陈坚 陈涛 编著

利用 Visual C++ 2.0/4.0 编制 Windows 95应用程序

西安电子科技大学出版社

利用 Visual C++ 2.0/4.0 编制 Windows 95 应用程序

陈 坚　陈 涛 编著

西安电子科技大学出版社

1997

(陕)新登字 010 号

内 容 提 要

本书详尽介绍了使用 Visual C++ 2.0/4.0 设计 Windows 95 应用程序的各种技术，并通过丰富的范例帮助读者迅速掌握这一技术。

本书分十二章，共五个部分：第一和第二章为第一部分，作为本书的基础，简单介绍了 AppWizard 、AppStudio 和 ClassWizard 编程工具、MFC 基本类库以及如何使用 AppWizard 生成应用程序框架；第三到第五章为第二部分，具体讨论了用户界面设计技术，包括对话框、上下文菜单、工具条、状态条、对话条以及 Windows 95 新的通用控制等；第六到第八章为第三部分，介绍了 Windows 95 编程的一些高级技术，诸如图形图像显示、联机帮助系统、切分窗口、"画中画"等；第九到第十一章为第四部分，介绍了 Windows 95 的一些系统知识和关键技术，包括 OLE 控制、内存分配、钩子、进程和线程；第十二章为第五部分，详细介绍了 Windows 95 风格的安装软件设计技术。

本书适用于熟悉 C 语言、掌握基本的 C++知识、了解 C++继承特性的读者。Windows 95 编程初学者、熟悉 Windows SDK 或 Borland 公司的 OWL 编程的程序员，以及准备从 Windows 3.x 转向 Windows 95 编程的程序员都能从本书找到感兴趣、有用的章节。

利用 Visual C++ 2.0/4.0 编制 Windows 95 应用程序

陈 坚 陈 涛 编著

责任编辑 李惠萍 殷咸安

西安电子科技大学出版社出版发行

地址：西安市太白南路 2 号 邮编：710071

陕西省富平县印刷厂印刷

新华书店经销

开本 787×1092 1/16 印张 22 12/16 字数 540 千字

1997 年 5 月第 1 版 1997 年 5 月第 1 次印刷 印数 1—6 000

ISBN 7-5606-0529-X/TP·0260 定价：29.50 元

前　言

Microsoft 公司的新一代 32 位操作系统 Windows 95 自 1995 年 8 月问世以来，广为流行，至今一年多的时间销售量已超过 3000 万套。同时它引发了计算机界一场新的软件革命，国内外各厂家纷纷推出 Windows 95 应用程序。据统计资料表明，在各种软件销售中，Windows 软件独占鳌头，一枝独秀。

与此同时，Microsoft 相继推出 32 位 Windows 开发工具——Visual C++ 2.0 和 Visual C++ 4.0。"Visual"就是"可视的"，Visual C++就是可视编程语言。利用 Visual C++可以简化程序员的编程，提高工作效率：

● 利用 AppWizard 可以简单快速地生成一个应用程序框架，而不需要程序员加入一句程序代码。

● 利用 AppStudio 资源编辑器可以使用户界面的创建简单直观，所见即所得。

● 利用 ClassWizard 可以使消息或命令自动与消息处理函数或控制函数相联系，程序员只需考虑函数核心代码的实现。

● 利用 Visual C++ 4.0 的 Component Gallery 可以加入现成的组件，诸如 OLE 控制、闪烁窗口、定制状态条等组件。

Visual C++的核心是 Microsoft 基本类库(Microsoft Foundation Classes，MFC)，称之为"应用程序框架"。它一方面用类封装了 Windows 95 API，另一方面使用称为"消息映射"的机制把 Windows 消息和命令传递到窗口、文档、视图以及 MFC 应用程序中的其它对象。因而 MFC 成功地将面向对象和事件驱动编程概念联系起来，并得到很好的配合。使用 MFC 编写 Windows 95 应用程序简单方便，代码量小。

本书共分十二章：

第一、第二两章是本书的基础。其中第一章简单介绍了 MFC 应用程序结构、编程工具以及 MFC 3.0 类的层次结构。第二章介绍了 AppWizard 生成应用框架的步骤以及自动生成的各文件内容。

