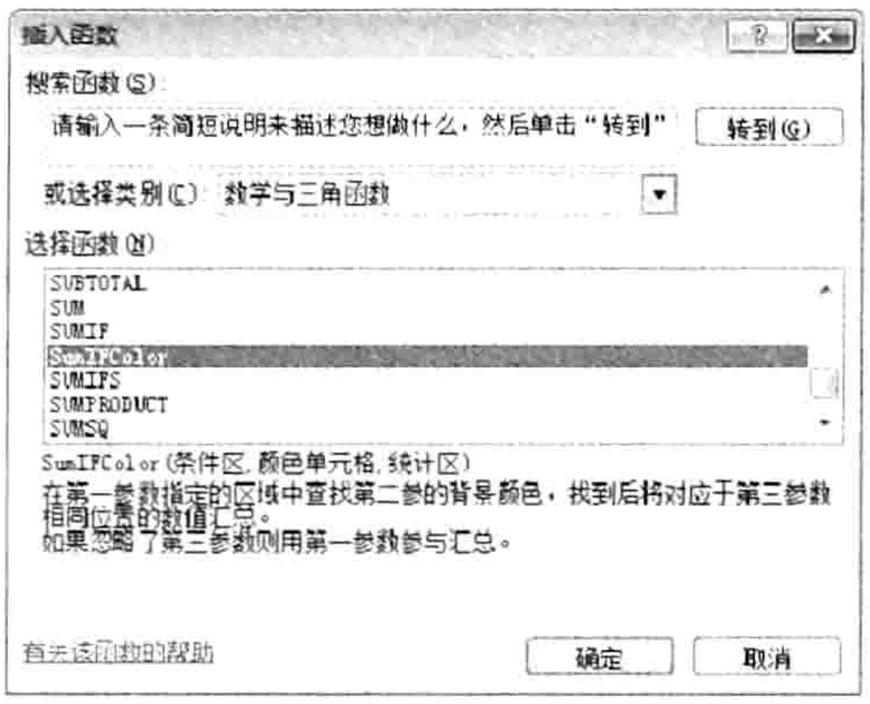


图 16.24 在内置的函数类别中查找自定义函数

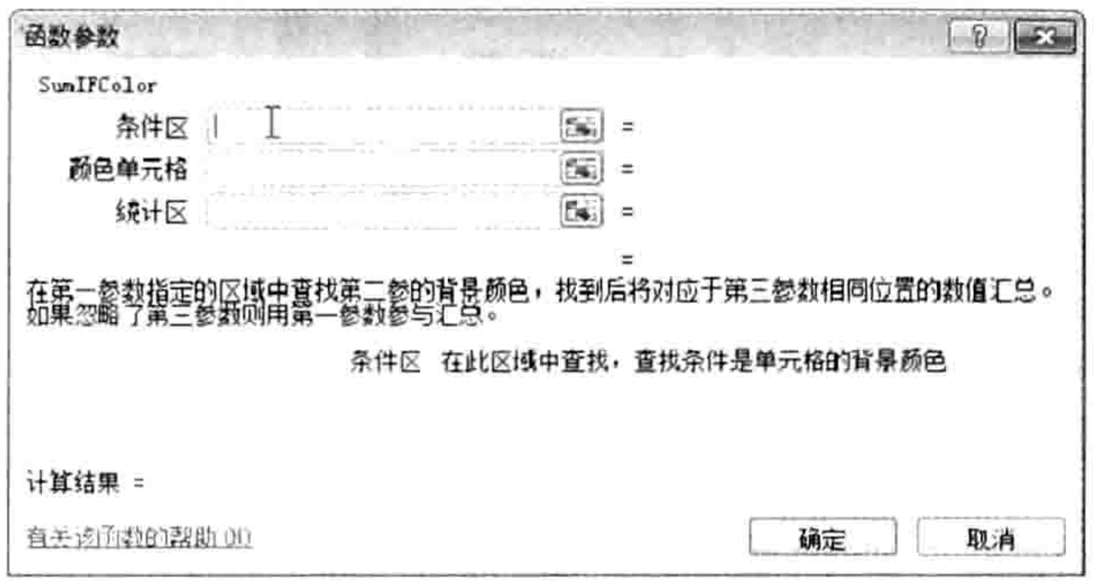


图 16.25 函数的功能描述和参数描述

表 16-2 内置函数的类别名称与编号

编号	类别名称	编号	类别名称
1	财务	8	逻辑
2	日期与时间	9	信息
3	数学与三角函数	10	命令
4	统计	11	自定义
5	查找与引用	12	宏控件
6	数据库	13	DDE/外部
7	文本	14	用户定义

其二是用于指定参数描述的 ArgumentDescriptions 参数，它是 Excel 2010 版之后才加入的功能，可以对所有参数添加说明，因此如果读者使用的是 Excel 2003 或者 Excel 2007 则不能使用此参数。

本例文件参见光盘：..\第十六章\16-14 为函数分类及添加说明.xlsm

◆ ◆ ◆

第17章 设计窗体

窗体可以实现与表格的交互，也可以通过窗体设计精美的操作界面。
善用窗体和窗体中的控件可以使自己的程序更具个性化，并增强 Excel 的功能。

17.1 UserForm 简介

UserForm 即用户窗体，通过窗体可以操作工作簿、工作表、单元格、批注、图形对象等，也可以利用窗体设计一个单独的操作界面，完全脱离单元格、工作表等数据载体而工作。

17.1.1 窗体与控件的用途

VBA 中的窗体与控件主要用于以下几方面：

- 设计登录窗口；
- 制作数据输入界面；
- 数据查询界面；
- 选项设置窗口；
- 制作程序帮助。

本章将对以上各类用途进行案例演示。

17.1.2 插入窗体与控件的方法

设计窗体的两个基本步骤是插入空白窗体和在窗体中添加控件。

1. 插入用户窗体

插入用户窗体的方法是在 VBE 界面中单击菜单中的“插入”→“用户窗体”命令，然后在当前工程中会出现一个名为“UserForm1”的窗体，效果如图 17.1 所示。

插入用户窗体需要注意以下三点：

（1）必须在有工作簿的前提下才能插入窗体，没有工作簿时菜单将呈禁用状态。

（2）当工程受密码保护时，必须先解除保护，然后才能插入窗体。

（3）当打开多个工作簿时，只能在当前选中的工程中插入窗体，而当前选中的工程不一定隶属于活动工作簿。

2. 在窗体中添加控件

控件位于工具箱中，而工具箱中的控件只能在窗体中才能发挥其作用。

在默认状态下工具箱是处于隐藏状态的，只有插入用户窗体后才会显示出来。

在窗体中添加控件必须先选中工具箱中的控件，然后按下鼠标左键并将其拖到窗体中。图 17.2 展示了将命令按钮添加到窗体中的过程。

要了解工具箱中每个控件的名称，可以将鼠标指针移到控件之上，Excel 会产生一个提示框，在提示框中标注控件的名称。

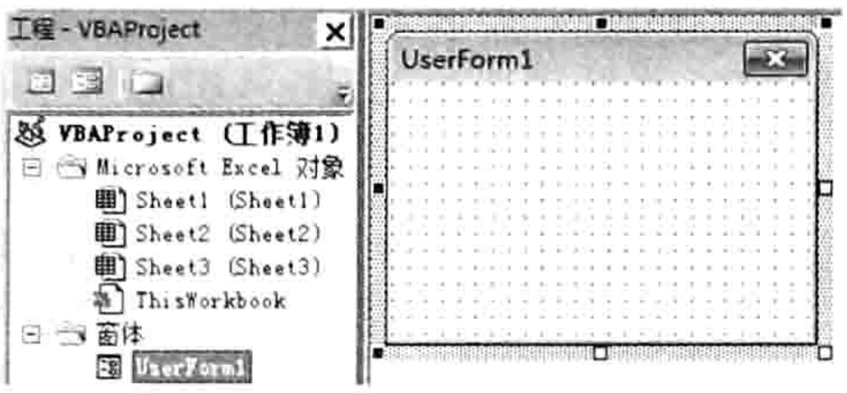


图 17.1 第一个插入的窗体

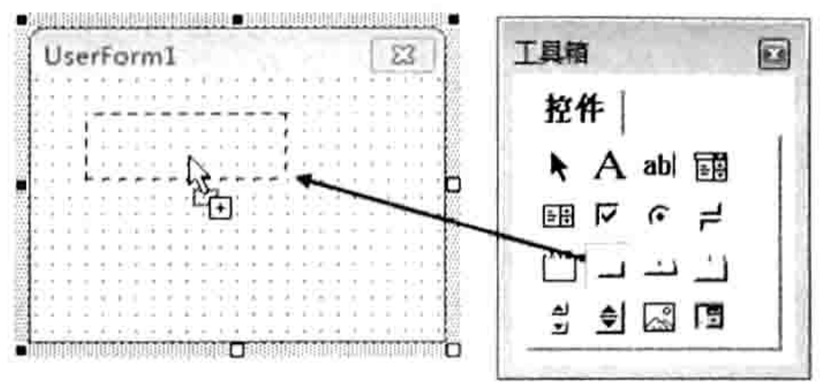


图 17.2 窗体与包含默认控件的工具箱

17.1.3 使用 Excel 5.0 对话框

除了在 VBE 中插入窗体外，在表中也可以插入窗体与控件。具体步骤如下。

step 1 在工作表标签上单击鼠标右键，从弹出的快捷菜单中选择“插入”命令。

step 2 在弹出的对话框中选择“MS Excel 5.0 对话框”，从而插入一个宏表对话框。

step 3 在表中默认已经产生一个窗体和两个按钮，读者可以通过开发工具中的表单控件来丰富窗体的内容。例如图 17.3 中左边是设计状态下的窗体界面，右边是运行状态下的窗体界面。

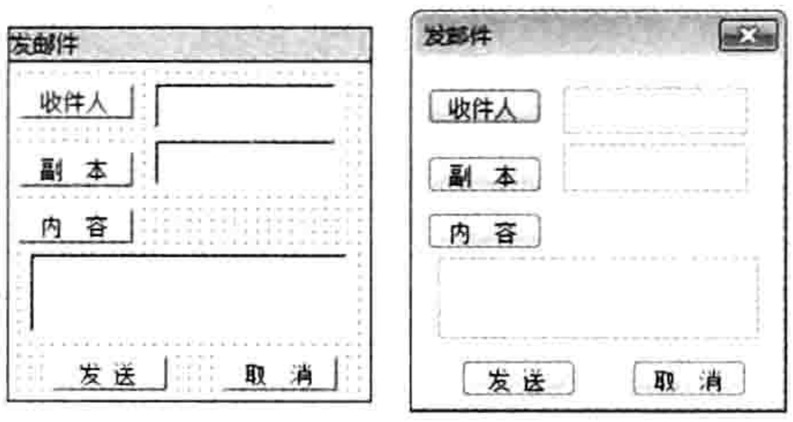


图 17.3 通过 Excel 5.0 对话框设计发邮件的窗体界面

Excel 5.0 对话框的执行方式有两种：一种是在对话框中的空白区域单击鼠标右键，从弹出的快捷菜单中选择“执行对话框”命令，另一种方法是利用 VBA 的 Show 方法执行窗体，例如对话框的名字是“窗体”，那么执行窗体的代码如下：

```
Sub 执行对话框()'代码存放位置：模块中
  DialogSheets("窗体").Show
End Sub
```

本例文件参见光盘：..\第十七章\ 17-1 Excel 5.0 对话框.xlsm

Excel 5.0 对话框是比较古老的窗体，在实际工作中应用极少，读者的学习重心应放在 VBE 界面中所创建的用户窗体上。

17.2 窗体控件一览

没有控件的窗体是没有存在价值的。本节向读者介绍可以应用于窗体中的常用控件的名称、功能，以及其外观。

17.2.1 标签

标签的代码名称是 Label，它在工具箱中的图标为**A**。

在窗体中添加标签控件后，标签的默认名称和默认的标题由控件名称加编号组成，第一个标签控件的名称和标题是“Label1”，第二个标签的默认名称和默认标题是“Label2”。

标签控件的功能是在窗体中显示说明性的文本。在如图 17.4 所示的窗体中使用了两个标签控件，其中选中的那个标签的名称是“Label1”，它的 Caption 属性值则已经由“Label1”被修改为“请选择学历：”。

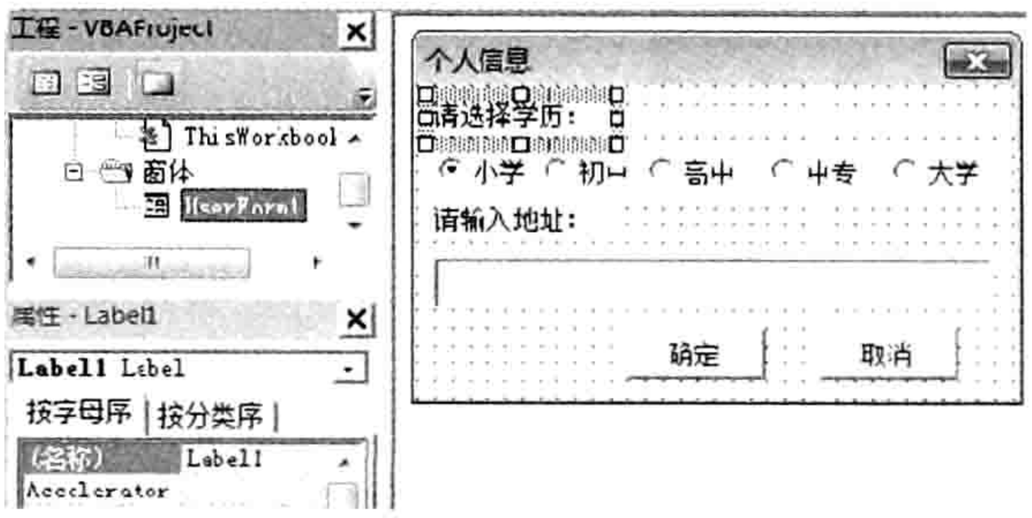


图 17.4　窗体中的标签控件

标签控件显示出来的字符在窗体执行阶段是不可被修改的，只能在设计窗体时才能被修改。

17.2.2　文本框

文本框的代码名称是 TextBox，它在工具箱中的图标为 **ab|**。

在窗体中添加文本框控件后，文本框的默认名称将由“TextBox”和编号组成。例如添加第二个文本框后，它的默认名称是“TextBox2”。

文本框的用途是在运行窗体时让用户输入文字或者数值，图 17.4 中“请输入地址：”文字下方的控件就是文本框。

文本框有名称，没有标题（即 Caption 属性）。

17.2.3　命令按钮

命令按钮的代码名称是 CommandButton，它在工具箱中的图标为 ▭。

命令按钮用于执行一个或者多个任务。它的默认名称和标题皆由“CommandButton”与编号组成。与标签控件一样，命令按钮的名称和标题都无法在运行窗体过程中被修改。

图 17.4 中的“确定”和“取消”两个控件就是命令按钮控件。

17.2.4　复合框

复合框（也称组合框）的代码名称是 ComboBox，它在工具箱中的图标为 ▤。

复合框可以理解为是列表框与文本框的组合，用户可以从复合框的列表中选出一个项目，也可以在复合框的文本框中输入任意字符串。

17.2.5　列表框

列表框的代码名称是 ListBox，它在工具箱中的图标为 ▤。

列表框用来显示可供用户选择的项目列表。如果不能显示全部项目，可以拖动它的滚动条来显示其他项目，也可以通过代码修改列表框的高度来适应列表项目的数量。

列表框和复合框的区别有两点，其一是列表框可以显示多项内容，复合框只能显示一项内容；其二是复合框允许从列表中选择一个项目也可以手工输入字符，而列表框没有输入功能。

17.2.6　复选框

复选框的代码名称是 CheckBox，它在工具箱中的图标为 ☑。

复选框用于创建一个方框，单击后可以在其中产生一个钩，再次单击时则清除钩。

复选框用于标示某个对象的状态，或者让用户在两个选项中二选一。

17.2.7 选项按钮

选项按钮的代码名称是 OptionButton，它在工具箱中的图标为 。

选项按钮的功能是让用户从多个选项中选择一个项目，它和复选框的区别是选项按钮只能单选，复选框可以多选。

17.2.8 分组框

分组框（也称框架）的代码名称是 Frame，它在工具箱中的图标为 。

分组框的功能是创建控件的功能组，将窗体中的若干个控件分组，从而让窗体的设计更美观，同时也可以避免多个选项按钮之间相互干扰。

17.2.9 切换按钮

切换按钮的代码名称是 ToggleButton，它在工具箱中的图标为 。

切换按钮的功能是创建一个切换开关，可以在按下和突起时分别实现不同的功能。

17.2.10 多页控件

多页控件的代码名称是 MultiPage，它在工具箱中的图标为 。

多页控件类似于分组框，可以将有某种内在联系的若干个控件单独作为一组显示。它与分组框控件的区别在于多个分组框可以同在一个页面中显示，而多页控件一次只能显示一页。它的功能与 TabStrip 控件相近。

17.2.11 滚动条

滚动条的代码名称是 ScrollBar，它在工具箱中的图标为 。

滚动条提供在长列表项目或大量信息中快速浏览的图形工具，以比例方式指示出当前位置，或是作为一个输入设备，成为速度或者数量的指示器，通常用它替代数字输入。

滚动条的功能与旋转按钮相近。旋转按钮的图标为 。

17.2.12 图像

图像的代码名称是 Image，它在工具箱中的图标为 。

图像控件用于在窗体上显示位图、图标，不能显示动画。

通常用图像控件装饰窗体，当然也可以通过它调用硬盘中的任意图片。

17.2.13 RefEdit

RefEdit 也称单元格选择器，它在工具箱中的图标为 。

RefEdit 运行时将弹出对话框让用户选择单元格，并返回该单元格对象。

Application.Inputnox 方法的最后一个参数为 8 时也可以返回用户选择的区域对象，它和 RefEdit 控件的功能一致。

17.2.14 附加控件

除默认控件外，用户还可以调用附加控件以强化窗体的功能。事实上很多有用的控件都没有在工具箱中罗列出来，需要用户手工调用。添加附加控件的步骤如下。

step 1 在显示窗体的前提下，单击菜单中的“视图”→“工具箱”命令。

step 2 在工具箱右上角的空白区单击鼠标右键，在弹出的菜单中选择“新建页”命令。

step 3 在空白区单击鼠标右键，在弹出的菜单中选择“附加控件”命令，在“附加控件”对话框中将需要的控件打钩，然后单击“确定”按钮，工具箱的“新建页”中将产生新添加的控件。

图 17.5 中包含 Flash 动画控件、网页控件和 ListView 三个控件，它们都是通过“附加控件”菜单添加的。

图 17.5 在新页中添加附加控件

17.3 设置控件属性

添加到窗体中的控件如果都保护默认状态将不利于使用，所有控件都需要对其部分或者所有属性进行设置，然后再投入工作中使用。

设置控件属性的主要目的有两个——其一是显示或者引用指定的字符，其二是美化窗体。

17.3.1 调整窗体控件位置与大小

将控件拖到窗体中后，通常会根据需求调整其大小与位置。

调整控件大小的方法是先选择控件，然后按住鼠标左键不放往四周的九个控制点之一向任意方向拖动鼠标，直到控件大小合适为止。

对于按钮和标签这类控件，还可以通过选择菜单中的“格式”→“正好容纳”命令使其自动调整大小，以适应控件的字符宽度与高度，类似于批注框的 AutoSize 属性。

调整控件位置和调整控件的大小一样，包括手工和菜单两种调整方式。手工调整位置的方法是选择控件后拖动到目标位置；而菜单调整方式没有手工调整的随意，但可以使控件按一定的方式对齐，例如“格式”菜单中的子菜单“水平间距”、“垂直间距”、“窗体内居中”、“排列按钮”等，读者可以逐个测试它们的对齐效果。

17.3.2 设置控件的顺序

当多个控件重叠时，可以调整其位置顺序。例如窗体中有一个按钮和一个图像控件，如果先插入命令按钮后插入图像控件，当两者重叠时，图像控件必定会覆盖命令按钮。如果需要将按钮移至图像控件之上，可以采用以下步骤：

step 1 选择图像控件。

step 2 单击菜单中的“格式”→“顺序”→“移至底层”命令。

当有超过两个控件重叠时，也可以对某个控件进行“上移一层”或者“下移一层”操作，菜单中有相应的功能按钮。

17.3.3 共同属性与非共同属性

当窗体中有多个控件时，它们总有部分共同属性。对于共同的属性，可以一次性设置完成。例如窗体中有一个标签和一个按钮控件，那么背景色就是它们的共同属性，可以在同时选择两个控件后按<F4>键调出“属性”窗口，然后在“属性”窗口中将 BackColor 属性设置为绿色，此时选中的两个控件都会同时显示为绿色。

图 17.6 中包含一个命令按钮和一个标签，由于是选中两个控件后再设置的 BackColor 属性，因此两个控件的背景色会同时修改成功。

图 17.6　设置共同属性

对于非共同属性，只能逐个设置——选择单个控件，然后在属性窗口中对属性赋值。

VBA 中的每一个控件都有 10～50 个属性，这些属性都显示为英文字母，对于 VBA 新手而言学习难度比较大。笔者对于常用的命令按钮控件、文本框控件和列表框控件的所有属性专门整理过，将它们的属性名称、含义、赋值范围和默认值都一一罗列在表格中，读者可以在本书的随书光盘中获得这些文件，它们的名称分别是“附录 E 命令按钮属性一览.pdf”、“附录 F 文本框属性一览.pdf”和“附录 G 列表框属性一览.pdf”。

17.3.4　设置颜色属性

很多控件都有颜色属性，包括背景颜色和字体颜色，而这两种属性的设置方式一致。

下面以设置 ForeColor（字体颜色）为例演示命令按钮的属性设置过程。

step 1　选择窗体中的命令按钮控件，按<F4>键显示“属性”对话框。

step 2　在“ForeColor”右边的列表框中单击从而调出颜色设置选项，其默认选项卡为“系统”。

step 3　切换到“调色板”选项卡，在该选项卡中罗列了 48 种颜色，如图 17.7 所示。

step 4　单击颜色块中的红色方块，按钮的字体立即显示为红色，如图 17.8 所示。

图 17.7　通过调色板设置字体色

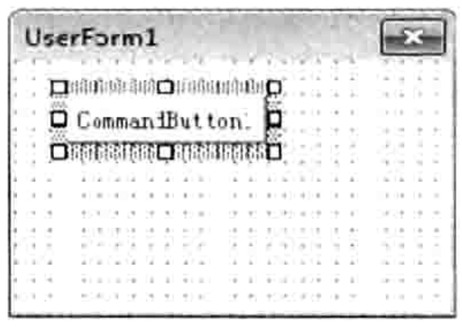


图 17.8　修改字体颜色之后的按钮

假设要通过代码设置命令按钮的字体颜色，最好的办法是先手工设置一次，将调色板中产生的颜色代码复制到 Sub 过程中。

例如本例中红色的颜色代码为“&H000000FF&”，假设窗体中有一个名为 CommandButton1 的命令按钮，将该控件的字体颜色调整为红色的操作步骤如下。

step 1　双击窗体进入代码窗口。

step 2　将自动产生的代码删除，然后录入以下代码：

```
'窗体的 Activate 事件，显示窗体时执行
Private Sub UserForm_Activate()  '代码存放位置：窗体代码窗口中
 CommandButton1.ForeColor = &HFF&'将命令按钮 CommandButton1 的字体颜色设置为红色
End Sub
```

step 3 按<F5>键执行代码，窗体中的命令按钮的字体将会显示为红色。

对于以上代码要注意两点：其一是代码中的“&HFF&”是“&H000000FF&”的缩写，将“&H000000FF&”录入到代码中窗口后会自动显示为“&HFF&”。其二是 UserForm_Activate 是一个事件过程，可以通过单击对象列表和过程列表录入代码，尽量不要手工录入事件过程的程序外壳。

本例文件参见光盘：..\第十七章\17-2 通过代码修改窗体中的命令按钮字体颜色.xlsm

17.3.5 设置控件的宽度与高度

所有控件都有高度和宽度两个属性，属性名称分别为 Height 和 Width。

尽管高度与宽度可以利用鼠标调整，但要精确调整时只有修改属性值。

设置高度与宽度属性比修改颜色属性更简单，直接在输入框中输入数字即可，例如输入 100 或者 25，按<Enter>键后可立即生效。

假设要利用代码来设置命令按钮的高度，那么可在 UserForm_Activate 事件中加入以下语句，表示激活窗体时自动将命令按钮的高度设置为 50：

```
CommandButton1.Height = 50
```

17.3.6 设置 Picture 属性

命令按钮、复选框、切换按钮、框架、多页控件、图像控件都可以设置背景图片。

下面以命令按钮为例，介绍设置背景图片步骤。

step 1 选择命令按钮，按<F4>键打开属性对话框。

step 2 命令按钮的 Picture 属性默认显示为“(None)”，单击“(None)”则会弹出一个浏览按钮，单击浏览按钮会再次弹出“加载图片”对话框，然后从“加载图片”对话框中选择一个提前准备好的图片，例如“1.jpg”，最后单击“打开”按钮返回 VBE 界面，窗体中的命令按钮将会显示出该图片的内容。

命令按钮加载图片后，默认状态是图片在按钮文字的上方，当按钮的高度不够时只能看到图片内容看不到文字，此时可以将按钮拉高使其显示完整内容，如图 17.9 所示即为命令按钮同时显示图片和文字的效果。

step 3 在属性窗口中将命令按钮的 PicturePosition 属性修改为 fmPicturePositionLeftCenter，表示图片显示在按钮的左方、文字显示右边居中的位置，然后再根据图片的大小修改按钮的高度，最终效果如图 17.10 所示。

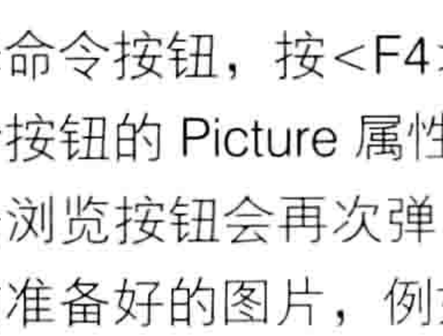


图 17.9 图片在上，文字在下

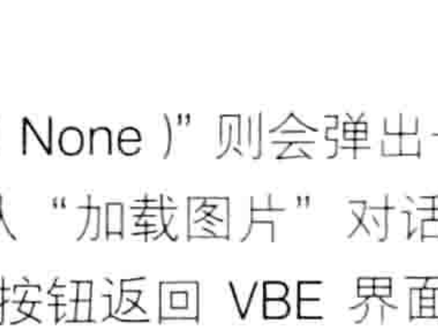


图 17.10 图像在左，文字居中

如果要用代码设置以上效果，应使用以下代码：

```
'①代码存放位置：窗体代码窗口中②随书光盘中有每一句代码的含义注释
Private Sub UserForm_Activate()
```

```
    If Len(Dir("d:\1.jpg")) > 0 Then
    CommandButton1.Picture = LoadPicture("d:\1.jpg")
    CommandButton1.PicturePosition = fmPicturePositionLeftCenter
    CommandButton1.Height = 35
    End If
End Sub
```

本例文件参见光盘:..\第十七章\17-3 使用代码为命令按钮指定 Picture 属性.xlsm

17.3.7 设置 RowSource 属性

RowSource 属性的功能是为列表框或者复合框指定数据来源，这也是极为常见的属性。

RowSource 属性总是和 BoundColumn、ColumnWidths、ColumnHeads、ListStyle 等属性同时使用，从而使复合框或者列表框能将数据更完整、更美观地呈现出来。

假设工作表的 A1:B5 区域中有各部门的名称和人数，如图 17.11 所示，要将该区域的值显示在复合框中可按以下步骤操作。

step 1 在窗体中插入一个复合框。

step 2 在属性窗口中将 RowSource 属性的值设置为“A2:B5”，表示复选框的数据源来自此区域。

step 3 将 BoundColumn 属性设置为 2，表示复合框同时显示两列。

step 4 将 ColumnWidths 属性设置为 “40,40”，表示每一列的宽度都是 40。

step 5 将 ColumnHeads 属性设置为 True，表示让复合框显示表头（也称标题行）。

step 6 将 ListStyle 属性设置为 1- fmListStyleOption，表示让复合框显示为单选样式，即每一行的左方都显示一个像选项按钮那样的图案。

step 7 单击窗体的空白区域，按<F5>键运行窗体，单击复合框的下拉箭头，在复合框的列表中会显示工作表中 A2:B5 区域中的值，而 A1:B1 中的值会显示在列标行中，效果如图 17.12 所示。

	A	B
1	部门	人数
2	企划	1
3	生管	2
4	财务	1
5	生产	30

图 17.11 数据源

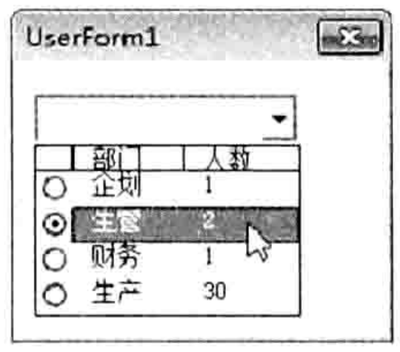


图 17.12 利用复合框引用工作表中的数据

当然，也可以用代码实现以上几个设置步骤的相同功能，代码如下：

```
'①代码存放位置：窗体代码窗口中②随书光盘中有每一句代码的含义注释
Private Sub UserForm_Activate()
  ComboBox1.RowSource = "a2:b5"
  ComboBox1.ColumnHeads = True
  ComboBox1.ColumnCount = 2
  ComboBox1.ColumnWidths = "40;40"
  ComboBox1.ListStyle = fmListStyleOption
End Sub
```

本例文件参见光盘：..\第十七章\17-4 将工作表中的数据显示在复合框中.xlsm

17.3.8 设置 Flash 动画

Excel 集成了 Flash 控件，可以嵌入并播放 Flash 动画。

在默认状态下 Flash 控件未出现在工具箱中，需要用户手工将它添加到工具箱中才可调用。

下面演示附加 Flash 控件并设置其属性的基本步骤。

step 1 单击菜单中的“插入”→“用户窗体”命令，从而生成一个新的空白窗体。

step 2 在工具箱的右下角空白区域中单击鼠标右键，在弹出的快捷菜单中选择“附加控件”命令。

step 3 打开如图 17.13 所示的对话框，单击列表中的任意控件，然后按<S>键，再按<H>键，光标将定位于“Shockwave Flash Object”处，单击将该项目从而使其被选中，最后单击“确定”按钮返回工具箱界面。

step 4 将工具箱中新建的 ShockwaveFlash 控件拖到窗体中。

step 5 选择 ShockwaveFlash 控件，然后在属性对话框中将 Movie 属性设置为 Flash 动画文件的完整路径，例如“E:\动画\andy.swf”，表示在窗体中播放该动画。

step 6 将 EmbedMovie 属性设置为 True，表示将动画文件嵌入到工作簿中。

step 7 手工拖动 ShockwaveFlash 控件，适当调整其大小和边距，然后按<F5>键运行窗体，窗体中的 ShockwaveFlash 控件会自动调用该动画文件，效果如图 17.14 所示。

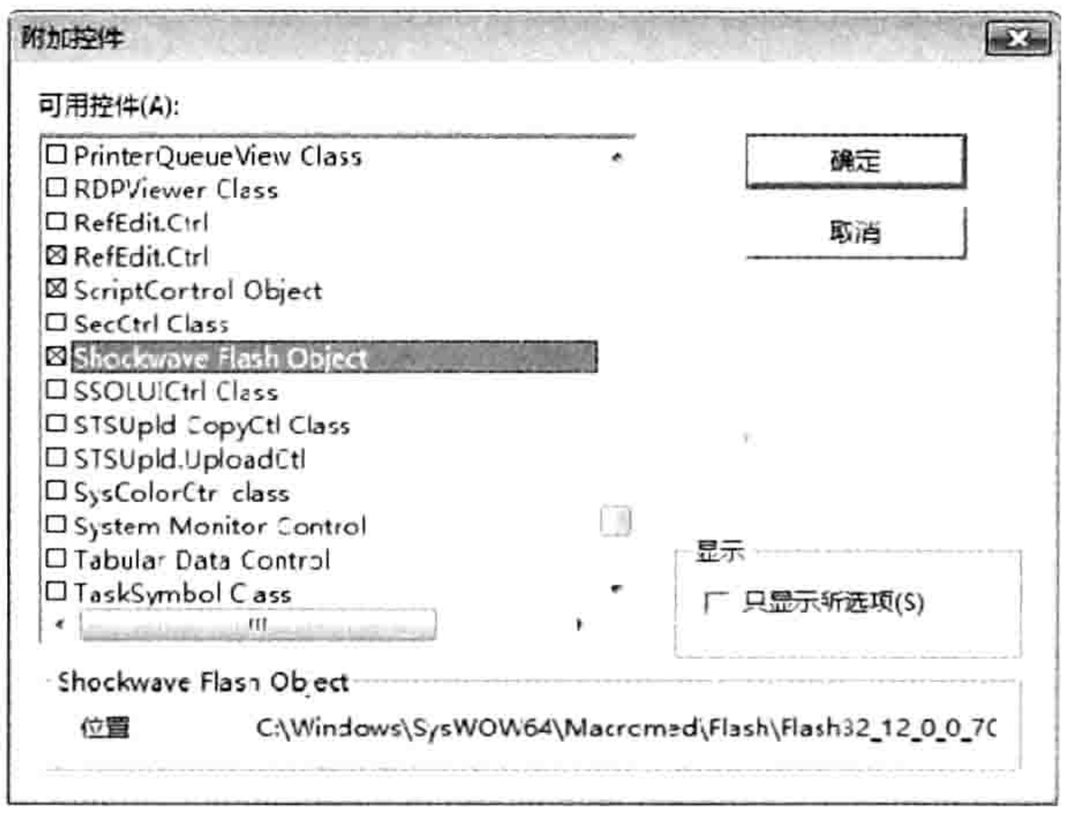


图 17.13 引用 Shockwave Flash Object 控件

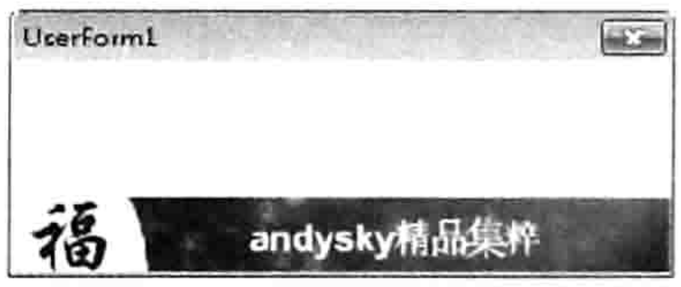


图 17.14 在窗体中播放 Flash 动画

CD-ROM 本例文件参见光盘：..\第十七章 17-5 播放 Flash 动画.xlsm

17.4 窗体与控件的事件

窗体中的 UserForm 对象和所有控件都拥有多个事件，脱离事件后 UserForm 对象和所有控件都不再有存在的价值。本节重点介绍 UserForm 对象和常用控件的常用事件。

17.4.1 UserForm 对象的事件

UserForm 对象的事件是指 UserForm 对象在满足内部设定的条件时所触发的事件，例如关闭窗体前触发 UserForm 对象的 QueryClose 事件，激活窗体后触发 UserForm 对象的 Activate 事件。

UserForm 对象有 22 个事件，事件名称与含义如表 17-1 所示。

表 17-1 窗体事件列表

事件名称	触发条件（在何时执行这个事件）
AddControl	当将控件插入到窗体时
Activate	激活窗体时
BeforeDragOver	当窗体中执行拖放操作时
BeforeDropOrPaste	即将在一个对象上放置或粘贴数据时
Click	在窗体中用鼠标单击控件时
DblClick	在窗体中双击鼠标时
Deactivate	窗体失去焦点时
Error	当控件检测到一个错误，并且不能将该错误信息返回调用程序时
Initialize	加载窗体之后、显示这个窗体之前
KeyDown	按下任意键时
KeyUp	当某键弹起时
KeyPress	当用户按下一个 ANSI 键时（即按键后能产生某字符时）
Layout	修改窗体的位置时
MouseDown	按下任意鼠标键时
MouseUp	释放鼠标按键时
MouseMove	移动鼠标时
QueryClose	关闭窗体之前
RemoveControl	从窗体删除一个控件时
Resize	更改窗体的大小时
Scroll	按下滚动条时
Terminate	关闭窗体之后
Zoom	修改窗体的 Zoom 属性时（即修改窗体的缩放比例时）

以上 22 个事件中常用的事件是 Activate 事件，它通常用于激活窗体时对窗体中的某些控件的属性赋值。例如指定复合框、列表框的宽度与边距，或者设置复选框的默认值、设置 Flash 动画的路径等，在前面已经有这方面的介绍。在接下来的内容中会演示几个关于窗体事件的应用。

17.4.2 激活窗体时将所有工作表名称导入到列表框中

案例要求：在窗体中添加一个列表框控件，激活窗体时将活动工作簿中的所有工作表名称导入到列表框中，而且根据表的数量调整窗体与列表框的高度。

操作步骤：

step 1 单击菜单中的“插入”→“用户窗体”命令。

step 2 将工具箱中的列表框控件拖到窗体中。

step 3 双击窗体进入窗体的代码窗口，单击过程列表，从列表中选择 Activate，如图 17.15 所示。单击后会在窗体的代码窗口中产生 UserForm_Activate 事件的程序外壳。

step 4 在 UserForm_Activate 事件中插入设置窗体属性和列表框属性的代码。完整代码如下

```
'①代码存放位置：窗体代码窗口中②随书光盘中有每一句代码的含义注释
Private Sub UserForm_Activate()
```

```
  Me.Caption = "工作表目录"
  ListBox1.Width = 80
  Dim sht As Worksheet
  For Each sht In Sheets
    Me.ListBox1.AddItem sht.Name
  Next
  Me.Height = ListBox1.ListCount * ListBox1.Font.Size + 35
  ListBox1.Height = ListBox1.ListCount * ListBox1.Font.Size + 5
End Sub
```

step 5 单击菜单中的“插入”→“模块”命令，然后在模块中录入以下代码：

```
Sub 显示窗体()'代码存放位置：模块中
  UserForm1.Show
End Sub
```

step 6 运行过程“显示窗体”，窗体中的列表框会显示活动工作簿中所有工作表的名称，效果如图 17.16 所示。

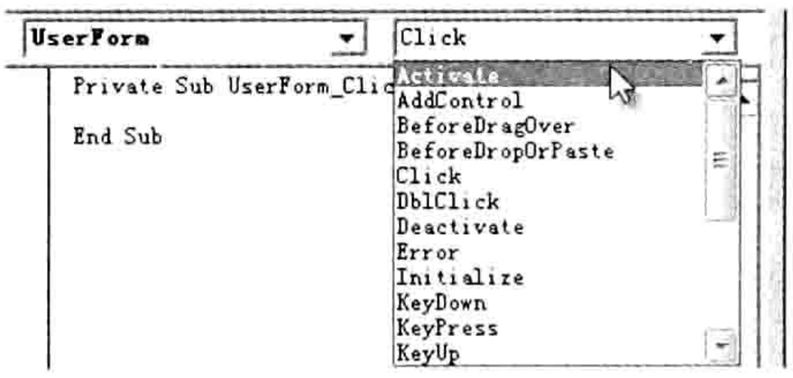


图 17.15 单击过程列表产生事件过程外壳

图 17.16 在列表框中显示所有工作表的名称

思路分析：

预先设置窗体和窗体中的控件的某些属性应该在 UserForm_Initialize 事件或者 UserForm_Activate 中完成，前者是加载窗体时触发的事件，后者是显示窗体时触发的事件。

在本例中，UserForm_Activate 事件过程设置了窗体的标题文字、高度和列表框的宽度与高度。其中有两个重点，其一是使用 AddItem 方法配合循环语句将数据逐一添加到列表框中，AddItem 方法仅对列表框和复合框有效。

其二是计算列表框的高度。ListBox1.ListCount 属性代表列表框的行数，但是列表框没有行高这个属性，无法以行数乘以行高得到列表框的高度，所幸列表框的 Height 属性以磅为单位，而字体也以磅为单位，因此用行数乘以字体的大小再加上、下边界的高度即可得到列表框的高度。

语法补充：

（1）AddItem 方法用于向列表框或复合框中添加数据，通常配合循环语句使用，语法如下：

```
object.AddItem [ item [, varIndex]]
```

其中参数 item 代表要添加的文本，参数 varIndex 代表存放位置，忽略该参数时表示将文本默认添加到末尾。

（2）UserForm.Show 方法用于运行或者显示一个指定名称的窗体，其语法如下：

```
object.Show [modal]
```

可选参数 modal 代表运行窗体时是否允许编辑工作表，当赋值为 1 时表示窗体是模态的，运行窗体期间不能编辑工作表；当赋值为 0 时表示窗体是无模式的，运行窗体时可以编辑工作表，默认值是 1。读者可以分别测试该参数是 0 和 1 时对操作单元格的影响。

本例文件参见光盘：..\第十七章\17-6 激活窗体时将在列表框中创建工作表目录.xlsm

17.4.3 双击时关闭窗体

案例要求：双击窗体的空白区域关闭窗体。

操作步骤：

step 1 单击菜单中的“插入”→“用户窗体”命令从而生成一个空白窗体。

step 2 在窗体中随意添加几个控件。

step 3 双击窗体进入窗体代码窗口，删除自动产生的代码，然后录入以下代码：

```
'代码存放位置：窗体代码窗口中
Private Sub UserForm_DblClick(ByVal Cancel As MSForms.ReturnBoolean)
  Unload Me
End Subk
```

step 4 按<F5>键运行窗体，然后双击窗体的空白区域可以关闭窗体。

思路分析：

UserForm_DblClick 事件是窗体的双击事件，双击窗体的空白区域时触发此事件，双击标题区域不触发事件，双击窗体中的控件则只触发该控件的事件，与窗体无关。

Unload 方法用于关闭指定名称的窗体，在 UserForm_DblClick 事件中使用 Unload 方法即可关闭当前窗体。

语法补充：

（1）Unload 方法用于关闭指定名称的窗体，例如“Unload Userform1”表示关闭名为“Userform1”的窗体，对于当前窗体则可以简写为 Me，不用窗体名称。

（2）如果要单击关闭窗体，那么应改用 UserForm_Click 事件。

本例文件参见光盘：..\第十七章\ 17-7 双击时关闭窗体.xlsm

17.4.4 窗体永远显示在屏幕的左上角

案例要求：让窗体永远显示在屏幕的左上角，无法移动其位置。

操作步骤：

step 1 单击菜单中的“插入”→“用户窗体”命令从而生成一个空白窗体。

step 2 双击窗体进入窗体代码窗口，删除自动产生的代码，然后录入以下代码：

```
'①代码存放位置：窗体代码窗口中②随书光盘中有每一句代码的含义注释
Private Sub UserForm_Terminate()
  Me.Left = 0
  Me.Top = 0
End Sub
Private Sub UserForm_Layout()
  Me.Left = 0
  Me.Top = 0
End Sub
```

step 3 按<F5>键运行窗体，窗体将显示在屏幕的左上角，拖动窗体到其他位置后，窗体总会返回屏幕的左上角。

思路分析：

让窗体永远显示在屏幕的左上角包含两个步骤：其一是刚显示窗体前调整它的 Left 和 Top 属性，其二是拖动窗体时修改它的 Left 和 Top 属性。UserForm_Terminate 事件和 UserForm_Layout 刚好对应这两个动作，因此将修改窗体边距的代码插入到这两个事件中。

语法补充：

（1）UserForm_Terminate 事件在加载窗体之后、显示窗体之前触发，而 UserForm_Activate 事件是显示窗体时触发，因此本例宜用前者，使用后者会闪屏——显示窗体后才调整边距。

（2）UserForm_Layout 事件在修改窗体位置时触发，没有参数。

本例文件参见光盘：..\第十七章\ 17-8 让窗体只能显示在左上角.xlsm

17.4.5 按下左键移动窗体、按下右键移动控件

案例要求：在窗体中按下鼠标左键不放并移动鼠标时窗体会相应地移动，而按下鼠标右键不放并移动鼠标时，窗体中的所有控件都跟着移动。

操作步骤：

假设已有一个名为 Userform1 的窗体，在窗体中有名为 Image1、CommandButton1 和 CommandButton2 的三个控件，要按下鼠标左键不放移动窗体、按下鼠标右键不放移动控件可按以下步骤操作：

step 1 双击窗体进入窗体代码窗口，删除自动产生的代码，然后录入以下代码：

```
'①代码存放位置：窗体代码窗口中②随书光盘中有每一句代码的含义注释
Dim MouseX As Double, MouseY As Double, ImageX As Double, ImageY As Double
Dim CommandButton1X As Double, CommandButton1Y As Double, CommandButton2X As
Double, CommandButton2Y As Double
Private Sub UserForm_MouseDown(ByVal Button As Integer, ByVal Shift As Integer,
ByVal X As Single, ByVal Y As Single)
  MouseX = X
  MouseY = Y
  ImageX = Image1.Left
  ImageY = Image1.Top
  CommandButton1X = CommandButton1.Left
  CommandButton1Y = CommandButton1.Top
  CommandButton2X = CommandButton2.Left
  CommandButton2Y = CommandButton2.Top
End Sub
Private Sub UserForm_MouseMove(ByVal Button As Integer, ByVal Shift As Integer,
ByVal X As Single, ByVal Y As Single)
  If Button = 1 Then
    Me.Left = Me.Left + (X - MouseX)
    Me.Top = Me.Top + (Y - MouseY)
  ElseIf Button = 2 Then
    Image1.Left = ImageX + (X - MouseX)
    Image1.Top = ImageY + (Y - MouseY)
    CommandButton1.Left = CommandButton1X + (X - MouseX)
    CommandButton1.Top = CommandButton1Y + (Y - MouseY)
    CommandButton2.Left = CommandButton2X + (X - MouseX)
    CommandButton2.Top = CommandButton2Y + (Y - MouseY)
  End If
End Sub
```

以上代码包含了公共变量和 UserForm_MouseDown、UserForm_MouseMove 两个事件。

step 2 按<F5>键运行窗体，然后在窗体中的任意空白区域按住鼠标左键不放并拖动鼠标，窗体会相应地移动，和拖动窗体的标题栏一样。

step 3 在窗体的任意空白区域按下鼠标右键不放并拖动鼠标，窗体中的三个控件会随鼠标指针的移动而移动。换而言之，使用工作簿事件可以在窗体运行状态下修改控件的位置。如图 17.17 和图 17.18 所示分别是按下鼠标右键并拖动的控件状态比较。

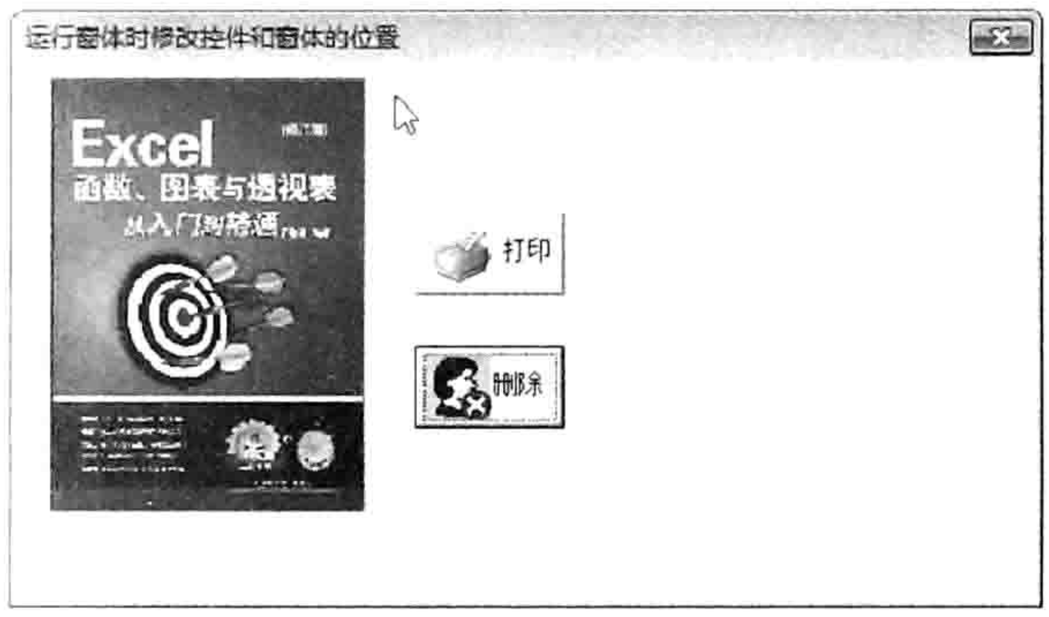


图 17.17 按下鼠标右键时

图 17.18 按下鼠标右键并且拖动鼠标时

思路分析：

按下鼠标左键不放并拖动鼠标时可移动窗体，主要分三步完成：

第一，刚按下鼠标时在 UserForm_MouseDown 事件中将鼠标的横坐标记录在公共变量 MouseX 中，将鼠标的纵坐标记录在公共变量 MouseY 中。

第二，在 UserForm_MouseMove 事件中判断是否按下了鼠标左键，在表 17-2 中罗列了该参数的每一个返回值所对应的功能说明。

第三，用户按下鼠标左键不放并且拖动时，将窗体的左边距设置为窗体的当前左边距加上当前鼠标横坐标与 MouseX 之差，将窗体的上边距设置为当前上边距加上当前鼠标纵坐标与 MouseY 之差。

按住鼠标右键不放并拖动鼠标时移动窗体中的控件与按住鼠标左键不放并拖动鼠标时移动窗体的思路相近，不过由于多个控件的位置并不相同，不能对每个控件都采用相同的数据去设置它们的边距，因此需要在 UserForm_MouseDown 事件中分别记录每个控件的当前左边距与上边距，然后在 UserForm_MouseMove 事件中根据鼠标移动的距离，将该距离累加到控件原来的边距上即可。

语法补充：

（1）UserForm_MouseDown 事件是按下鼠标按键时触发的事件，UserForm_MouseMove 是按下鼠标按键并拖动时触发的事件，两者都有 Button、Shift、X 和 Y 四个参数，其中 Button 参数用于识别用户按下了哪些鼠标按键，表 17-2 中罗列了该参数的返回值及其含义。

表 17-2 Button参数的取值范围与含义说明

值	说明	值	说明
0	按键未被按下	4	按下中键
1	按下左键	5	同时按下左键和中键
2	按下右键	6	同时按下中键和右键
3	同时按下左键和右键	7	三个按键全都按下

（2）UserForm_MouseDown 事件和 UserForm_MouseMove 的 Shift 参数用于识别用户是否按下了 Alt、Ctrl 或者 Shift 键。表 17-3 中罗列了该参数的返回值及其含义说明。

表 17-3　Shift参数的取值范围与含义说明

值	说明	值	说明
1	按下 Shift 键	5	同时按下 Alt 和 Shift 键
2	按下 Ctrl 键	6	同时按下 Alt 和 Ctrl 键
3	同时按下 Shift 和 Ctrl 键	7	同时按下 Alt、Shift 和 Ctrl 键
4	按下 Alt 键		

（3）UserForm_MouseDown 事件和 UserForm_MouseMove 的 X 和 Y 参数分别代表鼠标指针的横坐标和纵坐标。

本例文件参见光盘：..\第十七章\17-9 按下左键移动窗体、按下右键移动控件.xlsm

17.4.6　控件事件介绍

窗体中的任何控件都有自己的专用事件，但大部分事件与窗体的事件在语法上一致。

限于篇幅，这里不再一一罗列各种控件所支持的事件，读者可以在帮助中查询所有控件的事件。例如查询复选框控件的事件，可以在帮助窗口中输入“复选框控件”，然后选择“复选框控件”的帮助说明，再单击“事件”即可看到它所支持的所有事件，从列表中单击事件名称可以查看详细解释与实例。

如图 17.19 所示是命令按钮的事件列表，读者可以用相同的方法调用其他控件的事件列表。

图 17.19　命令按钮控件的事件列表

17.4.7　在窗体中建立超链接

案例要求：在窗体中建立三个网址的超链接，鼠标移过显示为蓝色，同时显示下画线，并在窗体的标题栏中显示网址，单击网址可以打开该网页。

操作步骤：

step 1　在窗体中添加三个标签控件，并且分别将它们的 Caption 属性赋值为百度搜索引擎、163新闻网、Excel 插件《E 灵》。

step 2　双击窗体进入窗体的代码窗口，删除自动产生的代码，然后录入以下代码：

```
'①代码存放位置：窗体代码窗口中②随书光盘中有每一句代码的含义注释
Private Sub Label1_MouseMove(ByVal Button As Integer, ByVal Shift As Integer,
ByVal X As Single, ByVal Y As Single)
  Label1.Font.Underline = True
  Label1.ForeColor = &HFF0000
  Me.Caption = "http://baidu.com"
End Sub
Private Sub Label1_Click()
  Shell "explorer.exe http://baidu.com", vbMaximizedFocus
End Sub
Private Sub Label2_MouseMove(ByVal Button As Integer, ByVal Shift As Integer,
ByVal X As Single, ByVal Y As Single)
  Label2.Font.Underline = True
  Label2.ForeColor = &HFF0000
  Me.Caption = "http://news.163.com"
End Sub
Private Sub Label2_Click()
  Shell "explorer.exe http://news.163.com", vbMaximizedFocus
End Sub
Private Sub Label3_MouseMove(ByVal Button As Integer, ByVal Shift As Integer,
ByVal X As Single, ByVal Y As Single)
  Label3.Font.Underline = True
  Label3.ForeColor = &HFF0000
  Me.Caption = "http://excelbbx.net"
End Sub
Private Sub Label3_Click()
  Shell "explorer.exe http://excelbbx.net", vbMaximizedFocus
End Sub
Private Sub UserForm_MouseMove(ByVal Button As Integer, ByVal Shift As Integer,
ByVal X As Single, ByVal Y As Single)
  Label1.Font.Underline = False
  Label1.ForeColor = &H0&
  Label2.Font.Underline = False
  Label2.ForeColor = &H0&
  Label3.Font.Underline = False
  Label3.ForeColor = &H0&
  Me.Caption = "请选择网址"
End Sub
```

step 3 按<F5>键运行窗体，将鼠标指针移到第一个标签上，标签的文字会显示为蓝色，并且添加下画线，效果如图 17.20 所示。

step 4 将鼠标指针移到第二个标签上，第二个标签的文字会显示为蓝色，并且添加下画线，第一个标题则恢复原状。

step 5 将鼠标指针移到标签以外的空白区域，所有标签都恢复为原状，效果如图 17.21 所示。

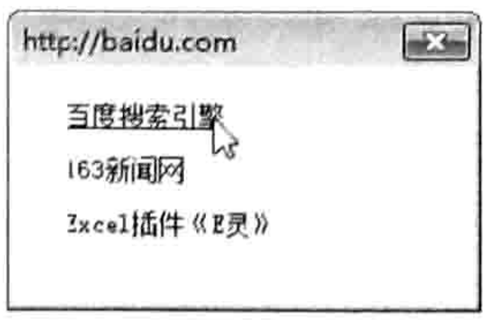


图 17.20 鼠标移过标签时的效果

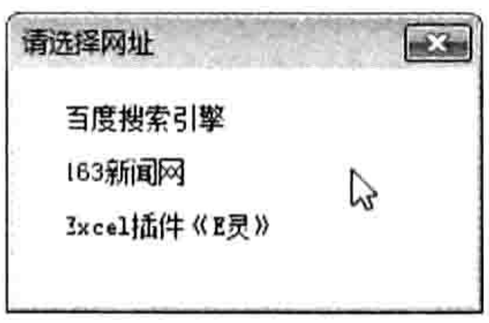


图 17.21 鼠标离开标签时的效果

思路分析：

实现添加网页超链接的效果主要包含三个步骤，其一是通过标签控件的 MouseMove 事件让标签显示为蓝色，同时添加下画线。其中蓝色的颜色代码可以从属性窗口中取得，添加下画线的代码则可以通过录制宏取得。

其二是鼠标指针离开标签控件时让控件恢复原状。任何控件都没有鼠标指针离开控件时触发的事件，不过由于鼠标指针离开控件时会触发窗体对象的 UserForm_MouseMove 事件，因此本例将还原所有控件的下画线和颜色代码置于在 UserForm_MouseMove 事件中。

其三是单击标签时打开对应的网站，本例采用的是 Shell 函数，用“Explorer.exe 网址”作为它的参数即可。要注意“Explorer.exe”与网址之间必须有一个空格。

语法补充：

（1）Label1_Click 事件是单击名为 Label1 的控件时触发的事件，没有参数。若要使用双击事件则应改用 Label1_DblClick。窗体和控件都没有右击事件，不过可以通过 MouseDown 事件变相地实现右击事件，完整代码如下：

```
Private Sub Label1_MouseDown(ByVal Button As Integer, ByVal Shift As Integer,
ByVal X As Single, ByVal Y As Single)
  If Button = 2 Then MsgBox "已触发右击事件"
End Sub
```

（2）Shell 函数的功能是打开一个可执行程序，本例中的可执行程序是“Exploer.exe”，网址是“Exploer.exe”的参数，因此“Exploer.exe”与网址之间需要有一个空格。Shell 函数的语法如下：

```
Shell(pathname[,windowstyle])
```

其中第 1 参数是需要被执行的文本形式的程序名称，第 2 参数代表打开该程序后是否放大窗口，以及是否让窗口获得焦点。表 17-4 中罗列了第 2 参数的取值范围及其功能描述。

表 17-4　Shell函数的第 2 参数的取值范围与功能描述

常量	值	描述
vbHide	0	窗口被隐藏，并且焦点会移到隐式窗口
VbNormalFocus	1	窗口具有焦点，并且会还原到它原来的大小和位置
VbMinimizedFocus	2	窗口会以一个具有焦点的图标来显示
VbMaximizedFocus	3	窗口是一个具有焦点的最大化窗口
VbNormalNoFocus	4	窗口会被还原到最近使用的大小和位置，而当前活动的窗口仍然保持活动
VbMinimizedNoFocus	6	窗口会以一个图标来显示，而当前活动的窗口仍然保持活动

本例文件参见光盘：..\第十七章\ 17-10 设计超链接.xlsm

17.4.8　鼠标移过时切换列表框数据

案例要求：工作表中有如图 17.22 所示的数据，包含一班、二班、三班的学员表。现在需要在窗体中鼠标指针移过单选按钮时自动切换列表框中的数据。

操作步骤：

step 1　单击菜单中的“插入”→“用户窗体”命令从而创建一个空白窗体。

step 2　在窗体中添加三个选项按钮和一个列表框控件，按如图 17.23 所示的方式布局。

	A	B	C
1	一班	二班	三班
2	赵	王	韩
3	钱	冯	杨
4	孙	陈	朱
5	李	褚	秦
6	周	卫	尤
7	吴	蒋	许
8	郑		何

图 17.22　数据源

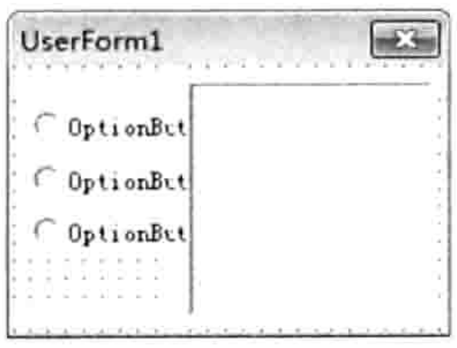


图 17.23　窗体控件布局

step 3　双击窗体进入代码窗口，删除自动产生代码，然后录入以下代码：

```
'①代码存放位置：窗体代码窗口中②随书光盘中有每一句代码的含义注释
Private Sub UserForm_Activate()
  OptionButton1.Caption = Range("A1")
  OptionButton2.Caption = Range("B1")
  OptionButton3.Caption = Range("C1")
  Me.Caption = "请选择班级"
End Sub
Private Sub OptionButton1_MouseMove (ByVal Button As Integer, ByVal Shift As
Integer, ByVal X As Single, ByVal Y As Single)
  OptionButton1.Value = True
  ListBox1.RowSource = "A1:A" & Cells(Rows.Count, 1).End(xlUp).Row
End Sub
Private Sub OptionButton2_MouseMove(ByVal Button As Integer, ByVal Shift As
Integer, ByVal X As Single, ByVal Y As Single)
  OptionButton2.Value = True
  ListBox1.RowSource = "B1:B" & Cells(Rows.Count, 2).End(xlUp).Row
End Sub
Private Sub OptionButton3_MouseMove(ByVal Button As Integer, ByVal Shift As
Integer, ByVal X As Single, ByVal Y As Single)
  OptionButton3.Value = True
  ListBox1.RowSource = "C1:C" & Cells(Rows.Count, 3).End(xlUp).Row
End Sub
```

step 4　按<F5>键运行窗体，窗体中的三个选项按钮将分别显示为一班、二班、三班，将鼠标移向第一个选项按钮之上时，列表框中会显示工作表中 A 列中与一班相关的数据，效果如图 17.24 所示。当鼠标指针移到第二个选项按钮之上时，第二个选项按钮将呈选中状态，同时列表框的内容更新为工作表中 B 列中与二班相关的数据，效果如图 17.25 所示。

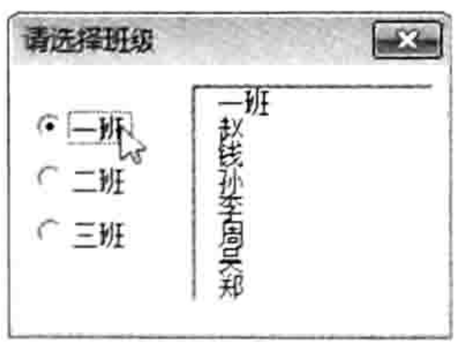


图 17.24　指向“一班”时显示一班的数据

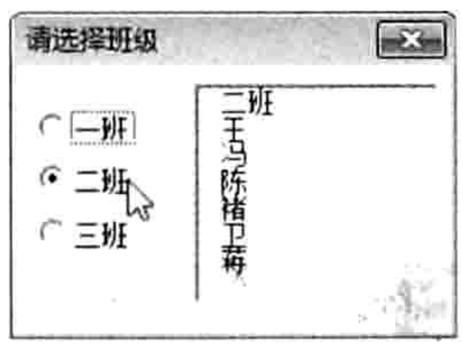


图 17.25　指向“二班”时显示二班的数据

思路分析：

将区域中的数据导入到列表框中有三种办法：一是将区域地址赋值给列表框中的 RowSource 属性，二是通过 AddItem 方法逐一次添加数据到列表框中，三是将区域的值转换成数组，再将数组赋值给列表框的 List 属性。本例采用的是第一种办法，因此当鼠标移过选项按钮时，在选项按钮的 MouseMove 事件中修改列表框的 RowSource 属性值即可更新列表框的数据。

使用选项按钮的单击事件也可以实现本例的同等功能，不过 MouseMove 事件的工作效率会更高一些。

语法补充：

RowSource 属性代表数据源地址，它是文本形式的，因此不能将对象赋值给 RowSource。RowSource 属性的语法如下：

```
object.RowSource = String
```

本例文件参见光盘：..\第十七章\ 17-11 让列表框随鼠标移动而更新数据.xlsm

17.4.9 让输入学号的文字框仅能录入 6 位数字

案例要求：为了规范学号，要求从窗体中输入学号并导入到单元格中；必须输入数字，而且必须是 6 位才能导入到单元格中。

操作步骤：

step 1 单击菜单中的“插入”→“用户窗体”命令从而创建一个空白窗体。

step 2 在窗体中插入一个标签，将其 Caption 属性设置为“请输入学号：”，然后添加一个文字框、两个按钮，按钮的 Caption 属性分别设置为“确定”和“关闭”。各控件的布局方式如图 17.26 所示。

