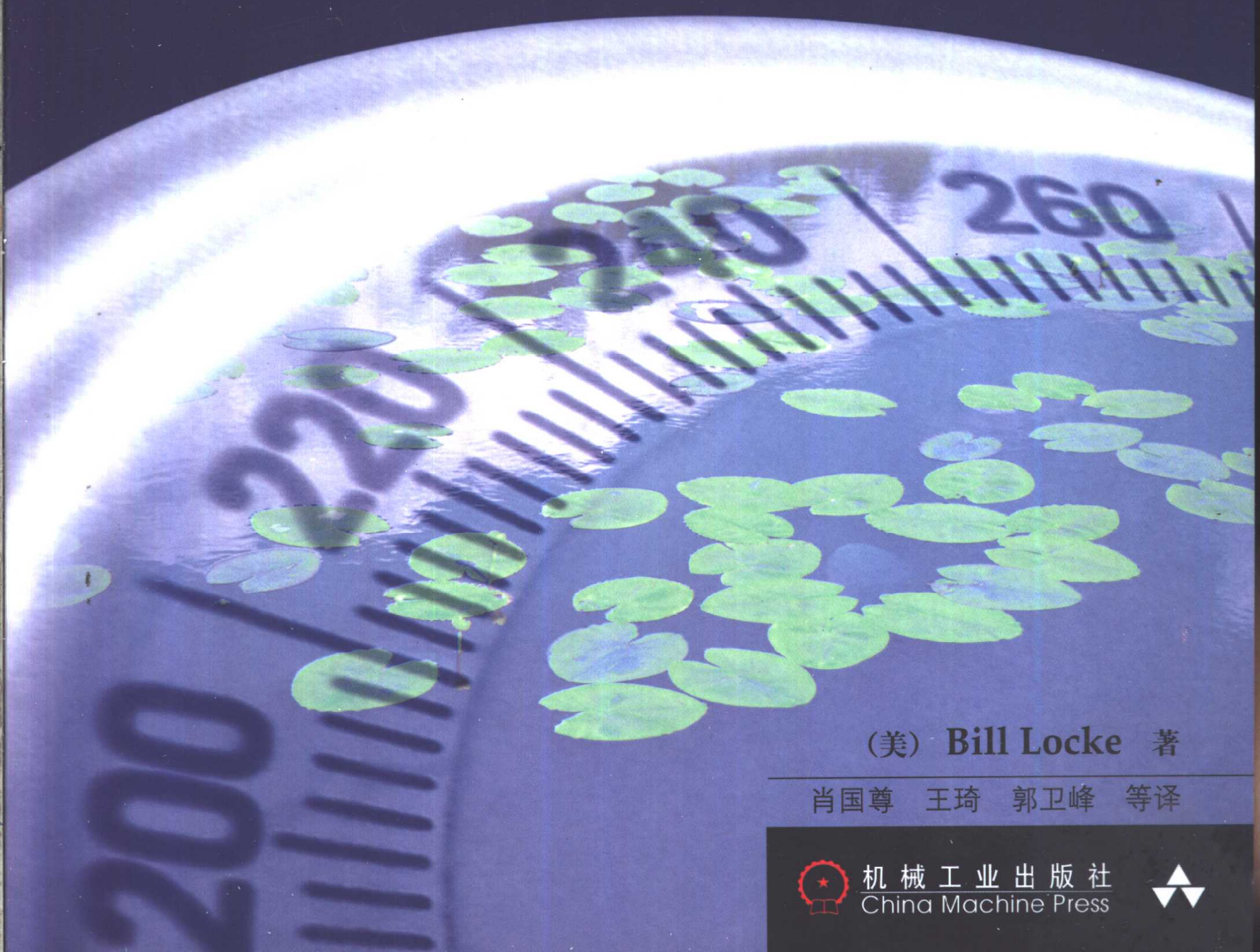
HZ BOOKS

Visual C++ for
Visual Basic Developers

开发人员专业技术丛书

Visual C++ 程序设计

——VB程序员指南



(美) Bill Locke 著

肖国尊 王琦 郭卫峰 等译

机械工业出版社
China Machine Press

开发人员专业技术丛书

Visual C++程序设计

——VB 程序员指南

（美） Bill Locke 著

肖国尊 王琦 郭卫峰 等译

机械工业出版社
China Machine Press

本书是为要掌握C语言的VB程序员编写的。全书首先介绍了VB和C语言的历史和异同点，然后分章介绍了C语言和C++基础、C++类、C++和Windows、C DLL、组件与控件以及C#。作者在介绍每一项内容时，都与VB进行了一一比较。另外，作者还对两种语言之间的互操作进行了介绍。

本书适合于要学习C语言的VB程序员，以及那些希望了解两种语言区别的开发人员。

Bill Locke: Visual C++ for Visual Basic Developers.

Authorized translation from the English language edition published by Pearson Education, an imprint of Macmillan Computer Publishing U.S.A.

