面向对象程序设计系列教材

Visual C++ 6.0

程序设计简明教程

主编 范 辉
副主编 刘惊雷 傅 铭

高等教育出版社

面向对象程序设计系列教材

Visual C++ 6.0 程序设计简明教程

主编　范辉　副主编　刘惊雷　傅铭

高等教育出版社

内容提要

Visual C++ 6.0 是 Microsoft 公司开发套件中的主推产品，是当今程序员首选的开发工具之一。它在编程的深入性、运行的快速性等方面具有强大的功能。

本书对 Visual C++ 6.0 的使用与开发做了全面、系统的分析，着重针对有一定编程基础的读者，列举了大量详尽的实例，从易到难、循序渐进地把 Visual C++ 6.0 程序设计的主要特色及难点展示给读者。内容包括：VC 的基本概念——类与对象，绘图程序的设计，菜单的设计，文本的操纵与显示，键盘与鼠标消息的运用，对话框程序的设计，各种常用控件的使用，位图文件的显示，类的建立，动态链接库的建立与使用，利用 MFC 构造 ActiveX 控件以及 VC 在数据库程序设计上的应用等。

本书可作为高等院校有关专业本科及研究生的教材或参考书，其中列举的实例对程序设计人员也有一定的参考价值。

图书在版编目(CIP)数据

Visual C++ 6.0 程序设计简明教程/范辉等编著. —北京：高等教育出版社，2001

ISBN 7-04-009239-5

I. V… II. 范… III. C 语言-程序设计-教材 IV. TP312

中国版本图书馆 CIP 数据核字（2001）第 18738 号

责任编辑 关 旭 **封面设计** 李卫青 **版式设计** 马静如
责任校对 陈 荣 **责任印制** 杨 明

Visual C++ 6.0 程序设计简明教程
范 辉 主编

出版发行 高等教育出版社
社 址 北京市东城区沙滩后街 55 号 **邮政编码** 100009
电 话 010-64054588 **传 真** 010-64014048
网 址 http:// www.hep.edu.cn
http:// www.hep.com.cn

经 销 新华书店北京发行所
印 刷 中国农业出版社印刷厂

开 本 787×1092 1/16 **版 次** 2001 年 7 月第 1 版
印 张 23.75 **印 次** 2001 年 7 月第 1 次印刷
字 数 580 000 **定 价** 24.90 元

前　言

面向对象的程序设计技术是当今全球程序员普遍采用的一种程序设计方法，它是软件开发的最新潮流。在众多的面向对象程序设计语言中，Microsoft(微软)公司的 Visual C++ 6.0 独树一帜，它将面向对象的程序设计方法与可视化的软件开发环境完美地结合起来，使得开发 Windows 平台的应用程序更加方便、深入。因此，目前大多数 VC 程序员采用 MFC(Microsoft Foundation Class,Microsoft 基本类库)来进行程序的编制，它在很大程度上屏蔽了隐藏在上层应用程序背后的复杂性。所以，本书的全部例子都是采用 AppWizard 生成的基于 MFC 的应用程序。

本书共分十七章，主要介绍了 VC 的基本概念——类与对象，绘图程序的设计，各种形式菜单的设计，文本的操纵与显示，键盘与鼠标消息的运用，对话框程序的设计，各种常用控件的使用，位图文件的显示，如何自己建立一个新的类，如何建立与使用动态链接库，如何利用 MFC 构造一个 ActiveX 控件以及如何将 VC 应用在数据库程序设计上。

目前，介绍 VC 的书很多，而本书的特色是理论联系实际，不将内容停留在使用层次，而是提出问题（Windows 系统中常见的功能，如如何在程序中实现变化的进度条、走动的时钟以及如何在程序中修改注册表等）让读者思考，然后给出实现的方法。对于每一章，不仅介绍了与该章有关的一些基本知识，还分别列举一个综合实例，一步一步引导读者完成，并在实例中给出了详尽的解释。在每一个例子中，不仅重点介绍了本章新的知识点，同时还把本章以前所学知识（包括一些常用的函数的用法及一些 VC 原理的再次体会）作一些回顾。更为重要的是，这些实例引出了后面章节中将要出现的知识。相信只要读者认真研读每一个程序，并亲自上机实践，一定会有较大收益。另外，本书中涉及的程序，读者如有兴趣，可与作者联系索取。

