本书内容覆盖了Visual C++应用开发领域所涉及的关键技术。

精通 Visual C++ 6.0

王 晖 等编著
王毓政 审校

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

精通 Visual C++ 6.0

王 晖 等编著

王毓政 审校

電子工業出版社

Publishing House of Electronics Industry

北京·BEIJING

前　言

Visual C++ 6.0是Microsoft公司推出的最新可视化C++程序开发工具，具有许多先进的特性(在本书1.2节予以介绍)，被广泛应用于Windows 32位平台的基础应用程序开发。本书编写的目的是让读者能够在较短的时间里掌握Visual C++ 6.0这一强大的开发工具。读者应当首先掌握C++程序设计的基本知识，以便于理解书中内容。本书分4篇，24章，各篇内容如下：

第一篇(1～7章)介绍Visual C++ 6.0的不同版本、特性，及如何使用编程界面、Wizard、文本编辑器、资源编辑器、调试器等环境与开发工具。本篇的目的是使程序设计人员能够快速了解、掌握Visual C++ 6.0的开发环境。

第二篇(8～12章)主要介绍如何使用MFC编写基本的Windows应用程序，包括对常用类的介绍。通过介绍一个画线程序的例子，描述典型的MFC应用程序的编写过程。本篇还介绍使用通用控件、对话框、资源编程的方法以及GDI绘图、打印与打印预览的操作等。通过阅读本篇，读者能够基本了解MFC编程的机理、方法并能够开发简单的Windows应用程序。

第三篇(13～20章)介绍使用Visual C++ 6.0进行高级应用程序设计的方法，主要有内存管理、创建和使用动态链接库、Active X/OLE编程、使用ODBC、DAO、ADO访问数据库；编写多线程应用程序；使用MCI API、多媒体控件、Direct X5开发多媒体应用程序；使用WinSock类、WinInet类、ISAPI类开发网络通信和Internet应用程序。读者可以根据需求选择了解本篇内容，以开发各类高级应用程序。

第四篇(21～24章)介绍一些辅助开发工具，如Spy++、DDESpy和Pview、Install Shield、Active X Control Test Container、OLE/COM对象浏览器、Help Workshop及其他一些工具。读者使用这些工具可以加快程序的设计和调试过程。

本书是在王毓政教授的指导下，由编写组人员共同努力完成的，是集体智慧的结晶。本书第1章至第5章、第7章至第9章、第21章至第24章由陈丹编写；第6章、第10章至第12章由余安萍编写；第13章至第15章由蔡付东编写；第16章至第18章由俞俊平编写，第19章至第20章由王晖编写。王建、杨柳也参与了校对工作。

感谢电子工业出版社图书策划与开发部的同志给予我们的支持。

作　者

第 一 篇

Visual C++ 6.0

使用指南

第 1 章　Visual C++ 6.0 开发环境概述

本章将对 Microsoft Visual C++ 6.0 的开发环境进行简单的描述。无论对于初次使用 Visual C++还是曾经使用过 Visual C++低版本进行编程的人员来说，熟悉 Visual C++ 6.0 的开发环境、了解 Visual C++ 6.0 的新特点都是非常必要的。

1.1　Visual C++ 6.0 的不同版本

Microsoft Visual C++ 6.0 包括以下 3 个版本：

1. 标准版（Standard Edition）

Microsoft Visual C++ 6.0 标准版（即先前的学习版）包含了丰富的专业工具以帮助用户学习 C 和 C++及其他技术（如 MFC、OLE、ODBC、DAO、ActiveX 和 COM 等）。除了代码优化、定制向导、InstallShield 和与 MFC 的静态链接之外，标准版几乎包含了专业版的所有特征。

2. 专业版（Professional Edition）

Microsoft Visual C++ 6.0 专业版用以帮助用户开发商业质量的软件产品，开发应用程序及 win32 平台（包括 Windows95 及后继版本、Windows NT）的控件，并可以定制操作系统的图形用户界面或控制台 API。

3. 企业版（Enterprise Edition）

Microsoft Visual C++ 6.0 企业版提供了许多工具和部件，用于构造和实现企业级的分布式 COM（Component Object Model）应用程序，开发和调试用于 Internet 或 Intranet 客户服务器应用程序。企业版除包含了所有专业版的特征外，还包括了用于 SQL 数据库的开发工具及调试 SQL 的存储程序（stored procedures），其可视的源代码控制系统（Visual SourceSafe ）简化了组内的开发。

