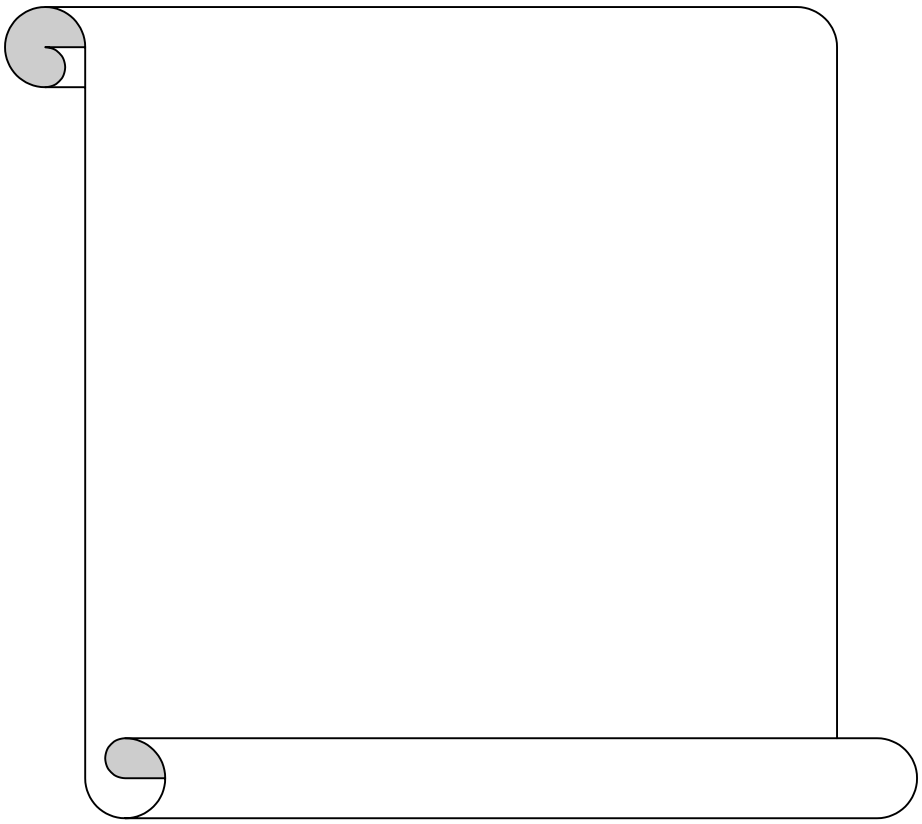
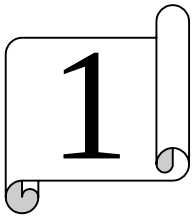
第 I 部分

Windows、Visual C++ .NET 和应用程序框架基础





Windows 和 Visual C++ .NET

在 20 世纪 90 年代早期，市场上主要的竞争在于桌面操作系统。现在，这场竞争已经结束了，绝大多数个人计算机系统中都在运行 Microsoft Windows。本章总结了 Windows 的底层编程模型(特别是 Win32)，并向您介绍了 Visual C++ .NET 的各组成部分是如何相互配合以帮助人们编写 Windows 应用程序的。同时，您也可以在本章中学到有关 Windows 的一些新东西。

1.1 Windows 编程模型

不管您使用什么样的开发工具，Windows 程序设计都已经与过时的面向批命令或面向事务的程序设计有了根本的不同。作为开始，您必须先了解一些有关 Windows 的基础知识。我们将参照大家熟知的 MS-DOS 编程模型来进行介绍。尽管您现在可能已不再编写纯粹的 MS-DOS 程序了，但相信您对 MS-DOS 编程还是比较熟悉的。

1.1.1 消息处理

当您用 C 来编写 MS-DOS 应用程序时，最起码要有一个 main 函数。当用户运行该应用程序时，操作系统会自动调用 main，从这一点上讲，您可以使用任何您所期望的程序结构。如果程序需要得到用户的键盘输入，或者需要使用操作系统所提供的服务，那么它就可以调用适当的函数，比如 getchar，或者使用基于字符的窗口库。

当 Windows 操作系统运行程序时，它首先调用程序中的 WinMain 函数。因此，在 Windows 应用程序中一定要有 WinMain 函数，该函数一般用来完成某些特殊的任务，其中最重要的任务就是要创建该应用程序的“主窗口”。“主窗口”中必须包含用来处理 Windows 所发送的消息的代码。基于 Windows 的程序和基于 MS-DOS 的程序之间的一个最根本的差别，就在于 MS-DOS 程序是通过调用操作系统的功能来获得用户输入的，而 Windows 程序则是通过操作系统发送的消息来处理用户输入