本书是作者在参考了大量的 VC 书籍与相当多共享的 VC 源代码的基础上，结合多年从事开发和教学工作的经验编写而成。本书在写作过程中得到了太原理工大学的余雪丽教授、陈俊杰教授、谢克明教授、段富教授及山西省经贸委信息中心的任小芬高工的大力支持，他们对本书提出了许多有价值的建议，在此一并表示感谢。最后衷心希望读者能够沿着本书所搭建的阶梯，通过不断学习实践，顺利达到 Windows 程序设计的高峰。

本书由范辉任主编，刘惊雷和傅铭任副主编。其中第一章、第二章、第三章、第四章、第五章、第十六章由刘惊雷编写，第十一章、第十二章、第十七章由傅铭编写，第六章、第七章、第八章、第九章由华臻编写，第十三章、第十四章、第十五章由孟亮编写，其余章节由范辉编写，全书由范辉统稿。青岛海洋大学的王锐教授和魏振刚副教授担任主审。

由于时间仓促，疏漏、错误在所难免，希望读者和广大同仁给予批评指正。

编　者

2001 年 2 月于太原理工大学

目　录

第一章　VC++ 6.0 概述

1.1　VC 的特点

可视化与面向对象的编程技术是当今世界流行的编程技术，在众多的开发工具中，Microsoft 公司的 VC++ 6.0 独树一帜，深得广大程序员的青睐。VC 的执行速度和对操作系统访问的权限之高，是其他许多语言难以比拟的，加之 Windows 操作系统的支持，就使得 VC 的高级程序员对整个计算机的硬件系统和软件系统在各个方面的访问和控制更加游刃有余。正如一位学者所说："用 VC 的眼光看，整个计算机都是透明的，或者说是完全裸露的。"可见 VC 对系统的访问深度是如何之深。

1.1.1　VC 的特点

VC 的核心是其 MFC 类库，MFC 建立在 C 语言 Windows 应用程序编程接口(API)之上，以 C++格式封装了大量的 Windows API，大量杂乱无章的函数在 MFC 的组织下变得有条不紊，且其基本函数的格式都相仿，大约 80%的 API 函数在 MFC 中都能得到实现。因此学 VC，精通MFC 是关键。VC 主要特点有：

① 程序中既可有 MFC 函数，也可直接调用 Windows 函数。因为大部分常用功能用 MFC 完成，而少数 MFC 难以胜任的任务，也可通过 API 函数这些最低层的调用来完成。

② 程序的代码精简，运行速度快。正因为如此使用 VC 的程序员都有一种"可以主宰计算机一切资源的自豪感"。

③ 利用应用程序向导(AppWizard)可以在很短时间内创建一个 Windows 的应用程序框架，而这种应用程序框架，为用户建立了一个稳固而又安全的骨架。

④ 利用类向导(ClassWizard)可以在利用 AppWizard 向导生成的应用程序框架之上，快速增加新的类、虚拟函数、成员变量、消息处理函数，它可以生成类的结构，可以编写函数的原型声明及实现函数体，并可方便而又迅速地将某个消息映射到某个处理函数上。而这些都是应用程序框架下的"血肉"。

⑤ 与操作系统紧密结合，硬件对程序员透明。有这样的观点，凡是在 Windows 操作系统上看到的功能，用 VC 都能实现。

⑥ 快速的数据库访问技术，可以方便地使用 ODBC、DAO、OLEDB 技术操作本地或异地数据库。而处理数据的快速与灵活性，会使用户对 VC 有一种"舍我其谁"的感觉。