本书主要围绕Microsoft Visual C++ 6.0 企业版的内容进行介绍，除非特别声明，书中的Visual C++均指 Microsoft Visual C++ 6.0 企业版，有时为了同低版本的 Visual C++区别，也称之为 Visual C++ 6.0。

1.2　Visual C++ 6.0 的新特点

Visual C++ 6.0 的新特点体现在以下几个方面：

1. 编译器

提供了_assume、_inline 关键字，可生成 dep 文件，添加了新的警告提示以帮助用户发现编程错误，支持 delete 操作符的 placement form。可在代码中嵌入特定的检测器来检查通常的错误。

2. 调试器（Debugger）

AfxDumpStack 函数可以对域内安装的应用程序进行诊断，它产生当前堆栈的映像后将该映像卸载到调试输出设备，这个特性使得用户的调试器在没有安装到用户终端的情况下也可获得诊断信息；编辑和继续（Edit and Continue）功能允许用户在调试期间进行简单的编辑而无须退出调试。调试器提供了一个名为 ERR 的虚寄存器（pseudo-register）以显示当前线程的最近一次错误代码。此外，调试器还支持 MASM 格式的十六进制数，提供了新的 Watch 变量符号如 hr、st、wm 等，还提供了为当前线程显示线程信息块的虚寄存器等。

3. 编辑器（Editor）

文本编辑器采用 Intellisense 使编程更容易且减少错误，当你在编写程序的时候，Intellisense 选项将在光标处显示类成员、函数原型、ID 声明以及代码的释义。Intellisense 还能为你完成部分已识别的字，从而避免了重复敲入长类名及成员名。

快速的宏记录（Quick Macro Recording）使你可以通过选择 Tool 菜单项中的 Record Quick Macro 项，把任何由 Developer Studio 对象模板支持的行为记录在一个临时的 VBscript 宏中。资源编辑器提供了 CComboboxEx、CDateTimeCtrl、CIPAddressCtrl、CMonthCalCtrl 等类支持最新的 IE4.0 通用控件。通过控件工具条可以很容易地将 IE 控件添加到表单或对话框中。

4. 连接器（Linker）

Linker 提供了推迟载入（delay loading）动态库，不再需要用 win32 SDK 函数 Loadlibrary 及 GetProcAddress 来完成。Visual C++ 6.0 为连接器、DUMPBIN 及 EDITBIN 提供了新的选项暴露更多的操作系统功能，支持更多的平台。新的选项有 DELAY，DELAYLOAD，LINK50COMPAT，MAPIINFO 等。另外，经过修改的导入库格式将库文件减至最小。

5. 自动化对象模型（Automation Object Model）

在 Visual C++ 6.0 中，自动化的功能得以加强使得编程更灵活，提供了新的属性与方法。如果用户安装了 Visual Studio，Script Debugger 使用户能够调试自动化的 script，如包含 VBscript 或 Jscript 的 HTML 页，如果用户的 script 含有一个错误，会自动出现一个对话框并提示用户运行 Script Debugger。

6. 工程（Project）

Visual C++ 6.0 中提供了新的 MSDEV 命令，允许从命令行的方式编译连接一个 Visual C++ 工程文件，而不需要先输出一个 makefile（工程扩展名为.mak）及运行 NMAKE 工具。

此外，Visual C++ 6.0 还提供了 ClassView 的动态更新的功能。在键入新的类、变量、成员函数时，Visual C++ 6.0 将不要求你先存储，而直接更新 ClassView 和 WizardBar。同时，Visual C++ 6.0 还可以实现直接从执行对话框的类跳到对话框编辑器，另外还支持.idl 文件；提供 New Form 命令，此命令产生一个表单，与 Class Wizard 不同的是，此命令也可以产生含有适当选项的对话框。Visual C++ 6.0 提供了新的资源类型：HTML，可以通过选择 Insert 菜单项中的 Resource 项插入 HTML 资源。还提供了两种新的工程类型：扩展存储程序（Extended Stored Procedures）工程类型，支持 OLE DB；实用工具（Utility）工程类型，支

