VBA Developer's Handbook

VBA 高级开发指南

〔美〕Ken Getz Mike Gilbert 著

邱仲潘 等译

SYBEX

電子工業出版社

Publishing House of Electronics Industry

URL:http://www.phei.com.cn

VBA Developer's Handbook

VBA高级开发指南

〔美〕 Ken Getz　　Mike Gilbert 著

邱仲潘　等译

電子工業出版社

Publishing House of Electronics Industry

内 容 提 要

这是由两位优秀的VBA专家为各级VBA开发人员编写的一本高级开发指南。

书中介绍了独特的VBA编程技巧、思路和学习方法。还介绍了许多编程问题的创造性解决办法，如：分析字符串、从注册表读取数值、数组排序、生成新的网络共享、在应用程序中播放电影、捕捉和处理应用程序错误、编写可以修改现有代码的代码，等等。通过对本书的学习，可帮助读者逐步成为VBA高级开发人员。

此外，书中还提供了丰富的样本代码（可在选配光盘中找到），开发人员可直接利用，将代码复制到自己的应用程序中，从而大大提高编程效率。

书　　名：**VBA高级开发指南**

著　　者：〔美〕Ken Getz　Mike Gilbert

译　　者：邱仲潘 等

责任编辑：张盛华

印 刷 者：北京顺义颖华印刷厂

装 订 者：三河金马印装有限公司

出版发行：电子工业出版社出版、发行

北京市海淀区万寿路173信箱　邮编：100036　发行部电话：68279077

北京市海淀区万寿路甲15号南小楼三层　邮编：100036　发行部电话：68215345

URL:http://www.phei.co.cn

经　　销：各地新华书店经销

开　　本：787×1092 1/16　印张：36.375　字数：950千字

版　　次：1997年11月第1版　1997年11月第1次印刷

书　　号：ISBN 7-5053-4461-7/ TP・2074

定　　价：60.00 元

著作权合同登记号　图字：01-97-1635

献给哈佛大学扩展项目的Paul Bamberg和Harry Leitner博士，他们为我打开了生活的新篇章；感谢我的学生Joan Weinstein，他让我真正知道自己所授课程的含义。

Ken Getz

献给我的双亲Robert和Donna Johnson，他们向我提供了用于生活的最重要开发工具；献给Robert和Rick Sterrett，他们提供了每个年轻人需要的角色模范。

Mike Gilbert

致　谢

和任何书籍一样，本书也是在许多人的帮助下完成的。首先感谢不知疲倦的编辑Dusty Bernard，通过她再三推敲，使我们的作品用更好的词语表达。加上我们编写的其它著作，Dusty已经为我们翻阅了5000多页手稿。仅仅读完这些内容已经非常不易，何况她还要让它变成更可读就更不容易。

我们还要特别感谢技术编辑David Shank，他不放过每一个细节，为本书提供了无数建议，大大改善了本书的内容和例子。这是我们与David合作的第二本书（另一本是Access 97高级开发指南）。如果有机会，我们希望与David进一步合作。David目前正处理Microsoft的开发文档，编写Access 97文档和处理Access 95与97样本应用程序。

当然，没有出版社，任何书籍都无法问世。在Sybex中，生产编辑Shelby Zimmerman投入大量精力管理我们多变的日程表；开发编辑Melanie Spiller像朋友、知已和老师，使我们的工作愉快、富有动力、顺利而及时，没有她的参与，本书无法形成。

谢谢FMS公司的Dan Haught，他在规划了类似于本书的计划后走了另外一条路，建立了Access源码大全，是个Microsoft Access源码库，包括与本书课题相对应的材料。Dan向我们提供了他的详细规划，使我们顺利地开始了本书的编写。关于FMS公司及其产品的进一步信息，请看其Web站点www.fmsinc.com。感谢FMS公司的Luke Chung，他向我们提供了第3章用到的涉及数字舍入和计算误差的文档和例子。

还要感谢MCW技术公司的高级顾问Mary Chipman和Key Data系统公司的总裁Andy Baron，他们深入探索了数字操作有关的VBA问题并提供了本书第3章的大部分材料。

Trigeminal软件公司的Michael Kaplan编写了第11章。该章介绍涉及用程序处理网络的问题，通过Michael在网络及其管理方面的研究和经历，使该章得以大大改进。

还要感谢Jin Ferguson、Mike Gunderloy和Brain Randell等专家，他们提供了章节安排和代码分析的设想和建议。Microsoft Access支持小组的MalcohnStewart提供了NEATCODE.MDB样本数据库，作为我们进行各种编程课题讨论的出发点。此外，还要感谢Microsoft的整个Internet平台和工具市场小组，包括Michael Risse、David Lazar、Scott Horn、Robert Green和Neil Charney。Neil撰写了本书的序。他们提供了编写本书所要的许多资料。我们还要感谢Microsoft公司Visual Basic小组项目经理Craig Symonds，本书前言中记录了我们对他的采访。最后还要特别感谢Greg Reddick和Paul Litwin。Paul不仅提供了精神和道义上的支持，还让我们使用姐妹篇《Access 97 高级开发指南》中的一些材料。Greg