Copyright © 2002 by Pearson Education.

All rights reserved.

Chinese simplified language edition published by China Machine Press.

Copyright © 2002 by China Machine Press.

本书中文简体字版由美国麦克米兰公司授权机械工业出版社独家出版。未经出版者书面许可，不得以任何方式复制或抄袭本书内容。

版权所有，侵权必究。

本书版权登记号：图字：01-2002-3597

图书在版编目（CIP）数据

Visual C++程序设计——VB程序员指南/（美）罗克（Locke, B.），著；肖国尊等译. -北京：机械工业出版社，2002.10

（开发人员专业技术丛书）

书名原文：Visual C++ for Visual Basic Developers

ISBN 7-111-10950-3

Ⅰ.V… Ⅱ.①罗… ②肖… Ⅲ.C语言-程序设计 Ⅳ.TP312

中国版本图书馆CIP数据核字（2002）第070383号

机械工业出版社（北京市西城区百万庄大街22号 邮政编码 100037）

责任编辑：张金梅

北京第二外国语学院印刷厂印刷·新华书店北京发行所发行

2002年10月第1版第1次印刷

787mm×1092mm 1/16·18.75印张

印数：0 001- 4000册

定价：39.00元

凡购本书，如有倒页、脱页、缺页，由本社发行部调换

译 者 序

Basic是一种拥有悠久历史的计算机程序设计语言，早在DOS时代，就有了一大批在计算机上使用Basic进行程序设计的程序员。但是，不管怎么样，Basic对程序员而言，总不是十分“专业”的语言，我们很难使用Basic来做一些底层的开发工作。而C语言就完全不同，我们既可以随心所欲地进行界面设计，又能以一种便捷的方式操作内存和硬件。随着Windows的推广，C语言基本上成了一种标准语言。

尽管Visual Basic的推出，改变了很多人对Basic的看法，用它生成用户界面非常简单，而且其在界面后面放置代码的方法也非常的简易，可能除了那些在.NET中能被管理的代码，现在还没有哪个工具能跟它比。但是Visual Basic还是只能编写一些简单的程序。

因此，对VB程序员而言，要学习C语言就存在一个语言和开发平台的转型问题，从基本的标识符、关键字、操作符、表达式、语言控制流程到使用面向对象编程和开发动态链接库，VB程序员似乎一切都要从头开始学习。

其实，语言与语言之间有很多东西是相通的，虽然各种语言的侧重点不同，但大多数的功能总是相互对应的。只要我们使用了好的学习工具，从一种语言转向另一种语言并不难。而本书的目标就是以一种一一对应的方法带领VB程序员掌握C（包括基本C、C++和C#），只要按照本书逐步学习，相信读者能够很快掌握C。

本书由肖国尊负责组织翻译，参加翻译工作的人员除封面署名外还有郭卫锋、柳林、李化、周玉梅、陈芝生、陈彦海、胡艳玲、丁滟、李满朝、左亚利、孙文明、廖建华等。全书最后由肖国尊统稿。

在翻译过程中，我们对本书中出现的术语都进行了仔细的推敲和研究，然而有些内容在我们的开发工作中也很少遇到，加上时间仓促，译者水平有限，疏漏和争议之处在所难免，望广大读者提出宝贵的意见。

肖国尊

2002年6月

前　言

本书是为中级到高级Visual Basic程序员编写的。本书不是为高级C和C++程序员编写的，但是中级C程序员可以从本书中受益。许多章对于不知道Visual Basic的人会觉得单调乏味，因为本书将重点对照Visual Basic代码。

我想问Visual Basic程序员一个简单的问题：你想要用C、C++或者C#编程吗？这些语言是否列入你希望掌握的语言工具之中？回答这个问题可能比你预料的要难。

为什么应当关心这些语言呢？简短的回答就是：对于希望自由自在编程的程序员而言，C语言系列是他们的最佳选择。你可以考虑在编程过程中希望做的任何事情，而使用C、C++和C#总可以找到完成这些任务的途径。只有很少的语言系列可以满足这样的要求。假如是那样，那么C语言是否是你愿意了解的语言呢？

较长的回答与Basic和C的发展有关系。在过去，Visual Basic在设计用户界面时具有明显的优势。在Visual Basic中设计用户界面比在C或C++中设计用户界面要简单得多。现在，C#和Visual Basic（以及许多其他语言（就此而言））使用相同的设计器并共享相同的用户界面设计功能。在本书中VB的这些优点都没有了。

Visual Basic具有其他性质，这些性质使得它较容易开发和使用最终用户程序，对于无经验的程序员而言尤其如此。这些优点也没有了。Visual Basic的最新版本比以前版本要复杂一个等级并且很难掌握。从Visual Basic 6迁移到最新的Visual Basic语言可能同从Visual Basic迁移到C#一样难。