1.1.2　VC++ 6.0 的特点

① VC++ 6.0 提供了一个高度集成的工具集。这使得在开发应用程序的整个过程中都有较

高的效率。如今的 VC++ 6.0 已是集编辑、编译、运行、调试为一体的功能强大的集成编程环境，因此为大型工程项目的开发提供了强有力的支持。

② 编辑器的语句自动完成特征。这种特征是由 IntelliSense 系统实现的，当用户编辑代码时，IntelliSense 会在光标或鼠标位置附近显示类的成员函数、变量以及全局变量。用户可以在成员列表中选择成员来插入到代码中。使用 IntelliSense，用户还可以查看代码注释、函数声明和变量类型等信息，如图 1.1 所示。

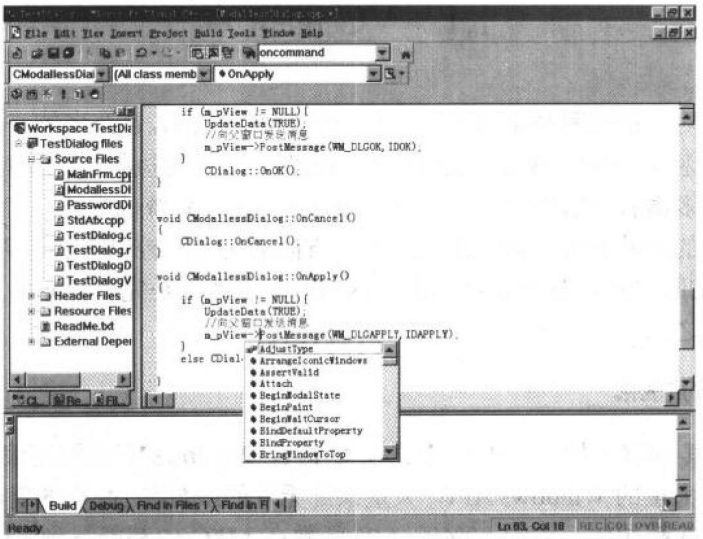

图 1.1 Intellisense 自动显示类的成员函数、变量

③ 在线帮助与集成环境相分离。在 VC++ 6.0 中，帮助放在 MSDN 中，它是一个独立的应用程序，可以单独运行。用户在集成开发环境中调用帮助系统时，系统会自动打开这个应用程序。启动后的 MSDN 如图 1.2 所示。

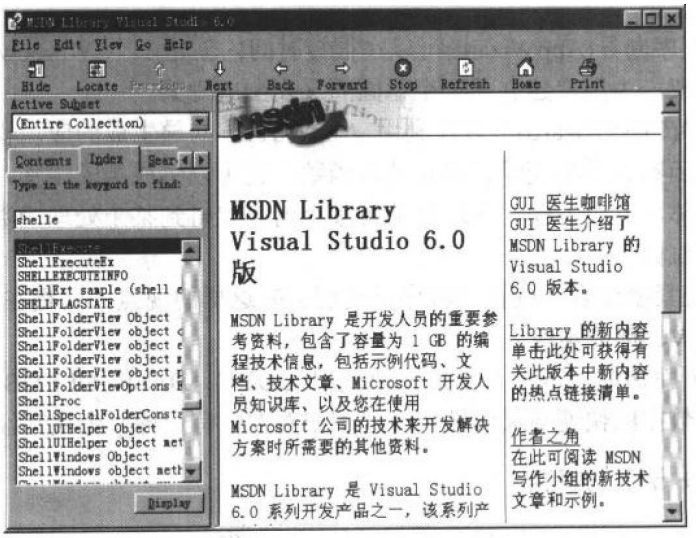

图 1.2 独立运行的 MSDN

1.2 VC 6.0 新增功能

VC 6.0 虽然在界面上与 VC 5.0 差别不大，但在实现技术上却有较大改进，下面分几个方面讨论。

1.2.1 编译器与连接器的改进

VC 6.0 的编译器不仅支持 COM(组件对象模型)应用程序开发，而且还进一步简化了 COM 应用程序的开发过程，同时它在代码生成方面更加优化，使得生成的目标代码更短小，代码运行速度更快。在对不同微处理器的支持上，VC 6.0 更有建树，因为用 VC 6.0 生成应用程序时，它可以根据微处理器的不同而产生不同的目标代码，提高了 VC 程序在不同硬件平台上的移植能力。

