引 言

毫无疑问，编写 Windows 应用程序比编写 DOS 应用程序要复杂得多。当大家普遍关注 DOS 的时候，大多数程序员并没有遇到早期 Windows 编程员常常碰到的可怕经历(比如，最常见的是，在 Windows 程序中输出 Hello World 这句话往往需要大量语句行)。

Windows 3.0 和 Windows 3.1 的普遍推广使大量程序员接触到一种新的编程技术，即建立 Windows 应用程序。用于 Microsoft 编译器的 Microsoft SDK 的出现并不十分令人满意。值得庆幸的是，面向对象的 C++语言的出现，已使程序员有所解脱，而 Windows 界面的类层次体系(如 Microsoft 基本类(MFC)库)，又极大地提高了 C++语言的能力。

1. 本书的使用者

本书指导读者怎样利用 MFC 类库编写 Windows 应用程序。读者必须具备以下知识：

- 熟悉 C++语言
- 熟悉 Windows 的用法
- 初次编写 Windows 应用程序

本书的主要目的，是用实用方便的方法，向用户提供 MFC 库中涉及用户界面的各种成分。本书没有全部介绍 MFC 库中的类，虽然大家希望如此，但是时间不允许，而且也受频繁发表的软件新版本的影响。

2. 怎样使用本书

本书包括十一章，分别代表了用 MFC 库编程的不同方面。

第 1 章介绍了 MFC 类、Windows API 函数和消息管理机制。本章主要讲述了利用 MFC 库建立 Windows 应用程序的背景知识。

第 2 章以简单的 MFC 程序来说明其基本成分。本章还讨论了使用菜单和菜单资源、建立 MFC 应用程序的多个实例、安全地关闭窗口的方法。

第 3 章更为具体地研究了操纵应用程序窗口的方法。本章给出了更多关于注册窗口类、建立非缺省窗口的细节，同时还讨论了向窗口写文字和保存此文字的方法。

第 4 章是第 3 章的收尾，描述了滚动窗口，并解释了激活水平和垂直滚动条的机制。

下面三章提供了各种窗口控制。第 5 章讨论了静态文字、编辑框和按钮控制。

第 6 章描述了组框、复选框和无线按钮控制。这些控制可以使读者能最好地调节 Windows 应用程序的动作。

第 7 章讨论了滚动条、列表框和组合框控制。本章还分析了处理多重选择列表框、建立同步滚动的列表框和合成诸如历史列表之类的组合框的控制。

第 8 章讨论了模式和非模式对话框。本章分析了建立和执行不同的对话框、将对话框

用作应用窗口和传送数据的方法。本章还提供了一些例子，来说明数据在应用程序和对话框之间的传送。

第 9 章描述了标准的 Windows 对话框。这些对话框支持文件选择、颜色、字体选择、打印、打印机设置、文字搜索和替换。

第 10 章描述了多文档界面(MDI)窗口。本章考察了符合 MDI 特点的应用程序的不同成分，提供了两个样本程序。这些例子是典型的、强化的 MDI 窗口实例。此类窗口还包括命令按钮和复选框控制。

第 11 章描述了建立各种控制的资源描述语法。本章提供的信息是建立对话框所必需的。

第 1 章

Microsoft 基本类库的基础知识

Microsoft 基本类库(MFC)提供了开发 Windows 应用程序的强有力的工具,没有它,编写 Windows 应用程序的过程会变得更困难、更易失败,而且代码量相当大。Microsoft 基本类库成功地将面向对象和事件驱动的编程概念联系了起来,并证明了这两种方法可以很好地配合使用。本章介绍了 Microsoft 基本类库和与 Windows 有关的一些基础知识,包括:

- 常用的 Windows 数据类型
- Microsoft 基本类库数据类型的划分约定
- Microsoft 基本类的层次体系
- Windows API 函数
- 如何响应 Windows 消息
- 怎样发送消息
- 用户自定义的消息

本章的信息提供了利用 Microsoft 基本类库建立 Windows 应用程序所需的各方面的知识。尽管 Microsoft 基本类库是开发 Windows 应用程序的良好工具,熟悉 Windows API 函数还是很必要的。

1.1 常用的 Windows 数据类型

Windows 编程包含了许多数据类型,本节重点介绍初次进行 Windows 编程时最可能涉及的数据类型,熟悉了它们有助于更好地理解 MFC 的类声明。Windows 数据类型包括一些简单类型和一组结构。简单数据类型包括许多 typedef 型的数据,它们声明的是含义更明显的别名,这使得函数和变量的声明变得很清楚。表 1.1 列出了最常用的 Windows 数据类型。