微软说，现在选择一种语言就是选择一种生活方式。我想语言的选择包含更多的含意。C（或者由C派生过来的语言）一直是微软的内部开发语言。首先是C，然后是C++，现在是C#，这些语言在微软广泛使用。这对于语言有何影响呢？其影响效果是你可能预料到的。如果微软程序员需要C、C++或C#中的某种特性来完成工作，那么一般情况下他可以获得这个特性。那么，现在，你要面对的是C#比Visual Basic要强的场合，但因为设计器的缘故，C#一样容易使用。

例如，C#可以用来编写难于管理的代码，而Visual Basic则不能；C#可以覆盖操作符，而Visual Basic不能。C#可以直接访问内存，而Visual Basic不能。我们可以继续研究这些方面的内容，但我想要讲的是：微软使用C#作为.NET技术中的首要开发语言。

我是在劝你现在就转到C#吗？不，我在劝你学习一种已经处于这个领域顶层许多年的语言。本书将着重于组件以及Visual Basic与C之间的差别。我将向你展示如何编写例程和对象，你可以使用这些例程和对象来提高Visual Basic技能。同时，本书中提供了足够的信息，你可以根据需要彻底转到C#。我的主要观点依旧是：你希望知道这些语言的足够信息来提高Visual Basic技能。如果你决定转到C系列，那么这是开始转换的好途径。

下面介绍本书的主要内容：

- 第1章："历史回顾"——本章讲述简短的VB和C++的历史课程。如果你从一开始使用的就

试读结束：需要全本请在线购买：　www.ertongbook.com

是Visual Basic，那么这一章可以作为复习用。然而，我想，从这一章可以很好地看出“你是从哪里来的”，这样就可以更好地明白“要往哪里去”。

- 第2章：“C语言基础”——该章讨论基本C（不存在C++扩展的C语言）。C语言的构成和C与Basic之间的差异。该章将使用示例来示范这种差异。该章是C++和C#二者的基础，因为这两种语言是建立在C概念的基础之上的。
- 第3章：“C编程”——这一章继续讨论基本C编程。该章为理解C和C++中的变量类型打下基础。还介绍了C中的常见程序设计结构，如判定和循环。
- 第4章：“C++基础”——该章讨论C++并讨论C同C++之间的基本不同点。C++编译器对该语言增加了不必直接与类相关的特性。我们将介绍这些变化并使你对于理解C++类有准备。
- 第5章：“C++类”——该章着重介绍C++类。C++类是程序员对实现继承入门的最常见的路线，并且我们将花一点点时间来理解实现继承。继承是你使用C++或者C#技术编程时希望理解的一个概念。
- 第6章：“C++和Windows”——该章简介WIN32程序中的底层组件并且包含示例C代码，这些示例代码显示如何与这些组件进行交互。如果你想使用C或C++在WIN32中做大量的工作，那么应当充分掌握这些基础知识。
- 第7章：“C DLL”——该章将介绍较为老但仍然有用的技术——基于函数的DLL。更为新式的技术依赖的许多特性可在简单DLL中实现，你将学习如何编写它们。该章还介绍了Visual Basic与C之间的基本类型转换。
- 第8章：“C DLL示例”——该章包含第7章的示例。它包括派生类和回调的示例。它还讨论关于使用一个类型库来简化对DLL的访问。
- 第9章：“组件和控件”——该章简要介绍使用不同的技术创建组件。该章讨论COM技术与.NET技术之间的差别。然后，深入介绍控件技术。它详细讲述了ActiveX控件开发。
- 第10章：“C#基础”——该章将引导你了解Microsoft提供的大多数新式语言之间的差别。我们将讨论C#的基础。C#从下到上都是为.NET技术设计的，并且有可能成为首要的MS开发语言。
- 第11章：“应用C#”——在最后这一章中，你将深入了解C#并使用C#建立组件。我们将示范C#与前面的开发语言之间的实际差别。

如果你一直都在考虑使用C编程，那么现在是开始的时候了。从Visual Basic转到.NET技术存在一定的难度，但可以扩展你的语言选择。

目　录

第1章 历史回顾

本章内容：

- Visual Basic和C++的历史
- Visual Basic的作用
- C++的作用
- C#

显然，学习使用C语言编程是我们的目的，并且，还要学习用C、C++和C#编程以支持Visual Basic。同样，对于Visual Basic的历史回顾也非常倾向于工具开发。所以，在开始探索C语言之前，先来讨论一下是什么引发你学习C语言编程的。从历史中可以学到很多东西。例如，我们知道，不管现在的开发环境怎样，这种开发环境总是会改变的。只要有可能，应该尽量使用最底层的通用标准来编程。

我的意思是要使用基本C代码。在现在的32位控件中，运行的还是基本C程序代码，这些代码是以Visual Basic 1和2的框架编写的。它们在做了较少的修改后，便可以像在16位系统中一样，在现在的环境中运行。