1.2.2 MFC 类库功能的增强

作为 VC 基石的 MFC 库到了 6.0 版本更是包罗万象，它不仅具有了 MFC 6.0 以前版本的所有功能，更重要的是 MFC 6.0 的许多特性使开发者可以支持新的 Microsoft Active Platform，其具体内容包括：

① 封装了作为 Internet 网络浏览器(Internet Explorer 4.0)的一部分，引入了的新的 Windows 通用控制的 MFC 类。

② 对 Dynamic HTML 的支持，它允许 MFC 程序员创建可以动态操作和生成 HTML 主页的应用程序。

③ Active Document Containment，它允许基于 MFC 的应用程序动态操作和生成 Active Document。

1.2.3 对 Internet 强有力的支持

VC 6.0 允许用户开发基于 Internet 的应用程序，允许异步下载文件和设置应用程序的属性，并且在任务完成后，应用程序会自动释放系统资源供其他应用程序使用。

1.2.4 快速的集成数据库访问

VC 6.0 的 MFC 库增加了对 DAO 的支持，并将原来的 ODBC API 函数进行了封装，提供了一系列 ODBC 类，以支持 ODBC 3.0 标准。

1.3 MFC 类库概述

建立 MFC 库的目的是为了简化 Windows 应用程序的编制，MFC 开发小组的工作是将此目的和程序设计原理结合起来。下面简单介绍一下 MFC 的开发历程，它将有助于读者正确理

解和使用 MFC 库。

微软于 1989 年成立了 Application Framework 开发小组，开始设计一个用于建立图形应用程序的可移植的 C++类库。AFX 的目的是建立一个单一的类库，用来简化 Windows、OS/2 Presentation Manager 和 Apple Macintosh 应用程序的开发。当时，开发小组希望利用面向对象的技术建立一个易于使用并可移植的 GUI。然而一年以后，AFX 计划失败了，其主要原因是 AFX 开发小组过于信任面向对象的程序设计方法的能力，一个纯粹抽象的类往往并不实用。

因此，AFX 开发小组转向开发 MFC 库，并将其实现目标从多目标平台转向对 Windows API 的封装。他们也不再试图建立高度抽象的类，而重点在于建立实际可用的类。他们还避开使用复杂的 C++的结构(例如，多重继承和模板)，而只是使用了一个 C++的子集。另外，他们不再使用抽象的类来隐藏 Windows 本身 API 的细节，而是使用户能够很方便地存取 Windows 原有的 API。这样做有两个原因：首先，可以简化已有的 Win16 代码到 MFC 库的转换。其次，确保原有 API 的所有功能对应用程序都是可以使用的——这意味着可以绕过类库而直接调用这些 Windows 原有的 API 函数。

AFX 和 MFC 的另一个区别是所使用的开发方法不同，AFX 采用自顶向下的设计方法，逐步将对象抽象出来并施加到 Windows 上；而 MFC 库采用的则是自底向上的设计方法，也就是说，从 MFC 的第一个版本开始，它的类就是建立在已有的 Windows API 对象基础上的，这样就避免了 AFX 中代码的超支与浪费。

当读者使用 MFC 进行编程时，将会发现在 MFC 源程序中有很多名称包含“AFX”的字符串，如函数名 AfxGetApp()，常量名 AFX_IDS_APP_TITLE，数据结构名 AFX_MSGMAP_ENTRY 等。这些名称在 Windows API 函数中是不存在的，它们是 AFX 开发小组遗留的痕迹，而 MFC 的基本框架正是建筑于这个基础之上，所以 MFC 多多少少保留了该开发小组的标志——AFX。可见 Microsoft 基本类库(MFC)封装了 SDK(软件开发工具包)结构、功能及应用程序框架内部技术。该应用程序框架隐藏了过去 Windows 程序员不得不处理的许多重复性工作。