的。

说明 许多 Windows 开发环境，包括使用 MFC（Microsoft Foundation Class）库 7.0 版的 Visual C++ .NET，都通过隐藏 WinMain 函数及构造消息处理过程来简化编程。当您使用 MFC 库的时候，您不再需要编写 WinMain 函数，但是，弄清楚操作系统和程序之间的这种联系是非常必要的。

Windows 中的大多数消息都经过了严格的定义，并且适用于所有的程序。例如，当窗口被创建时系统就会发送 WM_CREATE 消息；当用户按下鼠标的左键时系统就会发送 WM_LBUTTONDOWN 消息；当用户敲了一个字符键时系统就会发送 WM_CHAR 消息；而当用户关闭窗口时系统又会发送 WM_CLOSE 消息。所有的消息都有两个 32 位的参数，可以保存诸如光标坐标、键盘码等这样的信息。当用户进行菜单选择和单击对话框按钮等操作时，系统又会相应地发送 WM_COMMAND 消息（命令消息）给适当的窗口。命令消息的参数不尽相同，这完全依赖于窗口的菜单设计。您还可以定义一些自己的消息，您的程序可以向桌面上的任何窗口发送这些消息。这些用户自定义的消息实际上使 C++ 有点像 SmallTalk。

您完全用不着担心如何使这些消息与代码联系起来，因为这是应用程序框架的事情。不过需要注意的是，Windows 的这种消息处理机制同时也强加给用户程序许多固定的结构。因此，不要强迫使 Windows 程序看起来像老式的 MS-DOS 程序，只要仔细研究一下本书中的例子程序，您就不难发现这些程序看起来的确与以往的程序有所不同。

1.1.2 Windows 的图形设备接口（GDI）

许多 MS-DOS 程序都直接往视频存储区或打印机端口输送数据，这种做法的缺点在于需要对每种显示卡或打印机类型提供相应的驱动软件。Windows 则提供了一个抽象层，被称为图形设备接口（GUI，Graphics Device Interface）。Windows 已经提供了各种显示卡及打印机的驱动程序，这样您的程序就可以不必关心与系统相连的显示卡及打印机的类型。您的程序不用直接访问硬件，而可以通过调用 GDI 函数与硬件打交道，各种 GDI 函数会引用一个被称为设备描述表(device context)的数据结构。Windows 会将设备描述表结构映射到相应的物理设备，并且会发出正确的输入/输出指令。GDI 在处理速度上几乎和直接进行视频访问一样快，并且它还允许 Windows 的不同应用程序共享显示。

在本书的后面，我们还将会看到 GDI+。您可能已经猜到了，GDI+是 GDI 在.NET 中的接替者。GDI+中的服务是通过一组 C++类来体现的，这些 C++类都是被托管的代码。也就是说，这些代码将运行在公共语言运行库的环境下。GDI+在传统的 GDI 基础上做了很多增强，包括渐变画刷、三次样条曲线、独立的路径对象、可伸缩的区域、alpha 混合(blending)和多重图像格式。

1.1.3　基于资源的程序设计

在 MS-DOS 下，为了实现数据驱动的程序设计，您必须将数据定义为初始化常量，或者提供单独的数据文件供程序读取。而当进行 Windows 程序设计时，您可以用一些特定的格式将这些数据存储在资源文件中，这样，连接器就可以把由 C++编译器输出的二进制代码和二进制资源文件结合起来生成可执行程序。资源文件可以包含位图、图标、菜单定义、对话框布局设计和字符串，甚至可以包含用户自定义的格式。

您可以用文本编辑器来编辑程序，而对各种资源则通常用“所见即所得”(WYSIWYG)风格的工具来加以编辑。例如，您在设计对话框时，可以首先从被称为控件箱(control palette)的图标组中选取适当的元素(按钮、列表框等)，然后再通过鼠标来进行定位和尺寸设置。Visual C++ .NET 为各种标准的资源格式都提供了图形方式的资源编辑器。

