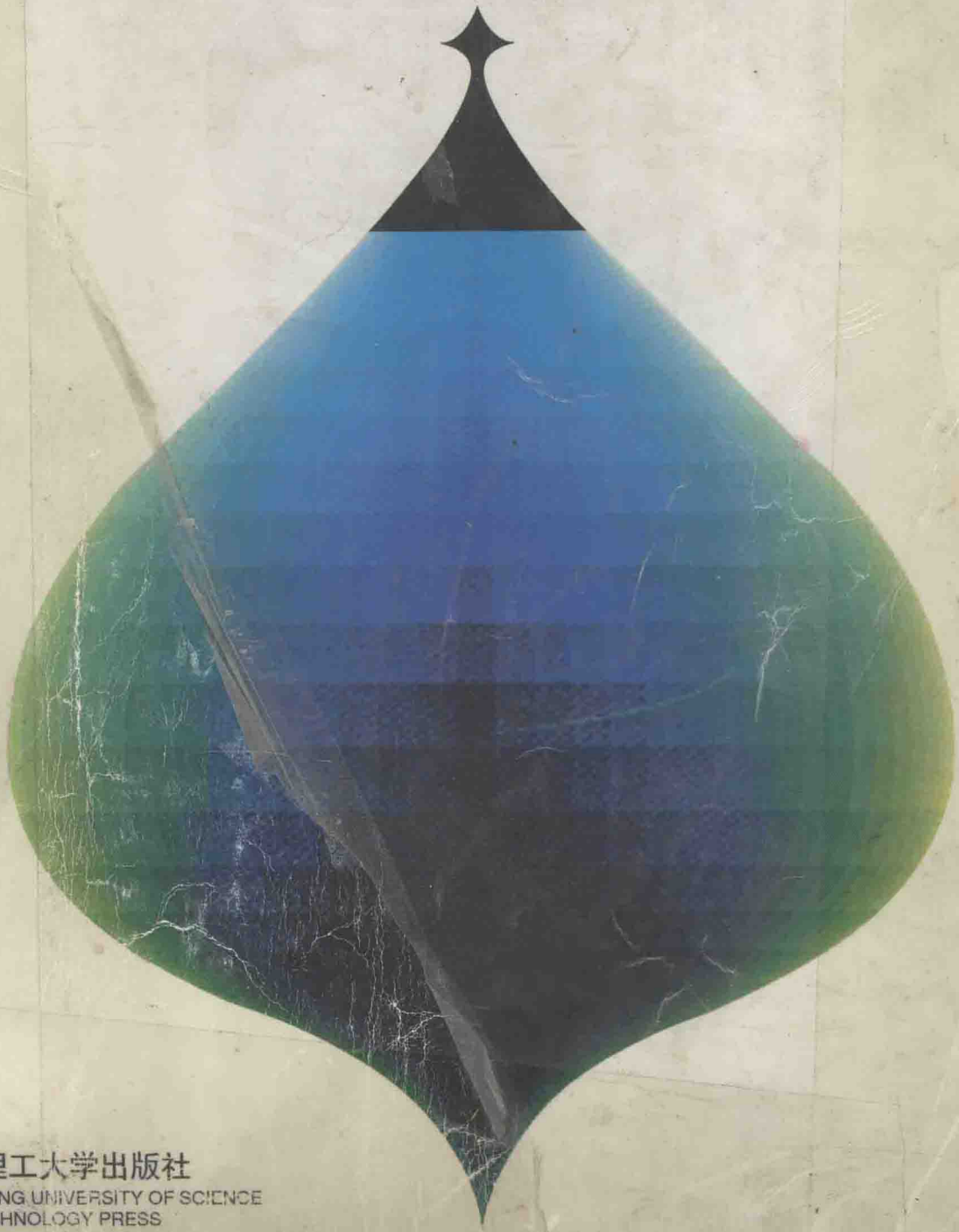
计算机程序设计与软件开发系列丛书

MFC深入浅出

——从MFC设计到MFC编程

李久进 编著



华中理工大学出版社
ZHONG UNIVERSITY OF SCIENCE
TECHNOLOGY PRESS

计算机程序设计与软件开发系列丛书

MFC 深入浅出

——从 MFC 设计到 MFC 编程

李久进　编著

华中理工大学出版社

图书在版编目(CIP)数据

MFC 深入浅出——从 MFC 设计到 MFC 编程/李久进　编著
武汉:华中理工大学出版社，　1999 年 9 月
ISBN 7-5609-2058-6

Ⅰ. M…
Ⅱ. 李…
Ⅲ. 程序设计-参考资料
Ⅳ. TP311

MFC 深入浅出——从 MFC 设计到 MFC 编程　　李久进　编著

责任编辑:周芬娜　　封面设计:梁书亭
责任校对:熊九龄　　监　　印:张正林

出版发行:华中理工大学出版社　　武昌喻家山　邮编:430074　电话:(027)87542624

经销:新华书店湖北发行所

录排:华中理工大学惠友科技文印中心
印刷:荆州市今印集团有限责任公司

开本:787×1092　1/16　　印张:17.75　　字数:408 000
版次:1999 年 9 月第 1 版　　印次:1999 年 9 月第 1 次印刷　　印数:1—5 000
ISBN 7-5609-2058-6/TP・343　　定价:28.00 元

内 容 提 要

这是一本关于 MFC 核心类库的书。全书深入浅出地分析了 MFC 的关键要素，揭示了 MFC 以面向对象的方法简化 Windows 编程的奥秘。内容可以分 5 个方面：