认真修改了Office 97和VBA的命名规则。这些文档已列在附录A中。

还要特别感谢Micky Friedman和Pinnacle出版公司的人员，他们让我们摘取了过去出版的Smart Access中的一些代码例子和文本。Smart Access是Pinnacle出版公司的快报，用于开发人员用Microsoft Access建立解决办法。

Kudos也在弗吉尼亚雷奇孟ParkSide Starbucks的小组中，本书的一部分是在这里写成的。这些伙计们不知不觉地在本书创造过程中起了重要的作用，提供了大量的咖啡、点心和免费机时。

特别值得一提的是，要感谢Karen Jaskolka和Peter Mason，没有他们就不可能有我们的著作。他们无穷的支持、良好的性格和自娱自乐而不纠缠我们，使我们得以进行写作。

关 于 作 者

Ken Getz

Ken Getz是MCW技术公司的高级顾问，精通Microsoft产品组，获得1993～1996年的Microsoft MVP奖（向CompuServe提供技术支持），合作编写了多本用Microsoft Access开发应用程序的著作，包括最畅销的Microsoft Access 2开发指南及其Access 95和97版本。Ken还是Smart Access的撰稿人和其他开发出版物的经常撰稿人。目前，Ken用大量时间在全国旅行，为应用程序开发培训公司讲授Access和Visual Basic开发的培训课程。他还在许多会议上发言并参加全球的展示，包括Teck*Ed、Advisor出版公司的DevCon、Windows Solutions和Access & VB Tech。业余时间，Ken先生还玩一手漂亮的钢琴。Ken的e-mail地址是keng@mcwtech.com。

Mike Gilbert

Mike Gilbert是MCW技术公司的高级顾问，精通用Microsoft Access、Visual Basic、SQL Server和Microsoft Office开发应用程序。他与Microsoft有过多个项目的合作，包括Workgroup样板、Office开发工具库和DevDast。他为多家刊物撰稿，是Smart Access和Access/Visual Basic Advisor的撰稿人。Mike是Microsoft Access 95开发指南和Access 97开发指南的合作者。他是应用程序开发培训公司的培训员，是Teck*Ed、Advisor出版公司的DevCon和VB Tech之类的经常发言人。他的业余时间与妻子Karen和两只可爱的小猫Chicage与Cair共享弗吉尼亚州雷奇孟的安静的南国生活。Mike的e-mail地址是mikegz@mcwtech.com。

序

不成熟的艺人靠模仿，成熟的艺人靠继承。

Lionel Trilling

好的开发从继承开始。当然，不是真的去继承，而是利用现成代码，更迅速地开发强大的应用程序。这也许是Microsoft Office成为世界上最著名的开发平台的一个原因：可以利用Microsoft Office 97提供的500多种预建立、预测试对象。

作为开发人员，我不想从头开始编写图表应用程序，利用Microsoft Excel，也就没必要从头开始。要生成公司日历或联络管理系统，只要定制Microsoft Outlook，复用其中的调度、联络管理和任务管理功能。

将Microsoft Office的功能用到定制应用程序中并不困难，利用Office的编程语言Visual Basic for Applications(VBA)即可达到这个目的。VBA已经在整个Microsoft Office中提供，与Visual Basic 5.0中的VBA相同。Microsoft已经出让Visual Basic for Applications的版权，一个应用程序中开发的程序可以用于几百个支持VBA技术的第三方应用程序中。

所以，预建对象随处可见，开发人员具有一致的开发环境和单一的跨应用程序编程语言，开发人员的日子好过多了。本书则会使开发人员的日子加倍好过。代码不仅可以在完整应用程序中复用。也可以在用VBA写成的程序中复用。Ken Getz和Mike Gilnbert在本书中提供了编写好的高质量程序，可以在任何VBA应用程序中立即采用。

如果你要一个牢不可破的错误处理器、要寻找处理和换算日期和时间值的优秀程序、要寻找完美的字符串处理程序，当然可以自己建立，但有谁喜欢一遍遍重复编写这类程序呢？我很高兴告诉你，你不再需要在任何应用程序中从头编写这些东西了，只要买了本书，你生成的应用程序就可保证建立在更坚实的基础上。

Ken Getz和Mike Gilbert在本书中提供了VBA资源的“办公室”，一组经充分测试的预建立工具和方法，可以在定制应用程序中立即采用。更妙的是，本书中的代码和要诀不仅适用于Visual Basic for Applicatios的应用程序，还适用于Visual Basic、Office、Visio和AutoDesk的开发人员，他们都是VBA开发人员。我唯一的担心是有了Office 97、VBA版权和本书选配光盘的预建代码，开发人员会有太多空闲时间，无是生非，制造麻烦。

当然，预建对象和代码只是使开发过程更容易，而不能使你变成良好的开发人员。本书将使你变成良好的开发人员。书中提供了编写专家水平应用人员能利用的最有效、最可靠的代码。

按照本书提出的程序和开发原则，你将得益于我所见过的两位最优秀的开发人员的帮助。更让人得意的是，他们毫无保留地支持你利用他们的代码！

Office 97开发版生产经理

Neil Charney

前　言

Visual Basic for Applications（VBA）最初出现在Excel中，然后出现在其它Microsoft Office应用程序中，作为一种编程控制环境的工具，并可通过OLE自动化与其它应用程序一起工作。1996年，Microsoft允许其它厂家购买VBA语言工具和环境版权，在自己的产品中使用，从而使VBA世界出现了一个飞跃发展。到本书编写时，已经有40多个厂家购买了这种激动人心的技术，使许多产品的用户可以用VBA控制应用程序和自动化服务器。