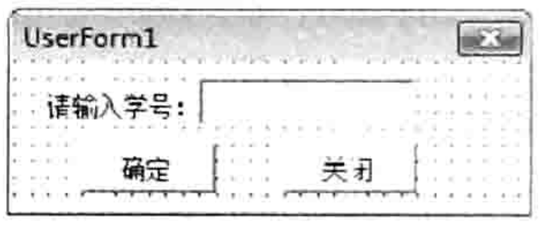


图 17.26 窗体控件布局

step 3 双击窗体进入代码窗口，删除自动产生代码，然后录入以下代码：

```
'①代码存放位置：窗体代码窗口中②随书光盘中有每一句代码的含义注释
Private Sub UserForm_Activate()
  Me.Caption = "请录入 6 位的学号"
  CommandButton2.Cancel = True
End Sub
Private Sub TextBox1_Change()
  If Len(TextBox1.Value) > 0 Then
    If Not IsNumeric(Me.TextBox1.Value) Then
      TextBox1 = Replace(Left(Me.TextBox1, Len(Me.TextBox1) - 1), " ", "")
    Else
      TextBox1 = Left(TextBox1, 6)
    End If
  End If
End Sub
Private Sub CommandButton1_Click()
  If Len(TextBox1.Value) = 6 Then
    Cells(Rows.Count, 2).End(xlUp).Offset(1, 0) = TextBox1.text
    ActiveCell.Offset(1, 0).Activate
    TextBox1 = ""
  End If
  Me.TextBox1.SetFocus
End Sub
Private Sub CommandButton2_Click()
  Unload Me
End Sub
```

step 4 按<F5>键运行窗体，在文本框中输入字母或者汉字，可以发现输入的字母和汉字后都会被自动清除，只有输入数字后才能保留下来。不过当数字超过 6 位时仅保留前 6 位。

step 5　输入 6 位数值 386245，然后按<Enter>键，焦点会自动转到“确定”按钮上，再次按<Enter>键则会将文本框中的学号写入到 B 列第一个空白单元格中，同时清除文本框中的值，并且将焦点转移到文本框中等待输入下一个学号，效果如图 17.27 所示。

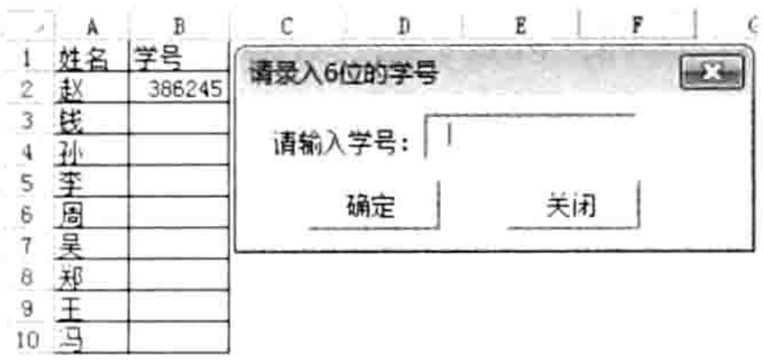


图 17.27　将学号导入到单元格后清空文本框

step 6　任意时候按下键盘上的<Esc>键可以关闭窗体，和单击“关闭”按钮的功能一致。

思路分析：

让文本框中只能输入 6 位数值主要包含两个步骤：

其一是用户每输入一个字符就在 TextBox1_Change 事件中判断一次用户输入的字符是否为数值，如果不是则删除最后输入的那个字符，如果是则删除第 6 位以后的字符。

其二是在 CommandButton1_Click 事件中计算文本框中的字符是否为 6 位，如果不是 6 位则将焦点返回文本框中等待用户输入新的字符，直到文本框中的字符有 6 位。

步骤一的目的是排除字母、汉字、标点符号，以及第 6 位以后的字符，步骤二的目的是排除学号少于 6 位的情况。两个过程套用才能确保输入的学号一定是 6 位数值。

尽管通过单元格的有效性设置也可以限制用户只能输入 6 位数值，但是它有两个缺点，其一是不能每输入一个字符就判断一次，其二是有效性设置容易被破坏，在单元格中粘贴数据时会清除有效性设置。

本例代码“CommandButton2.Cancel = True”的功能是将名为 CommandButton2 的命令按钮设置为本窗体中的 Cancel 按钮。所谓的 Cancel 按钮是指用户按<Esc>键时可以调用的按钮。通常将该按钮设置为关闭窗体的按钮，从而任何时候按<Esc>键都可以关闭窗体。

本例的窗体事件代码中仅设置了限制输入 6 位学号，事实上还可以限制输入的学号不能重复，读者可以试着练习在过程中添加判断学号是否重复的语句。

语法补充：

（1）TextBox1_Change 事件是在文本框中输入字符时触发的事件，每输入一个字符触发一次，每删除一个字符也会触发一次。本事件没有参数。

（2）SetFocus 方法的功能是设置焦点，它可将焦点转移到指定的按钮之上。其语法如下：

```
object.SetFocus
```

其中 object 代表控件名称，不过并非所有控件都有获得焦点的能力，标签就不可以。

本例文件参见光盘：..\第十七章\ 17-12 让文字框仅能录入 6 位数字.xlsm

17.4.10　运行窗体期间用鼠标调整文字框大小

案例要求：设计一个带有两个文字框的窗体，用于输入个人简历和求职意向。由于不同用户的简历长短不同，因此要求运行窗体期间也可以随意修改文本框的宽度。

操作步骤：

step 1　单击菜单中的“插入”→“用户窗体”命令从而创建一个空白窗体。

step 2 在窗体中插入两个标签、两个文本框和两个命令按钮，并且按以下方式布局，如图 17.28 所示。

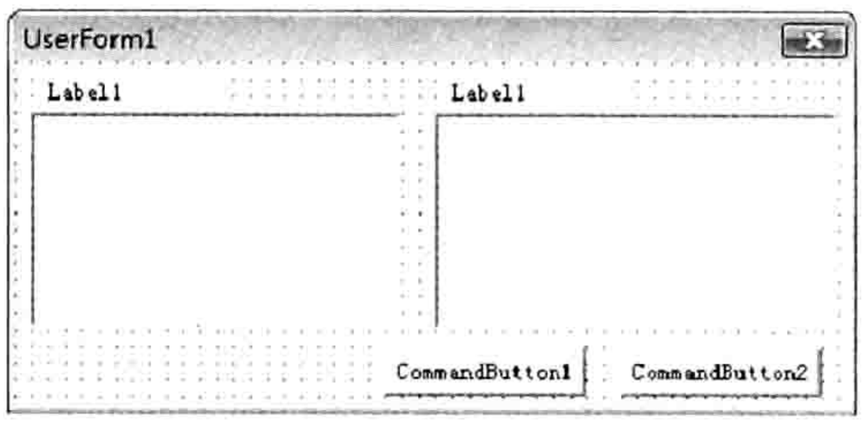


图 17.28　窗体控件布局

step 3 双击窗体进入代码窗口中，删除自动产生的代码，然后输入以下新代码：

```
'①代码存放位置：窗体代码窗口中②随书光盘中有每一句代码的含义注释
Private Sub UserForm_Activate()
  Label1.Caption = "自我介绍"
  Label2.Caption = "求职意向"
  CommandButton1.Caption = "导入到工作表"
  CommandButton2.Caption = "关闭窗体"
  Me.Caption = "应聘书"
End Sub
Private Sub UserForm_MouseMove(ByVal Button As Integer, ByVal Shift As Integer,
ByVal x As Single, ByVal Y As Single)
  On Error Resume Next
  Dim b As Long
  If Button = 1 Then
    With TextBox1
      b = .Width
      If x > Me.Width - 40 Or x < 40 Then Exit Sub
      TextBox1.Move .Left, , x - .Left
      TextBox2.Move x + TextBox2.Left - (.Left + b), , TextBox2.Width - (.Width
- b)
      Me.Label2.Left = Me.TextBox2.Left
    End With
  End If
End Sub
```

step 4 按<F5>键运行窗体，然后在窗体的空白区域按下鼠标左键不放并移动，窗体中的两个文本框会随鼠标的移动而相应地改变宽度，效果如图 17.29 所示。

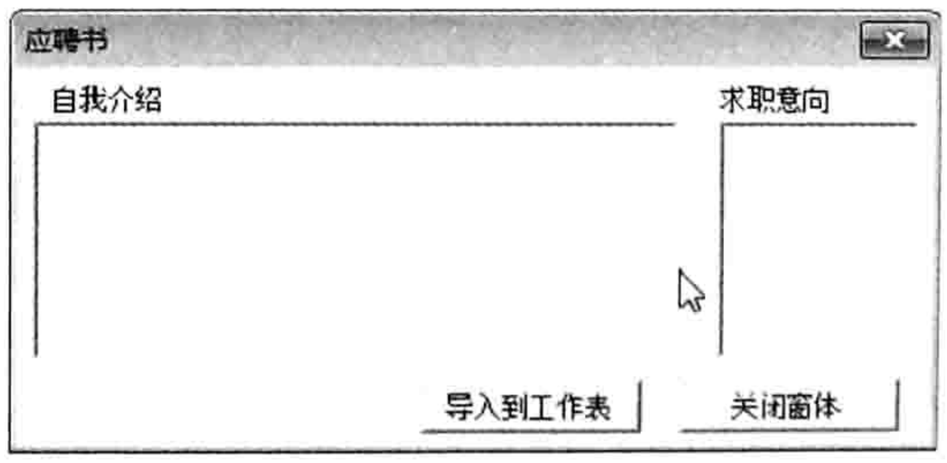


图 17.29　按下鼠标左键不放并且移动时改变两个文本框的宽度

思路分析：

文本框控件在窗体运行状态下不可以用鼠标修改其高度、宽度等属性，不过由于窗体的 UserForm_MouseMove 事件可以记录鼠标指针的位置变化，从而获得鼠标指针的移动距离，因此

在 UserForm_MouseMove 事件中以鼠标的横坐标 X 去调整两个文本框控件的宽度或者左边距即可实现本例需求。

本例中代码“TextBox1.Move .Left, , x - .Left”的作用是修改第一个文本框的宽度，代码“TextBox2.Move x + TextBox2.Left - (.Left + b), , TextBox2.Width - (.Width - b)”则是既修改第二个文本框的左边距又同步修改文本框的宽度。

语法补充：

控件的 Move 方法用于调整控件的大小和位置，它能同时修改控件的左边距、上边距、宽度与高度，具体语法如下：

```
object.Move( [Left [, Top [, Width [, Height [, Layout]]]]])
```

其中前 4 个参数根据参数名称即可明白其含义，第 5 个参数的功能是用于控制 Move 方法是否会触发 Layout 事件，赋值为 True 可以触发 Layout 事件，默认值为 False。

本例文件参见光盘：..\第十七章\ 17-13 鼠标调整文字框宽度.xlsm

17.4.11 为窗体中所有控件设置帮助

案例要求：在窗体中设计一个标签控件，用于显示所有控件的帮助信息，标明每个控件的用途。而且该帮助信息会随鼠标的移动而相应地变化。

操作步骤：

step 1 设计一个窗体，窗体中有三个选项按钮、两个复选框、一个列表框和一个标签控件，并且将它们按图 17.30 所示的方式布局。

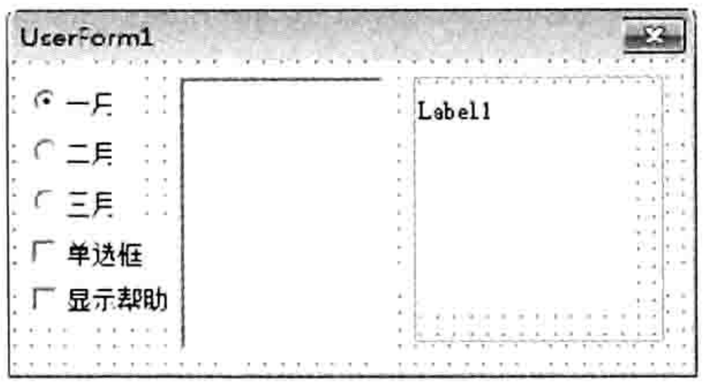


图 17.30 窗体中的控件布局

step 2 双击窗体进入窗体的代码窗口，删除自动产生的代码然后录入以下新代码：

```
'①代码存放位置：窗体代码窗口中②随书光盘中有每一句代码的含义注释
Private Sub UserForm_Activate()
  Me.Width = 150
  Me.Caption = "按需显示学员表"
  ListBox1.RowSource = "A1:A" & Cells(Rows.Count, 1).End(xlUp).Row
End Sub
Private Sub CheckBox2_MouseMove(ByVal Button As Integer, ByVal Shift As Integer,
ByVal X As Single, ByVal Y As Single)
  If CheckBox2.Value Then Label1.Caption = "控制是否显示帮助，打钩时显示帮助，否则
不显示"
End Sub
Private Sub CheckBox1_MouseMove(ByVal Button As Integer, ByVal Shift As Integer,
ByVal X As Single, ByVal Y As Single)
  If CheckBox2.Value Then Label1.Caption = "打钩时列表框将显示单选框，否则不显示单
选框。"
```

```
End Sub
Private Sub ListBox1_MouseMove(ByVal Button As Integer, ByVal Shift As Integer,
ByVal X As Single, ByVal Y As Single)
  If CheckBox2.Value Then Label1.Caption = "用于显示学员资料，随时单选框相应地变化。"
End Sub
Private Sub OptionButton1_MouseMove(ByVal Button As Integer, ByVal Shift As
Integer, ByVal X As Single, ByVal Y As Single)
  If CheckBox2.Value Then Label1.Caption = "单击时显示一班的学员资料。"
End Sub
Private Sub OptionButton2_MouseMove(ByVal Button As Integer, ByVal Shift As
Integer, ByVal X As Single, ByVal Y As Single)
  If CheckBox2.Value Then Label1.Caption = "单击时显示二班的学员资料。"
End Sub
Private Sub OptionButton3_MouseMove(ByVal Button As Integer, ByVal Shift As
Integer, ByVal X As Single, ByVal Y As Single)
  If CheckBox2.Value Then Label1.Caption = "单击时显示三班的学员资料。"
End Sub
Private Sub UserForm_MouseMove(ByVal Button As Integer, ByVal Shift As Integer,
ByVal X As Single, ByVal Y As Single)
  Label1.Caption = ""
End Sub
Private Sub OptionButton1_Click()
  Me.ListBox1.RowSource = "A1:A" & Cells(Rows.Count, 1).End(xlUp).Row
End Sub
Private Sub OptionButton2_Click()
  Me.ListBox1.RowSource = "B1:B" & Cells(Rows.Count, 1).End(xlUp).Row
End Sub
Private Sub OptionButton3_Click()
  Me.ListBox1.RowSource = "C1:C" & Cells(Rows.Count, 1).End(xlUp).Row
End Sub
Private Sub CheckBox1_Click()
  If CheckBox1.Value Then
    ListBox1.ListStyle = fmListStyleOption
  Else
    ListBox1.ListStyle = fmListStylePlain
  End If
End Sub
Private Sub CheckBox2_Click()
  If Me.Width = 150 Then Me.Width = 260 Else Me.Width = 150
End Sub
```

step 3 按<F5>键运行窗体，窗体会默认显示为如图 17.31 所示的状态。

step 4 单击“二月”选项按钮，以及“单选框”、“显示帮助”两个复选框，然后将鼠标指针移到“二月”之上，此时在右边的标签控件中会显示与“二月”控件相关的帮助信息，效果如图 17.32 所示。

step 5 将鼠标指针指向“单选框”，标签控件会显示为如图 17.33 所示的效果。

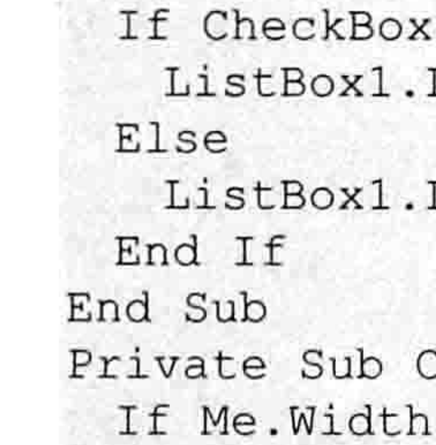


图 17.31 默认状态

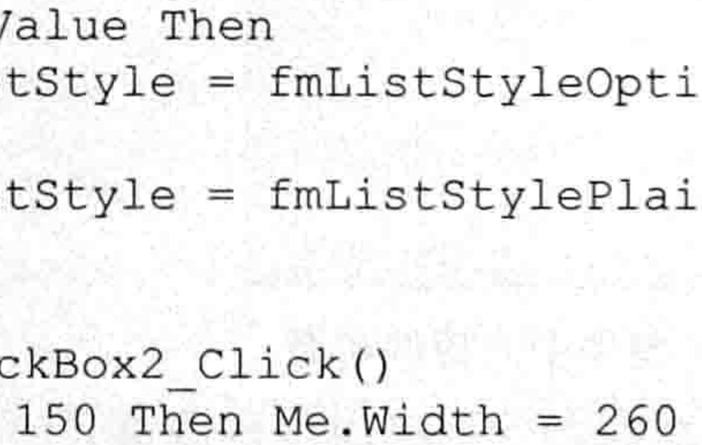


图 17.32 显示“二月”控件的帮助信息

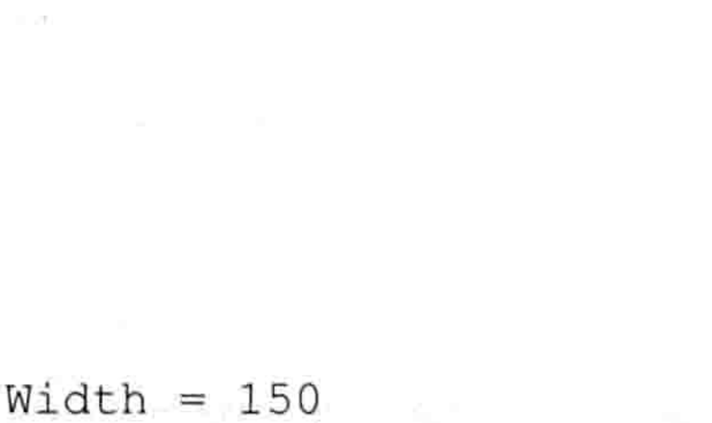


图 17.33 显示“单选框”控件的帮助信息

思路分析：

为控件指定帮助信息的重点在于在窗体中添加一个存放帮助信息的标签，然后为每个控件添加MouseMove事件，通过该事件将当前控件的帮助信息赋值给标签控件的Caption属性即可。为了避免鼠标指针离开控件后仍然显示该控件的帮助信息，还需要在窗体的MouseMove事件中清空标签的Caption属性值。

语法补充：

CheckBox1.Value表示复选框的勾选状态，当复选框处于勾选状态时则该值为True，否则该值为False。在判断复选框的值是否为True时不用完整地书写为“If CheckBox1.Value = True Then”，代码“If CheckBox1.Value Then”可以实现与之完全相同的功能。

本例文件参见光盘：..\第十七章\17-14 为控件设置帮助信息.xlsm

17.5 窗体的综合应用案例

17.4节中展示了诸多窗体事件与控件事件的应用，案例比较简单，本例会通过以下几个案例演示更复杂的窗体应用。

17.5.1 设计登录界面

案例要求：为工作簿设计一个启动Logo，打开工作簿时不显示Excel应用程序窗口只显示该Logo。5秒钟后关闭Logo，当然也允许在5秒之内按<Esc>键关闭Logo。

实现步骤：

step 1 单击菜单中的“插入”→“窗体”命令，并将窗体的Caption属性设置为空文本，将名称属性设置为Logo。

step 2 根据实际需求设计一张与公司相关的背景图片，通常包括公司名称、外景、地址、电话、传真等信息，然后将窗体的Picture属性设置为该图片的地址。

step 3 在窗体底部插入一个Flash控件，将其Movie属性设置为预先设计好的Flash动画文件的完整路径，并且将EmbedMovie属性设置为True，表示将Flash文件嵌入工作簿中。

step 4 将窗体拉大，然后从工具箱中将命令按钮拖到窗体中，在属性窗口中将命令按钮的Cancel属性赋值为True，再将窗体拉窄从而隐藏该按钮。

step 5 双击窗体进入代码窗口，删除自动产生的代码再录入以下新的代码：

```
Private Sub CommandButton1_Click()'代码存放位置：窗体代码窗口中
  Call 关闭
End Sub
```

step 6 单击菜单中的“插入”→“模块”命令，然后在模块中录入以下代码：

```
Sub Auto_Open()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Application.Visible = False
  Logo.Show 0
  Application.OnTime Now + TimeValue("00:00:05"), "关闭"
End Sub
Sub 关闭()
  Application.Visible = True
```

```
    Unload Logo
End Sub
```

step 7 保存并重启工作簿，工作簿启动时将出现如图 17.34 所示的登录窗口，在关闭此 Logo 之前看不到 Excel 的主窗口。

图 17.34 自定义启动画面

step 8 按<Esc>键关闭 Logo，Excel 的主窗口立即呈现出来。如果用户不按<Esc>键，那么 5 秒后系统自动关闭窗体，进入工作表界面。

案例补充：

本例中的登录窗口是模仿 Excel 的 Logo 设计的，但不可能做到与 Excel 自带的 Logo 一致。首先是无法屏蔽 Excel 内置的 Logo 从而单独显示当前自定义 Logo。其次是自定义窗体无法在 Excel 工作簿开启之前运行，而是在 Excel 的主窗口显示之后（不到 1 秒）再运行 VBA 代码，从而隐藏 Excel 的主窗口，接着显示自定义的 Logo。

本例文件参见光盘：..\第十七章\17-15 设计启动 LOGO.xlsm

17.5.2 权限认证窗口

案例要求：为工作簿设计权限验证，设置三个用户名和三个密码，只有用户名与密码完全正确时才能打开工作簿。如果录入用户名或者密码错误三次则自动关闭工作簿。

实现步骤：

step 1 插入一个窗体，将其 Caption 属性修改为“权限验证”。

step 2 为了让窗体更美观，在属性窗口中为窗体的 Picture 属性设置一个背景图案。

step 3 在窗体上插入两个标签、两个文本框和一个命令按钮，并且按如图 17.35 所示的方式布局。

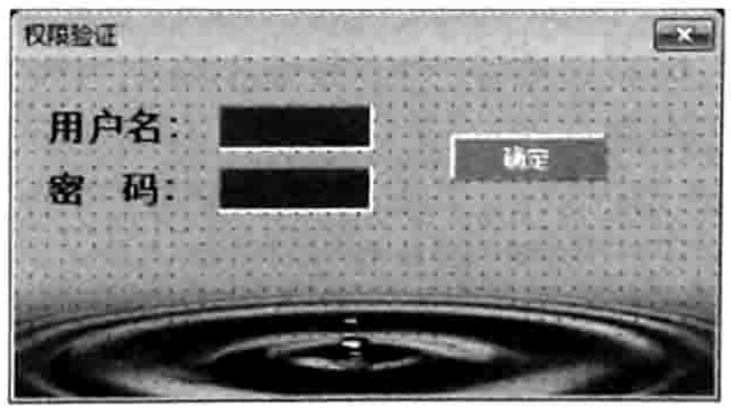


图 17.35 窗体控件布局

step 4 双击窗体进入窗体的代码窗口，删除自动产生的代码，然后录入以下新代码：

```
'①代码存放位置：窗体代码窗口中②随书光盘中有每一句代码的含义注释
Private Sub CommandButton1_Click()
  Static i
  If Len(TextBox1) = 0 Then MsgBox "用户名不能为空！", vbInformation, "警告": Exit Sub
  If Len(TextBox2) = 0 Then MsgBox "密码不能为空！", vbInformation, "警告": Exit Sub
  If TextBox1 = "andy" And TextBox2 = 123 Or TextBox1 = "sky" And TextBox2 =
456 Or TextBox1 = "andysky" And TextBox2 = 789 Then
    Unload Me
    Application.Visible = True
    Sheets(1).Activate
    Application.EnableCancelKey = xlInterrupt
  Else
    MsgBox "密码与用户名不匹配，请重新输入!", vbInformation
    i = i + 1
    If i >= 3 Then
      MsgBox "您已尝试三次错误，程序即将关闭！"
      Unload Me
      Application.Visible = True
      ThisWorkbook.Close False
    End If
  End If
End Sub
Private Sub UserForm_QueryClose(Cancel As Integer, CloseMode As Integer)
  If CloseMode <> 1 Then Cancel = True
End Sub
```

其中第一段代码用于验证用户是否正确输入了用户名与密码，如果未输入或者输入不正确则产生相应的提示。如果错误输入三次则自动关闭工作簿，如果用户和密码正确则关闭窗体进入第一个工作表中。

其中静态变量 i 用于记录用户输入错误的次数，当错误输入三次后工作簿会自动关闭。

第二个过程用于禁用窗体的关闭按钮，防止用户手工关闭验证窗口。

step 5 单击菜单中的“插入”→“模块”命令，然后在模块中录入以下代码：

```
Sub auto_Open()  '①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Application.EnableCancelKey = xlDisabled
  Application.Visible = False
  Userform1.Show
End Sub
```

以上过程表示启动工作簿时禁止用户使用<Ctrl+Break>组合键中断验证程序，然后隐藏 Excel 程序的主窗口，显示验证窗口。

step 6 保存并重新启动工作簿，VBA 代码会隐藏 Excel 工作簿窗口只显示“权限验证”窗口。如果单击右上角的关闭按钮不会产生任何反应，如果直接单击“确定”按钮会提示“用户名不能为空”，如果将用户名设置为 sky，将密码设置为 456，然后单击“确定”按钮则可以正常打开工作簿。

案例补充：

权限验证是极其简单的，仅仅使用 If Then 语句即可完成。

如果用户输入错误密码三次则关闭窗体、关闭工作簿，其重点在于静态变量 i，静态变量的特性是在过程结束后不会清除变量的值，因此每录入一次用户名和密码就会累加一次变量的值，从而记录用户的错误次数。

本例文件参见光盘：..\第十七章\17-16 权限验证(用户 andy 密码 123).xlsm

17.5.3 设计计划任务向导

案例要求：设计一个关于计划任务的向导。

在窗体中设计一个多页控件，每页进行一个方面的设置，最后根据窗体中的设置来完成一个任务。本例中的计划任务是在指定时间注销计算机或者关闭、重启计算机。

实现步骤：

step 1 插入一个模块，并在模块中输入以下代码，表示声明三个公共变量：

```
Public 时间类型 As Byte, 任务类型 As Byte, 时间 As String
```

step 2 插入一个窗体，并将其 Caption 属性设为“计划任务”。

step 3 在窗体中插入多页控件，并在其标题栏中通过单击鼠标右键新建四个空白页，然后将这四页更名为“说明”、“时间”、“任务种类”、“执行”，效果如图 17.36 所示。

step 4 返回多页控件的第一页，在其中插入一下标签，录入一些说明性的文字，表示本工具的用途。然后插入一个命令按钮，将其 Caption 属性设置为“开始→”，效果如图 17.37 所示。

图 17.36 多页控件

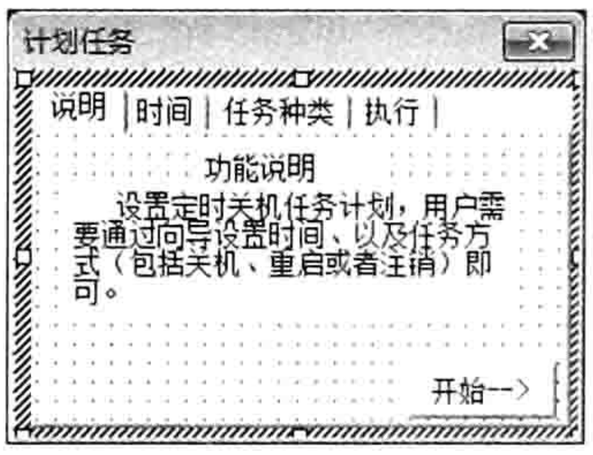


图 17.37 首页控件布局

step 5 双击命令按钮，输入以下代码，表示单击此按钮可以进入下一页：

```
Private Sub CommandButton1_Click()'代码存放位置：窗体代码窗口中
  Me.MultiPage1.Value = 1
End Sub
```

step 6 在第二页顶部插入标签、选项按钮、文本框和命令按钮，并按如图 17.38 所示的方式布局。

step 7 双击“下一步”按钮，输入以下代码：

```
'①代码存放位置：窗体代码窗口中②随书光盘中有每一句代码的含义注释
Private Sub CommandButton2_Click()
  If isDate(TextBox1.Text) Then
    时间 = TextBox1.Text
    MultiPage1.Value = 2
  End If
End Sub
Private Sub OptionButto1_Click()
    时间类型 = 1
End Sub
Private Sub OptionButton2_Click()
```

```
    时间类型 = 2
End Sub
```

第一段代码表示单击命令按钮时将用户录入的时间值赋予变量“时间”，并进入下一页；第二、第三段代码表示单击选项按钮 OptionButton1 时将变量“时间类型”赋值 1，否则赋值 2。

step 8 进入第三页，在其中添加标签、分组框、选项按钮和命令按钮，并按如图 17.39 所示的方式布局。

step 9 双击“下一步”按钮，然后输入以下代码：

```
'①代码存放位置：窗体代码窗口中②随书光盘中有每一句代码的含义注释
Private Sub CommandButton3_Click()
  Me.MultiPage1.Value = 3
  Label7.Caption = "  您指定的任务是：" & IIf(时间类型 = 1, 时间& "之后", "在" &时
间& "时间") & IIf(任务类型 = 1, "关机", IIf(任务类型 = 2, "重启", "注销")) & Chr(13)
& Label7.Caption
End Sub
Private Sub OptionButton3_Click()
    任务类型 = 1
End Sub
Private Sub OptionButton4_Click()
    任务类型 = 2
End Sub
Private Sub OptionButton5_Click()
    任务类型 = 3
End Sub
```

step 10 进入第四页，在其中添加一个标签和一个命令按钮，控件按如图 17.40 所示的方式布局。然后双击命令按钮进入代码窗口，并输入以下代码：

```
'①代码存放位置：窗体代码窗口中②随书光盘中有每一句代码的含义注释
Private Sub CommandButton4_Click()
    启动任务
  Unload Me
End Sub
```

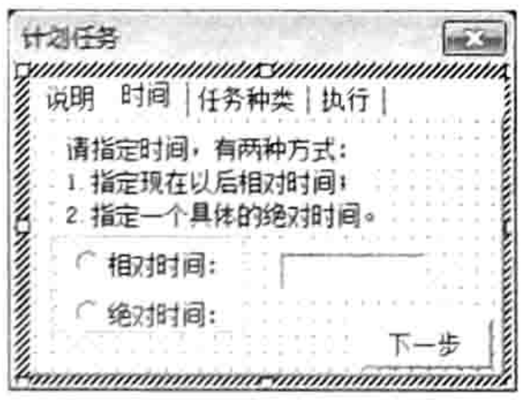


图 17.38 第二页控件布局

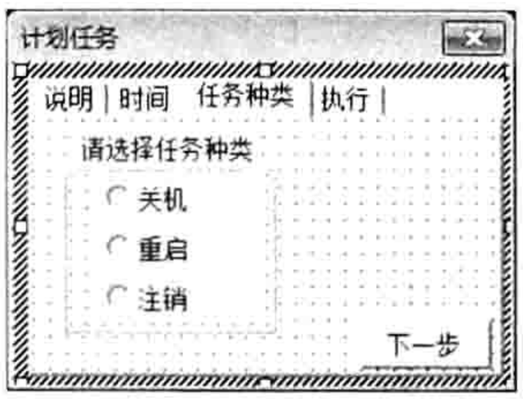


图 17.39 第三页控件布局

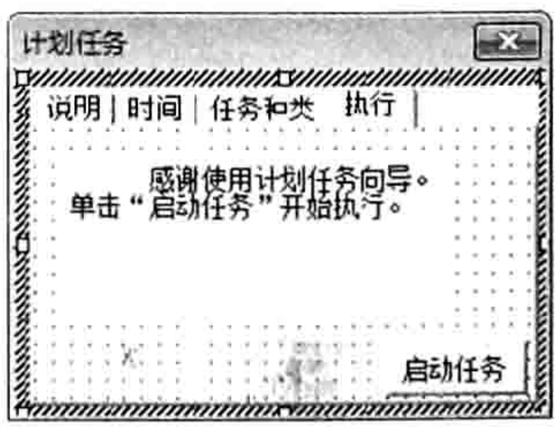


图 17.40 第四页控件布局

step 11 为了对窗体中的 4 个选项和两个公共变量设置为默认值，还需要在窗体事件代码窗口中输入以下代码：

```
'①代码存放位置：窗体代码窗口中②随书光盘中有每一句代码的含义注释
Private Sub UserForm_Activate()
  Me.MultiPage1.Value = 0
  Me.MultiPage1.Style = fmTabStyleNone
  OptionButton1.Value = True
  OptionButton3.Value = True
```

```
    时间类型 = 1
    任务类型 = 1
End Sub
```

其中代码“MultiPage1.Style = fmTabStyleNone”表示不显示多页控件的按钮，避免没有设置选项就可以直接进入最后一页中。

step 12 在模块中继续输入以下两段代码，用于指定任务向导相关联的任务过程：

```
Sub 启动任务()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  If 时间类型 = 1 Then
    Application.OnTime Now + TimeValue(时间), "任务"
  Else
    Application.OnTime TimeValue(时间), "任务"
  End If
End Sub
Sub 任务()
  Select Case 任务类型
  Case 1
    Shell "shutdown -s -t 1"
  Case 2
    Shell "shutdown -r -t 1"
  Case 3
    Shell "shutdown -l -t 1"
  End Select
End Sub
```

“启动任务”过程用于设置启动任务的时间，根据变量“时间类型”的值决定采用相对时间还是绝对时间；“任务”过程则用于指定任务的具体内容，包括关机、重启和注销，执行哪一类任务由变量“任务类型”的值所决定。

step 13 按<F5>键启动窗体，默认将显示第一页。单击“开始→”按钮进入第二页，保持默认选项为“相对时间”，然后在时间框中输入“00:00:05”，表示 5 秒之后执行任务，设置界面如图 17.41 所示。

step 14 进入第三页，单击“重启”选项按钮，然后单击“下一步”按钮进入最后一页，在最后一页中单击“启动任务”按钮，那么在 5 秒后计算机会重启。

图 17.41 至图 17.43 是计划任务在运行期间的第二、第三、第四页的操作界面。

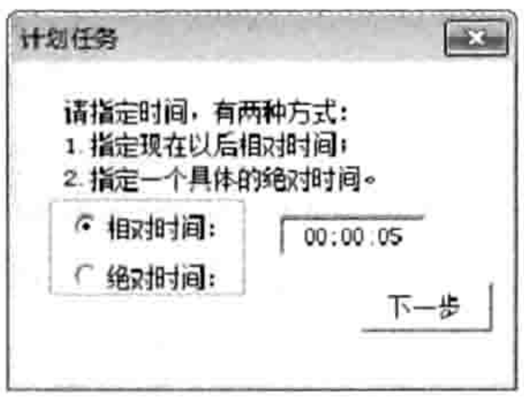


图 17.41 指定任务时间

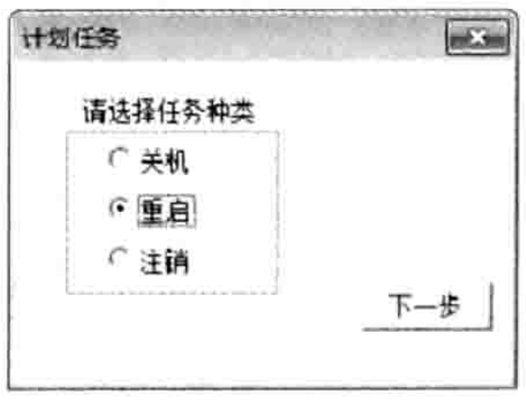


图 17.42 指定任务种类

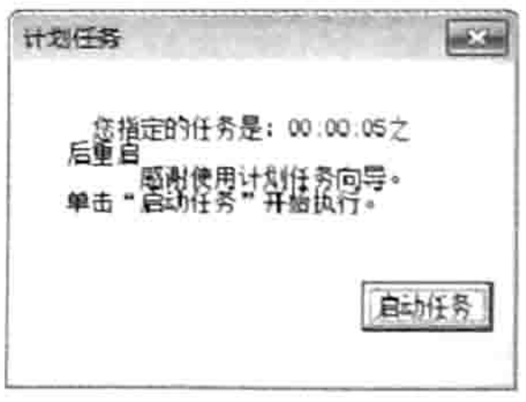


图 17.43 启动任务

案例补充：

创建计划任务的技术要求比较简单，只要掌握了 Application.OnTime 方法的语法即可，而真正的重点在于如何设计一个简单明了的操作向导，让用户不需要专业的知识，也不需要太多的思考就可能完成任务。

本案例的需求比较简单，不过向导的设计思路值得读者借鉴。

本例文件参见光盘：..\第十七章\ 17-17 设计计划任务向导.xlsm

17.5.4 设计动画帮助

案例要求：在窗体中显示滚动文字，为软件提供帮助信息。

实现步骤：

step 1 插入一个窗体，将其 Caption 属性修改为“关于”。

step 2 在工具箱中单击鼠标右键，在弹出的菜单中选择“附加控件”命令，然后在“附加控件”对话框中将“Microsoft Web Browser”控件打钩。

step 3 将工具箱中的 Web Browser 控件拖到窗体中，并将它的大小调整为窗体的大小。

step 4 双击 Web Browser 控件打开代码窗口，并输入以下代码：

```
'①代码存放位置：窗体代码窗口中②随书光盘中有每一句代码的含义注释
Private Sub UserForm_Activate()
  Dim str As String
  str = "本工具的功能如下：<P>①：.....<P>②：.....<P>③：.....<P>④：.....<P>
有建议请发送邮件到：<P><a href='mailto:888@excelbbx.com'>单击反馈意见</a>"
  WebBrowser1.Navigate "about:blank"
  WebBrowser1.Document.writeln "<html><body bgcolor=#00FF00  style='border:
none;overflow:hidden;margin:0'   oncontextmenu='return  false'><p><marquee
direction=up scrollamount='4'>" & str & "</marquee></p></body></html>" '向网
页中写入网页代码
End Sub
```

以上过程中 WebBrowser1 是 Web 控件的名称，“about:blank”表示将网页初始化为空白页。Document.writeln 方法用于向空白网页中写入网页代码，网页代码的内容是创建滚条的文字。

网页代码中“bgcolor=#00FF00”表示对网页设置背景色，可以随意修改颜色。“direction=up”表示动画文字从下向上滚动，“crollamount='4'”表示滚动速度，用户可以根据需求调整速度。

网页代码中的“<P>”代表换行，如果滚动文字需要显示为 X 行就插入 X-1 个“<P>”。

step 5 按<F5>键执行窗体，网页中的文字会从下向上滚动，如图 17.44 所示。当窗体的文字滚动到最后一行时会显示“单击反馈信息”字符串，如果单击该字符串将启动邮件发送窗口。

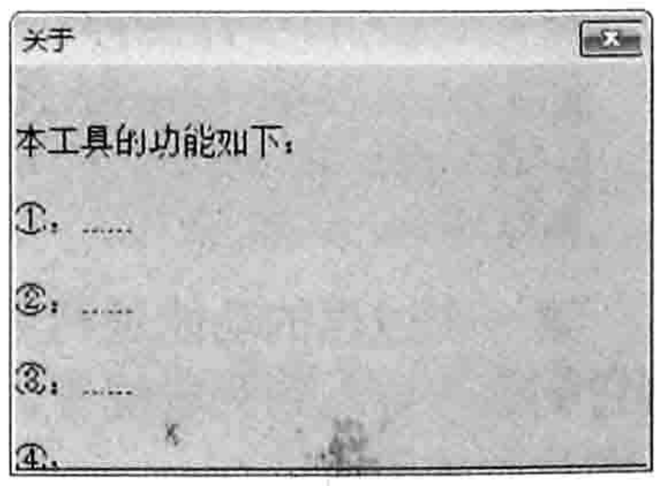


图 17.44 窗体中的动画文字

案例补充：

仅使用 VBA 的循环语句也可以在窗体中产生动态文字，不过这会占用太多的内存资源，影响 Excel 的其他操作。通过网页中的“<marquee>滚动字符</marquee>”创建可滚动显示的字符只耗用极少的资源，是动画文字的首选。

本例文件参见光盘：..\第十七章\ 17-18 设计动画帮助.xlsm

17.5.5 用窗体浏览图片

案例要求：通过窗体浏览任意文件夹中的图片。

实现步骤：

step 1 单击菜单中的“插入”→“用户窗体”命令，然后将 Caption 属性设置为“图片预览”。

step 2 在窗体左侧添加一个列表框控件，在右侧添加一个图像控件，并且按如图 17.45 所示的方式布局。

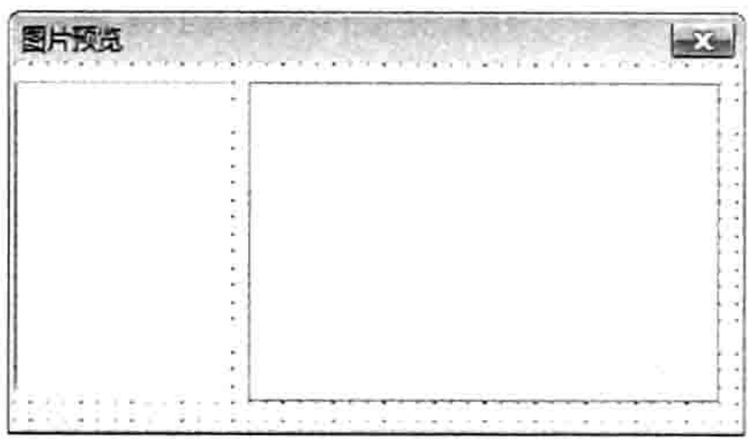


图 17.45 窗体控件布局

step 3 双击窗体进入代码窗口，并输入以下代码：

```
'①代码存放位置：窗体代码窗口中②随书光盘中有每一句代码的含义注释
Private Sub UserForm_Initialize()
  Dim FileName As String, n As Long
  ListBox1.BackColor = &HFFFF80
  Me.BackColor = 16761024
  FileName = Dir(PathStr & "*.jpg")
  Do
    If Len(FileName) = 0 Then Exit Do
    ListBox1.AddItem FileName
    n = n + 1
    FileName = Dir()
  Loop
  If n > 0 Then
    Image1.Picture = LoadPicture(PathStr & Dir(PathStr & "*.jpg"))
    Image1.PictureSizeMode = fmPictureSizeModeStretch
  Else
    MsgBox "选定的目录下不存在 Jpg 图片。"
    End
  End If
End Sub
Private Sub ListBox1_Click()
  Image1.Picture = LoadPicture(PathStr & ListBox1.Text)
End Sub
```

第一个过程的功能是利用 Do Loop 循环加 Dir 函数获取 PathStr 路径下的所有 jpg 图片名称，将它们逐一导入到列表框中，然后在窗体的图像控件中显示出第一张图片内容。如果在该路径下没有图片则关闭窗体。

第二个过程的功能是将列表框中选中的图片导入到图像控件中。

step 4 为了让用户可以随意选择图片路径，单击菜单中的“插入”→“模块”命令，并录入以下代码：

```
Public PathStr As String
Sub 图片浏览()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  With Application.FileDialog(msoFileDialogFolderPicker)
    If .Show = -1 Then PathStr = .SelectedItems(1) Else Exit Sub
    PathStr = PathStr & IIf(Right(PathStr, 1) = "\", "", "\"
```

```
    UserForm1.Show
  End With
End Sub
```

以上过程中 PathStr 是公共变量，在模块和窗体中都可以调用。过程“图片浏览”的功能是弹出对话框让用户指定图片路径，将路径赋值给变量 PathStr。同时，由于变量 PathStr 是公共变量，因此窗体的事件过程可以调用此变量的值。

step 5 执行过程“图片浏览”，程序会弹出一个“浏览”对话框，当用户选择了保存图片的文件夹后（光盘中准备了一个“图片”目录，里面有大量的 JPG 图片）会弹出“图片预览”窗体，默认显示用户选择的文件夹中的第一张图片。当用户单击左侧列表框中的任意图片名称时，右侧图像控件会显示相应的图片，效果如图 17.46 所示。

图 17.46 利用窗体预览图片

案例补充：

用窗体浏览图片时，由于图片的大小和长宽比例不尽相同，因此应将图像控件的 PictureSizeMode 调整为 fmPictureSizeModeZoom 或者 fmPictureSizeModeStretch，默认状态是 fmPictureSizeModeClip。其中 fmPictureSizeModeStretch 代表将图片缩放到图像控件的大小，但图片可能会变形，fmPictureSizeModeZoom 表示将图片缩放到图像控件的大小，图片的长宽比例保持一致，不会变形。

CD-ROM 本例文件参见光盘：..\第十七章\ 17-19 利用窗体预览图片.xlsm

17.5.6 设计多表录入面板

案例要求：当需要向多个工作表中输入值并且需要不停切换工作表时，直接在工作表中手动输入是很不现实的，效率太低。例如在图 17.47 中需要输入五个班的学员缴费数据，由于收费时并非按班级统一执行，而是按学员到场的时间先后顺序输入，因此可能每输入一次都要切换一次工作表。现要求利用 VBA 设计一个输入面板，让程序自动查找工作表并将数据输入到其中。

本例中假设收费标准为 800 元，在输入学生缴来的学费后，则自动将资料导出到相应的班级工作表中；同时查看缴费是否完整，如果缴费低于 800 元，则在欠费工作表中一一罗列出来，方便后续统计。

实现步骤：

step 1 单击菜单中的“插入”→“用户窗体”命令，并将其 Caption 属性修改为“资料输入面板”。

step 2 在窗体中插入 5 个标签、1 个复合框、1 个命令按钮和 4 个文本框，并且按如图 17.48 所示的方式布局。

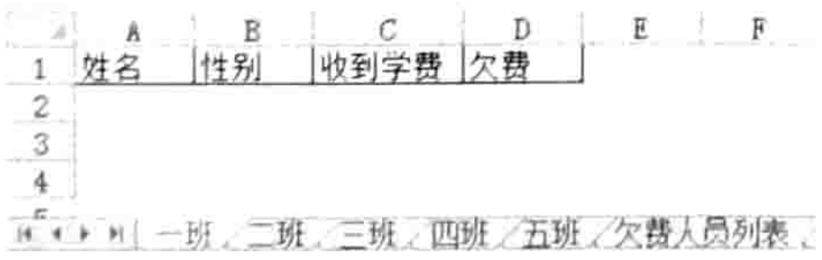


图 17.47 收费表分布图

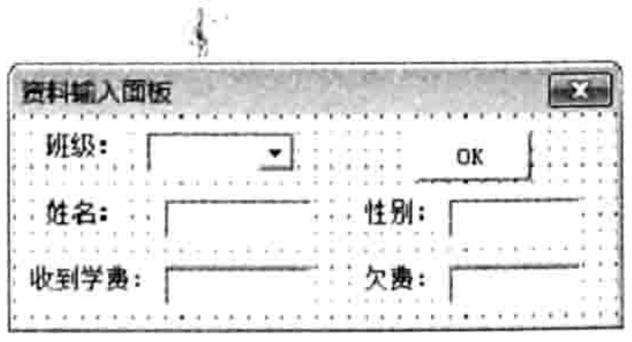


图 17.48 窗体控件布局

step 3 双击进入窗体的代码窗口，删除自动产生的代码，并录入以下新代码：

```
'①代码存放位置：窗体代码窗口中②随书光盘中有每一句代码的含义注释
Private Sub UserForm_Activate()
  ComboBox1.List = Array("一班", "二班", "三班", "四班", "五班")
  ComboBox1.Value = "一班"
End Sub
```

以上代码用于预设复合框的值和列表内容，它等于需要录入数据的所有工作表名称。

```
Private Sub TextBox3_Change()
  If Len(TextBox3) > 0 Then
    If Not IsNumeric(TextBox3) Then TextBox3 = ""
    If TextBox3.Value > 800 Or TextBox3.Value < 0 Then
      MsgBox "请检查后再输入"
      TextBox3 = ""
    Else
      TextBox4 = 800 - TextBox3.Value
    End If
  End If
End Sub
Private Sub TextBox3_Exit(ByVal Cancel As MSForms.ReturnBoolean)
  CommandButton1.SetFocus
End Sub
```

以上的TextBox3_Change事件过程表示在Textbox3中输入数值时检测用户输入的数值是否大于0且小于等于800，如果不是则提示用户，如果是则在Textbox4中显示欠费金额。

TextBox3_Exit事件过程用于转移焦点，因为在Textbox4中不需要输入数值。

```
'①代码存放位置：窗体代码窗口中②随书光盘中有每一句代码的含义注释
Private Sub CommandButton1_Click()
  If TextBox1 <> "" And TextBox2 <> "" And TextBox3 <> "" Then
    With Worksheets(ComboBox1.Value).Cells(Rows.Count, 1).End(xlUp)
      .Offset(1, 0) = TextBox1
      .Offset(1, 1) = TextBox2
      .Offset(1, 2) = TextBox3
      .Offset(1, 3) = TextBox4
      .Range("A1").CurrentRegion.Borders.LineStyle = xlContinuous
      Worksheets(ComboBox1.Value).Select
    End With
    If TextBox4.Value > 0 Then Worksheets(ComboBox1.Value).Cells(Rows.Count,
1).End(xlUp).Resize(1, 4).Copy Worksheets("欠费人员列表").Cells(Rows.Count,
1).End(xlUp).Offset(1, 0)
  Else
    MsgBox "所有文本框不能为空！", vbOKOnly + vbInformation, "提示"
  End If
  TextBox1 = ""
  TextBox2 = ""
  TextBox3 = 0
  TextBox4 = 0
  ComboBox1.SetFocus
End Sub
```

以上事件过程表示单击命令按钮时将4个文本框中的值写入到复合框所指定的工作表中，然后再计算是否欠费，如果欠费大于0则将这个单元格的值复制到“欠费人员列表”中。

step 4 按<F5>键运行窗体，窗体的复合框中默认显示为“一班”，4 个文本框中显示为空值。

step 5 按 3 次<↓>键从而将复选框中值的修改为“四班”，然后按<Enter>键将焦点转移到姓名文本框中，接着分别在姓名、性别、收到学费文本框中输入“刘新年”、“男”、“500”，而欠费文本框中会自动产生数值 300，效果如图 17.49 所示。

step 6 当在这 3 个文本框中完成数据输入后按两次<Enter>键，姓名、性别、收到学费和欠费 4 个数据会自动记录到工作表“四班”中，同时打开“四班”工作表，效果如图 17.50 所示。

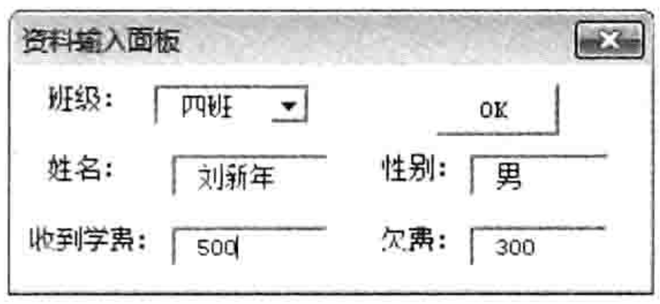


图 17.49 输入姓名、性别与学费

图 17.50 将窗体中的数据导出到工作表中

由于第一笔收费没有达到 800 元，因此在“欠费人员列表”工作表中会同步记录刘新年的缴费情况。

案例补充：

在一个固定的工作表中输入数据比较方便，但是在多个工作表中输入数据则比较烦琐，利用窗体输入数据可以有效解决此类问题。

在窗体中输入数据还可以对数据执行正确性校检、判断是否重复、判断是否为数值等过程。一个完善的窗体设计可以让输入数据更快捷、更准确。

本例文件参见光盘：..\第十七章\ 17-20 多工作表录入.xlsm

17.5.7 多条件高级查询

案例要求：工作簿中有多个工作表，每个工作表中都有学生的姓名、成绩与学号，如图 17.51 所示。现要求设计一个窗体，在窗体中对姓名、成绩与学号三者中任意一个执行模糊查询后，将查询结果罗列在列表框中。

实现步骤：

step 1 单击菜单中的“插入”→“用户窗体”命令，并将其 Caption 属性修改为“成绩查询”。

step 2 在窗体中添加一个复合框，在复合框右边添加一个文字框，用于输入查询条件，再在下方添加一个列表框，用于存放查找到的目标数据。控件按如图 17.52 所示的方式布局。

	A	B	C	D
1	姓名	成绩	学号	
2	张世后	94	022	
3	潘大旺	77	027	
4	刘玲玲	66	019	
5	张大中	80	016	
6	陈中国	53	010	
7	陈越	62	014	
8	周联光	78	026	
9	刘五强	52	005	

一班 二班 三班

图 17.51 成绩表

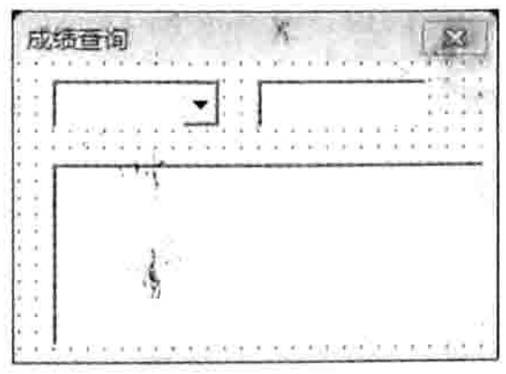


图 17.52 窗体控件布局

step 3 双击进入窗体的代码窗口，删除自动产生的代码，并录入以下新代码：

```
'①代码存放位置：窗体代码窗口中②随书光盘中有每一句代码的含义注释
Private Sub UserForm_Activate()
```

```
  ComboBox1.List = Array("姓名", "成绩", "学号")
  ComboBox1.Value = "姓名"
End Sub
Private Sub TextBox1_Change()
  ListBox1.Clear
End Sub
Private Sub TextBox1_Exit(ByVal Cancel As MSForms.ReturnBoolean)
  Dim arr(), sht As Worksheet, RowCount As Integer, Item As Integer, FindText
As String
  Item = 1
  For Each sht In Worksheets
  For RowCount = 2 To sht.Cells(Rows.Count, 1).End(xlUp).Row
    Select Case ComboBox1.Value
    Case "姓名"
      FindText = sht.Cells(RowCount, 1)
    Case "成绩"
      FindText = sht.Cells(RowCount, 2)
    Case "学号"
      FindText = sht.Cells(RowCount, 3)
    End Select
    If FindText Like "*" & TextBox1.Text & "*" Then
      Item = Item + 1
      ReDim Preserve arr(1 To 4, 1 To Item)
      arr(1, Item) = sht.Name
      arr(2, Item) = sht.Cells(RowCount, 1)
      arr(3, Item) = sht.Cells(RowCount, 2)
      arr(4, Item) = sht.Cells(RowCount, 3)
    End If
    Next RowCount
  Next sht
  If Item > 1 Then
    arr(1, 1) = "班级": arr(2, 1) = "姓名": arr(3, 1) = "成绩": arr(4, 1) = "
学号"
    ListBox1.ColumnCount = 4
    ListBox1.ColumnWidths = "30,40,30,40"
    ListBox1.List = WorksheetFunction.Transpose(arr)
  End If
  ComboBox1.SetFocus
End Sub
```

UserForm_Activate 事件过程用于设置复选框的列表内容和显示内容；TextBox1_Change 事件过程表示在文本框中输入数据时清空列表框中上一次的查找结果；TextBox1_Exit 事件在 TextBox1 控件失去焦点时触发，也就是输入查找对象后按<Enter>键时触发，该过程的功能是在活动工作簿的所有工作表中查找目标，然后将找到的所有数据保存在数组中，查找完成后将数组转置方向再赋予列表框。最后，为了节省鼠标单击复合框的时间，利用代码将焦点转移到复合框中，等待执行下一轮查找。

step 4 按<F5>键运行窗体，将查询类型设置为“姓名”，将查找对象设置为“张”，当按<Enter>键后可以得到如图 17.53 所示的查询结果。

step 5 将查询类型设置为“成绩”，将查询对象设置为“79”，当按<Enter>键后可以得到如图 17.54 所示的查询结果。

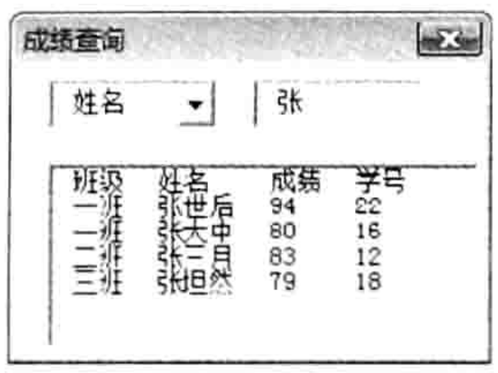


图 17.53　以姓名为查询条件

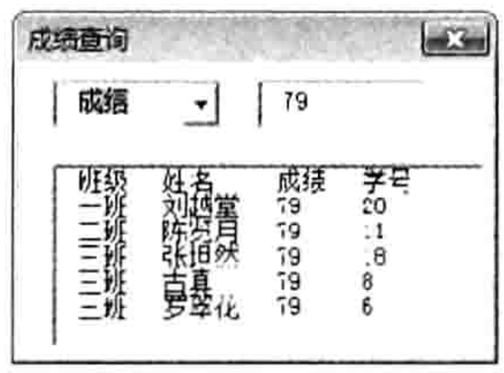


图 17.54　以成绩为查询条件

案例补充：

跨工作表查询对于 VBA 而言比较简单，在循环语句中使用 Like 运算符执行比较，然后将符合条件者写入数组中，循环完成后将数组的值显示在列表框中即可。

本例的重点和难点在于事件名称的判断过程。由于本例中未使用“确定”按钮，需要通过文本框的事件来执行命令，文本框有 13 个事件，各有不同的功能，熟悉每种事件的功能才能在需要时正确地判断该使用何种事件。

对于文本框的事件，在帮助中搜索“文本框事件”，然后单击“事件”菜单可以弹出与文本框相关的 13 个事件过程名称，单击事件过程名称可以查看详细的解释。

本例文件参见光盘：..\第十七章\ 17-20 跨工作表查询.xlsm

◆　◆　◆

第 18 章　处理文件与文件夹

FileSystemObject（也称 FSO），即文件系统对象，该对象有着强大的文件与文件夹管理能力。本章将全方位地展示 FileSystemObject 技术在文件与文件夹管理方面的应用。

18.1　认识 FSO 对象、属性与方法

FSO 对象有强大的文件与文件夹管理能力，在 VBA 中调用 FSO 对象的资源可以删除文件、复制文件、重命名文件、移动文件、获取文件的一切属性，也可通过 FSO 对象删除文件夹、创建文件夹，以及获取文件夹的体积、创建日期等信息。

对于日常工作中处理文件或者文件夹比较多的用户来说应该深入学习 FSO 对象的相关技术。

18.1.1　FSO 对象的调用方式

FSO 对象包含在 Scripting 类型库中，文件名为 Scrrun.Dll。

要调用 FSO 的资源首先得绑定对象的类型库。绑定类型库包含前期绑定和后期绑定两种方式，前期绑定的优点是书写代码时可以弹出属性与方法列表，缺点是将代码分享给他人使用时不够方便，需要对方手工添加引用；后期绑定的优点是代码分享给他人使用时可以直接使用，不需要其他操作，缺点是书写代码时没有属性与方法列表，不过可以通过做笔记的方法克服这个缺点——将编好的代码当作模板存放在笔记中，需要使用时复制出来即可，由于不需要手工编写代码，因此是否弹出属性与方法列表并不重要。

在此，笔者推荐读者在初学 VBA 时采用前期绑定方式，稍微熟练后采用后期绑定方式。

1. 前期绑定

前期绑定 FSO 对象的操作步骤如下：

step 1　在 VBE 窗口中单击菜单中的“工具”→“引用”命令。

step 2　在“引用”对话框中勾选“Microsoft Scripting Runtime”复选框，操作界面如图 18.1 所示。

step 3　单击菜单中的“插入”→“模块”命令，然后在模块中录入 FSO 相关的代码，之后会发现 VBA 会弹出 FSO 对象的属性与方法列表，效果如图 18.2 所示。

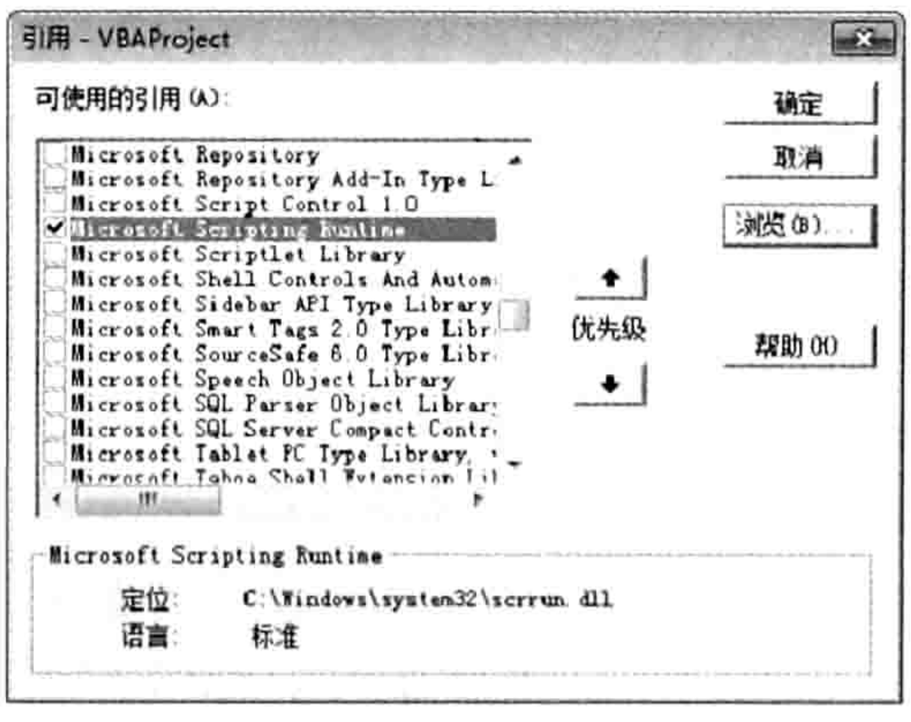


图 18.1　前期绑定 FSO

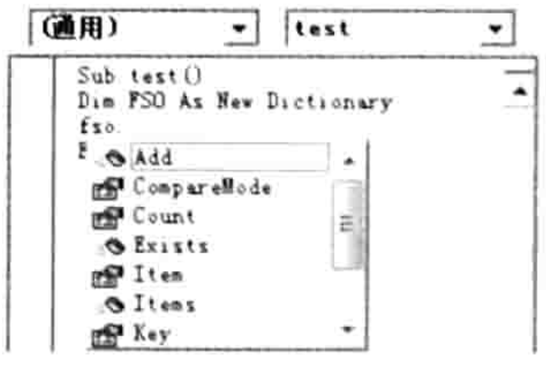


图 18.2　自动列出成员

2. 后期绑定

后期绑定是指利用 CreateObject 函数创建 FSO 对象并返回该对象的引用。FSO 的对象名称是“Scripting.FileSystemObject”，因此调用 FSO 对象的具体办法是通过“CreateObject("Scripting.FileSystemObject")”语句引用 FSO 对象，然后通过输入小圆点调用它的属性或者方法。例如 FSO 对象中的 FolderExists 方法用于判断文件夹是否存在，因此以下过程可以判断 D 盘中是否存在“生产日报表”这个文件夹：

```
Sub 判断文件夹是否存在()
  If CreateObject("Scripting.FileSystemObject").FolderExists("d:\生产日报表")
Then MsgBox "存在" Else MsgBox "不存在"
End Sub
```

18.1.2 FSO 的对象

FSO 提供了 5 个对象，VBA 调用 FSO 可以访问这 5 种对象。对象列表如表 18-1 所示。

表 18-1 FSO的对象

对象名称	功能描述
FileSystemObject 对象	操作计算机系统
Drive 对象	操作驱动器
Folder 对象	操作文件夹
File 对象	操作文件
TextStream 对象	操作文件的内容

虽然 FSO 包括 5 个对象，但 FileSystemObject 对象其实包含其他对象的所有属性与方法，仅仅熟练使用 FileSystemObject 对象也可以完成其他 4 个对象的所有功能，因此学习时可以将重心放在 FileSystemObject 对象之上。

