罗强 蔡兵兵 编著

循序渐进学 Visual C++.NET 编程

- 从入门与实用的角度出发讲解，引导读者及时地掌握Visual C++.NET的新技术，并配有大量的精选实例
- 条理化的结构安排，让读者快速查找解决各种问题的方法
- 提供最新的编程思想，能大幅度缩短您摸索的时间

北京科海电子出版社
http://www.KHP.com.cn

科海电脑技术丛书

循序渐进学 Visual C++.NET 编程

罗强　蔡兵兵 编著

北京科海电子出版社

http://www.khp.com.cn

内 容 提 要

读者对象：

- ☑ 希望快速掌握 Visual C++ .NET 的初学者
- ☑ 适用于从初级到高级水平的读者朋友

达成目标：

- ☑ 学会使用 Visual C++.NET 设计程序
- ☑ 设计出属于您的网络程序

技能要求：

- ☑ 对 C 语言有深入了解
- ☑ 接触过 C++编程

本书特点：

- ☑ 从入门与实用角度出发讲解
- ☑ 条理化的结构安排，让读者快速查找解决各种问题的方法
- ☑ 提供最新的编程思想，能大幅度缩短您摸索的时间

这是一本有关 Visual C++.NET 和 Microsoft.NET 的书籍。书中集成了大量 Visual C++.NET 的新知识、新技术，倾注了笔者多年来的编程经验。

.NET 的平台及框架是基于微软软件工业基础的又一次升级和演化，在网络飞速发展的今天，.NET 将很快成为主流。

本书分为 5 大部分，共 16 章，内容包括：Visual C++.NET 概述、Visual C++.NET 的开发环境、面向对象编程与 C++语言、MFC 及 Framework、文档视图结构、图形设备接口 GDI+、公用控制、动态链接库、多线程编程技术、Internet 编程、.NET 编程等技术，每个技术都配有精选的实例和大量图片。本书由浅入深，简单易学，能快速引导您顺利进入.NET 的编程世界。

本书可作为高等院校师生教学和自学参考用书，也可作为各类计算机培训班的教材。

品　　名：循序渐进学 Visual C++.NET 编程
作　　者：罗　强　蔡兵兵
责任编辑：徐建军
排　　版：吴文娟
出　　品：北京科海电子出版社
印 刷 者：北京朝阳科普印刷厂印刷
发　　行：新华书店总店北京科技发行所
开　　本：787×1092 1/16　印张：33.375 字数：811 千字
版　　次：2002 年 8 月第 1 版　2002 年 8 月第 1 次印刷
印　　数：0001~5000
盘　　号：ISBN 7-900107-37-1
定　　价：48.00 元（1CD）

目　录

第 1 部分　Visual C++.NET 简介

第 2 部分　Visual C++语言基础

第 3 部分 Visual C++编程初步

第 4 部分 基本控件

第 5 部分 高级编程

第 1 部分　Visual C++.NET 简介

第 1 章　Visual C++.NET 概述

MFC 类库是 Visual C++的基础与核心部分，可以说学习使用 Visual C++的过程就是学习使用 MFC 类库的过程，MFC 类库庞大而繁杂由数百个类组成，类与类之间的层次关系也颇为复杂，后面将有专门的章节讲述 MFC 类库的基本结构，本章主要介绍其发展史及其基本概念，此外还将简要介绍 Visual Studio.NET 集成开发环境的一些新特性与功能。

1.1　MFC 历史和 Visual C++

微软基础类库（MFC，Microsoft Foundation Class）是微软为 Windows 程序员提供的一个面向对象的 Windows 编程接口，它大大简化了 Windows 的编程工作。使用 MFC 类库的好处是：首先，MFC 提供了一个标准化的结构，这样开发人员不必从头设计创建和管理一个标准 Windows 应用程序所需的程序，而是“站在巨人肩膀上”，从一个比较高的起点编程，故节省了大量的时间；其次，它提供了大量的代码，指导用户编程时实现某些技术和功能。MFC 库充分利用了 Microsoft 开发人员多年开发 Windows 程序的经验，并可以将这些经验融入到你自己开发的应用程序中去。