第三到第五章涉及的是用户界面设计技术。其中第三章详细讨论了 Windows 极为重要的用户接口——对话框的构成、种类、创建和实现的整个过程。第四章讨论了另外两种非常重要的用户接口——菜单和控制条，其中特别介绍了提供简单快捷操作的上下文菜单，它在 Windows 95 中得到广泛的应用。另外还介绍了一种全新的控制条——对话条的设计技术。第五章介绍了 Windows 95 比 Windows 3.x 用户界面更友善和更易使用的关键技术之一——Windows 95 新的通用控制特性与编程。

第六到第八章介绍了 Windows 95 编程的一些较高级的技术。图形图像作为多媒体技术的核心技术，在第六章详细介绍了它们的显示处理技术。联机帮助系统已成为 Windows 应用程序不可缺少的部分，在第七章中给予了详细介绍。第八章首先介绍了文档模板、切分窗口和"画中画"等另外一些深层次的内容，然后通过一个完整的综合实例来帮助读者更好地掌握前面各章介绍的编程技术。

从第九章到第十一章介绍了 Windows 95 的一些系统知识和关键技术。其中第九章介绍了取代 VBX 控制的 OLE 控制的编程技术。第十章首先介绍了 Windows 95 的全新的线性内存体系结构，然后具体介绍了 Windows 95 的各种内存分配技术，最后讲解了 MFC 内存诊断技术。第十一章首先介绍了用于监视系统中各种消息的钩子和钩子过程，然后介绍了 Windows 95 实现抢先多任务机制的关键技术——线程，最后介绍了多线程的同步技术。

最后，第十二章介绍了如何将一个实用软件进行最后一步包装，即安装软件的设计技术。

为了帮助读者迅速掌握介绍的内容，本书给出了丰富的范例程序。所有程序全部经严格测试，无需调试。读者完全可以即拿即用，从而节省了大量宝贵的时间，提高了工作效率。本书还附有配套软盘，包含所有源程序，需要者可与西安电子科技大学出版社发行部联系。另外，有关 Visual C++2.0/4.0 编程技术介绍还可参见由西安电子科技大学出版社出版的《Windows 95 多媒体应用程序设计技术》一书。

从本书的酝酿到编写，自始至终都得到西安电子科技大学出版社李惠萍老师的关心和指导，在此表示衷心的感谢。

限于作者学识水平有限和时间仓促，书中难免有不足和错误之处，恳请各位同仁及广大读者给予批评指正。

作　者

1997 年 1 月 18 日于南京

目　录

第 1 章

Visual C++和基本类库

Microsoft 推出的 Visual C++ 2.0/4.0 和 Microsoft 基本类库(Microsoft Foundation Classes，MFC)，成功地将面向对象和事件驱动编程概念联系起来，并得到很好的配合，使得编写 Windows 95 应用程序的过程变得简单、方便，且所得程序的代码量小。为了使读者对 MFC 编程有一个整体了解，本章首先概述 MFC 应用程序结构等一些基础知识。

本章主要内容为：

- MFC 框架、文档和视图结构
- AppWizard、AppStudio 和 ClassWizard
- 利用 MFC 开发应用程序的过程
- 调试技术
- MFC 3.0 类的层次结构
- Visual C++ 4.0 新特性

1.1　MFC 框架、文档和视图结构

传统地利用 SDK 编写 Windows 应用程序时首先要包含起始函数 WinMain(如同 DOS 程序的 main 函数)，然后是初始化应用程序和注册窗口类(同时将创建的窗口与窗口回调过程相联系)，接着是一个消息循环。无论处理 Windows 消息还是命令消息，都使用 switch - case 结构。一方面，每个应用程序的结构大致相同；另一方面，由于编程规则多(如 WM_PAINT 消息处理中必须包含成对的 BeginPaint 和 EndPaint 函数)和大量的 switch - case 结构而造成程序臃肿、可维护性差、可移植性差。MFC 从根本上改变了这些缺陷，它实现两项功能：一是定义了一个应用程序的初始行为，二是对 Windows API 进行了类封装。MFC 类不仅封装处理 Windows API 的所有细节，还简化了 OLE(对象链接与嵌入)2.0 和 ODBC(开放式数据库互连)等处理过程。