除了简单的数据类型(如表 1.1 中所列)之外,Windows 编程中还涉及了一些其他结构。目前,只需了解两个简单但十分重要的结构:POINT 和 RECT。

POINT 结构用于存储一个点的 x 和 y 坐标,声明如下:

```
typedef struct tagPOINT{
        int x;
        int y;
} POINT;
```

RECT 结构定义了矩形的左上角和右下角坐标,其声明如下:

```
typedef struct tagRECT{
        int left;
        int top;
```

表 1.1 最常用的 Windows 数据类型

数据类型	意　义	数据类型	意　义
char	有符号的 8 位字符	FARPROC	指向函数的长型指针
int	有符号的 16 位整数	HANDLE	通用柄
long	有符号的 32 位整数	HDC	显示描述表标识柄
short	有符号的 16 位整数	HWND	窗口柄
void	空值	LONG	与 long 类型相同
BOOL	代表逻辑值的 16 位整数	LPSTR	指向字符串的长型指针
BYTE	无符号的 8 位整数	LPVOID	指向 void 类型的长型指针
UINT	无符号的 32 位整数	PWORD	指向 WORD 类型的指针
DWORD	无符号的 32 位整数或段/偏移量地址	WORD	无符号的 16 位整数
FAR	建立远指针的数据类型属性		

```
        int right;
        int bottom;
} RECT;
```

其中,left 和 right 成员是矩形左上角和右下角的 x 坐标。top 和 bottom 域分别为矩形左上角和右下角的 y 坐标。

1.2 Microsoft 基本类的层次体系

Microsoft 基本类的层次体系包含几个子层次,它们分别管理 Windows 应用程序的不同方面。这些子层次为:

- 窗口类
- GDI 类
- 杂项窗口类(即与 Windows 应用程序的可视界面无关,可用于 DOS 应用程序的那些类。)
- 对象链接和嵌入(OLE)类
- 文件类
- 对象 I/O 类
- 异常情况类
- 收集类
- 其它支持类

图 1.1 为 Microsoft 基本类层次体系中的窗口类和杂项窗口类。此图还包括了类 CArchive,它是 Microsoft 基本类库支持的两个对象 I/O 类之一。本书重点描述图 1.1 中的类,并用其他 MFC 类,如 CString 类,来支持例子程序。