对于VBA开发人员，其中最妙之处是一个产品中学到的技术可以直接应用于另一个支持VBA的产品中。编程环境相同，查错工具相同，语言也相同。最后，Basic编程人员（VBA也是Basic语言的一个变种）可以用最终用户喜欢和熟悉的工具编写他们可以适应、修改和扩展的应用程序。

1. 关于本书

VBA是一种胶水，将各种跨平台应用程序粘接在一起，许多新的编程人员不需编程帮助就可以立即上手。本书介绍许多编程问题的创造性解决办法，例如：

- 分析字符串（第1章）
- 从注册表读取数值（第10章）
- 数组排序（第4章）
- 寻找计算机包括的处理器类型（第9章）
- 生成新的网络共享（第11章）
- 在应用程序中播放电影（第13章）
- 捕捉和处理应用程序错误（第7章）
- 计算两个数之间的最大公因子（第3章）
- 编写可以修改现有代码的代码（第8章）

这本书是不是一本VBA编程参考手册？不是，本书不想提供完整的手册，而是要提供思路、办法和学习方法。我们总结了多年的Basic编程经验和我们认为对许多开发人员有难度的各种问题，希望以有趣而有用的方式表达这些信息。

首先，本书不是针对某个产品的，无论使用Microsoft Office 97或任何其它支持VBA 5.0或以上版本的产品，都可以利用本书中的代码。本书的代码用3种格式提供（Microsoft Access 97数据库、Microsoft Excel 97工作本和单个文本文件），任何人都可以利用光盘阅读器立即利用这些代码。再说一遍，尽管本书用Office 97写成，但这里的代码适用于任何支持VBA的产品。

为此，本书重在代码而不在用户接口，因此书中的例子很少成形的样子，大部分例子涉及调用Immediate窗口代码。别指望有许多漂亮的例子，本书的用意不在于此，而在于提

供工具，精彩的接口要由你来提供。

说明：什么是Visual Basic 5.0呢？VB5是VBA的变形，具有比任何其它VBA主机更丰富的特性。为了使本书的代码可供任何Office 97中和发给外部厂家的版权中的VBA复用，所以只对Visual Basic的新版本顺便提及，即只在VBA5.0的局限性已在VB5.0中解决时才在文中介绍。否则，本书的所有代码也适用于VB5，只是用VB5的新特性解决同一问题时可能有更妙的办法。

编写本书的目的是提供有用的代码和介绍这些代码，以使在需要修改或增加功能时进行修改和扩展。书中介绍了广泛的问题，但不可能面面俱到，一本面面俱到的书要比本书厚10倍。本书只是探索这些新问题的一个开端。

2. 本书适用对象

本书是针对在特定条件下需要帮助的开发人员的，包括VBA开发的新手和老将。如果要寻找If...Then结构的作用说明或初次编写VBA程序，则应当选别的书，而如果要编写大量代码，要将样本中代码和整个模块复制和粘贴到你的应用程序中，并一行一行地处理代码，则本书将满足你的需要。

本书适用于三类读者：

- 初级到中级VBA用户。写过一些程序，想把它们放到一个应用程序中，需要编写特定程序的帮助，独自琢磨代码很费劲，这时可以将书中的代码复制和粘贴到模块中，跳过书中对代码工作的繁琐介绍，回到你的应用程序中。
- 高级VBA用户。写过大量代码，面对更多更大的编程挑战，需要编写具体程序，可以亲手编写，但有其它方面的压力，如要尽快完成应用程序，这时可以利用本书的程序，修改成适合具体需要，并按照书中的介绍加进自己的VBA使用知识。
- VBA专家。即使是最熟练的VBA编程人员，也会有一些需要的程序还没有编写，这时可以利用本书提供的代码进行完善和调整。当然，你也许有更好的办法改写本书的代码，欢迎不吝赐教。

如果你是上述读者，则本书对你会有所帮助。

3. 预备知识

为了在有限的篇幅中放进尽可能多的代码，我们省去了一些最基本的材料。如果你不知道VBA中代码放在哪里、如何建立模块或不知道各种变量类型，则最好把本书放在一边，先读一些VBA所在应用程序中的参考资料。一定要充分理解下列问题之后再阅读本书：

- 建立模块
- 建立过程程序
- 变量及其数据类型
- VBA语法（包括If... Then、For...Next和其它控制结构）

充分理解这些概念之后再进入本书，会使你从书中得到更多的营养。

4. 本书使用的规则

共同进行多个项目之后，我们发现一致的样式和定义的规则能使多个编程人员更方便地共同进行一个项目。为了便于理解我们的代码，我们在本书和所有专业工作中采用了一个命名标准。

对象的命名采用Reddick VBA（RVBA）命名规则V4.0，许多Access和VBA开发人员都把它当作命名标准。这个标准是由著名Access和Visual Basic开发员和培训员Greg Reddick开发的。按照这个规则，本书保持了高度一致性。这个规则在Smart Access中发表过，也列在本书附录A中。