例如，想像一下完成一个图像设备接口（Graphic Device Interface，GDI）编程的几种方法——基本C、MFC、WFC、Visual Basic画图例程，还有现在的.NET类等，而且我相信还有一些其他的方法。

这些技术是从基础技术中抽象出来的。为什么这样说呢？一般来说，我们会对一项技术进行抽象，以使得其更容易使用，或者可以在不改变使用抽象层编写的程序的情况下，来改变基础技术。

然而，如果持续改变抽象层，那么，你将失去抽象的一个目的。我确信在某种意义上，基础技术（Windows）肯定会对以前遗留的代码做很多的改变。但是，在这里，我要告诉你的是，那些抽象层将比基础系统改变得更多。

我为16位的VBX控件用基本C代码写过GDI代码。然而，当OLE控件出现后，又面临着使用MFC来编写GDI代码的压力（相应的还有很多其他方面的改变）。我接受了这种改变，但是，接着，当ATL变成编写控件的最佳选择时，我们又面临着改变。

在ATL的很多实例中，我回到16位C代码并且对它们进行了转化，而这样做比使用MFC代码编程更容易、更适用。我不会期待着使用下一个程序包（.NET）来进行编程，很可能将继续使用C代码编程。

注意 .NET指的是微软公司最新推出的开发环境。在这个系统中，代码在通用语言运行时（Common language runtime，CLR）的基础上运行。底层平台的功能实现用使用CLR的某种语言（通常是C#、Visual Basic.NET或者是C++）编程封装到类库中。

然而，我会放弃代码跨平台执行的任何机会。更重要的是，我将放弃.NET中管理代码提供的优点，因为这些优点需要以重写代码作为代价。

CLR可能会给程序员一个很好的跨平台的编程环境。但是，这样做也很可能面临着只能在跨平台端口中成功使用的限制，就像现在这样。为了利用相应平台提供的服务，程序员很可能需要继续在代码运行的平台上编程，正如现在这样。所以我的C代码直到现在很可能是非常安全的。

在这种跨平台的问题上，Internet又提出了新的问题。当在Internet上进行编程的时候，你要将很多代码传送到浏览器中，而浏览器将与你的平台无关。你写的服务器方代码（通过浏览器将UI传送到用户方）实际上是不需要平台无关性的，它在一个专用服务器上运行，这个服务器是基于NT的，或者是基于UNIX的，或者是某种UNIX的变种，在服务器上对这个平台进行编程，并且生成普通的HTML（理想情况下是这样的）。

当然，实际上，在大多数情况下我们不仅仅是直接使用HTML，还可能使用DHML、脚本或者一些客户方控件来达到需要的目的。在很多情况下，我们限制能够访问我们站点的浏览器类型并得到全部功能，然而这比较复杂。

我们正处在发展和变革的中央，在Internet技术上还有很多的工作要做。然而，同样，在WIN32中也还有很多的代码需要完成。这正是我们很多人的立足点，它将继续发展，成为一项重要的技术。

1.1 Visual Basic和C++的历史

这两种语言的历史可以追溯到很久以前。当然，我们不是在研究圣经，而是计算机，相比较来说，计算机的历史就很短了。Visual Basic和C++在DOS时代就产生了（我个人在DOS方面的主要经验是Basic，而对于DOS中的C语言编程，我做得比较少）。

这两种语言的竞争可以归结到这个环境中。C语言是一种专业语言，而Basic语言是孩子在编程游戏中使用的。这种观点随着基于解释的Basic的出现而有所改变，GWBasic成为了一种普遍流行的变种。我们过去常常称其为Gee Wiz Basic，这种描述更加准确。从那以后，Basic开始不断赢得人们的注意。

当Windows出现后，再一次成为C程序员的天下，任何Basic程序员都不能够在这里编程，直到Visual Basic出现，才进入Visual Basic 1时代。

1.1.1 16位

我们需要从16位系统开始，因为Visual Basic是从这里开始的。从Visual Basic 1到Visual Basic 3都是运行在16位系统上的，Visual Basic 4是第一个32位的版本。

1. Thunder编程系统——Visual Basic 1

Thunder编程系统（是Visual Basic 1工程的代码名称）迈出了一大步。我记得当时听到人们议论说是一种时尚、一个玩具，或者其他类似的说法。但是实际结果是它既不是玩具也根本不是一种时尚。当时，我在MicroHelp工作，我们给出了Visual Basic的一些最初的库。由于当时这些东西都非常新，以至于人们认为是我们将那些工具整合到工具条中去的。

试读结束：需要全本请在线购买： www.ertongbook.com

Visual Basic 1有很多的缺点——只能使用伪代码（P-code），工具集很有限，窗体（Form）以二进制的格式保存，没有内插式（Add-in）附件。但是，它是很重要的第一步。在这个产品中，提出了定制控件的概念，而这是现在我们知道的组件编程的开端。