下面将描述图 1.1 中的各个类。描述包括类声明和各个类主要功能的简要描述。

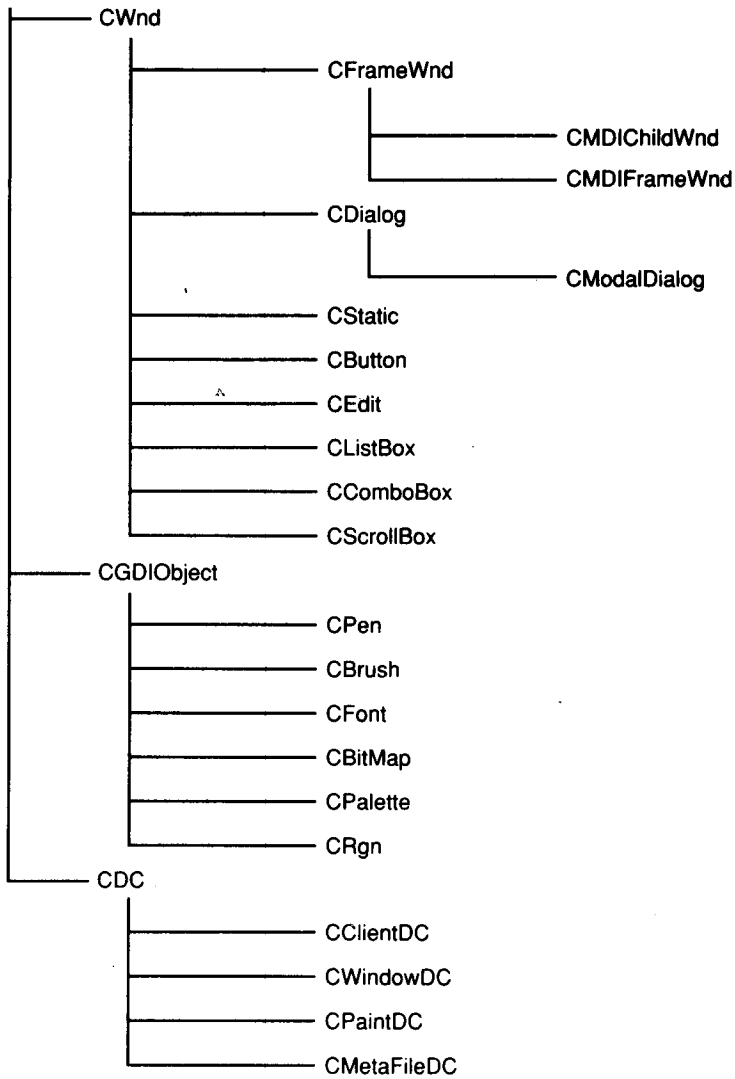

图 1.1 Microsoft 基本类层次体系中的窗口类和杂项窗口类

1.2.1 类 CObject

类 CObject 是 Microsoft 基本类库的抽象基类。其类声明为：

```
class CObject
{
public:

// Object model (types, destruction, allocation)
     virtual CRuntimeClass* GetRuntimeClass() const;
     virtual ~CObject();  // virtual destructors are necessary
```

```
    // Diagnostic allocations
    void* operator new(size_t, void* p);
    void* operator new(size_t nSize);
    void operator delete(void* p);

#ifdef _DEBUG
    // for file name/line number tracking using DEBUG_NEW
    void* operator new(size_t nSize, const char FAR* lpszFileName,
                       int nLine);
#endif

    // Disable the copy constructor and assignment by default
    // so you get compiler errors instead of unexpected
    // behavior if you pass objects by value or assign objects.
protected:
    CObject();
private:
    CObject(const CObject& objectSrc);
    void operator=(const CObject& objectSrc);

// Attributes
public:
    BOOL IsSerializable() const;
// Overridables
    virtual void Serialize(CArchive& ar);
    // Diagnostic Support
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;

// Implementation
public:
    // dynamic type checking/construction support
    BOOL IsKindOf(const CRuntimeClass* pClass) const;
    static void PASCAL Construct(void* pMemory);

    static CRuntimeClass classCObject;
};
```

类 CObject 是 Microsoft 基本类库中各类及其各种可能的派生类的根类。CObject 类并不是一个抽象的空类。相反，它向其所有派生类提供了大量操作。这些操作包括：

- 创建和删除对象
- 串行化的支持
- 诊断
- 与收集类的兼容性

类 CObject 声明了 new 操作符的三个版本和一个 delete 操作符。

Microsoft 基本类库还声明了一些支持 CObject 类及其所有派生类的宏。这些宏为

- 宏 RUNTIME_CLASS，返回与所命名的类匹配的 CRunTimeClass 结构。

- 宏 DECLARE_DYNAMIC，使程序能够存取运行时的类数据。该宏可用于各个类声明中。
- 宏 IMPLEMENT_DYNAMIC，使程序能够存取运行时的类数据。该宏用于类的实现部分。
- 宏 DECLARE_SERIAL，使得在 Windows 应用程序中可以串行化并存取运行时类数据。该宏可用于各个类声明中。
- 宏 IMPLEMENT_SERIAL，使得在 Windows 应用程序中可以串行化并存取类数据。该宏用于类的实现部分。

1.2.2 类 CWinApp

类 CWinApp 创建 Windows 应用程序。用户编写的应用程序必须恰好含有类 CWinApp 的一个派生类。应用程序中类 CWinApp 派生类的实例拥有其他所有对象。类 CWinApp 的声明如下所示：

```
class CWinApp : public CObject
{
    DECLARE_DYNAMIC(CWinApp)
public:

// Constructor
    CWinApp(const char* pszAppName = NULL);
    void SetCurrentHandles();

// Attributes
    // Startup args (do not change)
    const char* m_pszAppName;       // from constructor
    HANDLE m_hInstance;
    HANDLE m_hPrevInstance;
    LPSTR m_lpCmdLine;
    int m_nCmdShow;

    // Running args
    CWnd* m_pMainWnd;       // main window (optional)

// Operations
    // Cursors
    HCURSOR LoadCursor(LPSTR lpCursorName);
    HCURSOR LoadCursor(UINT nIDCursor);
    // for IDC_ values
    HCURSOR LoadStandardCursor(LPSTR lpCursorName);
    // for IDC_ values
    HCURSOR LoadOEMCursor(UINT nIDCursor); // for OCR_ values

    // Icons
    HICON LoadIcon(LPSTR lpIconName);
    HICON LoadIcon(UINT nIDIcon);
    HICON LoadStandardIcon(LPSTR lpIconName); // for IDI_ values
    HICON LoadOEMIcon(UINT nIDIcon); // for OIC_ values
```

```
    BOOL PumpMessage();

// Overridables
    // hooks for your initialization code
    virtual BOOL InitApplication();
    virtual BOOL InitInstance();

    virtual int Run();

    // called by standard 'Run' implementation
    virtual BOOL PreTranslateMessage(MSG* pMsg);
    virtual BOOL OnIdle(LONG lCount); // return TRUE if
                                      // more idle processing
    virtual int ExitInstance(); // return app exit code

// Implementation
#ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif

protected:
    MSG  m_msgCur;

#ifdef _DEBUG
// Diagnostic trap for when going back to
// message pump is not permitted.
protected:
    int m_nDisablePumpCount;
public:
    void EnablePump(BOOL bEnable);
#endif

};
```