除了遵循RVBA标准外，我们还将用户在程序中可能用到的所有公共函数、子程序和用户定义类型冠以前缀dh（表示开发指南），将所有公共Windows API声明冠以dh_api，将所有全局常量冠以dhc。这些规则应避免与应用程序中的现有代码冲突。如果将多个章节的样本模块输入到同一应用程序中，本书的统一命名可能导致命名冲突，这时应注明冲突的API声明和用户定义类型。

关于错误处理请注意，编写实用过程程序时（如本书所列的实用过程程序），总会遇到要不要包括扩展错误处理的问题。为了简单起见，加上我们都不喜欢提供显示错误消息的程序，所以我们只在过程本身需要时才包括少量错误处理，否则基本不包括错误处理。这样，调用这些过程的代码要捕捉和处理从这里提供的代码中产生的错误。当然，也可以在输入过程中加入自己的错误处理（关于错误处理，详见第7章）。

5. 章节说明

本书分成三个部分，每章都有许多例子，大部分例子代码都放在本书选配光盘中。

（1）使用内置VBA功能

第1～第4章介绍内置VBA功能。每一章首先介绍VBA内置特性，然后将其扩展为专用VBA过程。第1章重点介绍字符串处理，包括字符串分析。第2章介绍关于日期的一切（除了如何取得）。第3章介绍数字数据类型，使用数据换算和其它数字型分析。第4章介绍数组结构，详细介绍其搜索和排序。

（2）VBA高级用途

本书第二部分着重介绍更高级的VBA问题。第5章介绍类模块及其在VBA应用中的使用。学习本章之后，你会发现，类模块是编写坚实的可复用VBA码的关键。第6章谈一些花边知识，介绍用类模块生成大多数VBA编程人员认为不可能生成的数据结构：链接表、二元树、堆栈和查询。第7章介绍编写牢靠的、调整好的VBA应用程序，包括处理错误、建立事件登记和建立应用程序的过程跟踪堆栈。最后，第8章介绍除了类模块以外最激动人心的VBA特性：作为自动化服务器，利用本章的知识，可以建立和修改代码、编写加入件和分析与建立文档代码。

（3）用Windows API扩展VBA

本书第三部分介绍用Windows API完成许多VBA本身无法完成的任务。第9章提供了9个类模块，可以提供和设置关于当前软件和硬件环境的信息。第10章深入研究系统注册表，提供了一系列类模块，将所有复杂的编码细节包装在这些类模块中。第11章提供了处理网络的信息：建立共享、选择设备和处理用户。第12章探讨处理磁盘和设备的API功能，提供了一系列处理和分析计算机系统文件的函数。最后，第13章介绍多媒体，包括声音、电影和位图在应用程序中的使用。

6. 使用书中样本

本书每章对应选配光盘的一个文件夹。在每个文件夹中有书中所用的各个例子文件。每章例子至少提供了三种格式。第一，每章的文件夹中有一个Microsoft Excel 97工作本，包含本章介绍的所有模块，可以随时试用。此外，我们提供了每个模块的BAS或CLS文件，可以输入任何支持VBA的项目中。最后，为了便于Microsoft Access开发人员使用，每章建立了一个数据库文件，输入了所有模块。

除了书中的样本外，光盘中还收集了有用的演示和工具，关于光盘中的最新内容的信息，请看光盘根目录中的Readme.txt文件。

提示： *光盘上的所有文件都是只读文件。将光盘文件复制到硬盘时，可以用Windows Explorer或其它方法删除文件的只读属性。例如，在Explorer中右单击文件并选择下拉菜单的Properties，然后取消Read Only属性。*

7. 如何使用本书

本书有两种使用办法。一种办法是从封面到封底一页一页地细嚼慢咽。这时就准备好贴纸，随时将有趣的代码标注出来，否则以后再寻找这些有趣的材料会很费劲。

但更好的办法是阅读前言，浏览几章，然后把本书当作一本参考资料。这时要注意两点：

- 如果不熟悉类模块，请阅读第5章看看其如何工作，如何进入VBA中。
- 如果没有用过Windows API调用，请阅读附录B。其中介绍了Windows API在编程工作中的作用和如何使用这个宝贵的技术。

以上两点是了解本书其它部分所必须的，没有对这两点的充分了解，很难吃透其它部分内容。

本前言余下部分是Ken Getz与Microsoft现任Visual Basic小组生产经理Craig Symonds之间的交谈。其中透露了VBA历史的一些内幕、VBA的来历和具体实现中的一些细节。

8. VBA到底是什么（与专家一席谈）

1996年夏季，Ken对Microsoft的Craig Symonds作了简短的采访，了解了Visual Basic for

Applications(VBA)历史、未来和现状方面的一些细节。

Ken：我是Ken Getz，在雷奇孟的Craig Symonds办公室中，Craig，你的职务是什么？

Craig：我是Visual Basic小组的项目经理。小组的课题包括VB、VBA和Visual Basic Scripting。

Ken：你在这个小组工作多久了？

Craig：5年，与小组同龄。

（1）VBA的历史

Ken：首先问一下，VBA是怎么来的，能否讲讲VBA的历史？

Craig：VBA是几年前开始的。VBA的前代产品是Embedded Basic，用在Access第一、第二版和前三个Visual Basic版本中。它实际上与DOS版本的Basic有联系，我们把DOS版Basic的一些实现方法转变到Windows中。

