HZ BOOKS

微软 .NET经典原版系列

Microsoft Press

6

Sixth Edition

PROGRAMMING WITH MICROSOFT VISUAL C++ .NET

Visual C++ .NET 技术内幕（英文版·第6版）

（美） George Shepherd David Kruglinski 著

Microsoft .net

机械工业出版社
China Machine Press

微软.NET经典原版系列

Visual C++ .NET技术内幕

（英文版·第6版）

Programming with Microsoft Visual C++ .NET (6 Edition)

（美） George Shepherd
David Kruglinski 著

机械工业出版社
China Machine Press

本书版权登记号：图字：01-2002-5815

图书在版编目（CIP）数据

Visual C++ .NET技术内幕（英文版·第6版）/（美）谢弗里（Shepherd, G.），（美）克鲁林斯基（Kruglinski, D.）著. －北京：机械工业出版社，2003.1

(微软.NET经典原版系列)

书名原文：Programming with Microsoft Visual C++ .NET (6Edition)

ISBN 7-111-11209-1

Ⅰ.V… Ⅱ.①谢… ②克… Ⅲ.C语言－程序设计－英文 Ⅳ.TP312

中国版本图书馆CIP数据核字（2002）第088597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：华 章

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

2003年1月第1版第1次印刷

787mm×1092mm 1/16·67.5印张

印数：0 001-3 000册

定价：139.00元

凡购本书，如有倒页、脱页、缺页，由本社发行部调换

导　读

北京大学　潘爱民

本书第4版、第6版中文版译者

每一本书的背后都有一个故事，对于这本书，这个故事尤其充满了神奇的色彩。这本书的开创者David Kruglinski因为事故身亡，为本书留下了无尽的遗憾。David本身就是一个传奇，他是一名成功的程序员和教育家，也是一名成功的户外活动家，他的人生给我们一种启示：原来技术人员的生活也可以过得如此丰富多彩，技术之外还有那么多东西值得我们追求。当他驾着滑翔机遇难的时候，我想他一定很无畏。这是一种敢于挑战的本色，同时他也谱写了一曲生命的赞歌。

这是一本经典之作。一本书在十年之内出版到了第6版，这本身已经说明了其意义所在。它伴随着Visual C++的发展而发展，几乎成了每一个Visual C++程序员的必备参考书。在所有的Visual C++参考书中，它享有独特的声誉，因为它引导了一批又一批的Visual C++开发人员。

本书的第6版是由George Shepherd续写的（George Shepherd是《MFC Internals》的作者），以George对于MFC和Visual C++的理解深度，以及他的写作经验和教学经验，新版本只会增色而不会逊色。相信读者在读过这本书之后就能够体会到这一点。

在第6版英文影印版出版之际，我有幸再一次介绍这本书。我自从1994年开始使用Visual C++开发环境以来，惟一认真阅读过的Visual C++程序设计书籍仅此一本。在偶然的巧合之下我翻译了本书第4版（英文书名为《Inside Visual C++》，中文书名为《Visual C++技术内幕》），曾经期望能继续翻译下一个版本，可惜失去了机会。我原以为自己和这本书的缘份已尽，幸好在今年新版出来之际，我又有机会再次翻译第6版，并为英文影印版写下这篇介绍文章。

一、本书与Visual C++ .NET

在我手上，有这本书的第2版（中文版）、第4版（中文版和英文版），以及第6版（英文版，中文版还在翻译之中，但我已经可以预见了）。书很沉，很有分量，拿在手里沉甸甸的，当然

更重的是内容。关于本书具体的内容我后面再谈，读者也可以直接参看本书的目录或者Introduction部分。这部分我想说明关于我对本书技术倾向的看法。

Visual C++开发环境包含哪些内容呢？这是被很多人误解的一个问题。有人把MFC（Microsoft Foundation Class）和Visual C++等同起来，也有人把Visual C++和C++混淆，实际上，它们都有明确的分界线，在软件开发过程中，它们有各自的意义。