第一，讨论 MFC 对 Windows 对象和底层 API 的封装，介绍 MFC 的窗口类、设备上下文类、GDI 类的使用与实现，以及根类 CObject 的特性和实现方法。

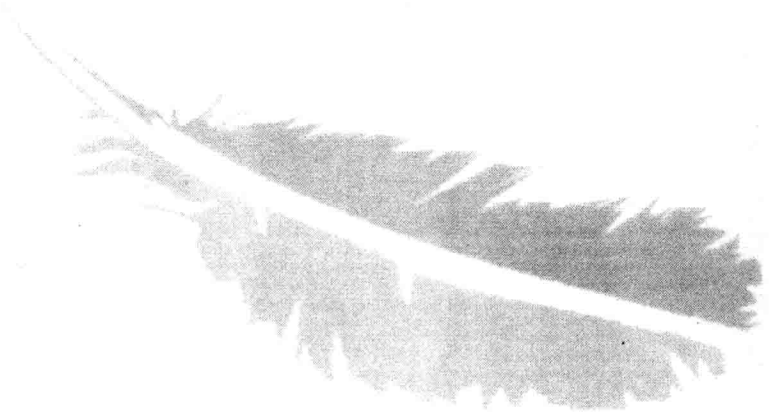
第二，讨论 MFC 的窗口过程和消息映射，揭示 MFC 通过消息映射机制实现 C++虚拟函数功能的内幕，并详尽地分析了多种消息的映射和处理过程。

第三，讨论 MFC 的文档-视编程模式，沿着 MFC 应用程序的启动和退出顺序，揭示了 MFC 以文档模板为中心创建 MFC 对象以及这些对象相互作用直到最后销毁的过程。

第四，讨论 MFC 的动态链接库和对多进程、特别是多线程编程的支持机制，分析 MFC 的模块、线程、模块线程状态的设计和实现，深入讨论 MFC 规则 DLL、扩展 DLL、MFC 对象和 Windows 对象的映射、资源查找等内容。

第五，讨论一些 MFC 类的实现。阐述如何设计并实现特定目的的类。讨论 MFC 的文件类、对话框相关的类、工具条、状态栏、Socket 类等。

本书的目的在于帮助读者了解或者深入地理解、更好地使用 MFC。



约定和说明

本约定和说明是针对第一章至第十三章的。

1. 图解或者说明的流程都是 MFC 的缺省实现。

2. 对于 Win32 全局函数，用“::”为前缀，以区分 MFC 的成员函数；如果从上下文可以明确判定一个函数是全局函数，则省掉“::”前缀。

3. 本书图解时，使用灰色框注释。如果注释某个函数是某个类的成员函数，则表示该类定义该函数的最上层的类。

4. 流程图所描述的函数的流程不一定是该函数的程序流程，可能是该函数运行时的执行流程。例如，函数 A 图解为函数 B→函数 C，可以是函数 A 先调用函数 B，然后调用函数 C；或者是函数 A 调用函数 B，函数 B 调用函数 C。