18.1.3 FSO 常用对象的方法与属性

Drive 对象包含了磁盘的所有信息，它具有 12 个属性，所有属性见表 18-2。

表 18-2 Drive对象的属性

属性名称	含义解释
AvailableSpace	返回在驱动器或网络共享上的用户可用的空间容量
DriveLetter	返回本地驱动器或网络驱动器的字母
DriveType	返回驱动器的磁盘类型
FileSystem	返回驱动器使用的文件系统类型
FreeSpace	返回驱动器上或共享驱动器可用的磁盘空间
IsReady	确定驱动器是否准备好
Path	返回文件、文件夹或驱动器的路径
RootFolder	返回一个 Folder 对象，该对象表示一个驱动器的根文件夹的只读属性
SerialNumber	返回用于唯一标识磁盘卷标的十进制序列号

续表

属性名称	含义解释
ShareName	返回驱动器的网络共享名
TotalSize	以字节为单位，返回驱动器或网络共享的总空间大小
VolumeName	设置或返回驱动器的卷标名

Drives 对象包含磁盘对象集合，要获取单个磁盘的信息应该使用 For Each Next 循环语句遍历 Drives 对象，然后通过上述属性调用单个磁盘对象的若干信息。

例如要获取所有磁盘的盘符与剩余空间，以下代码可以实现（代码采用了后期绑定）：

```
'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
Sub 获取所有磁盘的盘符与剩余空间()
  On Error Resume Next
  Dim Drv As Object, Msg As String
  For Each Drv In CreateObject("Scripting.FileSystemObject").Drives
    Msg = Msg & Drv.DriveLetter & vbTab & Format(Drv.FreeSpace / 1024 ^ 3, "0.0G")
& Chr(13)
  Next
  MsgBox Msg, vbInformation, "本机磁盘信息"
End Sub
```

由于前面讲过 FileSystemObject 对象本身也具有获取磁盘信息的功能，那么不用 Drive 对象也足以完成同等功能。

假设要求使用 FileSystemObject 的方法与属性获得 D 盘的总空间与剩余空间，仍然采用后期绑定方式，那么可用以下代码实现：

```
Sub 获取 D 盘的总空间与剩余空间() '①代码存放位置：模块中②随书光盘中有每一句代码含义注释
  With CreateObject("Scripting.FileSystemObject").GetDrive("D:")
    MsgBox "总空间：" & .TotalSize & Chr(13) & "剩余空间：" & .FreeSpace,
vbInformation, "D 盘"
  End With
End Sub
```

代码中的 GetDrive 属于 FileSystemObject 对象的方法。

本例文件参见光盘：..\第十八章\ 18-1 获得所有磁盘的信息.xlsm

表 18-3 中罗列了 FileSystemObject 对象的关于磁盘应用的 3 个方法。

表 18-3　FileSystemObject关于磁盘应用的 3 个方法

方法名称（包含参数）	功能描述
DriveExists（drivespec）	判断磁盘是否存在，返回值为 True 时表示磁盘存在
GetDrive（drivespec）	返回指定磁盘的 Drive 对象的引用，GetDrive("D:")即引用 D 盘这个对象
GetDriveName（drivespec）	返回指定路径所在的磁盘的名称。参数必须是文件或文件夹的绝对路

表 18-4 中罗列了 FileSystemObject 对象的关于文件夹操作的 8 个方法。

表 18-4 FileSystemObject对象的关于操作文件夹操作的 8 个方法

方法名称（包含参数）	功能描述
CreateFolderfoldername	创建一个文件夹
DeleteFolder folderspec[, force]	删除一个文件夹
MoveFolder source, destination	移动一个文件夹
CopyFolder source, destination[, overwrite]	复制一个文件夹
FolderExists（folderspec）	判断文件夹是否存在，返回值为逻辑值
GetFolder（folderspec）	根据路径获得文件夹对象，返回值为对象
GetParentFolderName（path）	找出一个文件夹的父文件夹的名称，返回值为文本
GetSpecialFolder（folderspec）	找出系统文件夹的路径，返回值为文本

表 18-4 中的几个方法都比较简单，看一句代码即可明白各个方法的使用方法。

以下 4 个过程包含了 FolderExists、CreateFolder、DeleteFolder 和 CopyFolder 方法的应用：

```
Sub 判断文件是否存在()
  MsgBox  IIf(CreateObject("Scripting.FileSystemObject").FolderExists("D:\生
产表"), "存在", "不存在")
End Sub
Sub 创建文件夹()
  CreateObject("Scripting.FileSystemObject").CreateFolder ("D:\生产表")
End Sub
Sub 删除文件夹()
  CreateObject("Scripting.FileSystemObject").DeleteFolder ("D:\生产表")
End Sub
Sub 将 D 盘的文件夹复制到 E 盘()
  CreateObject("Scripting.FileSystemObject").CopyFolder "D:\生产表", "E:\" ,
True
End Sub
```

本例文件参见光盘：..\第十八章\18-2 操作文件夹.xlsm

表 18-5 中罗列了 FileSystemObject 对象的关于文件夹的 15 个属性。

表 18-5 FileSystemObject对象的关于文件夹的 15 个属性

属性名称	功能描述
ShortPath	返回 8.3 文件命名约定的短路径
Attributes	设置或者返回文件或文件夹的属性
Size	返回的文件或者文件夹的大小，以字节为单位
Path	返回指定文件、文件夹或驱动器的路径
Drive	返回指定文件或文件夹所在的驱动器符号
IsRootFolder	判断目录是否是根目录
Name	设置或返回指定文件或文件夹名
DateLastAccessed	返回最后一次访问文件或文件夹的日期和时间

续表

属性名称	功能描述
Files	返回 Files 集合，包括指定的文件夹中所有文件
DateLastModified	返回最后一次修改的日期和时间
DateCreated	返回创建日期和时间
ShortName	返回 8.3 命名约定的程序所使用的短名字
Type	返回关于某个文件或文件夹类型的信息
SubFolders	返回 Folders 集合，包括所有文件夹
ParentFolder	返回指定文件或文件夹的父文件夹对象

其中比较常用的属性包含 Attributes、Size、DateLastAccessed、Files、Files、SubFolders，在 18.2 节中会介绍相关的应用。

表 18-6 中罗列了 FileSystemObject 对象的关于文件操作的 11 个方法。

表 18-6　FileSystemObject对象的关于文件操作的 11 个方法

方法名称（包含参数）	功能说明
CopyFile（source,destination,overwrite）	将指定的一个或多个文件复制到新的文件夹中，destination 参数代表是否覆盖目标文件
CreateTextFile（filename,overwrite,unicode）	用指定的文件名 filename 在磁盘上创建一个新的文本文件，并返回与其对应的 TextStream 对象
DeleFile（filespec,force）	删除由 filespec 指定的一个或多个文件（可以包含通配符）
FileExists（filespec）	判断文件是否存在，返回值为逻辑值
GetBaseName（filespec）	获取文件名称，不包含文件的扩展名
GetExtensionName（filespec）	获取文件的扩展名，返回值为文本
GetFile（filespec）	获取文件对象，返回值为对象
GetFileName（pathspec）	获取文件的路径或名称，不检查文件是否存在
GetTempName()	返回一个随机产生的文件名
MoveFile（source,destination）	将一个或多个源文件移动到指定的目的文件夹，可以包含通配符
OpenTextFile（filename,iomode,create,format）	创建或者打开一个名叫作 filename 的文件

表 18-6 中 CopyFile、DeleFile、FileExists、GetBaseName、GetFile、MoveFile 等方法比较常用，读者可以在 VBA 自带的帮助中查询更详细的语法说明，以及查看其案例应用。

18.2　用 FSO 处理文件与文件夹

文件与文件夹的管理比较简单，涉及的技术比较单一。

下面通过 4 个案例演示 FSO 技术的综合应用。

18.2.1　让 D 盘中所有隐藏的文件夹显示出来

案例要求：D 盘中有部分文件夹处于隐藏状态，要求利用 VBA 将它们显示出来。

过程代码：

```
Sub 取消隐藏文件夹的隐藏属性()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  On Error Resume Next
  Dim FolderObj As Object
  With CreateObject("Scripting.FileSystemObject")
    For Each FolderObj In .GetFolder("D:\").SubFolders
      FolderObj.Attributes = Normal
    Next FolderObj
  End With
End Sub
```

D 盘中假设存在隐藏的文件夹，执行以上过程后将会全部显示出来。

思路分析：

本例要操作的对象是 D 盘的子文件夹，因此先使用 FSO 对象的 GetFolder 方法生成一个文件夹对象，然后通过 SubFolders 属性获得该文件夹的所有子文件夹对象，接着使用 For Each Next 循环语句遍历每一个文件夹。

文件夹对象的 Attributes 属性代表文件夹是隐藏的、只读的还是常规的文件夹，此属性可读亦可写。本例的需求是将隐藏的文件夹取消隐藏，因此将文件夹的属性设置为 Normal，如果需要将非隐藏文件夹隐藏起来，则应将文件夹对象的 Attributes 属性赋值为 Hidden。

语法补充：

（1）GetFolder 方法可以根据路径生成对应的 Folder 对象，似类于使用 Evaluate 方法将文本“a1”转换成 Range 对象的原理，它的具体语法如下：

```
object.GetFolder(folderspec)
```

其中 object 代表 FSO 对象，参数 folderspec 代表文本形式的文件夹路径。

（2）SubFolders 对象代表文件夹对象的所有子文件夹，它是一个对象集合。使用循环语句遍历这个集合时，变量 FolderObj 应使用 Object 型，不能使用 String 或者 Integer 型。

本例文件参见光盘：..\第十八章\ 18-3 让 D 盘中所有隐藏文件夹显示出来.xlsm

18.2.2 遍历子文件夹创建文件目录

案例要求：对用户随意指定的文件夹创建文件目录，包含子文件夹中的文件。

过程代码：

```
Dim arr(), i As Long
Sub 创建文件目录()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
Dim PathSht As String
With Application.FileDialog(msoFileDialogFolderPicker)
  If .Show Then PathSht = .SelectedItems(1) Else Exit Sub
End With
PathSht = PathSht & IIf(Right(PathSht, 1) = "\", "", "\")
i = 0
Call Contents(PathSht)
Range("A1").Resize(i, 1) = WorksheetFunction.Transpose(arr)
If MsgBox("是否创建链接?", vbYesNo, "链接") = vbYes Then
  Application.ScreenUpdating = False
  For i = 1 To i
    ActiveSheet.Hyperlinks.Add Anchor:=Cells(i, 1), Address:=arr(i), TextToDisplay:
```

```
=Dir(arr(i))
    Next
    Application.ScreenUpdating = True
  End If
End Sub
Sub Contents(Folder As String)
  On Error Resume Next
  Dim FolderObj As Object
  With CreateObject("Scripting.FileSystemObject")
    For Each FolderObj In .GetFolder(Folder).Files
      i = i + 1
      ReDim Preserve arr(1 to i)
      arr(i) = FolderObj.Path
    Next FolderObj
    For Each FolderObj In .GetFolder(Folder).SubFolders
      Call Contents(FolderObj.Path)
    Next FolderObj
  End With
End Sub
```

执行以上过程会弹出一个浏览文件夹的对话框，当选择一个文件夹并单击“确定”按钮后，程序会将该文件夹中的所有文件和子文件夹中的文件的名称存放在活动工作表的 A 列中。然后弹出一个“是否创建链接”的询问信息框，如果用户单击了“是”按钮，程序会对在 A 列的文件目录添加超链接。

思路分析：

创建文件目录比较简单，使用 FileDialog 对象让用户指定一个路径，然后使用 GetFolder 方法加 Files 属性即可获取该路径下的所有文件对象集合。接着使用循环语句遍历文件对象集合，从而提取所有文件的名称。

本例的难点在于获取指定路径下的所有文件名称时要兼顾其子文件夹中的文件，因此本例在“创建文件目录”过程之外编写了一个带参数的过程，该过程的参数代表文件夹路径，在过程中首先使用循环语句提取所有子文件的名称到数组 arr 中，然后再次使用循环语句遍历所有子文件夹，并且通过递归方式调用过程自身继续执行下去，从而将所有子目录中的文件名称一并提取到数组 Arr 中。

公共变量 i 必须在过程“创建文件目录”中强制初始化为 0。尽管第一次执行过程时该变量的默认值就是 0，但是如果不强制赋值为 0，在第二次执行过程“创建文件目录”时，由于公共变量会在过程结束后保留其值，因此第二次或者第三次执行过程时变量的默认值不再是 0，它会影响后面的执行结果。

语法补充：

（1）Files 属性代表文件夹中的文件对象集合，其返回值是对象，因此遍历 Files 时的循环语句中使用的变量必须声明为 Object，当然也可以采用变体型变量。

（2）FSO 中的 Path 属性代表文件的全名，包括路径，相当 Workbook.FullName 属性。

本例文件参见光盘：..\第十八章\18-4 遍历子文件夹创建文件目录.xlsm

18.2.3 删除 D 盘中大小为 0 的文件夹

案例要求：在 D 盘中有若干个空白的文件夹，要求利用 VBA 删除空白的文件夹。

过程代码：

```
Sub 删除D盘中大小为0的文件夹()'①代码存放位置：模块中②随书光盘中有每一句代码含义注释
```

```
  Dim Folder As Object
  With CreateObject("Scripting.FileSystemObject")
    On Error Resume Next
    For Each Folder In .getfolder("D:\").SubFolders
      If Folder.Size = 0 Then .DeleteFolder (Folder)
    Next Folder
  End With
End Sub
```

执行以上过程后，D 盘中体积为 0 的文件夹会全部被删除。

思路分析：

SubFolders 代表文件夹对象集合，不能使用索引号来访问其子集，必须使用 For Each Next 循环语句遍历这个对象集合，然后逐一判断其中每一个元素是否符合“体积为 0”这个条件，如果符合条件则使用 DeleteFolder 方法删除。

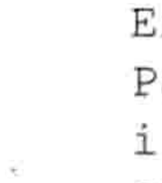
语法补充：

（1）Size 属性代表文件夹的体积，单位为字节，假设要换算成 GB，则应将它除以 1024^3。

（2）DeleteFolder 方法用于删除文件夹，其语法如下：

```
object.DeleteFolder folderspec[, force]
```

其中 folderspec 代表要删除的文件夹名称，force 参数代表是否删除只读的文件夹，赋值为 True 时表示要删除，赋值为 False 时表示不删除。



本例文件参见光盘：..\第十八章\ 18-5 删除 D 盘中大小为 0 的文件夹.xlsm

18.2.4 罗列最近 3 天修改过的所有文件的名称

案例要求：磁盘中有数千个文件，要求罗列出最近 3 天修改的文件，同时列出修改时间。

过程代码：

```
'①代码存放位置：模块中②随书光盘中有每一句代码含义注释
Dim arr(), i As Long
Sub 罗列最近3天修改过的所有文件的名称()
  Dim PathSht As String
  With Application.FileDialog(msoFileDialogFolderPicker)
    If .Show Then PathSht = .SelectedItems(1) Else Exit Sub
  End With
  PathSht = PathSht & IIf(Right(PathSht, 1) = "\", "", "\")
  i = 0
  Call Contents(PathSht)
  Range("A1").Resize(i, 2) = WorksheetFunction.Transpose(arr)
  Columns("a:b").EntireColumn.AutoFit
End Sub
Sub Contents(Folder As String)
  On Error Resume Next
  Dim FolderObj
  With CreateObject("Scripting.FileSystemObject")
    For Each FolderObj In .GetFolder(Folder).Files
      If Now - FolderObj.DateLastModified <= 3 Then
        i = i + 1
        ReDim Preserve arr(1 To 2, 1 To i)
        arr(1, i) = FolderObj.Path
```

```
        arr(2, i) = FolderObj.DateLastModified
      End If
    Next FolderObj
    For Each FolderObj In .GetFolder(Folder).SubFolders
      Call Contents(FolderObj.Path)
    Next FolderObj
  End With
End Sub
```

执行以上过程会弹出一个浏览文件夹的对话框，当选择一个文件夹并单击“确定”按钮后，程序会将该文件夹中所有的最近 3 天修改过的文件和修改时间罗列出来。如图 18.3 所示是作者以 F 盘为路径的执行结果，由于执行时间是 2014 年 3 月 4 日，因此会将 3 月 2 日、3 日和 4 日修改过的 3 个文件都罗列出来。

	A	B
1	F:\3月计划书.docx	2014/3/2 22:10
2	F:\生产表\3月报表.xlsx	2014/3/4 19:35
3	F:\生产表\品质月报表.xlsm	2014/3/3 11:12

图 18.3　最近 3 天修改过的文件列表

思路分析：

本例代码是基于 18.2.2 小节的代码修改而来的，与该例不同之处在于将文件名称写入数组之前使用了条件语句判断当前时间 Now 与文件的最后修改时间 DateLastModified 之间的差值是否小于等于 3，只有符合条件者才写入到数组中。

由于本例需要将文件名称和最后一次的修改时间一并写入到数组中，因此需要将数组变量重置为二维数组。

语法补充：

（1）DateLastModified 属性代表文件或者文件夹的最后一次修改时间。

（2）DateLastAccessed 属性代表文件或者文件夹的最后一次访问时间，DateCreated 属性代码文件的创建时间。

本例文件参见光盘:..\第十八章\18-6 罗列最近三天修改过的所有文件的名称.xlsm

◆　◆　◆

第19章 认识类和类模块

类，是一个隐藏的对象，也是一个神奇的对象。通过类可以实现诸多神奇的功能。

类的应用比较多，也比较复杂。在初级应用中主要偏重于两个方面，其一是借助类创建应用程序级别的事件，其二是用于在窗体中简化控件的事件过程代码，使原本需要重复编写的代码得以大大简化。

本章先介绍类的一些基础理论，然后通过5个案例展示类的简单应用。

19.1 类模块基础

类是一个很特殊的对象，类的代码同样也很特殊。类的应用比普通的Sub过程更复杂，更难理解，在介绍具体应用类之前，先来了解类的一些基础知识。

19.1.1 类的概念与用途

类是包含了方法及属性的高级代码模块，在这个模块中通过代码定义了一个类的对象。

类可以理解为一批具有类似属性的对象的统称，而其中单个对象则是类的一个实例。

通俗地讲，所有人都有一些共同的习性，所以每个人都属于“人”这个类，而每个人则是这个类中的一个个体。

在VBA中，类代表着一批具有相同属性、方法和事件的对象，为了简化应用，通过类来管理这些对象。

类有方法、属性和事件，但是它是抽象存在的，类是没有图形界面的，只能通过代码调用它，并用心去感知它的存在。

由于类代表着一批相同属性的对象，所以通常使用类来简化代码。例如有100个相同的对象，每个对象都需要一个相同的功能，如果没有类这个概念，需要针对100个对象编写100次代码。当使用类后，只需要对类编写一次代码，这个代码会自动应用到100个对象中去，这正是类的价值所在。

类是一个比较抽象、糊模的概念，不需要纠结于类的解释，往往越解释越糊涂。更好地理解类的方法是通过具体案例了解类的特性。

19.1.2 声明与调用类

1. 声明类

在VBE界面中单击“插入”→“类模块”命令，此时就产生了一个空白的类，即类模块就代表类，而在类模块中的代码则定义了这个类的一些属性和方法。

声明类时，必须使用WithEvents关键字，表示它是一个用于类模块的对象变量。

```
Public WithEvents Btn As CommandButton
```

以上语句就声明了一个CommandButton类，即按钮。当修改CommandButton类的属性时所有与此类相关的按钮都会同步修改属性。当对CommandButton类指定事件时，所有与此类相关的按钮都可以调用此事件。

```
Public WithEvents xlApp As Application
```

以上语句则声明了一个 Application 类，即应用程序。所有需要调用 Application 的对象都可以通过调用这个类来实现同等功能。

```
Public WithEvents Mysht As Worksheet
```

以上语句声明了一个 Worksheet 类，即工作表对象。

```
Public WithEvents MyLabel As MSForms.Label
```

以上语句声明了一个 MSForms.Label 类，即窗体中的标签对象。
当声明好类后，需要通过类的事件来调用类，否则类将无用武之地。

2. 类的事件

类有两个事件，包括 Initialize 事件和 Terminate 事件。其中 Initialize 事件在引用类时触发，或者说创建一个类的实例时触发 Initialize 事件；而 Terminate 事件则是删除对象时触发，例如将代表类的对象变量设置为 Nothing。

可以通俗地理解为创建对象时触发 Initialize 事件，而对象消失时则触发 Terminate 事件。

在实际使用中，通常在 Initialize 事件中对 WithEvents 语句所声明的变量赋值，而在 Terminate 事件中则释放此变量的值。例如：

```
Dim WithEvents myApp As Application
Private Sub Class_Initialize()'代码存放位置：类模块中
  Set myApp =Application
End Sub
Private Sub Class_Terminate()
  Set myApp = Nothing
End Sub
```

3. 调用类的基本步骤

使用类主要包括 3 个步骤：在模块中声明类、对变量赋值从而触发类的事件、对类所代表的对象设置事件。具体步骤如下。

step 1 单击菜单中的“插入”→“类模块”命令，在属性窗口中将类模块的默认名称“类 1”修改为“新类”，然后在类模块中输入以下代码：

```
Dim WithEvents myApp AsApplication
```

此代码表示声明一个应用程序类，即 Application。

step 2 在类模块中继续输入代码：

```
Private Sub Class_Initialize()'代码存放位置：类模块中
  Set myApp = Application
End Sub
```

此代码表示触发类的 Initialize 事件时为变量 myApp 指定对象，此后变量 myApp 即成为一个应用程序对象，可以利用它指定应用程序的一切事件，包括 WorkbookOpen、NewWorkbook、SheetChange 等事件。

接着需要编写触发类的 Initialize 事件的代码，从而使变量 myApp 处于可用状态。

单击菜单中的“插入”→“模块”命令，并在模块中录入以下代码：

```
Dim a As 新类          '声明一个类的实例
Sub Auto_open()        '开启工作簿时自动执行(代码存放位置：模块中)
  Set a = New 新类     '将变量 a 赋值给类，从而触发类的 Initialize 事件
```

```
End Sub
```

此过程使用 Set 语句对代表类的变量赋值，从而触发类的 Initialize 事件。要注意变量 a 的类型必须使用类的名称“新类”，即类模块的名称。而对变量赋值时则需要使用“New 新类”，New 关键字表示创建一个类的实例，此创建过程会触发 Initialize 事件。

step 3 根据需求编写对象 myApp 的事件。

由于此时变量 myApp 代表 Application，所以可以在类模块中实现任意应用程序级的事件。

在类模块中单击“对象”列表，列表中会出现 Class 和 myApp 两个对象，其中 myApp 代表当前声明的类的对象。当选择此对象后，再单击右边的“过程”列表，会出现 Application 相关的事件。图 19.1 和图 19.2 分别表示类模块中的“对象”列表和“过程”列表。

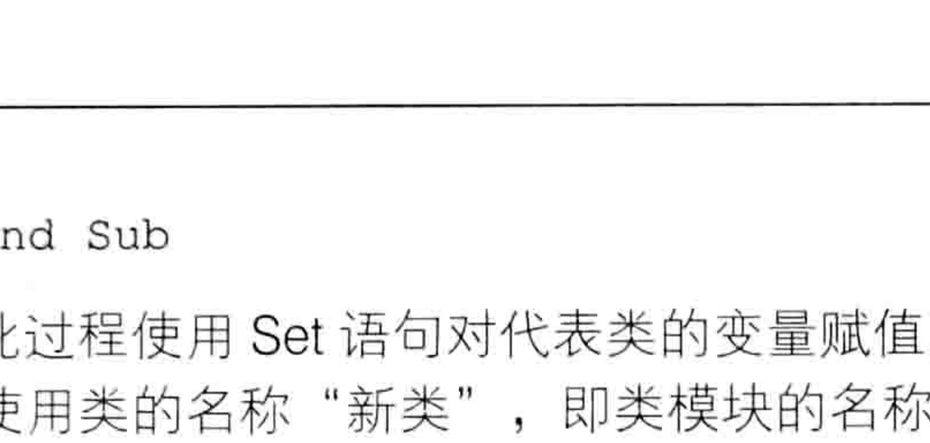


图 19.1 在对象列表中选择“myApp”

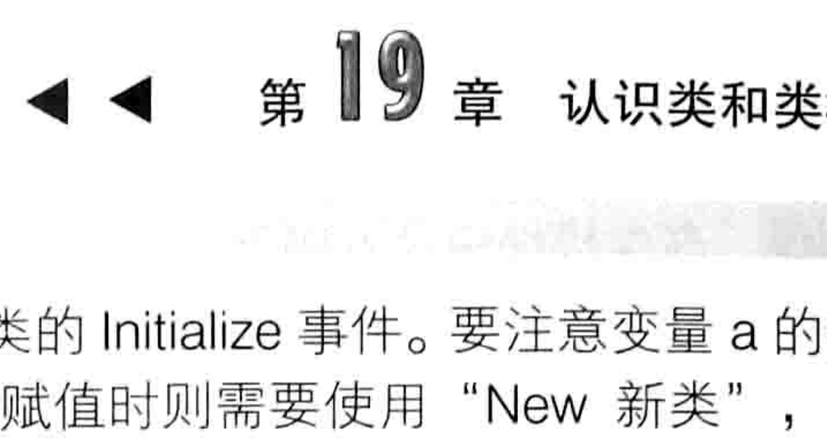


图 19.2 在过程列表中选择事件过程

假设需要创建类 myApp 的 SheetSelectionChange 事件，以测试代码是否正常工作，那么可以在类模块中录入以下代码：

```
Private Sub myApp_SheetSelectionChange(ByVal Sh As Object, ByVal Target As
Range)
  Application.StatusBar = Target.Address'在状态栏中显示 Target 区域的地址
End Sub
```

完成以上 3 个步骤后，保存并重启工作簿，然后选择任意工作表的任意区域，在状态栏中将出现选区的地址，说明类的代码已正常工作。

由于类比较难理解，操作上比普通的 Sub 过程更难，所以在实际应用中读者并不需要每次都重复以上三个步骤，可以将此工作簿作为类模块应用的模板保存起来，后续需要使用类时，直接打开此模板稍加修改即可。

本例文件参见光盘：..\第十九章\19-1 类模块应用模板.xlsm

19.2 类与应用程序级事件

Excel 允许用户使用应用程序级的事件，但是却不像工作表事件和工作簿事件那样提供直接的代码窗口。所幸可以通过声明 Application 类来创建一个应用程序级的变量，在类模块中通过此变量调用应用程序级的事件。本节通过两个案例展示应用程序级事件的应用思路。

19.2.1 在状态栏显示当前行的最大值与最小值地址

案例要求： 打开任意工作簿都在状态栏显示当前行的最大值和最小值。

知识要点： 应用程序级的 myApp_SheetSelectionChange 事件。

操作步骤：

step 1 在 VBE 界面中单击菜单中的“插入”→“类模块”命令，将类模块的名字修改为“ExcelApp”。

step 2 在类模块中录入以下代码：

```
Dim WithEvents myApp As Application '声明一个 application 型的变量（放置位置：类模块中）
Private Sub Class_Initialize()     '创建类的对象时触发此事件
  Set myApp = Application          '对变量赋值为 Excel 应用程序
End Sub
```

以上代码表示创建类的实例（关联到 myApp 类的一个对象）时触发事件，在事件中将代表 Excel 应用程序的 Application 对象赋值给变量 myApp。

```
'应用程序级的 SheetSelectionChange 事件，改变选区时触发此事件
'参数 Sh 代表工作表，Target 代表区域
Private Sub myApp_SheetSelectionChange(ByVal Sh As Object, ByVal Target As
Range)
  Dim MaxValue As Double, Minvalue As Double
  MaxValue = WorksheetFunction.Max(Target(1).EntireRow)
  Minvalue = WorksheetFunction.Min(Target(1).EntireRow)
  Application.StatusBar = "当前行最大值:" & MaxValue & "    最小值" & Minvalue
End Sub
```

以上代码是应用程序级的 SheetSelectionChange 事件过程，表示改变选区地址时在状态栏中显示当前行的最大值和最小值。

step 3 单击菜单中的“插入”→“模块”命令，并在模块中录入以下代码：

```
Dim a As ExcelApp          '声明一个类的实例
Sub Auto_open()            '开启工作簿时自动执行（放置位置：模块中）
  Set a = New ExcelApp     '将变量 a 赋值给类，从而触发类的 Initialize 事件
End Sub
```

以上代码的作用是打开工作簿时通过 Set 语句触发类的 Initialize 事件，从而使变量 myApp 处于可用状态。如果缺少以上代码，那么事件过程“myApp_SheetSelectionChange”不会正常执行。

step 4 保存并重启工作簿，任意选择单元格，状态栏中将出现当前行的最大值和最小值，效果如图 19.3 所示。

图 19.3 在状态栏显示当前行的最大值与最小值

代码分析：

（1）代码“Set a = New ExcelApp”用于触发类的 Initialize 事件，执行此代码后类模块中的代码才会产生作用。

（2）本例将包含“Set a = New ExcelApp”的代码放在名为“Auto_open”的过程中是为了让代码自动执行，开启工作簿后类模块的代码就会马上正常工作。如果在编写好代码后不想重启工作簿，可以手动运行过程“Auto_open”达成相同目的。

（3）如果要求在代码所在工作簿中的任意工作表的选区变化时将当前行的最大值与最小值显示在状态栏中，那么可以采用工作簿事件“Workbook_SheetSelectionChange”。本例采用了应用程序级事件，相对于工作簿事件，它的优势在于可以同时应用在所有打开的工作簿中，而非局限于

代码所在的工作簿。所以读者可以试着新建一个工作簿，然后在新工作簿的工作表中任意录入数据，然后再查看状态栏的显示内容是否有相应的变化。

本例文件参见光盘：..\第十九章\19-2 在状态栏显示当前行的最大值与最小值地址.xlsm

19.2.2 录入数据时自动将“M”后面的数字“2”显示为上标

案例要求：建筑行业经常需要输入平方米字符（㎡），每次都通过单击菜单中的“插入”→“符号”命令的方式录入的话，录入效率会相当低。现要求在任意工作簿中录入字符时，自动将“M”后面的“2”显示为上标。

知识要点：应用程序级的 myApp_SheetChange 事件。

操作步骤：

step 1 在 VBE 界面中单击菜单中的“插入”→“类模块”命令，将类模块的名字修改为“ExcelApp”。

step 2 在类模块中录入以下代码：

```
Dim WithEvents myApp As Application
Private Sub Class_Initialize()'①代码存放位置：类模块中②随书光盘中有每一句代码的含义注释
  Set myApp = Application
End Sub
Private Sub myApp_SheetChange(ByVal Sh As Object, ByVal Target As Range)
  On Error Resume Next
  Dim i As Integer
  If Target.Count = 1 Then
    If Len(Target) > 1 Then
      For i = 1 To Len(Target.Value) - 1
        If UCase(Mid(Target, i, 2)) = "M2" Then Target.Characters(Start:=i + 1,
Length:=1).Font.Superscript = True
      Next
    End If
  End If
End Sub
```

以上代码是应用程序级的 SheetChange 事件过程，表示修改单元格的值时将单元格中“M”后面出现的“2”显示为上标。

step 3 单击菜单中的“插入”→“模块”命令，并在模块中录入以下代码：

```
Dim a As ExcelApp          '声明一个类的实例
Sub Auto_open()            '开启工作簿时自动执行
  Set a = New ExcelApp     '将变量 a 赋值给类，从而触发类的 Initialize 事件
End Sub
```

step 4 保存并重启工作簿，在任意单元格中输入“1282M2+500M2”，两个“M”后面的“2”将显示为上标，效果如图 19.4 所示。

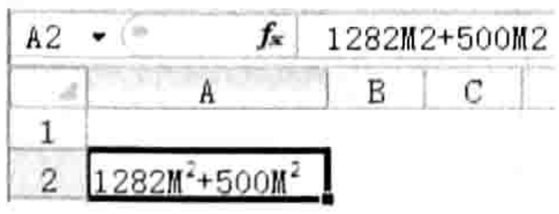


图 19.4 将“M”后面的“2”显示为上标

只要当前工作簿处于打开状态时，那么在其他任意工作簿中录入字符时，都会将“M”后面的“2”显示为上标。

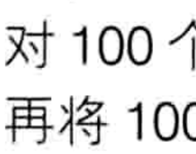
代码分析：

（1）事件过程的参数 Target 有可能是单个单元格，也可能是区域，当它代表区域时使用代码“Len(Target)”计算字符长度会出错，所以代码中需要通过条件语句加以限制，当“Target.Count = 1”时才执行后面的代码。

（2）字符串“M2”的长度是 2，因此单元格的字符长度小于 2 时没有必要判断单元格中是否存在“M2”，所以本例的过程中使用了代码“Len(Target) >1”加以限制，表示字符长度大于 1 时才执行后面的命令。

（3）代码“Mid(Target, i, 2)”表示从录入的字符串中第 i 位提取长度为 2 的字符串。在其外面使用 UCase 函数表示将提取的字符串转换成大写形式，此举的目的是提升代码的通用性，否则用户录入小写的“m”时，“m”后面的“2”不会显示为上标。

（4）Font.Superscript 赋值为 True 表示将字符显示为上标，但是赋值为 False 时并不代表显示为下标，而是将 Font.Subscript 属性赋值为 True 时才显示为下标。



本例文件参见光盘：..\第十九章\19-3 录入数据时自动将 M 后面的 2 显示为上标.xlsm

19.3 类模块与窗体控件

类除了用于调用应用程序级别的事件外，更多的是应用于窗体中，通过类可以简化窗体中各种控件的事件过程代码。

19.3.1 何时需要使用类

窗体中的控件属于对象，而对象本身是分类的，例如按钮类、标签类、选项按钮类……

当窗体中某类对象比较多而且需要对它们编写相同的事件过程代码时，借用类可以大大简化代码。例如窗体中有 100 个选项按钮，每个选项按钮都有一个 Click 事件，如果不使用类，那么需要对 100 个选项按钮编写 100 个 Click 事件过程。如果声明一个类（例如 MSForms.Option Button 类），再将 100 个选项按钮都关联到此类，那么只需要对类的对象编写一次事件过程代码，100 个选项按钮都可以自动调用此过程，从而将工作量简化到原来的 1%。

不过，这同时也说明，类只在同一类控件大量出现的时候才有用武之地，否则直接在窗体中编写代码即可。

19.3.2 为按钮批量指定 MouseMove 事件

案例要求：假设窗体中有 6 个按钮，使用类实现鼠标指针下的按钮变化，其他按钮总是显示为默认的浅灰色，从而达到醒目的作用。

知识要点：MouseMove 事件、BackColor 属性。

操作步骤：

step 1 单击菜单中的“插入”→“用户窗体”命令。

step 2 在窗体中添加 6 个按钮，即 CommandButton 控件，并且分别使用 CommandButton1、CommandButton2 、 CommandButton3 、 CommandButton4 、 CommandButton5 和 CommandButton6 作为控件的名称。6 个控件的排列方式如图 19.5 所示。

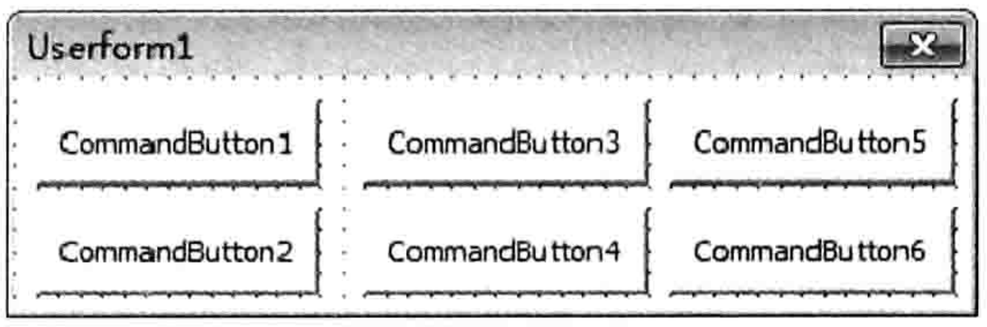


图 19.5 包括 6 个按钮的窗体

step 3 单击“插入”→“类模块”命令，并将类模块的默认名称“类 1”修改为“ButtonClass”。修改名称仅为方便识别，并非必须使用此名称，改用“ABC”或者“我的类”也可以。

step 4 在类模块中录入以下过程代码：

```
'①代码存放位置：类模块中②随书光盘中有每一句代码的含义注释
Public WithEvents 按钮 As MSForms.CommandButton
Private Sub 按钮_MouseMove(ByVal Button As Integer, ByVal Shift As Integer, ByVal
x As Single, ByVal Y As Single)
  For j = 1 To 6
    If Userform1.Controls("CommandButton" & j).Name <>按钮.Name Then
      Userform1.Controls("CommandButton" & j).BackColor = &H8000000F
    Else
      Userform1.Controls("CommandButton" & j).BackColor = &HFF8080
    End If
  Next
End Sub
```

变量“按钮”被声明为 MSForms.CommandButton，表示这是一个窗体中的按钮类，将窗体中的任意按钮关联到此类，按钮控件就可以直接应用“按钮”的事件过程。

事件过程“按钮_MouseMove”表示鼠标移过“按钮”时触发的事件，“按钮”在此处是指对象，但不是单个具体的对象，而是一类对象，所有属于此类的按钮都可以调用此事件。

不过此事件中的对象“按钮”只代表当前鼠标指针下的那一个按钮，而不是同时代表窗体中的 6 个按钮。

当鼠标指向窗体中的第一个按钮控件时，“按钮”就代表第一个按钮控件；当鼠标移到第二个按钮控件时，“按钮”就代表第二个按钮控件……

step 5 双击窗体进入窗体的代码窗口，在窗口中录入以下代码：

```
'①代码存放位置：窗体的代码窗口中②随书光盘中有每一句代码的含义注释
Dim Buttons(1 To 6) As New ButtonClass
Private Sub UserForm_Initialize()
  For j = 1 To 6
    Set Buttons(j).按钮 = Controls("CommandButton" & j)
  Next
End Sub
```

以上代码表示声明一个关联到 ButtonClass 类的数组 Buttons，它包含 6 个成员。当加载窗体时，使用 Set 语句分别将窗体中的 6 个按钮关联到 Buttons 中的 6 个成员，从而使窗体中的这 6 个按钮都可以调用类模块中的“按钮_MouseMove”事件过程。

step 6 单击菜单中的“插入”→“模块”命令，并在模块中录入以下代码：

```
Sub 显示窗体() '代码存放位置：模块中
  Userform1.Show 0
End Sub
```

step 7 执行过程“显示窗体”，鼠标在窗体中随意移动，鼠标指针下的按钮将显示为浅蓝色背景，而其他按钮则显示为默认的浅灰色背景，如图 19.6 和图 19.7 所示。

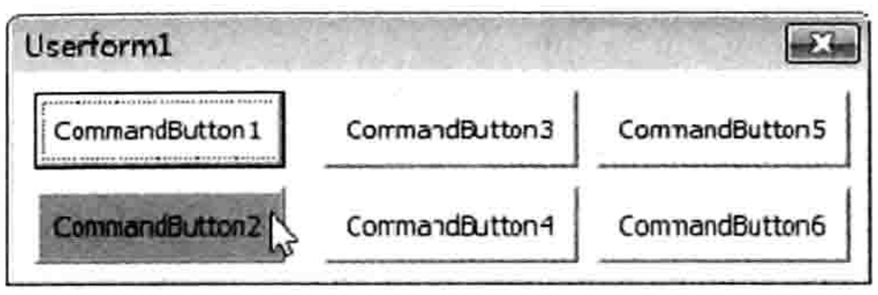


图 19.6 鼠标移过第 2 个按钮时

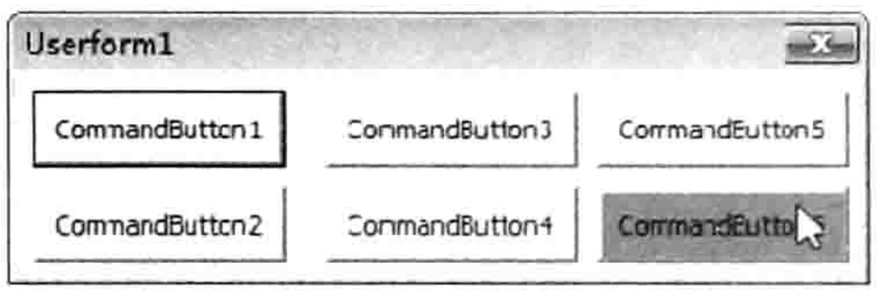


图 19.7 鼠标移过第 6 个按钮时

代码分析：

（1）按钮控件分两种，其一是应用在工作表中的 ActiveX 控件，即功能区中的“开发工具”→“插入”→“命令按钮 ActiveX 控件”，它的类名称为 CommandButton；其二是窗体中的按钮控件，它的类名称为 MSForms.CommandButton。

（2）“Controls("CommandButton" & j)”表示引用窗体中名为“"CommandButton" & j”的控件对象。使用此方式引用控件的前提是 6 个按钮必须命名为 CommandButton1、CommandButton2、CommandButton3 、 CommandButton4 、 CommandButton5 和 CommandButton6，否则无法使用循环语句引用这 6 个控件。

（3）设置按钮的背景颜色时，颜色代码“&H8000000F”和“&HFF8080”代表什么颜色不需要花时间去记忆，更不需要记代码本身，在属性窗口中手动修改按钮的背景色后，属性窗口的“BackColor”属性将自动显示为此颜色所对应的编码，手工复制出来并粘贴到代码中即可使用。如图 19.8 所示即为通过手工指定颜色来获取颜色编码的过程演示。

（4）类的应用过程偏于复杂，读者可以直接将本例文件保存为模板，后续有类似的需求时直接使用此文件稍加修改即可。例如需要对窗体中的 100 个标签编写 MouseMove 事件过程，只需要将本例代码中的 CommandButton 改为 Label 即可。

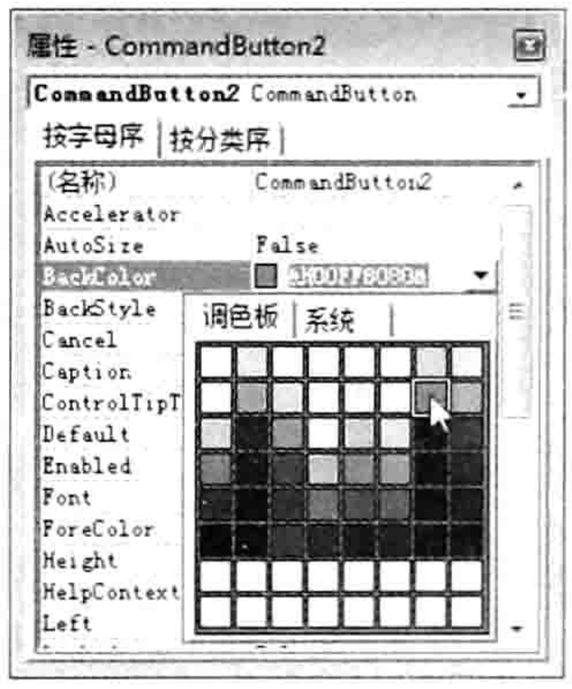


图 19.8 手工设置颜色后自动产生颜色编码

本例文件参见光盘：..\第十九章\19-4 为按钮批量指定鼠标的 MOVE 事件.xlsm

19.3.3 开发颜色面板

案例要求：Excel 的功能区中有选择字体颜色及背景色的操作面板，但它无法通过命令调用到窗体中。如果在窗体中需要让用户选择颜色，可以自己开发一个颜色选择面板。

知识要点：窗体的 Initialize 事件、类的 Click 事件。

操作步骤：

step 1 插入一个窗体，将其名称修改为“ColorS”，Caption 属性修改为“单击选择一种颜色”。

step 2 在窗体中插入一个标签，删除其 Caption 属性值，并将 BorderStyle 属性设置为 1。

step 3 将标签复制为 40 个，并且按 5 行均匀排列，每行 8 个标签，如图 19.9 所示。

step 4 双击窗体，进行代码窗口。并输入以下代码：

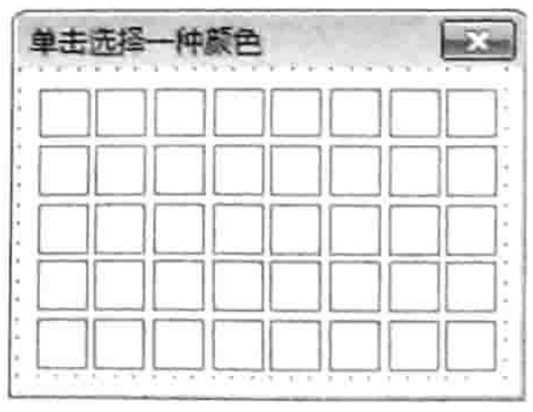


图 19.9 在窗体中添加 40 个标签控件

```
'①代码存放位置：窗体的代码窗口中②随书光盘中有每一句代码的含义注释
Dim Buttons(1 To 40) As New ClassColors
Private Sub UserForm_Initialize()
  Dim i As Byte
  ColIndex = Array(1, 53, 52, 51, 49, 11, 55, 56, 9, 46, 12, 10, 14, 5, 47, 16,
3, 45, 43, 50, 42, 41, 13, 48, 7, 44, 6, 4, 8, 33, 54, 15, 38, 40, 36, 35, 34,
37, 39, 2)
  ColName = Array("黑色", "褐色", "橄榄色", "深绿", "深青", "深蓝", "靛蓝", "灰度
80%", "深红", "橙色", "深黄", "绿色", "青色", "蓝色", "蓝灰", "灰色-50%", "红色", "
浅橙色", "酸橙", "海绿色", "水绿色", "浅蓝", "紫罗兰", "灰色-40%", "粉红", "金色", "
黄色", "鲜绿", "青绿", "天蓝", "梅红", "灰色-25%", "玫瑰红", "茶色", "浅黄", "浅绿
", "浅青绿", "淡蓝", "淡紫", "白色")
  For i = 1 To 40
    Set Buttons(i).颜色标签 = Me.Controls("Label" & i)
    Me.Controls("Label" & i).BackColor = ActiveWorkbook.ColorS(ColIndex(i - 1))
    Me.Controls("Label" & i).ControlTipText = ColName(i - 1)
  Next
End Sub
```

以上代码表示运行窗体时将数组中的颜色代码逐个赋予 40 个标签，同时设置标签的中文提示信息。

step 5 插入一个类模块，将其名称改为“ClassColors”，名称必须与前面的 Buttons 变量的类型一致。

step 6 在类模块中录入以下代码：

```
Public WithEvents 颜色标签 As MSForms.Label
Private Sub 颜色标签_Click()'代码存放位置：类模块中
  MsgBox "颜色值为：" &颜色标签.BackColor
End Sub
```

以上代码表示单击窗体中的标签按钮时弹出按钮所代表的颜色值。在实际工作中读者可以将代码修改为设置某对象的颜色，假设要设置窗体的背景色，那么可用以下代码：

```
ColorS.BackColor = 颜色标签.BackColor
```

如果对单元格设置背景色，那么可用以下代码：

```
ActiveCell.Interior.Color = 颜色标签.BackColor
```

step 7 运行窗体，窗体中的 40 个标签将会显示为 40 种预设的颜色，当鼠标移过任意标签时会有中文的颜色提示，效果如图 19.10 所示。

step 8 单击任意标签，VBA 会弹出该标签所代表的颜色值。

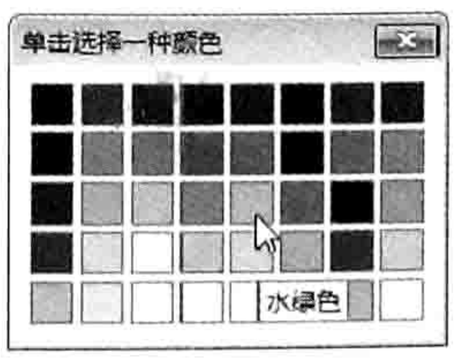


图 19.10 鼠标移过标签时提示颜色信息

代码分析：

（1）为了便于通过索引号引用标签，在窗体中添加控件时必须要以从左到右、从上到下的顺序排列 40 个标签，使标签的名称刚好是 Label1、Label2、Label3、Label4……否则通过循环语句中的代码“Controls("Label" & i)”引用标签时可能错位。

（2）工作簿的调色板中有 56 个颜色，ActiveWorkbook.ColorS 代表了这 56 种颜色，通过索引号可以调用其中每一种颜色。本例仅取其中 40 种颜色显示在自定义的调色板中，读者也可以在窗体中添加 56 个标签，然后在窗体中将这 56 种颜色显示完整。

（3）标签控件的 BackColor 属性代表标签的背景颜色值，它和单元格的 Interior.Color 属性值采用相同的编码方式，和 Interior.ColorIndex 采用不同的编码方式。

（4）ControlTipText 属性代表鼠标移过控件时的提示信息。

本例如果不使用类模块的知识，那么需要在窗体中添加 40 个标签的单击事件，而使用类模块则用一个单击事件即可完成同等功能，这是类的优势。

本例文件参见光盘：..\第十九章\ 19-5 颜色面板.xlsm

第 20 章　VBA 与注册表

注册表是 Windows 的核心，计算机中大部分重要的设置都记录在注册表中。使用 VBA 编写 Sub 过程时也可以将与过程相关的某些数据保存在注册表中。

VBA 中用于读写注册表中的代码是 DeleteSetting、SaveSetting、GetSetting 和 GetAllSettings，其中常用的是 SaveSetting 和 GetSetting，本章会逐一讲解它们的语法与案例应用。不过 SaveSetting 和 GetSetting 有它自身的缺点，因此除 SaveSetting 和 GetSetting 以外，本章还会介绍更强大的注册表读写工具——脚本语言。

20.1　VBA 对注册表的控制方式

VBA 对注册表具有读、写权限，所以工作中常用代码将某些设置信息保存在注册表中供其他过程调用，或者在下一次执行当前过程时调用，从而让过程对某些设置具有记忆功能，提升程序的易用性。

20.1.1　什么是注册表

注册表的英文名称是 Registry，它的编辑器是 Regedit，它是一个记录系统信息的数据库。程序员通常会使用注册表来保存软件的安装信息、用户名、版本号、日期和序列号等。

注册表是一个很重要的数据库，如果对注册表设置不规范可能会导致软件无法使用或者 Windows 无法启动，如果对注册表的各键值不熟悉，不宜随意编辑注册表。

手工打开注册表的方法是按<Win+R>组合键打开“运行”对话框，然后输入注册表名称 Regedit 并按<Enter>键，其后 Windows 会打开名为“注册表编辑器”的程序界面。

在注册表中有一个专门供 VBA 读、写数据的项“VB and VBA Program Settings”，VBA 自带的注册表函数只能读、写该项下的数据。图 20.1 展示了 VBA 可以读写的注册表项。

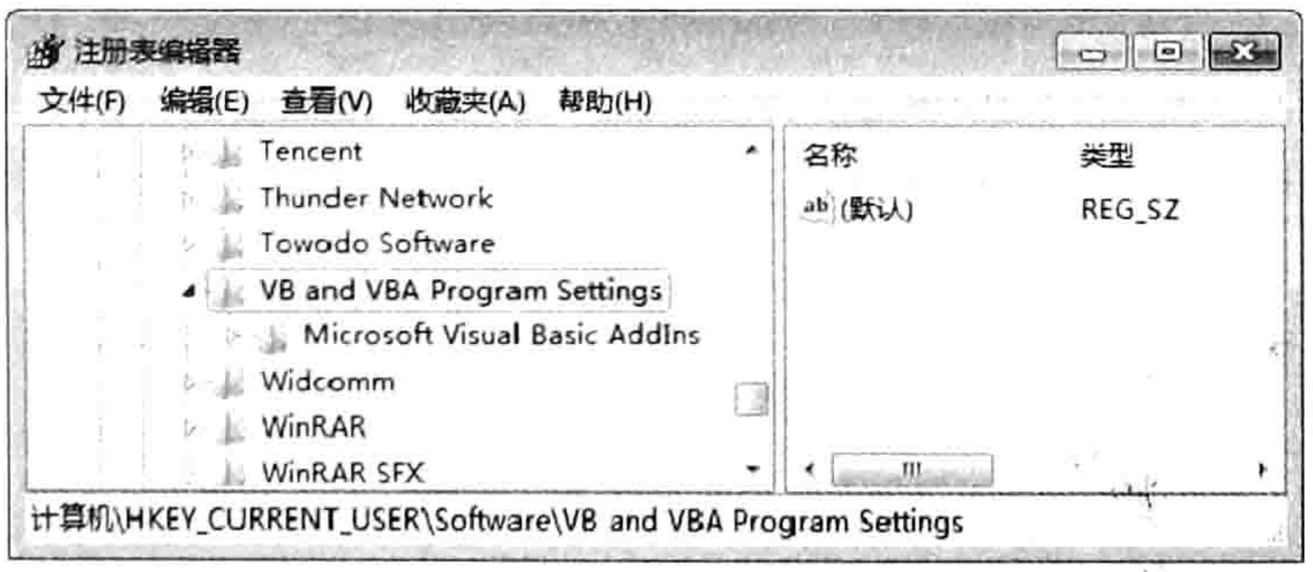


图 20.1　注册表中专供 VBA 读取的项

20.1.2　VBA 操作注册表的方法

VBA 提供了 4 个与注册表相关的关键字，如表 20-1 所示。

表 20-1　可读写注册表的语句

语句/函数	功能描述
DeleteSetting	删除程序设置
GetSetting、GetAllSettings	读入程序设置
SaveSetting	保存程序设置

其中 GetSetting 和 GetAllSettings 两个函数用于读取注册表数据，而 SaveSetting 和 DeleteSetting 语句分别用于写入和删除注册表的值，它们都只能操作注册表的以下位置：

```
[HKEY_CURRENT_USER\Software\VB and VBA Program Settings\]
```

1. 保存数据到注册表

SaveSetting 语句可以将数据保存到注册表中，它的语法如下：

```
SaveSetting appname, section, key, setting
```

它的 4 个参数都是必要参数，缺一不可。各参数的含义如表 20-2 所示。

表 20-2　SaveSetting语句的参数详解

参数名称	含义描述
appname	应用程序或工程的名称。可以理解为主目录
section	区域名称，在该区域保存注册表项设置。可以理解为子目录
key	要保存的注册表项设置的名称，可以理解为要保存在注册中的键名
setting	对应于 key 的值。可以理解为键值

用代码和图示可以更好地展示参数与注册表的位置对应关系。
以下代码可以向注册表中写入当前时间，保存在“Excel”→“时间”→“值”中：

```
SaveSetting appname:="Excel", section:="时间", Key:="值", setting:=Now
```

以上 4 个参数写入的键值会显示在注册表中如图 20.2 所示的位置：

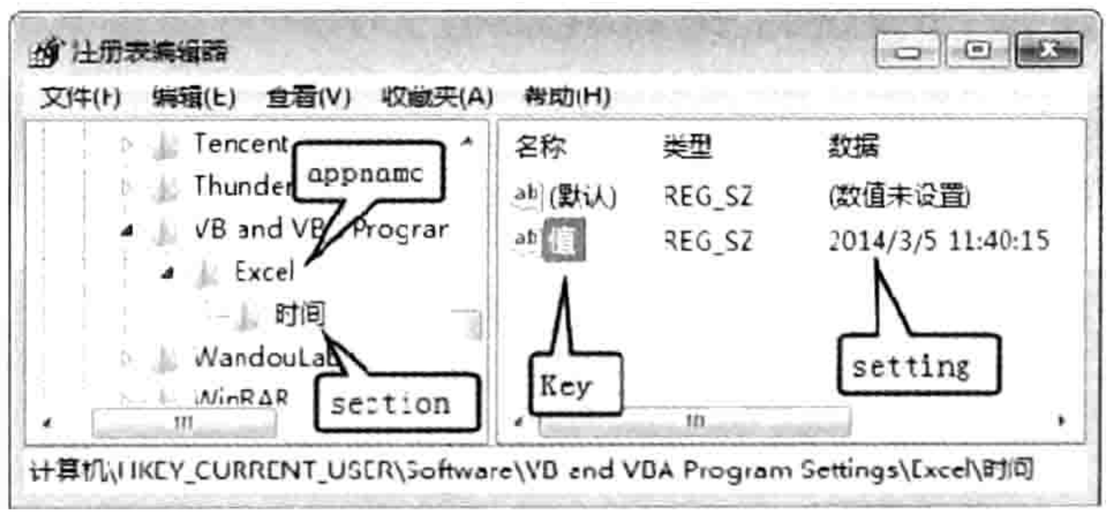


图 20.2　SaveSetting 的参数在注册表中的位置

一个注册表的项下可以创建多个键，因此可以只修改 SaveSetting 的第 3 参数和第 4 参数从而继续向注册表的同一个项中创建多个键与键值。

2. 读取注册表设置

GetSetting 和 GetAllSettings 两个函数用于读取注册表中的设置信息。其中前者读取某个注册表项的值，它有 4 个参数，需要指定 SaveSetting 语句所设置的每个项目名称；后者用于读取某个程序项目的所有注册表项设置及其相应值，前者的应用更广泛。

GetSetting 函数的具体语法如下：

```
GetSetting(appname, section, key[, default])
```

它的 4 个参数和 SaveSetting 的 4 个参数刚好对应。其中第 4 参数是可选参数，当注册表中不存在要读取的键值时，GetSetting 函数将返回第 4 参数的值，如果注册表中存在要读取的键值则返回该键的实际值。第 4 参数的默认值是空文本。

以下代码可以从注册表中读取到前面用的 SaveSetting 语法写入到注册表中的时间值。

```
MsgBox GetSetting("excel", "时间", "值")
```

3. 删除注册表信息

DeleteSetting 语句用于删除某个注册表设置，它的语法如下：

```
DeleteSetting appname, section[, key]
```

如果参数所指定的键值不存在，那么删除语句会产生错误，所以使用该语句前要防错。

DeleteSetting 的第 3 参数是可选参数，如果忽略该参数则可删除注册表中指定名称的项，如果指定该参数则只删除项下的一个键。

20.2 注册表的应用

善用注册表可以让程序更具人性化——自动应用上一次的设置。例如 Excel 的“查找”对话框就具有记忆功能，第二次打开该对话框时会自动调用上一次的设置。

本节通过 3 个案例展示注册表在程序中的应用思路。

20.2.1 记录当前工作簿最后一次打开时间

案例要求：通过工作簿的 BuiltinDocumentProperties 属性可以记录每个工作簿最后一次保存的时间，但无法获取最后一次打开工作簿的时间。本例通过工作簿事件配合注册表技术记录本机中最后一次打开工作簿的时间。

知识要点：GetSetting 函数、SaveSetting 语句、Workbook_Open 事件。

操作步骤：

step 1 打开需要记录时间的工作簿，按<Alt+F11>组合键进入 VBE 窗口。

step 2 双击 Thisworkbook 从而进入工作簿事件代码窗口，然后录入以下代码：

```
Private Sub Workbook_Open()'①代码存放位置：ThisWorkbook②随书光盘中有每一句代码含义注释
  Application.StatusBar = "上次打开时间：" & GetSetting("Excel", "上次打开时间",
ActiveWorkbook.Name, "")
  SaveSetting "Excel", "上次打开时间", ActiveWorkbook.Name, Now
End Sub
```

step 3 保存工作簿，然后反复打开，可以发现在状态栏中记录了本工作簿的上一次打开时间。

图 20.3 所示的是工作簿的上一次打开时间在状态栏的显示效果，图 20.4 则用于说明“上一次打开时间”在注册表中的存放位置。

代码分析：

（1）本例的重点在于通过工作簿的 Open 事件调用 SaveSetting 语句，用 Now 函数产生当前时间作为 SaveSetting 语句的 setting 参数。Open 事件负责在正确的时间产生记录，SaveSetting

语句则负责将时间保存在正确的位置。

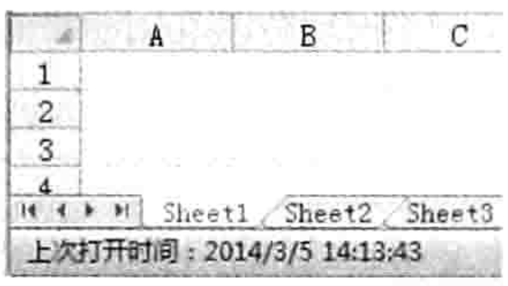


图 20.3　记录上一次打开时间

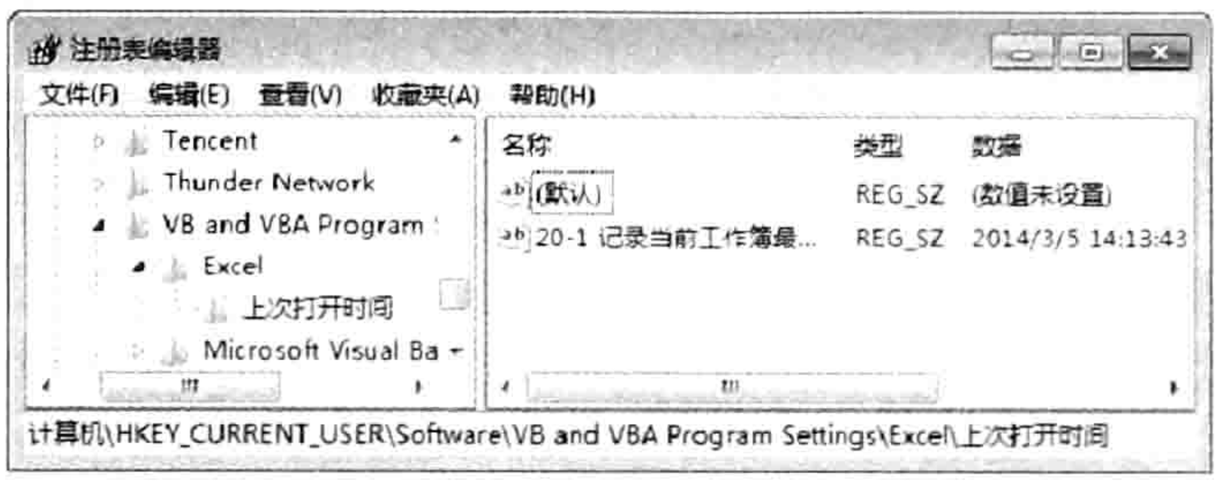


图 20.4　上一次打开时间在注册表中的位置

（2）读取注册表的值使用 GetSetting 函数，由于第一次打开工作簿时注册表中并没有时间记录，因此会在状态栏显示空文，第二次打开工作簿时才会产生如图 20.3 所示的结果。

（3）注册表是存在本机中的数据库，只能在本机上才可以读取它的数据，因此本例的代码是记录工作簿在本机中的打开时间，当工作簿被发送给其他用户后，其他用户不能读取本机中最后一次打开工作簿的时间。

本例文件参见光盘：..\第二十章\ 20-1 记录当前工作簿最后一次打开时间.xlsm

20.2.2　创建文件目录时自动记忆上一次的路径

案例要求：对任意路径下的所有文件创建目录，要求在选择文件夹路径时默认显示上一次选择的路径。

知识要点：GetSetting 函数、SaveSetting 语句、FileDialog 对象、FSO 对象。

实现步骤：

step 1　在 VBE 界面中单击菜单中的“插入”→“模块”命令。

step 2　在模块中录入以下过程代码：

```
Dim arr(), i As Long
Sub 创建文件目录()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Dim PathSht As String
  PathSht = GetSetting("Excel", "文件目录", "路径")
  With Application.FileDialog(msoFileDialogFolderPicker)
    If Len(PathSht) > 0 Then .InitialFileName = PathSht
    If .Show Then PathSht = .SelectedItems(1) Else Exit Sub
  End With
  PathSht = PathSht & IIf(Right(PathSht, 1) = "\", "", "\")
  SaveSetting "Excel", "文件目录", "路径", PathSht
  i = 0
  Call Contents(PathSht)
  Range("A1").Resize(i, 1) = WorksheetFunction.Transpose(arr)
  If MsgBox("是否创建链接?", vbYesNo, "链接") = vbYes Then
    Application.ScreenUpdating = False
    For i = 1 To i
      ActiveSheet.Hyperlinks.Add Anchor:=Cells(i, 1), Address:=arr(i), TextTo
Display:=Dir(arr(i))
    Next
    Application.ScreenUpdating = True
  End If
```

```
End Sub
Sub Contents(Folder As String)
  On Error Resume Next
  Dim FolderObj
  With CreateObject("Scripting.FileSystemObject")
    For Each FolderObj In .GetFolder(Folder).Files
      i = i + 1
      ReDim Preserve arr(1 To i)
      arr(i) = FolderObj.Path
    Next FolderObj
    For Each FolderObj In .GetFolder(Folder).SubFolders
      Call Contents(FolderObj.Path)
    Next FolderObj
  End With
End Sub
```

step 3 执行以上过程，当弹出浏览文件夹的对话框时选择“D:\生产表”，然后单击“确定”按钮，程序会在活动工作表的A列中创建文件目录。当程序弹出“是否创建链接?”的询问时单击按钮“否”按钮结束过程。

step 4 关闭工作簿，然后重新打开刚才的工作簿并执行过程“创建文件目录”，此时会发现对话框的默认文件路径是“D:\生产表”，这是因为代码读取了上一次保存在注册表中的路径。