Visual C++是C++语言的一个具体实现，最核心之处在于C++编译器，它可以将你的源代码转变成可执行文件。因为它是以Windows平台为基础的，所以，利用Visual C++，你可以把C++源代码文件转变成EXE文件或者DLL文件。当然，Visual C++针对Windows平台的许多特性，在标准C++基础上做了很多扩展，这不仅使得Visual C++的功能更加强大，而且使得Visual C++用起来更加复杂，甚至复杂到让人望而却步的程度。另一方面，这也降低了C++在开发过程中的地位。其实，用Visual C++来开发应用程序，理想的知识结构是Win32 SDK加上C++。C++是一门功能强大、内涵丰富的语言，用Visual C++来学习C++语言只会使得事情更加复杂，所以，我建议C++的初学者只利用Visual C++的编译器，忽略掉与平台相关的各种特性（也不要用Visual C++的集成开发环境），以便集中精力掌握语言本身的内容。经过一段时间的学习之后，再全面使用Visual C++来开发Windows应用程序。

Windows的API是C语言风格的，但它不是面向C++的，所以，当你用C++来开发Windows应用程序的时候，往往难以将两者结合起来。不过，如果你要在Visual C++中开发Windows应用程序的话，你已经不用再担心了，因为Visual C++已经提供了一套标准的C++类库——MFC库。从这层意义上讲，我们可以把MFC看作是Windows的C++ API。也正因为如此，在过去几年中，MFC几乎成了Visual C++的代名词。但是，随着这几年技术多元化的趋势，MFC已经不再拥有昔日的辉煌，所以，我们应该清楚地看待技术的发展趋势，以及MFC的优势和局限性。

无疑，MFC很好地封装了Win32 API，它已经历了多个版本，这些版本不仅保持了很好的兼容性，而且MFC总是能够将最新的软件技术纳入进来。例如，MFC的早期版本支持Microsoft Windows 3.1的界面风格，但是，最新的MFC 7.0可以支持Windows 2000/XP的界面风格，这使得MFC开发人员可以以最经济的途径获得新的技术支持和知识上的拓展。

随着近几年网络应用和Internet应用的飞速发展，仅仅桌面应用已经不能满足软件市场的需求，于是大量与Internet相关的技术也应运而生。Visual C++作为Windows平台上最主流的开发

工具，当然也会提供相应的支持。有些支持被纳入到MFC中了，但有些则难以融入到MFC中，或者与MFC的设计本意不一致。于是，在Visual C++ .NET开发环境中，除了MFC之外，又有了其他的内容，其中最主要的是ATL和对.NET的支持。

ATL是一套模板库，它的设计目标是利用C++来开发COM组件。尽管利用MFC也可以开发COM组件，但是由于MFC框架结构的庞杂，所以用MFC开发的COM组件往往比较臃肿，适合于在桌面环境中使用。相反，使用ATL开发出来的COM组件则比较轻量，适合于在网络环境中使用，或者用于服务器平台之中。

在Visual C++ .NET中，一项新引入的开发支持是对于.NET的支持，其中包括对客户端Windows Forms开发的支持，对服务器端ASP.NET的支持，以及对于数据库ADO.NET开发的支持等。.NET将彻底改变Windows应用的开发模型，但是，这一切不会一夜之间马上发生变化。在几年（3~5年）之内，仍将是所有这些技术共存的状况，甚至传统技术还会有明显的优势，毕竟.NET从推出走向成熟和普及需要一个漫长的过程。

本书的发展历程基本上也反映了Microsoft Visual C++的技术路线。早期的时候，一切以MFC为主，后来把对OLE的支持也纳入进来。即使到了今天，如果要想快速地用C++编写出支持OLE对象的应用程序，那么，除了MFC之外，几乎没有其他的通用开发工具。然后，对于数据库的支持也被加入进来，但仅限于对于ODBC的支持。