MFC 相当彻底地封装了 Win32 软件开发工具包（Software Development Kit，即通常所说的 SDK）中的结构、功能，它为编程者提供了一个应用程序框架，这个应用程序框架为编程者完成了很多 Windows 编程中的例行性工作，如管理窗口、菜单和对话框，执行基本的输入和输出、使用集合类来保存数据对象等等；并且，MFC 使得在程序中使用很多过去很专业、很复杂的编程课题，如 ActiveX、OLE、本地数据库和开放数据库连接性（Open Database Connectivity，简写为 ODBC）、Windows 套接字和 Internet 应用程序设计等，以及其他的应用程序界面特性，如属性页（也叫标签页）、打印和打印预览、浮动的和可定制的工具栏变得更加容易。

对用户来说，用 MFC 开发的最终应用程序具有标准的、熟悉的 Windows 界面，这样的应用程序易学易用；另外，新的应用程序还能立即支持所有标准 Windows 特性，而且是用普通的、明确定义的形式。事实上，也就是在 Windows 应用程序界面基础上定义了一种新的标准——MFC 标准。

为了更好地理解 MFC，我们有必要了解一下 MFC 的发展史。

1.1.1 MFC 发展史

开始，Microsoft 建立了一个 AFX 小组，AFX 代表 Application Framework，即应用程序框架。据说创建该小组原意是为了发布一个 Borland C++的 OWL（Object Windows Language）的竞争性产品，因为那时侯 Borland 公司的应用程序框架 OWL 已经做得相当成功。AFX 小组像 OWL 那样，提出了一个高度抽象 Windows API 的一个类库。

他们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。然后，他们试着花了几个月的时间用这个类库来编写应用程序，结果发现这个类库偏离 Windows API 实在太远，过分抽象并没有太大的实用性，相反大大降低了应用程序的效率。

于是，他们干脆放弃了整个 AFX 类库，对类库进行重新设计。这次，他们采用了自底向上的方法，从已有的 Windows API 着手，将类建立在 Windows API 对象基础上，设计出后来成为 MFC 1.0 的一个类库。但是，你现在仍然可以看到 AFX 时期的痕迹，许多源程序文件有 afx 前缀，如 afxabort.cpp，afxmem.cpp。MFC 延用了许多 AFX 类库的宏，因此我们经常会看到以 AFX 开头的宏。

AFX 小组实际上做了两件工作：MFC 类库和对 MFC 的 IDE 支持（即资源编译器和操作向导）。在 1994 年 4 月份之后，AFX 的名字停止使用，该小组成员成为 Visual C++开发组的一部分，即现在的 MFC 小组。

MFC1.0 版于 1992 年同 Microsoft C/C++同时发布。它提供了对 Windows API 简单的抽象和封装，还没有我们现在常用的文档/视结构特性。但它引入了 CObject，通过 CArchive 的持续化和其他一些 MFC 中仍然使用的特性，从而奠定了 MFC 的基础。

MFC 2.0 在 MFC1.0 基础上增加了文档/视结构框架、OLE1.0 类、消息映射和公用对话框类，废弃了 1.0 版中的 CModalDialog 类并将它的功能移入到 CDialog 中，并增加了工具栏、对话条、分割视窗的支持。MFC 2.1 随同 Visual C++ 1.1 for NT 发布，它把 MFC2.0 移植到了 Win32 上。MFC 2.5 随同 Visual C++1.5 一起发布，它引入了 OLE 2.0 和 ODBC 类。它是最后官方的 16 位发行版，于 1993 年 12 月发布。目前，在开发 16 位 Windows 程序时，Visual C++1.5 和 MFC 2.5 仍然有大量的用户。随后的 MFC 2.51、2.52 纠正了 MFC2.5 中的一些错误，增加了标签式对话框、WinSock 和 MAPI（Microsoft 电子邮件应用程序接口）支持。MFC 3.1 同 Visual C++ 2.1 一起于 1995 年 1 月份发布，它引入了 Windows 95 公共控件（包括动画、热键、图像列表和工具栏提示等）。MFC 4.0 于 1995 年 12 月份同 Visual C++ 4.0 一起发布。Microsoft 直接从 Visual C++ 2.0 一下子跳过一个版本号，升级到了 4.0，以保持 MFC 版本号和 Visual C++版本号的一致性，但这种一致性又在 Visual C++ 5.0 中打破了。在 MFC 4.0 中增加了 CSynchronize,CMutex,CEvent,CMultiLock,CShellNew 以更好地支持多线程以及 Windows 95 的其他一些特性。Visual C++还引入了 Component Gallery（组件画廊）、STL 支持和大量的新特性。MFC 4.1 最重要的特性是支持 Win32。许多 MFC 开发者一直都在使用该版本。MFC 4.1 修正了 4.0 的一些错误并增加了 Internet 特性。MFC 4.2 增加了 ISAPI 和 OCX 容器支持。