Ken：那么Access和VB间的代码是否有共同之处？

Craig：是的，整个语言机制是完全共享的。Access的Embedded Basic产品有一些扩展，开发VBA的一个原因是要开发一个通用的共享技术。1990年，我们开始了所谓Object Basic的项目，这是Visual Basic for Applications的原名。我们用一个小组开发了一个Basic版本，面向对象的Basic版本。其主要目标是加快执行速度、可移植到多个平台和改进基础技术。1993年，我们推出了第一个版本，是作为Excel 5的一部分推出的。

Ken：有人说VBA或基于Basic的某种批命令会成为操作系统本身的一部分，事实是这样吗？

Craig：在每一个版本中，我们都想把VBA变成操作系统本身的一部分。我们考察了特性请求的成本、版本问题，等等，但迄今为止，我们还得花时间满足Visual Basic和Application两个组。我们要满足这些主要的客户，然后再将其推广到其它地方。

Ken：似乎这已经大功告成。标准VB版本已经很便宜，几乎人人都享受得起。

Craig：是的，有些国家已经随书附送。

Ken：实际上美国也有随书附送的，1993年以来，已经在所有Office产品中附带提供。

Craig：Access、Excel、VB和Project已经采用，PowerPoint已用它进行内部向导开发，Word和PowerPoint和Office 97版本中也都已经有VB。

（2）VB的语言特性及其与Office的集成

Ken：VBA93和VBA94(Office 95中的版本）之间好像没有语言特性方面的大变化。能讲讲这两个版本间的主要改变和今后的变化计划吗？

Craig：为了回答你的第一个问题，关于VBA 93和94间的差别，最大的差别在于性能和能力。为了支持Vtable关联而不是只用IDispatch接口关联对象，VBA内部作出了巨大的改变，删除了许多功能限制，执行性能大有改进，并加进了语言上的补充，使VB4与VB3完全向下兼容。

将来的方向是继续改进语言的面向对象性和继续提高性能。我们打算加进必要的语言特性和结构，保证使其适合Internet环境，继续改进与基础COM对象模型的交互作用。94中已经可以直接访问COM接口，可以处理COM支持的大多数数据类型，我们还将在这方面进

一步改进。

Ken：是VBA小组推动VBA改变，还是使用VBA的产品小组作出改变请求？

Craig：这是一个相互促进的过程。语言的结构性改变是由我们在语言定义中提出的，但我们随时了解每个应用程序小组的用户需求，保证熟悉各个应用程序中的可编程性，即面对用户、面对各个小组中程序管理器的可编程性，了解其需求，满足其需求。这是个艰难的决策，VBA技术的客户越多，这种需求就越复杂。我们要综合考虑各方面的需求，给出利索地解决问题的一般方法，而不是一次性解决每个要求。

Ken：但不能针对任何特定产品，是吗？

Craig：一般来说，是这样的。但Access中直接采用了一些VBA功能以便将编码存放在数据库中，这些接口是VBA3环境中没有的。由于Access不用这个构件，所以不必做这个工作。

Ken：那么Excel加进大量财务功能时，这是纯粹的非VBA问题，你根本闻所未闻？

Craig：是的，VBA是个非常可扩展的结构，从而能处理各种客户要求。我们的整个运行环境建立在可扩展机制中。事实上，这可以从产品中看出。我们的运行环境函数，如Sin和Cos，通常是由类型库描述的DLL项目。编译器只是查找类型库并猜出如何关联。很少内置在编译器中的运行函数。运行环境的工作和其它DLL一样，任何人都可以取一个DLL，提供一些功能和建立它的类型库，它就成为该语言的集成部分。你不能说这些不是内部函数。这样，扩展功能非常方便，任何支持VBA的应用程序都可以方便地加进自己的DLL函数和COM对象。主对象模型是通过这个机制加入的，Data Access Objects（数据访问对象）之类的加入技术也是通过这个机制加入的，这样就提供了巨大的灵活性。

Ken：COM接口是双向的吗？你说过可以这样与应用程序交谈，应用程序是否也用COM接口响应呢？

Craig：是的，COM对象可以启动VBA可以收集的事件。然后VBA可以调用这个COM对象的方法和属性。VBA本身也可以向COM接口显示其可以面对其它对象，使这些对象调用VBA。

Ken：是不是所有支持VBA的应用程序都共享相同的表达式求值器？在Access中，查询中有简单函数调用时，是否调用VBA解析这个表达式？

Craig：Jet语言工具提供使用VBA表达式求值器的表达式求值器。这是使用表达式求值器的唯一技术。Access在字段中使用表达式时，实际上是个Jet表达式，然后Jet回过头来调用求值器，这是使用表达式求值器的唯一技术。

Ken：那么，Excel要进行自己的所有表达式求值？

Craig：是的，出于性能考虑要这么做。它们能做我们无法进行的优化工作。

Ken：我发现Office 95的语言中有个新特性，卷标不必是模块中唯一的。我不知道这是何时如何引入的，任何文献中都没有看到说明。

Craig：设计Object Basic（即现在的Visual Basic for Applications）时，我们将它设计成该语言在整个语言上是正交的。前三版Visual Basic和前两版Access中的许多方法不再需要，许多限制已悄悄删除，如结构中不能有变长数组的限制。新技术出现了一些新的问题，但大多数原有限制均已消失，这一点意义重大。我曾回过头去用各种Visual Basic版本编写程序，发现到处遇到限制，不能这样，不能那样，应该这样，应该那样。我简直无法想象，