持在编译连接中无需连接的工程。另外还提供在 IDE 中使用环境变量。

7. 向导（Wizard）

MFC AppWizard 提供了更多的选项来生成一个可执行程序，它还同时支持基于和不基于文档/视结构的单文档或多文档应用程序，从而提高了用 MFC 开发此类应用程序的效率；AppWizard 还提供了类似 IE 控件的工具条；除了 DAO、ODBC 外，AppWizard 还提供对 OLE DB 的支持；为了产生帮助文件，AppWizard 采用自定义构造规则，而不用 MakeHelp.bat，这样，必要时，每一个资源文件才重新编译，同时保留 MakeHelp.bat 以提供向下兼容。在 ATL 对象向导中提供了数据访问（Data Access）项，包括两个向导：消费者（Consumer）及提供者(Provider)。在 WizardBar 及 ClassView 中提供了删除成员函数的命令；在 ClassWizard 中提供了 IE4 控件支持；另外在自定义 Wizard 中提供了新的内容，支持工具工程、MakeFile 工程、静态库、控制台应用程序及 ActiveX 控件工程；Visual C++ 6.0 还提供了新的扩展存储程序（Extended Stored Procedure ）向导以生成 SQL Server 扩展存储程序（外部 C 函数，可在 SQL Server 上调用的 win32 动态库函数输出）。Visual C++ 6.0 提供了非 MFC 工程类型的向导，如 win32 应用程序、静态库、动态链接库等，而不只是产生一个空的、一般的工程，这些向导使你能够指定选项，为非 MFC 应用程序产生样板文件代码。

8. OLE DB （Object Link ）

OLE DB 是来自微软的一种最新的数据访问方法。它将数据访问技术与 COM 相结合。Visual C++ 6.0 通过提供 OLE DB 模板来抽取 COM 的界面。这些模板简化了 OLE DB 编程。你可以使用 OLE DB 消费者（consumer）模板的 Schema Rowset 类来获得元数据，这些类封装许多获取元数据的细节。

9. MFC （Microsoft Fundation Class Library）

通过使用新类 ColeDocObjectItem，AppWizard 提供了动态文档容器（Active Document Contaiment），为文档提供了单一的结构，而不是要求用户为每一个文档类型产生或采用多种应用程序结构。这与原先的 OLE 技术不同，原先 OLE 作用的嵌入对象所在的复合文档中，只有一项内容可激活，而采用动态文档容器时，用户可以激活整个文档，包括相关的菜单和工具条。新类 CHtmlView 使用户的 MFC 应用程序成为一个有效的 Web 浏览器。新类 CRebar 以控件的形式提供包含添加的子窗口的工具条。

10. 数据库支持（Database Support）

在专业版中，Visual C++ 6.0 提供了 ADO（ActiveX Data Objects）数据绑定型对话框向导。为 ADO 及 OLE DB 提供了绑定型控件。ADO 数据控件使你可以用 ADO 而不用 RDO 进行数据访问，而且不局限于基于 SQL 的 RDBMS 进行数据访问。在数据库工具中，对 Oracle 的支持得到了加强，用户可以采用可视化设计工具管理 Oracle 数据库。

11. 实用工具

Visual C++ 6.0 提供了部件管理器（Component Manager），用户可以在软件工具及定制的可重用的部件间存储和共享对象。Visual C++ 6.0 还提供了 HTML 帮助生成系统（Help

Workshop），提供基于 HTML 的可与网页集成的帮助。Visual C++ 6.0 也提供了最新的 InstallShield 版本，最新的测试容器（Test Container），增添了对 OCX96 的支持。在企业版中，还提供了可视的 Studio 分析器（Visual Studio Analyzer）和可视的模型器（Visual Modeler），通过分析器提供的客户端的信息，用户可以了解分布式应用程序的运行情况，模型器根据用户的模型产生 C++代码。

12. Windows NT 4.0 选项包（Option Pack）

Visual C++ 6.0 的企业版包含 Windows NT 4.0 选项包，它包含一系列新的应用程序服务，可以用来开发基于 Windows NT Server 4.0 操作系统的新一代分布式网络应用程序。

使用过 Visual C++5.0 的读者对于 Visual C++ 6.0 提供的这些新特性可能很快就有了一个大致的印象，而对于初学者来说，可能不太明白，这不要紧，在以后的章节中将陆续对以上内容作进一步的介绍。

1.3 用户界面基础

用户界面是开发环境的一部分，用户通过这个界面显示信息、指定操作。Visual C++的用户界面包括了集成的窗口、工具、菜单、目录以及其他元素，用户可以在同一个环境下创建、测试、优化应用程序，也可以在 Visual C++中操作其他类型的文档，如 Microsoft Excel 或 Microsoft Word 文档。

Develop Studio 用户界面使用标准的 Windows 界面，但在功能上具有一些附加的特性，使得开发环境更易于使用。使用最多的基本特性有：窗口、文档视图、工具条、菜单、目录及快捷键。