5. 流程图省略了对 OLE 的处理。

6. 流程图中表示调用了某个成员函数，使用了类限制符号。如果用正体表示类名和函数名(形式为 ClassName::FunctionName)，则程序源码中没有类名约束，这可分几种情况：如果是虚拟函数则表示该函数动态约束的结果是调用了指定类的函数；如果是消息处理函数则表示指定类的消息处理函数被调用；如果不是上述两种情况，表示该函数调用了指定类的实现。如果用斜体表示类名和函数名(形式为 *ClassName::FunctionName*)，则程序的源码明确使用了类的限制符号来调用函数。

7. 动态连接到 MFC DLL 定义了_AFX_DLL 符号。引用的 MFC 源码如果定义了该符号，则表示在动态连接情况下使用。

8. “MFC DLL”指 MFC 的核心 DLL，即 MFCXX.DLL。

9. 某个类的对象，例如，CWinApp 对象表示 CWinApp 类的一个实例。“类的实例”和“类的对象”两种说法可以互换。

10. MFC 对象是 C++对象，即一个 C++类的实例；Windows 对象是 Windows 操作系统定义的数据结构的实例。一个 Window 对象对应一个 MFC 对象。

11. “类的成员数据”和“类的成员变量”两种说法可以互换，都表示类的属性。

12. 文档和文件，使用“文件”的地方可以用“文档”代替，表示一个操作系统的文件；使用“文档”的地方一般还有“文档”对象的含义。

13. MFC 窗口类表示 CWnd 或其派生类；Windows“窗口类”表示程序注册的 Windows 的类，书中表示 Windows“窗口类”的地方用引号加以区分。

前　言

现在流行的 Windows 下的编程语言实在不少，所以在 BBS 上常常有人会问：我应该使用什么编程语言呢？其中，有一个大家认可的答案：真正的程序员使用 Visual C++。

的确，Visual C++是一个功能强大、灵活、方便的编程工具，可以完成其他编程语言所无法完成的任务，可以让程序员方便地实现自己的设计，尽情地发挥自己的创造性。

Visual C++的强大无比的功能除了得益于 C++的特性之外，更重要的是由于它具有体系完整、机制灵活、功能丰富的 MFC 类库。

所以，要讲 Visual C++，必须讲 MFC 类库。

MFC 类库可以分两个层次，首先是实现 MFC 编程框架体系的核心 MFC 类库，然后是建立在核心 MFC 类库基础之上的扩展类库，例如，支持 COM 的类库，实现网络功能的类库，等等。随着 Visual C++的不断升级，MFC 类库的功能越来越丰富，越来越强大，但是，MFC 核心类库是相对稳定的，特别是从 Visual C++ 4.2 开始到现在的 Visual C++ 6.0。

本书的中心就是深入浅出地解析 MFC 类库，分析怎么使用 MFC 类库以及 MFC 类库的内部实现，揭开 MFC 复杂、深奥的面纱，让读者对 MFC 有一个全面、透彻、清晰的理解。关于 MFC 的核心实现，主要有以下几个方面。

首先，MFC 采用 C++的面向对象的特征封装了 Windows 的对象和 Win32 函数，一定程度上隐蔽了底层 Win32 的复杂性。

其次，MFC 采用消息映射的方法来处理 Windows 消息和事件，隐藏了 Windows 窗口的窗口过程，简化了消息处理的复杂性和烦琐性。

还有，MFC 提供了一个以文档-视为中心的编程模式，并实现了以文档-视为中心的编程框架，简化了数据处理的过程。

而且，MFC 提供了模块状态、线程状态、模块线程状态来支持多线程的编程设计和 DLL 的编程。

本书分别从使用 MFC 的角度和 MFC 内部设计及实现的角度讨论了上述内容，分析了 MFC 核心的设计和实现；然后，在此基础上，进一步讨论了 MFC 对一些常用类的实现。有关章节的内容如下：

第一章，MFC 概述。

第二章，解释 MFC 对 Win32 API 和 Windows 对象的封装，讨论各类 MFC 对象的使用，分析 MFC 对象和 Windows 对象的关系。

第三章，讨论 CObject 的特性及其实现，包括动态类信息、动态创建、序列化的实现等内容。

第四章，讨论 MFC 的消息映射机制，分析 MFC 对各类消息的处理，例如对 Windows 消息、控制通知消息、命令消息、状态更新消息、反射消息等的处理；并揭示了 MFC 通过消息映射手段实现 C++虚拟函数机制的原理。