类 CWinApp 含有大量公用数据成员，这些数据成员存储 Windows 传递给预定义的 WinMain()函数的数据(WinMain()函数的作用类似于 C 程序中的 main()函数)。下面列出这些数据成员的功能：

- m_hInstance 成员，存储应用程序当前实例的标识柄。
- m_hPrevInstance 成员，存储上一个实例的标识柄。如果当前实例是目前唯一运行着的实例，m_hPrevInstance 具有值 NULL。
- m_lpCmdLine 成员，存储调用 Windows 应用程序时，用户可能键入的命令行参数。
- m_nCmdShow 成员，存储 Windows 应用程序的状态：正常、最小化、最大化或隐藏。

其他公用数据成员为：

- m_pMainWnd 成员，指向主程序窗口的指针。

• m_pszAppName 成员，存储类构造函数提供的应用程序名称。

类 CWinApp 只有一个构造函数，该函数以可选的应用程序名称作为参数。类的功能由两组成员函数支持。第一组函数提供基本操作，例如装入光标、图符、执行正常的初始化操作。类 CWinApp 成员函数提供装入自订的、预定义的，或 OEM 光标的功能，装入图符的功能与此类似。

> **注释**：类 CWinApp 声明的许多成员函数可能会在用户建立的应用程序中被覆盖掉。对这些函数的覆盖应根据实际情况的不同而变化。例如，为了满足 Windows 程序中对窗口初始化的要求，用户常常必须在 CWinApp 派生类中覆盖 InitInstance()函数。另一方面，用户也覆盖了 InitApplication()函数以执行一次性的、应用程序级的初始化。

成员函数 Run()执行缺省消息循环，以管理输入消息。只有在对输入消息的管理方式需要更多控制时，才覆盖 Run()函数。

成员函数 OnIdle()使得在应用程序空闲时，执行后台任务变得可能。这些任务可能为数据排序或备份。

成员函数 ExitInstance()在终止 Windows 程序时执行清除任务。要想加进自定义的清除任务，例如删除辅助文件，就必须覆盖该函数。

成员函数 PreTranslateMessage()在消息到达 Windows API 函数 TranslateMessage()和 DispatchMessage()前将其截住。要想执行特殊的消息处理过程，必须覆盖此函数。

1.2.3 类 CMenu

类 CMenu 是类 CObject 的派生类，用于管理菜单。菜单管理包括创建、更新、跟踪和删除与窗口相关的菜单资源。类 CMenu 的声明如下：

```
class CMenu : public CObject
{
    DECLARE_DYNAMIC(CMenu)
public:

// Constructors
    CMenu();

    BOOL CreateMenu();
    BOOL CreatePopupMenu();
    BOOL LoadMenu(const char FAR* lpMenuName);
    BOOL LoadMenu(UINT nIDMenu);
    BOOL LoadMenuIndirect(const BYTE FAR* lpMenuTemplate);
    BOOL DestroyMenu();
```

```
// Attributes
    HMENU m_hMenu;
    HMENU GetSafeHmenu() const;

    static CMenu* FromHandle(HMENU hMenu);
    static void DeleteTempMap();
    BOOL Attach(HMENU hMenu);
    HMENU Detach();

// CMenu Operations
    BOOL DeleteMenu(UINT nPosition, UINT nFlags);
    BOOL TrackPopupMenu(UINT nFlags, int x, int y,
                        const CWnd* pWnd,
                        LPRECT lpRectReserved = 0);

// CMenuItem Operations
    BOOL AppendMenu(UINT nFlags, UINT nIDNewItem = 0,
                    const char FAR* lpNewItem = NULL);
    BOOL AppendMenu(UINT nFlags, UINT nIDNewItem,
                    const CBitmap* pBmp);
    UINT CheckMenuItem(UINT nIDCheckItem, UINT nCheck);
    UINT EnableMenuItem(UINT nIDEnableItem, UINT nEnable);
    UINT GetMenuItemCount() const;
    UINT GetMenuItemID(int nPos) const;

    UINT GetMenuState(UINT nID, UINT nFlags) const;
    int GetMenuString(UINT nIDItem, LPSTR lpString, int
                      nMaxCount, UINT nFlags) const;
    CMenu* GetSubMenu(int nPos) const;
    BOOL InsertMenu(UINT nPosition, UINT nFlags,
                    UINT nIDNewItem = 0,
                    const char FAR* lpNewItem = NULL);
    BOOL InsertMenu(UINT nPosition, UINT nFlags, UINT
                    nIDNewItem, const CBitmap* pBmp);
    BOOL ModifyMenu(UINT nPosition, UINT nFlags,
                    UINT nIDNewItem = 0,
                    const char FAR* lpNewItem = NULL);
    BOOL ModifyMenu(UINT nPosition, UINT nFlags, UINT
                    nIDNewItem, const CBitmap* pBmp);
    BOOL RemoveMenu(UINT nPosition, UINT nFlags);
    BOOL SetMenuItemBitmaps(UINT nPosition, UINT nFlags,
                    const CBitmap* pBmpUnchecked,
                    const CBitmap* pBmpChecked);

// Implementation
#ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif

    virtual ~CMenu();
};
```