用户也可以根据需要定制 Develop Studio，例如，创建一个与某个工程工作空间相关的所有窗口的布局图，也可以定制工具条和快捷键。

1.3.1 窗口与文档视图

不管是编程或是进行调试，Visual C++工作区中的窗口及文档视图都是完全可调整的，用户可以根据自己的喜好放置窗口。每一个工程都有各自的设置。

窗口类型有两种：文档窗口及泊位窗口。文档窗口的位置和大小可以在 Develop Studio 窗口内进行改变，如最大化和最小化。泊位窗口可以漂浮在应用窗口的边界上，或者在屏幕的任何一个地方，如工程工作空间窗口、输出窗口、调试窗口（变量窗口、观察窗口等）。

窗口类型的布局，即窗口的可见性、位置和大小，是与工程相联系的（就文档窗口而言），或者是与编辑或操作相联系的（就泊位窗口而言）。一旦选择了一个布局，那么该布局的设置将保留在用户正在工作的工程当中。如果用户关闭该工程，然后再打开它，那么文档窗口将具有用户最后一次使用的布局，即具有同样的大小和位置。当用户创建泊位窗口或工具条的布局时，不管用户是用来调试、编辑还是以全屏方式查看，这些布局将适用于所有后续的任务，直到用户改变布局。

另外，有一些窗口既不是文档窗口也不是泊位窗口。例如，当用户使用 Tools 菜单中的实用工具时，其特征由实用工具程序来决定。

1. 窗口的操作

由于 Developer Studio 是基于 Windows 系统的，因此，其窗口的操作大都与其他 Windows 应用程序一样，如窗口的移动、大小改变等。下面介绍一些 Developer Studio 特有的一些操作：

（1）在 Developer Studio 主窗口中显示或隐藏状态条

单击 Tools 菜单中的 Options 选项，出现如图 1.1 所示的对话框，然后选择 WorkSpace，选择 Display statue bar 显示状态条，或清除 Display statue bar 隐藏它。状态条可以显示很多有用的状态信息，如在最左边的文本区中，描述了当前选择的菜单命令或鼠标的按钮动作。状态条还显示了当前操作的进程信息，对文本编辑窗口而言，它显示插入点的行列位置，记录 Record 键的状态和列模式，以及文件是否被设成只读模式等等。

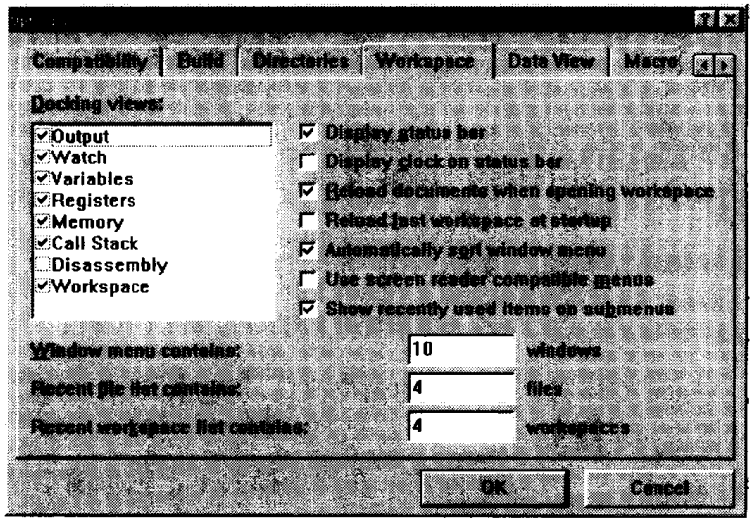

图 1.1　Option 对话框

（2）在状态条上显示时钟操作如步骤（1），只需在图 1.1 的对话框中，选择 Display clock on statue bar 即可。

（3）选择 Automatically sort window menu 项，就可将最近用过的窗口名放置在 Windows 菜单列表的上部。

2. 文档窗口

文档窗口包括编辑器窗口（如文本或资源编辑器）、资源浏览窗口（当用于工程工作空间中的活动工程资源时例外）。

文档窗口是与工程空间相联系的。当用户关闭一个工程时，Develop Studio 会记录它所含有的文档窗口的位置、大小、设置以及任意窗口的切分等等。当用户再次打开该工程时，这些特性就会被恢复。当使用文本编辑器或其他资源编辑器时，可以通过全屏模式切换，使用户可以在全屏模式下，能够一次看到更多的文本或资源信息。