AFX 开发小组根据他们开发第一个类库的经验，确定了在建立 MFC 库时需要遵循的一些基本原则。下面的设计原则，也就是他们的设计目标：

① 使用 C++创建 Windows 应用程序的过程容易和直观。

② 合并已有的 Windows 模型和概念。

③ 确保类库有足够的扩展性，以便加入 Windows 后续版本的特性和功能。

④ 使用 Windows API 特性更容易，同时采用 Windows API 的概念、编程风格和函数名称。

⑤ 使用标准 Windows 命令约定和编码风格。

⑥ 用类封装 Windows API 和 Windows 对象，并按照 API 的指引完成工作。

几乎全部 MFC 库都从 CObject 派生，这意味着形成主类层次的各种子系统构成 CObject 派生类。下面给出 CObject 类及其派生的 MFC 子类的描述。

1.3.1 根类 CObject 与 CRuntimeClass 结构

CObject 和 CRuntimeClass 是 MFC 中两个非常重要的类结构，绝大部分 MFC 类都是以 CObject 作为基类，CRuntimeClass 结构同 CObject 密不可分，了解它们对于深入理解 MFC 具

有重要意义。

1. CRuntimeClass 结构

要理解 CObject，先来看一下 CRuntimeClass 这个在 MFC 中至关重要的结构。

每个从 CObject 中派生的类都有一个 CRuntimeClass 对象同它关联以便完成在运行时得到类实例的信息或者它的基类。 在 afx.h 中它的定义如下:

```
struct CRuntimeClass
{
// Attributes
LPCSTR   m_lpszClassName; //类名，一般是指包含 CRuntimeClass 对象的类的名称
int m_nObjectSize;
//包含 CRuntimeClass 对象的类 sizeof 的大小，不包括它分配的内存
UINT m_wSchema; // schema number of the loaded class
CObject* (PASCAL* m_pfnCreateObject)();
// NULL => abstract class 指向一个建立实例的构造函数
#ifdef _AFXDLL
CRuntimeClass* (PASCAL* m_pfnGetBaseClass)();
#else
CRuntimeClass* m_pBaseClass;
#endif
/*以上 m_pBaseClass 的指针(函数)是 MFC 运行时确定类层次的关键，它是一个简单的单向链表 */
// Operations
CObject* CreateObject(); //这个函数给予 CObject 派生类运行时动态建立的能力
BOOL IsDerivedFrom(const CRuntimeClass* pBaseClass) const;
/*这个函数使用 m_pBaseClass 或 m_pfnGetBaseClass 遍历整个类层次确定 pBaseClass 指向的类是否基类，使用它可以判断某类是否是从 pBaseClass 指向的类派生而来。*/
// Implementation
void Store(CArchive& ar) const;
static CRuntimeClass* PASCAL Load(CArchive& ar,  UINT* pwSchemaNum);
// CRuntimeClass objects linked together in simple list
CRuntimeClass* m_pNextClass; // linked list of registered classes
};
```

2. CObject 类

CObject 类是 MFC 的抽象基类，是 MFC 中多数类和用户自定义类的根类，它为程序员提供了进入所编程序的许多公共操作。主要包括：对象的建立与删除、序列化(即串行化)的支持、对象诊断输出、运行时信息以及集合类的兼容等。它的作用是，使得 MFC 以及基于 MFC 的应用程序更加稳固和易于调试。

注意：要想使用 CRuntimeClass 结构得到运行时类的信息，必须在类中包括 DECLARE_DYNAMIC/IMPLEMENT_DYNAMIC ， DECLARE_DYNCREATE/IMPLEMENT_

DYNCREATE 或 DECLARE_SERIAL/IMPLEMENT_SERIAL。但类必须是从 CObject 派生的才能使用这些宏，因为通过 DECLARE_DYNAMIC 将定义一个如下的函数：

```
CRuntimeClass* PASCAL B::_GetBaseClass()
{
return RUNTIME_CLASS(base_class_name);
}
```

其中的 RUNTIME_CLASS 是这样定义的

```
#define RUNTIME_CLASS( class_name ) \
(CRuntimeClass *)(&class_name::class##class_name);
```