代码分析：

（1）本例代码是基于本书18.2.2案例的代码修改而来的，在原来的代码中加入了读取注册表和写入注册表两句代码。由于FileDialog对象的InitialFileName属性代表默认路径，因此使用GetSetting函数从注册表中读取上一次的路径，然后赋值给InitialFileName属性即可。

（2）GetSetting函数第一次从注册表中取值时必定只能取到空文本，而不是上一次所选择的路径，因此在过程中需要使用Len函数作判断，不能直接将GetSetting函数的返回值赋值给InitialFileName属性，否则首次执行代码时会出错。

本例文件参见光盘：..\第二十章\20-2 创建文件目录时自动记忆上次的路径.xlsm

20.2.3 让是否显示零值的设置适用于所有工作表

案例要求：Excel对于是否显示零值的设置仅仅对活动工作表有效，切换工作表后会发现Excel并没有调用相同的设置去控制新表。现要求设计一个快捷键来设置零值的显示方式，而且一旦设置好后一切工作簿中的所有工作表都能应用该设置。

知识要点：GetSetting函数、SaveSetting语句、类模块、Window.DisplayZeros属性、应用程序级别的SheetActivate事件和WindowActivate事件。

实现步骤：

step 1 在VBE界面中单击菜单中的“插入”→“模块”命令。

step 2 在模块中录入以下代码：

```
Dim aa As 新类
Sub Auto_open()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Set aa = New 新类
  Application.OnKey "^T", "零值"
End Sub
```

```
Sub 零值()
  Dim msg As VbMsgBoxResult
  msg = MsgBox("显示零值吗？", vbYesNo, "设置零值显示方式")
  ActiveWindow.DisplayZeros = (msg = vbYes)
  SaveSetting "Excel", "零值", "显示状态", msg = vbYes
End Sub
```

其中过程“Auto_open”用于打开工作簿时自动将名为“新类”的类赋值给变量 aa，然后为过程“零值”指定快捷键<Ctrl+Shift+T>。过程“零值”则用于弹出一个信息框让用户选择零值的显示方式，然后将用户的选择应用到活动工作表，同时保存到注册表中。

step 3 单击菜单中的“插入”→“类模块”命令，然后在类模块中录入以下代码：

```
'①代码存放位置：类模块中②随书光盘中有每一句代码的含义注释
Dim WithEvents myApp As Application
Private Sub Class_Initialize()
  Set myApp = Application
End Sub
Private Sub myApp_SheetActivate(ByVal Sh As Object)
  ActiveWindow.DisplayZeros = GetSetting("Excel", "零值", "显示状态",
ActiveWindow.DisplayZeros)
End Sub
Private Sub myApp_WindowActivate(ByVal Wb As Workbook, ByVal Wn As Window)
  ActiveWindow.DisplayZeros = GetSetting("Excel", "零值", "显示状态",
ActiveWindow.DisplayZeros)
End Sub
```

step 4 保存工作簿，然后重启工作簿。

step 5 按<Ctrl+Shift+T>组合键，打开如图 20.5 所示的对话框，单击对话框中的“否”按钮。此时会发现所有值为 0 的单元格会被自动隐藏起来。

step 6 新建工作簿或者切换到其他工作簿中，然后在单元格中录入 0，单元格中的零值会同样被隐藏起来，效果如图 20.6 所示。

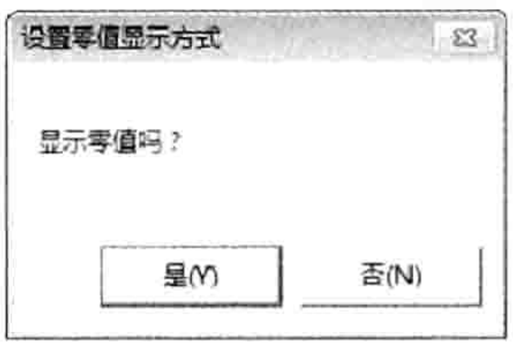


图 20.5 选择零值显示方式

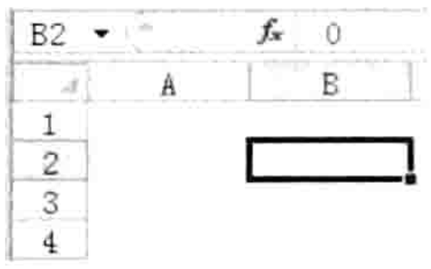


图 20.6 已隐藏零值的单元格

step 7 再次按下<Ctrl+Shift+T>组合键，在对话框中单击按钮“是”按钮，此时所有工作表中的零值都会显示出来。

代码分析：

（1）本例中主要涉及三方面技术，一是控制零值的显示与否，对 DisplayZeros 属性赋值即可，它控制当前窗口的零值显示方式；二是使用注册表读、写技术将设置应用于所有活动工作簿的其他工作表；三是使用类模块技术将设置应用于所工作簿中的所有工作表。如果不使用类模块，则只有活动工作簿中有生效。

（2）打开设置零值显示方式的代码所在工作簿才可能通过快捷键调用过程“零值”，因此本例的案例文件必须处于打开状态才能让代码正常工作。工作中不可能总是先打开这个工作簿再做设置，正确的做法是将本工作簿保存为加载宏格式的工作簿，保存在 Office 2010 的自启动路径下，

后续使用时该工作簿会自动在后台打开，不影响日常操作又能方便地调用其中的代码。

如果用户的计算机的操作系统是 64 位的、Office 2010 是 32 位的，那么路径为：

```
C:\Program Files (x86)\Microsoft Office\Office14\XLSTART
```

如果用户是 32 位的操作系统、32 位的 Office 或者 64 位的操作系统、64 位的 Office 则路径为：

```
C:\Program Files\Microsoft Office\Office14\XLSTART
```

在本书的第 23 章会具体讲述加载宏的设计过程，以及更多的案例。

本例文件参见光盘：..\第二十章\ 20-3 让是否显示零值的设置适用于所有工作表.xlsm

20.3 注册表函数的缺点与改善方法

VBA 内置的 GetSetting 函数与 SaveSetting 语句分别用于读、写注册表，它们的语法简单、功能强大，不过也有明显的缺点。本节主要介绍它们的缺点和改善办法。

20.3.1 VBA 操作注册表的优缺点

VBA 提供的 DeleteSetting、GetSetting、GetAllSettings 和 SaveSetting 可以读、写注册表，通过在注册表中保存当前设置信息，然后在下一次执行过程时读取这些信息，从而让程序更人性化。不过它有明显的缺点，主要表现在以下两个方面。

1. 位置固定

默认的注册表操作方式仅仅限于操作以下位置：

```
HKEY_CURRENT_USER\Software\VB and VBA Program Settings
```

如果想在注册表的其他位置写入信息则无法实现。

2. 类型单一

SaveSetting 语句无法在注册表中创建二进制（REG_BINARY）的值。

20.3.2 借用脚本自由控制注册表

用脚本语言读写注册表远比 VBA 自带的语句更灵活，所幸 VBA 可以随意调用脚本语言，所以 VBA 也可以借助脚本代码随心所欲地控制注册表。

利用脚本读、写注册表可用以下三个方法完成：

CreateObject("WScript.Shell").RegWrite——写入数据；

CreateObject("WScript.Shell").RegRead——读取数据；

CreateObject("WScript.Shell").RegDelete——删除数据。

其中 RegWrite 的语法如下：

```
CreateObject("WScript.Shell").RegWrite strName, any Value, [strType]
```

RegWrite 有 3 个参数，第 1 参数表示注册表的主键位置或者称之为路径，例如“HKEY_CURRENT_USER\Software\VB and VBA Program Settings\Excel”，其中最后一个“Excel”是键名，而不是项名，相当于 SaveSetting 语句的第 3 参数。

RegWrite 方法的第 2 参数是键值，相当于 SaveSetting 语句的第 4 参数。

RegWrite 方法的第 3 参数代表键值的类型，支持 REG_SZ、REG_EXPAND_SZ、REG_DWORD 和 REG_BINARY 4 种类型。

使用 RegWrite 方法向注册表中写入数据时，如果第 1 参数指定的键名不存在，那么脚本语言会自动在注册表中创建这个键。

以下代码可以在注册表中的“HKEY_CLASSES_ROOT\任意目录”位置处创建名为“你喜欢 VBA 吗”的键，其键值是“当然”，值的类型是 REG_SZ，如图 20.7 所示。

```
Sub 向注册表写入数据()'代码存放位置：模块中
  CreateObject("WScript.Shell").RegWrite "HKEY_CLASSES_ROOT\任意目录\你喜欢 VBA
吗", "当然", "REG_SZ"
End Sub
```

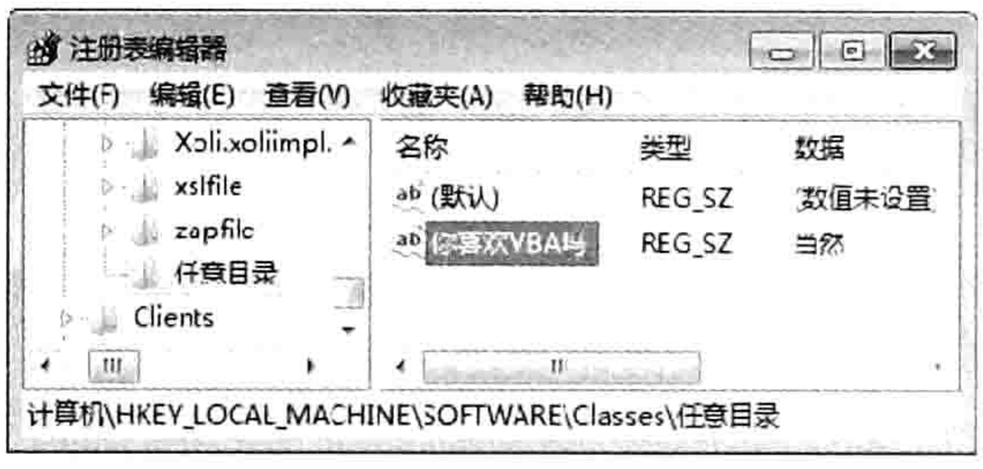


图 20.7　脚本代码向注册表中写入的值

20.3.3　禁止使用 U 盘

案例要求：U 盘可以传播病毒，而且计算机中的资料也可以被他人利用 U 盘盗走，基于此目的可以禁用 U 盘。由于注册表可以控制 U 盘是否被允许使用，因此要求利用 VBA 调用脚本语言修改注册表，从而实现禁止与允许使用 U 盘。

知识要点：脚本语言、注册表、Shell 函数。

实现步骤：

step 1　单击菜单中的“插入”→“模块”命令，然后在模块中录入以下代码：

```
Sub 控制 U 盘是否可用()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Dim msg As VbMsgBoxResult, RegPath As String
  Shell "taskkill  /f  /im explorer.exe"
  msg = MsgBox("允许使用 U 盘吗？", vbYesNo, "设置 U 盘状态")
  RegPath  =  "HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\USBSTOR\
Start"
  CreateObject("WScript.Shell").RegWrite RegPath, 4 + (msg = vbYes), "REG_DWORD"
  Shell "explorer.exe", vbNormalFocus
End Sub
```

step 2　执行过程“控制 U 盘是否可用”，此时 Windows 的资源管理器会关闭，然后弹出如图 20.8 所示的信息框。

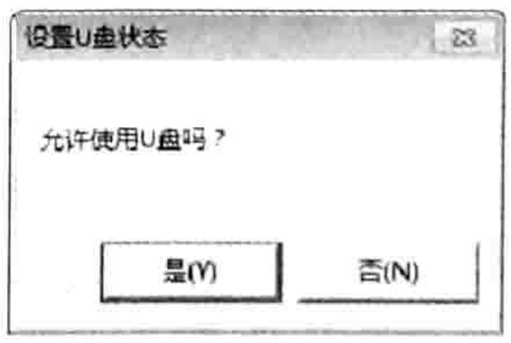


图 20.8　选择 U 盘的使用状态停息框

step 3 在对话框中单击“否”按钮，此时资源管理器会重新被打开。

step 4 在计算机的 USB 接口中插入 U 盘，可以发现 U 盘已经无法被读取。只有再次执行过程“控制 U 盘是否可用”，在弹出的对话框中单击“是”按钮才可以让 U 盘正常读取。

代码分析：

（1）注册表中以下键值为 3 时表示允许使用 U 盘，值为 4 时表示禁用 U 盘：

“HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\USBSTOR\Start”

因此本例使用脚本语言向以上路径写入 3 或者 4 足以控制 U 盘是否可用。

（2）本例中 Msgbox 函数的返回值是 VbYes 或者 VbNo，那么代码“msg = vbYes”的计算结果则是 True 或者 False。逻辑值在参与加法算法时将 True 当作-1 处理，将 False 当作 0 处理，因此当用户在对话框中选择“是”时代码“4 + (msg = vbYes)”的结果是 3，当用户选择“否”时代码“4 + (msg = vbYes)”的结果是 4。

（3）修改注册表后必须重启资源管理器才能让注册表的修改生效。本例中用 DOS 中的 taskkill 命令关闭资源管理器，待注册表写入完成后再重启资源管理器，从而使禁用或者启用 U 盘的代码在过程结束后可以立即生效。

（4）本例中的 Shell 函数用于调用 taskkill 命令，而代码中的“/f　/im explorer.exe”属于 taskkill 命令的参数，因此不能将代码理解为 Shell 函数调用了“explorer.exe”或者调用了“/f”。

taskkill 命令的功能是用于关闭指定名称的软件，本例中的“explorer.exe”代表资源管理器软件，而参数“-f”代表强制关闭。

本例文件参见光盘：..\第二十章\ 20-4 控制 U 盘是否可用.xlsm

注册表的用武之地相当广泛，由于本书篇幅有限，本章不再一一列举。读者可以将本章前面章节的部分 Sub 过程拿来练习，通过注册表技术让过程更人性化，就像本章的 20.2.2 的案例一样。

◆　◆　◆

第 21 章　Ribbon 功能区设计

微软公司从 Excel 2007 版开始使用功能区替代传统的工具栏和菜单。

从功能区第一次出现到现在已经有 7 年了，实践证明，功能区远比传统菜单更实用，其既美观又操作方便，而且可以同时在相同的空间中放置更多的按钮，节约查找命令按钮的时间。对于 VBA 开发者而言，定制属于自己的专用选项卡可提升调用过程的便捷性，同时也使自己的程序可以更好地融入 Excel 中。

本章详细阐述功能区选项卡中各部件的开发思路，同时教会读者制作功能区模板，从而提升开发效率。

21.1　功能区开发基础

开发功能区选项卡之前先要做一些准备工作，包括了解功能区的特性与结构、定制功能区的方法和安装代码编辑器。

21.1.1　Ribbon 的特点

功能区的英文名称是 Ribbon，它的外形就像一条飘浮在工作表顶端的带子。

功能区将 Excel 的常用功能按用户需求分布在 9 个选项卡中，分别为“文件”、“开始”、“插入”、“页面布局”、“公式”、“数据”、“审阅”、“视图”和“开发工具”。其中“开始”选项卡中包含最常用的功能命令，打开 Excel 后默认显示“开始”选项卡界面；“开发工具”选项卡属于高级用户专用，所以默认处于隐藏状态，需要在“Excel 选项”对话框中手工调整其显示状态。

功能区中的任何一个命令按钮都支持快捷键操作，例如<Alt+H+M+C>组合键表示调用“开始”选项卡中的“合并后居中”命令，<Alt+R+P+S>组合键表示调用“审阅”选项卡中的“保护工作表”命令……当利用代码创建新的选项卡和按钮后，也同样支持快捷键操作。

功能区虽然是 Excel 的功能之一，但是不能用 VBA 开发功能区的选项卡和命令按钮。

21.1.2　功能区的组件图示

功能区包含选项卡、组、命令按钮、切换按钮、标签、复选框、文字框、弹出式菜单、拆分按钮、下拉列表控件、分隔线和对话框启动器等组件。不过这些组件不会同时显示在一个界面中，图 21.1 中包含了功能区中的多数控件，读者可以从此图中了解各组件的外观。

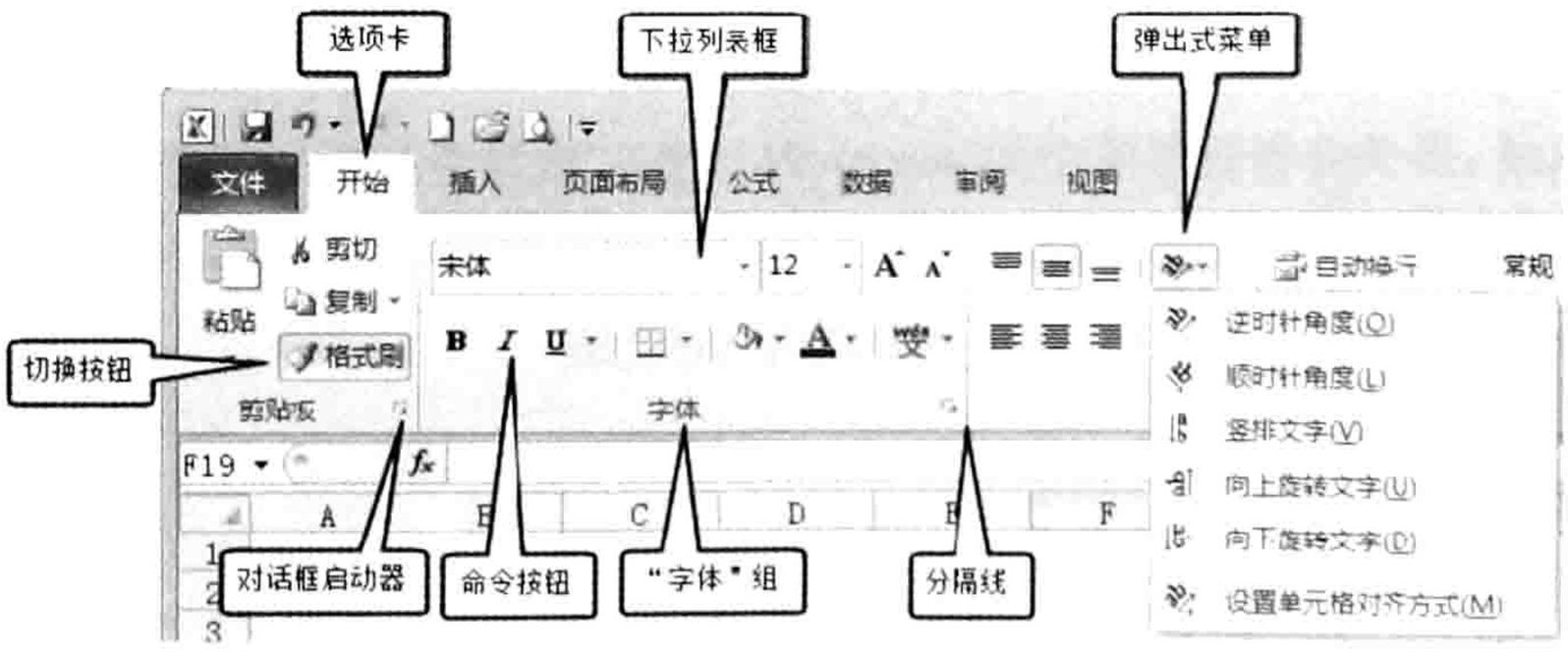


图 21.1　常见的功能区组件图示

21.1.3 手工定制功能区

Excel从Excel 2007版本开始使用功能区，但从Excel 2010版本中才支持手工定制功能区。

使用<Alt+T+O>组合键打开“Excel 选项”对话框，单击左方的“自定义功能区”选项即可看到手工定制功能区选项卡的界面。在此界面中可以新建选项卡、新建组，以及新建命令按钮。

不过手工定制的功能区远远不能满足需求，因为手工定制所产生的功能区组件并没有集成到文件中，所以当把Excel文件发送出去后，其他计算机打开文件时无法看到定制的组件，仍然只能使用<Alt+F8>组合键调用过程。

所以，尽管Excel允许手工定制功能区，但在实际工作中仍采用xml代码定制。

21.1.4 认识Ribbon代码编辑器

功能区代码是xml语言，无法通过VBA代码创建功能区选项卡和选项卡中的各种组件。

本章主要介绍通过外置软件来定制功能区，软件全名叫“Custom UI Editor for Microsoft Office 2010”，即Excel 2010专用的功能区界面编辑器，简称为CustomUIEditor。在以下地址可以下载此软件：

http://t.cn/8FkzlaT

当安装好软件后，在“开始”菜单中将显示“Custom UI Editor for Microsoft Office 2010”，单击即可打开软件。如图21.2所示是CustomUIEditor软件的操作界面。

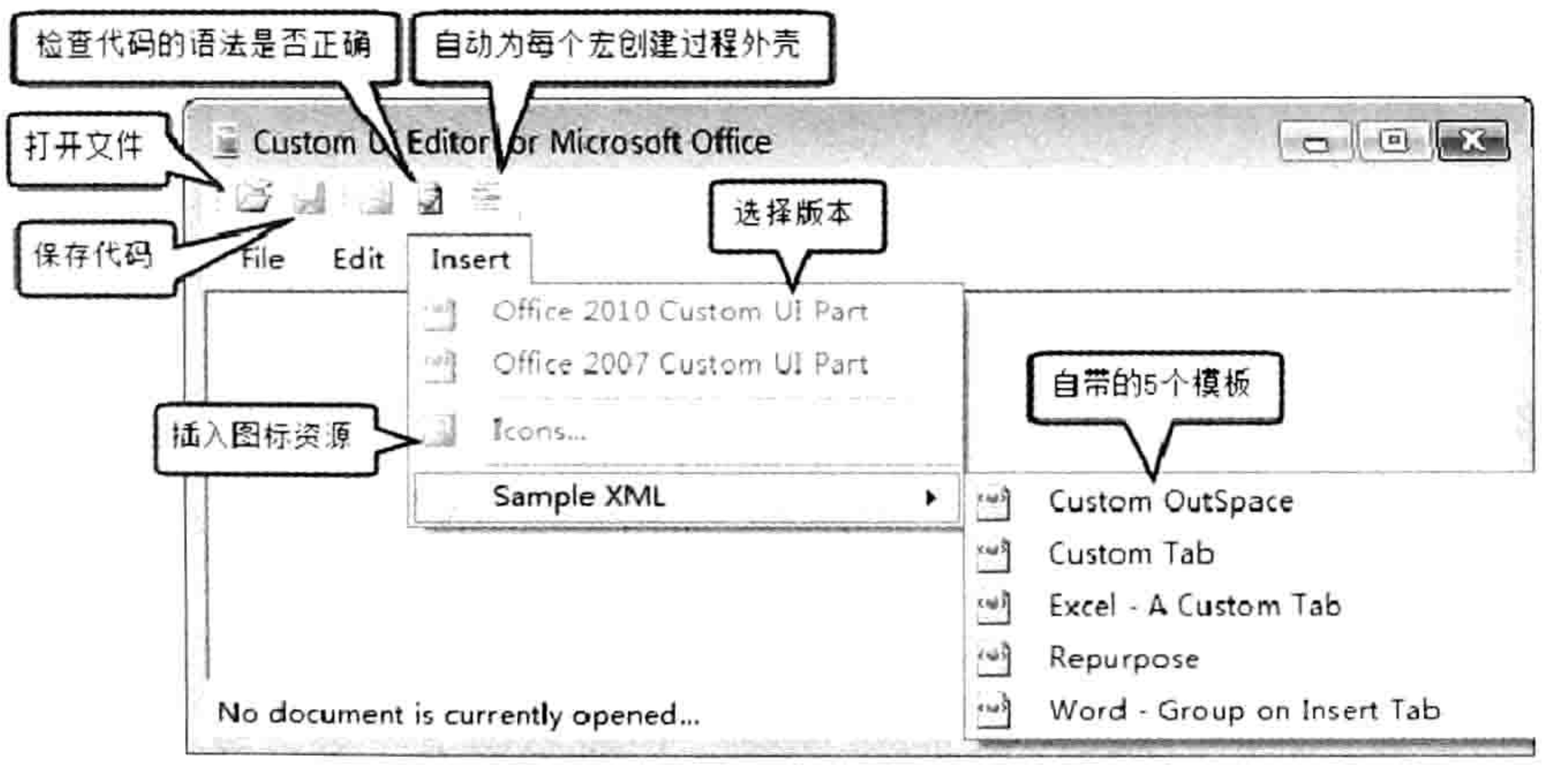


图21.2 CustomUIEditor软件的操作界面

软件是英文界面的，好在只有5个按钮而已，不认识英文也足以正常操作。

第一个按扭（图标：）用于打开文件。必须打开一个xlsx或者xlsm格式的Excel文件，然后再写入定制功能区的代码，最后单击第二个按钮保存代码，才可以在Excel文件中创建功能区相关的组件。

第三个按钮（图标：）用于插入图片，当需要对命令按钮分配自定义图标时使用。不过建议调用Excel的内部图标资源，一是为了减小文件体积，二是为了提升开启文件的速度。

第四个按钮（图标：）用于检查用户输入的代码是否存在语法问题，如果语法有错误将提示错误原因，以及在第几行的第几个位置。引号不配对或者名称中含有非法字符等都属于语法错误。

第五个按钮（图标：）用于产生代码中每个回调过程的程序外壳。由于回调过程都有若干个参数，而这些参数名称很难记忆，所以软件提供此功能对开发者而言帮助比较大。

“Insert”菜单中的前两个菜单分别代表生成Excel 2010格式的xml文件和Excel 2007格式的xml文件。如果单击第一个菜单后输入定制功能区的代码，那么此文件只能在Excel 2010中才会正常显示自定义的功能区组件，如果单击第二个菜单后输入定制功能区的代码，那么此文件用Excel

2007 和 Excel 2010 打开都会正常显示自定义的功能区组件。所以如果考虑兼容性，编写代码应采用 Excel 2007 格式。

当打开一个 xlsx 或者一个 xlsm 格式的文件后，单击第二个子菜单“Office 2007 Custom UI Part”，将会在该文件中插入一个存放功能区代码的文件，名为“customUI.xml”，如图 21.3 所示；如果单击第一个子菜单“Office 2010 Custom UI Part”，将插入一个名为“customUI14.xml”的文件，如图 21.4 所示，此文件中的代码所创建的功能区不能在 Excel 2007 中正常显示。

图 21.3　Excel 2007 格式的代码文件

图 21.4　Excel 2010 格式的代码文件

Excel 2007 格式的功能区代码和 Excel 2010 格式的功能区代码的第一句并不相同，前者代码如下：

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui">
```

后者代码如下：

```
<customUI xmlns="http://schemas.microsoft.com/office/2009/07/customui">
```

CustomUIEditor 软件自带的 5 个模板中的代码是 Excel 2010 格式。

21.1.5　获取内置按钮图标

在开发传统菜单和工具栏按钮时，对 FaceId 赋值为内置图标的序号即可为自定义菜单或者工具栏按钮指定图标，例如 3 代表保存文件的图标，4 代表打印文件的图标……

定制功能区的按钮和菜单时必须使用内部图标的英文名称。在进入第二节开始定制功能区之前，有必要先认识一下 Excel 的内置图标，在后面的教学过程中会大量调用内置图标。

Excel 提供了一个比较笨拙的办法获取所有内置图标的名称——在“Excel 选项”对话框的“自定义功能区”右侧的窗口中，鼠标指向任意内置命令，屏幕中将出现该命令的名称和图标名称，图标名称显示在括号中，如图 21.5 所示。

为了方便读者调用图标名称，笔者制作了一个 Excel 内置图标查看器，如图 21.6 所示。打开此工作簿后将自动生成一个新的选项卡“图标浏览”，该选项卡中每页显示 20 个内置图标，鼠标单击图标时会在活动单元格中产生该图标的名称。单击左侧的“前一页”和“下一页”按钮可以查看更多的图标。

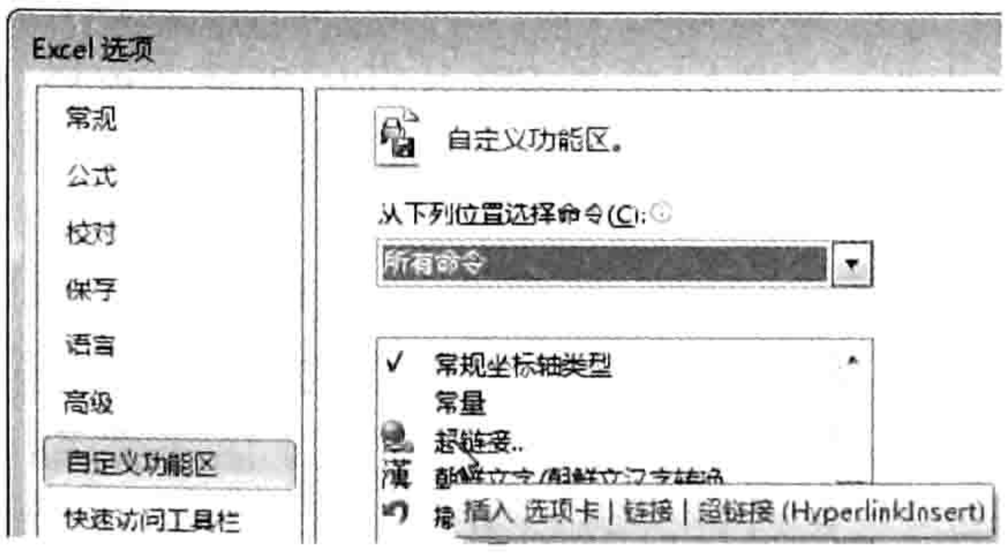


图 21.5　鼠标指向命令时显示命令名称和图标名称

图 21.6　内置图标浏览器

本例文件参见光盘：..\第二十一章\21-1 内置图标浏览器.xlsm

21.2 Ribbon定制之语法分析

功能区中的控件种类比较多，修改不同类型的控件有不同的语法。

本节对功能区中的各类控件逐一分析语法，并提供效果图示。学习本章前请安装CustomUIEditor软件。

21.2.1 功能区代码的结构

功能区代码的结构相当严谨，每句代码的顺序都有严格的规定，而且所有代码都必须配对，严格区分大小写。

功能区代码的结构比较复杂，通过一段具体的代码来理解则相对容易一些。

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui">
  <ribbon>
    <tabs>
      <tab id= label=>
        <group id= label= >
          <button id= label= screentip= supertip=onAction= image= />
          <menu id= label= screentip= supertip= size= image= >
            <button id= label= screentip= supertip=onAction= image= />
          </menu>
          <dialogBoxLauncher>
            <button id= label= screentip= supertip=onAction= image= />
          </dialogBoxLauncher>
        </group>
      </tab>
    </tabs>
  </ribbon>
</customUI>
```

以上代码用于在功能区中创建新的选项卡，在新选项卡中创建弹出式菜单，在弹出式菜单中创建一个按钮。可以按以下方式理解这段代码的结构：

第一句和最后一句是配对的，它是功能区代码的根或者称之为壳，类似于VBA中Sub与End Sub的关系。首尾两句都包含“customUI”，表示这是一个自定义功能区的容器。首句代码使用了“<>”符号，末句代码使用配对的“</>”符号表示结束。开头缺少“<>”或者结尾缺少“</>”符号都会产生错误。其他任何表示容器的语句均遵循此规则。不过命令按钮、标签、分隔条等底层的控件不属于容器，它们不需要遵循此原则。

代码中的“2006/01”是通用于Excel 2007和Excel 2010的，如果需要编写Excel 2010专用的功能区代码，应改用“2009/07”。

第二句与倒数第二句是配对的，“<ribbon>”代表后面的代码用于定制功能区，“</ribbon>”代表功能区定制过程结束。如果将“ribbon”修改为“backstage”则表示后面的代码用于定制backstage视图对象，而非Ribbon对象。如果将“ribbon”修改为“officeMenu”则表示后面的代码用于定制Excel 2007专用的Office按钮（Excel 2010中改成backstage视图了）

第三句与倒数第三句是配对的，“<tabs>”表示它后面的代码作用于选项卡，“</tabs>”则表示定制选项卡的代码结束。如果将此处的“tabs”修改为“qat”则表示定制快速启动工具栏，而非选项卡。选项卡tabs与快速启动工具栏qat是同级别的对象。

第四句与倒数第四句是配对的，“<tab>”表示此代码用于添加新选项卡或者修改内置选项卡，后面的id参数代表选项卡的id，而label参数则表示选项卡显示在屏幕上的名称，可以随意自

定义。在此模板中，等号后面故意留出空格是为了方便读者理解，表示此处可以根据实际需求赋值。模板中的 id 是指创建一个新的选项卡并为其指定一个 id，如果要引用内置的选项卡，那么需要将模板中的 id 修改为 idMso，例如“idMso="TabHome"”表示引用内部的“开始”选项卡。

第五句与倒数第五句是配对的，“<group>”表示此代码用于创建一个组，后面的 label 用于指定组的名称。当出现“</group>”时表示定制组的代码结束。

第六句代码“<button/>”用于创建一个命令按钮，由于按钮是底层的元素，所以只需要一行代码，行首为“<”，行尾为“/>”。此处也可以添加其他的控件，包括命令按钮、切换按钮、标签、复选框、文字框、弹出式菜单、拆分按钮、下拉列表控件、分隔线等。

第七句与倒数第九句是配对的，“<menu>”表示添加一个弹出式菜单，label、screentip、supertip、size、image 分别代表此弹出式菜单的菜单名称、提示信息、详细提示、大小和图标。当出现“</menu>”时表示定制弹出式菜单的代码结束。

第八句又是创建一个命令按钮，所以只需要一行代码。由于此按钮放置在“<menu>”与“</menu>”之间，说明它是弹出式菜单中的一个子菜单，而前面的“<button/>”放在“<group>”之后，表示它位于组中而不是弹出式菜单中。

第九句与倒数第六句是配对的，“<dialogBoxLauncher>”代表创建一个对话框启动器。对话框启动器是组 group 的子元素，与弹出式菜单 menu 是同级别的对象。

第十句又是创建一个命令按钮，此按钮处于对话框启动器 dialogBoxLauncher 中。一个对话框启动器中只能放置一个命令按钮，但是一个组 group 和一个弹出式菜单 menu 中可以放置多个命令按钮。

可以使用如图 21.7 所示的架构图来展示以上功能区部件的结构，从而有助于读者理解部件与部件间的关系与顺序。

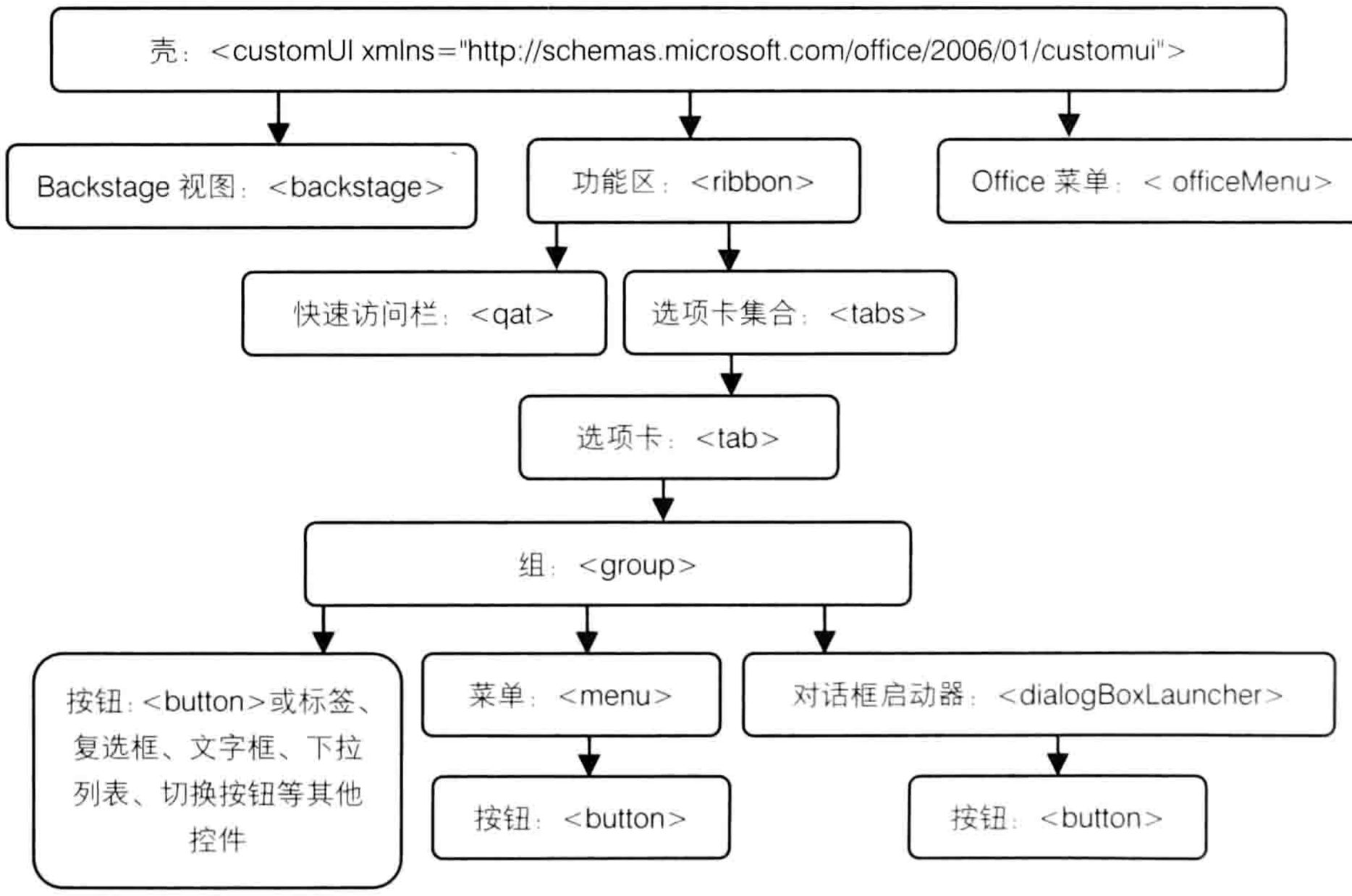


图 21.7　功能区部件的结构

21.2.2 显示与隐藏功能区：ribbon

功能区的代码是 ribbon，控制功能区显示与隐藏的语法如下：

```
<ribbon startFromScratch="AA">
```

```
</ribbon>
```

其中“"AA"”属于占位符，为方便叙述而存在。此处若赋值为“true”则表示隐藏功能区，赋值为“false”或者直接忽略 startFromScratch 参数则表示显示功能区。

要注意定制功能区的代码是严格区分大小写的，“true”和“false”每一个字母皆为小写。

实现隐藏功能区的具体操作如下（后续实现其他功能仅讲解定制功能区的代码，不再详述操作步骤，可以参照此处的步骤操作，替换代码即可）：

step 1　新建一个 Excel 文件，并将它保存为“隐藏功能区.xlsm”。

step 2　打开 CustomUIEditor 软件，从 CustomUIEditor 软件中打开前面保存的 Excel 文件。

step 3　选择菜单中的“Insert”→“Office 2007 Custom UI Part”命令从而插入一个 customUI.xml 文件。

step 4　在右边的代码窗口中录入以下代码：

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui">
  <ribbon startFromScratch="true">
  </ribbon>
</customUI>
```

step 5　单击“保存”按钮，然后关闭 CustomUIEditor 软件。

step 6　双击打开刚才保存的 Excel 文件，此时的 Excel 功能区将处于隐藏状态。

图 21.8 是 CustomUIEditor 软件操作界面，图 21.9 是最终的隐藏效果。

图 21.8　CustomUIEditor 软件操作界面

图 21.9　隐藏功能区的效果

创建功能区代码的语言不是 VBA 语言，而是 xml 语句。xlsm 和 xlsx 格式的工作簿中都可以存放 xml 代码，从而对工作簿定制功能区。但是 xlsx 格式的工作簿不能保存 VBA 代码，而定制功能区的代码总是与 VBA 代码搭配应用才能体现其价值，所以工作中应将文件保存为 xlsm 格式。

CD-ROM　本例文件参见光盘：..\第二十一章\21-2 隐藏功能区.xlsm

21.2.3　隐藏选项卡：tab

语法：

```
<tab idMso="AA" visible="BB">
</tab>
```

其中 idMso 表示调用内置选项卡的 id 号，如果改用 id 则是创建一个新的选项卡。
语法表中的 AA 代表选项卡的名称，表 21-1 中罗列了 Excel 2010 的选项卡 id。

表 21-1　内置选项卡名称

idMso	选项卡名称	idMso	选项卡名称
TabHome	开始	TabReview	审阅
TabInsert	插入	TabView	视图
TabPageLayoutExcel	页面布局	TabDeveloper	开发工具
TabFormulas	公式	TabAddIns	加载项
TabData	数据	TabPrintPreview	打印预览

其中 BB 代表可见性，当赋值为“true”时表示隐藏内置选项卡。

案例：隐藏“开始”选项卡。

代码：

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui">
  <ribbon startFromScratch="false">
    <tabs>
      <tab idMso="TabHome" visible="false">
      </tab>
    </tabs>
  </ribbon>
</customUI>
```

代码中 TabHome 代表”开始”选项卡，对 visible 参数赋值为“false”表示隐藏此选项卡。要注意代码中每个字符的大小写形式，不能有任何错误。

效果：如图 21.10 所示。

图 21.10　隐藏”开始”选项卡

本例文件参见光盘：..\第二十一章\21-3 隐藏开始选项卡.xlsm

21.2.4　创建新选项卡：tab

语法：

```
<tab id="AA" visible ="BB"label="CC" insertAfterMso="DD"insertBeforeMso=
"EE"keytip="FF">
</tab>
```

其中参数 id 表示新选项卡的 id，必须为选项卡指定唯一的 id 名称，不能与其他选项卡同名；visible 代表选项卡的可见状态；label 表示选项卡显示在屏幕上的名称，允许重名；insertAfterMso 表示新选项卡放置在此参数所指定的选项卡之后；insertBeforeMso 表示新选项卡放置在此参数所指定的选项卡之前，不能与 insertAfterMso 同时出现，当同时忽略这两个参数时表示新选项卡放置在最后位置；keytip 表示选项卡的加速键，也称快捷键。

语法中的 AA、BB、CC、DD、EE 等都是占位符，在编写代码时可根据需求修改其值。

对 id、Label 和 keytip 参数赋值时不区分大小写，对其他参数赋值时，由于皆采用内部常量，所以必须区分大小写。

visible、label、insertAfterMso、insertBeforeMso 和 keytip 等为可选参数。

所有参数之间没有顺序要求，任何一个写在前面都可以。不过有三点值得注意，其一是参数与参数之间需要至少一个空格，其二是在对参数赋值时必须使用半角的引号，其三是逻辑值必须全部小写。

案例：创建名为“E 灵”的空选项卡，显示在“开始”选项卡之后，加速键为 B。

代码：

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui">
  <ribbon startFromScratch="false">
    <tabs>
      <tab id="NewTab" visible="true" label="E 灵" insertAfterMso="TabHome"
keytip= "B">
      </tab>
    </tabs>
  </ribbon>
</customUI>
```

效果：如图 21.11 所示。

图 21.11 在“开始”选项卡之后创建新选项卡

本例文件参见光盘：..\第二十一章\21-4 创建新选项卡.xlsm

21.2.5 创建新组：group

语法：

```
<group id="AA" visible="BB" label="CC" insertAfterMso="DD"insertBeforeMso=
"EE">
</group>
```

语法表中 id、visible、label 参数与创建选项卡时的参数规则一致，不同的是 insertAfterMso 和 insertBeforeMso 两个参数。只有新组位于内置选项卡中时才需要使用这两个参数之一表示新组的位置，否则直接忽略参数即可。

案例：在“E 灵”选项卡中创建一个名为“财务工具”的新组，在“开始”选项卡的“字体”组之后也创建一个为“财务工具”的新组。

代码：

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui">
  <ribbon startFromScratch="false">
    <tabs>
```

```
    <tab id="NewTab" visible="true" label="E 灵" insertAfterMso="TabHome"
keytip= "B">
      <group id="Group1" visible="true" label="财务工具">
      </group>
    </tab>
    <tab idMso="TabHome" visible="true">
      <group id="Group2" visible="true" label="财务工具" insertBeforeMso=
"GroupFont">
      </group>
    </tab>
    </tabs>
  </ribbon>
</customUI>
```

由于第一个新组处于自定义的选项卡中，所以不需要指定位置。第二个新组放在“开始”选项卡的“字体”组之后，所以指定 tab 的 id 时改用“idMso”，同时对 insertBeforeMso 参数赋值为“GroupFont”，即“字体”组。如图 21.12 和图 21.13 所示分别是两个新组的外观。

效果：如图 21.12 和图 21.13 所示。

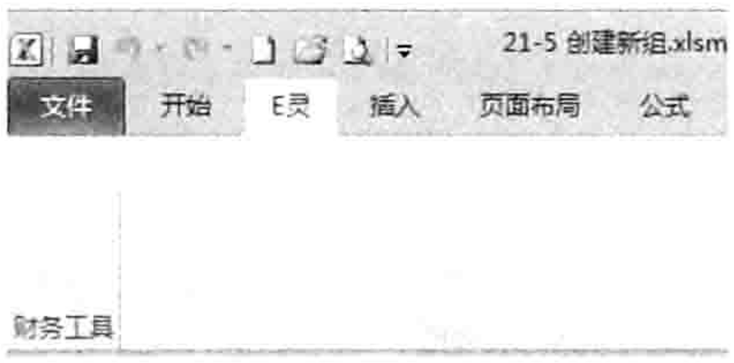


图 21.12　在“E 灵”选项卡中添加“财务工具”组

图 21.13　在“开始”选项卡插入新组

表 21-2 中罗列了 Excel 中大部分的组，读者在编写代码时可以调用这些组名称。

表 21-2　内置组的名称

组名	说明	组名	说明
GroupClipboard	剪贴板	GroupCalculation	计算
GroupFont	字体	GroupGetExternalData	获取外部数据
GroupAlignmentExcel	对齐方式	GroupConnections	连接
GroupNumber	数字	GroupSortFilter	排序和筛选
GroupStyles	样式	GroupDataTools	数据工具
GroupCells	单元格	GroupOutline	分级显示
GroupEditingExcel	编辑	GroupProofing	校对
GroupInsertTablesExcel	表格	GroupComments	批注
GroupInsertIllustrations	插图	GroupChangesExcel	更改
GroupInsertChartsExcel	图表	GroupWorkbookViews	工作簿视图
GroupInsertLinks	链接	GroupViewShowHide	显示
GroupInsertText	文本	GroupZoom	显示比例
GroupInsertBarcode	符号	GroupWindow	窗口
GroupThemesExcel	主题	GroupMacros	宏
GroupPageSetup	页面设置	GroupCode	代码
GroupPageLayoutScaleToFit	调整为合适大小	GroupControls	控件

续表

组名	说明	组名	说明
GroupPageLayoutSheetOptions	工作表选项	GroupXml	XML
GroupArrange	排列	GroupPictureStyles	调整
GroupFunctionLibrary	函数库	GroupPictureStyles	图片样式
GroupNamedCells	定义的名称	GroupArrange	排列
GroupFormulaAuditing	公式审核	GroupPictureSize	大小

本例文件参见光盘：..\第二十一章\21-5 创建新组.xlsm

21.2.6 创建对话框启动器：dialogBoxLauncher

语法：

```
<dialogBoxLauncher>
  <button id="AA" label="BB" screentip="CC" supertip="DD" onAction="EE"
keytip="FF"/>
</dialogBoxLauncher>
```

dialogBoxLauncher 代表对话框启动器，它总是和“<button/>”一起使用，因为对话框启动器只是一个容器，需要在其中放置一个按钮才能发挥作用。

对话框启动器中的按钮的 screentip 参数表示屏幕提示；supertip 表示更详细的提示内容；onAction 参数则表示单击对话框启动器时需要执行的子过程。

案例：在“E灵”选项卡的新组中创建一个对话框启动器，单击此启动器时可执行名为“工资条设计”的宏，同时需要对对话框启动器指定屏幕提示信息。

代码：

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui">
  <ribbon startFromScratch="false">
    <tabs>
      <tab id="NewTab" visible="true" label="E 灵" insertAfterMso="TabHome"
keytip= "B">
        <group id="Group1" visible="true" label="财务工具">
          <dialogBoxLauncher>
            <button id="dialogOne" screentip="工资条工具" supertip="单击可将工资
明细表转换成工资条" onAction="wage" keytip="G"/>
          </dialogBoxLauncher>
        </group>
      </tab>
    </tabs>
  </ribbon>
</customUI>
```

位于对话框启动器中的命令按钮所关联的子过程为“工资条设计”，加速键为<G>，即用<Alt+B+G>组合键可以执行此过程。

图 21.14 所示的是对话框启动器的外观，图 21.15 所示的是按<Alt>键后产生的快捷键提示，图 21.16 所示的是继续按<B>键后产生的快捷键提示。

效果：如图 21.14 至图 21.16 所示。

图 21.14　对话框启动器及其提示

图 21.15　选项卡的加速键

图 21.16　对话框启动器的加速键

回调：

将 Sub 过程关联到功能区中的按钮和将 Sub 过程关联到传统菜单的方法不同。前者需要使用回调参数，否则模块中的 Sub 过程无法被功能区代码所识别。

功能区的各种控件都有自己独特的回调参数，同时由于功能区中的控件种类繁多，很难记忆这些参数。好在 CustomUIEditor 软件支持自动生成 Sub 过程的外壳，其中包含所有参数。

对于本案例，当在 CustomUIEditor 软件中编辑好创建对话框启动器的代码后，单击工具栏的"Generate Callbacks"按钮即可看到以下包含参数的回调过程外壳：

```
'Callback for dialogOne onAction
Sub wage(control As IRibbonControl)
End Sub
```

此过程外壳对应于对话框启动器的命令按钮 button，当单击对话框启动器时会执行此 Sub 过程。

不过此代码并非存放在 CustomUIEditor 软件中，而是保存并关闭 CustomUIEditor 软件后进入工作簿的 VBE 界面，将以上代码复制到 VBE 界面的模块中。

由于 CustomUIEditor 软件只能产生程序外壳，所以需要根据实际需求手工补充代码。

为了简单地展示操作结果，本例采用以下简码：

```
Sub wage(control As IRibbonControl) '代码放置位置：模块中
  MsgBox "创建工资条.....请补充代码！", vbOKOnly, "友情提示"
End Sub
```

在模块中输入以上代码后，进入工作表界面单击刚创建的对话框启动器即可得到如图 21.17 所示效果，表示 Sub 过程与对话框已关联成功。

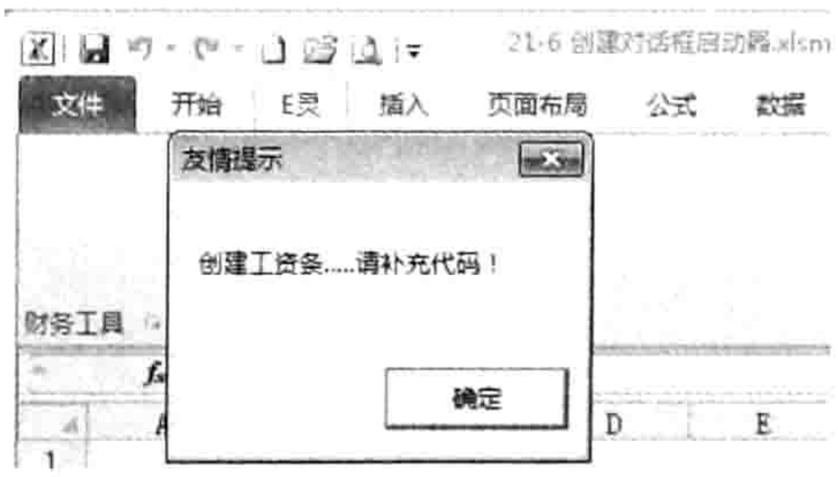


图 21.17　单击对话框启动器调用 Sub 过程

本例文件参见光盘：..\第二十一章\21-6 创建对话框启动器.xlsm

Sub 过程的过程名称带参数时不能使用<F8>键逐步运行，所以在编写代码时尽量不要使用参数，调试完成后再补充参数。另外，CustomUIEditor 软件在生成回调参数时不支持汉字，所以对

onAction 参数赋值时尽量采用字母。

21.2.7 在组中添加命令按钮：button

语法：

```
<button id="AA" label="BB" visible="CC" enabled="DD" imageMso="EE" size="FF"
onAction="GG" screentip="HH" supertip="II" keytip="JJ"/>
```

语法中的 enabled 参数用于控制按钮是否处于可用状态，赋值为“true”或者忽略此参数时表示可用，赋值为“false”时表示不可用；imageMso 参数表示为按钮指定一个内置的图标，如果需要采用外置的图标，则将参数名称替换为“image”，然后赋值为外置图标的名称；size 参数用于控制按钮图标的大小，赋值为“large”时表示此按钮显示为大图标，赋值为“normal”或者忽略此参数时将显示为小图标。

命令按钮的 label、visible、enabled、imageMso、size、onAction、keytip、screentip、supertip 都是可选参数。不过实际工作中 label 和 onAction 两个可选参数都需要赋值，否则无法使用此按钮。

命令按钮放在不同的地方有不同的语法，本语法仅适用于放在组中的命令按钮。

案例：在“E 灵”选项卡中创建一个名为“创建工资条”的命令按钮，其图标为内置的大图标“ControlLayoutStacked”，加速键为<C>；再创建一个名为“个人所得税”的命令按钮，用外置图片作为按钮的图标。

代码：

step 1 先准备一个 ico 格式的图标文件，例如如图 21.18 所示的文件。

step 2 在 CustomUIEditor 软件中打开需要添加功能区按钮的 xlsm 文件。

step 3 单击菜单中的“Insert”→“Office 2007 Custom UI Part”命令从而插入一个 customUI.xml 文件。

step 4 单击工具栏中第 3 个按钮，将图片文件插入到 customUI.xml 文件中，如图 21.19 所示。

图 21.18 图标文件

图 21.19 将图标插入到功能区代码文件中

step 5 在右边的代码窗口中录入以下代码：

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui">
  <ribbon startFromScratch="false">
    <tabs>
     <tab id="NewTab" visible="true" label="E 灵" insertAfterMso="TabHome"
keytip= "B">
       <group id="Group1" visible="true" label="财务工具">
         <button id="button1" label="创建工资条" visible="true" enabled="true"
imageMso="ControlLayoutStacked" size="large" onAction="wage" keytip="C"/>
         <button id="button2" label="个人所得税" visible="true" enabled="true"
image="A" size="large" onAction="tax" keytip="G"/>
       </group>
     </tab>
    </tabs>
  </ribbon>
```

```
</customUI>
```

以上代码可以创建一个选项卡、一个组和两个命令按钮。命令按钮的 size 参数赋值为“large”，所以将显示为大图标，一行只能显示一个按钮。如果将其赋值为 normal 或者忽略此参数时将显示为小图标，每列可以显示 3 个按钮。

由于命令按钮的 label、visible、enabled、imageMso、size、onAction、keytip、screentip、supertip 等属性都是可选参数，所以第一个按钮也可以简化为：

```
<button  id="button1"  label=" 创 建 工 资 条 "  imageMso="ControlLayoutStacked"
size="large" onAction=" wage "/>
```

但是在实际工作中尽量将所有可选参数书写完整，方便维护代码。例如以后需要修改某个参数时直接修改值即可，而不用去查找参数名称，同时也避免添加参数时录错字母造成代码无法运行。

命令按钮不可以直接放在选项卡中，它需要一个比选项卡低一级的容器，此容器可以是对话框启动器、组或者弹出式菜单。

如图 21.20 和图 21.21 所示的分别是大图标和小图标样式：

效果：如图 21.20 和 21.21 所示。

图 21.20　在“E 灵”选项卡中插入大命令按钮

图 21.21　在“E 灵”选项卡中插入小命令按钮

回调：

单击 CustomUIEditor 软件的“Generate Callbacks”按钮将看到以下代码，读者可以将它复制到工作簿的模块中，然后根据需要在过程中添加代码。本章仅为展示功能区中各种按钮、菜单、复选框、下拉列表等控件的生成方法，对于 Sub 过程皆从略，读者可以将自己的 Sub 过程代码复制进去，从而将过程与功能区菜单关联起来。

```
'Callback for button1 onAction
Sub wage(control As IRibbonControl)
End Sub
'Callback for button2 onAction
Sub tax(control As IRibbonControl)
End Sub
```

本例文件参见光盘：..\第二十一章\21-7 在自定义选项卡中创建命令按钮.xlsm

21.2.8　创建切换按钮：toggleButton

语法：

```
<toggleButton  id="AA"  label="BB"  visible="CC"  enabled="DD"  imageMso="EE"
size="FF" onAction="GG" screentip="HH" supertip="II" keytip="JJ"/>
```

切换按钮 toggleButton 只能放在组 group 中，它与同样放在组中的命令按钮的语法完全一样。

案例：在“E 灵”选项卡中创建一个切换按钮，标题文本为“显示零值”，表示按下按钮时可显示零值，否则隐藏零值。

代码：

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui">
  <ribbon startFromScratch="false">
    <tabs>
      <tab id="NewTab"visible="true"label="E 灵"insertAfterMso="TabHome"keytip="B">
        <group id="Group1" visible="true" label="视图">
          <toggleButton id="toggleButton1"label="显示零值"visible="true" enabled=
"true"onAction="zero"imageMso="ChartTypeOtherInsertGallery"    size="large"
screentip="零值切换" supertip="按下时显示零值，弹起时不显示零值" keytip="L"/>
        </group>
      </tab>
    </tabs>
  </ribbon>
</customUI>
```

如果需要添加两个切换按钮，那么两个按钮的 id 绝对不能相同，其他参数允许相同，但是为了使用方便，有必要加以区分，包括 label 、imageMso、onAction、screentip 和 supertip 等参数皆要赋予不同的值。

切换按钮和命令按钮的区别在于切换按钮相当于两个命令按钮，单击时执行一个命令，再次单击时执行另一个命令，显示的外观样式也不一样。图 21.22 和图 21.23 分别是按下与弹起时的按钮外观。

切换按钮适用于视图切换类需求，可以通过切换按钮展示某类对象的显示状态。

效果：如图 21.22 和图 21.23 所示。

图 21.22　按下状态的切换按钮

图 21.23　弹起状态的切换按钮

回调：

单击 CustomUIEditor 软件的“Generate Callbacks”按钮将看到以下代码：

```
'Callback for toggleButton1 onAction
Sub zero(control As IRibbonControl, pressed As Boolean)
End Sub
```

参数中的 control 代表切换按钮，pressed 代表按钮的状态，当按下按钮时该值为 True，按钮弹起时该值为 False。所以可以根据该参数的值来编写对应的 Sub 过程代码。例如：

```
Sub zero(control As IRibbonControl, pressed As Boolean)'代码存放位置：模块中
   '如果切换按钮处于按下状态,那么显示零值，否则不显示零值
    If pressed = True Then ActiveWindow.DisplayZeros = True ElseActiveWindow.
DisplayZeros = False
End Sub
```

事实上，以上代码也可以简写为如下形式：

```
Sub zero(control As IRibbonControl, pressed As Boolean) '简写形式
```

```
    '零值的显示状态由切换按钮的状态决定
    ActiveWindow.DisplayZeros = pressed
End Sub
```

本例文件参见光盘：..\第二十一章\21-8 创建切换按钮.xlsm

21.2.9 标签与复选框：labelControl/checkBox

语法：

创建标签的语法如下：

```
<labelControl id="AA" label="BB"visible="CC"/>
```

Label 参数表示标签显示在屏幕上的文字，visible 参数表示可见性，赋值为“false”时可隐藏标签，赋值为“true”或者忽略此参数时可显示标签。

创建复选框的语法如下：

```
<checkBox id="AA" label="BB" visible="CC" enabled="DD" onAction="EE"
screentip="FF" supertip="GG" keytip="HH" />
```

复选框 checkBox 没有图标，所以没有 image、imageMso 和 size 等参数。除此之外，命令按钮有的其他参数复选框都有。

案例：在“E 灵”选项卡中添加一个标签和两个复选框。

代码：

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui">
  <ribbon startFromScratch="false">
    <tabs>
      <tab id="NewTab" visible="true" label="E灵" insertAfterMso="TabHome" keytip="B">
        <group id="Group1" visible="true" label="视图工具">
          <labelControl id="label1" label="单击时切换"/>
          <checkBox id="check1" label="隐藏错误值" visible="true" enabled="true"
onAction="ErrorConversion" screentip="错误值" supertip="切换错误值的显示状态"
keytip="C"/>
          <checkBox id="check2" label="隐藏零值" visible="true" enabled="true"
onAction="ZeroConversion" screentip="零值" supertip="切换零值的显示状态"
keytip="L"/>
        </group>
      </tab>
    </tabs>
  </ribbon>
</customUI>
```

以上代码可以创建一个名为“单击时切换”的文字标签和两个复选框，分别对应于“隐藏错误值”和“隐藏零值”。标签与复选框都没有图标。

效果：如图 21.24 所示。

回调：

单击 CustomUIEditor 软件的“Generate Callbacks”按钮将看到以下代码，读者可将它复制到模块中，然后根据需求对过程添加代码，本节重点在于创建功能区中的各类控件。

图 21.24 标签与复选框

```
'Callback for check1 onAction
Sub ErrorConversion(control As IRibbonControl, pressed As Boolean)
End Sub
'Callback for check2 onAction
Sub ZeroConversion(control As IRibbonControl, pressed As Boolean)
End Sub
```

参数中 control 代表复选框，pressed 代表复选框的状态，如果复选框已打钩，该参数的值为 true，否则值为 false。在随书光盘中有关于获取 control 的 id 号，以及根据参数的值来编写不同操作命令的源代码和思路分析，书中不再具体介绍。

本例文件参见光盘：..\第二十一章\21-9 创建标签与复选框.xlsm

21.2.10 在按钮之间添加分隔条：separator

语法：

```
<separator id="AA"/>
```

分隔条只需要一个 id 参数。将代码插入到两个命令按钮或者复选框、弹出式菜单之间就可以产生一个分隔条。

案例：创建两个复选框和两个命令按钮，在复选框与命令按钮之间添加分隔条，而且复选框和命令按钮分别采用 A、B、C、D 共 4 个加速键。

代码：

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui">
  <ribbon startFromScratch="false">
    <tabs>
      <tab id="NewTab" visible="true" label="E 灵" insertAfterMso="TabHome"
keytip="B">
        <group id="Group1" visible="true" label="Excel 界之精灵">
          <labelControl id="label1" label="单击时切换"/>
          <checkBox id="check1" label="隐藏错误值" visible="true" enabled="true"
onAction="错误值" screentip="错误值" supertip="切换错误值的显示状态" keytip="A"/>
          <checkBox id="check2" label="隐藏零值" visible="true" enabled="true"
onAction="零值" screentip="零值" supertip="切换零值的显示状态" keytip="B"/>
          <separator id="separator1"/>
          <button id="button1" label="创建工资条" visible="true" enabled="true"
imageMso="ControlLayoutStacked" size="large" onAction="工资条设计" keytip="C"/>
          <button id="button2" label="个人所得税" visible="true" enabled="true"
image="A" size="large" onAction="个人所得税" keytip="D"/>
        </group>
```