MFC 既支持单文档接口(SDI)应用程序，还支持多文档接口(MDI)应用程序。SDI 应用程序(如 NOTEPAD 记事本)一次只允许打开一个文档框架窗口，而 MDI 应用程序(如 Visual C++工作平台，如图 1.1 所示)允许在同一个应用实例中打开一个或多个文档框架窗口。MFC 支持 MDI 应用程序的“Last File Used(最近使用过的文件)”列表。另外，MFC 对工具栏

的泊位或漂浮、联机帮助也提供自动支持。

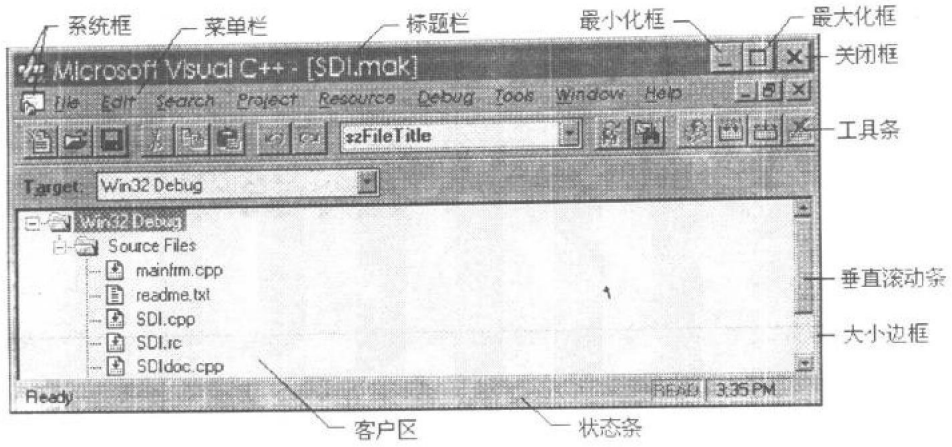

图 1.1　MDI 应用程序及其组成

一个 SDI 或 MDI 程序主要由框架(Frame)、文档(Document)和视图(View)构成。单文档窗口和 MDI 各子窗口都是框架窗口，它的核心是“文档－视图”结构。用户处理的数据单位就是一个文档对象。文档对象用于维护、加载及保存数据，它是由 File 菜单中的 New 或 Open 命令创建的，一般都保存在一个文件中。用户通过文档之上的视图与文档进行交互。视图是镶嵌在框架窗口客户区域内的一个窗口，它显示其关联的文档的数据，接收鼠标与键盘输入并把它翻译为选择与编辑等操作。另外通过用户接口对象，如菜单、按钮，将命令发送给文档、视图及应用程序中的其它对象，然后由这些对象完成命令的执行。

具体而言，我们可以认为应用程序主要由以下几个对象构成：

1. 应用程序对象

应用程序类(从 CWinApp 派生而来，MFC 各类介绍见 1.4 节)完成应用程序的初始化和清除工作。有且只有一个应用程序对象，它创建和管理所有文档类型的文档模板。

2. 线程对象

应用程序类是应用程序的主执行线程(称为主进程)，其它独立的执行线程(称为辅进程)需从 CWinThread 派生。CWinApp 的基类也是 CWinThread 类。

3. 文档模板

文档模板定义了文档类、视图类和框架窗口类三者之间的关系。CSingleDocTemplate 类实现 SDI 文档模板，CMultiDocTemplate 类实现 MDI 文档模板。

4. 文档

文档类(从 CDocument 派生而来)指定了应用程序的数据。若应用程序支持 OLE 功能，则文档类可从 COleDocument 或其派生类中派生。

5. 视图

视图类(从 CView 派生而来)是用户的“数据窗口”。视图类指定了用户查看文档数据的方式和交互方式，如见到的文档可能是文本，也可能是图表或图像等。通常情况下，一个文档只有一个视图，但在有些情况下一个文档可能要有多个视图。

MFC 已定义了许多视图类，应用程序可以根据不同需要从不同的视图类派生。如果需

要滚动，则视图类可从 CScrollView 派生。如果视图中有一个基于对话框资源的用户接口，则视图类可从 CFormView 派生。对于简单的文本数据，可从 CEditView 派生。对于基于格式的数据访问应用程序，可从 CRecordView 派生。