我还记得在MicroHelp的最初的软件包中比较普遍的控件。Visual Basic最初的列表框仅仅是单选列表框，而现在有了多选列表框。Visual Basic最初的滚动条要等到鼠标松开以后才改变值，而现在的滚动条都有连续的反馈。时间改变了这一切。

组件编程的概念导致了最大的“外接编译器”段的产生，而它曾经同编译器有过相同的经历。

除了组件编程的概念，Visual Basic 1还向Basic程序员提供了事件驱动编程的概念。到那时为止，除非你是一个C/SDK程序员，否则没有多少机会进行事件驱动编程。

对于从DOS系统刚刚进入Windows系统的程序员来说，事件驱动编程是个新鲜的概念。但由于Visual Basic的存在，进入到这个编程环境的人数非常多。突然之间，曾经梦想着在Windows系统中进行程序设计的程序员都实现了他们的目标。这不能不说是一个变革。

VBX是最初的组件。从本质上说，同现在的组件相比，它们是非常简单的，但是，VBX仍然有很多优点，例如它们非常容易编写，它们提供了实例数据，它们提供了共享数据。VBX是最早提供通用属性和事件的编程元素。VBX是一个动态链接库（DLL），它具备了16位DLL的所有优点和缺点。

在现在的32位系统中，DLL装载入可执行程序空间中。当然，16位的DLL只装载一次。数据段（Data Segment，DS）和堆栈段（Stack Segment，SS）的方法比较复杂，但是具有其自身的优越性。你可以把它们作为一个非常快速的进程内通信方法。只要能确定被装载代码的一个实例，而且有一个数据空间正在程序间共享，你就可以进行代码的优化。16位的DLL实际上最能够说明DLL的最初作用。DLL的驱动思想是节省空间（不管是在内存中还是在硬盘中）。在现在的环境中，我们不断地增加了操作DLL的复杂性（不管我们称之为VBX、OCX、COM对象或者是其他什么东西）。而现在在微软的.NET中，又回到了以前的时代，库不需要注册，而驻留在EXE文件中。

为了解决“DLL恶梦（DLL hell）”带来的问题，又增了一些复杂性。大多数的复杂性实际上不是必需的。为了解决这个问题，第一步就是在EXE文件对应的目录中安装相应需要用到的私有DLL。在16位系统中，这并不能够实际解决DLL恶梦问题。这是因为只有一个DLL的拷贝装载到了内存中，相应的就只有一个拷贝存在。于是就会发生这种现象，只要有其他程序在运行相应DLL的某个版本，那么就会导致这个程序运行失败。

例如，一个16位的程序可能会装载另一个程序正在使用的DLL。但是可能这个DLL是一个旧的版本，和第二个程序不兼容。在第一个程序还在运行的时候，如果第二个程序就试着装载，那么就不能保证在内存中是正确的DLL，这是因为第一个程序已经装载了另一个版本的动态链接库。然而，32位的系统就根本不会产生这种问题，这是因为DLL装载入程序的代码空间中，所以两个版本的DLL可以同时存在，它们也是这样保存在硬盘上的，这样，每个程序就可以装载相应版本的DLL。

在32位系统中，把DLL和EXE程序放在一起是可行的。因为DLL被装载入EXE程序的代码空间中，不同版本的DLL将不会互相冲突。由它带来的注册、复杂性和问题用于解决一个问题，

现在这个问题已经不复存在了。

在进入到32位代码之前，我们依然还要体验一些16位编程的经验。

2. Visual Basic 2

对于Visual Basic 1来说，Visual Basic 2没有很大的改进，但是它提供了一些中断控件。前面提到的DLL恶梦给了控件开发者非常大的打击，我相信正是从那个时候开始，微软决定要对任何事物都进行注册。每个DLL都已有一个拷贝，并且必须放在系统目录中。从此，事情变得可怕了。

另外，Visual Basic 2增加了在控件中操作高保真彩色灰度的能力。我们称之为支持256色，但是，实际上是为包含了一幅图片的控件提供了一种操作调色板的方法。显然，这时操作的颜色不止256色。我们这时仍然只能使用位图、图标和图元文件，并且要维持很长一段时间。

Visual Basic 2也增加了亮度控件。所有用于这些控件的消息实际上都是通过窗体或控件来实现的，亮度控件就放置在它们上面。由于亮度控件并没有窗口操作，因此，这使得对它们的管理变得有些错综复杂。它们虽然有用，但我还是不愿意使用它们。

在Visual Basic 2中支持Visual Basic格式（VBFormat），这使得在VBX中对C代码调用变得可能。这个功能可以使用Visual Basic字符串格式化“引擎”对字符串进行格式化。这在很多控件中都是非常有用的，但是只能在Visual Basic中使用。

现在回头看看这一新的改变——允许对控件进行中断，如果你的控件使用的是Visual Basic 2的某个函数，那么就需要检查Visual Basic的版本；如果使用的是Visual Basic 1，那么它还会以某种方式退化。这看上去可能有些琐碎，但是这是与当时这个控件完成的工作有关的。