```
      </tab>
    </tabs>
  </ribbon>
</customUI>
```

本例的代码直接借用了前面两个案例所产生的标签、复选框和命令按钮，中间插入代码“<separator id="separator1"/>”即可。

分隔条与标签、命令按钮等控件 id 不能相同，否则代码会产生错误。

效果：如图 21.25 和图 21.26 所示。

图 21.25　在复选框与命令按钮之间添加分隔条

图 21.26　4 个组件的加速键

本例文件参见光盘：..\第二十一章\21-10 在复选框与命令按钮之间添加分隔条.xlsm

21.2.11　创建弹出式菜单：menu

语法：

```
<menu id="AA" label="BB" imageMso="CC"  size="DD"  itemSize="EE" visible= "FF"
screentip="GG" supertip="HH" keytip="II">
<button  id="AA2"  label="BB"  visible="CC"  enabled="DD"  imageMso="EE"
onAction="FF" screentip="GG" supertip="HH" keytip="II"/>
</menu>
```

弹出式菜单 menu 可通过 size 参数控制图标的显示大小，还可以指定其子菜单的大小，所以既有 size 参数又有 itemSize 参数。后者赋值为“true”表示所有子菜单都显示为大图标，忽略参数或者赋值为“false”时表示子菜单显示为小图标。

弹出式菜单 menu 没有 onAction 参数，如果使用了此参数，代码将产生错误。

弹出式菜单的子菜单没有 size 参数，统一通过弹出式菜单的 itemSize 参数控制大小。

弹出式菜单 menu 和子菜单 button 都可以通过 keytip 参数指定加速键。

弹出式菜单可以有多个子菜单，所以在“<menu>”与“<menu/>”之间可以复制多份“<button/>”产生多个命令按钮，不过要注意 id 不允许重复。

案例：在“E 灵”选项卡中创建一个弹出式菜单和两个子菜单。

代码：

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui">
  <ribbon startFromScratch="false">
    <tabs>
      <tab id="NewTab" visible="true" label="E 灵" insertAfterMso="TabHome"
keytip= "B">
        <group id="Group1" visible="true" label="Excel 界之精灵">
          <menu id="menu" label="统计工具" imageMso="AutoSum"  size="large"
itemSize="large">
            <button id="button1" label="多表数据汇总"  imageMso="QueryAppend"
```

```
onAction= "SheetsQather" />
          <button   id="button2"   label=" 按 颜 色 汇 总 "     imageMso=
"AppointmentColorDialog"  onAction="ColorQather" />
        </menu>
      </group>
    </tab>
  </tabs>
 </ribbon>
</customUI>
```

要注意按钮“<button/>”必须放在“<menu>”与“</menu>”之间，并且不能使用 size 参数，否则代码会出错。

本例中弹出式菜单对 size 和 itemSize 参数都赋值为“large”，所以弹出式菜单和子菜单都显示为大图标，效果如图 21.27 所示。如果删除这两个参数将得到如图 21.28 所示效果。

效果：如图 21.27 和图 21.28 所示。

图 21.27　弹出式菜单与两个子菜单

图 21.28　将弹出式菜单和子菜单都显示为小图标

回调：

单击 CustomUIEditor 软件的“Generate Callbacks”按钮将看到以下代码，读者可以将它复制到模块中，然后根据需求对过程添加代码，本节重点在于创建功能区中各类控件。

```
'Callback for button1 onAction
Sub SheetsQather(control As IRibbonControl)
End Sub
'Callback for button2 onAction
Sub ColorQather(control As IRibbonControl)
End Sub
```

本例文件参见光盘：..\第二十一章\21-11 创建弹出式菜单.xlsm

21.2.12　创建拆分按钮：SplitButton

语法：

```
<splitButton id="AA" size="BB" visible="CC" >
  <button id="AA2" label="BB"  imageMso="CC"  enabled="DD"  onAction="EE" />
  <menu id="AA3"   itemSize="BB" visible="CC" enabled="DD" keytip="EE" >
    <button id="A44" label="BB"  imageMso="CC"   onAction="DD" />
    <button id="AA5" label="BB"  imageMso="CC"   onAction="DD" />
  </menu>
</splitButton>
```

拆分按钮 splitButton 是一个容器，包括一个命令按钮 button 和一个弹出式菜单 menu，弹出

式菜单中有若干个子菜单 button。

使用拆分按钮时必须在“<splitButton>”之后通过“<button/>”语句添加一个命令按钮，然后再使用“<menu>”和“</menu>”创建弹出式菜单，在弹出式菜单中放置若干子菜单。

拆分按钮的图标大小由“<splitButton>”语句中的 size 参数控制，所以使用“<button/>”创建按钮时没有 size 属性。后面的“<menu>”也没有 size 参数，一并由“<splitButton>”语句中的 size 参数控制大小，不过它拥有 itemSize 参数，可以控制子菜单的显示大小。

案例：在“E 灵”选项卡中创建一个拆分按钮，其弹出式菜单中包含两个子菜单。

代码：

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui">
  <ribbon startFromScratch="false">
    <tabs>
      <tab id="NewTab" visible="true" label="E 灵 " insertAfterMso="TabHome"
keytip="B">
        <group id="Group1" visible="true" label="Excel 界之精灵">
        <splitButton id="splitButton1" size="large" visible="true" >
          <button id="button1" label="多表数据汇总" imageMso="QueryAppend"
enabled="true" onAction="SheetsQather " />
          <menu id="menu1"    itemSize="large" visible="true" enabled="true"
keytip="D" >
            <button id="button2" label="多表数据汇总" imageMso="QueryAppend"
onAction= "SheetsQather" />
            <button    id="button3"    label="    按    颜    色    汇    总    "
imageMso="AppointmentColorDialog"  onAction="ColorQather " />
            </menu>
          </splitButton>
        </group>
      </tab>
    </tabs>
  </ribbon>
</customUI>
```

拆分按钮 splitButton 的 size 参数赋值为“large”，所以按钮将显示为大图标。拆分按钮是一个容器，它的 size 参数决定了它的子元素的大小；弹出式菜单 menu 也是一个容器，它的 itemSize 参数决定了它的子菜单的大小，所以在拆分按钮中，作为子元素的命令按钮和弹出式菜单本身都没有 size 参数，它们的大小由父对象控制。

总体而言，拆分按钮包括 3 个元素：命令按钮、弹出式菜单和子菜单。通常是弹出式菜单中包含了多个具有相似功能的工具，将其中常用的那一个工具提取出来作为默认的命令按钮。例如“开始”选项卡中的“合并后居中”就是一个拆分按钮，它的弹出式菜单中包含了与合并相关的多个工具，但是“合并后居中”工具比较常用，所以将它设为默认按钮。

本例代码创建的弹出式菜单包含两个子菜单，将第一个子菜单作为默认的命令按钮，所以执行“多表数据汇总”既可以从弹出式菜单中执行也可以单击拆分按钮上端的命令按钮。

图 21.29　拆分按钮的外观

效果：如图 21.29 所示。

回调：

单击 CustomUIEditor 软件的“Generate Callbacks”按钮将看到以下代码，读者可将它复制到模块中，然后根据需求对过程添加代码，本节重点在于创建功能区中的各类控件。

```
'Callback for button1 onAction
Sub SheetsQather(control As IRibbonControl)
End Sub
'Callback for button3 onAction
Sub ColorQather(control As IRibbonControl)
End Sub
```

由于拆分按钮的默认命令和弹出式菜单的第一个命令按钮共用一个 Sub 过程，所以 CustomUIEditor 软件只生成两个过程外壳。

本例文件参见光盘：..\第二十一章\21-12 创建拆分按钮.xlsm

21.2.13 创建下拉列表：DropDown

语法：

```
<dropDown id="AA" showLabel="BB" label="CC" onAction="DD" enabled="EE">
  <item id="AA2" label="BB" imageMso="CC" />
  <item id="AA3" label="BB" imageMso="CC" />
</dropDown>
```

语法中的 dropDown 表示创建下拉列表，它的 label 参数表示下拉列表旁边显示的文字，showLabel 参数用于控制是否显示该文字，赋值为“false”时可隐藏文字。onAction 参数决定下拉列表关联的宏，它和弹出式菜单完全不同，弹出式菜单是每个子菜单都关联一个宏，而下拉列表控件只有一个宏，关联到 dropDown 自身，其列表中的子元素没有 onAction 参数。

“<item>”语句用于创建列表项目，可对列表项目设置显示的文字标签，以及图标。若需要调用内置图标就改用 imageMso 参数，若需要调用自定义的图片文件就用 image 参数。

案例：在“开始”选项卡中创建一个名为“定位”的下拉列表，列表中包含“错误值”、“空单元格”、“公式”、“负数”和“可见单元格”。

代码：

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui">
  <ribbon startFromScratch="false">
    <tabs>
        <tab idMso="TabHome">
        <group id="Group1" visible="true" label="E灵">
          <dropDown id="dropDown1" showLabel="true" label="定位" onAction=
"locate" enabled="true">
            <item id="错误值" label="错误值" imageMso="FunctionsDateTimeInsert
Gallery" />
            <item id="空单元格" label="空单元格" imageMso="FunctionsFinancialInsert
Gallery" />
            <item id="公式" label="公式" imageMso="FunctionsLogicalInsert
Gallery" />
            <item id="负数" label="负数" imageMso="FunctionsLookupReferenceInsert
Gallery" />
            <item id="可见单元格" label="可见单元格" imageMso="FunctionsText
InsertGallery" />
          </dropDown>
        </group>
      </tab>
```

```
    </tabs>
  </ribbon>
</customUI>
```

代码中“<tab idMso="TabHome">”表示“开始”选项卡，所以它后面的“<group>”命令所创建的组将显示在“开始”选项卡中。同时，由于未指定新组的位置，所以将自动放在“开始”选项卡的最右端。

效果：如图 21.30 所示。

图 21.30　下拉列表

回调：

单击 CustomUIEditor 软件的“Generate Callbacks”按钮将看到以下代码：

```
'Callback for dropDown1 onAction
Sub locate(control As IRibbonControl, id As String, index As Integer)
End Sub
```

其中参数 control 代表下拉列表控件，id 代表用户从列表中选择的项目的 id，例如选择第二项时返回“空单元格”。参数 index 代表用户选择的下拉列表项目的索引号，从 0 开始，所以用户选择列表中第一项时该参数返回值为 0。

通过以下 Sub 过程可以更深入地了解各参数含义：

```
Sub locate(control As IRibbonControl, id As String, index As Integer)
 MsgBox "您已选择了列表框控件 " & control.id & " 的第" & index + 1 & "个子元素:    "
& id
End Sub
```

本例文件参见光盘：..\第二十一章\21-13 创建下拉列表.xlsm

21.2.14　创建编辑框：editBox

语法：

```
<editBox  id="AA"  label="BB"  imageMso="CC"  sizeString="DD"  maxLength="EE"
visible="FF" showLabel="GG"   onChange="HH" keytip="II" />
```

其中 label 参数表示显示在编辑框旁边的文字；imageMso 参数表示调用内置图标，改用 image 则可以调用自定义的图片文件作为图标，它将显示在编辑框的最左端；sizeString 参数用于控制编辑框的宽度，这个宽度是由赋值的字符串所占的宽度决定的，而非赋值的内容决定的，例如赋值为“999”表示编辑框的宽度等于这 3 个字所占的宽度；maxLength 参数用于控制编辑框中录入的字符的数量，赋值为 2 则表示在编辑框中输入字符时不能超过两位；showLabel 参数用于控制编辑框旁边的标签的显示状态；onChange 参数表示在编辑框中输入值后按<Enter>键时需要执行的 Sub 过程名称，类似于命令按钮的 onAction 参数。

案例：创建一个“E 灵”选项卡，位于“开始”选项卡之前，在其中创建一个名为“查找”的编辑框。

代码：

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui">
  <ribbon startFromScratch="false">
  <tabs>
```

```
    <tab id="rxtabCustom" label="E灵" insertBeforeMso="TabHome">
      <group id="NewCustom" label="http://excelbbx.net">
        <editBox id="FindTxt" label="查找" imageMso="ZoomPrintPreviewExcel"
sizeString ="9999999999" maxLength="10" visible="true" showLabel="true"
onChange= "Click" keytip="R" />
      </group>
      </tab>
    </tabs>
  </ribbon>
</customUI>
```

代码中 sizeString 参数赋值为 10 个 9，表示编辑框的宽度等于 10 个 9 的宽度；maxLength 参数赋值为 10 表示顶多输入 10 位数，虽然此值是数值，但也需要在前后添加半角的双引号。

编辑框控件包含图标、标签和编辑框 3 项内容，其外观如图 21.31 所示。

当在编辑框中录入的字符超过 10 位时将会产生如图 21.32 所示的提示。

效果：如图 21.31 和图 21.32 所示。

图 21.31　图标、标签与编辑框

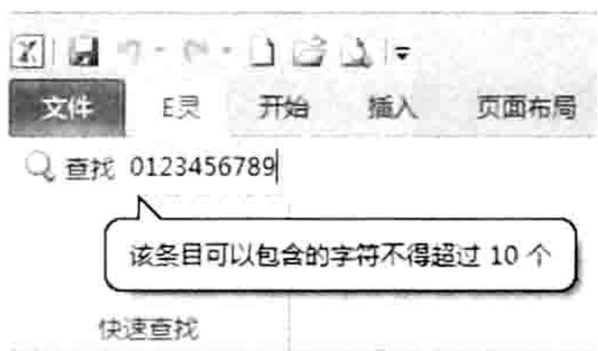


图 21.32　输入超过 10 位产生提示

回调：

单击 CustomUIEditor 软件的“Generate Callbacks”按钮将看到以下代码：

```
'Callback for FindTxt onChange
Sub Click(control As IRibbonControl, text As String)
End Sub
```

其中参数 control 代表下拉列表控件，text 代表在编辑框中录入的文本。

本例文件参见光盘：..\第二十一章\21-14 创建编辑框.xlsm

21.2.15　锁定或隐藏内置功能

语法：

功能区中所有内置的命令都可以被锁定，即被禁止使用，锁定内置命令的语法如下：

```
<commands>
  <command idMso="AA" enabled = "BB" />
</commands>
```

其中“<commands>”与“<ribbon>”是同级别的对象，所以不能放在“<ribbon>”与“<ribbon/>”之间。

“<command />”代表调用内置命令，通过 idMso 参数指定内置命令的 id 即可。当 enabled 参数赋值为“false”时表示禁用此内置命令。

案例：禁用内置的“另存为”、“合并居中”、“复制”和“剪切”功能。

代码：

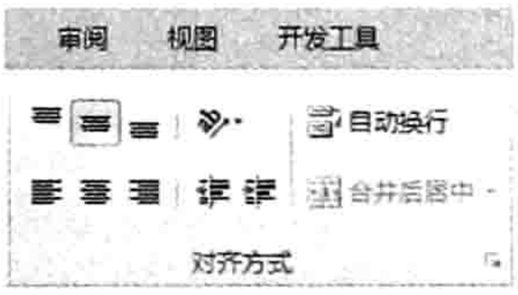
```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui">
<commands>
  <command idMso="FileSaveAs" enabled = "false" />
  <command idMso="MergeCenterMenu" enabled = "false"  />
  <command idMso="Copy" enabled = "false" />
  <command idMso="Cut" enabled = "false" />
</commands>
</customUI>
```

禁用内置命令直接将“<commands>”与“</commands>”放置在“<customUI>”与“</customUI>”壳中。并且将 enabled 赋值为“false”即可。

内置命令可以被禁用，但是不能被隐藏，command 对象是没有 visible 参数的。

不过内置的选项卡 tab 和组 group 支持 visible 参数，可以隐藏任意内置组或选项卡。

效果：如图 21.33 至 21.35 所示。



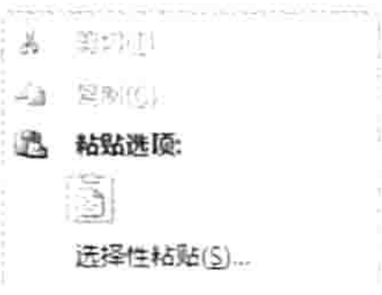


图 21.33 禁用“另存为”命令　图 21.34 禁用“合并后居中”命令　图 21.35 禁用“复制”与“剪切”命令

本例文件参见光盘：..\第二十一章\21-15 禁用内置功能.xlsm

21.3 使用回调函数强化功能区

回调函数专用于功能区控件，对自定义的功能区控件使用回调函数能强化控件的功能，实现动态获取数据，并根据获取的值调整控件的某种状态。

熟用回调函数后，在开发功能区控件领域将呈现另一片天地。

21.3.1 为什么需要使用回调函数

使用回调函数后，定义功能区控件的代码可以获取单元格中的值或者 Sub 过程中变量的值，从而使自定义的功能区控件更灵活，能满足更多的需求。

例如“21-8 创建切换按钮.xlsm”文件中的 VBA 代码可以读取切换按钮的显示状态，然后根据其状态去调整工作表中的零值的状态。但是这只是单向的控制，不够完美，若切换按钮也能根据当前工作表的零值的显示状态调整自己的状态则更人性化。

简而言之，使用回调函数就是为了方便在 VBA 中控制功能区中某个控件的某个属性。

21.3.2 回调函数详解

回调函数绝大多数是以 get 开头的，例如 getLabel、getText、getImage 等，表示控件对象的某个参数在后期通过 VBA 的 Sub 过程赋值，而非在前期编写功能区代码时赋值。

当然 onChange 和 onAction 也是一种回调。

表 21-3 中提供了所有回调函数的名称及其功能描述。

表 21-3 回调函数

函数	功能	函数	功能
getDescription	获取控件描述	getSelectedItemID	获取选中的子项目的 ID
getEnabled	确定按钮是否可用	getSelectedItemIndex	获取选中的子项目的序号
getImage	获取图像	getText	获取文本
getKeytip	获取控件的加速键	getItemHeight	获取子项高度
getLabel	获取控件的标签	getItemWidth	获取子项宽度
getScreentip	获取控件的提示	getTitle	获取标题
getSupertip	获取控件的详细提示	getContent	获取 XML 代码内容
getShowImage	判断控件是否显示图标	GetEnabledMso	获取内置控件的可用性
getShowLabel	判断控件是否显示标签	GetImageMso	获取内置控件的图像
getSize	获取控件的大小	GetLabelMso	获取内置控件的标签
getVisible	获取控件的可见性	GetPressdMso	判断内置控件是否被按下
getItemCount	获取子项的数量	GetScreenTipMso	获取内置控件的提示
getItemID	获取子项的 ID	GetSuperTipMso	获取内置控件的详细提示
getItemImage	获取子项的图像	GetVisibleMso	获取内置控件的可见性
getItemLabel	获取子项的标签	onChange	指定文本改变时执行的宏名称
getItemScreentip	获取子项的提示	onAction	指定单击按钮时执行的宏名称
getItemSupertip	获取子项的详细提示		

其中应用最频繁的是以下几个回调函数：getEnabled、getImage、getItemImage 、getLabel、getVisible 和 onAction。

1. getEnabled

getEnabled 函数表示从 VBA 代码中取值，并根据该值决定控件是否可用。getEnabled 函数其实就是动态的 enabled 参数。换而言之，一个控件是否处于禁用状态，可以通过回调函数 getEnabled 来决定，它的返回值决定控件是否可用，而非编写功能区代码时直接赋值为“true”或者“false”来决定。

例如功能区中的自定义命令按钮的功能是合并单元格的值，那么使用 getEnabled 后可以在 VBA 代码中判断活动表是否为图表，以及 Selection 是图片还是区域，如果活动表是图表或者 Selection 是图片，那么传递一个逻辑值 false 给 getEnabled 函数，该函数再将 false 传递给命令按钮 button，那么命令按钮将自动显示为禁用状态，从而避免执行命令出错。

以上思路可以用于所有支持回调函数的控件中。

再如某自定义的命令按钮用于汇总上月的生产数据，领导要求每月 4 日之前汇总完毕。在此前提下，可以通过 getEnabled 函数获取当前日期，假设日期是 1 日、2 日、3 日，则按钮处于可用状态，在这 3 天可以单击按钮实现汇总上月的数据；如果日期大于 3 日则按钮自动呈现禁用状态。此思路可以让按钮具有智能，根据需求自动调节。

getEnabled 函数的用法如下：

（1）在编写功能区控件的代码时，将 getEnabled 替换原来的 enabled 参数，然后对参数指定一个 sub 过程名称，而不是使用逻辑值“true”或“false”；

（2）在 CustomUIEditor 软件中单击工具栏中的“Generate Callbacks”按钮即可看到 getEnabled 的回调过程外壳。假设对 getEnabled 函数赋值为“ABC”，那么将产生以下代码：

```
Sub ABC(control As IRibbonControl, ByRef returnedVal)
```

```
End Sub
```

其中参数 control 代表当前控件，参数 returnedVal 代表传递给 getEnabled 函数的值，所以在 Sub 过程中根据需求对变量 returnedVal 赋值即可，该值会在显示控件时由 getEnabled 函数传递给控件。

（3）将回调过程的程序外壳复制到 VBE 界面的模块中，然后在该过程中对参数 returnedVal 赋值，该值将在显示控件时自动传递给控件，从而改变控件的显示状态。

21.3.3 节会展示此功能的具体操作步骤。

所有具有 enabled 参数的控件都支持 getEnabled 函数。

2. getImage

getImage 函数表示从 VBA 代码中取值，并根据该值决定控件的图标名称。getImage 函数其实相当于动态的 image 参数。

和获取 getEnabled 的回调参数的方法一样，当在开发功能区控件的代码中使用了 getImage 函数时，在 CustomUIEditor 软件中单击工具栏中的“Generate Callbacks”按钮即可看到 getImage 函数的回调过程外壳。假设对 getImage 函数赋值为“ABC”，那么 getImage 函数对应的回调过程外壳如下：

```
Sub ABC(control As IRibbonControl, ByRef returnedVal)
End Sub
```

参数 control 代表当前控件，参数 returnedVal 代表传递给 getImage 函数的值，所以在 Sub 过程中根据需求对变量 returnedVal 赋值即可，该值会在显示控件时由 getImage 函数传递给控件。

所有具有 Image 参数的控件都支持 getImage 函数。

3. getItemImage

getItemImage 函数表示从 VBA 代码中取值，并根据该值决定控件的图标名称。getImage 函数其实就是动态的 itemImage 参数。

假设对 getItemImage 函数赋值为“ABC”，那么 getItemImage 函数对应的回调过程外壳如下：

```
Sub ABC(control As IRibbonControl, index As Integer, ByRef returnedVal)
End Sub
```

参数 control 代表当前控件，参数 index 代表控件的子元素的索引号，参数 returnedVal 代表传递给 getItemImage 函数的值。

getItemImage 函数仅用于 comboBox、dropdown、gallery 三种带有下拉框的控件。

4. getLabel

getLabel 函数表示从 VBA 代码中取值，并根据该值决定控件显示在功能区中的字符。getLabel 函数其实就是动态的 label 参数。

假设对 getLabel 函数赋值为“ABC”，那么 getLabel 函数对应的回调过程外壳如下：

```
Sub ABC(control As IRibbonControl, ByRef returnedVal)
End Sub
```

参数 control 代表当前控件，参数 returnedVal 代表传递给 getLabel 函数的值。

所有具有 label 参数的控件都支持 getLabel 函数。

5. getVisible

getVisible 函数表示从 VBA 代码中取值，并根据该值决定控件是否可见。getVisible 函数其实

就是动态的 visible 参数。

假设对 getVisible 函数赋值为“ABC”，那么 getVisible 函数对应的回调过程外壳如下：

```
Sub ABC(control As IRibbonControl, ByRef returnedVal)
End Sub
```

参数 control 代表当前控件，参数 returnedVal 代表传递给 getVisible 函数的值。

所有具有 visible 参数的控件都支持 getVisible 函数。

21.3.3 创建在每月的 1 日到 3 日才能使用的按钮

案例要求：在“开始”选项卡中创建一个名为“汇总上月资料”的按钮，在每月的 1 到 3 日打开工作簿时该按钮呈可用状态，其他时间打开工作簿时呈禁用状态。

知识要点：getEnabled

操作步骤：

step 1 新建一个空白工作簿，然后保存为 xlsm 格式的文件。

step 2 打开 CustomUIEditor 软件，并从软件中打开刚才所保存的工作簿。

step 3 单击菜单中的“Insert”→“Office 2007 Custom UI Part”命令从而插入一个 customUI.xml 文件，然后在代码窗口中录入以下代码，用于在“开始”选项卡创建新组及命令按钮：

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui">
  <ribbon startFromScratch="false">
    <tabs>
    <tab idMso="TabHome">
      <group id="Group1" visible="true" label="汇总">
        <button id="button1" label="汇总上月资料" visible="true" getEnabled=
"ABC" imageMso="ControlLayoutStacked" size="large" onAction="Qather" />
      </group>
    </tab>
  </tabs>
 </ribbon>
</customUI>
```

step 4 单击“Generate Callbacks”按钮产生 getEnabled 与 onAction 两个回调函数所对应的过程外壳，将它复制出来，然后保存并关闭 CustomUIEditor 软件。

step 5 使用 Excel 打开前面创建的工作簿文件，使用<Alt+F11>组合键打开 VBE 窗口。

step 6 单击菜单中的“插入”→“模块”命令，然后将刚才所复制的两段代码粘贴到模块中。

step 7 对名为 ABC 的过程添加赋值语句，最终代码如下（过程 Qather 的代码从略，本例主要展示回调函数 getEnabled 的应用思路）：

```
Sub ABC(control As IRibbonControl, ByRef returnedVal)'代码存放位置：模块中
  If Day(Date) >= 1 And Day(Date) <= 3 Then '如果打开文件的日期大于等于 1 而且小于
等于 3
    returnedVal = True                      '将变量赋值为 true
  Else                                      '否则
    returnedVal = False                     '将变量赋值为 false
  End If
End Sub
```

step 8 保存工作簿并重启，如果当前日期不属于1日、2日或者3日，那么按钮将呈禁用状态，如图21.36所示。

图21.36 禁用状态的命令按钮

案例补充：

（1）在“ABC”过程中，首先利用代码“Day(Date)”计算打开工作簿的日期，如果处于1日到3日，那么对returnedVal变量赋值为“True”，否则赋值为“False”，此值将传递给getEnabled函数，getEnabled函数再根据此值决定命令按钮的可用状态。

（2）在定制功能区的代码中对 enabled 参数赋值时采用“"false"”或者“"true"”，它是一个所有字符都小写的文本字符串，但是回调过程对getEnabled函数赋值时需要采用逻辑值“False”或者“True”，不能添加双引号，也不区分大小写。

CD-ROM 本例文件参见光盘：..\第二十一章\21-16 1到3号能使用的命令按钮.xlsm

21.3.4 创建按下与弹起时自动切换图标的按钮

案例要求：在“视图”选项卡中创建一个“显示零值”的命令按钮，按钮被按下时可显示零值，按钮的图标显示为“√”；按钮弹起时隐藏零值，图标显示为“×”。

知识要点：getImage、onLoad

操作步骤：

step 1 新建一个空白工作簿，然后保存为xlsm格式的文件。

step 2 打开CustomUIEditor软件，并从软件中打开刚才所保存的工作簿。

step 3 单击菜单中的“Insert”→“Office 2007 Custom UI Part”命令从而插入一个customUI.xml文件，然后在代码窗口中录入以下代码，用于在“视图”选项卡创建新组及切换按钮：

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui" onLoad=
"Intialize">
  <ribbon startFromScratch="false">
    <tabs>
      <tab idMso="TabView">
        <group id="Group1"label="零值控制"insertAfterMso="GroupViewShowHide">
          <toggleButton id="toggleButton1" label="显示零值" visible="true"
enabled= "true" onAction="zero" getImage="getImage" size="large" screentip=
"零值切换" supertip="按下时显示零值，弹起时不显示零值" keytip="L"/>
        </group>
      </tab>
    </tabs>
  </ribbon>
</customUI>
```

要注意第一句代码的末尾加了“onLoad="Intialize"”，表示调用此代码时先执行名为“Intialize”的Sub过程。

另外在“<tab>”语句中采用了“idMso="TabView"”表示当前对象是“视图”选项卡。

对切换按钮 toggleButton 指定图标时采用了“getImage="getImage"”，表示按钮的图标通过名为“getImage”的Sub过程来指定。

step 4 单击“Generate Callbacks”按钮产生Intialize、zero与getImage 3个回调过程所对应的过程外壳，将它们复制出来，然后保存并关闭CustomUIEditor软件。

step 5　用 Excel 打开前面创建的工作簿文件，使用<Alt+F11>组合键打开 VBE 窗口。

step 6　单击菜单中的“插入”→“模块”命令，然后将刚才所复制的三段代码粘贴到模块中。

step 7　对 3 个过程添加代码，使其能正常工作，最终代码如下：

```
Dim bl As Boolean
Dim rib As IRibbonUI
Sub Intialize(ribbon As IRibbonUI)'①代码存放位置：模块中②随书光盘中有每一句代码的
含义注释
  Set rib = ribbon
End Sub
```

以上过程将在启动工作簿时执行，作用是将功能区对象 Ribbon 赋予变量 rib，即载入缓存中，供其他代码随时调用。

```
'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
Sub zero(control As IRibbonControl, pressed As Boolean)
    ActiveWindow.DisplayZeros = pressed
    bl = pressed
  rib.Invalidate
End Sub
```

此过程对应于切换按钮的 onAction 参数，即单击切换按钮时可执行此过程。过程中的 pressed 参数代表按钮的状态，单击按钮时 VBA 会将按钮的状态传递给变量 pressed，而过程中的代码又将此变量的值传递给公共变量 bl，从而使后面的过程“getImage”可以接收此变量的值，然后根据变量的值决定为按钮指定何种图标。公共变量 bl 的作用是作为过程“zero”与“getImage”的中转站，没有这个公共变量过程“getImage”就无法识别切换按钮的当前状态，从而也无法正确地对切换按钮的图标属性赋值。

```
'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
Sub getImage(control As IRibbonControl, ByRef returnedVal)
  If bl = False Then
    returnedVal = "DeclineInvitation"
  Else
    returnedVal = "AcceptInvitation"
  End If
  End Sub
```

此过程表示接收到公共变量 bl 的值后，根据变量的值决定参数 returnedVal 的值。而 returnedVal 参数的值会传递给切换按钮，从而决定切换按钮的图标。

step 8　保存并重启工作簿，然后进入“视图”选项卡，单击“显示零值”按钮，按钮将呈按下状态，同时按钮的图标显示为“√”，如图 21.37 所示，再次单击按钮时，工作表中的所有零值都会被隐藏起来，同时按钮的图标更新为“×”，如图 21.38 所示。

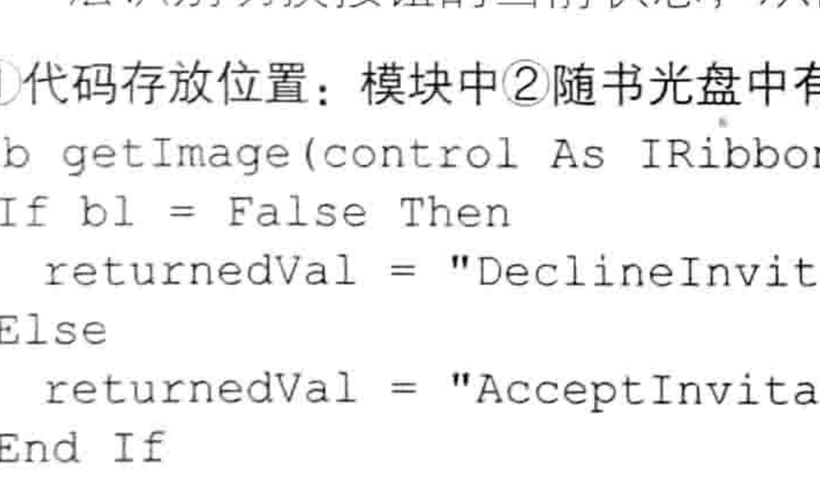


图 21.37　按钮按下时显示的图标

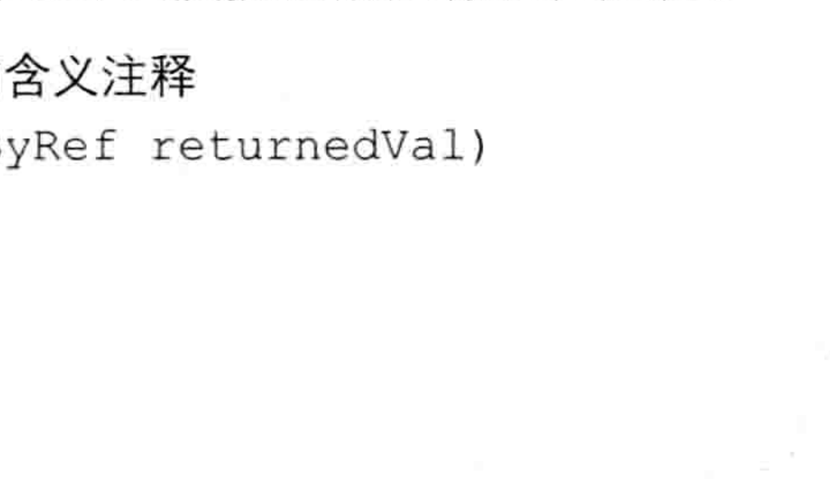


图 21.38　按钮弹起时显示的图标

案例补充：

（1）当单击切换按钮时，切换按钮的状态会传递给“zero”过程的 pressed 参数，但是不会传递给“getImage”过程，所以需要使用一个公共变量 bl 来做中转站，在过程“zero”中赋值，再在过程“getImage”中接收值。

（2）过程“Intialize”的作用是将功能区 IRibbonUI 赋予变量 rib，从而载入缓存供“rib.Invalidate”语句调用，没有过程“Intialize”就不能更新功能区，过程“getImage”中的 returnedVal 的值就不能传递给切换按钮了。

（3）本例重点在于根据按钮的状态修改按钮的图标，VBA 中的 Sub 过程并不完善。例如当打开工作簿时，不管工作表中是否显示零值，按钮都默认为弹起状态，图标为“×”。也就是说切换按钮的图标与状态并没有与零值的状态同步，只有单击按钮后才同步。请读者思考应该如何修改代码。

本例文件参见光盘：..\第二十一章\21-17 创建按下与弹起时自动切换图标的按钮.xlsm

21.3.5 创建一个能显示图形对象数量的标签

案例要求：当工作表中存在大量图形对象时，开启或者编辑工作簿的速度都会变慢。而且工作表中有时可能存在隐藏的图形对象，无法了解工作表中真实的图形对象数量。现要求在“开始”选项卡中创建一个标签，显示活动工作表的图形对象数量，包括隐藏的图形对象。

知识要点：getLabel、onLoad

操作步骤：

step 1 新建一个空白工作簿，然后保存为 xlsm 格式的文件。

step 2 打开 CustomUIEditor 软件，并从软件中打开刚才所保存的工作簿。

step 3 单击菜单中的“Insert”→“Office 2007 Custom UI Part”命令从而插入一个 customUI.xml 文件，然后在代码窗口中录入以下代码，用于在“开始”选项卡创建新组及标签控件：

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui" onLoad=
"Intialize">
  <ribbon startFromScratch="false">
    <tabs>
      <tab idMso="TabHome">
        <group id="Group1"  label="图形对象数量">
          <labelControl id="Label1" getLabel="getLabel"/>
        </group>
      </tab>
    </tabs>
  </ribbon>
</customUI>
```

以上代码中使用了回调函数 getLabel，表示通过 VBA 后期赋值。

step 4 单击“Generate Callbacks”按钮产生 Intialize 与 getLabel 两个回调过程所对应的过程外壳，将它复制出来，然后保存并关闭 CustomUIEditor 软件。

step 5 用 Excel 打开前面创建的工作簿文件，使用<Alt+F11>组合键打开 VBE 窗口。

step 6 单击菜单中的“插入”→“模块”命令，然后将刚才所复制的两段代码粘贴到模块中。

step 7 对两个过程添加代码，使其能正常工作，最终代码如下：

```
Public rib As IRibbonUI
Sub Intialize(ribbon As IRibbonUI)'①代码存放位置：模块中②随书光盘中有每一句代码的
含义注释
  Set rib = ribbon
End Sub
Sub getLabel(control As IRibbonControl, ByRef returnedVal)
returnedVal = ActiveSheet.Shapes.Count
End Sub
```

step 8 双击 Thisworkbook 对象进入工作簿事件代码窗口，然后录入以下代码：

```
'①代码存放位置：ThisWorkbook ②随书光盘中有每一句代码的含义注释
Private Sub Workbook_SheetActivate(ByVal Sh As Object)
  On Error Resume Next
  rib.Invalidate
End Sub
```

以上过程表示切换工作表时更新标签中显示的值。

step 9 保存并重启工作簿，“开始”选项卡中新建的标签将显示活动工作表中的图形对象数量，如图 21.39 所示。当激活其他工作表时，该标签的显示数值会自动更新。

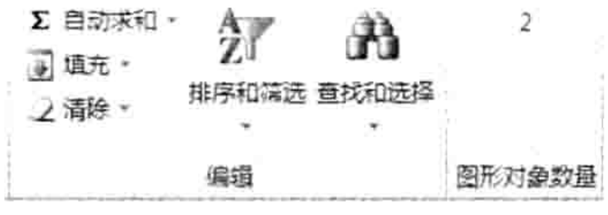


图 21.39 在标签中显示图形对象数量

案例补充：

（1）回调函数 getLabel 可用于所有支持 label 参数的功能区控件，用于替换 label 参数，从而实现 VBA 代码控制控件所显示的文字。本例中通过 Sub 过程“getLabel”获取活动工作表中的图形对象数量，并将值传递给参数 returnedVal，回调函数 getLabel 再将此值传递给标签。

（2）由于需要实时显示不同工作表中的图形对象，所以需要配合工作簿事件 SheetActivate 更新标签的值，从而确保切换工作表后标签的显示文字准确无误。

本例文件参见光盘：..\第二十一章\21-18 创建一个能显示图形对象数量的标签.xlsm

21.3.6 在功能区中快速查找

案例要求：在“开始”选项卡中创建一个下拉列表和一个编辑框，下拉列表决定查找方式，编辑框决定查找内容，两者结合实现快速定位目标单元格。

知识要点：onChange

操作步骤：

step 1 新建一个空白工作簿，然后保存为 xlsm 格式的文件。

step 2 打开 CustomUIEditor 软件，并从软件中打开刚才所保存的工作簿。

step 3 单击菜单中的“Insert”→“Office 2007 Custom UI Part”命令从而插入一个 customUI.xml 文件，然后在代码窗口中录入以下代码，用于在“开始”选项卡创建新组及下拉列表、编辑框：

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui">
  <ribbon startFromScratch="false">
  <tabs>
    <tab idMso="TabHome">
      <group id="group1" label="快速查找" insertBeforeMso="GroupFont">
        <dropDown  id="Style"  showLabel="true"  label=" 匹 配 方 式 "  onAction=
"dropDownChange" >
          <item id="xlWhole" label="xlWhole" imageMso="WatchWindow" />
          <item id="xlPart" label="xlPart" imageMso="ZoomPrintPreviewExcel" />
        </dropDown>
        <editBox id="FindTxt" label="查找内容" imageMso="ZoomPrintPreviewExcel"
sizeString="999999999999"  maxLength="30"  visible="true"  showLabel="true"
onChange="editBoxChange" keytip="R" />
      </group>
      </tab>
    </tabs>
  </ribbon>
</customUI>
```

step 4 单击“Generate Callbacks”按钮产生 dropDownChange 与 editBoxChange 两个回调过程所对应的过程外壳，将它复制出来，然后保存并关闭 CustomUIEditor 软件。

step 5 用 Excel 打开前面创建的工作簿文件，使用<Alt+F11>组合键打开 VBE 窗口。

step 6 单击菜单中的“插入”→“模块”命令，然后将刚才所复制的两段代码粘贴到模块中。

step 7 对两个过程添加代码，使其能正常工作，最终代码如下：

```
'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
Public str As String
Sub dropDownChange(control As IRibbonControl, id As String, index As Integer)
    str = id
End Sub
Sub editBoxChange(control As IRibbonControl, text As String)
    If Len(text) = 0 Then Exit Sub
    If str = "" Then MsgBox "请设置 lookat 参数": Exit Sub
    Dim FirstCell As Range, rng As Range
    With Cells
        Set  FirstCell  =  .Find(text,  LookIn:=xlValues,  lookat:=IIf(str  =
"xlWhole", xlWhole, xlPart))
        If Not FirstCell Is Nothing Then
            firstAddress = FirstCell.Address
            Do
                If rng Is Nothing Then Set rng = FirstCell Else Set rng =Union(rng,
FirstCell)
                Set FirstCell = .FindNext(FirstCell)
            Loop While FirstCell.Address <> firstAddress
        End If
    End With
    If Not rng Is Nothing Then
        rng.Select
    Else
        MsgBox "Sorry,未找到 "" & text & "" "
    End If
```

```
End Sub
```

以上代码中先声明了一个公共变量 Str，它作为两个 sub 过程的中转站传递下拉列表控件的值给编辑框，编辑框接收到值后根据该值决定搜索方式。至于搜索的内容则由编辑框的 text 参数决定。

step 8　保存并重启工作簿，在“开始”选项卡中将看到如图 21.40 所示的控件外观。

step 9　在编辑框中录入字母“T”并按<Enter>键，此时 Excel 弹出如图 21.41 所示的信息示框，表示未设置查找方式。

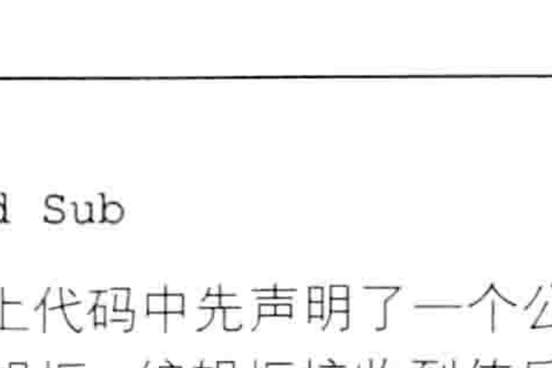


图 21.40　控件外观

图 21.41　查找前未指定查找方式

step 10　将匹配方式设置为 xlPart，然后在编辑框中录入字符“te”并按<Enter>键，程序会瞬间定位所有包括“te”的单元格，如图 21.42 所示。

step 11　将匹配方式设置为 xlWhole，并在编辑框中录入字符“tex”，然后按<Enter>键，程序会弹出如图 21.43 所示的提示框，表示按精确匹配方式查找时未找到目标。

图 21.42　糊模匹配方式查找 te

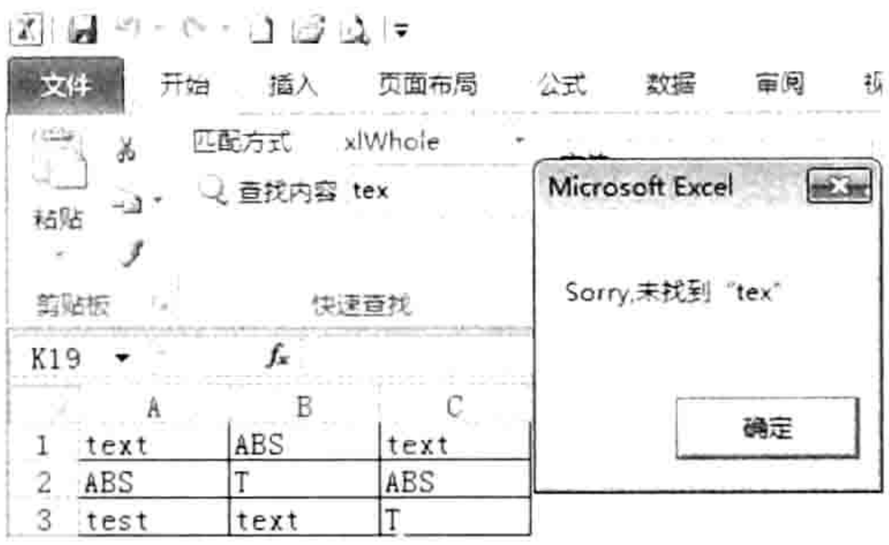


图 21.43　精确匹配方式查找 tex

案例补充：

（1）本例中执行查找任务时需要使用下拉列表的值和编辑框的值，由于编辑框无法获得下拉列表的值，所以先声明一个公共变量，在下拉列表中的“dropDownChange”过程中将值传递给变量，然后在编辑框中读取该变量的值。

（2）下拉列表的两个项目使用了 xlWhole 和 xlPart，也可以修改为汉字，例如“精确匹配”和“糊模匹配”，修改功能区代码后 Sub 过程也需要同步修改。

本例文件参见光盘：..\第二十一章\21-19 通过编辑框执行精确查找.xlsm

21.3.7　在组的标签处显示问候语

案例要求：修改前一案例的功能区代码，使“快速查找”组的文字（label 属性）显示为“您好”及当前日期。

知识要点：getLabel

操作步骤：

step 1 新建一个空白工作簿，然后保存为 xlsm 格式的文件。

step 2 打开 CustomUIEditor 软件，并从软件中打开刚才所保存的工作簿。

step 3 单击菜单中的“Insert”→“Office 2007 Custom UI Part”命令从而插入一个 customUI.xml 文件，然后在代码窗口中录入以下代码，用于在“开始”选项卡创建新组及命令按钮：

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui">
  <ribbon startFromScratch="false">
  <tabs>
    <tab idMso="TabHome">
      <group id="group1" getLabel="getLabel" insertBeforeMso="GroupFont">
        <dropDown  id="Style"  showLabel="true"  label="匹配方式"  onAction=
"dropDownChange" >
          <item id="xlWhole" label="xlWhole" imageMso="WatchWindow" />
          <item id="xlPart" label="xlPart" imageMso="ZoomPrintPreviewExcel" />
        </dropDown>
        <editBox id="FindTxt" label="查找内容" imageMso="ZoomPrintPreviewExcel"
sizeString="999999999999"  maxLength="30"  visible="true"  showLabel="true"
onChange="editBoxChange" keytip="R" />
      </group>
      </tab>
    </tabs>
  </ribbon>
</customUI>
```

代码中组 group 使用了回调函数 getLabel，表示通过 VBA 代码后期指定组的文字。

step 4 单击“Generate Callbacks”按钮产生 getLabel、dropDownChange 和 editBoxChange 3 个回调过程所对应的过程外壳，将其中 getLabel 的回调过程复制出来，然后保存并关闭 CustomUIEditor 软件。

step 5 用 Excel 打开前面创建的工作簿文件，使用<Alt+F11>组合键打开 VBE 窗口。

step 6 单击菜单中的“插入”→“模块”命令，然后将刚才所复制的代码粘贴到模块中。

step 7 对过程添加代码，使其能正常工作，最终代码如下：

```
'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
Sub getLabel(control As IRibbonControl, ByRef returnedVal)
 returnedVal = "您好 " & Format(Date, "yyyy-mm-dd AAAA")
End Sub
```

step 8 保存并重启工作簿，在“开始”选项卡中的新组将显示包含日期、星期的问候语，如图 21.44 所示。

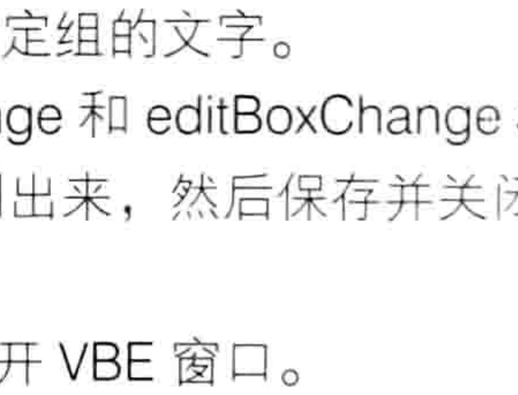


图 21.44 问候语

案例补充：

（1）本例是回调函数 getLabel 的又一应用，借助回调函数使功能区中自定义的控件更人性化，满足更多需求。

（2）本例只要求显示日期，所以过程比较简单，不需要更新内容。如果要求有时、分、秒，并且每秒钟变化，那么除了修改 Format 函数的第 2 参数外还得使用 Application.OnTime 方法创建计划任务。

本例文件参见光盘：..\第二十一章\21-20 在组的标签处显示问候语.xlsm

21.3.8 调用大图片创建下拉菜单

图片库是功能区独有的新功能，它可以实现实时预览，从而大大提升工作效率，专用名词称之为 gallery。

gallery 也是功能区的基本控件之一，不过由于它的应用比较复杂，必须配合回调函数才能使用，所以放在本章末尾介绍。

gallery 的语法如下：

```
<gallery id="AA" label="BB" size="CC" showLabel="DD" image="EE"
columns="FF" rows="GG" itemHeight="HH" itemWidth="II" supertip="JJ"
getItemCount="KK" getItemID="LL" getItemImage="MM" getItemSupertip="NN"
getItemLabel="OO" onAction="PP"/>
```

库的参数有近 20 个，相对于其他控件来说要复杂得多，不过大多是可选参数，实际使用时不需要每个参数都赋值。

参数中的 showLabel 用于控制是否显示标签，有 true 和 false 两个选项；columns 和 rows 分别代表库的显示方式，表示由几行、几列组成；itemHeight 和 itemWidth 参数分别代表子项目的显示高度和宽度；getItemCount、getItemID、getItemImage、getItemSupertip 和 getItemLabel 分别用于指定子项目的数量、id、图标、提示和标签。

接下来通过案例展示 gallery 的制作过程。

案例要求：在“开始”选项卡中添加一个库 gallery，库的子菜单中包含“合并居中”和“取消合并”，分别采用两张大照片作为图标，实现从菜单图标预览菜单功能。

知识要点：gallery、getItemCount、getItemID、getItemImage、getItemSupertip、getItemLabel

操作步骤：

step 1 新建一个空白工作簿，然后保存为 xlsm 格式的文件。

step 2 打开 CustomUIEditor 软件，并从软件中打开刚才所保存的工作簿。

step 3 单击菜单中的“Insert”→“Office 2007 Custom UI Part”命令从而插入一个 customUI.xml 文件，然后在代码窗口中录入以下代码，用于在“开始”选项卡创建新组及库：

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui">
  <ribbon>
    <tabs>
      <tab idMso="TabHome">
        <group id="group1" label="合并单元格" insertAfterMso="GroupClipboard">
          <gallery id="gallery1" label="合并单元格" size="large" showLabel="true"
imageMso="TableStyleClear"  columns="1" rows="2" itemHeight="157" itemWidth=
"142" supertip="合并单元格且保留所有数据" getItemCount="getItemCount" getItemID=
"getItemID"  getItemImage="getItemImage"  getItemSupertip=  "getItemSupertip"
getItemLabel="getItemLabel" onAction="Action"/>
        </group>
      </tab>
    </tabs>
  </ribbon>
</customUI>
```

以上代码可在“开始”选项卡中创建一个新组，组中包含一个图片库 gallery，库的行数为 2，列数为 1，每行的高度为 157，宽度为 142。

其中最重要的是使用了 getItemCount、getItemID、getItemImage、getItemSupertip、getItemLabel

5个回调函数，可以后期指定图片的路径、id、数量提示和显示字符等属性。

step 4 单击“Generate Callbacks”按钮产生 getItemCount、getItemImage、getItemLabel 和 Action 4 个回调过程所对应的过程外壳。事实上应该有 6 个才对，还需要包括 getItemSupertip 和 getItemID，由于未知原因并未一并罗列出来，笔者在此将其补充完整：

```
'Callback for gallery1 getItemCount
Sub getItemCount(control As IRibbonControl, ByRef returnedVal)
End Sub
'Callback for gallery1 getItemImage
Sub getItemImage(control As IRibbonControl, index As Integer, ByRef
returnedVal)
End Sub
'Callback for gallery1 getItemLabel
Sub getItemLabel(control As IRibbonControl, index As Integer, ByRef
returnedVal)
End Sub
'Callback for gallery1 onAction
Sub Action(control As IRibbonControl, id As String, index As Integer)
End Sub
'Callback for gallery1 getItemID
Sub getItemID(control As IRibbonControl, index As Integer, ByRef id)
End Sub
'Callback for gallery1 getItemSupertip
Sub getItemSupertip(control As IRibbonControl, index As Integer, ByRef
supertip)
End Sub
```

保存并关闭 CustomUIEditor 软件。

step 5 准备两个图片文件，分别作为合并居中与取消合并两个功能按钮的图标，从而使用户可以在执行代码之前预览效果。

两个图片尽量将到宽度调整为 142、高度调整为 157，与代码中指定的高度与宽度一致，然后将图片存放在与当前工作簿相同的路径中，如图 21.45 所示。

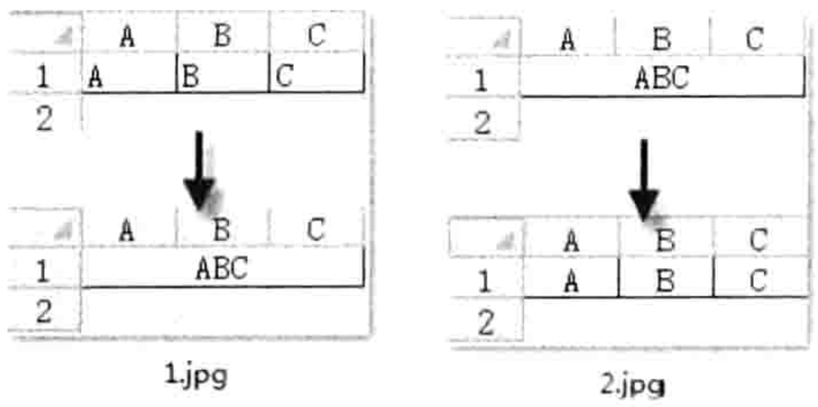


图 21.45 将要显示在库中的图片

step 6 用 Excel 打开前面创建的工作簿文件，使用<Alt+F11>组合键打开 VBE 窗口。

step 7 单击菜单中的“插入”→“模块”命令，然后将上述 6 段代码复制到模块中。

step 8 对 6 个过程添加代码，使其能正常工作，最终代码如下：

```
'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
Sub getItemImage(control As IRibbonControl, Index As Integer, ByRef
returnedVal)
    Set returnedVal = LoadPicture(ThisWorkbook.Path & "\" & Index + 1 & ".jpg")
End Sub
Sub getItemCount(control As IRibbonControl, ByRef returnedVal)
    returnedVal = 2
```

```
End Sub
Sub getItemID(control As IRibbonControl, Index As Integer, ByRef Id)
    Id = Index + 1
End Sub
Sub getItemSupertip(control As IRibbonControl, Index As Integer, ByRef
supertip)
    supertip = Array("将数组合并居中，但是保留所有合并前的数据", "取消合并居中，还原合
并前的状态，不丢失数据")(Index)
End Sub
Sub getItemLabel(control As IRibbonControl, Index As Integer, ByRef
returnedVal)
    returnedVal = Array("合并居中", "取消合并")(Index)
End Sub
Sub Action(control As IRibbonControl, Id As String, Index As Integer)
Call 合并居中(Index)
End Sub
```

事实上过程“Action”调用了另一个名为“合并居中”的过程，从而完成合并与取消两个功能。本例重点展示库的应用，而且由于过程“合并居中”的代码比较长，读者可从随书光盘中获取该过程源代码。

step 9 保存并重启工作簿，在“开始”选项卡中将看到创建的新组，组中包含一个名为“合并单元格”的库 gallery。单击“合并单元格”命令将弹出库的两个子项目，分别为“合并居中”和“取消合并”。从功能上讲，库 gallery 与弹出式菜单 menu 极为相似，不过库 gallery 更人性化，可以预览效果，如图 21.46 所示。当鼠标指向库的子元素时还提供屏幕提示，如图 21.47 所示。

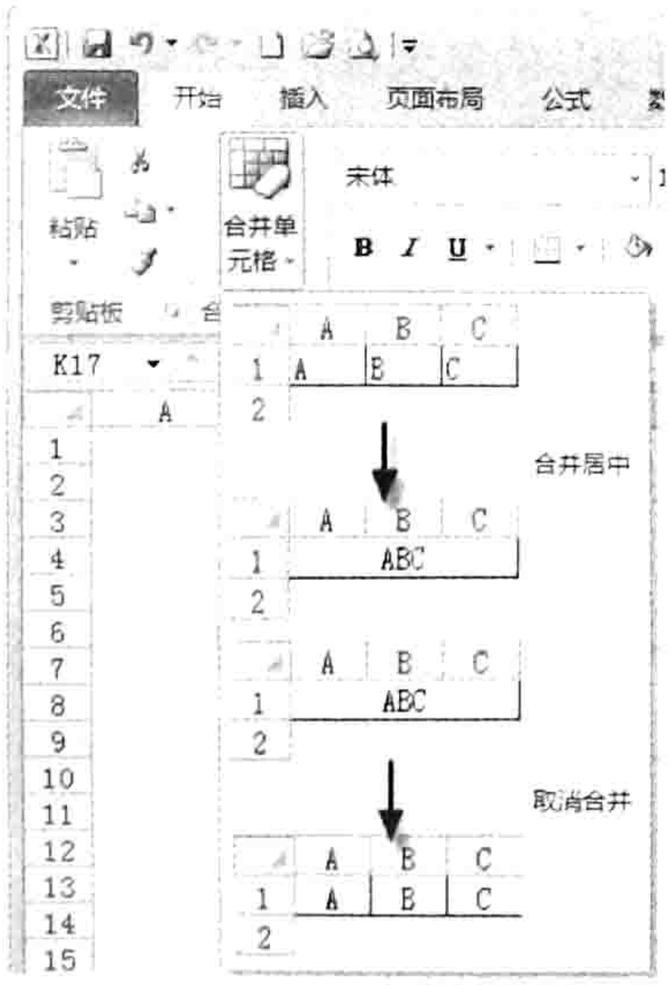


图 21.46 可预览功能的图片库菜单

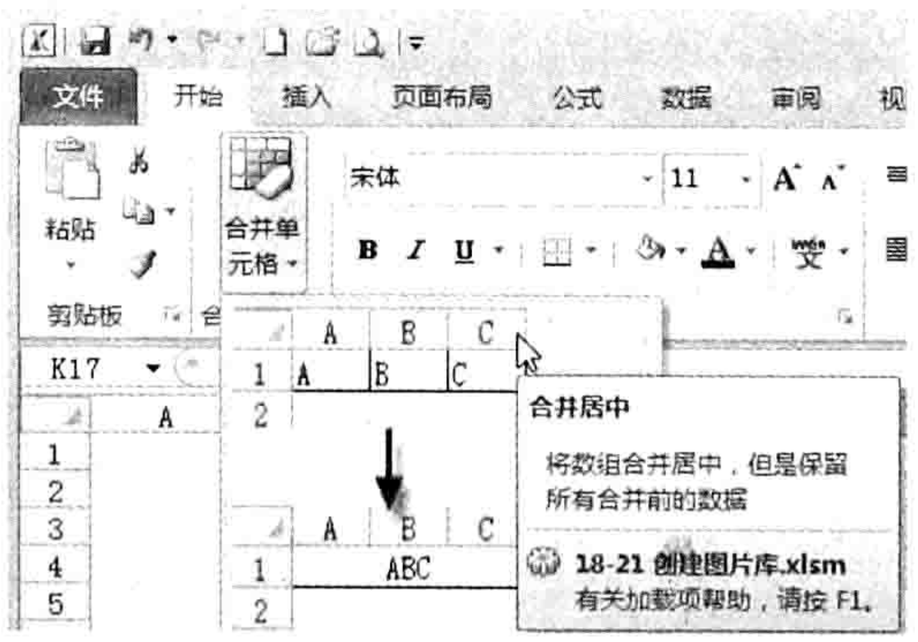


图 21.47 指向库时显示功能预览与提示

step 10 假设 A1:C1 区域分别有“中国”、“湖南”、“长沙”三个字符串，选择 A1:C1 区域后单击“开始”→“合并单元格”→“合并居中”命令，程序将弹出如图 21.48 所示的对话框。

step 11 在“确定分隔符”对话框中可以随意定义分隔符，也可用默认的“-”作为分隔符。假设选择“-”作为分隔符，单击“确定”按钮后 A1:C1 区域将合并为如图 21.49 所示的效果。

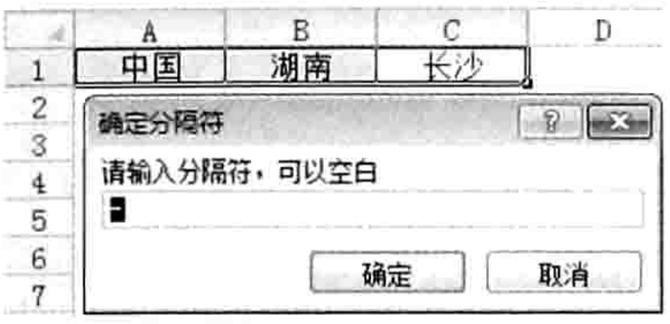


图 21.48　指定分隔符

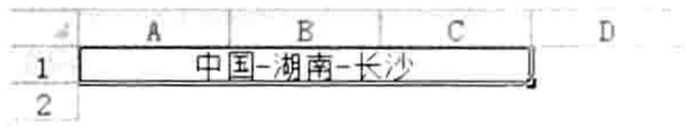


图 21.49　合并效果

step 12 选择合并后的 A1 单元格，再依次单击“开始”→“合并单元格”→“取消合并”命令，在弹出的“确定分隔符”对话框中保持默认的“-”作为分隔符，然后单击“确定”按钮，合并后的 A1:C1 区域转换为合并前的状态，即 A1、B1 和 C1 三个单元格分别存放“中国”、“湖南”、“长沙”三个字符串。

案例补充：

（1）库是 Excel 2007 开始应用在功能区中的一项新的技术，通过它可以实现需要执行的功能的效果预览，让用户在执行过程之前就能看到最终效果，从一定程度上实现了菜单的智能化。

（2）如果修改代码，还可以实现让库显示多行多列，并且任意调整每行每列的显示高度与宽度，也可以随意指定图片的路径。

（3）将硬盘中的图片绑定到库时，不能直接将路径赋值给参数，而是需要使用 LoadPicture 将图片路径转换成图片对象，否则库无法识别。

本例文件参见光盘：..\第二十一章\21-21 创建图片库.xlsm

21.3.9　通过复选框控制错误标识的显示状态

案例要求：在“视图”选项卡中创建一个名为“隐藏错误标识”的复选框，当对复选框打钩时，隐藏工作表中所有错误标识；如果取消打钩则显示所有错误标识。

知识要点：onChange

操作步骤：

step 1 新建一个空白工作簿，然后保存为 xlsm 格式的文件。

step 2 打开 CustomUIEditor 软件，并从软件中打开刚才所保存的工作簿。

step 3 单击菜单中的“Insert”→“Office 2007 Custom UI Part”命令从而插入一个 customUI.xml 文件，然后在代码窗口中录入以下代码，用于在“视图”选项卡创建新组及标签、复选框：

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui">
  <ribbon startFromScratch="false">
    <tabs>
      <tab idMso="TabView">
        <group id="Group1" visible="true" label="视图工具" insertBeforeMso=
"GroupViewShowHide">
          <labelControl id="label1" label="单击时切换"/>
          <checkBox id="check1" label="隐藏错误标识" visible="true" enabled=
"true" onAction="ErrorConversion" screentip="错误值" supertip="切换错误标识的显
示状态" keytip="C"/>
        </group>
      </tab>
    </tabs>
  </ribbon>
```

```
</customUI>
```

step 4 单击“Generate Callbacks”按钮产生 ErrorConversion 回调过程所对应的过程外壳，将它复制出来，然后保存并关闭 CustomUIEditor 软件。

step 5 打开前面创建的工作簿文件，使用<Alt+F11>组合键打开 VBE 窗口。

step 6 单击“插入”→“模块”命令，然后将刚才所复制的代码粘贴到模块中。

step 7 对过程添加代码，使其能正常工作，最终代码如下：

```
'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
Sub ErrorConversion(control As IRibbonControl, pressed As Boolean)
  On Error Resume Next
  Dim rng As Range, cell As Range
    Set rng = Cells.SpecialCells(xlCellTypeFormulas, 23)
    If rng Is Nothing Then Exit Sub
    For Each cell In rng
       cell.Errors.Item(1).Ignore = pressed
    Next
End Sub
```