无法调用用户定义的窗体方法时，日子怎么过。

Ken：Access 2中调用窗体的专用方法很费神，新的特性真是妙不可言！

我想问问堆栈空间的问题。在旧产品中递归是个大问题，堆栈空间在32位版本中是否大多了？

Craig：是的，堆栈空间是由操作系统定义的，所以在32位的操作系统版本中，它只受系统内存的限制，可以动态增长。应用程序可以设置堆栈空间的上限（比如1M），但操作系统只是根据需要分配内存。随着堆栈增大，操作系统可以根据请求分配更多的内存。

Ken：尽管VB3和Access2中的递归有限，但现在只要控制递归，好象就可以无限制地使用了。

Craig：的确如此。

（3）Office和VBA间的通信

Ken：VBA如何挂接到Office应用程序？

Craig：这里有两个问题，一个是Office应用程序要求取得宏表之类的东西和挂接用户接口，我们提供了进行这些工作的接口；另一问题是VBA调用Office应用程序提供的对象模型，前面说过，这是通过自动化进行的。对象出现为COM对象，我们只是用COM机制调用这些对象，引用为对象模型建立的类型库，对象即变成语言中的第一类对象，可以直接访问。无论从应用程序中运行VBA或是在其它应用程序中运行另一版本的VBA，其作用完全相同，都可以引用其类型库和用同样方法在过程内、过程外和机器间调用，得到的功能完全相同。只要引用类型库，COM即处理所有位置透明度。

Ken：Office 97的VBA中有什么新东西呢？

Craig：这一版VBA的主要用意是将VB可编程性模型引入Office。我们在Excel 5中加入VBA语言，但有人提出，在这里加入该语言虽然美妙，却不象个VB，因为其中只有该语言和编辑器。所以我们考虑要把整个VB可编程性模型加进Office，包括所有可编程性特性。可以控制文档，文档“后面”有代码模块。可以双击控制引出代码模块，文档及其控件启动的事件即推出下拉的所有对象和过程。可以建立事件过程，可以在事件过程中编码，可以编写形如Me.BackColor=255的代码，用Me指文档，这就显式关联到文档。可以不加限定访问文档成员，即只写BackColor=255。现在支持Office内的类，所以可以实际建立包装功能的专用类。方式模型类似于VB，但更松散些，设计和运行模式与断点方式间没有明显分别。我们还支持类似VB中的窗体包。

Ken：除了Access以外，所有Office产品都有这些新特性，而Access只继续语言的改变，没有继承编辑器的改变，是吗？

Craig：我们加进了构件Visual Basic Environment （VBE）Access，出于定时和调度考虑，我们只选取传统VBA技术而不选取环境。因此，它取VBA提供的所有技术，而没有取环境，而是提供自己的环境。这里失去的是一些在环境中停靠的能力和一些菜单机制。它们选取所有编辑器加强：下拉清单、指示器条和所有自动完成技术，并且还取得所有查错能力的加强，包括Debug窗口的Locals视窗。

Ken：兼容性方面如何？Office和VB是否有相同的VBA功能？

Craig：Office版的语言工具与VB5版的语言工具有一些差别。对VB5中的语言作了一些

补充。

Ken：我一直在想，Excel 95中的VBA与别处都不相符。Office 97中所有Office产品是否更一致些？

Craig：是的，所有Office都用相同版本的VBA。VB5的功能只是比Office略有增加。

Ken：可以想象，理所当然。

Craig：我们将更多地转入由Visual Basic主导变化的模型。新的VBA改变将集成到产品中，并在一定程度上推动语言的扩展，然后Office选用上一版VBA。但在许多情况下，Office并不是严格的VB子集，而要加入独特的扩展。

Ken：有没有人在研究Pascal之类的语言手册，看看Pascal之类有什么特性可以用于VBA中？

Craig：有的，我们曾参照PERL之类的语言进行字符串处理，参照竞争性产品中的其它Basic版本中加入的革新特性，参照Pascal加进一些我们能控制的特性。

Ken：Office 97中的VBA与VB5中的VBA有没有差别？

Craig：VB版本中有些语言特性是Office版本中没有的。在Office版本中，我们的主要工作是把整个VB模型放在基于文档的应用程序中，还加进了事件同步之类的语言特性。

Ken：但Access 97两种特性都没有，是吗？

Craig：记住，窗体模块是个类模块。VBA有个类的概念，无论从窗体派生还是从简单的基本类派生，对VBA而言是一样的。这是我们从主应用程序提供的对象派生所谓“类”的另一原因。有时应用程序传递基于窗体的类，有时传递另一种类。对于VB5，我们用几个语言特性改进OLE自动化服务器的包装层，加进了项目范围内公开的成员不暴露在外部的能力，加进了向类暴露的枚举能力。常量不显示，但枚举会显示。