另外，还有一些属性是只在Visual Basic 2中有用，这些并不是很重要，除非你想在几个月以后对控件增加一些属性，并且希望这些属性在未来的Visual Basic版本中还能够正常使用。但是，这只是Visual Basic 2的属性，在Visual Basic 1中，必须将这些属性“取消”才能够保证正确运行。

3. Visual C++ 1

Visual Basic 1和2的控件是用DOS编辑器和一个C++编译器来开发的（至少对于我本人来说是这样的）。我从命令行进行编译，并且使用CodeView来调试。然而，Visual C++ 1.5改变了这一切。

在这里我要提到Visual C++ 1.52和其对于通用控件的支持（虽然这个功能还很弱），这是因为它与组件编程的思想已经非常相近了。它之所以值得提起，只是因为它保持了Visual Basic 1.0原始的定制控件规范不变，即使在今天也是这样。C++ 1.52可以在当前的MSDN发行版光盘中作为当前的16位C++找到。

几年以后，你很可能已经忘记了那些不被Visual Basic 1兼容的程序，但是不会太多。Visual Basic 1.0并不是很久以前的历史。在Visual Basic 2发行后，Visual Basic 1仍然非常流行。同时，微软公司决定增加对C++的VBX支持。当时，他们决定只对Visual Basic 1.0规范进行操作，但是没想到最后却变成了Visual C++的最后一个16位版本。所以在VC 1.52中存在*Visual Basic 1控件*规范。就在今年，我还修复了一个在VBX中关于Visual C++ 1.52的问题，我现在依然可以写出那些代码，但是这样做没有太大的意义。这是第一个Visual C++，也是最后一个16位版本。

4. Visual Basic 3

Visual Basic 3增加了数据库支持，这是一大进步。虽然比起SQL Server或者其他的高级数据库引擎，它并不是真正完全支持，但是，很多人还是愿意使用它。

就涉及到的通用控件，我们第一次尝试了数据绑定，这是非常初级的。虽然大多数为Visual Basic 3编写重要代码的人不使用数据绑定，但是我们为控件编写了代码。

大多数在Visual Basic 3中进行程序开发的人仍然使用相当多的第三方支持工具，我记得的有UnInstaller，这是MicroHelp开发的一个终端用户实用工具。这个实用工具的16位版本使用了用C语言编写的13个定制VBX和一些DLL。其32位版本则大不一样，因为其开发环境，需要更多的工具。

就我看来，Visual Basic 3是所有的Visual Basic中最好的16位版本，如果你必须在16位环境下编程，那么这是可供使用的最好的Visual Basic版本。在Windows 2000之前，它都能够在多个平台（WIN9X/NT）上运行，甚至比某些32位的代码还要稳定。这是因为在NT中16位的仿真做得非常好，但是，对于32位的情况，在WIN9X和NT之间有些实质性的不同，这就在32位程序中产生了一些兼容性问题。

5. Visual Basic 4

就16位编程而言，Visual Basic 4是一个倒退。虽然它增加了对16位OLE的支持，但是我不能确定为什么微软公司变得糊涂了。因为微软公司基本上已经淘汰了16位平台，所以，16位的Visual Basic 4，特别是其支持的一些工具，并没有得到广泛的测试。

所以，Visual Basic 4从来就不是16位开发的一个重要工具。由于它是重新编码，并且是建立在OLE的基础上，所以在很多情况下，与Visual Basic 3代码比较起来就显得非常慢。

16位Visual Basic 4支持VBX，它同样支持16位OLE控件（OLE Control，OCX），这是一项与Visual Basic4-16同生同灭的技术。在Visual Basic 4中VBX规范没有任何改变，所以最新的VBX规范是Visual Basic 3版本中的。

由于这是“当前”的16位Visual Basic，所以16位的OCX和来自Visual Basic 3的VBX是当前16位下的技术。因此，管理整个过程的16位Visual Basic控件规范是第二版的VBX。所有其他的附加规范现在都支持：VBX1.0可以在Visual C++1.52中使用，VBX3.0可以在Visual Basic 4中使用，而16位的OCX可以在Visual Basic 4中使用。

6. Visual Basic 4——第一次尝试

Visual Basic 4的重要之处在于它能够运行在32位的环境中。但是还存在一些问题，同样，在支持32位OCX方面也还有问题。即使如此，它仍然是一个重要的成就，毕竟在Visual Basic所有的版本中只有这一个既支持16位又支持32位的系统。

我不能够确定为什么微软公司会开发这样一个产品，使它既支持16位又支持32位版本。这样不但对微软公司是很困难的，对于第三方服务提供商也是很困难的。两个版本中的任何东西都要构建并测试，这对于16位和32位系统都很麻烦。