文档窗口的操作：

（1）平铺所有的文档窗口

在 Window 菜单中点击 Windows，从对话框中选中所要平铺的窗口，再点击 Tile Horizontally 或 Tile Vertically。

（2）重叠文档窗口

在 Window 菜单中，选中 Cascade。

（3）切分文档窗口

在 Window 菜单中，点击 Split，将拖动窗口中的切分条至适当的位置，然后单击鼠标即可设置切分条的位置。

（4）显示和隐藏窗口，当用户打开一个工程时显示工程文档

在 Tools 菜单中选择 Options，再单击 Workspace 跳格键；选择 Reload Documents When Opening Workspace ，这样就可以显示工程文档。

（5）启动或禁止全屏模式

在 View 菜单中选择 Full Screen 即可启动全屏模式。单击 Full Screen 或按下[Esc]终止它。当用户第一次选择全屏模式时，将会显示一个带有一小幅计算机屏幕图象的按钮，用户可以单击[Esc]来退出全屏模式。

（6）改变全屏模式窗口的设置

先启动全屏模式，按下[Alt]+[T]显示 Tools 菜单，选择 Options，然后单击 Editor，选择想使用的 Windows settings，然后单击 OK。

在全屏模式下，菜单条是自动隐藏的，用户也可以用键盘来显示菜单。

3. 泊位窗口

在用户操作中，Visual C++打开了不同的泊位窗口，例如，当用户在源文件窗口运行代码时，输出窗口就被用来显示结果。

泊位窗口有两种显示模式：浮动和停泊。在浮动模式下，泊位窗口带有一个极细的标题条，可以出现在屏幕的任何位置。通常浮动窗口总是在其他窗口的上面。在停泊模式下，窗口则被固定到 Developer Studio 主窗口的某个边界上。

对泊位窗口可以进行如下操作：显示和隐藏一个泊位窗口；在一个泊位窗口的停泊区扩展和收缩该窗口；浮动窗口和停泊窗口互换等等。

4. 窗口中的快捷菜单

许多窗口都包含快捷菜单，该菜单中的命令是与窗口的当前状态相适宜的。快捷菜单使用户可以快速地访问当前的工作窗口。例如，当用户在一个编辑窗口中进行编辑时，快捷菜单会显示 Cut、Copy 和 Paste 命令，而当用户在进行调试的时候，快捷菜单则会显示 Toggle Breakpoint 和 QuickWatch 命令。

显示快捷键的方法是在任何一个窗口中单击鼠标右键。

在沿着应用窗口边界的停泊条上，也有与之相关联的快捷菜单。若在停泊条区域或工具条上单击鼠标右键，即可弹出快捷菜单，并显示一些命令，通过这些命令，可以控制显示或隐藏停泊的窗口以及定制工具条。

窗口的操作比较简单，读者在上机过程中很快就能掌握。

1.3.2 工具条

工具条是用来显示操作工具信息的一种显示标志条。第一次以默认配置启动 Developer Studio 时，标准的工具条将会出现在菜单条的正下方，当你打开不同的编辑器，相应的工具

条会自动出现。因此，用户可以根据需要或屏幕大小，任意选择显示工具条的数目。工具条的操作包括改变大小、添加菜单、放大工具条中的按钮，以及移动工具条到不同的地方以满足用户需要。

菜单条是位于屏幕顶部的特殊的工具条，它包含了如 File、Edit、Build 等菜单。菜单条与其他工具条的区别在于：当菜单条处于水平位置时，它总是要占有一整行；除全屏模式外，菜单条是不能被隐藏的。注意，以上所述适用于 Developer Studio 工具条，对操作过程中的其他工具条则未必适用。例如，当在 Developer Studio 中打开一个 Word 文档时，Developer Studio 中的工具条将会被 Word 工具条所代替，而这时这些 Word 工具条的功能与一个 Word 应用窗口是一样的。当用户在 Developer Studio 中操作不同的文档如 Word、Excel 或其他文档软件，Developer Studio 的工具条是不可用的。

1.3.3　目录设置

Developer Studio 提供了一些选项来帮助用户设置和搜索用户工程目录中的文件目录。初始目录路径是由安装程序决定的，仅适用于表 1.1 所列出的一些特定的文件类型。系统根据这些路径来更新 Directories 列表。

表 1.1　特定文件类型的路径内容

文件类型	路径内容
执行文件	确定在何处创建应用程序，如 NMAKE、CL、LINK、BSCMAKE 和 reside 等
头文件	确定编译器在何处寻找头文件（如#include <stdio.h>）
库文件	确定连接器在何处寻找库以便进行外部引用
源文件	确定调试器在何处寻找缺省的原文件,如 MFC 和 Microsoft 实时运行库