第五章和第六章，分析 MFC 编程框架启动和关闭一个应用程序的过程，揭示 MFC 框架的内幕，剖析以文档模板为核心创建基于文档-视的应用程序的过程，展示 MFC 框架处理消息和调用虚拟函数的时机和位置。

第七、八、九章，介绍 MFC 的动态链接库、进程、线程等概念，以及 MFC 动态链接库的种类和使用，讨论 MFC 下多线程编程的问题。并且进一步阐述 MFC 的核心概念之一：状态（模块状态、线程状态、模块线程状态），揭示 MFC 对多线程的支持机制，MFC 实现规则 DLL 和扩展 DLL 的内幕。

第十章，阐述 MFC 下的调试手段。

第十一章，讨论 CFile 类，主要分析了 CFile 的使用和它对 Win32 文件函数的封装。

第十二章，讨论模式和无模式对话框，分析 MFC 如何设计和实现这两种对话框的功能，分析 CDialog 和 CFormView 为实现有关功能而设计的虚拟函数、消息处理函数等。

第十三章，讨论 MFC 工具栏和状态栏的设计及其实现，分析 MFC 是如何以 CControlBar 为基础，派生出 CStatusBar、CToolBar、CDialogBar 等子类，实现 MFC 工具栏和状态栏标准处理。

第十四章，讨论 MFC 的 Socket 类。

第一章到第十章介绍了 MFC 的核心概念以及实现。在此基础上，第十一章到第十四章讨论了 MFC 一些常用类的实现。

本书的内容对 MFC 的初学者（最好对 Visual C++和 Windows 有所了解）和提高者都是很有帮助的。

如果您是一个初学者，可以读第一至第六章。主要目的是建立对 MFC 的全面理解，了解 MFC 框架是如何支持程序员编程的。如果有读不懂的地方，可以跳过，直接阅读有关分析的结论。特别是第五和第六章，可以重点阅读，了解 MFC 是怎样来处理有关消息、调用有关虚拟函数的。

然后，还可以读第十章，第十一至第十四章。特别第十二章，可以重点阅读，它是 MFC 从 CWnd 或者 CView 派生出特定的类实现特定功能的例子，可以帮助您进一步理解 MFC，并且学习如何设计和实现一个特定的类。

如果您对 MFC 有一定的掌握，可以进一步阅读第八和第九章，了解 MFC 处理 DLL 和线程的知识。对于第一至第六章、第十至第十四章，应该把重点放在 MFC 的设计和实现的分析上。这样，可以深化您对 MFC 和 Windows 编程的理解与掌握。

如果您可以较熟练地使用 MFC，建议您进一步阅读第九章，并且对所有有关章节的设计和实现分析作重点阅读，这样，不仅可以帮助您深入地理解和掌握 MFC，而且，从 MFC 的有关内部设计和实现上，必然可以提高您的程序设计和编写能力。

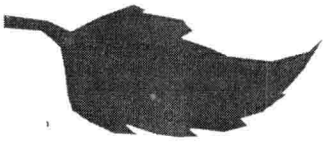
由于成书仓促，书中可能存在一些缺点和错误，恳请您不吝赐教！作者的电子邮箱：jjin@public.szonline.net。

作者

1999 年 6 月

目　录

第1章



MFC 概述

1.1 MFC 是一个编程框架

MFC(Microsoft Foundation Class Library)中的各种类结合起来构成了一个应用程序框架，它的目的就是让程序员在此基础上来建立 Windows 下的应用程序，这是一种相对 SDK 来说更为简单的方法。因为总体上，MFC 框架定义了应用程序的轮廓，并提供了用户接口的标准实现方法，程序员所要做的就是通过预定义的接口把具体应用程序特有的东西填入这个轮廓。Microsoft Visual C++提供了相应的工具来完成这个工作：AppWizard 可以用来生成初步的框架文件(代码和资源等)；资源编辑器用来帮助直观地设计用户接口；ClassWizard 用来协助添加代码到框架文件；编译则通过类库实现应用程序特定的逻辑。

1.1.1 封装

构成 MFC 框架的是 MFC 类库，MFC 类库是 C++类库。这些类或者封装了 Win32 应用程序编程接口，或者封装了应用程序的概念，或者封装了 OLE 特性，或者封装了 ODBC 和 DAO 数据访问的功能，等等，分述如下。