也就是得到类中的 CRuntimeClass 对象指针。显而易见，如果没有基类而用 IMPLEMENT_DYNAMIC 时将得到一个编译错误。除非像 CObject 一样不用 DECLARE_DYNAMIC 而定义和实现这些函数，CObject 中的 GetBaseClass 只是简单的返回 NULL。实际的 DECLARE_DYNAMIC 和 IMPLEMENT_DYNAMIC 在 afx.h 中有具体的声明，第一级的宏是 DECLARE_DYNAMIC/IMPLEMENT_DYNAMIC，它允许在运行时处理类名和类层次中的位置，允许做有意义的诊断 Dump。第二级的宏是 DECLARE_SERIAL/IMPLEMENT_SERIAL，它包括第一级宏所有的功能，允许进行对象的序列化。有兴趣的读者可以查看 afx.h。

CObject 实现这些功能绝大部分是通过它里面的 CRuntimeClass 对象 classObject 实现的，且它不支持多重继承，即表示以 CObject 为基类的类层次中只能有一个 CObject 基类。之所以会这样，就是因为 CRuntimeClass 对象的成员 m_pBaseClass 的关系，它只是一个单链表。

要想彻底了解序列化，不得不了解 Archive 类，后面的章节中将详细论述这个类。

1.3.2 应用程序体系结构类

1. 命令相关类——CCmdTarget 类

该类是 CObject 的子类，是 MFC 库中所有具有消息映射属性的基类，消息映射规定了当对象接到消息命令时，应调用哪一个函数对该消息进行处理。消息命令来自菜单项、命令按钮和加速键的信息。由 CCmdTarget 派生出的主框架类包括 CView、CWinApp、CDocument、CWnd 和 CFrameWnd。程序员若需要一个新类来处理消息时，可以从 CCmdTarget 派生出一个新类。程序员很少从 CCmdTarget 类中直接派生出新类。

2. 窗口应用程序类——CWinApp 类

每一个使用 MFC 的应用程序只能包含一个从 CWinApp 派生的应用程序对象，当 C++的其他全局对象被构造时，这个对象也同时被构造，当 Windows 调用 WinMain 函数时，产生的对象已经有效了。在全局的层次上说明用户的 CWinApp 对象，在运行程序中该对象与其他对象相互协调。

(1) 访问 CWinApp 对象以及其他全程信息的全局函数

① AfxGetApp 获得一指向 CWinApp 对象的指针。

② AfxGetInstanceHandle 获得当前应用程序实例的句柄。

③ AfxGetResourceHandle 获得一应用程序资源的句柄。

④ AfxGetAppName 获得一指针，它指向一个包含应用程序名的字符串。相反，如果有一个指向 CWinApp 对象的指针，使用 m_pszAppName 可以取得应用程序的名称。