到了本书第4版的时候，桌面应用技术已经非常成熟，MFC结构完全定型，同时COM技术在Internet应用中大行其道，由此产生了一批Active技术，包括ActiveX控件、ASP（Active Server Page）、ADO（Active Data Object）、Active Document等。在第4版中我们可以看到关于这些技术的讲解。第5版又引入了ATL相关的一些内容。

现在呢，最大的变化，也是最激动人心的变化应该是.NET架构的引入，当然也随之带来了一大批新的技术。从长远的发展来看，.NET必将取代现有的大量应用技术，但不是所有，另一方面，新的技术和老的技术将会共存相当长的时间。

本书第6版基本上反映了现在的技术发展状况，MFC仍然存在，但是一些新兴的用户界面元素也被更新到MFC中，比如MTI（Multiple Top-level Interface）界面风格、Rebar工具栏、HTML Help等。并且，本书也全面地讨论了COM和相关的开发支持，包括MFC和ATL对于COM的支持，这是目前仍然非常实用的两项技术。然后，本书还讨论了针对Internet的程序设

计技术，包括MFC对于Winsock和WinInet的支持，以及DHTML和ATL Server等。最后，本书还介绍了Visual C++对于.NET的支持，其中包括托管的C++以及如何用托管的C++来开发Windows Forms、ASP.NET和ADO.NET。这些内容涵盖了当前的主流技术，既有对老技术的继承和更新，又有对新技术的精辟阐述，所以，通过这本书你可以全面地学习到这些实用开发技术。

二、本书的内容和结构

本书前面的三分之二基本上继承了以前的版本（第4版和第5版），但是在继承的同时，也纳入了一些新的元素；后面的内容是新增加的。内容组织如下（摘自“Introduction”部分）：

第一部分：Windows、Visual C++ .NET和应用程序框架基础

在这部分中，既有抽象理论的阐述，又有实际应用，本书力求使这两者达到一种平衡。在这部分中，作者快速地回顾了Win32和Visual C++ .NET的基本组成，进而逐步将MFC应用程序框架以及文档-视图结构（document-view architecture）展现在读者面前。同时，在这部分中还给出了一个通过MFC类库中的类来构造的简单的“Hello，world!”程序，该程序只需30行代码。

第二部分：MFC的本质

在假定读者已经熟悉基本的Windows API的前提下，MFC库的文档材料简捷而又完整地介绍了应用程序框架的所有元素。在本书的第二部分中，读者的注意力将集中在应用程序框架的主要构成元素之一——“视图”（view）上，“视图”实际上是一个窗口。在这里，读者将从C++和MFC库中的类的角度来学习和掌握熟练的Windows程序员早已熟知的一些知识。读者还将使用Visual C++ .NET工具，这些工具避免了以前Windows程序员不得不忍耐的一些繁杂的编码工作。

第二部分介绍了很多内容，包括用位图进行图形编程、对话框数据交换、ActiveX控件使用、32位内存管理，以及多线程编程。其中的练习将帮助读者编写比较复杂的基于Windows的程序，但这些程序并没有充分利用应用程序框架的高级特性。

第三部分：MFC的文档-视图结构

这一部分介绍了应用程序框架的核心——文档-视图结构。在这部分中，读者将了解什么是“文档”，也将学习如何把文档同第二部分中所介绍的视图联系起来。一旦你动手编写出了一个自己的文档类，就会禁不住对MFC库简化文件I/O及打印的方式感到非常惊讶。

另外，读者在这部分中也会接触到命令消息处理（command message processing）、工具栏（toolbar）、状态栏（status bar）、切分框架（splitter frame）以及上下文相关帮助（context-sensitive help）。同时，这一部分中还将介绍单文档界面（SDI）、多文档界面（MDI），以及多顶级窗口界面（MTI），这是当前基于Windows的应用程序（比如Microsoft Word）的标准。