(1) 对 Win32 应用程序编程接口的封装

用一个 C++ Object 来包装一个 Windows Object。例如：class CWnd 是一个 C++ window object，它把 Windows window(HWND)和 Windows window 有关的 API 函数封装在 C++ window object 的成员函数内，后者的成员变量 m_hWnd 就是前者的窗口句柄。

(2) 对应用程序概念的封装

使用 SDK 编写 Windows 应用程序时，总要定义窗口过程，登记 Windows Class，创建窗口，等等。MFC 把许多类似的处理封装起来，替程序员完成这些工作。另外，MFC 提供了以文档–视图为中心的编程模式，MFC 类库封装了对它的支持。文档是用户操作的数据对象，视图是数据操作的窗口，用户通过它处理、查看数据。

(3) 对 COM/OLE 特性的封装

OLE 建立在 COM 模型之上，由于支持 OLE 的应用程序必须实现一系列接口(Interface)，因而相当繁琐。MFC 的 OLE 类封装了 OLE API 大量的复杂工作，这些类提供了实现 OLE 的更高级接口。

(4) 对 ODBC 功能的封装

以少量的能提供与 ODBC 之间更高级接口的 C++类，封装了 ODBC API 的大量的复

杂的工作，提供了一种数据库编程模式。

1.1.2 继承

首先，MFC 抽象出众多类的共同特性，设计出一些基类作为实现其他类的基础。这些类中，最重要的类是 CObject 和 CCmdTarget。CObject 是 MFC 的根类，绝大多数 MFC 类是其派生的，包括 CCmdTarget。CObject 实现了一些重要的特性，包括动态类信息、动态创建、对象序列化、对程序调试的支持，等等。所有从 CObject 派生的类都将具备或者可以具备 CObject 所拥有的特性。CCmdTarget 通过封装一些属性和方法，提供了消息处理的架构。在 MFC 中，任何可以处理消息的类都从 CCmdTarget 派生。

针对每种不同的对象，MFC 都设计了一组类对这些对象进行封装，每一组类都有一个基类，从基类派生出众多更具体的类。这些对象包括以下种类：窗口对象，基类是 CWnd；应用程序对象，基类是 CWinThread；文档对象，基类是 CDocument，等等。

程序员将结合自己的实际，从适当的 MFC 类中派生出自己的类，实现特定的功能，达到自己的编程目的。

1.1.3 虚拟函数和动态约束

MFC 以“C++”为基础，自然支持虚拟函数和动态约束。但是作为一个编程框架，有一个问题必须解决：如果仅仅通过虚拟函数来支持动态约束，必然导致虚拟函数表过于臃肿，消耗内存，效率低下。例如，CWnd 封装 Windows 窗口对象时，每一条 Windows 消息对应一个成员函数，这些成员函数为派生类所继承。如果这些函数都设计成虚拟函数，由于数量太多，实现起来不现实。于是，MFC 建立了消息映射机制，以一种富有效率、便于使用的手段解决消息处理函数的动态约束问题。

这样，通过虚拟函数和消息映射，MFC 类提供了丰富的编程接口。程序员在继承基类的同时，把自己实现的虚拟函数和消息处理函数嵌入 MFC 的编程框架。MFC 编程框架将在适当的时候、适当的地方来调用程序的代码。本书将充分地展示 MFC 调用虚拟函数和消息处理函数的内幕，让读者对 MFC 的编程接口有清晰的理解。

1.1.4 MFC 的宏观框架体系

如前所述，MFC 实现了对应用程序概念的封装，把类、类的继承、动态约束、类的关系和相互作用等封装起来。这样封装的结果对程序员来说，是一套开发模板(或者说模式)。针对不同的应用和目的，程序员采用不同的模板。例如，SDI 应用程序的模板，MDI 应用程序的模板，规则 DLL 应用程序的模板，扩展 DLL 应用程序的模板，OLE/ActiveX 应用程序的模板，等等。

这些模板都采用了以文档-视为中心的思想，每一个模板都包含一组特定的类。典型的 MDI 应用程序的构成将在下一节具体讨论。

为了支持对应用程序概念的封装，MFC 内部必须作大量的工作。例如，为了实现消息映射机制，MFC 编程框架必须要保证首先得到消息，然后按既定的方法进行处理。又如，

为了实现对 DLL 编程的支持和多线程编程的支持，MFC 内部使用了特别的处理方法，使用模块状态、线程状态等来管理一些重要信息。虽然，这些内部处理对程序员来说是透明的，但是，懂得和理解 MFC 内部机制有助于写出功能灵活而强大的程序。