以上代码表示遍历所有公式结果为错误值的单元格，并根据复选框的状态设置错误标识的状态，如果复选框为打钩状态，那么隐藏活动工作表中所有错误标识。

step 8 保存并重启工作簿，在“视图”选项卡中将看到如图 21.50 所示的控件外观。

step 9 在任意单元格中输入公式“=0/0”，由于公式使用 0 作除数，所以公式的计算结果为错误值，在单元格的左上角将出现绿色小三角标识。

step 10 单击“视图”→“隐藏错误标识”复选框，此时复选框将呈选中状态，单元格中的错误标识也同时被隐藏起来，效果如图 21.51 所示。

图 21.50 复选框外观

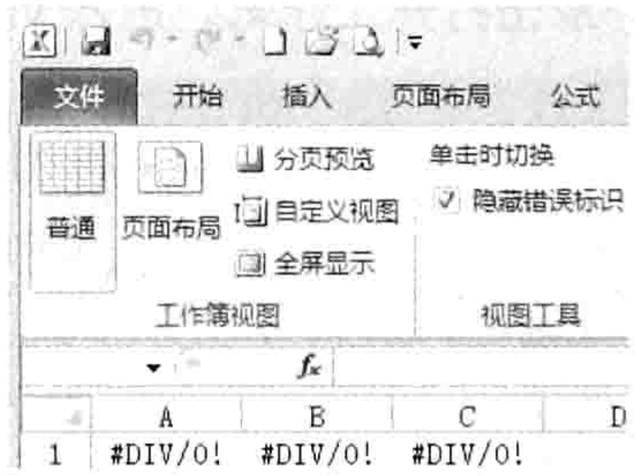


图 21.51 一键隐藏错误标识

案例补充：

（1）cell.Errors.Item(1)代表单元格第一项错误检查选项，Ignore 属性表示忽略，对此属性赋值为 True 可忽略错误标识，赋值为 False 可显示错误标识。而复选框的状态刚好也是 True 和 false，所以在 Sub 过程中直接将代表复选框状态的参数 pressed 赋值给 Ignore 属性。

（2）复选框与切换按钮各有所长，不过总体而言其功能相近，读者可以根据需求或者个人习惯选择使用何种控件。

CD-ROM 本例文件参见光盘：..\第二十一章\21-22 通过复选框控制错误标识的显示状态.xlsm

21.3.10 在功能区中创建工作表目录

案例要求：在“页面布局”选项卡中创建工作表目录。即把所有工作表名称罗列在下拉列表控

件中，单击列表中的工作表名称时可以打开对应的工作表。

知识要点：getItemCount、getItemLabel

操作步骤：

step 1 新建一个空白工作簿，然后保存为 xlsm 格式的文件。

step 2 打开 CustomUIEditor 软件，并从软件中打开刚才所保存的工作簿。

step 3 单击菜单中的“Insert”→“Office 2007 Custom UI Part”命令从而插入一个 customUI.xml 文件，然后在代码窗口中录入以下代码，用于在“页面设置”选项卡创建新组及下拉列表控件：

```
<customUI onLoad="ribbonLoaded" xmlns="http://schemas.microsoft.com/office/
2006/01/customui">
  <ribbon startFromScratch="false">
    <tabs>
      <tab idMso="TabPageLayoutExcel">
        <group id="List" label="工作表目录" insertBeforeMso="GroupPageSetup">
          <dropDown  id="Sheets"  label=" 单 击 切 换 "  getItemCount="ItemCount"
getItemLabel= "ListItem" onAction="Action" />
        </group>
      </tab>
    </tabs>
  </ribbon>
</customUI>
```

以上代码表示在“页面设置”选项卡中创建一个名为“工作表目录”的组，组中显示一个下拉框，下拉框的子项和子项数量由两个回调函数 getItemCount 和 getItemLabel 来决定。

step 4 保存代码，关闭 CustomUIEditor 软件。

step 5 双击打开工作簿，进入 VBE 界面后插入一个模块后，然后在模块中录入以下代码：

```
Dim rib As IRibbonUI, xlApplication As MyEvent
Sub auto_Open()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Set xlApplication = New MyEvent
  Set xlApplication.xlApp = Application
End Sub
Sub ribbonLoaded(ribbon As IRibbonUI)
  Set rib = ribbon
End Sub
Sub Refresh ()
  On Error Resume Next
  rib.Invalidate
End Sub
Sub ItemCount(control As IRibbonControl, ByRef returnedVal)
  returnedVal = Worksheets.Count
End Sub
Sub ListItem(control As IRibbonControl, index As Integer, ByRef returnedVal)
  returnedVal = Worksheets(index + 1).Name
End Sub
Sub Action(control As IRibbonControl, ID As String, index As Integer)
  On Error Resume Next
  Worksheets(index + 1).Select
End Sub
```

为了让目录随工作表增减而自动更新，需要创建一个类模块，从而通过应用程序级别的事件更新工作表目录。

step 6 单击菜单中的“插入”→“类模块”命令，然后在属性对话框中将它重命名为“MyEvent”。

step 7 在类模块中录入以下代码：

```
'①代码存放位置：类模块中②随书光盘中有每一句代码的含义注释
Public WithEvents xlApp As Application
Private Sub xlApp_NewWorkbook(ByVal Wb As Workbook)
  Refresh
End Sub
Private Sub Workbook_SheetActivate(ByVal Sh As Object)
  Refresh
End Sub
Private Sub xlApp_SheetActivate(ByVal Sh As Object)
  Refresh
End Sub
Private Sub xlApp_WindowActivate(ByVal Wb As Workbook, ByVal Wn As Window)
  Refresh
End Sub
Private Sub xlApp_WorkbookNewChart(ByVal Wb As Workbook, ByVal Ch As Chart)
  Refresh
End Sub
```

step 8 保存并且关闭工作簿，然后重启工作簿，在“页面布局”选项卡中的下拉列表控件可以看到如图21.52所示的工作表目录。

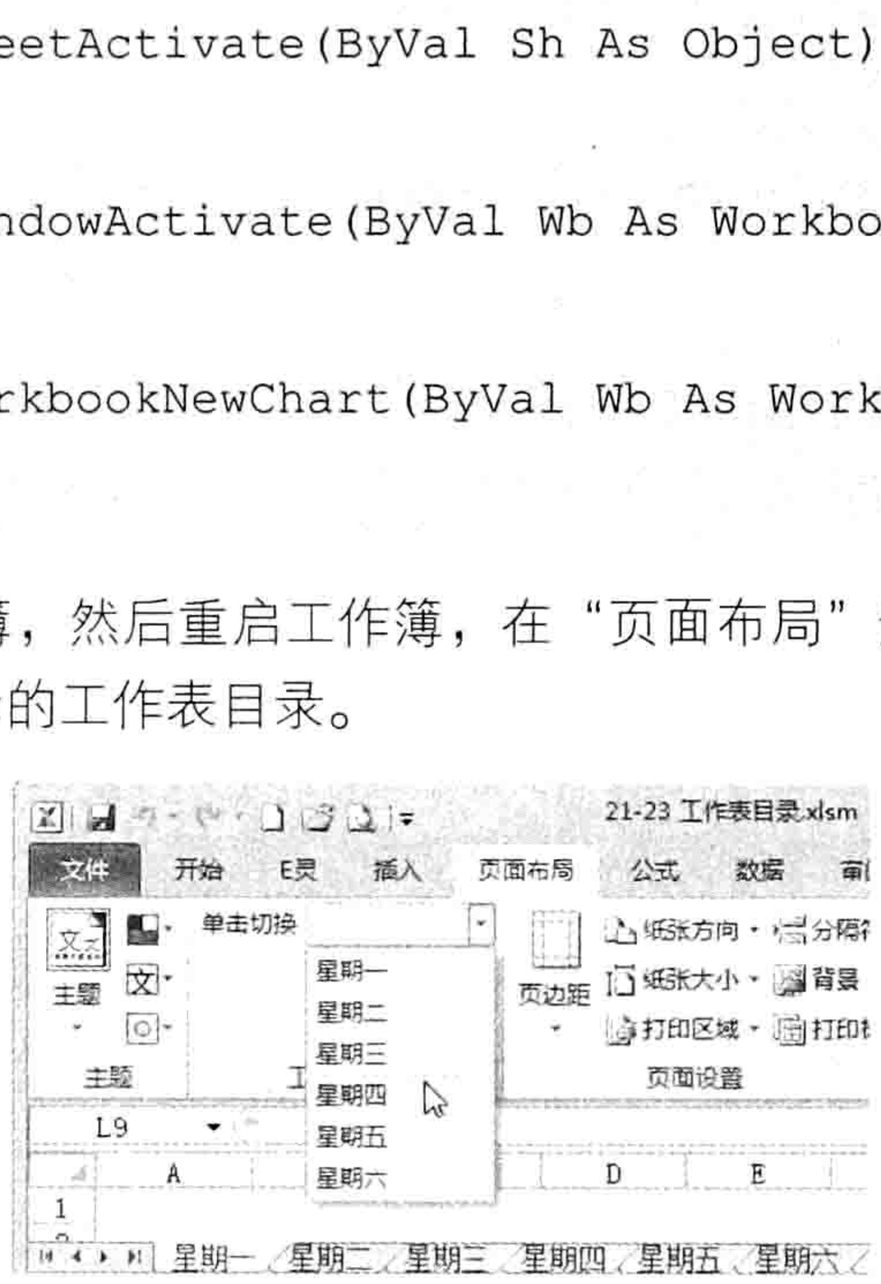


图21.52 工作表目录

step 9 按<Shift+F11>组合键新建一个工作表，然后单击工作表目录列表，可以发现该目录已经自动更新，将新工作表的名称也显示在下拉列表中，单击可以进入该工作表。

CD-ROM 本例文件参见光盘：..\第二十一章\21-23 工作表目录.xlsm

功能区的应用和工作表菜单一样，同样是调用一个Sub过程，重点在于Sub过程本身，菜单是次要的，所以读者需要在Sub过程方面多下工夫。

21.4 使用模板

功能区设计比传统的菜单与工具栏更强大，同时也更复杂，在编写代码时会有不小的困难，所以最好的办法是设计一个或者多个模板来简化工作。

本节详述模板的重要性和设计方法。

21.4.1 模板的重要性

模板就是预先做好的样本，可以重复使用，提供参考作用，简化工作量。

功能区的设计是不可以录制的，所以对用户的要求比较高，必须懂得各种控件的语法，而事实上 VBA 用户皆非专业程序员，没有精力也没有必要去记忆这许多语法表。在此前提下，模板的存在就显得格外重要。

功能区模板是指预先做好的包含各种功能区组件的 xml 文件，后续设计功能区时直接调用此模板中的代码并稍加修改即可。

事实上，CustomUIEditor 软件就自带了 5 个模板。

21.4.2 模板的使用方法

CustomUIEditor 软件自带了 5 个模板，分别保存在以下路径：

◆ 64 位系统

C:\Program Files (x86)\CustomUIEditor\Samples

◆ 32 位系统

C:\Program Files\CustomUIEditor\Samples

读者可以根据自己的实际情况打开文件路径，查看软件自带的模板。

在以上路径下的 5 个文件都是 xml 格式的，用“记事本”程序打开即可看到其中的源代码。

模板文件默认是采用 Excel 2010 格式的，即包含“<customUI>”的根元素中使用了“2009/07”，所以代码不能在 Excel 2007 中正常使用，读者将其修改为“2006/01”即可通用。

当然，以上仅说明模板文件的保存路径，实际调用模板时单击菜单中的“Insert”→“Sample XML”下的文件名称即可，不用理会文件存放在何处。图 21.53 显示了调用模板的菜单。

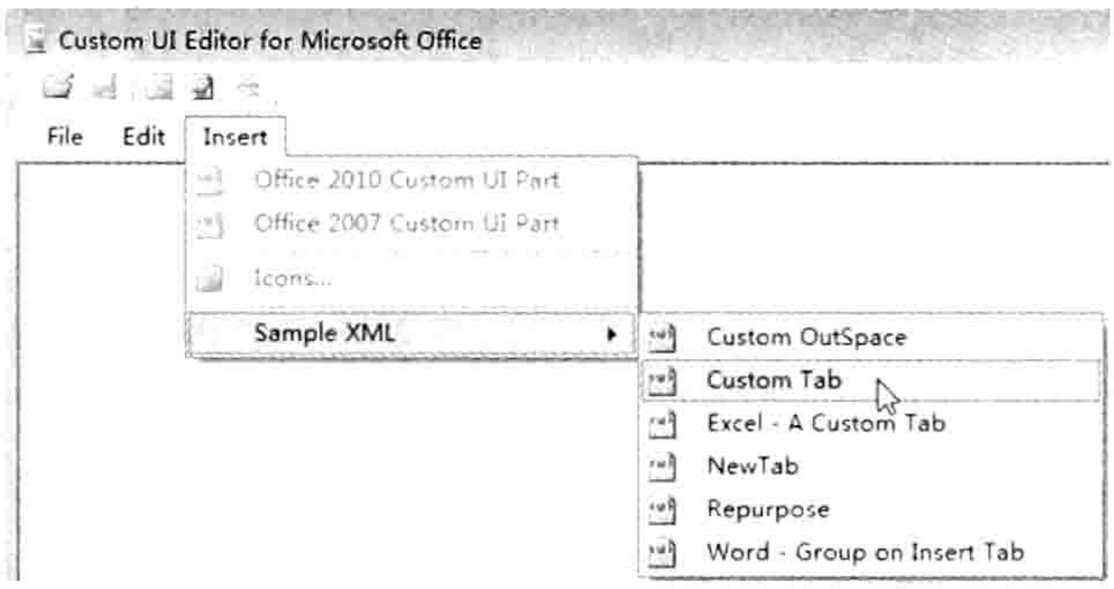


图 21.53 调用模板文件中的代码

21.5 制作两个模板

制作模板可以是全功能型的模板，也可以是有针对性的单一型模板。

全功能型的模板可以将常用的功能区控件都放进去，使用时调出此代码后删除不需要的部分即可。其优势是一个模板可解决绝大部分问题。

单一型模板是根据需求制作多个模板，每一个模板应对一种需求。例如弹出式菜单类、复选框类、对话框启动器类、切换按钮类、图片库类……其优势是调用时速度快，特别适合无法读懂代码的新手。

接下来展示制作两个模板的过程，从而加深读者对模板的理解。

1. 全功能型模板

模板要求：模板需要包含组、标签、命令按钮、切换按钮、弹出式菜单、复选框、对话框启动

器、下拉列表和编辑框等常用控件。

操作步骤：

step 1 从 Windows 的“开始”菜单中打开“记事本”软件，从而创建一个空白的 txt 文件.

step 2 在文件中录入以下代码，代码用于创建含组、标签、命令按钮、切换按钮、弹出式菜单、复选框、对话框启动器、下拉列表和编辑框等常用控件：

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui">
  <ribbon>
    <tabs>
      <tab id="新选项卡" label="新选项卡" insertAfterMso="TabHome">
        <group id="group1" label="One">
          <labelControl id="label1" label="标签"/>
          <button id="customButton1" label="命令按钮 1" screentip="提示"
supertip="详细提示" onAction="Macro1" imageMso="Club" />
          <toggleButton id="toggleButton1" label="切换按钮" visible="true"
enabled="true" onAction="zero" imageMso="AnimationAudio" size="normal"
screentip="零值切换" supertip="按下时显示零值，弹起时不显示零值" keytip="L"/>
          <separator id="分隔条 1" />
          <menu id="One" label="弹出式菜单" screentip="Excelbbx" supertip=
"http://excelbbx.net" size="large" imageMso="AppointmentColorDialog" >
        <button id="customButton2" label="命令按钮 2" screentip="提示"
supertip="详细提示" onAction="Macro3" imageMso="AccessListCustom" />
        <menuSeparator id="菜单分隔条" />
          <button id="customButton3" label="命令按钮 3" screentip="提示"
supertip="详细提示" onAction="Macro4" imageMso="AddressBook" />
        </menu>
        <labelControl id="label2" label="单击时切换"/>
          <checkBox id="checkBox1" label="复选框 1" onAction="Macro5"/>
          <checkBox id="checkBox2" label="复选框 2" onAction="Macro6"/>
          <separator id="分隔条 2" />
        <button id="customButton4" label="命令按钮 4" screentip="提示" supertip="
详细提示" onAction="Macro7" size="large" imageMso="AccessListEvents" />
        <dialogBoxLauncher><button id="dialogBox1" label="对话框启动器"
screentip="提示" supertip="详细提示" onAction="Macro8" /></dialogBoxLauncher>
        </group>
        <group id="group2" label="Two" >
        <dropDown id="Style" showLabel="true" label="匹配方式" onAction=
"dropDownChange" >
          <item id="xlWhole" label="xlWhole" imageMso="WatchWindow" />
          <item id="xlPart" label="xlPart" imageMso="ZoomPrintPreviewExcel" />
        </dropDown>
        <editBox id="FindTxt" label="查找内容" imageMso="ZoomPrintPreviewExcel"
sizeString="999999999999" maxLength="30" visible="true" showLabel="true"
onChange="editBoxChange" keytip="R" />
        </group>
          </tab>
    </tabs>
  </ribbon>
</customUI>
```

以上代码中分别采用“标签”、“分隔条 1”、“命令按钮 1”、“命令按钮 2”等对控件的 label 参数赋值，从而便于识别当前代码所创建的控件的名称。

step 3 使用<Ctrl+S>组合键保存文件，在“另存为”对话框中将文件名称设置为“多功能样本.xml”，同时从“编码”列表中选择“Unicode”，如图 21.54 所示。

step 4 选择保存文件的路径：64 位系统——C:\Program Files (x86)\CustomUIEditor\Samples。
32 位系统——C:\Program Files\CustomUIEditor\Samples。
然后单击“保存”按钮。

step 5 打开 CustomUIEditor 软件，在菜单中“Insert”→“Sample XML”下将看到新的模板名称“多功能样本”，单击模板名称可以将代码插入到当前窗口中。

在实际使用时，可以在导入模板中的代码后根据需要删除多余的控件代码，并对剩下的代码稍做修改即可。如图 21.55 所示是本例模板文件的最终效果预览。

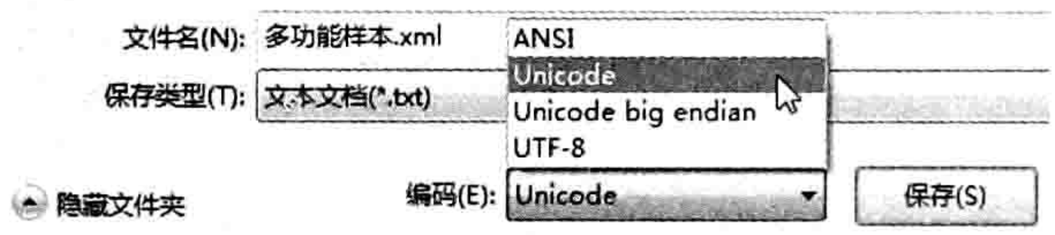


图 21.54 保存文件时选择 Unicode 格式

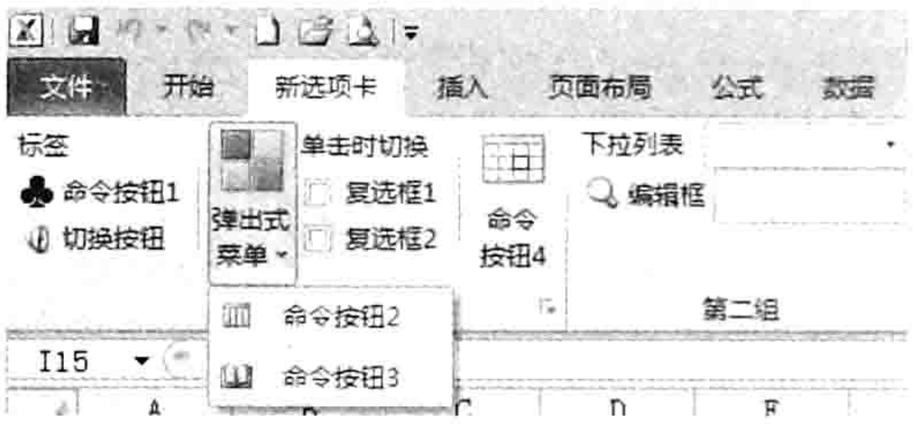


图 21.55 样本文件包含的功能区控件一览

CD-ROM 本例文件参见光盘：..\第二十一章\多功能样本.xml

2. 单一型模板

模板要求：模板主要展示组的编写方式，例如在“开始”选项卡中的组，包括“插入到中间”和“放在末端”，新选项卡中的组和在“视图”选项卡中的组。通过此模板可以对以后创建组提供便利。

操作步骤：

step 1 从 Windows 的“开始”菜单中打开“记事本”软件，从而创建一个空白的 txt 文件。

step 2 在文件中录入以下代码，代码用于创建位于不同位置的 4 个组：

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui">
  <ribbon startFromScratch="false">
    <tabs>
      <!-- 以下代码表示在开始选项卡中创建两个组，首尾各一个-->
     <tab idMso="TabHome" >
      <group id="Group1" label="第一组" insertBeforeMso="GroupFont">
      </group>
      <group id="Group2" label="第二组" >
      </group>
    </tab>
      <!-- 以下代码表示在新建选项卡中创建一个组-->
    <tab id="CustomTab" label="新选项卡" insertAfterMso="TabHome">
      <group id="Group3" label="第三组">
      </group>
    </tab>
      <!-- 以下代码表示在视图选项卡的最前面创建一个组-->
```

```
      <tab idMso="TabView" >
        <group id="Group4" label="第四组" insertBeforeMso="GroupWorkbookViews">
        </group>
      </tab>
    </tabs>
  </ribbon>
</customUI>
```

代码中以“<!--”开头、“-->”结尾的注释，执行代码时会自动跳过，不会产生语法问题。使用注释后更利于后续的编写工作提速与代码维护。

step 3 使用<Ctrl+S>组合键保存文件，在“另存为”对话框中将文件名称设置为“组模板.xml”，同时从“编码”列表中选择“Unicode”。

step 4 选择保存文件的路径为 CustomUIEditor 软件的模板目录 Samples，然后单击“保存”按钮。

step 5 打开 CustomUIEditor 软件，在菜单“Insert”→“Sample XML”下将看到新的模板名称“组模板”，单击模板名称可以将代码插入到当前窗口中。

如图 21.56 至图 21.59 所示的 4 个图片是以上模板文件的最终效果预览。

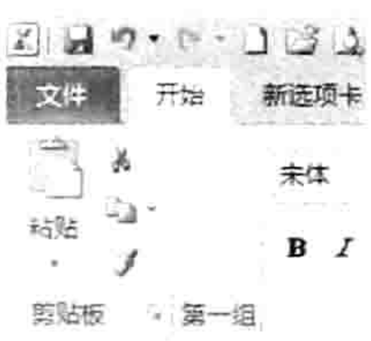


图 21.56 第一组

图 21.57 第二组

图 21.58 第三组

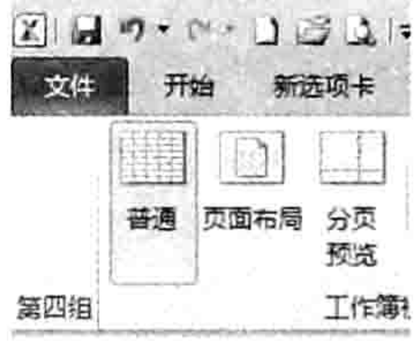


图 21.59 第四组

本例模板中的代码仅针对组 group 提供样本，读者可以借助此思路再分别对切换按钮、下拉列表等控件制作模板。

本例文件参见光盘：..\第二十一章\组模板.xml

第 22 章　开发通用插件

Excel 插件包括加载宏和加载项，类似于游戏中的外挂插件。使用插件可以大大强化 Excel 的功能，简化操作步骤，提升工作效率。

本章将讨论加载宏与加载项的特点、分类，并通过两个案例展示 xlam 格式的加载宏开发思路与技巧，对于加载项的设计过程将放在本书的第 23 章中进行介绍。

22.1　关于加载宏

加载宏是从 Excel 97 版本开始推广的一类工作簿格式，扩展名为 xla，到 Excel 2007 时新增了一种功能更强大的加载宏，其扩展名为 xlam。

加载宏工作簿的存在价值是设计插件，本章所介绍的插件都采用 xlam 为扩展名。

22.1.1　加载宏的特点

加载宏也是一个工作簿，但它是一个特殊的工作簿。加载宏与其他工作簿的区别是它拥有普通工作簿所没有的一些特性，主要体现在以下 6 个方面。

- 加载宏的 IsAddin 属性为 True，代表它是加载宏工作簿。
- 使用 Workbooks.Count 统计工作簿数量时会自动忽略加载宏工作簿。
- 加载宏的窗口是隐藏的，并且无法通过功能区中任何功能按钮操作加载宏工作簿。
- 安装加载宏后，在任意工作簿中都可以调用存放在加载宏中的代码，打开任意工作簿时都会自动打开加载宏工作簿。
- 无法通过按<Alt+F8>组合键的方式调用加载宏中的任何代码，但可以通过菜单或者快捷键调用。
- 加载宏的工作簿结构无法手工修改，例如增减工作表、拆分窗口、设置工作表背景、删除工作表中的数据等，此特性可保护加载宏工作簿的数据及确保程序的正常运行。

将带有宏的普通工作簿另存为加载宏有以下 3 个好处。

- 加载宏的所有程序可以应用于当前打开的所有工作簿，而不仅限于代码所在工作簿。
- 加载宏工作簿不管存放在任何文件夹下都可以通过简单地安装就能永远自动运行下去，直到手工删除文件或者卸载加载宏。
- 加载宏中的代码不受宏的安全性限制，即禁用宏也一样可以调用加载宏中的代码。

将普通工作簿转换成加载宏的方法是另存文件，在“保存类型”对话框中选择扩展名为“xla”或者“xlam”格式即可，操作界面如图 22.1 所示。

22.1.2　为什么使用加载宏

加载宏就是一个插件，用于弥补 Excel 自身功能的不足。

通常在以下情况下需要使用插件来完成工作。

图 22.1　保存文件时选择加载宏格式

（1）使用 Excel 自身的功能可以达成需求，但操作步骤过多，效率偏低。

例如根据工资明细表生成工资条，虽然插入空行、复制标题行、粘贴标题行，然后循环以上步操作可以完成工作，但是在效率上与插件相比有天壤之别，利用加载宏可以瞬间完成，而手工操作可能不少于 10 分钟。

（2）Excel 自身的函数可以完成，但需要诸多函数嵌套，教会新手使用的成本偏高。

例如要利用公式计算“诺基亚 920 购入一台 1980.5 元、三星 Note 2 手机购入两台 4500 元”中的金额合计可以使用以下公式完成：

```
=SUMPRODUCT(--TEXT(MID(TEXT(MID(SUBSTITUTE("★"&A1,"元",REPT(" ",15)),ROW($1:
$999),15),),2,15),"0.00;-0;0;!0"))
```

显然以上公式过于复杂，要向新手解释公式的功能和运算过程，以及教会对方根据实际需求修改以上公式可能需要 10 天以上的时间，而采用本书 16.5.4 节的自定义函数完成相同统计仅需使用以下简短的公式，既使是函数的初学者也仅需一分钟足以学会使用：

```
=合计(A1,"元")
```

（3）Excel 自身的功能无法完成的一些工作，需要借助 VBA 开发的插件来实现。

查询股票或者天气信息、按字体颜色汇总、分页小计、底端标题等都是 Excel 本身不具备的功能，借助加载宏却可以实现。

在扩展名为 xlsm 的工作簿中添加了生成功能区选项卡的 xml 代码后，只有打开该工作簿才能看到新建的功能区选项卡或者菜单，而将工作簿另存为扩展名为 xlam 的加载宏文件，然后加载这个工作簿，那么打开任意工作簿都可以看到新建的功能区选项卡和菜单，可以随意调用其功能。如果要求工作簿中的代码应用于所有工作簿，应该将工作簿保存为加载宏。

22.1.3　加载宏管理器

Excel 提供了一个加载宏管理器，在其中显示了当前的所有加载宏，并通过是否打钩来标示每个加载宏的可用状态。

打开加载宏管理器的快捷键是<Alt+T+I>，按下<Alt>键后松开，再按<T>键，松开后再按<I>键。

在默认状态下，在加载宏管理器中有若干个内置的加载宏，例如“分析工具库”、“欧元向导”、“规划求解加载项”等，其中打钩的加载宏表示可用状态。

用户可以单击“加载宏”对话框的“浏览”按钮添加新的加载宏文件，图 22.2 中“Excel 百宝

箱”、“Excel 代码百宝箱”、“鼠标移动着色”等工具属于自定义加载宏，“规划求解加载项”和“标签打印向导”属于 Excel 的内部加载宏，它们都处于可用状态。

22.1.4 加载内置的加载宏

Excel 自带有若干个内置的加载宏，但默认状态下它只是罗列在加载宏管理器中，却并没有发挥功效。如果需要运行这些内置的加载宏，应使用<Alt+t+i>组合键打开“加载宏”对话框，然后对需要使用的加载宏打钩，当单击“确定”按钮后，如果磁盘中有 Office 的安装文件并且从未移动过位置，那么 Excel 会自动从该位置复制出相应的加载宏文件，否则可能会提示用户选择安装文件路径，此时只需要提供 Office 的安装文件路径即可完成安装。

图 22.3 是安装“规则求解加载项”之后产生的新菜单，在加载前是没有此菜单的。

图 22.2 “加载宏”对话框

图 22.3 规划求解加载宏

如果用户使用的是精简版 Office，那么“加载宏”对话框中可能一片空白。

22.1.5 安装与卸载自定义加载宏

用户开发的加载宏（xla 和 xlam 格式的文件）需要安装才能长期使用，双击打开加载宏只能使用一次，下次要调用其功能时又需要手动打开工作簿。

安装自定义加载宏通常有两种方法：“加载宏”对话框和自启动文件夹。

1. 加载宏管理器

假设在“D:\我的开发工具”文件夹中有一个名为“快速查找.xlam”的加载宏，通过“加载宏”对话框安装这个加载宏的步骤如下。

step 1 按<Alt+T+I>组合键打开“加载宏”对话框。

step 2 单击“浏览”按钮打开“浏览”对话框，进入“D:\我的开发工具”文件夹中选中“快速查找.xlam”文件，然后单击“确定”按钮返回“加载宏”对话框，在对话框中可以看到“快速查找”加载宏，效果如图 22.4 所示。

step 3 单击“确定”按钮回到工作表界面，在“开始”选项卡中可以看到“快速查找”加载宏产生的菜单，效果如图 22.5 所示。

由于安装加载宏后每次打开 Excel 都会自动打开加载宏工作簿，因此任何时候都可以在“开始”选项卡中看到“快速查找”加载宏所产生的功能区菜单。

2. 自启动文件夹

安装加载宏的唯一目的是让文件自动启动，因此将文件保存在自启动文件夹中也可以实现安装

加载宏一样的功能。

图 22.4　“加载宏”对话框中的“快速查找”

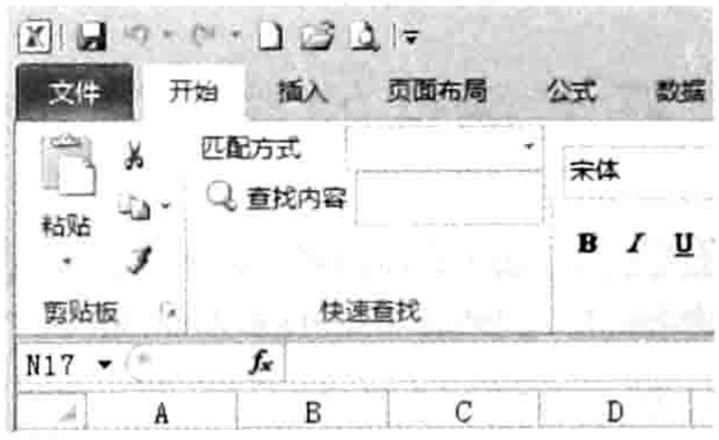


图 22.5　加载宏产生的功能区组件

不同版本的 Office 的自启动路径是不一样的，以下罗列 4 个版本的 Office 的自启动文件夹路径：

C:\Program Files\Microsoft Office\Office11\XLSTART——Office 2003 版；

C:\Program Files\Microsoft Office\Office12\XLSTART——Office 2007 版；

C:\Program Files\Microsoft Office\Office14\XLSTART——Office 2010 版；

C:\Program Files\Microsoft Office\Office15\XLSTART——Office 2013 版。

如果读者和笔者一样使用 64 位的操作系统和 32 位的 Office，那么应将“Program Files”修改为“Program Files (x86)”。

当然也可以通过代码直接打开自启动文件，代码如下：

```
Sub 打开自启动文件夹()'代码存放位置：模块中
  Shell "explorer.exe " & Application.Path & "\XLSTART", vbMaximizedFocus
End Sub
```

将加载宏文件保存在本机的自启动文件夹中可以在打开 Excel 时自动打开加载宏文件，与通过“加载宏”对话框安装加载宏的功能一致，不过卸载加载宏时会麻烦一些，只能关闭 Excel 后再进入自启动路径中删除文件，而通过“加载宏”对话框加载的加载宏仅需进入“加载宏”对话框，然后去掉勾选“快速查找”选项即可。

本例文件参见光盘：..\第二十二章\快速查找.xlam

22.2　关于加载项

加载项是 Excel 的外置控件，用于扩展 Excel 的功能。加载项由 VB、C、VSTO、Delphi 或者.net 等语言编写，其扩展名可以是 DLL 也可以 Ocx，最常见的扩展名是 DLL。

DLL 格式的加载项相对于 xla 或者 xlam 格式的加载宏更安全，其他人无法查看到代码。

22.2.1　加载项的分类

加载项有自动化加载项和 COM 加载项之分，最常用的开发软件是 VB。利用 VB 6.0 企业版可以开发出与 Excel 完美结合的自动化加载项，以及 COM 加载项。

开发加载项通常称之为封装，封装自定义函数可以用自动化加载项，而封装 Sub 过程则宜用 COM 加载项。本书的第 23 章会讲述如何封装自定义函数和 SUB 过程。

22.2.2　加载项的开发方式

加载项包含 DLL 和 Ocx 格式，本书仅教授 DLL 格式的加载项的封装办法。

DLL 加载项通常用 VB、VC、C++、Delphi 等软件开发。而其中最理想的莫过于 VB 6.0 企业

版，它与 Office 各组件的联系最紧密，VB 6.0 开发的 DLL 插件在速度上也占有优势。

VB 6.0 企业版的下载地址为：http://t.cn/8FknyKa。

22.3 开发插件的准备工作

加载宏包含 xla 和 xlam 两种格式，后者是 Excel 2007 及以上版本专用的加载宏格式。

本节将讲述开发一个加载宏的准备工作，包括如何选择加载宏格式、如何引用加载宏的数据，以及设计加载宏的附加工作。

22.3.1 加载宏的格式

利用 Excel VBA 可以直接生成的加载宏文件包括 xla 和 xlam 两种格式，它们的区别如下。

xla 格式是早期版本的加载宏格式，它的工作表仅包含 65 536 行×256 列，只支持传统菜单，不支持功能区。但是它的优点是在 Excel 2003、Excel 2007、Excel 2010 和 Excel 2013 版本中都可以正常工作。

xlam 是从 Excel 2007 开始投入使用的一种新格式，它的工作表支持 1 048 576 行×16 384 列，支持传统菜单和功能区，也支持 SUMIFS、IFERROR 等新函数。

在实际工作中，如果确定不再使用 2003 版 Office，那么应采用支持更多功能的 xlam 加载宏；如果为了确保兼容性，让加载宏文件可在多个版本的 Office 中运行，那么应使用扩展名为 xla 的加载宏。

22.3.2 引用加载宏的数据

加载宏也是一个工作簿，但它是处于隐藏状态的工作簿，因此要引用加载宏的工作表中的数据和引用普通工作簿的工作表中的数据有所不同。

加载宏处于隐藏状态，因此它不可能成为活动工作簿，也没有活动工作表，要引用其中的数据时必须将工作簿名称、工作表名称和单元格地址书写完整。例如加载宏的名字是“快速查找.xlam”，要引用它的第 1 个工作表中的 A1:A10 区域应采用以下代码：

```
Workbooks("快速查找.xlam").Worksheets(1).Range ("a1:a10")
```

简而言之，引用加载宏工作簿中的单元格时不能忽略工作簿和工作表对象。

22.3.3 设计加载宏的附加工作

加载宏尽管也是一个工作簿，但它本质上属于工具性质，因此在设计加载宏时与设计普通工作表稍有区别。对于加载宏工作簿，编好代码后还需要注意以下 3 点。

1. 工程加密

将工程加密的目的是防止意外破坏代码，而不是防止他人查看。

在 22.4 节中会讲述开发日历控件时会讲述加密代码的步骤。

2. 提供菜单

加载宏中的所有 Sub 过程都不能通过<Alt+F8>组合键调用，因此应该为加载宏工作簿设计菜单，方便用户快速调用过程。

3. 提供帮助

加载宏的开发者和终端用户往往并不是同一个人，即开发加载宏后多数情况是给他人使用。为

了降低学习成本，在设计加载宏时应该为加载宏设计帮助，指示加载宏中的每个 Sub 过程或者 Function 过程的具体使用方法。

22.4　开发公/农历日历控件

Excel 自带一个日历控件，可以在单击控件后将选中的日期写入到活动单元格中。

不过 Excel 自带的日历控件缺点太多，不适合应用在工作中。本节介绍更强大、更易用的日历控件的开发过程，同时也演示设计通用加载宏的基本步骤。

22.4.1　确认程序需要具备的功能

类似于建筑行业在筑房前需要绘制图纸、写作文前需要罗列提纲，开发插件前也需要罗列出插件的基本功能，避免开发过程中有遗漏的功能。

本节所开发的日历控件需要具备的功能如下。

（1）可以在窗体中随意选择年、月、日，然后将选中的日期录入在活动单元格中。

（2）日历控件默认显示今天的日期，而且要突出显示。

（3）单击录入日期后，需要自动激活下一个待输入的单元格，不用手工操作。

（4）支持公历和农历。

（5）日历控件要做成加载宏形式，使其功能可以应用于所有工作簿。

22.4.2　定义公历转农历的函数

对于 22.4.1 节中罗列的 5 个需求，首先要完成的是将公历日期转为农历日期。

中国的公历日期转为农历日期没有规律可循，Office 也从来没有提供它们之间的转换工具，通过 VBA 开发自定义函数实现公历日期与农历日期之间的转换也没有任何捷径可走，只能通过其他途径获取每月 1 日的公历日期与农历日期对应的关系，并储存在工作表中作为辅助条件，然后借助函数实现快速转换。

具体的操作步骤如下。

step 1　通过其他农历软件或者万年历查询出从 1900 年 1 月 1 日开始的农历每月初一所对应的公历日期。例如 1900 年 1 月 31 日对应农历正月初一，1900 年 3 月 1 日对应农历二月初一，那么将这些公历日期逐一罗列在 A 列，农历日期逐一罗列在 B 列，直到 2100 年 12 月 31 日为止。

图 22.6 中 B 列的所有信息都可以通过查询得到，市面上很多日历软件都带有农历日期。

	A	B
1	日历	【数据】
2	1900/1/1	己亥年(猪)腊月
3	1900/1/31	庚子年(鼠)正月
4	1900/3/1	庚子年(鼠)二月
5	1900/3/31	庚子年(鼠)三月
6	1900/4/29	庚子年(鼠)四月
7	1900/5/28	庚子年(鼠)五月
8	1900/6/27	庚子年(鼠)六月
9	1900/7/26	庚子年(鼠)七月

Sheet1

图 22.6　每月初一的公农历对照表

step 2　按<Alt+F11>组合键进入 VBE 界面，单击菜单中的“插入”→“模块”命令，然后在模块中录入以下代码：

```
Function 农历(rng) As String'①代码存放位置：模块中②随书光盘中有每一句代码含义注释
```

```
  Dim Arr() As Variant
  Arr = Array("初一", "初二", "初三", "初四", "初五", "初六", "初七", "初八", "初九",
"初十", "十一", "十二", "十三", "十四", "十五", "十六", "十七", "十八", "十九", "二十
", "廿一", "廿二", "廿三", "廿四", "廿五", "廿六", "廿七", "廿八", "廿九", "三十")
  If IsDate(rng) Then rng = DateAdd("Y", 0, rng) * 1
  If rng >= 1 And rng < 73415 Then
    农历 = WorksheetFunction.VLookup(rng, ThisWorkbook.Worksheets(1).Range
("A2: B2488"), 2, True) & Arr(rng - WorksheetFunction.VLookup(rng, ThisWorkbook.
Sheets(1).Range("A2:A2488"), 1, True))
  Else
    农历 = ""
  End If
End Function
```

step 3 返回工作表界面，在工作表中随意输入公历日期，然后通过自定义函数测试公历转农历的准确性。如图 22.7 所示为公历转农历的测试结果。

图 22.7 测试农历函数

22.4.3 设计日期输入器窗体

创建一个用于产生每个月日期的窗体，这属于第二个设计步骤。具体操作步骤如下。

step 1 单击菜单中的“插入”→“用户窗体”命令，将其 Caption 属性修改为“窗体日历”，将名称属性修改为“calendar”。

step 2 在窗体上方插入两个旋转按钮（图标 ），左边的按钮名称属性修改为“调整年”，并且将 Min 和 Max 属性分别修改为 1900 和 2100，表示它的范围在 1900～2100；将右边的按钮名称属性修改为“调整月”，同时将 Min 和 Max 属性分别修改为 0 和 13。

step 3 在两个旋转按钮之间添加两个标签，并且将 Caption 属性设置为空文本。然后对上一个标签的名称属性修改为“Years”，用于显示年和月，再将下方的标签的名称属性修改为“Weekdays”。

step 4 在下方添加一个标签，将 Name 属性修改为“Label1”，将 Caption 属性设置为空文本，将其背景设置为白色，将其形状调整为正方形，并且将 SpecialEffect 属性修改为 fmSpecialEffectRaised，使标签呈突起状态。

step 5 将步骤 4 所设计的标签复制为 42 份，并且保持 6 行 7 列。因为一周包含 7 天、一月中跨越最多的周期数是 6 周。

step 6 在窗体的右上方添加一个图形控件，并且指定一幅图片作为其 Picture 属性值。这一步的作用仅仅是装饰，填补窗体右上角的空白区域。

step 7 在图形控件下方添加一个 Frame 框架控件，在框架中添加两个选项按钮，将第一个按钮的名称属性和 Caption 属性分别修改为“公历日期”和“以公历日期录入”；将第二个按钮的名称属性和 Caption 属性分别修改为“农历日期”和“以农历日期录入”。

step 8　添加一个命令按钮，并且将其名称属性和 Caption 属性都改为“返回当前日期”。最后的效果如图 22.8 所示。

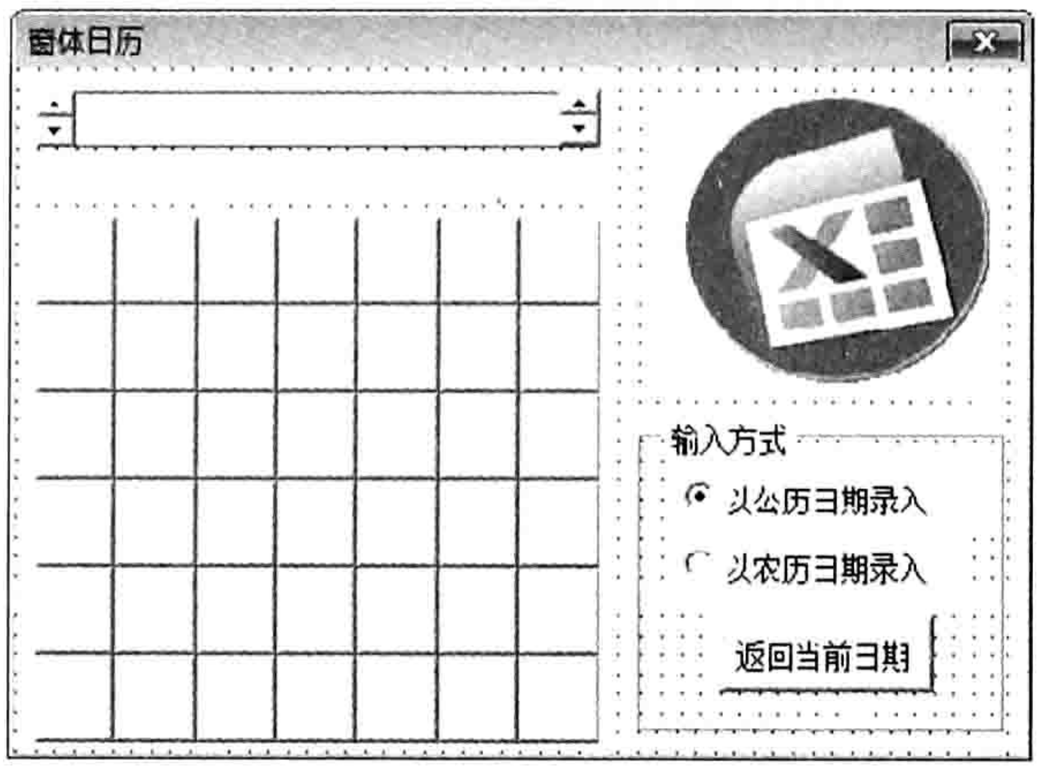


图 22.8　窗体日历的控件布局方式

22.4.4　编写窗体初始化代码

窗体初始化时应该将窗体中并排的 42 个标签根据实际情况调整显示方式，例如超过本月天数的那部分标签需要隐藏起来，剩下的标签则显示为本月每一天的日期。具体设置步骤如下。

step 1　在存放“农历”自定义函数的模块的顶端输入以下代码，用于声明公共变量：

```
Public 月 As Integer, 年 As Integer, 公历 As Boolean, Labels(1 To 42) As New
MyEvent
```

其中数组 Labels 的参数 1～42 对应图 22.8 中背景为白色的 42 个标签控件。

step 2　在模块中继续录入以下过程代码：

```
Sub ShowCalendar()'①代码存放位置：模块中②随书光盘中有每一句代码含义注释
  Dim LabelCount As Long, ctrl As control
  月 = Month(Date)
  年 = Year(Date)
  calendar.Show 0
  For Each ctrl In calendar.Controls
    If Left(ctrl.Name, 5) = "Label" Then
      LabelCount = LabelCount + 1
      Set Labels(LabelCount).LabelGroup = ctrl
    End If
  Next ctrl
End Sub
```

以上过程首先将变量“年”和“月”赋值为本年的年份和本月的月份，然后在显示窗体时将窗体中的 42 个空白标签控件赋值给数组变量 Labels。

step 3　双击窗体进入窗体的事件代码窗口，并在窗口中输入以下代码：

```
'①代码存放位置：窗体代码窗口中②随书光盘中有每一句代码含义注释
Private iChangeSettings As Integer
Private Sub UserForm_Initialize()
  Dim i As Long, 本月初 As Long, 本月末 As Long
  本月初 = DateSerial(年, 月, 1)
```

```
本月末 = DateSerial(年, 月 + 1, 0)
调整年.Value = 年
调整月.Value = 月
On Error Resume Next
If Me.公历日期 Then 公历 = True Else 公历 = False
For i = 1 To 42
  With Me.Controls("Label" & i)
    If Weekday(本月初, 2) - i >= 1 Or Day(本月末) < i - Application.Weekday(本月初, 2) + 1 Then
      .Visible = False
    Else
      .Visible = True
      .Caption = i - Weekday(本月初, 2) + 1
      If i = Day(Now) + Weekday(本月初, 2) - 1 And 年 = Year(Date) And 月 = Month(Date) Then
        .BackColor = vbYellow
      Else
        .BackColor = RGB(255, 255, 255)
      End If
    End If
  End With
Next
Years.Caption = 年& "年-" &月& "月"
Weekdays.Caption = "一二三四五六日"
iChangeSettings = iChangeSettings + 1
On Error GoTo 0
End Sub
```

以上 Initialize 事件过程主要让窗体初始化时完成 4 件事情：

- 让两个旋转按钮分别显示今年的年份与本月的月份，以及让中间的两个标签控件分别显示当前年月与代表星期的标题文字。
- 根据“公历日期”和“农历日期”两个选项按钮的状态决定变量“公历”的值；
- 根据本月的天数，以及本月 1 日是星期几来判断哪些标签需要隐藏，再对不需要隐藏的标签的 Caption 属性赋值为 1 到月末天数的自然数序列。
- 对代表今天的标签用黄色显示出来。

step 4 插入一个类模块，将其重命名为 MyEvent。并在类模块中输入以下代码：

```
'①代码存放位置：类模块中②随书光盘中有每一句代码含义注释
Public WithEvents LabelGroup As MSForms.Label
Private Sub LabelGroup_MouseMove(ByVal Button As Integer, ByVal Shift As Integer,
ByVal X As Single, ByVal Y As Single)
  LabelGroup.ControlTipText = 农历(DateSerial(年, 月, LabelGroup.Caption))
End Sub
```

以上过程表示鼠标在标签上移过时提示农历日期。

如果今天是 2014 年 3 月 8 日，那么执行过程“ShowCalendar”，窗体中的所有控件会初始化为本月相关的状态。图 22.9 中显示了本月的所有日历，其中鼠标指向标签 25，那么提示信息“甲午年（马）二月廿五”，表示 2014 年 3 月 25 日对应的农历日期是甲午年（马）二月廿五。

图 22.9　本月日历

22.4.5　实现输入器与工作表交互

以上步骤仅仅实现在窗体中产生日历，要单击标签向单元格中输入日期还需要完成以下步骤。

step 1　双击进入窗体代码窗口，并输入以下代码：

```
Private Sub 调整月_Change()'①代码存放位置：窗体中②随书光盘中有每一句代码的含义注释
  On Error Resume Next
  If iChangeSettings < 1 Then Exit Sub
  If 调整月.Value > 12 Then 调整月.Value = 1: 年 = 年 + 1
  If 调整月.Value < 1 Then 调整月.Value = 12: 年 = 年 - 1
  If 年< 1901 Then GoTo err
  月 = 调整月.Value
  UserForm_Initialize
  Exit Sub
err:
  MsgBox "不能小于1901年", , "友情提示"
  年 = 1901
  月 = 1
  UserForm_Initialize
End Sub
```

以上代码表示单击旋转按钮的箭头时可以上调或者下调月份，而且在 1～12 范围内。

```
Private Sub 调整年_Change()'①代码存放位置：窗体中②随书光盘中有每一句代码的含义注释
  On Error Resume Next
  If iChangeSettings < 1 Then Exit Sub
  年 = 调整年.Value
  UserForm_Initialize
End Sub
```

以上代码表示单击另一个旋转按钮的箭头时可以上调或者下调年份。

```
Private Sub 返回当前日期_Click()'①代码存放位置：窗体中②随书光盘中有每一句代码的含义注释
  年 = Year(Date)
  月 = Month(Date)
  UserForm_Initialize
End Sub
```

以上代码表示在任意情况下单击按钮可以返回当前月。

```
Private Sub 公历日期_Click()'①代码存放位置：窗体中②随书光盘中有每一句代码的含义注释
  公历 = True
End Sub
Private Sub 农历日期_Click()
  公历 = False
End Sub
```

以上代码表示通过选项按钮改变变量“公历”的值，它决定输入到单元格中的日期是公历还是农历。

step 2 进入类模块，录入以下代码：

```
Private Sub LabelGroup_Click()'①代码存放位置：类模块中②随书光盘中有每一句代码含义注释
  On Error Resume Next
  If 公历 = True Then
    ActiveCell.Value = DateSerial(年, 月, LabelGroup.Caption)
  Else
    ActiveCell.Value = LabelGroup.ControlTipText
  End If
  ActiveCell.Offset(1, 0).Select
End Sub
```

这是窗体中标签控件的单击事件过程。如果变量“公历”的值为 True，那么将变量“年”、“月”和标签中显示的值通过 DateSerial 函数转换成日期序列显示在活动单元格中，否则将该标签的 ControlTipText 属性值显示在单元格中。ControlTipText 属性值即为农历日期。

22.4.6 设计帮助

帮助可以用来描述插件的功能、操作方法和注意事项，甚至版本号、更新信息、作者联系方式等。当开发者与终端用户不是同一人的情况下编写帮助将显得尤为重要。

本例仅仅讲述设计简要帮助的方法，不强调如何细致入微地设计一个非常完善的帮助。本例中会简单地设计一个窗体，在窗体中通过动态文字描述加载宏的功能，以及作者联系方式，具体步骤如下。

step 1 选择菜单中的“插入”→“用户窗体”命令，然后将其名称属性修改为“Help”，将 Caption 属性修改为“关于日历控件”。

step 2 在窗体中插入一个 Web 控件，并且让 web 控件填充整个窗体，其效果如图 22.10 所示。

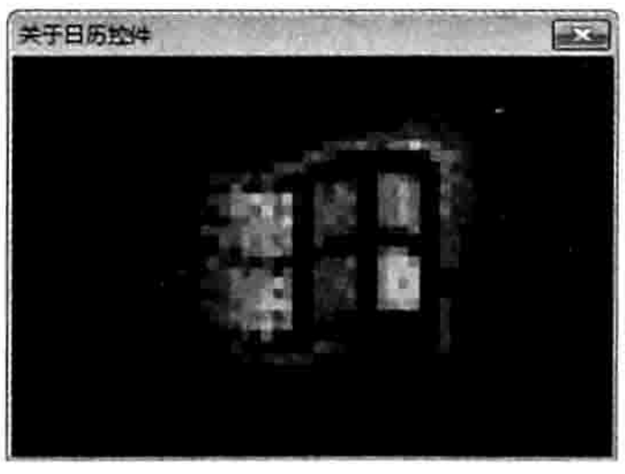


图 22.10 帮助窗体

step 3 双击窗体进入事件代码窗口，删除自动产生的代码，然后录入以下代码：

```
Private Sub UserForm_Initialize()'①代码存放位置：窗体中②随书光盘中有每一句代码的含义注释
  WebBrowser1.Navigate "about:blank"
```

```
WebBrowser1.Document.writeln "<html><body bgcolor=#299BFB style='border:
none;overflow:hidden;margin:0' oncontextmenu='return false'><p><marquee id=help
onmouseover=help.stop() onmouseout=help.start() scrollamount=""2"" bgcolor=
""#299BFB""" & " direction=up>    当工作中需要录入很多日期时，
手工录入是一件低效的事，本工具可以实现单击录入日期，而且多次产生的日期总会统一格式，避免影响后
续的运算。</P>    为了适应国人的习惯，还特意加入了农历日期。可以随意
在公历与农历间切换。</P>    使用本插件时单击功能区中的菜单出现窗体，
然后自己选择年、月、日即可，单击可以输入到活动单元格中。可以随时在公历与农历间切换。</a><a
href='mailto:888@excelbbx.com'>单击反馈意见</a></marquee></p></body></html>"
End Sub
```

以上过程表示初始化窗体时向窗体中的网页控件写入网页代码，从而产生动态文字。

step 4 进入模块中，录入以下用于显示帮助窗体的代码：

```
Sub about()'①代码存放位置：类模块中②随书光盘中有每一句代码含义注释
  Help.Show 0
End Sub
```

22.4.7 定制功能区菜单

当工作簿生成插件后，无法利用<Alt+F8>组合键来调用插件中的 Sub 过程，因此应按以下步骤为加载宏文件设计功能区菜单。

step 1 将工作簿保存为“日历控件.xlsm”。

step 2 打开 CustomUIEditor 软件，然后按<Ctrl+O>组合键打开“日历控件.xlsm”。

step 3 单击菜单中的“Insert”→“Office 2007 Custom UI Part”命令从而插入一个 customUI.xml 文件，然后在代码窗口中录入以下代码，用于在“视图”选项卡创建新组及按钮：

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui">
  <ribbon startFromScratch="false">
    <tabs>
      <tab idMso="TabHome" >
        <group id="Group1" label="日历控件" insertBeforeMso="GroupFont">
          <button id="button1" label="插入日期"  imageMso="Pushpin" size="large"
onAction="ShowCalendar"/>
          <button id="button2" label="关于"  imageMso="Help" size="large" onAction=
"about"/>
        </group>
      </tab>
    </tabs>
  </ribbon>
</customUI>
```

以上代码表示在“开始”选项卡的字体组之前插入一个名为“日历控件”的新组，然后在组中创建两个名为“插入日期”和“关于”的命令按钮。

step 4 保存文件，关闭 CustomUIEditor 软件。

step 5 双击打开“日历控件.xlsm”，按<Alt+F11>组合键进入 VBE 界面，进入模块中，在过程“ShowCalendar”和“about”的括号中加上“control As IRibbonControl”，表示这两个过程可以用于响应功能区菜单。如果忽略参数，单击菜单时不会执行命令。

如图 22.11 所示是本节所设计的“日历控件.xlsm”功能区菜单界面。

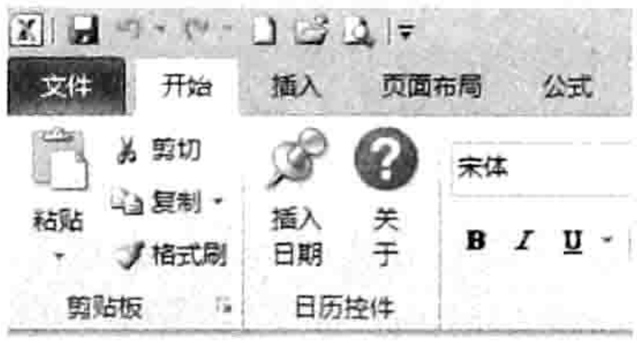


图 22.11　日历控件的功能区菜单

本例文件参见光盘：..\第二十二章\日历控件.xlsm

22.4.8　测试并发布插件

编写完代码后有必要测试代码，测试成功后再发布插件。测试步骤如下。

step 1　双击打开工作簿“日历控件.xlsm”。如果处于打开状态则重启工作簿。

step 2　按<Shift+F11>组合键新建一个工作表，然后单击功能区中的“开始”→“插入日期”命令，从而弹出“窗体日历”对话框。窗体中默认显示本月的日历。

step 3　单击其中编号是 11 的标签，A1 单元格将自动产生 2014/3/11，然后活动单元格下移一格，效果如图 22.12 所示。

图 22.12　单击录入日期

step 4　单击“以农历日期录入”选项，再单击编号为 10 的标签，活动单元格中自动输入“2014/3/14”，同时活动单元格下移一格。

step 5　单击窗体顶部右方的旋转按钮，每单击一次可以将日期前移或者后移一个月，而单击左方的旋转按钮则可以将日期前移或者后移一年。

step 6　单击“以农历日期录入”选项，然后再单击标签，录入在单元格中的日期将显示为农历。

step 7　新建工作簿，可以发现“开始”选项卡中没有与日历控件相关的菜单。

经过以上操作可以得出结论——所有代码在当前工作簿中工作正常，但是不能跨工作簿使用，唯一的解决办法是将当前工作簿发布为加载宏并安装。

事实上，将工作簿保存为加载宏并且安装可以用两句代码完成，代码如下：

```
Sub 发布()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  ThisWorkbook.IsAddin = True
  ThisWorkbook.SaveCopyAs Application.Path & "\XLSTART\日历控件.xlam"
End Sub
```

以上过程中的 ThisWorkbook 代表代码所在工作簿，使用 ThisWorkbook 要比使用 ActiveWorkbook 稳妥得多，不管“日历控件.xlsm”是否为活动工作簿都可以执行成功。代码中的 SaveCopyAs 方法代表将工作簿另存一份，在另存的同时可以改名也可以更改文件格式。

执行以上过程后，关闭当前的“日历控件.xlsm”，然后打开其他任意工作簿，将会发现在任意工作簿的“开始”菜单中都产生了“插入日期”和“关于”菜单，而且可以随时调用，表示插件设计成功，同时也安装成功。

知识补充：在发布插件之前应该锁定工程，从而避免在工作中因操作失误而破坏代码。操作方法为：在 VBE 界面单击菜单中的“工具”→“VBAProject 属性”→“保护”命令，然后在对话框中将“查看时锁定工程”选项打钩，并且在下方的“密码”和“确认密码”两个文本框中录入密码（本例的工程密码是 123）。

22.5 开发文本与数值互换插件

将选区中的数值批量转换成文本，或者将选区中文本形式的数字转换成数值，这两者都是在日常工作中比较常见的需求。本节将展示文本与数值互换的插件的设计过程。

22.5.1 确认所需具备的功能

本插件功能如下。

（1）将选区中的数值批量转换成文本。

（2）将选区中文本形式的数字转换成数值。

（3）支持区域，即选区可以是不连续、不规则的多个区域。

（4）过程通过功能区中的菜单调用，菜单放置在“数据”选项卡中。

22.5.2 编写主程序

将选区中的数值批量转换成文本或者将选区中文本形式的数字转换成数值都是比较简单的，直接使用一个循环语句即可完成。不过设计插件时除了完成既定的功能外还需考虑代码的容错性和速度，因此除循环语句外还需要一些辅助性的代码。具体步骤如下。

step 1 打开 Excel，按<Alt+F11>组合键进入 VBE 窗口中。

step 2 单击菜单中的“插入”→“模块”命令，然后在模块中录入以下三段代码：

```
Sub WenBeng()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Dim i As Byte
  If ActiveSheet.ProtectContents Then MsgBox "工作表已保护，本程序拒绝执行！",
vbInformation, "友情提示": Exit Sub
  If TypeName(Selection) <> "Range" Then Exit Sub
  Selection.NumberFormatLocal = "@"
  Application.ScreenUpdating = False
  For i = 1 To Selection.Areas.Count
    Selection.Areas(i) = Evaluate("=" & Selection.Areas(i).Address & "&""""")
  Next i
  Application.ScreenUpdating = True
End Sub
```

以上过程表示将数值转换成文本。

```
Sub ShuZhi()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  Dim i As Byte
```

```
  If ActiveSheet.ProtectContents Then MsgBox "工作表已保护,本程序拒绝执行！",
vbInformation, "友情提示": Exit Sub
  If TypeName(Selection) <> "Range" Then Exit Sub
  Application.ScreenUpdating = False
  Selection.NumberFormatLocal = "G/通用格式"
  For i = 1 To Selection.Areas.Count
    Selection.Areas(i) = Selection.Areas(i).Value
  Next i
  Application.ScreenUpdating = True
End Sub
```

以上过程表示将文本转换成数值。

```
Sub about()'代码存放位置：模块中
  MsgBox "数值转文本：将当前选择的区域中的数值转换成文本形式，从而使其不参与汇总。" & Chr(13)
& "文本转换值：将当前选择的区域中的文本形式的数字转换成数值，从而避免运算时漏算这些数值。",
vbInformation
End Sub
```

以上过程用于提示本插件的功能。

step 3 将工作簿保存为“文本与数值互换.xlsm”。

22.5.3 定制功能区菜单

设计功能区菜单并不是必需的，但是为插件提供功能区菜单能提升插件的易用性。

设计功能区菜单的步骤如下。

step 1 打开 CustomUIEditor 软件，然后按<Ctrl+O>组合键打开“文本与数值互换.xlsm”。

step 2 单击菜单中的“Insert”→“Office 2007 Custom UI Part”命令从而插入一个 customUI.xml 文件，然后在代码窗口中录入以下代码，用于在“数据”选项卡创建新组及按钮：

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui">
  <ribbon startFromScratch="false">
      <tabs>
  <tab idMso="TabData" >
      <group id="Group1"label="文本与数值"insertBeforeMso="GroupGetExternalData">
        <button   id="button1"   label="  文  本  转  数  值  "      imageMso=
"EquationMatrixGallery" size="large" onAction="ShuZhi"/>
        <button   id="button2"   label="  数  值  转  文  本  "      imageMso=
"FormControlEditBox" size="large" onAction="WenBeng"/>
        <button  id="button3"  label="关于"   imageMso="Help"  size="large"
onAction="about"/>
      </group>
    </tab>
   </tabs>
  </ribbon>
</customUI>
```

step 3 按<Strl+S>组合键保存代码，然后关闭 CustomUIEditor 软件。

step 4 双击打开“文本与数值互换.xlsm”，在“数据”选项卡中可看到如图 22.13 所示的菜单。

step 5 按<Alt+F11>组合键进入 VBE 界面，进入模块中在 3 个过程的括号中加上“control As IRibbonControl”，表示这 3 个过程可以用于响应功能区菜单。如果忽略参数，单击菜单时不会执行命令。