1.3.4　键盘加速键

键盘加速键给那些喜欢键盘而不喜欢用鼠标的人提供了一种方便的执行命令的方法。Developer Studio 中的一些命令在缺省情况下有被指定的键盘加速键。而有一些命令并没有指定相应的键盘加速键。用户可以改变已有的键盘加速键，也可创建新的键盘加速键。

1.4　本 章 小 结

本章简要地介绍了 Visual C++ 6.0 的开发环境，读者阅读本章后可以对 Visual C++ 6.0 的特性有一个初步的了解。

Visual C++ 6.0 包括 3 个版本，即标准版、专业版和企业版，不同的版本其功能特性略有不同，读者可以根据需要和兴趣选择一个版本进行学习与开发。

相对于 Visual C++ 5.0，Visual C++ 6.0 增加包括编辑器的动态提示功能等许多新的功能特性，在后继章节将对这些新特性进行详细介绍。

开发环境的用户界面也在这里简单介绍了一下，读者可以了解界面元素基本概念、界面操作、并可以根据自己的喜好对界面进行简单的设置。

第 2 章　工程及其应用

工程包含了用户在开发、编译、连接和调试应用程序时所需的所有文件。这些文件可以产生一些程序或二进制文件。在 Visual C++中，工程是在工程空间中组织完成的。一个工程空间包含多个工程，可以是相关的，也可以是独立的工程。工程工作空间是一个开发工程的容器。当用户创建一个新的工程时，工程工作空间将会同时产生。在用户创建了一个工程空间后，可以进行：

- 添加新的工程（包含独立工程）。
- 添加已有工程的新配置。
- 添加子工程（独立工程）。工作空间目录是工程空间的根目录，要添加的工程可以在不同的路径，甚至是在不同的驱动器中。

2.1　工程工作空间文件

创建一个工程工作空间时，系统会产生一个工程工作空间文件，工程扩展名为.dsw。此文件用来存储位于工程空间一级的信息。同时，系统也会产生一些其他的相关文件，包括一个用来构造工程或子工程的工程文件和一个存储工程工作空间设置的选项文件（.opt）。

注意：如果用户在一个成员组中工作，只要将工程工作空间文件和工程文件置于源代码控制下，就可以与组中的其他成员共享，这样其他成员就可以构造定义在该工程工作空间中的工程了。但是，工作空间选项文件（扩展名为.opt）是不能共享的，因为它包含了该计算机的特定信息。当 Developer Studio 打开某一工作空间而没有找到相应的工作空间选项文件时，它会自动地再创建一个工作空间选项文件。如果用户使用的是输出 makefiles，就不能共享这些文件。其他用户可以共享工程工作空间文件和工程文件，但不能在本地输出 makefiles。

2.2　工程工作空间（Project Workspace）窗口

工程工作空间窗口包含了一组标签面板，这些面板显示了与用户的工程相关的信息视图，如图 2.1 所示。

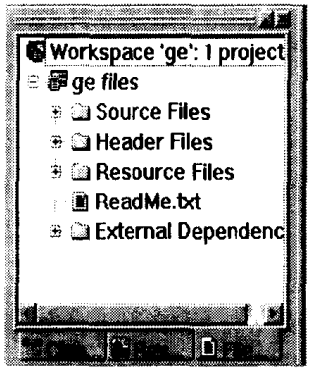

图 2.1　标签面板

它们用于用户查看工程和联机文档。每一个标签对应一个视图。通常工程视图包含：FileView、ClassView、ResourceView 和 DataView 等。每一个工程视图包含一个或多个文件夹，它们包含了工程工作空间的不同元素。扩展该文件夹可以得到工程工作空间在该视图中的详细信息。Project Workspace 窗口可包含的视图及其属性如表 2.1 所示。

表 2.1　工程工作空间窗口可包含的视图及其属性

视 图 名	描　　述
FileView	显示用户已创建的工程。扩展其中的文件夹可以显示工程中的所有文件
ClassView	显示在用户的工程中所定义的 C++类，扩展其中的文件夹可以显示类，而扩展一个类可以显示它的所有元素
ResourceView	显示包含在工程中的资源文件。扩展其中的文件夹可以显示资源类型
DataView	显示数据库工程中关于 ODBC 数据源的信息。DataView 仅在包含了数据库工具的 Visual C++专业版中出现
InfoView	显示联机文档的目录列表