总之，MFC 封装了 Win32 API、OLE API、ODBC API 等底层函数的功能，并提供更高一层的接口，简化了 Windows 编程。同时，MFC 支持对底层 API 的直接调用。

MFC 提供了一个 Windows 应用程序开发模式，对程序的控制主要是由 MFC 框架完成的，而且 MFC 也完成了大部分的功能，预定义或实现了许多事件和消息处理，等等。“框架”或者由其本身处理事件，不依赖程序员的代码；或者调用程序员的代码来处理应用程序特定的事件。

MFC 是 C++类库，程序员就是通过使用、继承和扩展适当的类来实现特定的目的。例如，继承时，应用程序特定的事件由程序员的派生类来处理，不感兴趣的由基类处理。实现这种功能的基础是 C++对继承的支持，对虚拟函数的支持，以及 MFC 实现的消息映射机制。

1.2　MDI 应用程序的构成

本节解释一个典型的 MDI 应用程序的构成。

用 AppWizard 产生一个 MDI 工程 t (无 OLE 等支持)，AppWizard 创建了一系列文件，构成了一个应用程序框架。这些文件分四类：头文件(.h)，实现文件(.cpp)，资源文件(.rc)，模块定义文件(.def)等。

1.2.1　构成应用程序的对象

图 1-1 解释了该应用程序的结构，箭头表示信息流向。

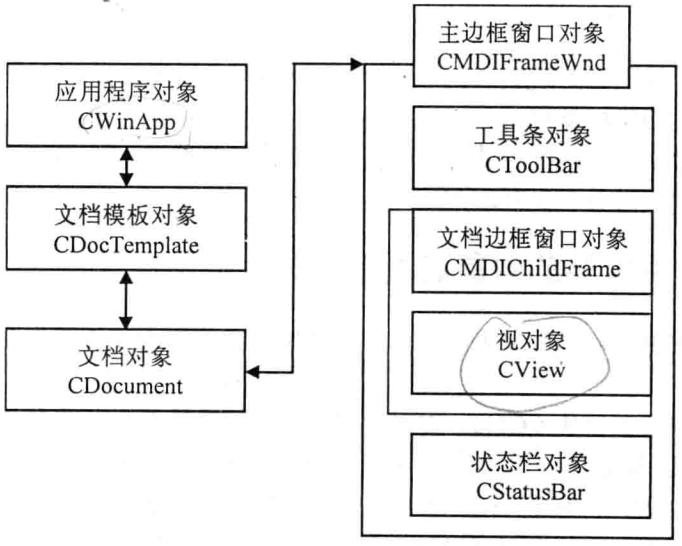


图 1-1　一个 MDI 应用程序的构成

从 CWinApp、CDocument、CView、CMDIFrameWnd、CMDIChildWnd 类对应地派生

出 CTApp、CTDoc、CTView、CMainFrame、CChildFrame 五个类，这五个类的实例分别是应用程序对象、文档对象、视对象、主框架窗口对象和文档边框窗口对象。主框架窗口包含了视窗口、工具条和状态栏。对这些类或者对象解释如下。

(1) 应用程序

应用程序类派生于 CWinApp。基于框架的应用程序必须有且只有一个应用程序对象，它负责应用程序的初始化、运行和结束。

(2) 边框窗口

如果是 SDI 应用程序，从 CFrameWnd 类派生边框窗口类，边框窗口的客户子窗口(MDIClient)直接包含视窗口；如果是 MDI 应用程序，从 CMDIFrameWnd 类派生边框窗口类，边框窗口的客户子窗口(MDIClient)直接包含文档边框窗口。

如果要支持工具条、状态栏，则派生的边框窗口类还要添加 CToolBar 和 CStatusBar 类型的成员变量，以及在一个 OnCreate 消息处理函数中初始化这两个控制窗口。