图 22.13　文本与数值互换的功能区菜单

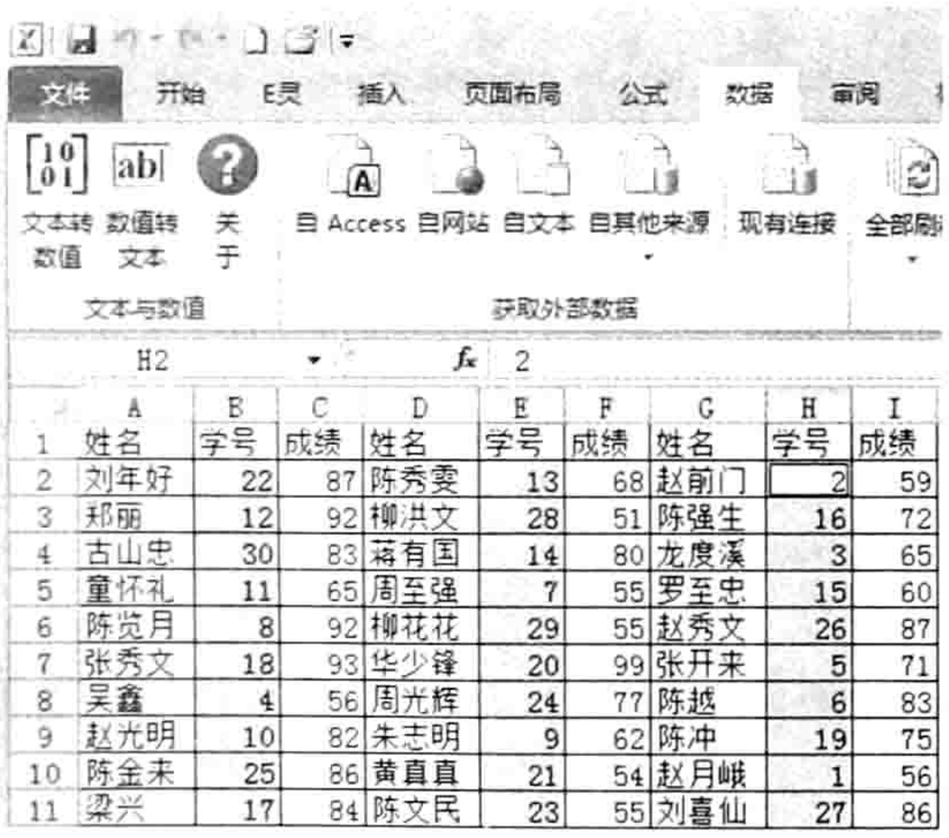
本例文件参见光盘：..\第二十二章\文本与数值互换.xlsm

22.5.4　测试代码并发布插件

在发布插件前必须测试代码的功能，完善后才能发布。本插件的测试步骤如下。

step 1　在工作表中录入如图 22.14 所示的数据，其中 B 列、E 列和 H 列的学号需要转换成文本。

step 2　选择学号，然后单击功能区中的“数据”→“数值转文本”命令，转换结果如图 22.15 所示。



	A	B	C	D	E	F	G	H	I
1	姓名	学号	成绩	姓名	学号	成绩	姓名	学号	成绩
2	刘年好	22	87	陈秀雯	13	68	赵前门	2	59
3	郑丽	12	92	柳洪文	28	51	陈强生	16	72
4	古山忠	30	83	蒋有国	14	80	龙度溪	3	65
5	童怀礼	11	65	周至强	7	55	罗至忠	15	60
6	陈览月	8	92	柳花花	29	55	赵秀文	26	87
7	张秀文	18	93	华少锋	20	99	张开来	5	71
8	吴鑫	4	56	周光辉	24	77	陈越	6	83
9	赵光明	10	82	朱志明	9	62	陈冲	19	75
10	陈金来	25	86	黄真真	21	54	赵月峨	1	56
11	梁兴	17	84	陈文民	23	55	刘喜仙	27	86

图 22.14　待转换成文本的学号

	A	B	C	D	E	F	G	H	I
1	姓名	学号	成绩	姓名	学号	成绩	姓名	学号	成绩
2	刘年好	22	87	陈秀雯	13	68	赵前门	2	59
3	郑丽	12	92	柳洪文	28	51	陈强生	16	72
4	古山忠	30	83	蒋有国	14	80	龙度溪	3	65
5	童怀礼	11	65	周至强	7	55	罗至忠	15	60
6	陈览月	8	92	柳花花	29	55	赵秀文	26	87
7	张秀文	18	93	华少锋	20	99	张开来	5	71
8	吴鑫	4	56	周光辉	24	77	陈越	6	83
9	赵光明	10	82	朱志明	9	62	陈冲	19	75
10	陈金来	25	86	黄真真	21	54	赵月峨	1	56
11	梁兴	17	84	陈文民	23	55	刘喜仙	27	86

图 22.15　转成文本后的学号

step 3　单击功能区中的“数据”→“文本转数值”命令，选区中的所有文本将被还原为数值。

经过以上测试，表示代码符合需求，接下来的工作是发布插件。

首先单击“工具”→“VBAProject 属性”→“保护”命令，在对话框中将“查看时锁定工程”选项打钩，并且在下方的“密码”和“确认密码”两个文本框中录入密码（本例的工程密码是 123）。然后使用以下代码发布插件：

```
Sub 发布()'代码存放位置：模块中②随书光盘中有每一句代码的含义注释
  ThisWorkbook.IsAddin = True
  ThisWorkbook.SaveCopyAs Application.Path & "\XLSTART\文本与数值互换.xlam"
End Sub
```

发布成功后关闭活动工作簿，以后就可以随时在功能区中调用文本与数值互换的功能了。

◆　◆　◆

第 23 章　代码封装技巧

开发插件的首选是采用 xla 和 xlam 格式的加载宏，这是一切 Excel 插件中开发过程最简单、使用最方便的两种格式。不过这两种格式的插件安全性极低，即使对工程加密也不足以保护代码，几秒钟内即可获取插件的源代码，因此开发商业性质的插件一般采用安全性更高的 DLL 格式的加载项。

有很多软件可以将 VBA 代码封装成 DLL 格式的加载项，本章只讲述 VB 6.0 企业版封装代码的技术，以及通过 Inno Setup 软件将加载项打包成 EXE 格式的安装程序的思路。

23.1　封装自定义函数

封装代码包含封装自定义函数和 Sub 过程。装封自定义函数极其简单，将 VBA 代码复制到 VB 中，然后单击菜单生成 DLL 文件就完成了，因此本章从封装自定义函数开始讲起。

23.1.1　安装 VB 6.0 企业版

封装 VBA 代码之前需要安装 VB 6.0 企业版，安装步骤如下。

step 1　将 VB 6.0 企业版安装盘插入光驱中（如果硬盘中有安装文件也可以不使用光盘）。

step 2　双击“Setup.exe”启动安装向导，如图 23.1 所示。

图 23.1　启动 VB 6.0 中文企业版安装向导

step 3　单击“下一步”按钮，打开“最终用户许可协议”对话框，在对话框中选择“接受协议”选项，再单击“下一步”按钮。

step 4　在“产品号和用户”对话框中输入有效的产品密钥，而用户姓名和公司则随意填写。

step 5　单击“下一步”按钮，打开“自定义-服务器安装程序选项”对话框，选择“安装 Visual Basic 6.0 中文企业版”，然后单击“下一步”按钮。

step 6　可以直接单击“下一步”按钮，让安装目录保持默认状态，然后单击“确定”按钮开始安装。

step 7　在弹出的“安装类型”对话框中选择“典型安装”，表示安装 VB 的基本组件。当安装程序复制完数据后会提示安装成功。

安装完毕后应该重启一次计算机，重启后在 Windows 的“开始”菜单中可以找到“Microsoft Visual Basic 6.0 中文版”，单击它即可打开 VB 6.0 中文企业版。

23.1.2　封装自定义函数

以封装本书 16.5.4 节中名为“合计”的自定义函数为例，封装步骤如下。

step 1　单击 Windows“开始”菜单中的“Microsoft Visual Basic 6.0 中文版”，从而启动 VB 6.0。

step 2　在如图 23.2 所示的对话框中选择“ActiveX Dll”。

step 3　在 VB 软件靠右边的工程资源管理器中将工程的默认名称“工程 1”修改为“我的函数”，再将类模块的默认名称“Class1”修改为“合计”，效果如图 23.3 所示。

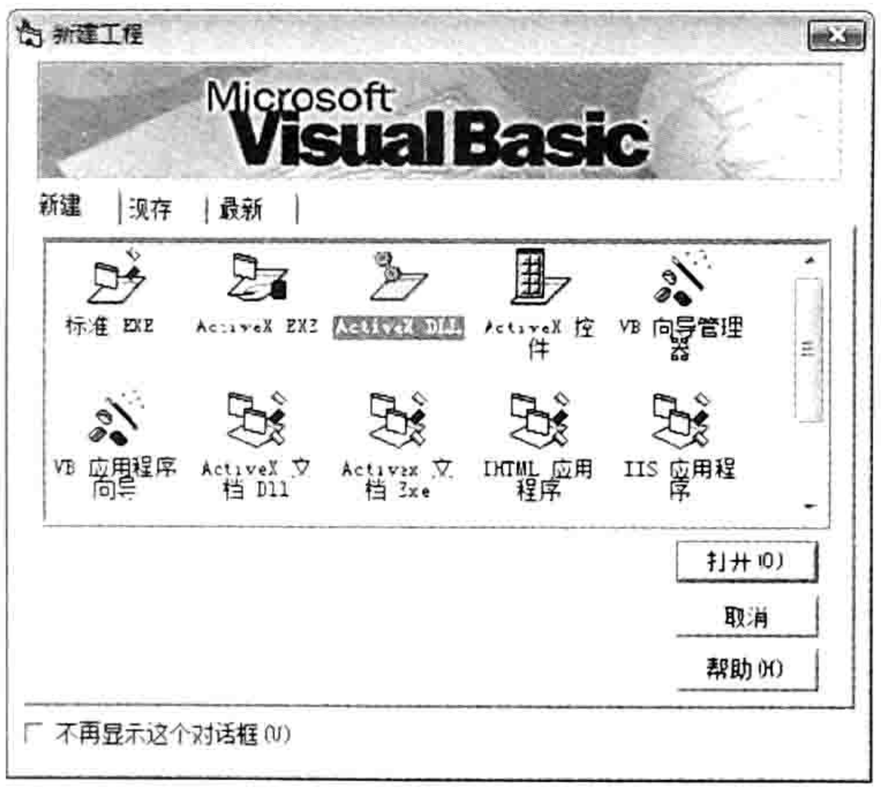


图 23.2　新建 ActiveX Dll 工程

图 23.3　修改工程名称和类模块名称

step 4　单击菜单中的“工程”→“引用”命令，打开“引用”对话框，并在对话框中将“Microsoft Excel 14.0 Object Library”复选框打钩，最后单击“确定”按钮。如图 23.4 所示的是 VB 的“引用”对话框。

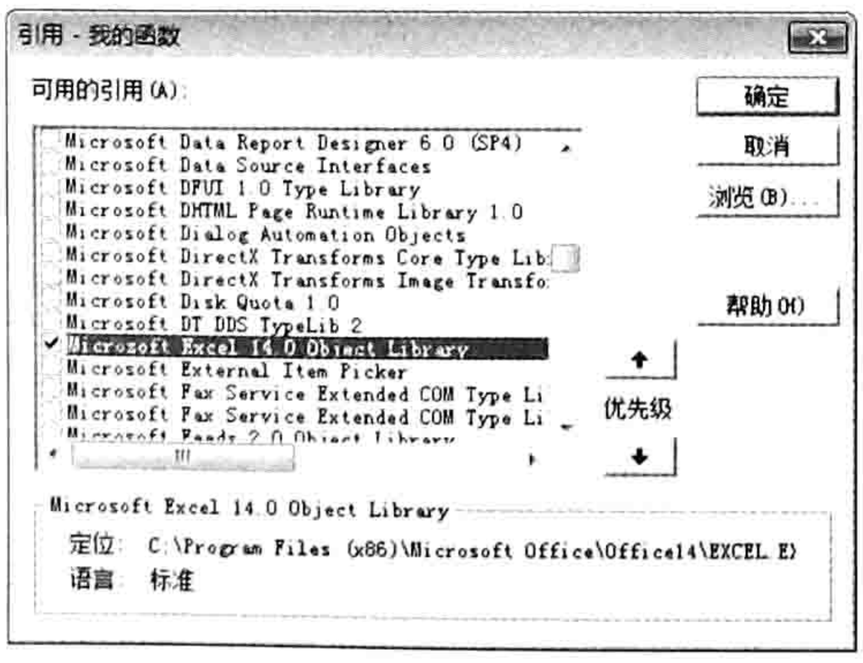


图 23.4　“引用”对话框

step 5　将本书 16.5.4 节中名为“合计”的自定义函数的源代码复制到类模块的代码窗口中。

step 6　单击菜单中的“文件”→“生成我的函数.dll”命令，在“生成工程”对话框中选择保存位置为“D:\我的开发工具”，并且将文件名称由“我的函数.dll”修改为“合计.dll”。如果 D 盘没有“我的开发工具”文件夹则可以手工新建一个。

step 7 单击“生成工程”对话框中的“确定”按钮完成封装过程。

23.1.3 安装自定义函数

23.1.2 节中封装好的自定义函数保存在“D:\我的开发工具”路径中，名为“合计.dll”。安装该自定义函数的步骤如下。

step 1 打开 Excel，按<Alt>键，松开后再按<T>键，接着松开后再按<I>键从而打开“加载宏”对话框。

step 2 单击“自动化”按钮打开“自动化服务器”对话框，单击“自动化服务器”对话框中的“浏览”按钮打开“浏览”对话框，进入“D:\我的开发工具”路径中选中文件“合计.dll”，然后单击“确定”按钮返回“自动化服务器”对话框中。

step 3 将滚动条下拉到末端，可以看到“我的函数.合计”项目，如图 23.5 所示。选中该项目后单击“确定”按钮返回“加载宏”对话框，“我的函数.合计”项目将出现在“加载宏”列表中，如图 23.6 所示。

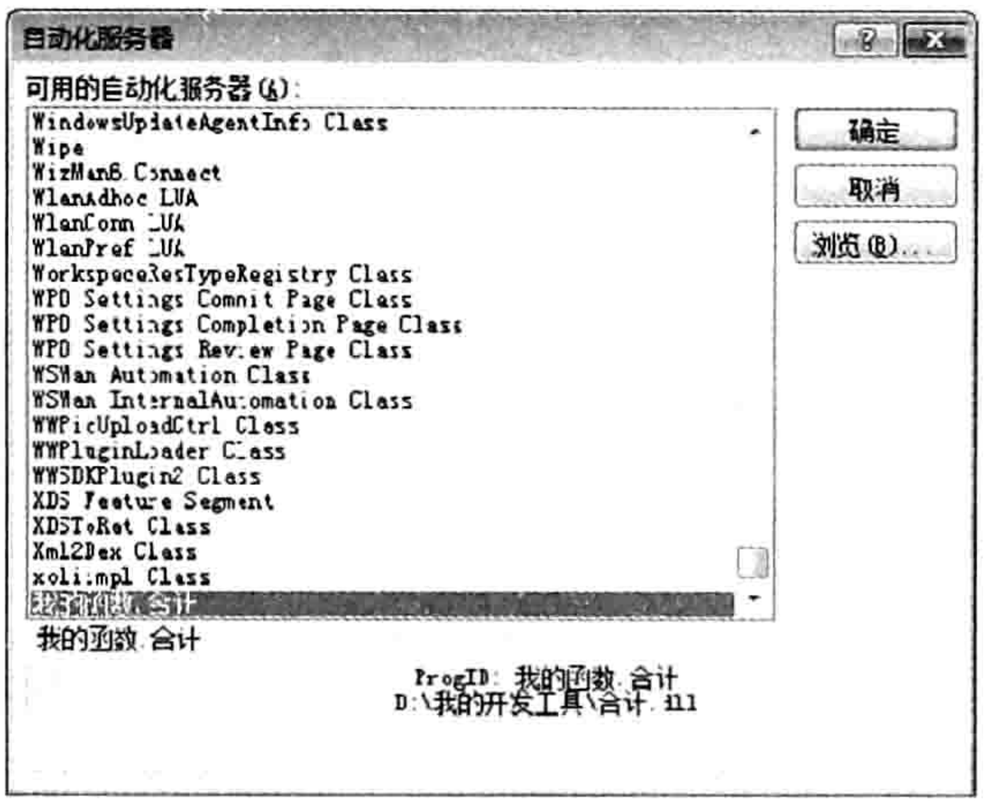


图 23.5 将函数安装到自动化服务器

图 23.6 加载宏列表中的自定义函数

step 4 在 A1 单元格录入用于测试公式的数据，然后在 B1 单元格录入以下公式：

```
=合计(A1,"元")
```

公式的计算结果为 6480.5，表明函数已经封装成功，可以正常执行运算，如图 23.7 所示。

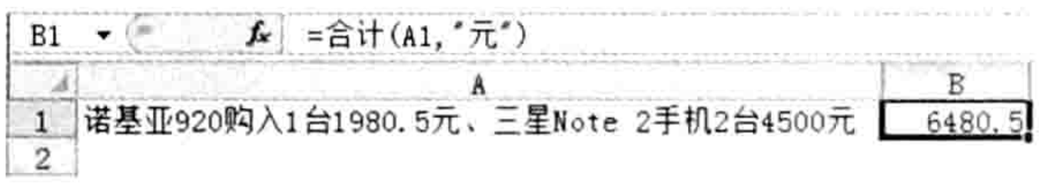


图 23.7 在单元格中调用函数“合计”

本例文件参见光盘：..\第二十三章\封装函数\

23.2 封装 Sub 过程

封装 Sub 过程远比封装 Function 过程复杂。

本节以 22.5.2 节中开发的“文本与数值互换”插件为例，展示 Sub 过程和功能区代码的封装过程。

23.2.1　建立 VB 工程

封装 Sub 过程和封装自定义函数的思路大大不同，封装 Sub 过程采用的 COM 加载项，在启动 VB 后的第一个步骤就不同，其后的封装过程和安装方法都不相同。

在 VB 中创建 COM 加载的工程包含以下步骤。

step 1　单击 Windows“开始”菜单中的“Microsoft Visual Basic 6.0 中文版”，从而启用 VB 6.0。

step 2　在图 23.8 所示的对话框中选择“外接程序”选项。

step 3　在右边的工程资源管理器中将工程默认名称“frmAddIn”修改为“文本与数值互换”，并且对名为“frmAddIn”的窗体单击鼠标右键，在弹出的快捷菜单中选择“移除 frmAddIn”命令。

此时的工程资源管理器将显示为图 23.9 所示的效果。

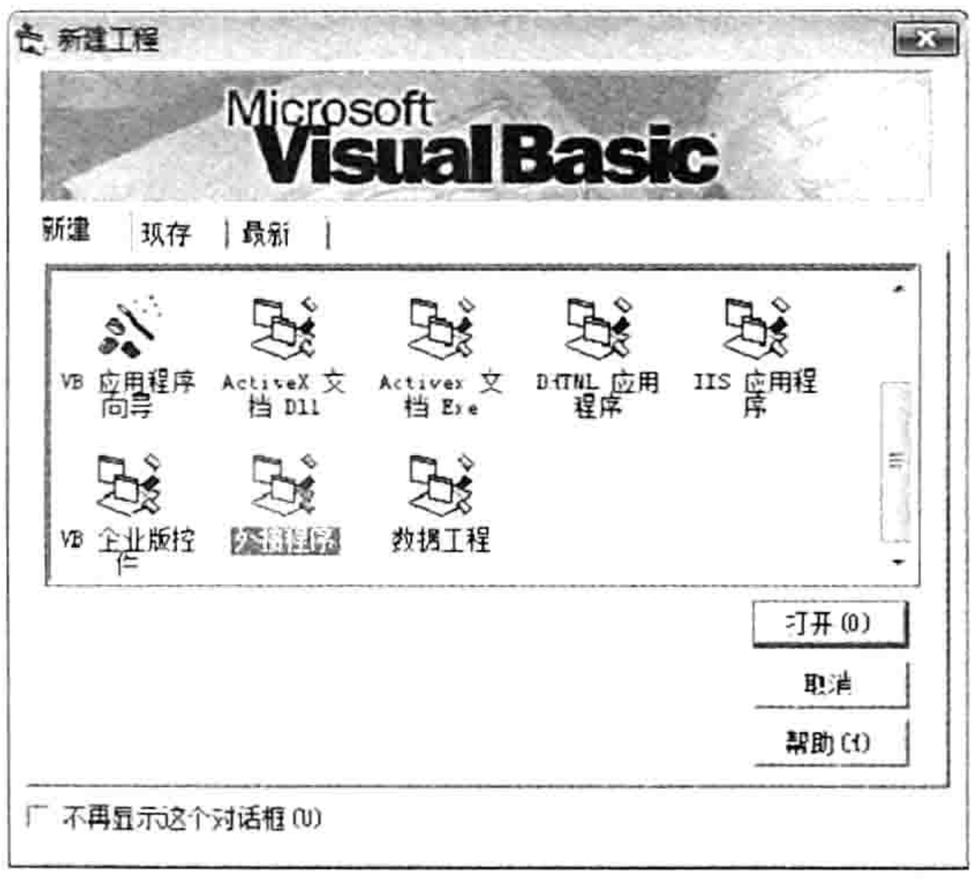


图 23.8　新建外接程序

图 23.9　工程资源管理器

23.2.2　添加引用

COM 加载项必须引用 Office 14.0 和 Excel 14.0（即 Office 2010，如果读者使用的是 Office 2013，那么相应地将引用对象改为 Office 15.0 和 Excel 15.0），否则程序无法在 Excel 中正常使用，具体的操作步骤如下。

step 1　单击菜单中的“工程”→“引用”命令，打开“引用”对话框。

step 2　将“Microsoft Office 14.0 Object Library”和“Microsoft Excel 14.0 Object Library”两个项目上打钩，然后单击“确定”按钮返回 VB 主窗口。

step 3　对工程资源管理器中名为 Connect 的设计器单击鼠标右键，在弹出的菜单中选择“查看对象”命令，在打开的新窗口中有两个文本框和 3 个复选框，它们用于描述当前外接程序的名称、补充说明和指定外接程序应该安装在哪一个应用程序之中。

step 4　将“外接程序显示名称”修改为“文本与数值互换”，将“外接程序描述”修改为“这是一个文本与数值互换的工具，支持不连续、不规则的多个区域……”，再将“应用程序”设置为“Microsoft Excel”，“应用程序版本”设置为“Microsoft Excel14.0”，“初始化加载行为”设置为“Starup”。最终效果如图 23.10 所示。

假设读者使用的 Office 2013 则应将版本由 14.0 改为 15.0。
初始化加载行为“Startup”表示安装此 COM 加载项后可随 Excel 软件同步启动。

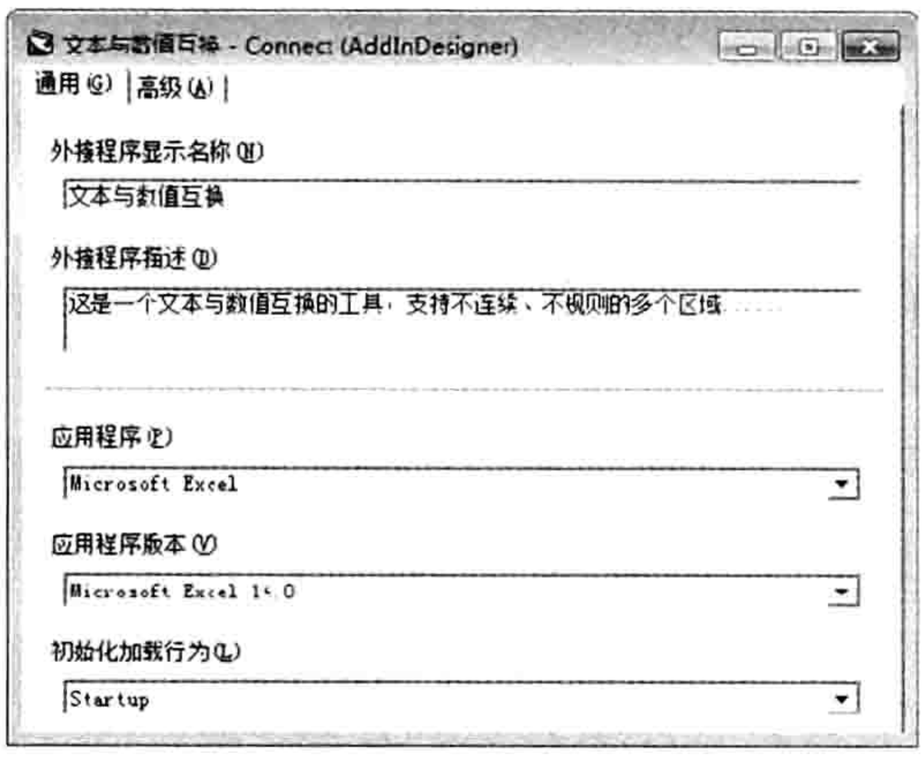


图 23.10　设置外接程序的名称、描述与应用程序

23.2.3　写入代码

前面的准备工作完成后，接下来的工作是在模块和 Connect 设计器中录入代码，这是封装代码最重要的一个环节。

制作 COM 加载项除了要封装的 Sub 过程之外还需要编写 VB 外接程序中必不可少的 OnConnection 过程，以及生成功能区菜单专用的 IRibbonExtensibility_GetCustomUI 函数。具体步骤如下。

step 1　单击菜单中的“工程”→“添加模块”命令，并在模块中录入以下用于声明公共变量的代码：

```
Public xlApp As Excel.Application  '声明一个公共变量，代表 Excel 应用程序
```

step 2　在工程资源管理器中对设计器 Connect 单击鼠标右键，在弹出的快捷菜单中选择“查看代码”命令。

step 3　删除代码窗口中的一切代码。

step 4　单击代码窗口上方的对象列表，从列表中选择“AddinInstance”，VB 会自动产生名为 AddinInstance_OnConnection 的程序外壳，具体代码如下：

```
Private Sub AddinInstance_OnConnection(ByVal Application As Object, ByVal
ConnectMode As AddInDesignerObjects.ext_ConnectMode, ByVal AddInInst As Object,
custom() As Variant)

End Sub
```

操作界面如图 23.11 所示。

图 23.11　从对象列表中选择 AddinInstance

step 5　在 AddinInstance_OnConnection 过程中插入以下代码，表示对公共变量 xlApp 赋值为 Application，此处的 Application 代表 Excel 应用程序。

```
Set xlApp = Application
```

step 6　将本书 22.5.2 节中关于“文本与数值互换”插件的 3 个 Sub 过程的源代码复制到设计器 Connect 的代码窗口中。

step 7　修改 Sub 过程中所有关于 Application 对象的代码，包含 Application 对象的子对象、Application 对象的方法和 Application 对象的属性相关的代码。

修改内容包含两项，其一是将代码中被省略掉的 Application 补充完整，其二是将 Application 替换成公共变量 xlApp。

本例要封装的 3 个过程中的 ActiveSheet 属于 Application 对象的子对象，Evaluate 属于 Application 对象的方法，ScreenUpdating 和 Selection 属于 Application 对象的属性，因此需要在它们之前添加“xlApp.”。由于 ScreenUpdating 前面已经有 Application，因此将 Application 替换成 xlApp 即可。

修改完成后的完整代码如下：

```
'①代码存放位置：设计器 Connect 的代码窗口中②随书光盘中有每一句代码的含义注释
Sub WenBeng(control As IRibbonControl)
  Dim i As Byte
  If xlApp.ActiveSheet.ProtectContents Then MsgBox "工作表已保护,本程序拒绝执行!
", vbInformation, "友情提示": Exit Sub
  If TypeName(xlApp.Selection) <> "Range" Then Exit Sub
  xlApp.Selection.NumberFormatLocal = "@"
  xlApp.ScreenUpdating = False
  For i = 1 To xlApp.Selection.Areas.Count
    xlApp.Selection.Areas(i) = xlApp.Evaluate("=" & xlApp.Selection.Areas(i).
Address & "&""""""")
  Next i
  xlApp.ScreenUpdating = True
End Sub
Sub ShuZhi(control As IRibbonControl)
  Dim i As Byte
  If xlApp.ActiveSheet.ProtectContents Then MsgBox "工作表已保护,本程序拒绝执行!
", vbInformation, "友情提示": Exit Sub
  If TypeName(xlApp.Selection) <> "Range" Then Exit Sub
  xlApp.ScreenUpdating = False
  xlApp.Selection.NumberFormatLocal = "G/通用格式"
  For i = 1 To xlApp.Selection.Areas.Count
    xlApp.Selection.Areas(i) = xlApp.Selection.Areas(i).Value
  Next i
  xlApp.ScreenUpdating = True
End Sub
Sub about(control As IRibbonControl)
  MsgBox "数值转文本：将当前选择的区域中的数值转换成文本形式，从而使其不参与汇总。" &
Chr(13) & "文本转换值：将当前选择的区域中的文本形式的数字转换成数值，从而避免运算时漏算
这些数值。", vbInformation
End Sub
```

step 8　单击菜单中的“外接程序”→“外接程序管理器”命令，在“可用外接程序”列表中选中“VB 6 资源编辑器”，然后将下方的“加载/卸载”复选框打钩，操作效果如图 23.12 所示。

step 9　单击菜单中的“工具”→“资源编辑器”命令，单击工具栏中包含“ABC”字样的图标（图标样式为 abc ）打开“编辑字符串表”对话框。

step 10 双击标识号 101 右方的文本框，然后将本书 22.5.3 节中用于生成功能区菜单的 XML 代码粘贴在其中，操作界面如图 23.13 所示。

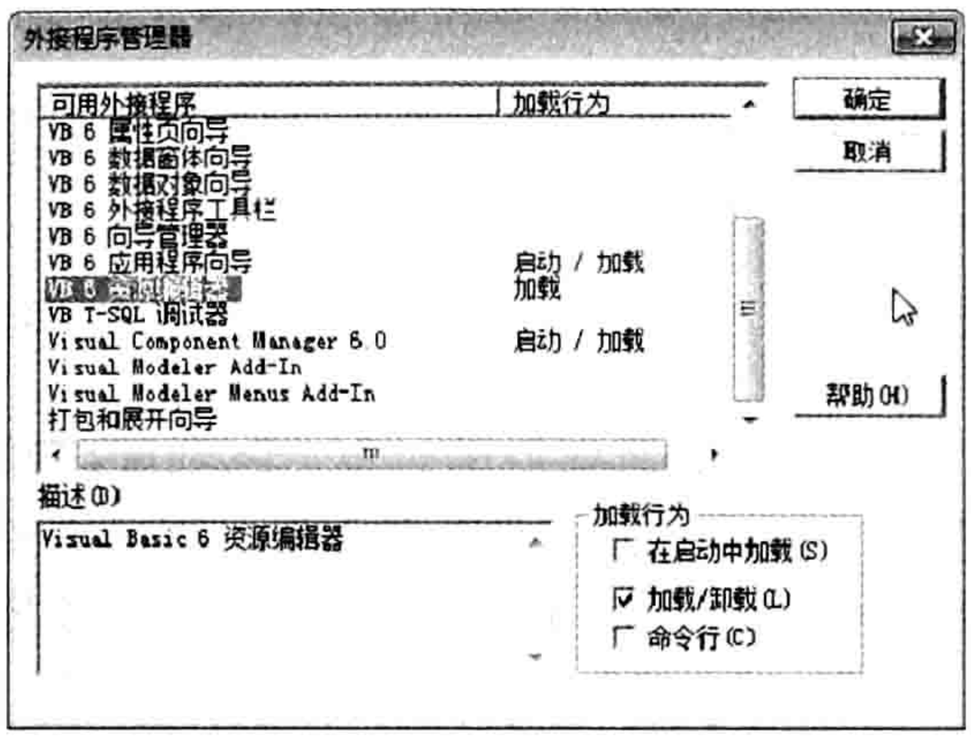


图 23.12 添加资源编辑器

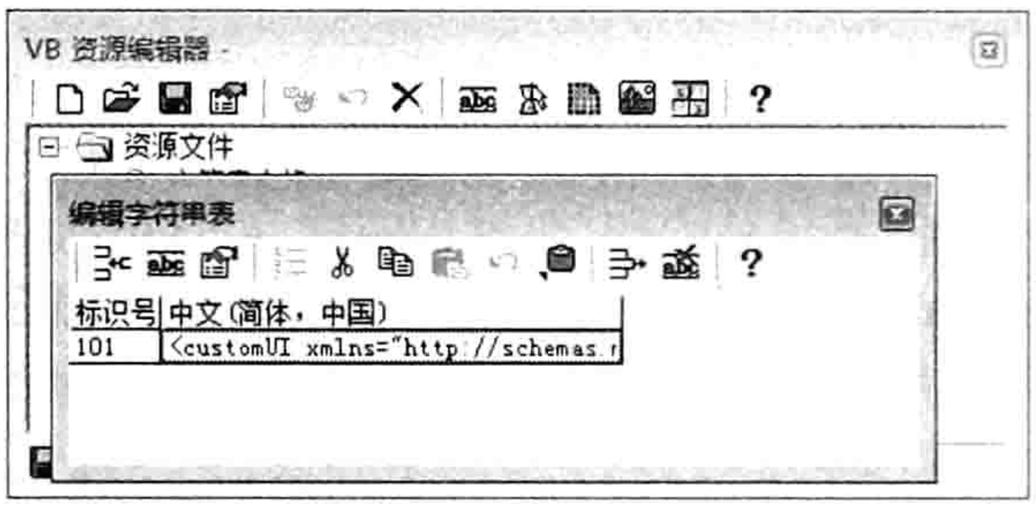


图 23.13 将 xml 代码粘贴到资源编辑器中

step 11 关闭“编辑字符串表”和“资源编辑器”对话框。

step 12 在 Connect 设计器的代码窗口顶端录入以下代码：

```
Implements IribbonExtensibility
```

此代码用于创建一个 IRibbonExtensibility 接口，使 VB 能与 Excel 建立关联。如果忽略此参数则下一步的函数将无法执行。

step 13 继续录入以下函数代码：

```
Public Function IRibbonExtensibility_GetCustomUI(ByVal RibbonID As String) As String
  IRibbonExtensibility_GetCustomUI = LoadResString(101)
End Function
```

以上过程定义了一个函数，在函数过程中使用了 VB 的 LoadResString 函数去读取资源编辑器中标识号为 101 的字符串，然后将这个字符串赋值给函数 IRibbonExtensibility_GetCustomUI，COM 加载项会调用本函数的返回值去创建功能区菜单。

23.2.4 发布 COM 加载项

将要封装的代码复制到 VB 6.0 企业版中并修改完成后，下一步就是发布 COM 加载项。

在 VB 中发布 COM 加载项其实就是生成 DLL 文件，操作步骤比较简单，按<Ctrl+S>组合键保存 VB 工程，然后再单击菜单中的“文件”→“生成文本与数值互换.dll”命令即可。

本例中保存工程和生成 DLL 文件时都将路径选为“D:\我的开发工具”。

23.2.5 安装 COM 加载项

COM 加载项是一个 DLL 文件，安装 COM 加载项其实就是向 Excel 注册这个 DLL 文件。

注册 COM 加载项应使用 DOS 命令 regsvr32，具体格式如下：

```
regsvr32 COM 加载项的完整路径
```

由于本例的 COM 加载项保存在“D:\我的开发工具\文本与数值互换.dll”中，因此注册 COM 加载项的步骤如下。

step 1 单击 Windows 的“开始”菜单，然后选择“运行”命令（也可以按<Win+R>组合键）。

step 2 在“运行”对话框中录入命令“regsvr32 D:\我的开发工具\文本与数值互换.dll”，然后单击“确定”按钮，Windows 会弹出如图 23.14 所示的提示，表明注册 COM 加载项成功。

安装成功后，可以打开 Excel 进入“数据”选项卡验证插件生成的菜单是否存在。

如果要卸载 COM 加载项，那么应将命令修改为“regsvr32 -u D:\我的开发工具\文本与数值互换.dll”，参数“-u”代表卸载，即单词 Uninstall 的缩写。

事实上，安装 COM 加载项还可以通过单击功能区中的“开发工具”→“COM 加载项”命令，然后在“COM 加载项”对话框中安装或者删除 COM 加载项。图 23.15 为“COM 加载项”对话框，其列表中罗列了所有 COM 加载项，打钩者表示当前处于可用状态。

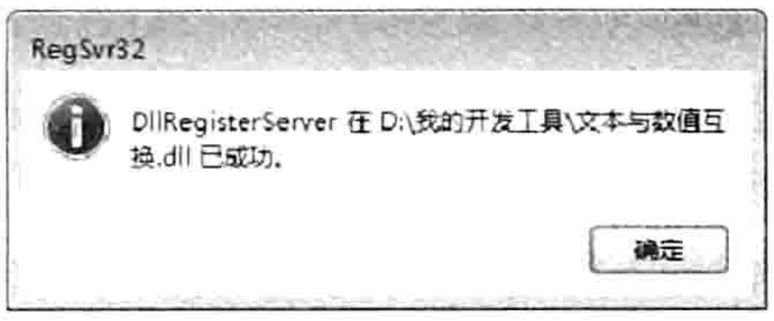


图 23.14　注册 COM 加载项成功

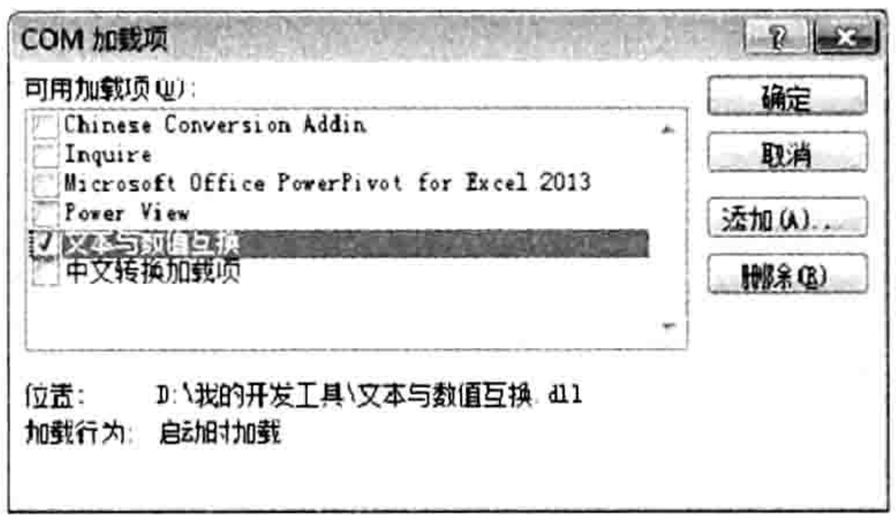


图 23.15　COM 加载项列表

本例文件参见光盘：..\第二十三章\文本与数值互换\

23.3　设计安装软件

COM 加载项可以使用命令注册，这意味着可以将它加工成安装程序，然后双击文件执行安装。

尽管 22.2.5 节中的 regsvr32 命令可以安装 COM 加载宏，但是显然专业性不够。本节向读者展示将 COM 加载项打包成 EXE 格式的可执行程序的操作步骤，以及提供专业的软件。

23.3.1　程序选择

制作安装文件的应用程序有很多，Windows 自带的 IExpress 就可以用于软件打包；著名的 WinRAR 也可以将 DLL 文件加工成 EXE 格式的安装软件，笔者推荐的是另一款更专业的免费开源软件——Inno Setup。

Inno Setup 软件功能多、体积小、设置方便，并且免费、开源，还支持多语言版本。本章以此软件为蓝本讲述制作安装程序的方法与思路。该软件最新版本可在以下网址下载：

http://www.newhua.com/soft/4693.htm。

23.3.2　使用程序向导制作安装软件

当安装好 Inno Setup 后，在 Windows 的“开始”菜单中将会出现 Inno Setup 软件的快捷方式，单击快捷方式可以启动软件的欢迎界面，如图 23.15 所示。在该界面中可以选择多种方式开启文件。对于新手而言应选择“用「脚本向导」创建新的脚本文件（S）”，从而按提示进行操作。

本节以“文本与数值互换.Dll”文件为例，展示通过脚本向导制作安装程序的方法，详细步骤如下。

step 1　在 D 盘创建“安装程序”文件夹，并将 23.2 节封装的“文本与数值互换.Dll”复制到文件夹中，然后再找一个 ico 图标文件存放在本文件夹中，并且命名为“图标.ICO”。

step 2　为了使软件更具专业性，在安装过程中显示相应的许可协议和软件介绍，在“安装程序”文件夹中新建“许可.txt”和“信息.txt”两个文本文件，并在其中存放与本软件相关的许

可内容和软件介绍（本案例中随意书写了一些许可协议，读者在可根据自己的需求更改）。

step 3 启动 Inno Setup，在图 23.16 的欢迎对话框中选择脚本向导，然后单击“确定”按钮。

step 4 在弹出的对话框中直接单击“下一步”按钮，不勾选“直接创建空的脚本文件”复选框。

step 5 在弹出的“应用程序信息”对话框中分别录入软件的名称、版本号、公司名和网址，然后单击“下一步”按钮。公司名和网址可以不填，对话框界面如图 23.17 所示。

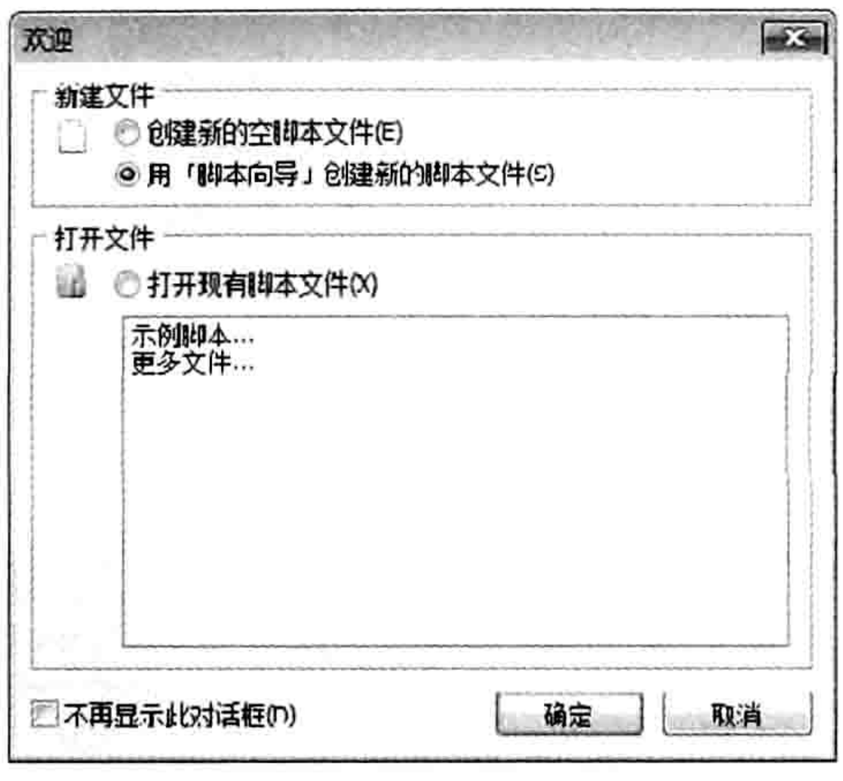


图 23.16 脚本向导

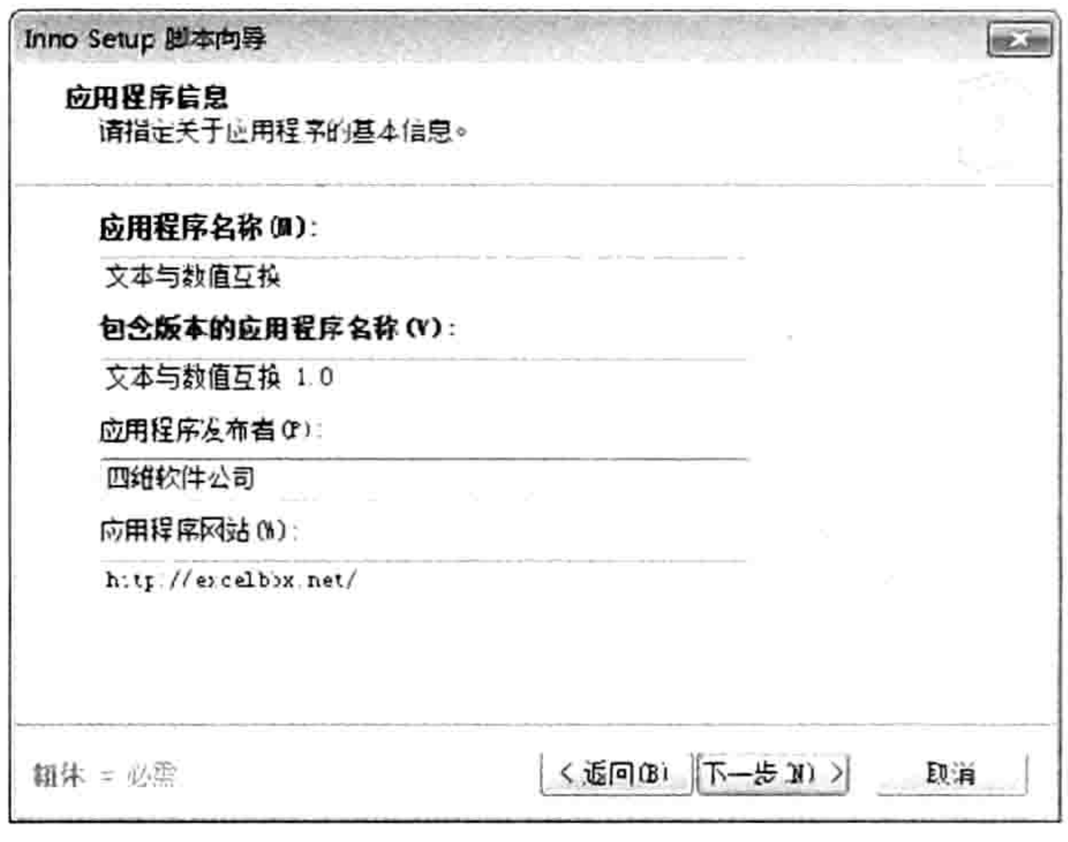


图 23.17 填写软件的名称、版本等信息

step 6 在“应用程序文件夹”对话框中有一个组合框，在其下拉列表中可以选择“程序文件文件夹”或者“自定义”，其中“程序文件文件夹”表示“Program Files”的默认位置，而选择自定义的话，可以自由指定位置，例如“D:\我的软件”或者“C:\123”等，本例采用“程序文件文件夹”，然后指定文件夹的名称，并且设置为不允许用户修改文件，最后单击“下一步”按钮。操作界面如图 23.18 所示。

step 7 在“应用程序文件”对话框中选择“应用程序没有主执行文本”复选框，然后单击“添加文件”按钮，在“打开”对话框中选择“D:\安装程序\文本与数值互换.dll”，单击“打开”按钮，表示将该文件封装为可执行程序。设置界面如图 23.19 所示。

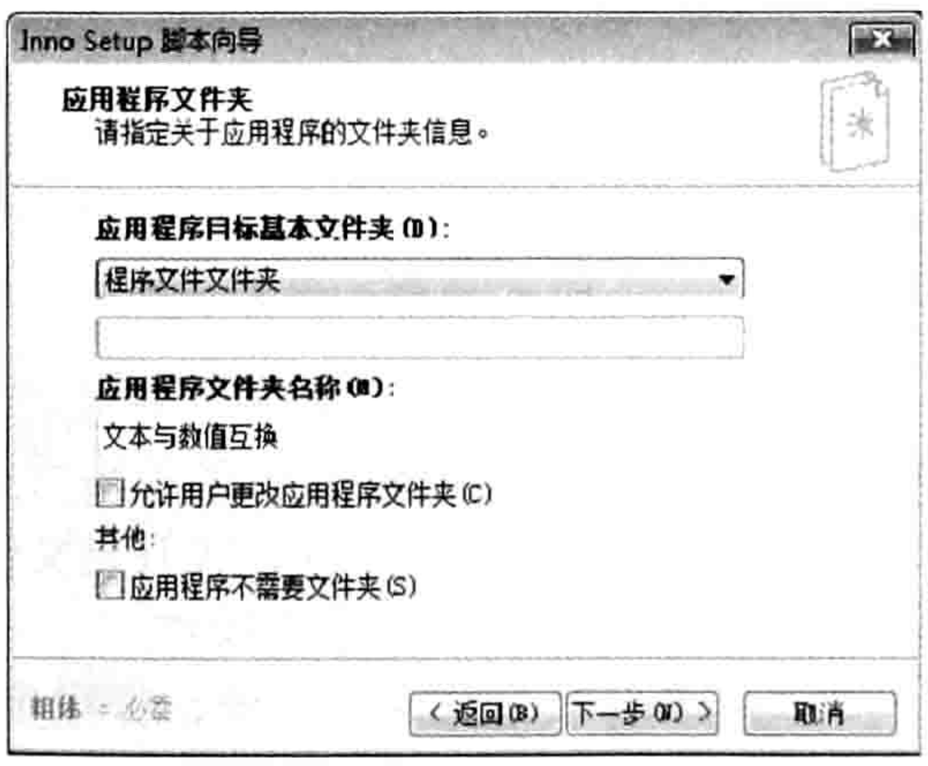


图 23.18 设置应用程序文件夹

图 23.19 指定应用程序的文件

step 8 在“应用程序图标”对话框中勾选所有复选框，然后将文件夹名称设置为“相同项与不同项”，并单击“下一步”按钮。设置界面如图 23.20 所示。

step 9 在“应用程序文档”对话框中分别指定许可文件和信息文件，如图 23.21 所示。

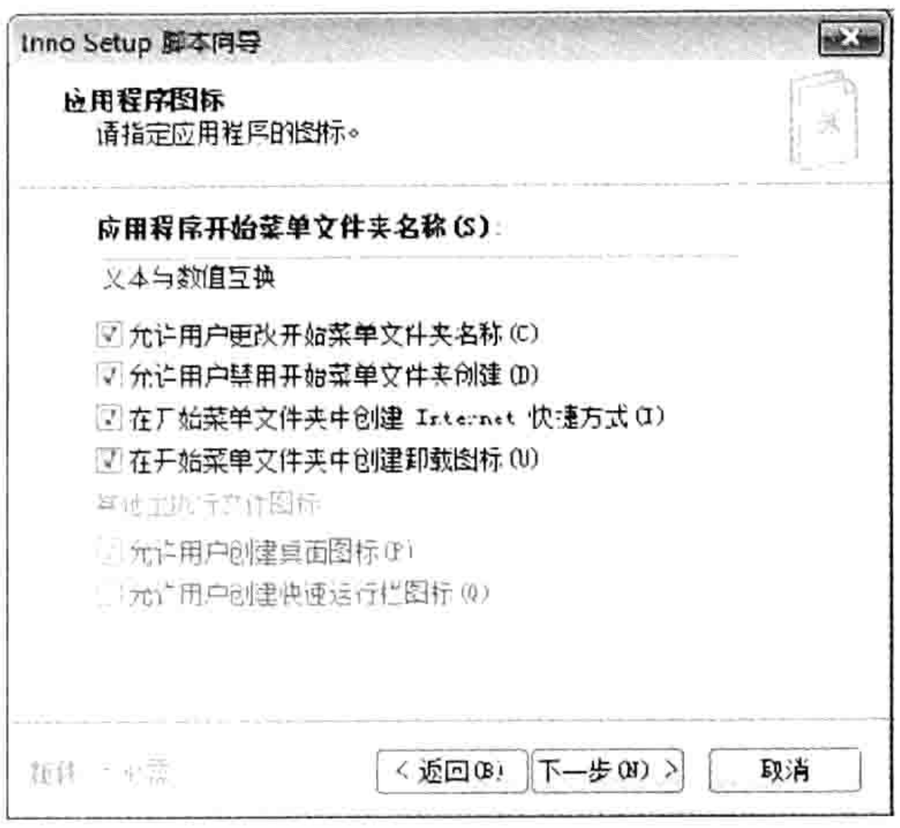


图 23.20　指定程序图标

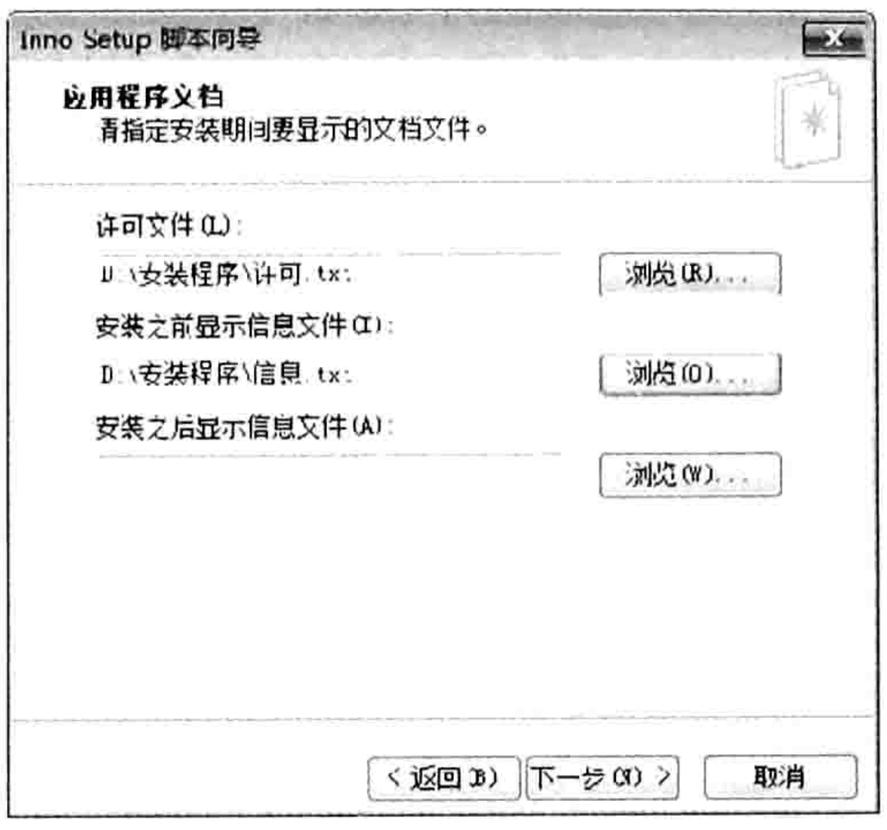


图 23.21　指定许可文件与信息文件

step 10　在“安装语言”对话框中直接单击“下一步”按钮，默认选择为“简体中文”。

step 11　在“编译设置”对话框中将输出路径和图标文件按如图 23.22 所示的方式设置，然后单击“下一步”按钮。

step 12　对于后面的两个提示对话框，直接单击“下一步”和“完成”按钮即可。最后一个对话框询问是否立即编译脚本，选择“是”，并将文件保存在“D:\安装程序”中，将文件取名为“安装程序.iss”。

执行以上所有步骤后，向导将生成如图 23.23 所示的脚本文件。

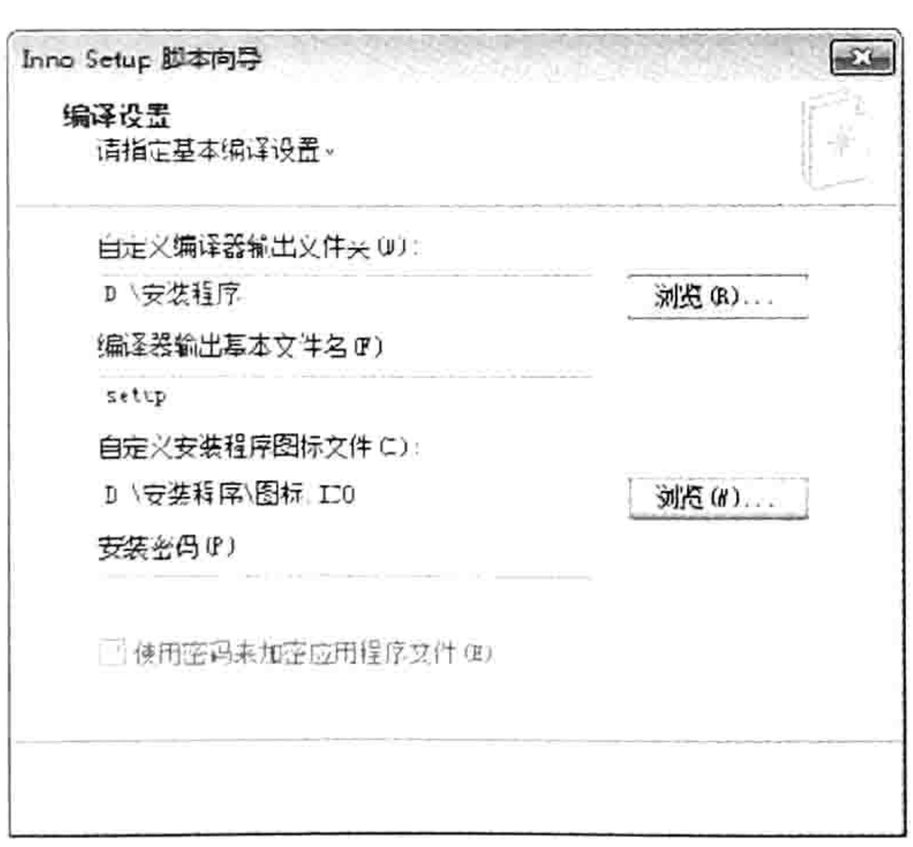


图 23.22　编译设置

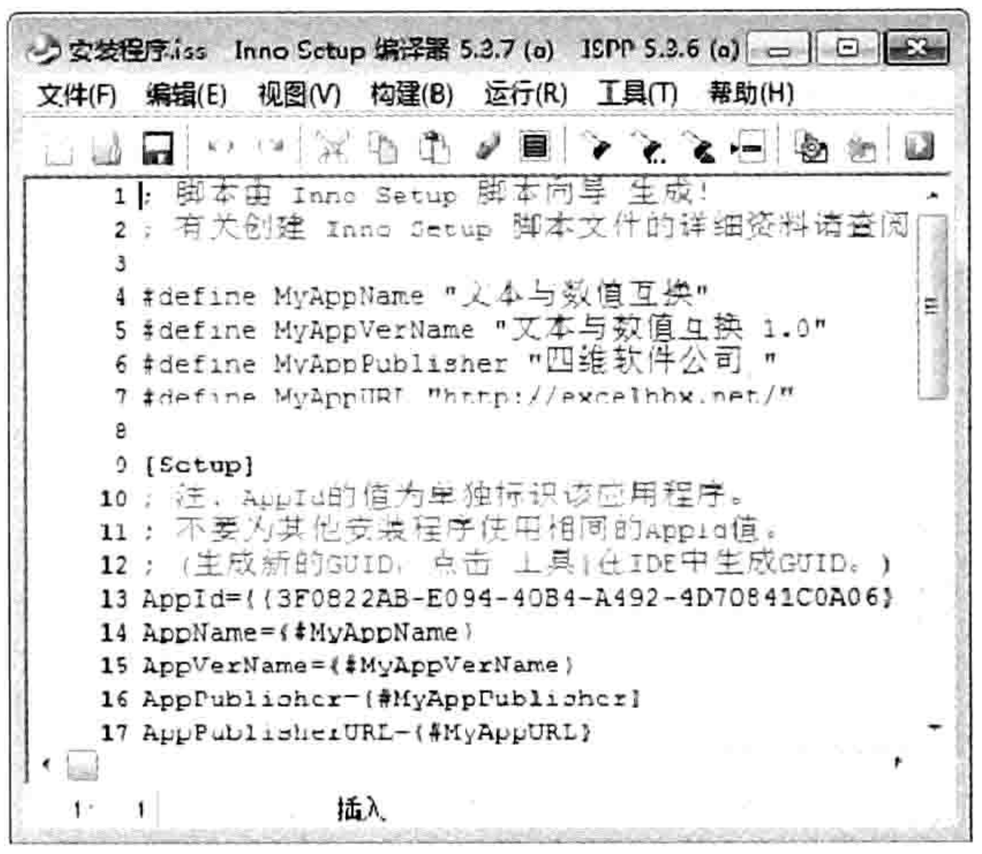


图 23.23　脚本文件代码

step 13　以上代码还不可以立即编译，因为它只能实现安装时复制文件到对应的文件夹，并创建快捷方式，还无法实现注册 DLL 文件。所以还需要在“[Files]”字段中的 ignoreversion 后面添加一个空格，然后再追加 regserver。完整代码如下：

```
Source: "D:\安装程序\文本与数值互换.dll"; DestDir: "{app}"; Flags:
ignoreversionregserver
```

此代码表示将“文本与数值互换.dll”文件复制到应用程序对应的文件夹中，并且执行注册。参数“{app}”代表脚本向导中“应用程序文件夹”对话框中选定的应用程序目录；参数 regserver 表示执行注册，相当于运行 regsvr32 命令。

由于“安装程序.iss”和“文本与数值互换.dll”文件在同一个文件夹中，因此代码中的路径可

以删除，即“D:\安装程序\文本与数值互换.dll"”可以修改为“文本与数值互换.dll"”。同理，“许可.txt”、“信息.txt”和“图标.ICO”的路径都可以删除。

对于命令“OutputDir=D:\安装程序”也应该修改，避免该文件夹被改名或者文件移动位置后生成的 EXE 文件仍然保存在“D:\安装程序”路径中，最理想的办法是将该路径改为当前文件名称。Inno Setup 将小数点定义为当前文件夹，因此为了提升代码的通用性，该句代码应修改为“OutputDir=.”。

step 14 单击菜单中的“构建”→“编译”命令，Inno Setup 会立即生成一个名为“setup.exe”的安装程序，图 23.24 中新产生的文件“setup.exe”即为安装程序，该文件可以直接在任何安装 Windows 系统的计算机中安装。

图 23.24　编译代码所产生的 Setup 文件

23.3.3　测试安装软件

在 23.3.2 节中通过向导生成的代码，编译代码后生成了可执行程序，不过该程序是否能正常工作还需要测试。

测试步骤比较简单，双击“setup.exe”图标从而启动安装界面，然后按提示操作即可。详细过程不再叙述，如图 23.25 至图 23.29 所示的 5 个截图是安装过程的操作界面，通过这 5 个截图读者可以了解 Inno Setup 的脚本向导中每一步所实现的效果，以及脚本向导与软件安装界面的对应关系。