6. 框架窗口

视图是显示在“文档框架窗口”中的。在 SDI 应用程序中，文档框架窗口也是该应用程序的“主框架窗口”。在 MDI 应用程序中，文档窗口是显示在主框架窗口中的子窗口，所派生的主框架窗口类指定了包含视图的框架窗口的格式及其它特性。

SDI 应用程序的文档框架窗口类的基类为 CFrameWnd 类。MDI 应用程序的主框架窗口类的基类为 CMDIFrameWnd 类，文档框架窗口类的基类为 CMDIChildWnd 类。

总之，应用程序的这些对象通过命令和消息相互联系，共同对用户动作进行响应。一个应用程序对象管理一个或多个文档模板；每个文档模板创建并管理一个或多个文档；用户通过包含在框架窗口中的一个视图观察和处理文档。

1.2　MFC 编程工具及其相互关系

Visual C++的 MFC 应用框架将编辑器、编译器、连接器、调试器、AppWizard、App Studio、ClassWizard 等工具集成在同一环境中，极大地方便了程序员开发 Windows 应用程序。其中以下三个工具尤为重要：

- AppWizard
- AppStudio
- ClassWizard

虽然在 MFC 应用程序编程时也可以不使用这些工具，但使用这些工具将大大地简化编程工作，提高编程速度。

1.2.1　AppWizard

AppWizard 是一个代码发生器，用于创建 MFC 应用程序。它按照用户通过对话框指定的特性、类名及源代码文件名来创建一个可启动的 Windows 应用程序框架。在此基础上程序员可完成自己应用程序的特殊功能。生成的应用程序项目文件包含应用程序、文档、视图和框架窗口等类实现文件；资源文件，包含标准菜单和可选择的工具条；其它必需的 Windows 文件；以及可选的含有 Windows Help 主题的.RTF 文件等。AppWizard 还可以包含对 OLE 和数据库的支持。具体创建过程以及生成的源文件解释详见第二章中的叙述。

在开发应用程序的进程中，必须通过修改菜单资源，加入处理函数等，使程序完全有血有肉。AppWizard“// TODO”注释表示在此处需要程序员加入实际操作代码。

1.2.2　AppStudio

AppStudio 是一个资源编辑器，用于创建用户界面和编辑程序资源，可用来创建包含 #define 常量在内的头文件。具体地讲，AppStudio 可以创建并编辑菜单、对话框、定制的控制框、加速键、位图、图标、光标、字符串和版本等资源。

通常该编辑器与 ClassWizard 一同使用，例如，创建了对话模板资源后，就可以利用

ClassWizard 创建新的对话框类，然后可以映射对话框中各命令按钮的消息处理函数以及增加成员变量。

AppWizard 建立含有基本资源(标准菜单、About 对话框、含有预定义字符串的字符串表等)的.RC 文件。其它文件包含缺省图标，版本资源。如果选择包括初始工具条的选项，则建立含有工具条按钮图像的位图。

在编程时常常需要建立新的资源和修改原有资源。对于菜单、工具条按钮和对话框等资源，在创建之后可以使用 ClassWizard 将它们链接到代码中。

另外，Visual C++ CD - ROM 盘\MSVC20\SAMPLES\MFC\CLIPART 目录下包含 COMMON.RC 资源文件，它含有一些"剪辑艺术"资源、工具条按钮、通用光标、图标等资源。程序员可以从中复制一些资源，再粘贴到自己的资源文件中，以节省开发时间。

1.2.3 ClassWizard

在编程过程中，最频繁使用的 MFC 工具就是 ClassWizard。程序员利用 ClassWizard 填写实施细节，选择把哪些消息映射给哪些对象，然后实现这些映射。即利用它管理类和 Windows 消息，将 Windows 消息及用户界面对象(例如菜单)与程序代码联系起来。