MFC 4.21 于 1997 年 3 月 19 日同 Visual C++ 5.0 一起发布，它是目前最新和最完善的 MFC 版本。它只增加了对微软的 IntelliMouse（智能鼠标器）的支持。Visual C++ 6.0 再次和 MFC 的版本同步。它包括了许多新特性：封装了作为 IE 4.0 一部分引入的 Windows 通

用控件的新的 MFC 类；对动态的 HTML 的支持；Active Document Containment:OLE DB 与 Privider 的绑定。

MFC 发行版列表如下：

- MFC Release MSVC Release 16 位或 32 位备注
- 1.0 16 简单的封装 Windows
- 2.0 1.0 16 增加了文档/视结构
- 2.1 1.1 for NT 32 第一个 NT 的发行版
- 2.5 1.5 16 OLE/ODBC，最后一个 16 位版本
- 2.51 2.0 16 修正错误
- 2.52 2.1 16 增加标签式对话框
- 3.0 2.0 32 标签式对话框、可停泊工具栏
- 3.1 2.1 32 Winsock/MAPI, Windows 公共控制
- 4.0 4.0 32 Win 95, 线程类, OCX 容器
- 4.1 4.1 32 sweeper (WinInet) classes

以上是最后支持 Win32s 的版本

- 4.2 4.2 32 修正错误, ISAPI classes
- 4.2b internet dl 32 修正错误
- 4.21 5.0 32 IntelliMouse™ support
- 6.0 动态 HTML;Active Document Containter
- .NET 全面基于.NET

1.1.2 MFC 类库概念和组成

类库是一个可以在应用中使用的相互关联的 C++类的集合。类库有些随编译器提供，如 Borland C++ Turbo Vision 等；有的是由其他软件公司销售，如用于数据库开发的 Code Base；有的则是由用户自己开发的。比如图像处理类库完成图像显示、格式转换和量化等；串行通信类库用于支持串行口输入输出。有些情况下用户可以直接利用类库中包含的类定义应用程序所需的变量，有时则需要从类库所提供的类中派生出新的类，这依赖于类库的设计和具体的应用程序。

Microsoft 提供了一个基础类库 MFC，其中包含用来开发 C++和 C++ Windows 应用程序的一组类。基础类库的核心是以 C++形式封装了大部分的 Windows API。类库表示窗口、对话框、设备上下文、公共 GDI 对象，例如画笔、调色板、控制框和其他标准的 Windows 部件。这些类提供了一个面向 Windows 中结构的简单的 C++成员函数的接口。

MFC 可分为两个主要部分：基础类、宏和全局函数。

MFC 基础类

MFC 中的类按功能来分可划分为以下几类：

- 基类
- 应用程序框架类

- 应用程序类
- 命令相关类
- 文档/视类
- 线程类
- 可视对象类
- 窗口类
- 视类
- 对话框类
- 属性表
- 控制类
- 菜单类
- 设备描述表
- 绘画对象类
- 通用类
- 文件
- 诊断
- 异常
- 收集
- 模板收集
- 其他支持类
- OLE2 类
- OLE 基类
- OLE 可视编辑包装程序类
- OLE 可视编辑服务器程序类
- OLE 数据传输类
- OLE 对话框类
- 其他 OLE 类
- 数据库类

宏和全局函数

若某个函数或变量不是某个类的一个成员，那么它是一个全局函数或变量。Microsoft 基本宏和全局函数提供了以下功能：

- 数据类型
- 运行时刻对象类型服务
- 诊断服务
- 异常处理
- CString 格式化及信息框显示
- 消息映射

- 应用消息和管理
- 对象连接和嵌入（OLE）服务
- 标准命令和 Windows IDs

注意：全局函数以“Afx”为前缀，所有全局变量都是以“Afx”为前缀，宏不带任何特别前缀，但是全部大写。

常见的全局函数和宏有：AfxGetApp，AfxGetMainWnd，AfxMessageBox 和 DEBUG_NEW 等，我们会在后面的章节中用到并对它们进行介绍。