不过微软的方向是很明确的，那就是32位版本，但是这就更令我费解，为什么还要在16位版本编译器上面费这么大的周折。Visual Basic 3是一个非常强大的16位编程环境，即使在今天，我仍然认为它比Visual Basic 4的16位版本要好用。使用Visual Basic 4的惟一原因就是在Visual

Basic 3下有些工作不能够完成，而在Visual Basic 4的16位版本中可以完成。

这其中的一个项目就是OLE自动化。在Visual Basic 3中没有提供这个功能，而在Visual Basic 4中有。如果你想要在16位系统中运行一个自动化可执行OLE程序，那么Visual Basic 4可以提供这个功能。你能够使用16位自动化程序来完成的一个功能就是构成一个“穷人”（Poor man's）的转换程序（这是一种从32位系统中调用16位DLL程序的方法）。系统并不关心自动化可执行程序是16位的还是32位的，所以，你可以从32位系统中调用16位的OLE自动化可执行程序。

如果你有16位的代码，并且需要从32位系统中对它们进行访问，那么可以把它们打包到Visual Basic 4 OLE自动化可执行程序中，并且用一个32位的程序对其进行调用。当然，这不是最快的调用例程的方法，如果你可以使用实际的转换程序，那么情况将好得多。不过，需要注意的是，转换程序是与平台相关的，而OLE自动化程序没有这个限制。

你可能有使用Visual Basic 4-16的理由，但是这并不普遍，而且这个平台存在不少的问题。我一直建议人们不要将Visual Basic 3的程序移植到Visual Basic 4-16中，除非他们有非常充足的理由，如我前面提到的那个理由，否则，还是直接转到32位系统比较好。

1.1.2 32位

WIN 32编程环境是当前我们非常熟悉的一个环境，16位Visual Basic中的技术只得到了非常少的保留。

1. Visual Basic 4

前面我提到过，第4版Visual Basic的重要之处在于它能够运行在32位的环境中，比起第3版的那些优点来，第4版的编译器非常平凡。

然而，Visual Basic 4却能够让我们编写32位的代码，这是Visual Basic在32位环境中的第一次尝试，当然，它仍然是一个伪代码编译器，几乎没有什么新的特性能够吸引人们来使用它。

2. Visual Basic 5——Visual Studio 5

随着Visual Basic 5的推出，相应地推出了一个重要的工具，使得现在可以对本地代码进行编译，并且有足够的工具来支持这个环境，整个开发系统非常的完整，而不需要购买其他的工具。在Visual Basic的这个版本中，ActiveX控件开发得到了应用，这使得Visual Basic程序员能够编写ActiveX控件了。

在Visual Basic中编写ActiveX控件始终没有流行，这是很难理解的。不管你是在编写商业程序或者只是在家里随便编程玩玩，将代码进行封装都是一个非常好的方法。在Visual Basic 5推出以后，我如果不使用通用控件，就不能够编写出任何出色的定制项目。也许在.NET技术中这种编写控件的应用将会最终流行起来。

在推出Visual Basic 5的同时，微软公司将一系列的开发产品捆绑在了一起。Visual Studio 5就是第一个提供了一套完整解决方案的系统，在其中包括了一个C++编译器。这就是在每个包中都有的一个入门级的C++编译器。既然你有了这个工具，那干吗不使用它呢？Visual C++ 2.0是第一个32位的C++编译器，但是第4版才是我积极使用的。你可以从版本4中看出它还有一些32位的问题，而版本5第一个捆绑到一套新的Visual Studio中。

使用Visual C++5编译器或者开发环境的方法中有很多都很平常，但是也有一些非常有挑战性。

有一些使用Visual C++环境平常的方法，如使用Find in Files（在一个文件中查找某个单词或某个短语）。因为此功能甚至装载Visual C++ 5编辑器也是值得的。在这里，将在文件中查找文本字符串，而且可以将查找范围限制在选定的文件类型之内，甚至可以对子目录进行查找。查找结果将提供一个清单，在其上面列出了这些文件中对应符合情况的行，你只要双击其中某行就可以将相应的文件装载入编辑器，而且会定位在相应的行。

Visual Basic直到目前最新的版本才提供了编辑器的宏记录能力（高级编辑的功能）。如果选择了正确的开关，你甚至可以在Visual C++中调试Visual Basic程序。

使用C和C++语言更加具有挑战性的方法将是本书的主题。Visual Studio 5是第一个使这些变得可行的系统，这是因为大多数程序员都具有这个版本的所有编译器。当然，大多数的Visual Basic程序员仍然不使用C++编译器。

3. Visual Studio 6

Visual Basic 6在数据访问方面进行了重新编写。虽然它也以向导的形式增加了一些控件和工具，但是对于Visual Studio 5来说，其相应的开发环境并没有太大的改变。

在Visual Basic 6中增加了报告的功能，但是这并没有给人们留下什么印像，微软公司在.NET技术中又回到了利用Crystal来报告，这也可以证明这一点。在Visual Basic 6中对记录集合和弯曲网格增加了分层结构，这是非常简洁的。我发现了分层记录集合的很多用途。