单击 Project Workspace 窗口底部的标签可切换到相应的视图中。一般视图是分层的。扩展文件夹和其他项可以显示其内容。在工程视图中所有不能进一步扩展的项都是可编辑的。

2.3　工程工作空间的操作

工程工作空间的操作包括打开已有的工程空间和创建新的工程空间。

打开已有的工程工作空间的步骤如下：

① 在 File 菜单中选择 Open Workspace；

② 选择包含要打开的工程工作空间的路径和驱动器；

③ 从 File Name 框中选择.dsw 文件单击 OK。

使用 Visual C++，有两种方法创建一个新工程工作空间：第一种方法是使用 New Project Wizard，创建一个最初的工程，Visual C++ Wizards 将自动创建一个包含该工程类型的初始文件的工程工作空间。第二种方法是创建一个空的工程工作空间，这样就要自己加入工程和文件。具体操作步骤如下：

① 在 File 菜单中选择 New；

② 单击 Workspaces 标签；

③ 从类型列表中选择 Blank Workspace，在 Workspace Name 框中输入工程工作空间的名称。如果必要的话，可以在 Location 框中输入工程工作空间文件存储的路径。

2.4　工程的元素

工程工作空间可以包含如下内容：

- **工程**　一系列空或非空源代码，带有一个或多个配置。工程通常指明要构造的应用程序的类型，用户的工程工作空间可以包含任意数目的工程（包含子工程）。
- **配置**　指明输出文件运行的平台及编译输出的工具设置，用户可以为工程添加任意数量的配置。缺省情况下，当用户创建一个新工程，也就创建了 Debug 和 Release 配置。

在工程工作空间中，包含以下两类工程：

1. 顶级工程

顶级工程是不依赖于其他工程的工程（不是任何一个工程的子工程），一个工程工作空间至少含有一个顶级工程。

2. 子工程

子工程是与其他工程有依附关系的工程，构造系统决定是否需要在构造父工程之前构造子工程。任何工程都可以是其他任何工程的子工程。

2.5 工程的类型

每一种工程都有一个工程类型。每种工程类型指定了要创建的内容，同时也指定了缺省的设置。例如指明编译源代码的设置、用于连接工程配置文件的库、输出文件的缺省位置等等。Visual C++的工程类型如下：

ATL COM AppWizard
Cluster Resource Type Wizard
Custom AppWizard
Database Project （Enterprise Edition only）
DevStudio Add-in Wizard
Extended Stored Procedure AppWizard
ISAPI Extension （Internet Server API） Wizard
Makefile
MFC ActiveX ControlWizard
MFC AppWizard （DLL version）
MFC AppWizard （.EXE version）
Utility Project
Win32 Application
Win32 Console Application
Win32 Dynamic Link Library
Win32 Static Library

2.6 工程的配置

工程是由配置设置来控制的。工程的配置具有一个层次结构，工程配置级设置适用于配置的所有文件，如果用户需要用与整个配置设置不同的设置来编译文件时，也可以为单个文件或文件夹指明设置。例如，用户在为一个配置指明缺省优化，所有在这个配置中的文件都将使用缺省的优化，用户也可以为其中任意一个文件单独确定它的优化设置，或者不进行任何优化设置。

在一个工程中，用户可以设定以下三个级别的工程配置：

- **工程配置级** 在本级设定的设置将适用于所有行为。除了在文件级被重载外，任何

位于工程配置级的特定设置都适用于该工程中的所有文件。

- **文件夹级**　在本级设定的设置适用于文件夹级的行为，如编译等。这些设置适用于整个文件夹，并且会在工程设置级的设置。
- **文件级**　在本级设定的设置适用于文件级的行为，如编译等。但这些配置只适用于文件本身，并且会重载工程配置级的设置。

2.7　工程的使用

工程的使用包括以下几类操作：

1. 设置活动工程

活动工程是指当使用 Build 或 Rebuild All 命令时将被连接的工程。它是构造独立工程的决定因素。步骤如下：在 Project 菜单中，单击 Set Active Project，从子菜单中选择工程名，或在 Build 工具条中，从 Set Active Project 下拉列表中选择工程。