从继承关系来看，又可将 MFC 中的类分成两大类：大多数的 MFC 类是从 CObject 继承下来；另外一些类则不是从 CObject 类继承下来，这些类包括：CString（字符串类），CTime（日期时间类），CRect（矩形类）和 CPoint（点）等，它们提供程序辅助功能。

由于 MFC 中大部分类是从 CObject 继承下来的，CObject 类描述了几乎所有的 MFC 中其他类的一些公共特性，因此我们有必要理解 CObject 类。

我们首先查看一下 CObject 类的定义，CObject 类的定义如下所示：

```
// class CObject is the root of all compliant objects
class CObject
{
public:
// Object model (types, destruction, allocation)
virtual CRuntimeClass* GetRuntimeClass() const;
virtual ~CObject(); // virtual destructors are necessary
// Diagnostic allocations
void* PASCAL operator new(size_t nSize);
void* PASCAL operator new(size_t, void* p);
void PASCAL operator delete(void* p);

#if defined(_DEBUG) && !defined(_AFX_NO_DEBUG_CRT)
// for file name/line number tracking using DEBUG_NEW
void* PASCAL operator new(size_t nSize, LPCSTR lpszFileName, int nLine);
#endif
// Disable the copy constructor and assignment by default so you will get
// compiler errors instead of unexpected behaviour if you pass objects
// by value or assign objects.
protected:
CObject();
private:
CObject(const CObject& objectSrc); // no implementation
void operator=(const CObject& objectSrc); // no implementation
```

```
// Attributes
public:
BOOL IsSerializable() const;
BOOL IsKindOf(const CRuntimeClass* pClass) const;
// Overridables
virtual void Serialize(CArchive& ar);
// Diagnostic Support
virtual void AssertValid() const;
virtual void Dump(CDumpContext& dc) const;
// Implementation
public:
static const AFX_DATA CRuntimeClass classCObject;
#ifdef _AFXDLL
static CRuntimeClass* PASCAL _GetBaseClass();
#endif
};
```

CObject 类为派生类提供了下述服务：

对象诊断

MFC 提供了许多诊断特性，它可以：

输出对象内部信息：CDumpContext 类与 CObject 的成员函数 Dump 配合，用于在调试程序时输出对象的内部数据。

对象有效性检查

重载基类的 AssertValid 成员函数，可以为派生类的对象提供有效性检查。

运行时访问类的信息

MFC 提供了一个非常有用的特性，它可以进行运行时的类型检查。如果从 CObject 派生出一个类，并使用了以下 3 个宏（IMPLEMENT_DYNAMIC，IMPLEMENT_ DYNCREATE 或 IMPLEMENT_SERIAL）之一，就可以：

运行时访问类名

安全可靠地把通用的 CObject 指针转化为派生类的指针。比如，我们定义一个主窗口类：

```
CMyFrame:public CFrameWnd
{
......
}
```

然后我们使用这个类：

```
CMyFrame *pFrame=(CMyFrame*)AfxGetMainWnd();
pFrame->DoSomeOperation();
```

AfxGetMainWnd 是一个全局函数，返回指向应用程序主窗口的指针，类型为 CWnd*。因此我们必须对它进行强制类型转换，但我们如何知道是否转换成功了呢？我们可以使用 CObject 的 IsKindOf()成员函数检查 pFrame 的类型，用法如下：

```
ASSERT(pFrame->IsKindOf(RUN_TIMECLASS(CMyFrame)));
```

将上一语句插入到 pFrame-> DoSomeOperation()之前，就可以在运行时作类型检查，当类型检查失败时，引发一个断言（ASSERT），中断程序执行。

对象持续性

通过与非 CObject 派生的档案类 CArchive 相结合，提供将多个不同对象以二进制形式保存到磁盘文件（Serialization）中以及根据磁盘文件中的对象状态数据在内存中重建对象（Deserialization）的功能。