类 CMenu 只声明了一个数据成员 m_hMenu，用来保存菜单的 HMENU 柄。该类声明了一个构造函数和一个析构函数。CMenu 类的构造函数只创建一个类实例并将 NULL 值赋给 m_hMenu 数据。类 CMenu 的成员函数集很丰富，具有创建、更新、查询和删除菜单及菜单项的功能。

菜单建立函数可根据选项以多种方式建立菜单，可以：

- 从可执行文件装入菜单资源。
- 从内存中已有的菜单模板装入菜单柄。
- 建立空菜单或弹出式菜单。在此情况下可以动态地增加或减少菜单项。

类 CMenu 的成员函数使得用户可以在与窗口相关的菜单中设置、查询、激活、屏蔽和跟踪不同的菜单项，以及弹出菜单项。

1.2.4 类 CWnd

类 CWnd 是类 CObject 的派生类，也是 Microsoft 基本类库中最重要的类。其声明为：

```
class CWnd : public CObject
{
     DECLARE_DYNAMIC(CWnd)
protected:
     static const MSG* GetCurrentMessage();

// Attributes
public:
     HWND m_hWnd;
     HWND GetSafeHwnd() const;

     static CWnd* FromHandle(HWND hWnd);
     static CWnd* FromHandlePermanent(HWND hWnd); // INTERNAL USE
     static void DeleteTempMap();
     BOOL Attach(HWND hWndNew);
     HWND Detach();
     DWORD GetStyle() const;

// Constructors
     CWnd();

protected: // This CreateEx() wraps CreateWindowEx
     BOOL CreateEx(DWORD dwExStyle, const char FAR*
          lpClassName, const char FAR* lpWindowName, DWORD
          dwStyle,
          int x, int y, int nWidth, int nHeight,
          HWND hwndParent, HMENU nIDorHMenu);

private:
     CWnd(HWND hWnd);     // just for special initialization

public:
     // for child windows...
```

```
    BOOL Create(const char FAR* lpClassName,
        const char FAR* lpWindowName, DWORD dwStyle,
        const RECT& rect,
        const CWnd* pParentWnd, UINT nID);

    virtual BOOL DestroyWindow();

// Message Functions
    LONG SendMessage(UINT message, UINT wParam = 0,
                                   LONG lParam = 0);
    BOOL PostMessage(UINT message, UINT wParam = 0,
                                   LONG lParam = 0);

// Window Text Functions
    void SetWindowText(const char FAR* lpString);
    int GetWindowText(LPSTR lpString, int nMaxCount) const;
    int GetWindowTextLength() const;
    void GetWindowText(CString& rString) const;
    void SetFont(CFont* pFont, BOOL bRedraw = TRUE);
    CFont* GetFont();

// CMenu Functions - non-Child windows only
    CMenu* GetMenu() const;
    BOOL SetMenu(CMenu* pMenu);
    void DrawMenuBar();
    CMenu* GetSystemMenu(BOOL bRevert) const;
    BOOL HiliteMenuItem(CMenu* pMenu, UINT nIDHiliteItem,
                        UINT nHilite);

// Special attributes for Child windows only
    int GetDlgCtrlID() const;

// Window Size and Position Functions void CloseWindow();
    BOOL OpenIcon();
    BOOL IsIconic() const;
    BOOL IsZoomed() const;
    void MoveWindow(int x, int y, int nWidth, int nHeight,
                    BOOL bRepaint = TRUE);
    void MoveWindow(LPRECT lpRect, BOOL bRepaint = TRUE);
    // For SetWindowPos's pWndInsertAfter
    static const CWnd NEAR wndTop;
    // For SetWindowPos's pWndInsertAfter
    static const CWnd NEAR wndBottom;
    void SetWindowPos(const CWnd* pWndInsertAfter, int x,
                      int y, int cx, int cy, UINT nFlags);
    UINT ArrangeIconicWindows();
    void BringWindowToTop();
    void GetWindowRect(LPRECT lpRect) const;
    void GetClientRect(LPRECT lpRect) const;

// Coordinate Mapping Functions
    void ClientToScreen(LPPOINT lpPoint) const;
```

```
    void ClientToScreen(LPRECT lpRect) const;
    void ScreenToClient(LPPOINT lpPoint) const;
    void ScreenToClient(LPRECT lpRect) const;

// Update/Painting Functions
    CDC* BeginPaint(LPPAINTSTRUCT lpPaint);
    void EndPaint(LPPAINTSTRUCT lpPaint); CDC* GetDC();
    CDC* GetWindowDC();
    int ReleaseDC(CDC* pDC);

    void UpdateWindow();
    void SetRedraw(BOOL bRedraw = TRUE);
    BOOL GetUpdateRect(LPRECT lpRect, BOOL bErase = FALSE);