创建数据源也是非常有用的。能够中断记录集合的连接，通过过程调用进行数据集的传送，甚至跨越过程，这些都是很大的进步。这就有可能使得创建多层代码变得很简单。

Visual Studio 6中我最喜欢的改变就是ATL的版本3。这使得在这个技术中对控件的开发变得实用，并且比ATL以前的版本要容易得多。

在Visual Studio 6中还有很多其他的改变，当然，不能都展现出来。重新编译你的Visual Basic 5项目，你会发现这个过程跟往常的业务都差不多。

1.1.3 .NET

微软公司最新的技术改变了所有的规则。只有Visual C/C++能够像以前一样运行。Visual J++没有了，Visual Interdev也没有了，Visual Basic则基于一个类模型（这些都是.NET语言），它已经不是以前的Visual Basic了。因此，这需要比以前的Visual Basic有更多的理解，同样对于大多数代码，需要移植或重写。

当微软公司报告Visual Basic将会使用继承实现时，我认为他们是想和在Visual C++中一样，是可选的。你可以使用C或者C++编写代码，而不是使用.NET语言（它们是基于类的），并且这是能够让它们正常运行的惟一途径。

C#也是这样。从Visual Basic 6转到Visual Basic .NET是很困难的，这个难度很可能跟直接转到C#差不多，我认为很多程序员应该这样做，那就是直接进入C#。

当我们讨论使用一种支持通用语言运行时（Common Language Runtime，CLR）的语言来编写代码的时候，我们实际上是在讨论编写可以管理的代码，并与那些不能被管理的代码比较。可以管理的代码运行在CLR的顶层，之所以这样称呼它们是因为对于程序员来说对象都集中到了一起。而那些不能被管理的代码指的是任何不运行在CLR顶层的代码。C和C++是不能管理的

代码（虽然C++提供了可管理的扩充），而新技术中的Visual Basic和C#则是可管理的代码。

1.2 Visual Basic的作用

Visual Basic改变了我们编写Windows程序的方法，用它生成用户界面非常简单，而且其在界面后面放置代码的方法也非常简易，以至于可能除了那些在.NET中能管理的代码，现在还没有哪个工具能跟它比。虽然Visual Basic是一个绝妙的开发环境，但是有些东西你是不能够用它来设计的。我曾经见过很多人试图使用Visual Basic来设计从没有设计过的东西，结果会怎样呢？一般情况是，如果你设计了新东西，那么它会不稳定，很难编写，而且难以调试。为什么很多我们见到的Visual Basic代码都非常简单呢，原因就在于它只能做一些简单的程序。

就好像你不会使用耙子挖洞一样，使用正确的工具往往能够编写出优秀的代码，当然，有时工具的选择不会带来那么大的差距，但有时却有可能。如果实际情况需要，请不要害怕使用C或C++语言，这正是本书的目标，那就是让这些工具为你服务。

1.2.1 Visual Basic的优点

在.NET系统中那些可管理的代码对于很多语言来说有相同的界面，要比较它们的特征，必须将这些可管理的代码集中到一起。例如对于C#提供的那些特征来说，它更像Visual Basic而不像C++语言。在.NET中，设计环境超越了以前的Visual Basic版本。

1. 绝好的界面设计工具

Visual Basic很可能是最强大的WIN32快速应用程序开发（Rapid Application Development，RAD）工具，由于它能够在几分钟之内完成界面的设计，所以常常成为程序员的最佳选择。.NET使得这一点可用于很多语言，但是C和C++仍然保持不变。

2. 强类型

Visual Basic一直对变量提供强类型的功能。这消除了对指针进行误操作的可能性，因为根本就没有指针，同样，也消除了不正确的类型转换问题，原因也一样，没有类型转换。

Visual Basic中也存在转换，当确信转换合适时，内部转换也可以将某种类型的变量转换到另一种。由于这种变换有可能产生预料不到的问题，所以最好不要进行显式的转换。不要将变量、变量转换与松散类型搞混淆了，这并不是松散类型。至于哪些变量可以转换，有一套非常严格的规则，而松散类型没有。

3. 交互式的调试环境

与界面设计能力相对应，Visual Basic的交互式环境对其发展也起了很大的作用。程序员之所以选择使用它，是因为你可以很快地在其中编写、运行、调试和更改代码，而不需要离开这个开发环境。没有什么能够与之相比。

4. 对第三方工具的支持

Visual Basic一直提供对第三方工具的支持，这些第三方工具提供商从8到10个增加到了数百个。Visual Basic内嵌工具的选择是非常惊人的，而且这种趋势还会在.NET系统中得到延续。

这样做可以使Visual Basic程序员的工作更加简单，你可以使用某个专家定制好的C代码来完成一项特定的工作，这些代码不仅已经编写好了，并且还经过了调试。

试读结束：需要全本请在线购买： www.ertongbook.com