然而，MFC 不仅仅是一个类库，它还提供了一层建立在 Windows API 的 C++封装上的附加应用程序框架。该框架提供了 Windows 程序需要的多数公共用户界面。

注意：所谓应用程序框架指的是为了生成一般的应用所必须的各种组件的集成。应用框架是类库的一种超集。一般的类库只是一种可以用来嵌入任何程序中的、提供某些特定功能（如图像处理、串行通信）的孤立的类的集合，但应用框架却定义了应用程序的结构，它的类既相互独立，又相互依赖，形成一个统一的整体，可以用来构造大多数应用程序。中国用户熟悉的 Borland C++的 DOS 下的 Turbo Vision 和 Windows 下的 OWL（Object Windows Language）都是应用框架的例子。

下面我们举个具体的例子来说明 MFC 所提供的应用程序框架，程序如下：

```
#include<afxwin.h>
//derived an application class
class CMinMFCApp:public CWinApp
{
public:
BOOL InitInstance();
};

//Derive the main window class
class CMainWindow:public CFrameWnd
{
public:
CMainWindow();
```

```
DECLARE_MESSAGE_MAP()
};
BEGIN_MESSAGE_MAP(CMainWindow,CFrameWnd)
END_MESSAGE_MAP()

/*CMinMFCApp Member Functions*/
BOOL CMinMFCApp::InitInstance()
{
m_pMainWnd=new CMainWindow();
m_pMainWnd->ShowWindow(m_nCmdShow);
m_pMainWnd->UpdateWindow();
return TRUE;
}

/*CMainWindow member functions*/
CMainWindow::CMainWindow()//constructor
{
Create(NULL,
"Min MFC Application",
WS_OVERLAPPEDWINDOW,
rectDefault,
NULL,
NULL);
}
/*an instance of type CMinMFCApp*/
CMinMFCApp ThisApp;
```

程序段定义了一个最小的 MFC 应用程序所需的框架程序。其中声明了 CMinMFCApp 类，它是从应用程序类 CWinApp 中派生出来的；而窗口 CMainWindow 类，是从框架窗口 CFrameWnd 类派生出来的。我们还用 CMinMFCApp 定义了一个全局对象 ThisApp。读者也许会问，为什么没有 WinMain 函数？因为 MFC 已经把它封装起来了。在程序运行时，MFC 应用程序首先调用由框架提供的标准的 WinMain 函数。在 WinMain 函数中，首先初始化由 CMinMFCApp 定义的惟一的实例，然后调用 CMinMFCApp 继承 CWinApp 的 Run 成员函数，进入消息循环。退出时调用 CWinApp 的 ExitInstance 函数。

从上面的说明可以看出，应用程序框架不仅提供了构建应用程序所需要的类（CWinApp，CFrameWnd 等），还定义了程序的基本执行结构。所有的应用程序都在这个基本结构的基础上完成不同的功能。

MFC 除了定义程序执行的结构之外，还定义了 3 种基本的主窗口模型：单文档窗口，多文档窗口和对话框作为主窗口。

注意：Visual C++提供了两个重要的工具，用于支持应用程序框架，它们就是前面提到 Application Wizard 和 ClassWizard。Application Wizard 用于在应用程序框架基础上迅速生成用户的应用程序基本结构。ClassWizard 用于维护这种应用程序结构。

1.1.3　MFC 的优点

Microsoft MFC 具有以下不同于其他类库的优势：

- 完全支持 Windows 所有的函数、控件、消息、GDI 基本图形函数、菜单及对话框。类的设计以及同 API 函数的结合相当合理。
- 使用与传统的 Windows API 同样的命名规则，即匈牙利命名法。
- 进行消息处理时，不使用易产生错误的 switch/case 语句，所有消息映射到类的成员函数，这种直接消息到方法的映射对所有的消息都适用。它通过宏来实现消息到成员函数的映射，而且这些函数不必是虚拟的成员函数，这样不需要为消息映射函数生成一个很大的虚拟函数表（V 表），节省内存。
- 通过发送有关对象信息到文件的能力提供更好的判定支持，也可确认成员变量。
- 支持异常错误的处理，减少了程序出错的机会。
- 运行时确定数据对象的类型，这允许实例化时动态操作各域。
- 有较少的代码和较快的速度。MFC 库只增加了少于 40kB 的目标代码，效率只比传统的 C Windows 程序低 5%。
- 可以利用与 MFC 紧密结合的 Application Wizard 和 ClassWizard 等工具快速开发出功能强大的应用程序。