    int GetUpdateRgn(CRgn* pRgn, BOOL bErase = FALSE);
    void Invalidate(BOOL bErase = FALSE);
    void InvalidateRect(LPRECT lpRect, BOOL bErase = FALSE);
    void InvalidateRgn(CRgn* pRgn, BOOL bErase = FALSE);
    void ValidateRect(LPRECT lpRect);
    void ValidateRgn(CRgn* pRgn);
    BOOL ShowWindow(int nCmdShow);
    BOOL IsWindowVisible() const;
    void ShowOwnedPopups(BOOL bShow = TRUE);

// Timer Functions
    UINT SetTimer(int nIDEvent, UINT nElapse,
        UINT (FAR PASCAL EXPORT* lpfnTimer)(HWND, UINT, int, DWORD))
    BOOL KillTimer(int nIDEvent);

// Window State Functions
    BOOL IsWindowEnabled() const;
    BOOL EnableWindow(BOOL bEnable = TRUE);

    static CWnd* GetActiveWindow();
    CWnd* SetActiveWindow();

    static CWnd* GetCapture();
    CWnd* SetCapture();
    static CWnd* GetFocus();
    CWnd* SetFocus();

    CWnd* SetSysModalWindow();
    static CWnd* GetSysModalWindow();

    static CWnd* GetDesktopWindow();
// Dialog-Box Item Functions
// (NOTE: Dialog-box Items are not necessarily in dialog boxes!)
    void CheckDlgButton(int nIDButton, UINT nCheck);
    void CheckRadioButton(int nIDFirstButton, int nIDLastButton,
                          int nIDCheckButton);
    int GetCheckedRadioButton(int nIDFirstButton,
                              int nIDLastButton);
    int DlgDirList(const char FAR* lpPathSpec, int nIDListBox,
                   int nIDStaticPath, UINT nFileType);
```

```
    int DlgDirListComboBox(const char FAR* lpPathSpec,
                           int nIDComboBox,
                           int nIDStaticPath, UINT nFileType);
    BOOL DlgDirSelect(LPSTR lpString, int nIDListBox);
    BOOL DlgDirSelectComboBox(LPSTR lpString, int nIDComboBox);

    CWnd* GetDlgItem(int nID) const;
    UINT GetDlgItemInt(int nID, BOOL* lpTrans = NULL,
                       BOOL bSigned = TRUE) const;
    int GetDlgItemText(int nID, LPSTR lpStr, int nMaxCount) const;

    CWnd* GetNextDlgGroupItem(CWnd* pWndCtl,
                              BOOL bPrevious = FALSE) const;

    CWnd* GetNextDlgTabItem(CWnd* pWndCtl,
                            BOOL bPrevious = FALSE) const;
    UINT IsDlgButtonChecked(int nIDButton) const;
    LONG SendDlgItemMessage(int nID, UINT message,
                            UINT wParam = 0, LONG lParam = 0);
    void SetDlgItemInt(int nID, UINT nValue, BOOL bSigned = TRUE);
    void SetDlgItemText(int nID, const char FAR* lpString);

// Scrolling Functions
    int GetScrollPos(int nBar) const;
    void GetScrollRange(int nBar, LPINT lpMinPos, LPINT lpMaxPos)
const;
    void ScrollWindow(int xAmount, int yAmount,
                      LPRECT lpRect = NULL,
                      LPRECT lpClipRect = NULL);
    int SetScrollPos(int nBar, int nPos, BOOL bRedraw = TRUE);
    void SetScrollRange(int nBar, int nMinPos, int nMaxPos,
                        BOOL bRedraw = TRUE);

    void ShowScrollBar(UINT nBar, BOOL bShow = TRUE);

// Window Access Functions
    CWnd* ChildWindowFromPoint(POINT point) const;

    static CWnd* FindWindow(const char FAR* lpClassName,
                            const char FAR* lpWindowName);
    CWnd* GetNextWindow(UINT nFlag = GW_HWNDNEXT) const;
    CWnd* GetTopWindow() const;

    CWnd* GetWindow(UINT nCmd) const;
    CWnd* GetLastActivePopup() const;

    BOOL IsChild(CWnd* pWnd) const;
    CWnd* GetParent() const;
    CWnd* SetParent(CWnd* pWndNewParent);
    static CWnd* WindowFromPoint(POINT point);

// Alert Functions
    BOOL FlashWindow(BOOL bInvert);
```

```
    int MessageBox(const char FAR* lpText,
                   const char FAR* lpCaption = NULL,