1.1.4　内存管理

Windows 的每一个新版本中，内存管理都会比过去更容易。如果您听说过关于锁住内存句柄、thunk(隐式替换块)以及内存申请管理等很繁琐的细节的话，您现在就不必担心了，因为这些都已经过去了。现在您可以非常简单地申请到所需要的内存，至于细节问题则完全由 Windows 来控制。第 10 章将介绍 Win32 所采用的内存管理技术，包括虚拟内存和内存映射文件。

1.1.5　动态连接库(DLL)

在 MS-DOS 环境下，所有程序的目标模块在创建(build)过程中都被静态地连接起来。而 Windows 则允许动态地连接，即一些专门构造的库可以在运行过程中被装载和连接到进程中，并且多个应用程序可以共享同一个动态连接库(DLL)，这样可以大大节省内存和磁盘空间。同时，动态连接还可以大大提高程序的模块灵活性，因为您可以对动态连接库单独编译和测试。

设计者起初创建 DLL 时完全是为 C 语言而设计的，C++则使这一问题变得有些复杂。MFC 库的开发者们已经将应用程序框架的所有类组合到几个已建好的动态连接库中，这意味着，您可以把应用程序框架类静态地或者动态地连接到您的应用程序中，并且还可以创建自己的基于 MFC DLL 的 MFC 扩展 DLL。第 22 章详细介绍了如何创建 MFC 扩展 DLL 和常规 DLL。

1.1.6　Win32 应用程序编程接口

以前的 Windows 程序员用 C 语言在 Win16 应用程序编程接口(API)上编写应用程序。当然，今天

很少再有人会编写 16 位应用程序。现在，大多数开发人员使用 Win32 API 来编写应用程序。Win16 函数和 Win32 函数之间的主要差别，在于 Win32 函数中的许多参数都已经被加宽了。所以，虽然在过去几年中 Windows API 已经有了变化(并且还将继续变化下去)，但是，使用 MFC 库的开发人员可以不管这些变化，因为 MFC 的标准在 Win16 和 Win32 下都是适用的。

1.2　Visual C++ .NET 的组成

Visual C++ .NET 在一个产品中包含了几套完整的 Windows 应用开发系统。如果您愿意，仍然可以只使用 Win32 API 来开发用 C 语言编写的 Windows 应用程序。关于使用 C 语言的 Win32 编程技术，在 Charles Petzold 的 *Programming Windows, fifth Edition* (Microsoft Press, 1998)中进行了详细的论述。(Petzold 写了一本新的关于 Windows 编程的书，书名为 *Programming Microsoft Windows with C#*(Microsoft Press)，该书讲述了如何使用 C#进行 Windows 编程和 Windows Forms 编程。在本书后面，我们也会介绍如何使用 Windows Forms 和 C++进行 Windows 编程。) 您可以利用 Visual C++ .NET 所提供的一些工具(包括资源编辑器)使底层 Win32 程序的编写更加方便。您也可以利用应用程序框架库(例如 MFC 库，以及托管库中的 Windows Forms)来进一步加速 Windows 应用程序的开发工作。

最后，Visual C++ .NET 也包括活动模板库(ATL, Active Template Library)，您可以用它来开发 ActiveX 控件。ATL 编程既不是 Win32 的 C 语言编程，也不是 MFC 的编程，而是非常复杂的，足够专门写一本书来讲述。然而，我们在本书中也会触及到 ATL 开发。ATL 最适用的场合在于高性能的 Web 服务器环境中，在那里它可以找到自己的一席之地。

本书的第一部分是关于如何用 Visual C++ .NET 所提供的 MFC 库应用程序框架来进行 C++程序设计。在这里，您将会使用在 Visual Studio .NET 文档中包含的 Microsoft Visual C++ MFC Library Reference 中所介绍的 C++类，并将使用针对应用程序框架的专用 Visual C++ .NET 工具，如 Class View。


说明　使用 MFC 库程序设计接口并不意味着您就不能使用 Win32 函数，实际上，您几乎总是需要在 MFC 库程序中直接调用 Win32 函数。

迅速地浏览一下 Visual C++ .NET 的各组成部分将有助于您了解应用程序框架。图 1.1 显示了 Visual C++ MFC 应用程序的大致创建(build)过程。

1.2.1　Visual C++ .NET 和创建过程

Visual Studio .NET 是一套开发工具，其中包括 Visual C++ .NET。Visual Studio .NET 集成开发环境(IDE)是 Visual C++ .NET、Microsoft Visual C#和 Microsoft Visual Basic .NET 公用的开发环境。该 IDE 来源于早期的 Visual Workbench，而 Visual Workbench 又来源于 QuickC for Windows。贴边窗口