边框窗口用来管理文档边框窗口、视窗口、工具条、菜单、加速键等，协调半模式状态(如上下文的帮助(Shift+F1 模式)和打印预览)。

(3) 文档边框窗口

文档边框窗口类从 CMDIChildWnd 类派生，MDI 应用程序使用文档边框窗口来包含视窗口。

(4) 文档

文档类从 CDocument 类派生，用来管理数据，数据的变化、存取都是通过文档实现的。视窗口通过文档对象来访问和更新数据。

(5) 视

视类从 CView 或它的派生类派生。视和文档联系在一起，在文档和用户之间起中介作用，即视在屏幕上显示文档的内容，并把用户输入转换成对文档的操作。

(6) 文档模板

文档模板类一般不需要派生。MDI 应用程序使用多文档模板类 CMultiDocTemplate；SDI 应用程序使用单文档模板类 CSingleDocTemplate。

应用程序通过文档模板类对象来管理上述对象(应用程序对象、文档对象、主边框窗口对象、文档边框窗口对象、视对象)的创建。

1.2.2 构成应用程序的对象之间的关系

这里，用图的形式可直观地表示所涉及的 MFC 类的继承或者派生关系，如图 1-2 所示。

图 1-2 所示的类都是从 CObject 类派生出来的；所有处理消息的类都是从 CCmdTarget 类派生的。如果是多文档应用程序，文档模板使用 CMultiDocTemplate，主框架窗口从 CMDIFarmeWnd 派生，它包含工具条、状态栏和文档框架窗口。文档框架窗口从 CMDIChildWnd 派生，文档框架窗口包含视，视从 CView 或其派生类派生。

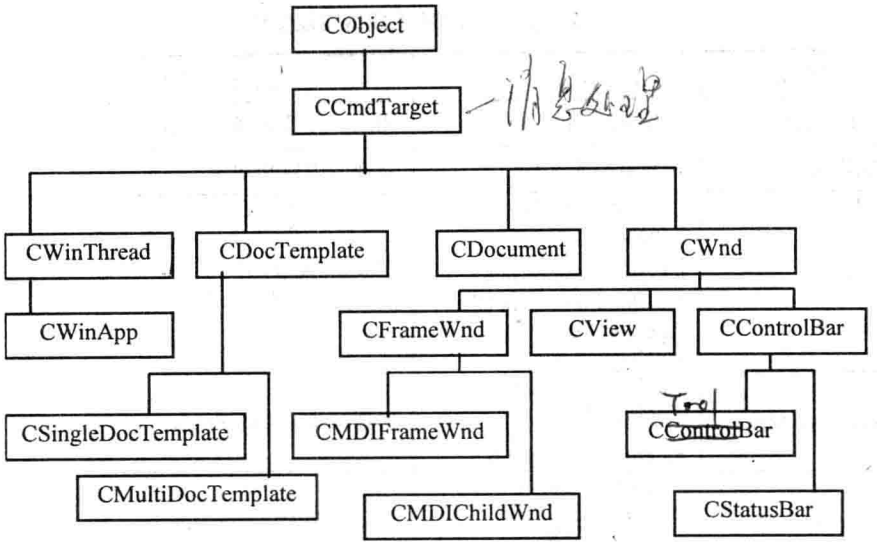


图 1-2　一些 MFC 类的层次

1.2.3　构成应用程序的文件

通过上述分析，可知 AppWizard 产生的 MDI 框架程序的内容、所定义和实现的类。下面，从文件的角度来考察 AppWizard 生成了哪些源码文件，这些文件的作用是什么。表 1-1 列出了 AppWizard 所生成的头文件，表 1-2 列出了 AppWizard 所生成的实现文件及其对头文件的包含关系。

表1-1　AppWizard所生成的头文件

头文件	用　途
stdafx.h	标准 AFX 头文件
resource.h	定义了各种资源 ID
t.h	#include "resource.h" 定义了从 CWinApp 派生的应用程序对象 CTApp
childfrm.h	定义了从 CMDIChildWnd 派生的文档框架窗口对象 CTChildFrame
mainfrm.h	定义了从 CMDIFrameWnd 派生的框架窗口对象 CMainFrame
tdoc.h	定义了从 CDocument 派生的文档对象 CTDoc
tview.h	定义了从 CView 派生的视图对象 CTView

从表 1-2 中的包含关系一栏可以看出：

CTApp 的实现用到所有的用户定义对象，包含了他们的定义；CView 的实现用到 CTDoc；其他对象的实现只涉及自己的定义。

当然，如果增加其他操作，引用其他对象，则要包含相应的类的定义文件。

对预编译头文件说明如下：