第三部分还讨论了如何用MFC库编写动态链接库（DLL）。读者将了解扩展DLL（extension DLL）和常规DLL（regular DLL）之间的区别。

第四部分：COM、自动化、ActiveX和OLE

要介绍COM不是一本书能够完成的。第四部分从MFC的角度出发，引导读者开始学习基本的COM理论。接下来读者将学习Automation（自动化），这是连接C++和Visual Basic for Applications （VBA）的纽带。读者还将熟悉统一数据传输（uniform data transfer），学习复合文档（compound document）和嵌入对象（embedded object）的基础知识。读者还将学习ATL类库对于OLE DB的支持。

第五部分：Internet程序设计

这部分从Internet的技术指南开始，涉及了TCP/IP协议和Internet程序设计的基础。读者将学习如何利用ATL Server来开发服务器端程序，还将学习如何编写动态HTML。

第六部分：.NET程序设计

Internet作为软件开发的新领域，目前正在不断发展。Internet不再仅限于构建一些Web网站供人们浏览，而是允许人们对Web网站进行编程。网络连接早已存在，但是在XML出现之前，人们还无法对“在Internet上发送方法调用”达成共识。.NET中两个最主要的推动力是Web服务和基于服务器的用户界面。.NET完全支持这些概念，并且还支持一种新的编写客户端用户界

面的方法：Windows Forms。第六部分介绍了.NET的内容，以及用户在.NET平台上可以做哪些事情。这部分中包含的章节涉及到公共语言运行库和托管的代码，以及利用C++、ASP.NET和ADO.NET来编写托管的组件。

三、本书特色和新增内容

虽然这是一本再版的书，并且有差不多三分之二的内容与以前的版本（这里主要指第4版）相同，但是，它仍然有自己的特色以及新增的丰富内容。

本书的叙述风格基本上没有变化，仍然按照以前的模式，既有原理性的说明，也有指导性的介绍，还有很好的例子来帮助理解一些技术细节。并且，本书在讲解的时候也不啰嗦，与以前的版本相比，本书的内容增加了不少，但是篇幅基本上没有明显的增加。作者对于前面部分的传统内容有所精简，以便为后面的内容腾出空间。

本书新增的内容可以分几个方面来看：

1. 本书的前三部分内容主要针对桌面应用程序的开发，这是MFC的传统优势，也是本书的传统优势。我们应该可以感受得到，这几年桌面应用技术还是有了一些变化的，特别是随着Windows 2000/XP的推出，尽管MFC的基本框架保持不变，但是在许多细微之处还是有了许多新增的界面元素的。例如，除了SDI和MDI之外，Windows 2000又引入了一种新的界面风格：多顶级窗口界面（MTI）。而且Internet Explorer的新版本更是引入了许多新的UI元素，包括一些扩展的控件，以及Rebar等。Visual C++ .NET为这些新的特性提供了支持，本书也介绍了相应的编程技术。

2. 本书第四部分介绍了COM、自动化、ActiveX和OLE。COM是一项底层的组件技术，这部分介绍了COM的原理，以及MFC和ATL提供的最基本的支持。要想在不到300页的篇幅中全面地介绍所有这些技术是不可能的，但是本书这一部分中，不仅成功地叙述了这些基本知识，同时还介绍了几项关键应用技术，包括自动化、统一数据传输、用ATL来开发ActiveX控件，以及用OLE DB模板来开发OLE DB Consumer和OLE DB Provider。

值得一提的是，除了传统的C++编程之外，Visual C++ .NET还引入了一种新的编程模式，被称为属性化的编程，用于支持COM组件的开发。它的基本思想是，允许C++类直接引用COM类和COM接口的一些属性，由编译器来解释这些属性并产生必要的代码，从而减轻了程

序员在开发COM组件过程中一些不必要的负担。

3. Internet开发是一个不可错过的部分。本书第五部分在介绍MFC对Winsock和WinInet支持的基础上，又介绍了动态HTML和ATL Server。请读者不要将ATL Server与COM联系起来，实际上，ATL Server主要是针对IIS（Internet Information Services）的一个开发工具，用来支持对于IIS服务器的扩展。