和可配置的工具栏，再加上一个可运行宏的可定制编辑器，这就是目前的 Visual Studio .NET 的组成部分。联机帮助系统(现在已经被集成到 MSDN Library 阅读器中了)的工作方式则像是一个 Web 浏览器。图 1.2 显示了 Visual C++ .NET 工作时的情形。

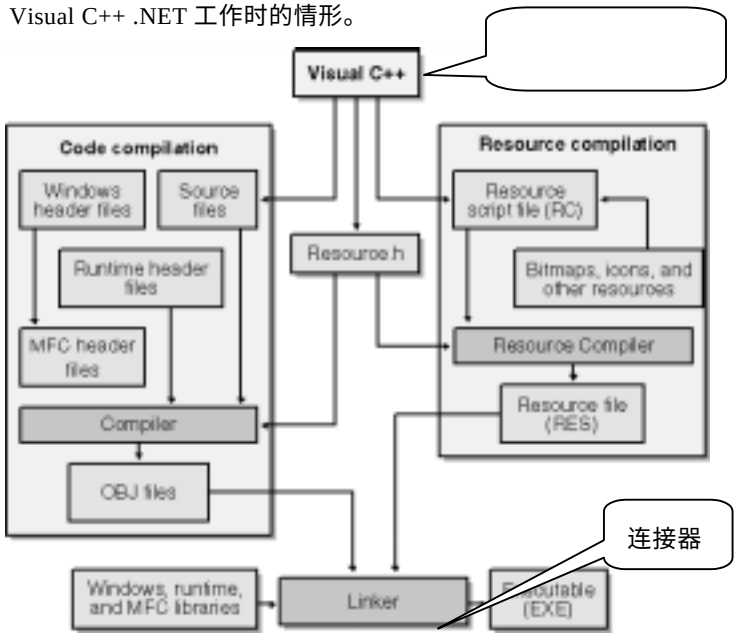


图 1.1　Visual C++ MFC 应用程序的创建过程

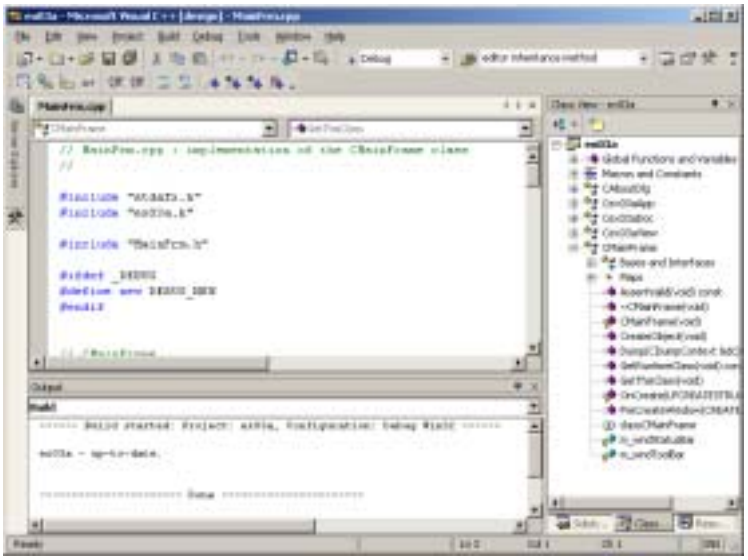


图 1.2　Visual C++ .NET 窗口

如果您使用过以前的 Visual C++版本，那么就已经知道 Visual C++ .NET 是如何工作的了。但如果您对集成开发环境(IDE)还比较陌生的话，则必须弄清楚什么是项目(project)。“项目”是一些相互关联的源文件的集合，这些源文件经过编译、连接，然后被组合在一起形成可执行的 Windows 应用程序或者 DLL。每个项目的源文件一般被存储在一个单独的子目录中，但它也需要依赖于该子目录之外的许多文件，如包含文件和库文件。