Ken：要提供枚举常量时会怎么样呢？

Craig：可以提供自己的类型库，用于显示这个常量，也可以编写产生常量的ODL。

Ken：妙哉！

Craig：只需编写自己的类型库，列出常量并从Basic引用，但不能将它包装在Basic本身中。

Ken：这是个好办法，但没有任何人提过这个过程。这个主意真好，只是对付ODL很不容易。

Craig：今年将推出一些工具，使ODL生成更直观，使更不熟练的用户也能产生类型库。

Ken：我也算一个。我曾一天花两个小时琢磨这个天书一般的语法。

发布在Office产品中的呢？那是不是一组DLL和接口规范？

Craig：是的，我们提供了集成VBA的完整SDK，包括一个DLL，具有描述接口的相关头文件。这使产品可以集成到我们提供的用户接口，提供启动产品所要的菜单项目等东西。我们还包括了如何记录宏、调用函数、处理工具条键交互作用、重复和猜出各个宏的内容以及诸如此类的信息。

（4）编译及其对VBA的意义

Ken：编译对于VBA的意义是什么呢？人们常把编译看作建立本地执行文件。

Craig：这不对，只有第1版中才有何时提供VBA编译的问题。

Ken：对！那时有这个问题，但怎么解释编译的真正意义呢？

Craig：编译是对源码进行语法和语义分析，对产生的码进行优化，然后产生其输出码。无论产生X86指令、MIPS指令、PowerPC指令，还是所谓外部代码（ex-code），都是代码。翻译器（芯片的翻译器或软件翻译器）如何执行这个代码与编译过程无关。人们把编译与本地码联系在一起，芯片通过这些产生的指令知道如何执行，好象与本地码是一回事。但实际上，有两个正交的问题。VBA的确做了我们称为编译的工作，进行了代码的语法分析和语义分析，有时也进行产生码的少量优化。我们实际做的工作象是内部生成编译树，然后通过这个树产生实际的外部码（ex-code）或P代码（p-code）。在Visual Basic中，我们实际上要把P代码系列化为Windows码段，我们产生真正可移植的执行文件格式（PE Exe），用Windows装入器将P代码装入码段。这样，Windows进行P代码内外的所有包装工作，就象对本地码一样。

Ken：那么说就是无关紧要了。

Craig：就VBA是否有“编译器”而言，的确无关紧要。字面上这就是完整意义上的编译。唯一的差别是产生的码不是X86芯片组直接理解的代码，而是运行在这些文件类型上的翻译器理解的代码。

Ken：那么控制结构是否进行了优化？是否简化了Select Case语句或删除了死码呢？

Craig：还没有，将来也许会做一些这方面的工作。我们进行了常量折叠（constant folding），即对于常量表达式，我们实际求值表达式并在线放上数值。我们进行了本地分配优化，如果本地变量要分配大量空间，我们实际上在过程入口处进行堆叠分配而不是进行堆栈分配。编译阶段复用了临时变量（temps）。我们进行了一系列相当简单的优化，保证在符合性能要求的前提下尽量减少码长。

Ken：显然，有些优化要求两步编译器。VBA中怎样采用这种优化呢？

Craig：我们实际上对码进行多步优化。这不同于传统编译意义上的步。输入代码时，我们实际上是在输入时立即分析。

Ken：这是你在一行输入时进行语法检查的方法。

Craig：是的。我们分析输入的文本，将它放在操作码（OP-code）形式中，实际上是P代码的分析版本。我们取出所有名称，放在名称表中；识别所有关键字，放在操作码流中，可以很快列出操作码流。输入一行后，输入的源码消失，代之以动态生成的二进制表示。

Ken：如果我输入一行代码，包括许多用于排版的空格，则该行会消失，换成表示该行代码的东西，是吗？

Craig：是的，变成了二进制格式，所以重新列出输入的码行时，有些间隔可能改变。有时，输入的间隔保留（如在Dim语句中），但我们只把间隔看成操作码的显式部分，并不保存间隔本身。如果在一行中输入多个空格……

Ken：则只是插入二进制格式中这些空格的一个操作码，是吗？

Craig：不，某些操作码确实有指定输入的间隔数的参数。但回到分析阶段：一旦将文本变成操作码，实际上要多个步骤进入编译过程。我们知道所有模块级声明语句、常量声明和模块级变量。我们也知道所有模块中定义的过程，可以不经整个编译而取出这些过程。这样我们就可以产生下拉清单，对象浏览器（Object Browser）可以显示项目的成员而不需要实际编译，等等，即使过程体中有语法错误或编程错误，我们仍然可以取出所有这些模块成

员。然后要经历另外两个阶段，可以找出所有声明的变量，包括局部变量。然后进入关联阶段。最后阶段实际发送产生的代码。

Ken：是在要求时还是在有人要求编译代码时？

Craig：在VB4.0和Access 95版VBA中，它根据用户选择的选项，可以两种方法进行。一种办法是一旦运行任何过程，我们就生成所有模块的所有代码，生成整个项目的代码。如果打开Compile On demand（需要时编译）开关，则在运行时动态产生外部码。这样做的原因是这种情况下要实际运行代码时才能执行或正确编译代码。