4. 最后一部分介绍Microsft .NET程序设计。这是本书新增的内容，虽然.NET的标准语言应该是C#，但是Visual C++ .NET也提供了完全的支持，它是通过扩展标准C++而实现的，被称为托管的C++。托管代码将运行在.NET的公共语言运行库之上，这是对组件技术的新发展，它将使得应用程序运行在一个更加完善的管理环境之中，底层大量的设施可以被直接使用，并且软件的发布和协作将更加理想。对于开发人员呢，负担减轻了，他们可以更加关注于自己的应用，而无需为一些琐碎的细节操心。这一部分首先介绍了.NET中最为核心的公共语言运行库，然后介绍了托管C++扩展，以及它的编程示例。最后用三章篇幅分别介绍了如何用托管C++来开发Windows Forms、ASP.NET和ADO.NET。

四、如何使用本书

面对一本涉及面如此广阔的计算机程序设计图书，我们该如何来使用呢？要想掌握书中每一项技术，这不仅需要大量的时间，还需要扎实的基本功，但是，掌握这些技术对于把握现代软件设计环境又非常有帮助。

按照这本书的内容结构，读者当然可以有所取舍，但是最好的做法是，首先按顺序阅读一遍，如果有可能的话，对于其中一些关键例子最好配合在机器上进行实战练习。然后，对于感兴趣的话题可以仔细钻研，或者把这本书当作参考书，以后随时翻阅。

这本书讲解的深度属于中等程度，读者最好有一些C++和Windows应用开发方面的基础。对于书中讲述的每一项技术，本书也只是起到基本的引导作用，如果读者希望进一步深入钻研的话，需要阅读系统提供的源代码或者查阅MSDN Library中的详细讲解。但是，通过这本书，你可以快速地领会每一项技术。

因为这本书既有原理性的讲解，也有示例说明，甚至还有一步一步的实战过程。所以，本书也适合作为教材，可以用于研究生程序设计课程或者高年级本科生课程。我在本书刚刚出版

之际，用本书作为北京大学软件学院研究生课程“程序开发环境分析与实践”的教材，取得了不错的效果。

本书的第6版仍然由我来翻译，在翻译过程中，我再次领略了这本书的精辟和独到之处。曾经听人提到过，说这本书只是教会你怎么用向导工具来生成一个应用程序，离开了向导工具你还是什么也不会。这种观点并不正确，我说过这本书是原理和实战的结合，向导是提高开发效率的好帮手，但是如果你明白了原理，难道还会离不开向导吗？难道会看不懂代码吗？我很赞成使用Visual C++提供的向导，但是一定要知道这些向导帮你做了哪些事情，这很重要，否则当向导不工作的时候，你就失控了。

这是第一次推出本书的英文版，我想，能够直接阅读原文版是读者的福分，我期望你会从阅读原文书籍中获得技术上的长进，并感受到阅读的乐趣。

2002年12月15日于北京大学燕北园宜静斋

Dedicated to Sandy Daston and Ted Shepherd

Acknowledgments

This part of book writing is always the best— everybody involved is nearly done with the manuscript and all that's left to do is to thank everybody. Because the author's name appears on the cover, it's sometimes easy to forget all the other folks involved in a project as large as this. Many other folks gave their time and energy to this project, and I wish to thank you.

Thank you Sandy Daston and Ted Shepherd—my family, for your support while I wrote this book.

Thank you, Denise Bankaitis. As the project editor, you kept me going by reminding me of the importance of this project (a key C++ reference for .NET) and by coordinating the efforts of the rest of the team, which includes Julie Xiao, Ina Chang, Danielle Bird, Juliana Aldous, Joel Panchot, Carl Diltz, and Gina Cassill.

Thank you, Julie Xiao, for keeping the manuscript accurate.

Thank you, Ina Chang, for making my sentences readable.