Visual Studio .NET 也支持在开发环境的外部创建项目。Visual Studio .NET 仍然支持 make 文件(makefile)。(makefile 保存了编译器和连接器的参数选项，还表述了所有源文件之间的关系。)也就是说，您仍然可以手工编写一个 makefile，然后通过 NMAKE.EXE 运行这个 makefile。(源代码文件需要某些特定的包含文件，而可执行文件要求特定的目标文件模块及库等。)NMAKE 首先读取 makefile，然后再激活编译器、汇编器、资源编译器和连接器以便产生最后的输出，最后输出并生成的通常是一个可执行文件。NMAKE 利用内置的推理规则来激活编译器，以便通过编译特定的 CPP 文件来产生相应的 OBJ 文件。注意，Visual C++ .NET 不再支持在开发环境中为活动项目导出 makefile 的功能。在命令行中，使用 Devenv 开关可以编译连接(build)Visual Studio .NET 项目。

在 Visual C++ .NET 的项目中，一个文本格式的项目文件(扩展名为 VCPROJ)维护着该项目各部分之间的相依关系。一个独立的文本格式的方案文件(solution file, 扩展名为 SLN)针对该方案中的每一个项目都有一个入口项。方案文件将项目、项目条目以及方案条目组织到一个单独的方案中，它引用了它们在磁盘上的位置信息，并将这些信息提供给开发环境。在一个方案中可以有多个项目，不过本书中的所有例子的方案都只有一个项目。要在一个已经存在的项目上继续工作，只需在 Visual C++ .NET 中打开相应的 SLN 文件，然后就可以编辑和编译连接该项目了。

Visual C++ .NET 也会创建许多中间文件。表 1.1 列出了 Visual C++ .NET 会在一个方案中产生的文件。

表 1.1 Visual C++ .NET 项目中产生的文件类型

文件扩展名	说明
APS	支持资源视图(Resource View)
BSC	浏览器信息文件
IDL	接口定义语言文件
NCB	支持Class View
SLN	方案文件*
SUO	包括方案中的可选项和配置信息
VCPROJ	项目文件*

* 表示不要删除或在文本编辑器中编辑

1.2.2　资源视图窗口和资源编辑器

当您打开 Visual C++ .NET 集成环境中的资源视图(Resource View)窗口(通过选择【视图】菜单中的【资源视图】选项)时，您可以选择一个资源进行编辑。主窗口会自动为该类型的资源选择适当的资源编辑器(resource editor)。该窗口也会为菜单选择一个所见即所得的编辑器，为对话框选择一个功能强大的图形编辑器，并有一些工具可用于编辑图标、位图和字符串。对话框编辑器允许您插入标准的 Windows 控件和新的 Windows 通用控件，也可以插入 ActiveX 控件。

每一个项目通常都有一个文本格式的资源描述(RC)文件，该文件描述了项目的菜单、对话框、字符串和快捷键资源。RC 文件也有#include 语句，可以将位于其他子目录下的资源包含进来。这些资源包括与该项目相关的项，如位图(BMP)和图标(ICO)文件，以及对所有 Visual C++ .NET 程序通用的资源，如错误消息串等。在资源编辑器以外对 RC 文件进行编辑是不提倡的。资源编辑器也可以处理 EXE 和 DLL 文件，因此您可以用剪贴板来“偷”资源，比如其他 Windows 应用程序的位图和图标等。

1.2.3　C/C++编译器

Visual C++ .NET 的编译器可以处理 C 和 C++源代码，它通过检查源代码文件的扩展名来识别代码本身所使用的语言。C 扩展名代表 C 源代码，CPP 或 CXX 扩展名则代表 C++源代码。该编译器与所有的 ANSI(American National Standards Institute，美国国家标准协会)标准兼容，包括 C++库工作组的最新建议，还增加了一些 Microsoft 扩展的内容。Visual C++ .NET 还全面支持模板(template)、异常(exception)和运行时类型信息(runtime type information, RTTI)。C++标准模板库(STL, Standard Template Library)也被包括了进来，尽管它并没有被集成到 MFC 库中。

1.2.4　源代码编辑器

Visual C++ .NET 包括一个复杂的源代码编辑器，它支持许多特性，例如动态语法着色、自动缩进(auto-tabbing)、与多种流行编辑器(如 VI 和 EMACS)的键盘绑定，以及美观的打印效果等。从 Visual C++ 6 开始，开发环境包含一个被称为 AutoComplete(自动完成)的特性。如果您用过任何一个 Microsoft Office 的产品，或者 Visual Basic，您可能已经对这项技术非常熟悉了。利用 Visual Studio .NET 的 AutoComplete 特性，您只需键入程序语句的起始部分，编辑器将会提供一个列表，其中列出了所有可能的完整语法单元，供您从中选择。当您在使用 C++对象的时候，如果忘记了确切的成员函数或者数据成员的名称，则这个特性可以为您提供极大的便利——您只需从列表中选取即可，所有可能的成员都会出现在列表中。应该感谢这个特性，您不用再记住上千个 Win32 API，也不用再过多地依赖于联