另外，在使用 MFC 时还允许混合使用传统的函数调用。

1.2　Visual C++ .NET 与老版本的区别

1. 工程种类更为丰富

新建的工程分为三大类：.NET projects，Win32 projects 和 Utility projects。

在.NET projects 工程中新加了 Managed C++ Application，Managed C++ class library，Managed C++ empty project 和 Managed C++ Web Service。在 Win32 projects 工程中新加了 ATL project，ATL server project，ATL Server Web service project，MFC Application 和 MFC DLL。在 Utility projects 工程中增加了 Custom Application Wizard 和 Makefile project。

2. 编译器和链接器的性能得到提高

使用/OI 选项编译生成的代码大小将比 Visual C++ 6 版小 5%到 10%。

链接器使用了/FIXED 选项来创建更小的供发行的应用程序。因此，在使用剖析程序时，由于需要重定位信息，链接器必须使用/PROFILE 和/FIXED:NO 选项。这同样适用于其他

如 Bounds Checker 或 Purify 之类的链接后（post-link）工具。

新增的/EH 编译选项可以更有效的控制 C++异常处理。C++同步异常处理允许编译器生成更小的代码，因此它是 Visual C++.NET 新的默认 C++异常处理模式。

对用来控制代码优化所面向处理器的编译器选项/G3，/G4，/G5，/G6 和/GB 作了修改。

将/GX 编译器选项映射为/EHsc。允许使用链接器选项/PDBTYPE 指定包括调试信息的程序数据库（PDB）。该选项可以节省磁盘空间并加快链接。

在 NMAKE 中支持批处理规则。

3. 增加了部分类的功能

可以在 Web 浏览器或支持 ActiveX 文档的 OLE 容器（例如 Microsoft Office Binder）的整个客户区显示活动的文档。Win32 Internet API (WinInet)使 Internet 成为任意应用程序的一个完整部分并简化了 Internet 服务，如 FTP，HTTP 和 Gopher 的访问。

增加了对 DAO 的支持。增加了对 ODBC 的支持，并对 MFC ODBC 类作了几个重要的修改。

COleDateTime 成员函数 SetDate，SetDateTime 和 SetTime 的返回值从 BOOL 改变为 int。每一个成员函数当 COleDateTime 对象被正确设置时返回 0，否则返回 1。该返回值基于 DateTimeStatus 枚举类型。

新增示例程序 IMAGE。该程序生成一个可以异步下载数据的 ActiveX 控件。ATL 新版支持创建既小又快的 ActiveX 控件。

改善了下列通用浮点超越函数的性能：pow，sqrt，log，log10，sin，cos，tan，asin，acos 和 atan。

改善了内存移动和内存拷贝函数的性能。

注意：OLE DB 是一组 OLE 接口，它使应用程序可以以统一的方式访问保存在不同信息源中的数据。这些接口支持适合于数据源的大量数据库功能，并允许数据源共享其数据。所配套的 OLE DB 软件开发工具包所提供的一组软件部件、工具和文档可以在开发 OLE DB 客户和提供程序提供帮助。

4. 集成开发环境的功能更为强大

新的 Application Wizard 可以自动管理基于对话框的应用程序中的对话框类。只需要简单的创建一个基于对话框的应用程序，并选择对自动化的支持，就可以像早期版本的 AppWizard 一样，得到一个支持基本自动化的基于对话框的应用程序。通过单独的代理类，对话框类也可以通过自动化导出。你可以添加方法和属性来导出对话框中的元素。

定制的 Application Wizard 可以改变工程创建时的设置。例如，你可以在目标创建之后调整编译器、链接器和查看设定或者添加定制的创建步骤。

asynchronous (URL) moniker 允许应用程序异步地下载文件和控件属性，以便在任务完成后为其他进程释放系统资源。

ERRLOOK 工具可以使用系统错误的值来检索相应的错误消息，其中包括 OLE HRESULT。错误值可以通过包括拖放、编辑命令等的多种方法给出。由 ERRLOOK 所返