Thank you, Danielle Bird and Juliana Aldous. As acquisition editors, you got this project rolling and kept it on track.

Thank you, Joel Panchot, for making sure the art in this book looks good.

Thank you, Carl Diltz and Gina Cassill, for composing the manuscript and making it look great.

I would also like to thank the folks at DevelopMentor, for providing a wonderful environment and community for thinking and learning about modern computing. You guys are wonderful.

Introduction

The release of the Microsoft Visual Studio .NET (and Visual C++ .NET in particular) has underscored Microsoft's increasing focus on Internet technologies, which are at the heart of the Microsoft .NET architecture. In addition to supporting the .NET initiative, Visual C++ .NET keeps all the productivity-boosting features you're familiar with, such as Edit And Continue, IntelliSense, AutoComplete, and code tips. Visual C++ .NET also includes many new features such as managed code extensions for .NET programming, support for attributed code, and a more consistent development environment. These features take Visual C++ .NET to a new level. This book will get you up to speed on the latest technologies introduced into Visual C++.

.NET, MFC, and ATL

The technology churn we face these days is pretty impressive. We went from no computers on our office desktops to nearly everyone having a computer running MS-DOS in the 1980s to nearly everyone running Microsoft Windows by the mid-1990s. The technology wheel is about to turn again. In the late 1990s, everyone was developing Web sites by hand using tools such as raw Hypertext Markup Language (HTML), Common Gateway Interface (CGI), Internet Server Application Programming Interface (ISAPI) DLLs, Java, and Active Server Pages (ASP). In July 2000, Microsoft announced to the world that it would change all that by betting the company on a new technology direction named .NET.

The current thrust of Microsoft is indeed .NET. For a number of years, it's been possible to build a Web site by setting up a server somewhere, getting an IP address, and putting up some content. Anyone with the URL of your site can surf there and check it out. Commercial enterprises have been taking advantage of the Web by posting information that's useful to customers. The Web has also become an invaluable research tool and efficient news broadcast medium.

The computing world of the near future will involve the Web heavily. However, rather than just having human eyeballs look at Web sites, computers themselves will look at Web sites. That is, Web sites will be programmable through Web services. The .NET vision also pushes the responsibility of providing a rich user interface out to the server.

With so much emphasis on Web services and server-based user interfaces, it might seem that standalone applications and client-side user interface scenarios—normally the realm of tools such as the Microsoft Foundation Class Library (MFC)—will be left in the dust. But the need for rich client-side user interfaces is unlikely to go away. Many thought that the advent of the PC and distribution technologies would spell the end of centralized processing on mainframes and minicomputers. It turns out that PCs and distribution technologies only added to the available computing arsenal. The .NET vision of Web services and rich user interfaces provided by the server only adds to the options available to software developers. Rich client-side user interfaces will continue to be viable for many types of applications, running alongside other applications that use other kinds of user interfaces (such as server-generated user interfaces).

MFC is a mature and well-understood technology that's accompanied by a host of third-party extensions. For at least a little while longer, MFC represents the most effective way to write full-featured standalone applications. A good portion of this book will focus on MFC-style development, but we'll also cover Windows Forms—the .NET way to write client-side user interfaces.

Of course, the next question is: Where does this leave COM? COM has solved many problems related to distributed processing, but it has some serious shortcomings—mostly centered around component versioning and type information. Microsoft's .NET vision is based on the common language runtime. The runtime takes the place of COM as the interoperability standard within .NET. We'll cover .NET and the common language runtime in depth in Part VI of this book.

COM and the common language runtime represent different approaches to component architecture, but Microsoft has taken great care to ensure a seamless coexistence. The interoperability path between COM and the runtime is smooth in most cases. Within the .NET world, you probably won't find yourself using COM as a component architecture. However, you might find yourself using Active Template Library (ATL) Server, which is a high-performance means of writing Web sites.