机帮助系统了。

1.2.5 资源编译器

Visual C++资源编译器从资源编辑器读取一个 ASCII 码 RC 文件，并生成一个二进制 RES 文件提供给连接器。

1.2.6 连接器

连接器读入由 C/C++编译器产生的 OBJ 文件和由资源编译器产生的 RES 文件，同时处理 MFC 代码的 LIB 库文件、运行时库代码及 Windows 代码，然后产生项目的 EXE 文件。有一个增量连接选项(incremental link option)可以在源文件变化不大的情况下最大限度地减少执行的时间。由于 MFC 头文件中包含的#pragma 语句(特殊的编译指令)已指定了所要求的库文件，所以您不必再显式地告诉连接器该读入哪些库。

1.2.7 调试器

如果您的程序是第一次运行，则用不着使用调试器，可是以后就少不了一次又一次地使用调试器了。Visual Studio .NET 提供了一个集成的调试器，它将 Visual C++和 Visual Basic 早期版本中的调试器结合起来，并且增加了许多新的特性。这些新特性包括：

- 跨语言的调试 Visual Studio .NET 使您可以调试同一个方案中的多个项目，即使这些项目是用不同的语言编写的。
- 将调试器附到一个正在运行的程序上 Visual Studio .NET 允许您将调试器附到一个在 Visual Studio .NET 之外运行的程序上，并进行调试。
- 远程调试 Visual Studio .NET 支持远程调试，也就是说，您可以将调试器附到一个在其他服务器上运行的程序上。
- 调试 **ASP.NET Web** 应用 ASP.NET 文件是经过编译的，所以在调试过程中，它们可以像其他语言一样看待。这使得调试 Web 应用比以前容易得多。
- 支持调试和代码跟踪的**.NET** 框架类 .NET 框架类使得在代码中支持调试和插入跟踪语句更加容易。因为这些类是托管的代码，所以您可以在托管的 C++代码中运行这些类。

图 1.3 显示了正在运行的集成调试器。

图 1.3　Visual C++ .NET 调试器窗口

注意　在 Variable(变量)和 Watch(观察)窗口中可将对象指针展开，以便显示出派生类和基类的所有数据成员。如果把光标放在简单变量的位置上，则调试器还会在一个小窗口中显示该变量的值。为了能够对程序进行调试，在编译连接(build)程序时，您必须设置相应的编译器和连接器选项，以便产生必要的调试信息。

Visual C++ .NET 包含了“ Edit And Continue ”(编辑并继续)特性。该特性使得您在调试一个应用程序的时候，可以修改该应用程序，然后继续在新代码的基础上进行调试。这个特性可以大大降低您花在调试上的时间，因为您不用再手工断开调试器，重新编译代码，然后再次进入调试。为了使用该特性，您只需在调试器中编辑任何代码，然后单击 Continue(继续)按钮即可。Visual C++ .NET 将会对您所做的修改进行编译，然后重新启动调试器。

1.2.8　MFC Application Wizard

MFC Application Wizard(MFC应用向导)是一个代码生成器，它会根据用户在对话框中指定的特性、类名及源代码文件名来产生Windows应用程序的代码骨架。您在学习本书中所提供的例子时，需要经常使用MFC Application Wizard。不过，不要把MFC Application Wizard同以前那种可以为应用产

生所有代码的代码生成器混淆起来。MFC Application Wizard所产生的代码只是一些最基本的代码，它所完成的功能则完全由应用程序框架的基类所决定。

MFC Application Wizard 可以使您很快地开始一个新的应用程序。而且，该向导是可扩展的，也就是说，您可以编写自己的代码生成器。如果您发现您的工作组需要建立很多带有远程通讯接口的项目，则可以建立一个特制的向导来自动完成这一过程。

1.2.9 Class View

您只要从 View(视图)菜单中选择 Class View(类视图)就可以打开 Class View 窗口。您可以看到一个树状视图结构，其中包含了项目中所有的类。该树状视图显示了成员函数和数据成员，如图 1.2 所示。用鼠标双击每一个元素，您马上就可以看到相应的源代码。当修改源代码的时候，Class View 也会更新它的内容，以反映出您所做的修改。Visual C++的早期版本包含一个被称为 ClassWizard 的独立部件，用来处理所有涉及到管理 Visual C++类代码的任务。ClassWizard 的功能已经被几个新的向导所替代，这几个向导独立地执行某些任务，比如添加全新的类、在一个类中增加虚函数，以及增加消息控制函数等。例如，增加类和函数的功能可以在 Class View 中完成。