                   UINT nType = MB_OK);

// Clipboard Functions
    BOOL ChangeClipboardChain(HWND hWndNext);
    HWND SetClipboardViewer();
    BOOL OpenClipboard();
    static CWnd* GetClipboardOwner();
    static CWnd* GetClipboardViewer();

// Caret Functions
    void CreateCaret(CBitmap* pBitmap);
    void CreateSolidCaret(int nWidth, int nHeight);
    void CreateGrayCaret(int nWidth, int nHeight);
    static CPoint GetCaretPos();
    static void SetCaretPos(POINT point);
    void HideCaret();
    void ShowCaret();

// Window-Management message handler member functions protected:
    virtual BOOL OnCommand(UINT wParam, LONG lParam);

    afx_msg void OnActivate(UINT nState, CWnd* pWndOther,
                            BOOL bMinimized);
    afx_msg void OnActivateApp(BOOL bActive, HANDLE hTask);
    afx_msg void OnCancelMode();
    afx_msg void OnChildActivate();
    afx_msg void OnClose();
    afx_msg int OnCreate(LPCREATESTRUCT lpCreateStruct);

    afx_msg HBRUSH OnCtlColor(CDC* pDC, CWnd* pWnd,
                              UINT nCtlColor);
    afx_msg void OnDestroy();
    afx_msg void OnEnable(BOOL bEnable);
    afx_msg void OnEndSession(BOOL bEnding);
    afx_msg void OnEnterIdle(UINT nWhy, CWnd* pWho);
    afx_msg BOOL OnEraseBkgnd(CDC* pDC);
    afx_msg void OnGetMinMaxInfo(LPPOINT lpPoints);
    afx_msg void OnIconEraseBkgnd(CDC* pDC);
    afx_msg void OnKillFocus(CWnd* pNewWnd);
    afx_msg LONG OnMenuChar(UINT nChar, UINT nFlags, CMenu* pMenu);
    afx_msg void OnMenuSelect(UINT nItemID, UINT nFlags,
                              HMENU hSysMenu);
    afx_msg void OnMove(int x, int y);
    afx_msg void OnPaint();
    afx_msg void OnPaintIcon();
    afx_msg void OnParentNotify(UINT message, LONG lParam);
    afx_msg HCURSOR OnQueryDragIcon();
    afx_msg BOOL OnQueryEndSession();
    afx_msg BOOL OnQueryNewPalette();
    afx_msg BOOL OnQueryOpen();
    afx_msg void OnSetFocus(CWnd* pOldWnd);
```

```
    afx_msg void OnShowWindow(BOOL bShow, UINT nStatus);
    afx_msg void OnSize(UINT nType, int cx, int cy);

// Nonclient-Area message handler member functions
    afx_msg BOOL OnNcActivate(BOOL bActive);
    afx_msg void OnNcCalcSize(LPRECT lpRect);
    afx_msg BOOL OnNcCreate(LPCREATESTRUCT lpCreateStruct);
    afx_msg void OnNcDestroy();
    afx_msg UINT OnNcHitTest(CPoint point);
    afx_msg void OnNcLButtonDblClk(UINT nHitTest, CPoint point);
    afx_msg void OnNcLButtonDown(UINT nHitTest, CPoint point);
    afx_msg void OnNcLButtonUp(UINT nHitTest, CPoint point);
    afx_msg void OnNcMButtonDblClk(UINT nHitTest, CPoint point);
    afx_msg void OnNcMButtonDown(UINT nHitTest, CPoint point);
    afx_msg void OnNcMButtonUp(UINT nHitTest, CPoint point);
    afx_msg void OnNcMouseMove(UINT nHitTest, CPoint point);
    afx_msg void OnNcPaint();
    afx_msg void OnNcRButtonDblClk(UINT nHitTest, CPoint point);
    afx_msg void OnNcRButtonDown(UINT nHitTest, CPoint point);
    afx_msg void OnNcRButtonUp(UINT nHitTest, CPoint point);

// System message handler member functions
    afx_msg void OnSysChar(UINT nChar, UINT nRepCnt, UINT nFlags);
    afx_msg void OnSysCommand(UINT nID, LONG lParam);
    afx_msg void OnSysDeadChar(UINT nChar, UINT nRepCnt,
                               UINT nFlags);
    afx_msg void OnSysKeyDown(UINT nChar, UINT nRepCnt,
                              UINT nFlags);
    afx_msg void OnSysKeyUp(UINT nChar, UINT nRepCnt,
                            UINT nFlags);
    afx_msg void OnCompacting(UINT nCpuTime);
    afx_msg void OnDevModeChange(LPSTR lpDeviceName);