I've updated the coverage of ATL and MFC in this edition of the book because you'll still find it very useful. More important, I'll show you how to leverage your heritage code (sounds better than "legacy code," doesn't it?) as you move into the .NET world.

Managed C++ vs. C#

The .NET platform has introduced a new C++-like language named C#. C# is a curly-brace-oriented language without all the headaches of C++. Much of C#'s appeal is due to the fact that it's missing some of the more problematic elements of C++ (such as raw pointer management) while maintaining the useful features (such as virtual functions). The C# compiler eventually emits managed code—the kind that runs under the common language runtime.

However, the entire world isn't going to switch over to C# overnight. There's just too much C++ code out there to convert. Also, it will take a bit of time for developers to become fully comfortable with C#. In the meantime, .NET has introduced extensions to C++ for producing managed code (code that runs under the common language runtime). Managed Extensions for C++ will help ease the burden of developing software for the .NET platform because they allow you to quickly update existing C++ code to work with .NET. Getting the managed code features in C++ means sprinkling your code with various keywords. In the end, C# and managed C++ boil down to the same executable code once the compilers are done with it. In the .NET world, you'll probably find yourself writing new components using C# while using managed C++ to add .NET features to your existing code base.

.NET vs. the Java Platform

In recent years, we've seen a great deal of interest in the Java programming language and platform. Java became a great boon for Internet developers by providing a useful means of distributing client user interfaces (through Java applets) and by providing enterprise solutions through Java Enterprise Edition. Now, .NET has become the best Internet development platform available today. Unlike the Java platform, which requires that you write all your code using the Java syntax, .NET often lets you use multiple syntaxes to arrive at the same machine instruction set. You can use C++ (the main focus of this book) and its managed extensions, Visual Basic .NET, C#, and even a host of third-party .NET languages to write your programs. Once you develop your source code, it is compiled to intermediate language and then eventually machine code before it runs. Because .NET code is managed by a runtime, you get benefits such as garbage collection and better code security.

Who This Book Is For

Visual C++ .NET, with its sophisticated application framework and support for .NET, is for professional programmers, and so is this book. I'll assume that you're proficient in the C language—you can write an if statement without consulting the manual. And I'll assume that you've been exposed to the C++ language—you've at least taken a course or read a book even if you haven't written much code. You might compare learning C++ to learning French. You can study French in school, but you won't be able to speak fluently unless you go to a French-speaking country and start talking to people.

The Visual C++ wizards save you time and improve accuracy, but programmers must understand the code that the wizards generate and, ultimately, they must understand the structure of the MFC and ATL libraries, the inner workings of the Windows operating system, and how .NET works. I won't assume, however, that you already know Windows and .NET programming. I'm sure that proficient C programmers can learn Windows the MFC way and the .NET way. It's more important to know C++ than it is to know the Win32 application programming interface (API). You should, however, know how to run Windows and Windows-based applications.

If you're already experienced with the Win32 API or with the MFC library, there's something in this book for you, too. You'll learn about new features such as the Multiple Top-Level Interface (MTI) and the Visual C++ .NET wizards. If you haven't already figured out the Component Object Model (COM), this book presents some important theory that will get you started on understanding ActiveX controls. You'll also learn about ATL Server and OLE DB templates. And you'll learn about C++ programming for the Internet (including Dynamic HTML). Finally, this book includes hard-to-find coverage of the new managed C++ extensions.

What's Not Covered

It's impossible to cover every aspect of Windows and .NET programming in a single book. I've excluded topics that depend on special-purpose hardware and software, such as MAPI, TAPI, and communications port access. I'll cover using ActiveX controls in an application and writing ActiveX controls using ATL, but I'll defer the in-depth coverage to Adam Denning and his *ActiveX Controls Inside Out* (Microsoft Press, 1997). I'll get you started with 32-bit memory management, DLL theory, multi-threaded programming techniques, and .NET programming, but you need to get the third edition of Jeffrey Richter's Programming Applications for Microsoft Windows (Microsoft Press, 1997) if