1.2.10 方案管理器

方案管理器(solution explorer)代表了整个项目的一个组织结构。整个 Visual Studio .NET 应用可能包括许多条目(item)——包括许多项目。方案管理器允许您管理一个方案的方方面面。

方案管理器包括一个树状视图，其中列出了项目中所有的条目。方案管理器允许您打开这些条目以便进行修改，或者执行其他的管理任务。这些条目的树状视图体现了该方案与项目和方案条目之间的逻辑关系。该视图不一定代表逻辑存储关系。您可以将文件直接与方案关联起来，而不是与项目关联起来；要做到这一点，您只需将文件直接添加到方案中即可。

1.2.11 对象浏览器

如果一个应用程序是由您从头开始编写的，那么您一定会对其中的源代码文件、类及成员函数了如指掌。然而，如果您接手其他人的应用程序，那么若想搞清楚就需要一些帮助了。Visual C++ .NET 的对象浏览器(简称“浏览器”)能够让您从类或函数的角度来了解(和编辑)一个应用程序，而不是直接从文件入手，这有点像其他的面向对象库(如 Smalltalk)的“检查器”工具。

为了调用浏览器，需从 View 菜单中选择 Other Windows(其他窗口)下的 Object Browser(对象浏览器)。该浏览器有如下几种查看模式：

- **Definitions and References** 只要选择任何函数、变量、类型、宏定义或者类，都会马上看到它是在项目中哪里被定义的，并且在哪些地方用到了它。
- **Sorting** 可以将对象和成员按字母顺序排序，或者按类型和访问属性进行排序。
- **Derived Classes and Members/Base Classes and Members** 这是类层次结构的图形表示。对于所选择的类，您可以看到它的派生类或者基类以及成员，并且可以利用鼠标来控制层次结构的展开与否。

在第 3 章中显示了一个典型的浏览器窗口。


说明　如果重新编排了源代码文件中的代码行，则 Visual C++ .NET 会在重新编译连接时更新浏览器数据库。这会增加编译连接的时间。

1.2.12　UML 工具

Visual C++ .NET 现在包含一些 UML(Unified Modeling Language, 统一建模语言)工具。UML 是一组图表和程序描述规范，被用于描述一个系统。UML 中包含的图表类型有类图、对象图、行为图和状态图。许多组织利用 UML 作为文档标准，来描述他们的系统。

Visual C++ .NET 的 Project(工程)菜单中包含一个命令可用于 UML 逆向工程，即可将一个项目转换为 UML 图。为了将一个 Visual C++ .NET 项目逆向工程转换为一个 UML 图，您首先要为该项目产生浏览器信息，然后从 Project 菜单中选择 Visio UML 和 Reverse Engineer(逆向工程)。Visual C++ .NET 将为该项目生成一个 UML 包(一组图的集合)，并启动 Visio 的一个复本，以显示该 UML 包。(这些 UML 图是按照 Visio 格式生成的。)


说明　为了查看 Visio UML 方案的联机帮助，您必须将 Visio 保持在激活并运行的状态。在安装 Visual Studio .NET Enterprise Architect 的最后阶段，您将看到一个关于安装 Visio 的选项。

1.2.13　联机帮助

从 Visual C++ 6 开始，联机帮助系统就被移到了一个称为 MSDN Library Viewer 的独立应用中。这是一个基于 HTML 的帮助系统，每一个主题都用一个独立的 HTML 文档来表达，而且所有的主题都被组合到一些索引文件中。MSDN Library Viewer 使用了 Microsoft Internet Explorer 4 的代码，所以它看起来非常像您所熟悉的 Web 浏览器。MSDN Library 可以访问来自 Visual Studio .NET 安装光盘的帮助文件，或者来自硬盘的帮助文件，这取决于您在安装过程中的选择，它还可以访问 Internet 上的 HTML 文件。