Ken：这个特性很少用，最好让它关掉。

Craig：下一版中将加进Compile All菜单项目，可以强制完全编译。当然，Access中一直有这个项目。

（5）VBA存储

Ken：VBA到底存储什么？用户输入并不存储，会消失掉，是吗？

Craig：是的，VBA主要存储两个东西。一个是我说过的操作码流。这是源码的分析表示，包含名称表的指针。名称表中有提供的所有名称。

第二个主要东西是实际编译码。对于编译代码，我们把编译码和源码一起系列化到存储中，这就是前面提到的外部码流。

此外，我们存储装入和运行外部码流所要的运行环境结构。我们系列化资源名称表，用于构造和解散对象和结构等复杂的数据类型。我们还系列化公共项目表，以便回答诸如“能否提供所有公开过程清单”之类的问题。我们系列化装入时进行的安排，根据装入的地点，必须安排跳转地址。我们实际使用所有过程的本地码项目。调用Basic过程时，即使是个P代码，也要生成调用的本地码端头。这个端头交接并调用工具，传递P代码的指针。

Ken：至少对于有些产品，可以选择是否保存编译码。在Access中只要改变一些码，即可放弃编译状态。

Craig：是的，编译取决于调用。Access中可以选择编译一切并使一切强制为编译状态后再保存。但一般来说，用VBA作为宏语言时，新用户并不知道或在意我们随时编译一切。记录宏、记录代码、执行代码，这就是许多用户所要的一切。

Ken：问题是动态编译和编译之后再运行编译码的速度相差很大，至少在Office 95版产品中是这样，特别是Access中。如果不事先编译，则代码边运行边编译非常慢。

Craig：在Office 97产品中，请求时编译已经大大改善，而且编译只发生在首次调用函数时。你说的不错，但这种差别只在到达一定阈值时才看得出来。如果只有100行代码，则编译起来很快，没必要预编译。

Ken：但对于几千行VBA码的大程序……

Craig：对于用Access写成的完整应用程序，这不成问题。我们还提供了Compile All和Save All Modules菜单选项，复杂程序开发人员可以编写保存优化码的应用程序。

Ken：关键字是不想做时可以不做，可以在需要时再编译。

Craig：是的。我们还存放一些编译的中间状态信息，所以进行部分反编译时，不必将操作码整个重新编译，而可以从中间编译状态开始进行完整的编译。反编译实际上相当灵巧。用户编辑某个函数时，我们寻找他们所作的编辑类型，有时根本不作反编译，只是现场修改

代码或只是删除该过程的外部码而不必退出“声明状态”，可以迅速地重新生成过程的外部码和继续执行。如果编辑是所谓粗编辑（例如修改公开全局变量），则必须将整个模块反编译，因为要重新生成所有公共成员，即要在声明状态之前退出。

Ken：VBA码存放在哪里？

Craig：取决于具体应用程序。应用程序传递一个OLE Istorage接口，我们写入这个Istorage。在VB中，我们用类似的接口使VBA可以在产生执行文件时系列化成PE EXE格式。Office中，我们系列化成文档文件。

Ken：是用户存放的某种文档吗？

Craig：是的。这一切都在OLE Istorage接口进行。

（6）VBA与本地执行码

Ken：VBA能否建立Office应用程序的本地编译码？即我们能否建立可以发布的Excel执行码？

Craig：理论上可以，这只是代码，在将来Office产品中有可能建立，但Office 97版本中没有。这里问题很多，要考虑如何用代码所配置的具体Office文档包装代码。最终的问题是我们是否重视资源，是否在Office应用程序中提供更好技术以达到实用解决办法？是否重视新的面向对象和包装特性，更快的对象访问性能？生成本地码无助于解决这些问题。

Ken：是的。但人们总想问：“为什么无法建立Access本地执行码？”我的反应是：“为什么要建立，有什么用？”

Craig：去年我们对执行P代码所要的时间量进行了一个试验。我们在一个执行每个P代码的小执行码中用P代码工具计算所花的时间量。

Ken：这是唯一可以变成本地码的东西。

Craig：我们自己的运行环境也是用C语言编写的，所以已经是本地码。例如调用Sin函数时，执行的不是P代码，而是本地码。唯一可做的事是将这些外部码变成本地码。这样，就可以不跳到这些P代码工具而只执行本地码。

我们发现，平均而言，只有大约5%的时间用在P代码内部！即使在看来相当高性能的情况下，例如在字符串接合的循环中，我们也只用不到50%的时间在P代码中。其它时间用在调用Windows系统分配和调用自动化进行字符串处理，而这些已经编写成本地码。

Ken：通常情况下人们的时间只是用在接口上、单击按键或输入文本，等等。

Craig：是的，这5%的数字表示5%的时间不在闲置循环中。标准情况下，95%的时间用在调用数据访问对象、调用窗体对象和Windows API、取得属性。所以大多数时间用在闲置循环中，执行P代码的处理器时间只占实际执行代码时间的5%。假设本地码可以使P代码的执行时间减少到0（显然，这是不可能的！），也不过将应用程序性能提高了5%。这并不是说本地码不重要。对于计算量很大的数组排序或其它受益于本地码性能的计算操作，本地码是很重要的，但对于端对端情形，本地码能提供的改进微乎其微。

这一点可以通过比较VB4和Delphi看出。也许你想不到，有些方面VB4性能更佳。结果取决于比较是针对实际应用程序（如客户机服务器程序）还是运行需要大量处理时间的算法（如执行Sieve）。比较中分析人员采用他们编写的客户机服务器应用程序。由于我们的数据访问技术在那种情况下更快捷，结果VB比Delphi更快。所有这些情况与P代码和本地执行