图 23.30 是安装成功后在 Windows 的“开始”菜单中产生的快捷方式。

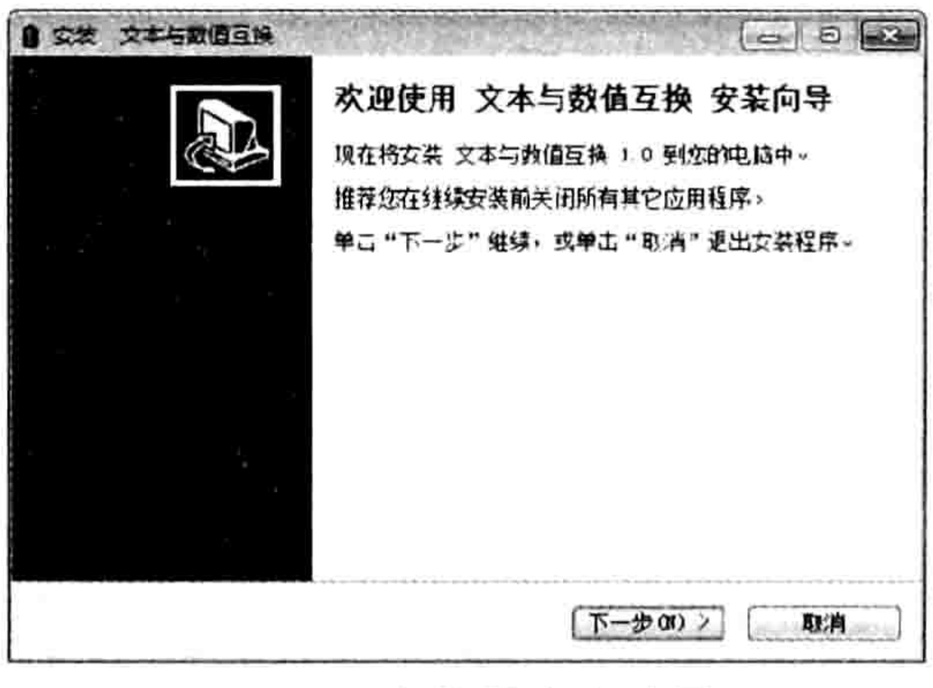


图 23.25　安装程序启动界面

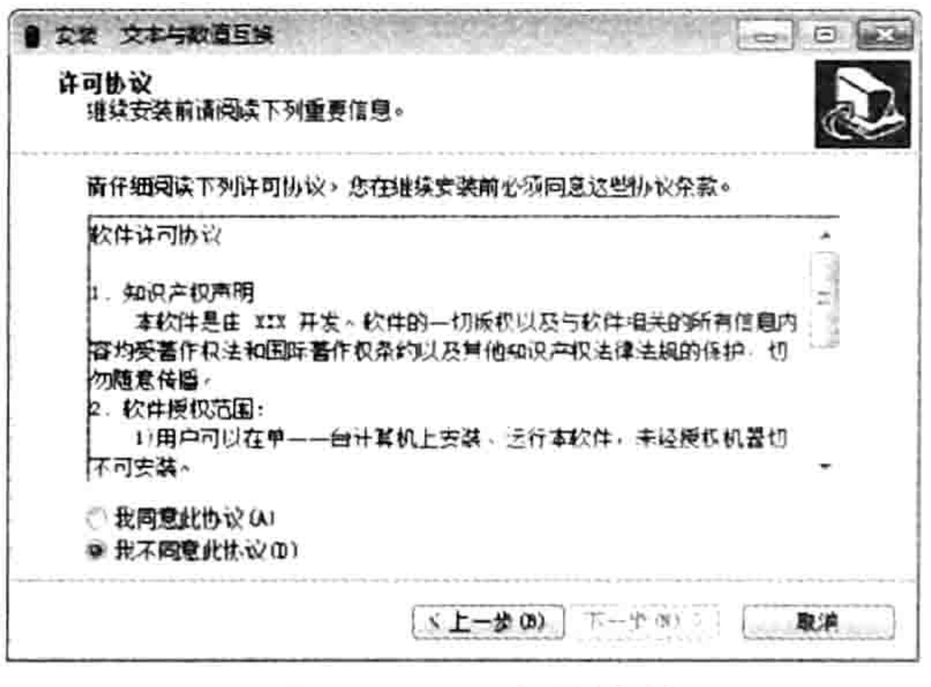


图 23.26　许可协议

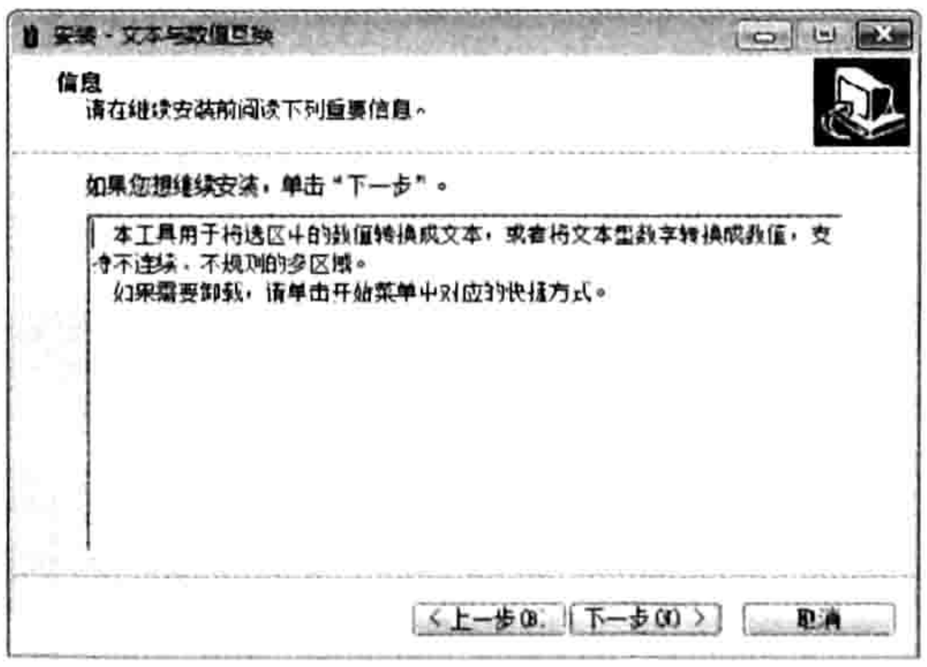


图 23.27　提示信息

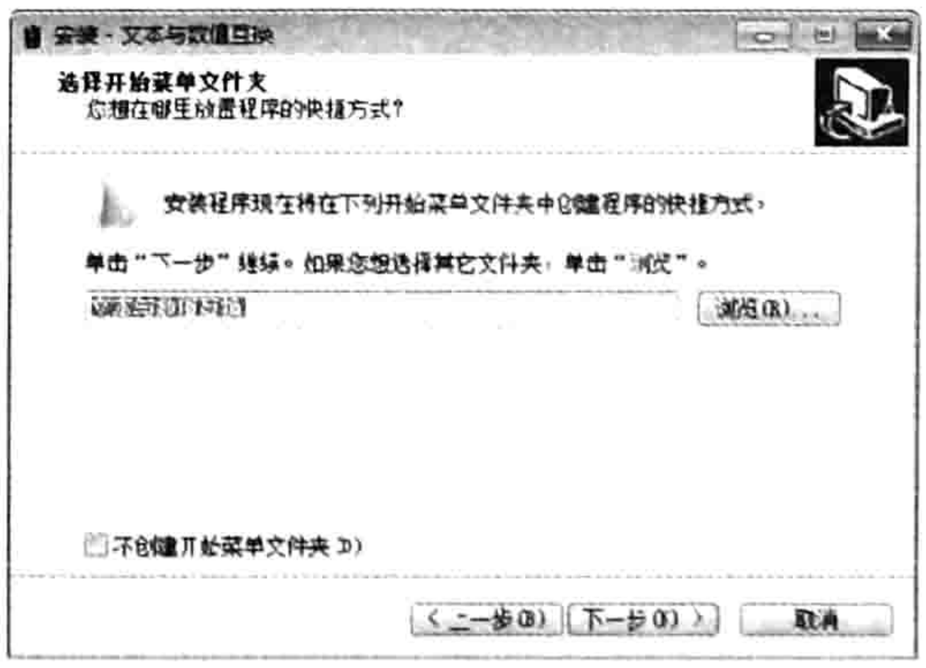


图 23.28　选择安装路径

图 23.29　完成安装

图 23.30　“开始”菜单中的快捷方式

CD-ROM　本例所有文件参见光盘：..\第二十三章\安装程序\

事实上 Inno Setup 软件还有很多参数，通过这些参数可以制作更强大也更美观的安装程序。由于本书篇幅有限不再一一罗列，有兴趣的读者可以加入本书的售后服务群，在群中与本书作者交流。

◆　◆　◆

第 24 章　开发逐步提示的数据录入助手

在录入比较长的地名、产品名称之类的字符串时，如果像百度的搜索栏一样每录入一个字符就自动产生相应的提示可以大大提高录入的速度和准确率，而将多个窗体控件的事件组合起来可以实现此等功能。

本章从设计窗体开始，到编写窗体事件、类模块中的应用程序事件、生成功能区菜单，以及发布加载宏，一步步展示数据录入助手的开发过程。本章不仅教授 Excel 插件的开发思路，也为每位读者提供了一个高效的 Excel 辅助工具。

24.1　罗列需求

和作文章前写提纲避免跑题一样，开发插件之前也应该罗列插件的一切需求，从而避免开发过程中漏掉某些功能。

24.1.1　插件功能描述

本插件为数据录入助手，专为录入数据提供便利而存在。它需要具备以下功能。

（1）提供一个对话框让用户指定数据源，数据源只能是一列。

（2）当用户指定的区域中没有数据时要提示用户，当用户指定的区域中有太多的空白单元格时需要自动忽略空白单元格。

（3）数据录入助手窗口只能在指定的列中出现，当活动单元格不在该列时自动隐藏窗口。

（4）让用户自行决定数据录入助手窗口产生在哪一列。

（5）允许从中间匹配关键词，也允许从左边匹配关键词，具体决定权由用户决定。例如输入“天信”时可以匹配“上海天信公司”属于从中间匹配；而输入“天信”只能匹配“天信公司”不能匹配“上海天信公司”则属于从左边匹配。

（6）在数据录入助手窗口中输入字符时，能实时地将数据源中匹配成功的字符串罗列在列表框中，而且在继续录入字符时能立即更新匹配结果。

（7）全程操作使用键盘完成，不需要使用鼠标来切换，从而确保录入工作的效率。

（8）当录入过程中发现数据源中没有目标时要提示用户是否将当前新词条追加到数据源中。

（9）数据录入助手的窗口的高度随匹配对象的多少而自动调整，而宽度则允许用户手工调整，从而适应匹配对象的数量变化及字符串长度。

（10）安装插件后对所有工作簿都生效，而非局限于代码所在的工作簿。

（11）代码需要有记忆功能，当工作簿被第二次打开时会记住上一次所设置的数据源地址，以及数据录入助手在哪一列出现、采用什么匹配方式，以及窗体的宽度是多少。

24.1.2　插件格式需求

插件的常用格式有 xla、xlam 和 DLL，其中 xla 格式兼容 Excel 2003、Excel 2007、Excel 2010 和 Excel 2013，xlam 格式兼容 Excel 2007、Excel 2010 和 Excel 2013，DLL 格式属于封装后的文件格式，具有保密性，不过设计过程和安装过程更麻烦一些。本章的数据录入助手要求采用 xlam 格式。

如果读者要将本章的插件改为 xla 格式也可以，删除生成功能区的 XML 代码，改用传统菜单

替代功能区，然后再将文件另存为 xla 格式。

24.2　设计窗体

Excel 没有单元格的 KeyDown 事件、KeyPress 事件和 KeyUp 事件，因此无法在单元格中每录入一个字符就触发一次事件。而窗体中的文本框控件拥有需要的一切事件，因此开发数据录入助手的重点在于设计窗体，以及编写窗体控件的事件过程。

24.2.1　设计选项窗体

本章要开发的数据录入助手工具是一个通用工具，每一个 Excel 用户都可能需要用到这个功能。由于不同用户会采用不同的使用方式，甚至同一个用户在不同工作簿中使用时也会有不同的使用方式，因此必须为工具设计一个选项对话框，让用户根据需求预设应用条件。

设计选项窗体的步骤如下。

step 1　新建一个工作簿，按<Alt+F11>组合键打开 VBE 窗口。

step 2　单击菜单中的“插入”→“用户窗体”命令，然后在属性对话框中将名称属性修改为“OptionForm”，将 Caption 属性修改为“选项”。

step 3　在窗体中添加一个标签控件，并在属性窗口中将它的 Caption 属性修改为“请指定数据源（只能是单列的区域）”。

step 4　在标签下方添加一个文本框，在文本框的右方末端添加一个命令按钮，并且将命令按钮的 Caption 属性修改为“—”，效果如图 24.1 所示。

step 5　在文本框下方添加两个标签控件，将它们的 Caption 属性分别设置为“在”和“列输入字符时逐步提示，其他列则忽略”。

step 6　在两个标签控件中间添加一个复合框控件，在复合框控件下方添加一个复选框控件，并且将它的 Caption 属性修改为“允许从中间匹配”。

step 7　在复选框控件下添加一个标签控件，并且将它的 Caption 属性修改为“例如用「螺」可以匹配「螺丝」也可以匹配「六角螺丝」...”。

step 8　在窗体的右下角添加两个命令按钮，将 Caption 属性分别修改为“确定”和“取消”，然后再将“取消”按钮的 Cancel 属性修改为“True”，表示在窗体中按<Esc>键时可以执行此命令按钮的 Click 事件和 Enter 事件，从而关闭窗体。如图 24.2 所示是“选项”窗体的最终效果。

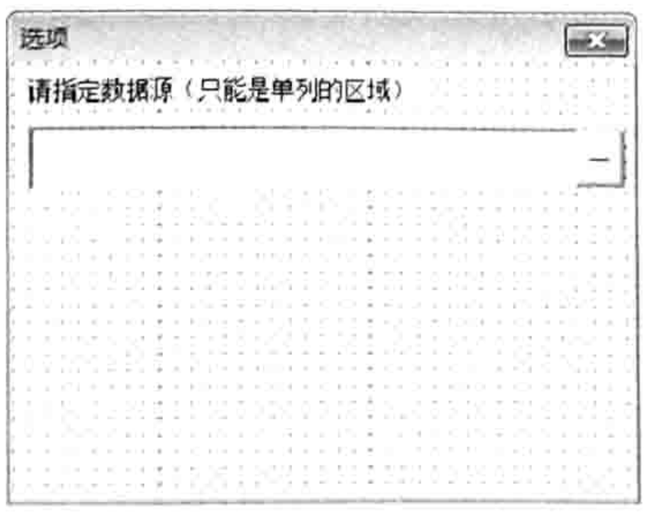


图 24.1　将命令按钮放置在文本框末端

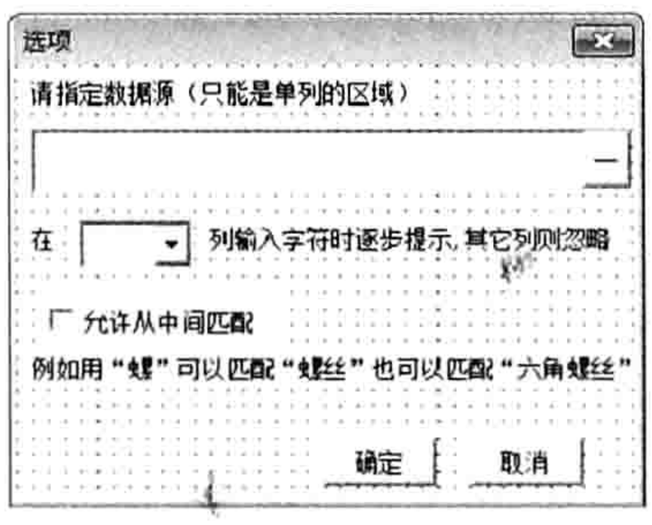


图 24.2　选项窗体的控件布局方式

24.2.2　设计数据录入助手窗体

数据录入助手窗体的功能强大而复杂，但是它的控件布局比较简单。具体设计步骤如下。

step 1 单击菜单中的“插入”→“用户窗体”命令，然后将窗体的名称属性修改为“HelperForm”，将窗体的 Caption 属性修改为“数据录入助手”。

step 2 将工具箱中的旋转按钮拖曳到窗体的左上角，然后修改它的大小使其两个箭头呈左右分布。此旋转按钮的功能是修改窗体的宽度，而高度则由代码自行修改，不需要用户手工调整。

step 3 在旋转按钮右方添加一个标签控件，将它的 Caption 属性修改为“请输入数据”。

step 4 在标签右边添加一个文本框，在下方添加一个列表框，最终效果如图 24.3 所示。

图 24.3 数据录入助手窗体的控制布局方式

24.3 编写代码

数据录入助手的代码比较复杂，涉及了 VBA 中近百个知识点。

为使数据录入助手能应用于所有工作簿，需要将数据录入助手加工成加载宏。在加载宏中，除了对两个窗体编写代码以外还涉及模块与类模块的代码，其中“数据录入助手”窗体中的各种事件过程代码最为复杂。

24.3.1 选项窗体代码

选项窗体的存在价值在于指定数据录入助手的执行方式，其编程思路是在模块中声明公共变量，然后在选项窗体中对公共变量赋值，最后数据录入助手读取到公共变量的值，并根据变量的值决定采用何种方式去匹配目标。

由于两个窗体都需要调用模块中的公共变量，因此在编写选项窗体的代码的同时还需要在模块中声明变量。具体操作步骤如下。

step 1 单击菜单中的“插入”→“模块”命令，然后在模块中录入以下声明公共变量的代码：

```
Public TiShiRng As Range
Public CellAddress As String
Public TiShiArr
Public ZhongJian As Boolean
Public TiShiCol As Integer
Public WidthLong
```

step 2 在工程资源管理器中对窗体 OptionForm 单击鼠标右键，在弹出的快捷菜单中选择“查看代码”命令，然后在代码窗口中录入以下 4 个事件过程：

```
Private Sub UserForm_Activate()'①代码存放位置：窗体中②随书光盘中有每一句代码的含义注释
  For i = 1 To 256
    ComboBox1.AddItem Split(Cells(1, i).Address, "$")(1)
```

```
  Next i
  ComboBox1.Text = Split(Cells(1, ActiveCell.Column).Address, "$")(1)
  If Len(CellAddress) > 0 And Not TiShiRng Is Nothing Then
    TextBox1.Text = TiShiRng.Parent.Name & "!" & TiShiRng.Address(0, 0)
  End If
End Sub
```

此过程首先向复合框中添加 1～256 列的列标，即字母 A～IV，并且让复合框默认显示为活动列的列标。然后判断公共变量 CellAddress 和 TiShiRng 是否已经被赋值，如果被赋值则将 CellAddress 和 TiShiRng 的父对象的名称显示在文本框中。其中变量 TiShiRng 代表数据源，是一个 Range 对象，变量 CellAddress 代表数据源的地址，是一个字符串。

```
'①代码存放位置：窗体的代码窗口中②随书光盘中有每一句代码的含义注释
Private Sub CommandButton1_Click()
  Dim T As Long, H As Long
  T = Me.Top: H = Me.Height
  Me.Top = -H
  On Error Resume Next
  Set TiShiRng = Application.InputBox("请指定数据区域，只能是单列的区域", "确定数据源", , , , , , 8)
  If Err <> 0 Then
    CellAddress = ""
  Else
    If TiShiRng.Columns.Count > 1 Then MsgBox "只能选择单列的区域", vbOKOnly + vbInformation, "友情提示": Me.Top = T: Exit Sub
    If WorksheetFunction.CountA(TiShiRng) <= 2 Then MsgBox "您的数据源中数据太少，请补充数据", vbOKOnly + vbInformation, "友情提示": Me.Top = T: Exit Sub
    Set TiShiRng = TiShiRng.Parent.Range(TiShiRng(1), TiShiRng.Parent.Cells(Rows.Count, TiShiRng.Column).End(xlUp))
    CellAddress = TiShiRng.Address
    TextBox1.Text = TiShiRng.Parent.Name & "!" & TiShiRng.Address(0, 0)
    SaveSetting ActiveWorkbook.Name, ActiveSheet.Name, "CellAddress", TiShiRng.Parent.Name & "!" & TiShiRng.Address(0, 0)
  End If
  Me.Top = T
End Sub
```

此过程用于向用户提供一个选择区域的对话框，然后将用户选择的区域赋值给变量 TiShiRng，并将该区域的地址记录在变量 CellAddress 中，以及显示在文本框中、保存在注册表中，便于下次调用。

其中变量 T 和变量 H 的作用是配合窗体的 Top 属性隐藏选项窗体。在选项窗体中单击按钮 CommandButton1 弹出选择区域的对话框时无法手工移动选项对话框，因此本例在 CommandButton1_Click 事件过程中通过修改窗体 Top 属性的方式将窗体移到屏幕之外，从而实现隐藏窗体的目的，当用户选择区域完毕后再让窗体恢复到原来的位置。

```
'①代码存放位置：窗体的代码窗口中②随书光盘中有每一句代码的含义注释
Private Sub CommandButton2_Click()
  ZhongJian = CheckBox1.Value
  TiShiCol = Cells(1, ComboBox1.Text).Column
  SaveSetting ActiveWorkbook.Name, ActiveSheet.Name, "TiShiCol", Cells(1, ComboBox1.Text).Column
  SaveSetting ActiveWorkbook.Name, ActiveSheet.Name, "ZhongJian", ZhongJian
  TiShiArr = TiShiRng.Value
```

```
  Unload Me
End Sub
```

此过程表示单击 CommandButton2 按钮时先用公共变量 ZhongJian 记录复选框的状态，然后用公共变量 TiShiCol 记录复合框中指定的列号，接着将这两个数据通过 SaveSetting 语句保存在注册表中，便于下次打开工作簿时调用。最后将数据源的值读取到数组变量 TiShiArr 中，并关闭窗体。

```
'①代码存放位置：窗体的代码窗口中②随书光盘中有每一句代码的含义注释
Private Sub CommandButton3_Click()
  CellAddress = ""
  Unload Me
End Sub
```

此过程是单击“取消”按钮时执行的过程，因此在过程中先将变量 CellAddress 还原为空文本，表示禁用数据录入助手，然后关闭窗体。

关于选项窗体的功能，可以简单地概括为——让用户选择数据源、匹配方式和在哪一列产生数据录入助手，同时将这些设置信息保存在注册表中，便于下次调用。

24.3.2 数据录入助手窗体代码

数据录入助手窗体的代码比较长，也比较复杂。完整代码如下：

```
'①代码存放位置：窗体的代码窗口中②随书光盘中有每一句代码的含义注释
Private Sub UserForm_QueryClose(Cancel As Integer, CloseMode As Integer)
  WidthLong = Me.Width
  SaveSetting ActiveWorkbook.Name, ActiveSheet.Name, "Width", WidthLong
End Sub
```

此过程表示关闭窗体时将窗体的宽度记录在变量 WidthLong 中，而且将变量的值保存在注册表中，便于下次运行窗体时调用该值。

```
'①代码存放位置：窗体的代码窗口中②随书光盘中有每一句代码的含义注释
Private Sub UserForm_Initialize()
  Me.Width = WidthLong
End Sub
```

此过程表示初始化窗体时调用变量 WidthLong 的值来调整窗体的宽度，从而确保窗体和上一次关闭前的宽度一致。

```
'①代码存放位置：窗体的代码窗口中②随书光盘中有每一句代码的含义注释
Private Sub UserForm_Activate()
  TextBox1.SetFocus
  TextBox1.IMEMode = fmIMEModeHanzi
  ListBox1.Height = 114
  SpinButton1.Max = 1
  SpinButton1.Min = -1
  SpinButton1.SmallChange = 1
  SpinButton1.Value = 0
  SpinButton1.TabStop = False
End Sub
```

此过程表示激活窗体时先将焦点转移到文本框中，并且将文本框的输入法调整为中文输入法。然后将列表框的高度设置为默认值 114，将旋转按钮的最大值设置为 1，最小值设置为-1，步长值设置为 1，当前设置为 0。最后将旋转按钮的 TabStop 属性赋值为 False，表示不接受焦点，当用户从列表框中按<Tab>键时可以直接跳转到文本框中，而不是跳转到旋转按钮。

```
'①代码存放位置：窗体的代码窗口中②随书光盘中有每一句代码的含义注释
Private Sub UserForm_Resize()
  If Me.Width <150 Then
    Me.Width = 150
  Else
    ListBox1.Width = Me.Width - 24
    TextBox1.Width = Me.Width - 91
  End If
End Sub
```

此过程表示修改窗体的大小时将列表框的宽度设为窗体的宽度减 24，让文本框的宽度设为窗体的宽度减 91，而且将窗体的最小宽度值设置为 150，避免窗体中的文本框和列表框太窄从而影响工作。

```
'①代码存放位置：窗体的代码窗口中②随书光盘中有每一句代码的含义注释
Private Sub SpinButton1_Change()
  Me.Width = Me.Width + SpinButton1.Value
  SpinButton1.Value = 0
End Sub
```

此过程表示单击旋转按钮时修改窗体的宽度，每单击一次旋转按钮就可以让窗体的宽度增加 1 或者减小 1。可以按住旋转按钮不松开，从而连续增加或者缩小窗体的宽度。

```
'①代码存放位置：窗体的代码窗口中②随书光盘中有每一句代码的含义注释
Private Sub TextBox1_KeyDown(ByVal KeyCode As MSForms.ReturnInteger, ByVal
Shift As Integer)
  On Error Resume Next
  If KeyCode = 27 Then
    Unload Me
  ElseIf KeyCode = 37 Then
    ActiveCell.Offset(0, -1).Activate
  ElseIf KeyCode = 38 Then
    ActiveCell.Offset(-1, 0).Activate
  ElseIf KeyCode = 39 Then
    ActiveCell.Offset(0, 1).Activate
  ElseIf KeyCode = 40 Then
    If Len(Me.TextBox1) = 0 Then ActiveCell.Offset(1, 0).Activate
  End If
End Sub
```

此过程是在文本框中按下任意键时触发的事件过程，本过程中利用参数 KeyCode 来判断用户按下了哪些键，其中 27 代表<Esc>键，因此使用代码 Unload Me 关闭窗体；37 代表左箭头，因此将活动单元格左移一格；38 代表上箭头，因此将活动单元格上移一格；39 代表右箭头，因此将活动单元格右移一格；40 代表下箭头，此时不能直接下移一格，而是文本框中没有字符时才下移一格，如果文本框中有字符会将窗体的焦点切换到列表框中，从而执行列表框的 Enter 事件，此时不需要下移活动单元格。

```
'①代码存放位置：窗体的代码窗口中②随书光盘中有每一句代码的含义注释
Private Sub Textbox1_KeyUp(ByVal KeyCode As MSForms.ReturnInteger, ByVal Shift
As Integer)
  Application.ScreenUpdating = False
  Dim i As Long, Mystr As String, arr(), j As Long, ListBox_H As Long
  With TiShiRng.Parent
    Mystr = LCase(TextBox1.Text)
```

```
    If Len(Mystr) > 0 Then
     If ZhongJian Then
      For i = 1 To UBound(TiShiArr)
       If InStr(TiShiArr(i, 1), Mystr) Then
        j = j + 1
        ReDim Preserve arr(1 To j)
        arr(j) = TiShiArr(i, 1)
       End If
      Next i
     Else
      For i = 1 To UBound(TiShiArr)
       If TiShiArr(i, 1) Like Mystr & "*" Then
        j = j + 1
        ReDim Preserve arr(1 To j)
        arr(j) = TiShiArr(i, 1)
       End If
      Next i
     End If
     If j > 0 Then ListBox1.List = arr Else Me.ListBox1.Clear
     ListBox_H = ListBox1.Font.Size * j + 5
     Me.Height = ListBox_H + 60
     ListBox1.Height = ListBox_H
     Me.Caption = "数据录入助手(有" & j & "个目标)"
    Else
     ListBox1.Clear
     Me.Caption = "数据录入助手"
     ListBox1.Height = 114
     Me.Height = 180
    End If
   End With
   Application.ScreenUpdating = True
  End Sub
```

此过程是在文本框中按下任意键并且弹起来时触发的事件过程，过程中首先将文本框中录入的字符串转换成大写，并赋值给变量 Mystr。其中 LCase 函数不是单独使用，在后面的循环语句中还会再次使用 LCase 函数将数组中的元素转换成大写，从而使判断结果不区分大小写。例如用户录入的字符是 a，而数组中的字符串是“Ace”，两者都通过 LCase 函数转换成大写，那么使用 Like 运算符或者 Instr 函数进行比较时都能匹配成功，否则会失败。

其次，事件过程判断文本框中是否有值（按<Delete>键并弹起时可能导致文本框中无值），以及公共变量 ZhongJian 的值是否为 True，当 ZhongJian 的值为 True 时使用 Instr 函数配合循环语句逐一计算数组 TiShiArr 的每个元素是否包含文本框的值，如果包含则将它追加到数组 Arr 中；如果公共变量 ZhongJian 的值是 False，那么使用 Like 语句配合循环语句逐一比较数组 TiShiArr 中的每个元素是否匹配文本框中的值（从左边匹配），如果匹配成功则将它追加到数组 Arr 中。

当循环完成后，如果有匹配成功的值则将数组 arr 赋予列表框的 List 属性，从而将 Arr 的值显示到列表框中，然后调整窗体与列表框的高度，并在窗体的标题栏中显示匹配成功的数量；如果没有匹配成功的值，那么清空列表框中上一次遗留下来的匹配对象。

最后根据列表框的字体大小和列表框项目数量计算列表框的高度和窗体高度，并在窗体的标题栏文字中记录当前匹配成功的数量。

```
'①代码存放位置：窗体的代码窗口中②随书光盘中有每一句代码的含义注释
Private Sub ListBox1_Enter()
  If Len(TextBox1) = 0 Then Exit Sub
```

```
  If ListBox1.ListCount > 0 Then
    TextBox1.Text = ListBox1.List(0)
  Else
    Dim Msg As VbMsgBoxResult
    If MsgBox("发现新词汇，需要添加到管理器吗？", vbYesNo + vbQuestion,
TextBox1.Text) = vbYes Then
      TiShiRng.Parent.Cells(Rows.Count, TiShiRng.Column).End(xlUp).Offset(1,
0) = TextBox1
      ActiveWorkbook.Save
      Set TiShiRng = Union(TiShiRng, TiShiRng.Offset(1, 0))
      TiShiArr = TiShiRng.Value
    End If
  End If
End Sub
```

此过程是列表框接受焦点时触发的事件过程。当在文本框中录入字符后，按<Tab>键或者<↓>键后焦点会从文本框转移到列表框中，此时需要判断文本框中是否有值，如果文本框中无值则直接结束过程；如果文本框中有值，那么还需要判断列表框中是否有值，当列表框中有值时直接将列表框的第一行数据显示在文本框中；当列表框中无值时表示当前录入的是新词汇，因此使用Msgbox 函数弹出一个对话框让用户决定是否将新词汇追加到数据中。

当用户单击“是”按钮时，将文本框的值追加到数据源的下一行中，同时保存工作簿、更新公共变量 Set TiShiRng 和 TiShiArr。

```
Private Sub ListBox1_Click()'①代码存放位置：窗体中②随书光盘中有每一句代码的含义注释
  TextBox1.Text = ListBox1.Text
End Sub
```

此过程表示单击列表框时将列表框的值显示在文本框中。所谓的单击并非仅限于按<Enter>键，按<↑>、<↓>键都会触发 ListBox1_Click 事件。换而言之，使用<↑>、<↓>键在列表框中上翻和下翻时，将当前选中的值显示在文本框中。

```
'①代码存放位置：窗体的代码窗口中②随书光盘中有每一句代码的含义注释
Private Sub ListBox1_KeyPress(ByVal KeyAscii As MSForms.ReturnInteger)
  On Error Resume Next
  If KeyAscii = 27 Then
    Unload Me
  ElseIf KeyAscii = 49 Then
    TextBox1 = ListBox1.List(0)
  ElseIf KeyAscii = 50 Then
    TextBox1 = ListBox1.List(1)
  ElseIf KeyAscii = 51 Then
    TextBox1 = ListBox1.List(2)
  ElseIf KeyAscii = 52 Then
    TextBox1 = ListBox1.List(3)
  ElseIf KeyAscii = 53 Then
    TextBox1 = ListBox1.List(4)
  ElseIf KeyAscii = 54 Then
    TextBox1 = ListBox1.List(5)
  ElseIf KeyAscii = 55 Then
    TextBox1 = ListBox1.List(6)
  ElseIf KeyAscii = 56 Then
    TextBox1 = ListBox1.List(7)
  ElseIf KeyAscii = 57 Then
    TextBox1 = ListBox1.List(8)
```

```
  Else
    Exit Sub
  End If
  If Err = 0 Then
    ListBox1.Clear
    TextBox1.SetFocus
  Else
    Err.Clear
  End If
End Sub
```

此过程是在列表框中按下任意键时触发的事件过程。在过程中使用条件语句根据参数 KeyAscii 的返回值决定操作方式，当 KeyAscii 的值为 27 时，表示用户按了<Esc>键，那么关闭窗体；当 KeyAscii 的值是 49 时，表示用户按了数值键<1>，那么将列表框的第 1 行内容显示在文本框中；当 KeyAscii 的值是 50 时，表示用户按了数值键<2>，那么将列表框的第 2 行内容显示在文本框中……如果用户按下的键不是<1>～<9>，那么直接结束过程。

如果用户在列表框中按了<1>～<9>键且没有出错时，那么清除列表框的值，然后将焦点转移到文本框中。

简而言之，本过程的目的在于为 ListBox1_Click 事件过程提供快捷方式。ListBox1_Click 事件过程的目的是在列表框中按<↑>箭头或者<↓>箭头时将当前选中的值显示在文本框中，而本过程则是让用户通过数字键来快捷地定位，例如按下<8>键就将第 8 行的值显示在文本框中，而不用按 7 次<↓>键。

```
Private Sub TextBox1_Enter()'①代码存放位置：窗体中②随书光盘中有每一句代码的含义注释
  If Len(TextBox1.Text) > 0 Then
    ActiveCell = TextBox1.Text
    ActiveCell.Offset(1, 0).Activate
    TextBox1.Text = ""
    ListBox1.Clear
  End If
  ListBox1.Height = 114
  Me.Height = 180
End Sub
```

此过程是文本框接受焦点时触发的事件过程。当在列表框中按<Tab>键时会先触发 ListBox1_KeyPress 事件，该事件过程将列表框中选中的值显示在文本框中，接着由于焦点从列表框转移到文本框，从而触发 TextBox1_Enter 事件。

在 TextBox1_Enter 事件中，首先利用 Len 函数判断文本框中是否有值，如果有则将文本框的值输出到活动单元格中，然后将活动单元格下移，接着清空文本框和列表框的值，将窗体的宽度和高度恢复到默认状态。

以上关于文本框和列表框的几个事件过程比较复杂，现将它们按顺序整理为表格形式，帮助读者理解，如表 24-1 所示。

表 24-1　文本框与列表框事件

事件名称	触发条件	功能
TextBox1_KeyDown	在文本框中按键时触发	判断用户是否按了方向键以及取消键。如果按了取消键则关闭窗体，如果按方向键则根据方向移动活动单元格

续表

事件名称	触发条件	功能
Textbox1_KeyUp	在文本框中按键后弹起来时触发	通过循环语句将数组 TiShiArr 中的每个元素与文本框中的值执行比较，将匹配成功的字符串显示在列表框中
ListBox1_Enter	列表框获点焦点时触发	在文本框中按<tab>键或者<↓>键时将焦点转移到列表框中，此时列表框的第 1 行值会显示在文本框中
ListBox1_Click	单击列表框时触发	使用<↑>、<↓>键在列表框中上翻和下翻时，将当前选中的值显示在文本框中
ListBox1_KeyPress	在列表框中按键时触发	判断用户是否按了<0>~<9>的数字键，以及取消键，如果是取消键则关闭窗体，如果是<0>~<9>的数字键则将列表框中对应的行显示在文本框中
TextBox1_Enter	文本框获得焦点时触发	在列表框中按<Tab>键时，焦点从列表框转移到文本框中，此时将文本框中的值输出到活动单元格，然后清空文本框、列表框，并将活动单元格下移一行

24.3.3　应用程序级事件代码

本章的最终目的是制作一个通用的加载宏，让代码可应用于所有工作簿，因此启动数据录入助手的代码应放在应用程序级的事件过程中，而不能放在工作簿事件中。

设计应用程序级事件过程的步骤如下。

step 1　双击工程资源管理器中的模块 1，然后在模块顶部录入以下代码：

```
Dim ClassOne As MyClass
Dim rib As IRibbonUI
Sub Auto_open()'①代码存放位置：模块中②随书光盘中有每一句代码的含义注释
    Set ClassOne = New MyClass
End Sub
```

此代码的目的是用于触发类的 Initialize 事件。

step 2　单击菜单中的“插入”→“类模块”命令，将类模块的名称修改为“MyClass”。

step 3　在类模块中录入以下代码：

```
Dim WithEvents myApp As Application
Private Sub Class_Initialize()'①代码存放位置：类模块中②随书光盘中有每一句代码的含义注释
    Set myApp = Application
End Sub
```

以上过程表示将 Excel 应用程序赋值给变量 myApp，使后面的 myApp_SheetSelectionChange 和 myApp_WorkbookActivate 事件过程生效。

```
'①代码存放位置：窗体的代码窗口中②随书光盘中有每一句代码的含义注释
```

```
Private Sub myApp_SheetSelectionChange(ByVal Sh As Object, ByVal Target As
Range)
  If TiShibl = True And CellAddress <> "" Then
    If Target.Column = TiShiCol Then
      HelperForm.Show 0
    Else
      HelperForm.Hide
    End If
  End If
End Sub
```

此过程是应用程序级的事件过程，选择任意工作簿的任意工作表中的任意单元格时触发。

过程中首先判断公共变量 TiShibl 是否等于 True、变量 CellAddress 是否等于空值，如果符合条件再检查活动单元格所在列的列号是否等于变量 TiShiCol 的值，如果 3 个条件都符合则显示数据录入助手窗口，否则隐藏数据录入助手窗口。

```
'①代码存放位置：窗体的代码窗口中②随书光盘中有每一句代码的含义注释
Private Sub myApp_WorkbookActivate(ByVal Wb As Workbook)
  WidthLong = GetSetting(Wb.Name, ActiveSheet.Name, "Width", 217)
  CellAddress = GetSetting(Wb.Name, ActiveSheet.Name, "CellAddress", "")
  TiShiCol = GetSetting(Wb.Name, ActiveSheet.Name, "TiShiCol", 0)
  ZhongJian = GetSetting(Wb.Name, ActiveSheet.Name, "ZhongJian", False)
  If Len(CellAddress) > 0 Then
    On Error Resume Next
    Set TiShiRng = Range(CellAddress)
    If Err = 0 Then TiShiArr = TiShiRng.Value
  End If
End Sub
```

此过程是应用程序级的事件过程，激活新工作簿时触发此事件。

事件过程的功能是激活新工作簿时通过 GetSetting 函数从注册表中读取活动工作簿相关的窗体宽度值、数据源地址、要产生数据录入助手的列号，以及是否从中间开始匹配的选项值。假设活动工作簿在以前某个时段曾经设置过这些参数值，打开工作簿时会自动调用这些参数，从而让数据录入助手立即进入工作状态，不再需要打开选项窗体手工设置参数。

24.4 创建功能区菜单

数据录入助手属于 Excel 插件，由于加载宏中的 Sub 过程不能通过按<Alt+F8>组合键调用，因此必须使用 CustomUIEditor 软件为其制作功能区菜单。

24.4.1 创建功能区菜单

数据录入助手需要 3 个菜单，一个用于启动选项窗体，一个用于控制数据录入助手的启动与隐藏，一个用于弹出信息框从而说明本工具的用途。

功能菜单的设计步骤如下。

step 1 打开 CustomUIEditor 软件，并按<Ctrl+O>组合键打开刚才所保存的工作簿。

step 2 单击菜单中的“Insert”→“Office 2007 Custom UI Part”命令从而插入一个 customUI.xml 文件，然后在代码窗口中录入以下代码：

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui" onLoad=
"Intialize">
```

```
  <ribbon>
    <tabs>
      <tab idMso="TabHome">
        <group id="Group1"  label="数据录入助手" insertBeforeMso="GroupFont">
          <button  id="button1"  label=" 选 项 &#13;"   imageMso="WordCountList"
size= "large" onAction="ActionA"  supertip="单击产生提示，再单击不提示"/>
          <button   id="button2"   getLabel="getLabel"     getImage="getImage"
size="large" onAction="ActionB" supertip="设置选项" />
          <button  id="button3"  label=" 关 于 &#13;"   imageMso="FunctionsLogical
InsertGallery"  size="large" onAction="ActionC" supertip="功能描述" />
        </group>
      </tab>
    </tabs>
  </ribbon>
</customUI>
```

以上代码表示在“开始”选项卡中创建一个名为“数据录入助手”的新组，并在该组中放置 3 个按钮。代码中的“
”代表换行符，按钮的名称末尾添加换行符可以让菜单名称不换行。例如菜单名称是“选项”，那么显示在功能区中的结果是“选”字和“项”字各占一行，而在“选项”二字后面追加一个换行符后，“选项”二字会显示在同一行中。

step 3　按<Ctrl+S>组合键保存代码，然后关闭 CustomUIEditor 软件。

24.4.2　回调过程

本例的 XML 代码涉及 6 个回调过程，读者可以单击 CustomUIEditor 软件中的“Generate Callbacks”按钮查看这 6 个回调函数的程序外壳。

编写回调过程的步骤如下。

step 1　用 Excel 打开前面刚才保存的工作簿，按<Alt+F11>组合键打开 VBE 窗口。

step 2　在工程资源管理器中双击“模块 1”，然后在模块顶部录入以下代码：

```
Public TiShibl As Boolean
```

此代码用于声明一个公共变量，变量的值决定了数据录入助手的窗体是否显示。

step 3　在模块底部继续录入以下代码：

```
Sub ActionA(control As IRibbonControl)'代码存放位置：模块中
  OptionForm.Show 0
End Sub
```

此过程对应于第 1 个菜单，用于显示选项窗体。

```
Sub ActionB(control As IRibbonControl)'代码存放位置：模块中
  TiShibl = Not TiShibl
  rib.Invalidate
End Sub
```

此过程对应于第 2 个菜单，用于切换公共变量 TiShibl 的值，使其在 True 与 False 之间变化。然后通过代码“rib.Invalidate”更新功能区，使第二个自定义菜单的图标和文字根据需示自动切换。

```
Sub ActionC(control As IRibbonControl)'代码存放位置：模块中
  MsgBox "使用此工具的第一步是打开「选项」对话框，指定数据源和要产生数据录入助手的列，然
后再单击「提示」菜单，使其显示为打钩的图标。最后，当激活指定列的单元格时会自动弹出数据录入
```

```
助手对话框。", vbInformation, "友情提示"
End Sub
```

此过程对应第3个自定义菜单，用于弹出信息框，告知用户本插件的功能和用法。

```
Sub getImage(control As IRibbonControl, ByRef returnedVal)'代码存放位置：模块中
  If TiShibl = True Then
    returnedVal = "AcceptInvitation"
  Else
    returnedVal = "DeclineInvitation"
  End If
End Sub
```

此过程用于切换第2个自定义菜单的图标，每单击一次就切换一次图标。当变量 TiShibl 的值为 true 时，对参数 returnedVal 赋值为 AcceptInvitation，表示显示为打钩的图标，否则赋值为 DeclineInvitation，表示显示为打叉的图标。

```
Sub getLabel(control As IRibbonControl, ByRef returnedVal)'代码存放位置：模块中
  If TiShibl = True Then
    returnedVal = "提示" & Chr(13)
  Else
    returnedVal = "不提示" & Chr(13)
  End If
End Sub
```

此过程用于切换第2个自定义菜单的文字，每单击一次切换一次。当变量 TiShibl 的值为 True 时，对参数 returnedVal 赋值为“提示”加换行符，否则赋值为“不提示”加换行符。

知识补充：VBA 中的换行符是 chr(13)，创建功能区组件的 XML 代码中的换行符是“
”。

step 4 保存工作簿，然后重启工作簿，在“开始”菜单处可以看到如图 24.4 所示的自定义菜单。

图 24.4 功能区菜单的外观

CD-ROM 本例文件参见光盘：..\第二十四章\数据录入助手.xlsm

24.5 发布插件与测试功能

开发任何插件后都需要经过反复测试，一切符合需求才发布。

不过包含了类模块的代码会有所不同，需要将工作簿转换成加载宏后再测试，否则部分代码不能正常工作。

24.5.1 发布插件

发布 xlam 格式的插件比较简单，和前面几个案例一样加密工程后直接用代码将它另存到自启

动路径中即可。操作步骤如下。

step 1　在 VBE 界面中单击菜单中的“工具”→“VBAProject 属性”→“保护”命令。

step 2　将“查看时锁定工程”选项打钩，然后在下方的“密码”和“确认密码”文本框中输入密码，然后单击“确定”按钮完成工程保护（光盘中的案例文件的密码为“andysky”）。

step 3　在模块中录入以下代码：

```
Sub 发布()'代码存放位置：模块中
  ThisWorkbook.IsAddin = True
  ThisWorkbook.SaveCopyAs Application.Path & "\XLSTART\数据录入助手.xlam"
End Sub
```

代码的含义是将代码所在工作簿转换成加载宏，然后另存到自启动路径下。

step 4　执行以上过程，然后关闭工作簿“数据录入助手.xlsm”。

step 5　删除工作簿“数据录入助手.xlsm”，随意打开一个工作簿，在“开始”选项卡中可以看到如图 24.4 所示的 3 个自定义菜单，表示插件已经安装成功。

24.5.2　测试插件功能

将插件投入使用前必须经过反复地测试，避免漏掉某些功能，或者在特殊情况下执行出错，或者执行程序时导致 Excel 崩溃。

测试插件要使用到以下两个工作表，图 24.5 中 A 列包含 68 个产品名称，图 24.6 是零售统计表，在该表的 B 列输入产品名称时需要借助数据录入助手来提升输入速度，同时也确保名称一致，不会因输入了同音字或者无意中多输入了一个空格而导致无法分类汇总。

	A
1	薄壁管电线管
2	焊接管电线管
3	镀锌管电线管
4	塑料管电线管
5	钢制槽式桥架
6	钢制梯式桥架
7	桥架支撑架
8	金属线槽
9	铜芯绝缘电线
10	铜芯阻燃绝缘电线

数据　零售统计表

图 24.5　数据源

	A	B	C	D
1	售货员	品名	数量	价格
2				
3				
4				
5				
6				
7				
8				
9				
10				

数据　零售统计表

图 24.6　要借助数据录入助手的工作表

测试步骤如下。

step 1　打开随书光盘中的“测试表.xlsx”，文件位于光盘中“..\第二十四章\测试表.xlsx”。

step 2　进入“零售统计表”工作表中，选择 B2 单元格，然后单击功能区中的“开始”→“选项”命令，弹出“选项”对话框。

此处之所以先选择 B2 单元格后再打开“选项”对话框，是为了让对话框中代表列号的复合框自动显示为“B”。当然也可以随意选择单元格，然后在窗体中手工调整复合框的值。

step 3　单击文本框右边名为“—”的按钮，此时“选项”对话框会自动隐藏起来，并弹出一个“确定数据源”的输入框。

step 4　选择“数据”工作表的 A 列，在“确定数据源”输入框会自动产生“数据!$A:$A”，效果如图 24.7 所示。单击“确定”按钮后返回“选项”对话框。

step 5　将“允许从中间匹配”复选框打钩，此时“选项”对话框将显示为如图 24.8 所示效果。

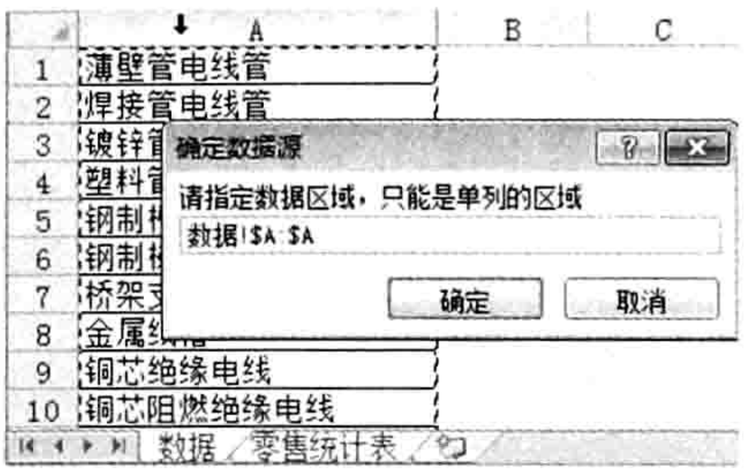


图 24.7　选择数据源

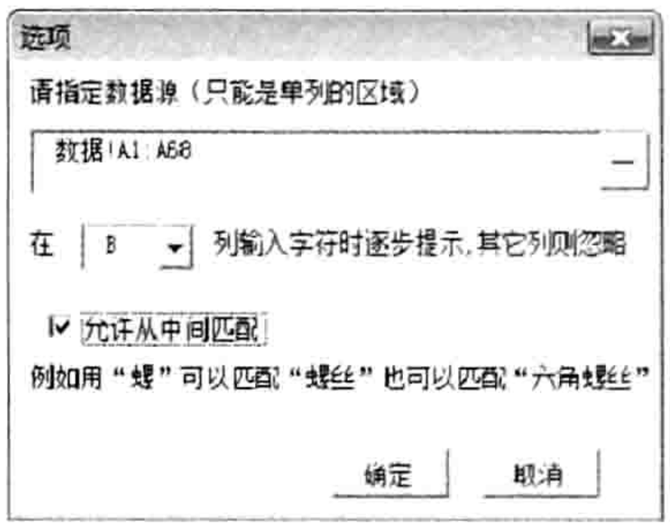


图 24.8　选项对话框

以上操作的目的是以“数据”工作表的 A1:A99 作为数据源，当选择“零售统计表”的 B 列任意单元格时弹出数据录入助手对话框，允许从中间匹配。

step 6 单击功能区中的“开始”→“不提示”菜单，菜单会将默认的打叉图标切换为打钩图标，同时将菜单文字由“不提示”切换为“提示”。如图 24.9 和图 24.10 所示是单击菜单前后的效果比较。

图 24.9　单击菜单前

图 24.10　单击菜单后

step 7 按下右箭头将活动单元格移到 C2，然后再按<←>箭将活动单元格切换到 B2 单元格，此时 Excel 会自动弹出数据录入助手对话框。

step 8 假设要输入“铜芯电力电缆”，那么在对话框的文本框中输入汉字“电”，文本框下方的列表框中会自动产生包含“电”的 20 个目标，效果如图 24.11 所示。

step 9 继续输入“线”，列表框会自动产生包含“电线”的 6 个目标，效果如图 24.12 所示。

图 24.11　包含 20 个匹配“电”的数据

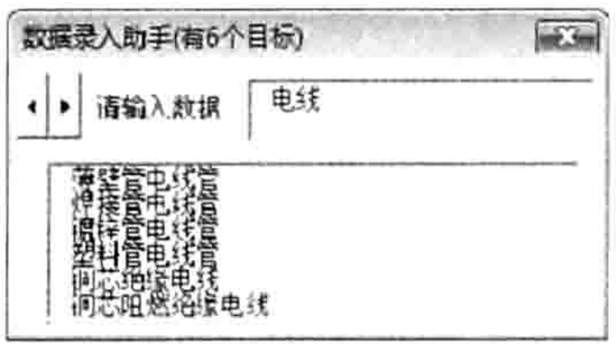


图 24.12　包含 6 个匹配“电线”的数据

step 10 按<Tab>键定位到列表框中，此时列表框的第一行数据会自动显示在文本框中，效果如图 24.13 所示。

step 11 按<Tab>键将焦点转移到文本框中，此时 Excel 会将文本框中显示的值输入到活动单元格 B2 中，然后清空文本框与列表框的值，同时将活动单元格下移到 B3 单元格，效果如图 24.14 所示。

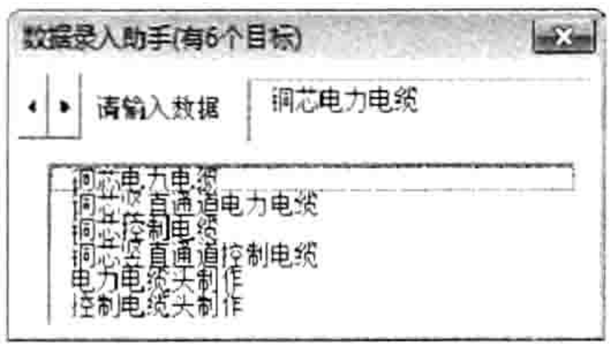


图 24.13　焦点转移到列表框

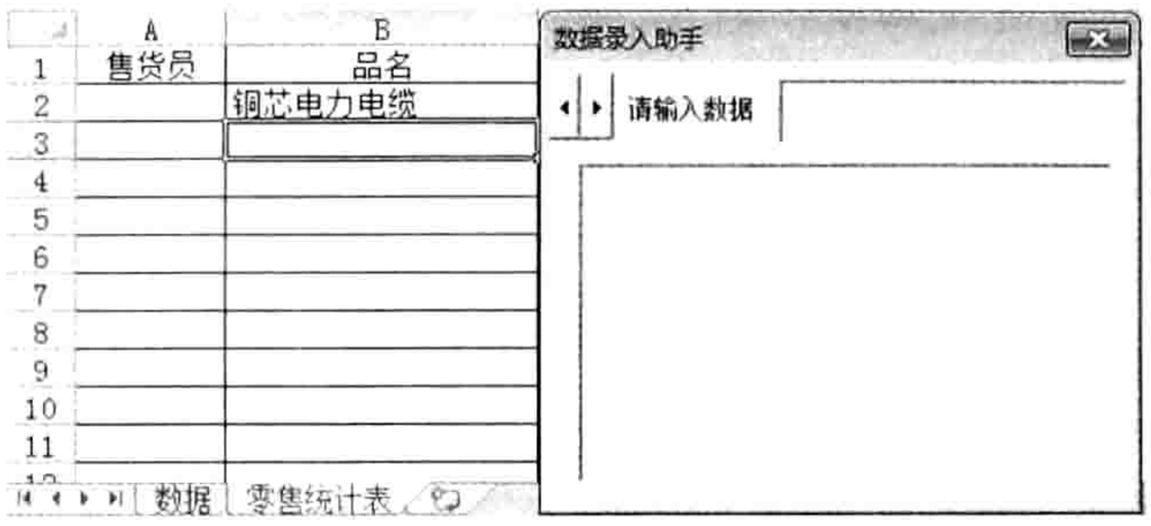


图 24.14　按<Tab>键输入目标字符且返回文本框

事实上，也可以从“铜”字开始输入，同样能快速地输入“铜芯电力电缆”。

step 12 在文本框中输入“铜”，下方的列表框中会自动产生包含“铜”字的 7 个目标，效果如图 24.15 所示。

step 13 按<Tab>键将焦点从文本框转移到列表框中，由于目标“铜芯电力电缆”出现在列表框的第 3 行，因此按<3>键即可将目标“铜芯电力电缆”输入到活动单元格 B3 中，同时清空列表框与文本框的值，将焦点从列表框转移到文本框中等待输入下一个字符，如图 24.16 所示。

当然，也可以按两次<↓>键，从而选中列表框的第 3 行，然后再按<Tab>键将焦点转移到文本框中。在转移焦点的过程中，程序会自动完成输入字符、清空文本框、清空列表框、下移活动单元格等操作。

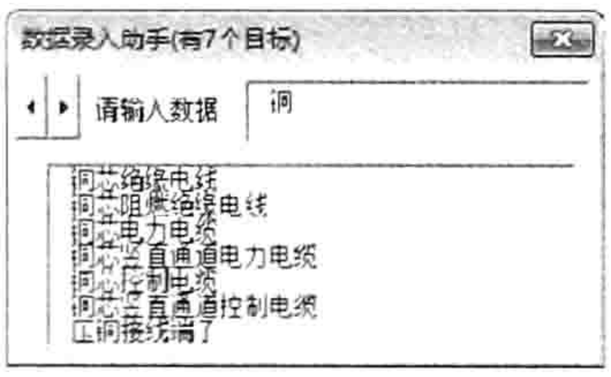


图 24.15　包含“铜”的 7 个目标

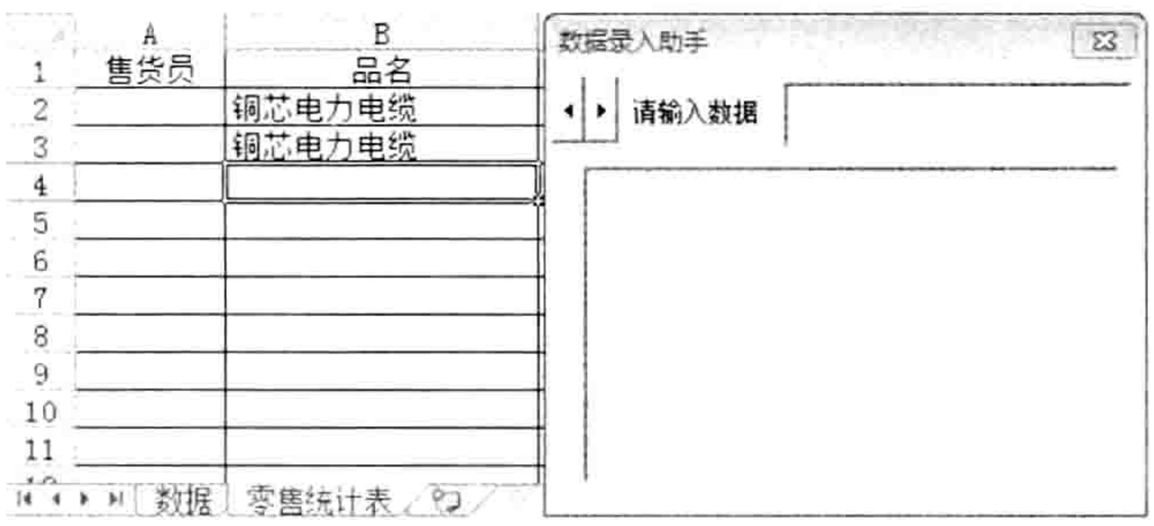


图 24.16　按数字键 3 录入目标且返回文本框

step 14 如果此时需要进入 C 列录入数量，那么可以按<→>键从而将活动单元格右移，此时数据录入助手会自动关闭。

假设要在 C4 单元格输入品名“风机盘管开关”，那么可以继续执行以下步骤。

step 15 移动光标从而定位于 B4 单元格，此时会自动弹出数据录入助手对话框。

step 16 在文本框中输入字符“风”，列表框中会自动罗列出包含“风”字的目标，效果如图 24.17 所示。

step 17 由于要输入的目标在列表框的第一行，因此快速按两次<Tab>键即可将目标输入到活动单元格，并且同时将活动单元格下移，清空列表框与文本框，自动定位于文本框等待录入下一个字符。

step 18 按<Esc>键关闭窗体。

step 19 关闭 Excel，然后重新打开工作簿“测试表.xlsx”。

step 20 单击功能区中的“开始”→“不提示”菜单，使菜单的文字变成“提示”。

step 21 单击 B5 单元格，Excel 会自动弹出数据录入助手对话框。这说明加载宏的自动记忆功能工作正常。

step 22 在文本框中输入“吸”，列表框中会自动罗列出包含“吸”字的所有数据，效果如图 24.18 所示。

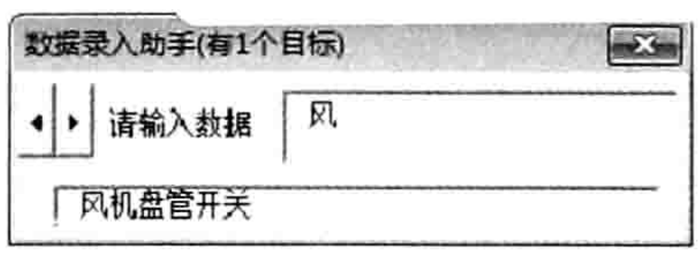


图 24.17 包含“风”的 1 个目标

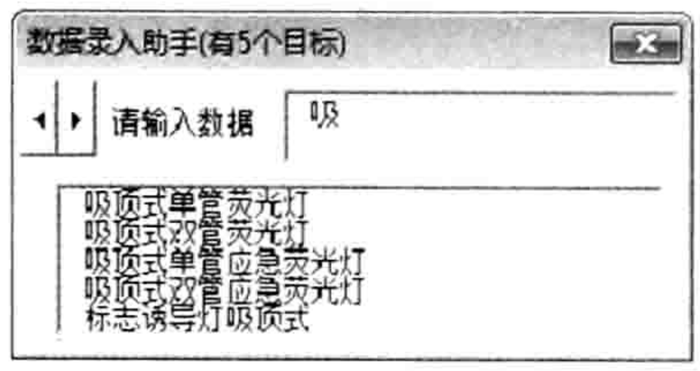


图 24.18 包含“吸”的 5 个目标

step 23 由于要输入的产品名称“吸顶式双管应急荧光灯”位于列表框中第 4 行，因此按<Tab>键或者<↓>键将焦点转移到列表框中，然后按<4>键，程序会自动将目标“吸顶式双管应急荧光灯”输入到动活单元格，将活动单元格下移一格，然后清空列表框与文本框，将焦点切换到文本框中等待输入下一个字符。

step 24 在文本框中继续输入“电视天线”，此时列表中没有匹配对象，按<Enter>键或者<Tab>键会弹出如图 24.19 所示的提示信息。

step 25 单击“是”按钮，关闭对话框，然后按<Tab>键将当前文本框中的字符串输入到单元格中，同时清空文本框的值，将活动单元格下移一格。

step 26 在文本框中继续输入“电视”，在列表框中会看到刚才所添加的新词汇“电视天线”已经产生在列表框中，效果如图 24.20 所示，表明新词汇添加成功。

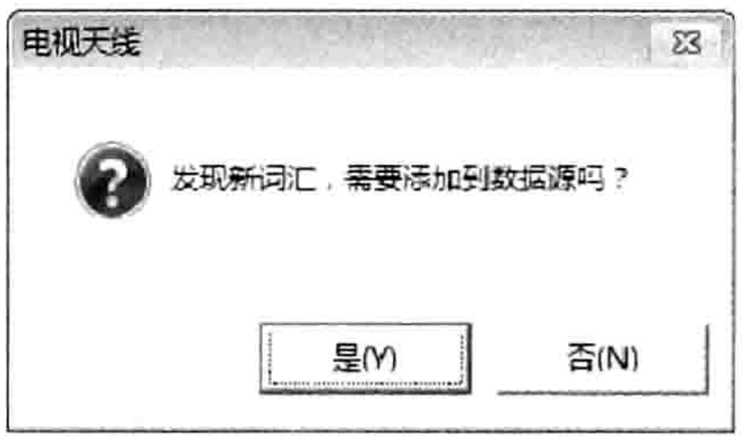


图 24.19 询问是否将新词添加到数据源中

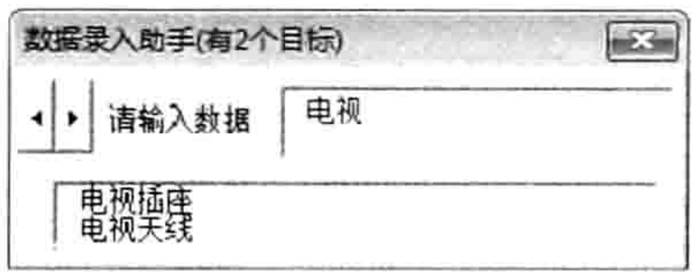


图 24.20 新词汇自动产生在列表框中

step 27 单击窗体左上角的旋转按钮，增加窗体的宽度，然后关闭窗体以及 Excel。

step 28 再次打开文件“测试表.xlsx”，单击功能区中的“开始”→“不提示”菜单，然后选择 B 列任意单元格，可以发现弹出数据录入助手窗体的宽度等于上一次关闭窗体时的宽度，这表明插件不仅能记忆上一次所设置的数据源、显示数据录入助手的列，还可以记录窗体的高度。

本例文件参见光盘：..\第二十四章\数据录入助手.xlam

◆ ◆ ◆

[General Information]
书名=Excel VBA程序开发自学宝典
SS号=13608473
DX号=000015211366
url=http://book.duxiu.com/bookDetail.jsp?dxNumber=000015211366&d=4B58E01EA3DD41F41BC92AB47039BFC4
作　　者=罗刚君著
出版发行=北京：电子工业出版社　，　2014.09
ISBN号=978-7-121-24032-4
页　　数=506
原书定价=75.00
主题词=表处理软件
中图法分类号=TP391.13
内容提要=《ExcelVBA程序开发自学宝典（第3版）》是VBA入门与提高的经典教材。全书分上下两篇，上篇包含入门知识，对VBA的基础理论、语法规则、编写思路、代码优化思路等都提供了详尽的理论阐述和案例演示。下篇包含进阶知识，提供窗体设计、正则表达式、字典、FileSystemObject、类模块、注册表、功能区设计、开发加载宏、封装代码等高级应用。本书基于Excel2010撰写，不过代码可在Excel20...
参考文献格式=罗刚君著．Excel VBA程序开发自学宝典．北京：电子工业出版社，2014.09．