利用它可以往 AppWizard 所产生的工作框架中添加新的类或函数，也可以对现存的类进行修改，这就避免了一般代码发生器经常遇到的一致性问题。

另外还可以创建类成员变量、OLE Automation 的方法和属性、数据库类和覆盖 MFC 类中的虚函数等，以实现应用程序工作的细节。

1. 消息映射

ClassWizard 允许程序员浏览自己应用程序中所有与用户接口对象相关的消息，并且允许为这些消息快速地定义消息处理函数，同时 ClassWizard 会自动更新消息映射。用户接口对象的类型可分为三类，与消息类型一一对应，如表 1.1 所示。

表 1.1 对象类型与对应的消息类型

对象类型	消 息 类 型
窗口类	Windows 消息，如 WM_CREATE 消息
菜单或加速键标识符	COMMAND 消息(命令处理消息) UPDATE_COMMAND_UI 消息(更新命令用户接口消息)
控制标识符	选中的控制类型的通知消息，如 EN_CHANGE 消息

2. ClassWizard 注释格式

当用 ClassWizard 增加一个新类时，ClassWizard 会自动向程序代码中加入一些特殊格式的注释，用以标志头文件和实现文件中 ClassWizard 进行编辑的那些部分。ClassWizard 就是利用这些注释来维护类和消息的管理。ClassWizard 可分为五类注释，下面分别给予具体介绍。

(1) 消息映射注释

当利用 ClassWizard 向类添加消息处理函数后，ClassWizard 向类的头文件和实现文件中分别加入相关代码，并用下面的注释代码表示。大多数类都有 ClassWizard 可以编辑的相关

的两部分代码：类的头文件中成员函数的定义和类的实现文件中的消息映射条目。实际例子可参看第二章中的 MDI. H 和 MDI. CPP 文件清单。

头文件中 ClassWizard 的注释：

```
//{{AFX_MSG(<classname>)
    // NOTE — the ClassWizard will add and remove member functions here.
    //      DO NOT EDIT what you see in these blocks of generated code !
//}}AFX_MSG
```

实现文件中 ClassWizard 的注释：

```
//{{AFX_MSG_MAP(<classname>)
    // NOTE — the ClassWizard will add and remove mapping macros here.
    //      DO NOT EDIT what you see in these blocks of generated code !
//}}AFX_MSG_MAP
```

程序员自定义消息映射接口必须放在该注释段之外，而且还需要人工维护，ClassWizard 只维护注释段内的消息映射。

(2) 虚拟函数注释

当利用 ClassWizard 覆盖一个类中的虚拟函数后，ClassWizard 向类的头文件加入相关代码和注释。

头文件中 ClassWizard 的注释：

```
//{{AFX_VIRTUAL(<classname>)
    // 虚拟函数原型说明
//}}AFX_VIRTUAL
```

实现文件中 ClassWizard 没有特殊的注释。

(3) 数据映射注释

对于对话框、格式视图和记录视图，ClassWizard 有以下三种注释：

头文件中成员变量的说明：

```
//{{AFX_DATA
...
//}}AFX_DATA
```

实现文件中成员变量的初始化：

```
//{{AFX_DATA_INIT
...
//}}AFX_DATA_INIT
```

实现文件中的数据交换宏：

```
//{{AFX_DATA_MAP
...
//}}AFX_DATA_MAP
```

(4) 字段映射注释

对于字段交换，ClassWizard 有以下三种注释：

头文件中成员变量的说明：

```
//{{AFX_FIELD
...
//}}AFX_FIELD
```

头文件中成员变量的初始化：

```
//{{AFX_FIELD_INIT
...
//}}AFX_FIELD_INIT
```

实现文件中的记录交换函数的调用：

```
//{{AFX_FIELD_MAP
...
//}}AFX_FIELD_MAP
```

(5) OLE 分发映射注释

对于 OLE 方法分发，ClassWizard 有以下四种注释：

头文件中的 OLE 事件：

```
//{{AFX_EVENT
...
//}}AFX_EVENT
```

实现文件中的 OLE 事件：

```
//{{AFX_EVENT_MAP
...
//}}AFX_EVENT_MAP
```

头文件中 OLE Automation 的说明：

```
//{{AFX_DISP
...
//}}AFX_DISP
```

实现文件中 OLE Automation 的说明：

```
//{{AFX_DISP_MAP
...
//}}AFX_DISP_MAP
```

1.2.4 MFC 应用程序开发过程

使用 MFC 开发应用程序的一般过程为：

①用 AppWizard 创建初始应用程序的框架文件，这些文件中的类都是由类库中的类派生的，其中包含一个文档类、一个视窗类、一个框架窗口类和一个应用程序类。AppWizard 还创建一个初始的资源文件及其它的辅助文件。