2. 插入或删除工程

用户可以向工程工作空间插入新的工程或已有工程，可以删除工程的配置而不删除整个工程。在已有的工程工作空间中添加新的工程的步骤如下：

① 打开工程工作空间；

② 在 File 菜单中，按下 New 然后单击 Projects 标签；

③ 选择工程类型；

④ 指定工程名和位置；

⑤ 单击 Add to Current Workspace 按钮；

⑥ 选择任何一个可用的平台，以便创建初始的 Debug 和 Release 配置，单击 OK，即完成了操作。

3. 从工作空间中删除一个工程

从 FileView 视图中，选择该工程，按下 Delete。在删除工程时，系统将它作为子工程从工作空间的其他工程中移走。

4. 从工程中添加或移去文件

用户将文件添加到工程时，也就把文件添加到该工程的所有配置中，例如，用户的工程名为 Myproject，并带有 Debug 和 Release 配置，向 Myproject 工程添加的文件就把该文件添加到这两个配置中。

如果从当前工程工作空间以外的目录中增加文件，Visual C++在工程的相应扩展名为.Dsp 的文件中为这些文件使用绝对路径。由于使用的是绝对路径，其他用户很难共享此文件。

为了支持工程目录外的文件共享，用户可以在源代码属性页使用 Persist As 域。在 Persist As 域中，将路径用环境变量替换，保存在工程文件中。例如，将位于 c:\basefiles\mybase.cpp 添加到工程中，工程位于 c:\myproject\project1，用下面两种方法使用 the Persist As 域：

（1）在 the Persist As 域中，改变字符串“c:\basefiles\mybase.cpp”为“..\..\basefiles\mybase.cpp”。这要求工程的所有用户都要保持同样的目录结构。

（2）定义环境变量，如"BASEFILES"替代用户机器上的"c:\basefiles"，这样就可以将 Persist As 域上的字符串由"c:\basefiles\mybase.cpp"改为"$（BASEFILES）\mybase.cpp"。使用这种方法，不同的用户只需定义相应的变量就可实现共享，而不需要保持同样的目录结构。

5. 向工程中添加文件

首先打开工程，在 Project 菜单中，按下 Add To Project，然后单击 Files；在 Files of type 框中，指明要添加的文件类型；选择一个或多个文件，单击 OK。如图 2.2 所示。

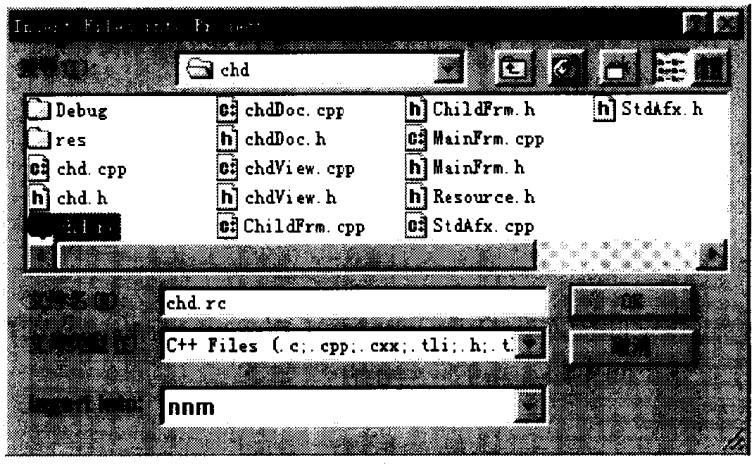

图 2.2 向工程添加文件

用户也可以使用 FileView 中的弹出菜单更快地添加文件。

6. 把一个工程工作空间的文件复制或移动到另一个工作空间

从 FileView 视图中选择要复制或移动的文件；在 Edit 菜单中，选择 Cut/Copy 命令；关闭当前工作空间；打开目标工程工作空间；选择接受文件的工程，在 Edit 菜单中，单击 Paste 命令即可实现移动或复制的功能。

7. 创建工程配置

一个工程中包含了很多设置，这些设置决定了该工程的输出文件特性。当用户创建一个新的配置时，其初始配置通常来源于某个已有的工程配置。此时，新的工程配置与已有的工程配置具有同样的文件组。新配置的使用对用户正在构造的工程会产生很多影响，例如，改变操作平台，确定不同的优化选项等等，都可以改变该工程的输出文件特性。创建一个工程配置的步骤如下：

① 在 Build 菜单中选择 Configurations。

② 在 Configurations 对话框中，选择需要添加配置的工程，然后按下 Add。

③ 在 Add Project Configuration 对话框中，输入新配置的名称。

④ 在 Copy Settings From 下拉列表框中，选择所需配置，新配置将从此配置中复制初始化设置。

⑤ 在 Platform 下拉列表框中选择新配置的平台。