    afx_msg void OnFontChange();
    afx_msg void OnPaletteChanged(CWnd* pFocusWnd);
    afx_msg void OnSpoolerStatus(UINT nStatus, UINT nJobs);
    afx_msg void OnSysColorChange();
    afx_msg void OnTimeChange();
    afx_msg void OnWinIniChange(LPSTR lpSection);

// Input message handler member functions
    afx_msg void OnChar(UINT nChar, UINT nRepCnt, UINT nFlags);
    afx_msg void OnDeadChar(UINT nChar, UINT nRepCnt,
                            UINT nFlags);
    afx_msg void OnHScroll(UINT nSBCode, UINT nPos,
                           CWnd* pScrollBar);
    afx_msg void OnKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags);
    afx_msg void OnKeyUp(UINT nChar, UINT nRepCnt, UINT nFlags);
    afx_msg void OnLButtonDblClk(UINT nFlags, CPoint point);
    afx_msg void OnLButtonDown(UINT nFlags, CPoint point);
    afx_msg void OnLButtonUp(UINT nFlags, CPoint point);
    afx_msg void OnMButtonDblClk(UINT nFlags, CPoint point);
    afx_msg void OnMButtonDown(UINT nFlags, CPoint point);
    afx_msg void OnMButtonUp(UINT nFlags, CPoint point);
```

```
    afx_msg int OnMouseActivate(CWnd* pFrameWnd, UINT nHitTest,
                                UINT message);
    afx_msg void OnMouseMove(UINT nFlags, CPoint point);
    afx_msg void OnRButtonDblClk(UINT nFlags, CPoint point);
    afx_msg void OnRButtonDown(UINT nFlags, CPoint point);
    afx_msg void OnRButtonUp(UINT nFlags, CPoint point);
    afx_msg BOOL OnSetCursor(CWnd* pWnd, UINT nHitTest, UINT message);
    afx_msg void OnTimer(UINT nIDEvent);
    afx_msg void OnVScroll(UINT nSBCode, UINT nPos,
                           CWnd* pScrollBar);

// Initialization message handler member functions
    afx_msg void OnInitMenu(CMenu* pMenu);
    afx_msg void OnInitMenuPopup(CMenu* pPopupMenu, UINT nIndex,
                                 BOOL bSysMenu);

// Clipboard message handler member functions
    afx_msg void OnAskCbFormatName(UINT nMaxCount,
                                   LPSTR lpString);
    afx_msg void OnChangeCbChain(HWND hWndRemove, HWND hWndAfter);
    afx_msg void OnDestroyClipboard();
    afx_msg void OnDrawClipboard();
    afx_msg void OnHScrollClipboard(CWnd* pClipAppWnd,
                                    UINT nSBCode, UINT nPos);
    afx_msg void OnPaintClipboard(CWnd* pClipAppWnd, HANDLE hPaintStruct);
    afx_msg void OnRenderAllFormats();
    afx_msg void OnRenderFormat(UINT nFormat);
    afx_msg void OnSizeClipboard(CWnd* pClipAppWnd, HANDLE hRect);
    afx_msg void OnVScrollClipboard(CWnd* pClipAppWnd,
                                    UINT nSBCode, UINT nPos);

// Control message handler member functions
    afx_msg int OnCharToItem(UINT nChar, CWnd* pListBox,
                             UINT nIndex);
    afx_msg int OnCompareItem(
                    LPCOMPAREITEMSTRUCT lpCompareItemStruct);
    afx_msg void OnDeleteItem(
                    LPDELETEITEMSTRUCT  lpDeleteItemStruct);
    afx_msg void OnDrawItem(LPDRAWITEMSTRUCT lpDrawItemStruct);
    afx_msg UINT OnGetDlgCode();
    afx_msg void OnMeasureItem(
                    LPMEASUREITEMSTRUCT lpMeasureItemStruct);

    afx_msg int OnVKeyToItem(UINT nKey, CWnd* pListBox, UINT nIndex);
// MDI message handler member functions
    afx_msg void OnMDIActivate(BOOL bActivate,
                               CWnd* pActivateWnd,
                               CWnd* pDeactivateWnd);

// Overridables and other helpers (for implementation of derived classes)
protected:
    // for deriving from a standard control
    virtual FARPROC* GetSuperWndProcAddr();
```