②用 AppStudio 创建和编辑资源，从而创建用户界面对象，如菜单、对话框等。

③用 ClassWizard 将用户界面与消息处理函数联系起来。ClassWizard 为 App Studio 产生的每个用户接口对象产生对应的消息处理函数和消息映射。

④用 Visual C++编辑器加入消息处理函数代码。

⑤用 Project|Build 菜单命令编译连接，建立应用程序。

以上②～⑤步需多次重复，直至建立的应用程序达到设计目标和要求。

1.3　MFC 应用程序调试技术

除非特别小的应用程序，另外需要您非常仔细，否则不可能程序一次就建立成功，且符合要求。在设计过程中，程序发生错误是不可避免的，且程序越大、越复杂，错误发生的可能性也就越大。通常可分为两种错误：语法错误和执行错误。程序员可以很容易发现并改正程序语法错误，但不易发现和改正执行错误。因此本节主要讨论如何调试应用程序执行中的错误。Visual C++和 MFC 提供了多种方法供程序员调试应用程序，主要有下列几种。

1.3.1　Visual C++内置的调试器

与 DOS 程序调试器不同，不能使用调试器来从头到尾跟踪 Windows 程序。因为 Windows 程序是消息驱动的，而非顺序驱动，我们只能在需要跟踪的消息处理函数的入口设置断点。

Visual C++内置调试器与 Visual C++工作平台紧密配合，可以设置断点、控制单步执行，可以查看变量、修改变量、查看寄存器，可通过 Quick Watch 快速查看变量值。菜单命令 Step Into 可跟踪进入类库，Step Over 可以避免进入类库中的代码。

只有将程序设置为 DEBUG 模式(含有调试信息)，才能进行调试。

1.3.2　TRACE 宏

TRACE 宏为程序员提供与 DOS 环境下 printf 函数类似的功能，它将格式化字符串输出送到工作台的调试窗口(如图 1.2 所示)。TRACE 宏是在程序执行时跟踪变量的值的简便的方法。

TRACE 宏用法如下：

TRACE(exp)，其中 exp 指定可变数目的参数，这些参数与 printf 函数中使用的可变数目的参数的用法相同。

例如：

```
int iLen = 3;
char strBuf[] = "abc";
TRACE("Length = %d, String = %s\n", iLen, strBuf);
```