(2) CWinApp 的公共数据成员

① m_pszAppName 指定应用程序的名称。

② m_hInstance 指定应用程序的当前实例。

③ m_hPrevInstance 指定应用程序的前一个实例。

④ m_nCmdShow 指定窗口最初将如何显示。

⑤ m_pMainWnd 包含一个指针，它指向一个应用程序的主窗口。

⑥ m_pszExeName 指定应用程序的模块名称。

⑦ m_pszHelpFilePath 指定应用程序的 Help 文件的路径。

(3) CWinApp 的公共成员函数

① LoadCursor 调取光标资源。

② LoadStandardCursor 调取 Windows 预定义光标，在 windows.h 中指定 IDC 常量。

③ LoadIcon 调取一个图标资源。

④ LoadStandardIcon 调取 Windows 预定义的图标，在 windows.h 中指定 IDI 常量。

⑤ GetProfileString 在程序的.INI 文件中的一个入口获取一个字符串。

⑥ WriteProfileString 在程序的.INI 文件中的一个入口写一个字符串。

⑦ AddDocTemplate 在程序已存在的文档模板的列表中添加一个文档模板。

⑧ GetFirstDocTemplatePosition 获取第一个文档模板的位置。

⑨ GetNextDocTemplate 获取下一个文档模板的位置。

⑩ OpenDocumentFile 框架调用此函数从一个文件中打开一个文档。

⑪ AddToRecentFileList 向最近使用(MRU)的文件列表中添加一个文件名。

⑫ SelectPrinter 选择用户通过打印机对话框指定的打印机。

(4) CWinApp 的可重载的公共成员函数

① InitApplication 重载以执行任何应用程序层次上的初始化。

② InitInstance 重载以执行 Windows 对象实例的初始化，诸如建立用户窗口对象等。

③ Run 运行缺省的消息循环，重载该函数以自定义消息循环。

④ OnIdle 重载以执行任何应用程序指定的空闲时间处理。

⑤ PreTranslateMessage 在窗口消息被发送到 Windows 函数 TranslateMessage 和 DispatchMessage 以前过滤它们。

⑥ SaveAllModified 提示用户将所有修改过的文档存盘。

(5) CWinApp 的保护成员函数

① LoadStdProfileSettings 加载标准的.INI 文件设置，并使能 MRU 文件列表特征。

② SetDialogBkColor 为对话框和消息框设置缺省的背景颜色。

③ Enable3dConrtols 允许具有三维外观的控件。

3. 应用程序线程类——CWinThread 类

该类为线程的基类，CWinApp 是从 CWinThread 派生而来。该对象描述程序中的线程执行，主线程的执行通常由一个 CWinApp 的派生对象提供。

(1) CWinThread 的公共数据成员

① m_bAutoDeletes 指示线程终止时是否销毁该对象。

② m_hThread 当前线程的句柄。

③ m_nThreadID 当前线程的 ID 值。

④ m_pMainWnd 保存指向程序的主窗口指针。

(2) CWinThread 的公共成员函数

① GetMainWnd 获取一个指向此线程的主窗口指针。

② GetThreadPriority 获取当前线程的优先权。

③ PostThreadPriority 将一条消息传递到另一个 CWinThread 对象。

④ ResumeThread 减小一个线程的挂起计数。

⑤ SuspendThread 增加一个线程的挂起计数。

4. 文档/视窗类

文档对象由文档模板对象创建，管理应用程序的数据。视窗对象表示一个窗口的客户区，它显示文档数据并允许用户与之交互。

① CDocTemplate 类 文档模板的基类。它负责协调文档、视窗和框架窗口的创建。

② CMultiDocTemplate 类 多文档界面(MDI)的文档模板。

③ CSingleDocTemplate 类 单文档界面(SDI)的文档模板。

④ CDocument 应用程序专用的文档的基类。

⑤ CView 显示文档数据的应用程序专用视图的基类。

1.3.3 可视对象类

1. 窗口类

CWnd 类提供了 MFC 中的所有窗口类的基类。CWnd 与 Windows 窗口有所不同，但它们又是紧密相连的。CWnd 对象是由 CWnd 的构造函数和析构函数建立和取消的，而 Windows 窗口与此不同，它是 Windows 的一个结构，是由 Create 成员函数建立的，DestroyWindow 函数破坏 Windows 窗口而不是破坏对象。

(1) CWnd 类的公共数据成员

m_hWnd 标识连接到该 CWnd 对象上的 HANDLE。

(2) CWnd 类的初始化函数

① Create 创建并初始化与该 CWnd 对象相联系的子窗口。

② PreCreateWindow 在建立连接到这个 CWnd 对象的 Windows 窗口之前调用此函数。

③ GetStyle 返回当前窗口的风格。

④ SubclassWindow 将一个窗口连接至一个 CWnd 对象，并使其通过该 CWnd 的消息映射传递消息。

⑤ FromHandle 给出一个窗口句柄时，返回一个 CWnd 对象的指针。

⑥ GetSafeHwnd 返回 m_hWnd，若此指针为 NULL，则返回 NULL。

(3) CWnd 类的窗口状态函数

① EnableWindow 允许或禁止鼠标和键盘输入。