Visual C++ .NET 允许您以不同的方式访问帮助系统：

- 按书**(By book)** 当从 Visual Studio .NET 的 Help(帮助)菜单中选择 Contents (内容)时，Contents 窗口打开，并显示 Visual Studio .NET 文档和 MSDN Library。在这里可以找到 Visual Studio .NET、.NET 框架的 SDK、Platform SDK 文档等等，所有这些内容按书和章节的层次被组织在一起。至于具体所显示的内容范围则依赖于您选择的过滤器(filter)。
- 按主题**(By topic)** 当从 Visual Studio .NET 的 Help 菜单中选择 Index(索引)时，Index 窗口打开。您只要在窗口中输入一个关键字，就可以看到包含该关键字的主题和文章。具体所显示的主题范围依赖于您选择的过滤器。
- 按词**(By word)** 当从 Visual Studio .NET 的 Help 菜单中选择 Search(搜索)时，Search 窗口打开。您可以用这个窗口来执行针对一组词的全文检索，从而查到包含这些词的文章。检索结果的范围依赖于您应用的过滤条件。
- 动态帮助 动态帮助允许您提供一些指针来启动 Visual Studio .NET，这些指针可以指向当前您正在使用的区域的信息，也可以指向您在 IDE 环境中正试图完成的任务。
- 按 **F1(F1 help)** 这是对程序员最友好的方式。只要把光标移到函数、宏或者类的名称上，然后按下 F1 键，就会启动帮助系统。如果该名称在几个地方都被找到——例如在 MFC 和 Win32 帮助文件中都有——则 Index 窗口将显示这些主题列表，您可以从中选择期望的主题。

不管使用什么方式来访问联机帮助系统，您都可以把帮助文本拷贝到剪贴板上，然后插入到您的程序中。

1.2.14 Windows 诊断工具

Visual C++ .NET 包含一些很有用的诊断工具。SPY++给出了系统进程、线程和窗口的一个树状视图，通过它您可以观察正在运行的应用程序的窗口及消息。Visual C++ .NET 还包含一整套 ActiveX 工具、一个 ActiveX 控件测试程序和其他一些实用工具。

1.2.15 MFC 库版本 7

MFC 库版本 7 是本书的主题之一。它定义了您马上要深入学习的应用程序框架。第 2 章将从一些实际的代码开始讲解，同时也将介绍一些重要的概念。

1.2.16 ATL 库版本 7.0

ATL 独立于 MFC 库，用它可以建立 ActiveX 控件。您既可以用 MFC 库建立 ActiveX 控件，也可以用 ATL 库建立 ActiveX 控件，但 ATL 控件相对来说又小又快，适合于在 Internet 上使用。第 27 章

和第 28 章概要地介绍了 ATL，以及如何用 ATL 创建 ActiveX 控件。在本书中我们也将讨论 ATL 服务程序(ATL Server)。

1.2.17　.NET 支持

Visual Studio .NET 全面支持.NET 框架。虽然 DLL、C++、MFC 库、COM 和 ATL 可以协同工作，共同创建 Windows 应用程序，但是，这样的系统会有一些缺陷。在有些情况下，它看起来好像有些部分是用绷带捆起来似的。.NET 的一个主要目标是统一编程模型，以便使 Windows 平台更加可靠和一致。例如，公共语言运行库(CLR)的作用就在于为所有的程序设计语言提供一组统一的数据类型。ASP.NET 也运行在该运行库下，从而使得 Web 应用的编程也更加一致。

除了 Visual Basic .NET 已经支持新的环境之外，Microsoft 也正在更新 C++，通过增加托管扩展，使得 C++程序也可以运行在新的环境下。使用托管扩展，就可以让 Visual C++ .NET 编译器产生出可在 CLR 下运行的代码。当前有大量的 C++遗留代码，利用 C++的托管扩展可使其到.NET 的迁移更加容易。我将在本书的后半部分细致地讨论在构建.NET 应用的过程中，.NET 和 Visual C++ .NET 所分别担当的角色。

文件名：	ch01.doc
目录：	C:\WINDOWS\Desktop\desk
模板：	C:\WINDOWS\Application Data\Microsoft\Templates\技术内幕.dot
题目：	第一部分 Windows、Visual C++
主题：	
作者：	liyj
关键词：	
备注：	
创建日期：	2004-4-19 14:36
更改编号：	6
上次保存日期：	2004-6-2 11:23
上次保存者：	zyp
总共编辑时间：	18 分钟
上次打印时间：	2004-6-2 11:26
打印最终结果	
页数：	15
字数：	1,819 （约）
字符数：	10,369 （约）