执行后，在 Out 窗口中输出：

```
Length = 3, String = abc
```

可以使用\MSVC20\BIN 目录下的 TRACER 应用程序设置 TRACE 宏的输出(如图 1.3

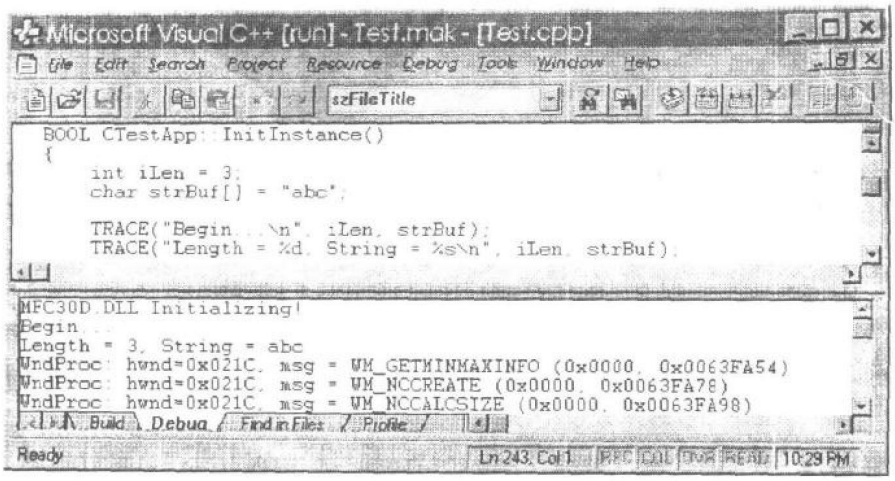

图 1.2　Visual C++调试窗口和 TRACE 输出

所示)，通过设置选项来控制输出窗口中的显示信息的类型，如设置 Enable Tracing 选项来激活框架的跟踪机制。

另外，类似功能的宏还有 TRACE0、TRACE1、TRACE2 和 TRACE3 宏。

图 1.3　通过 TRACER 应用程序设置选项

1.3.3　ASSERT 宏

ASSERT 宏为断言测试，用于检查函数作的假设，它在遇到错误条件时会引起程序终止。

通过设置适当的 ASSERT 参数，使它只在程序按预想方式工作时才为正确值。该宏指令判断 ASSERT 参数，如果参数表达式为假(0)，则通知用户并终止程序执行，反之若参数为真(非 0)，则不采取行动。

若 ASSERT 失败，则显示一个对话框，并在消息框中显示如下信息：

assertion failed in file <name> in line <num>

Abort Retry Ignore

其中<name>为源文件名，<num>为出错断言的行号。如果选择 Abort，则程序终止。如果选择 Ignore，则程序忽略该信息，继续执行。通过选择 Retry 钮有可能在 ASSERT 后进入调试器。Abort 和 Ignore 均不会激活调试器，因此它们无法提供检查查看栈的方法。

例如：

```
CWnd * pWnd;
... ...
ASSERT(pWnd ! = NULL);
```

如果 pWnd 为 NULL，则断言失败。

在应用程序的 Rlease 版本(不含调试信息)中，ASSERT 宏不判断该表达式，因而将不中断应用程序。如果不管环境怎样都必须判断该表达式，则应使用 VERIFY 宏代替 AS-

SERT 宏。

1.3.4　VERIFY 宏

VERIFY 宏类似于 ASSERT 宏。不同的是，在应用程序的 Rlease 版本中，VERIFY 仍要判断表达式的正确与否，但不检查结果，即不打印错误信息或中断程序。在应用程序的 Debug 版本中，VERIFY 宏同 ASSERT 宏功能一样。

VERIFY 宏用法如下：

VERIFY(*booleanException*)

其中参数 *booleanException* 指定其值将被判断为非 0 或 0 的一个表达式(包括指针值)。

在 Debug 版本中，VERIFY 宏判断其参数的值。若其值为 0，则 VERIFY 宏显示一个消息框，显示诊断消息 assertion failed in file <name> in line <num>，并终止程序。若条件为非 0，则该宏不采取任何动作。其中<name>为源文件名，<num>为源文件中断言出错的行号。

1.3.5　消息框

无论 Debug 版本，还是 Release 版本，都可以利用 AfxMessageBox 或 MessageBox 在需要输出信息的地方显示消息框，与用户交互。在消息框中可以显示各种值，适用于简单程序调试。有关这两个函数的具体用法请参见第三章中的叙述。

1.4　MFC 类库层次结构

为了使读者对于 Visual C++类库有一个整体了解，本节简要介绍 Microsoft 基本类库 2.0 的层次体系(如图 1.4 所示)。MFC2 包含六个子层次，分别管理 Windows 应用程序的不同方面。这些子层次为：

- 通用类
- 可视对象类
- 应用体系结构类
- 集合类
- OLE 2.0 类
- 数据库类

下面将具体描述各子层次。

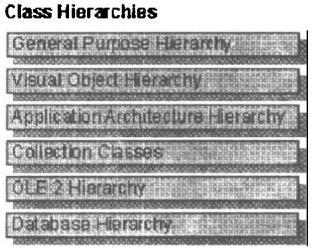

图 1.4　MFC2.0 层次体系

1.4.1　通用类

通用类(如图 1.5 所示)提供各种通用服务，例如文件 I/O、诊断和异常处理。

1. 基类

CObject 类是 MFC 大多数类的根类和基类，被称为“诸类之母”。CObject 类有很多有用的特性，包括对串行化的支持、运行时的类信息和对象诊断输出。

2. 文件类

CFile 类及其派生类提供文件服务。CFile 类提供无缓冲的、二进制磁盘输入/输出服