

30100

《电脑编程技巧与维护》杂志十周年庆典暨真情回馈读者活动  
《电脑编程技巧与维护》杂志社策划

编程技巧典型案例集锦系列

《电脑编程技巧与维护》杂志社 编著

# Visual C++

Visual C++ Programming

## 编程技巧典型案例解析

Visual C++ Programming Visual C++ Programming

### ——基础与应用篇

下

- 通用设备驱动程序的设计与实现
- 创建可多次扩展的对话框
- 实现内存使用率的动态图形显示
- 动态链接库输出函数的动态加载
- 在多线程中实现 MFC 成员函数的调用
- 控制线程的运行技术

注意：光盘

超值CD，46个经典案例，50000条程序代码，编程高手经验汇集，现学现用



中国电力出版社  
www.infopower.com.cn

《电脑编程技巧与维护》杂志十周年庆典暨真情回馈读者活动  
《电脑编程技巧与维护》杂志社策划

编程技巧典型案例集锦系列

《电脑编程技巧与维护》杂志社 编著

# Visual C++

## 编程技巧典型案例解析

### ——基础与应用篇

下



中国电力出版社  
www.infopower.com.cn

## 内 容 简 介

Visual C++ 是 Windows 编程的主要工具, 与 Windows 的紧密结合使它在软件底层开发上占有非常大的优势。它采用一种巧妙的方法将 Windows 的编程复杂性封装起来, 使得编程人员可以比较轻松地进行 Windows 应用程序的设计。本书从 Visual C++ 的基础与应用实例入手, 系统地介绍了 Visual C++ 在工程开发中的编程知识、方法和技巧, 内容包括用户界面、对话框、菜单、工具条、状态栏的创建方法以及各种类向导、控件、组件、动态链接库的特点和使用等, 使读者在学习 Visual C++ 软件的同时, 能够迅速掌握工程项目开发的思路, 轻松解决项目开发过程中出现的问题。

本书注重工程实践, 实用性强, 是广大 Visual C++ 程序员和编程爱好者的首选参考书, 也是进行课程项目开发、毕业项目设计的高等院校学生的优秀读物, 同时也可作为社会相关高等培训学校的理想案例教程。

### 图书在版编目 ( CIP ) 数据

Visual C++ 编程技巧典型案例解析——基础与应用篇 ( 下 ) / 《电脑编程技巧与维护》杂志社编著. —北京: 中国电力出版社, 2005  
( 编程技巧典型案例集锦系列 )  
ISBN 7-5083-3265-2

I.V... II.电... III.C 语言 - 程序设计 IV.TP312

中国版本图书馆 CIP 数据核字 ( 2005 ) 第 018205 号

### 版权声明

本书由中国电力出版社独家出版。未经出版者书面许可, 任何单位和个人不得以任何形式复制或传播本书的部分或全部内容。

本书内容所提及的公司及个人名称、产品名称、优秀作品及其名称, 均为所属公司或者个人所有, 本书引用仅为宣传之用, 绝无侵权之意, 特此声明。

策 划: 裴红义  
姚贵胜  
责任编辑: 李富颖  
责任校对: 崔燕菊  
责任印制: 李志强

丛 书 名: 编程技巧典型案例集锦系列

书 名: Visual C++ 编程技巧典型案例解析——基础与应用篇 ( 下 )

编 著: 《电脑编程技巧与维护》杂志社

出版发行: 中国电力出版社

地址: 北京市三里河路 6 号 邮政编码: 100044

电话: ( 010 ) 88515918 传真: ( 010 ) 88518169

印 刷: 利森达印务有限公司

开本尺寸: 185 × 260 印 张: 20.5

书 号: ISBN 7-5083-3265-2

版 次: 2005 年 7 月北京第 1 版

印 次: 2005 年 7 月第 1 次印刷

印 数: 1~5000

定 价: 35.00 元 ( 含 1CD )

# 丛书序

在《电脑编程技巧与维护》杂志创刊 10 周年之际，为了真诚回报多年来一直关爱和支持本刊的广大读者，《电脑编程技巧与维护》杂志社和中国电力出版社共同策划出版了《编程技巧典型案例集锦系列》丛书。《电脑编程技巧与维护》杂志是为从事电脑编程、系统应用开发人员创办的专业性和实用性都很强的技术刊物，它从 1994 年创刊，十多年来始终遵循着“实用第一，智慧密集”的办刊宗旨，紧跟计算机软硬件技术发展和应用趋势，不断求变创新，针对软件开发过程中许多关键技术问题，着重提供各类解决方案。对电脑编程人员来说，程序开发能力的提高，除了对语言和算法的学习外，还要集思广益，充分借鉴参考别人的长处，深入透彻地理解其中的精髓，然后融入到自己的设计方案中去，这样无论是对于自身还是整体都有莫大的提高，这也正是我们编写这套系列丛书的初衷。

本丛书包括《Visual C++ 编程技巧典型案例解析——基础与应用篇（上）》、《Visual C++ 编程技巧典型案例解析——基础与应用篇（下）》、《Visual C++ 编程技巧典型案例解析——图形图像处理与数据库篇》、《Visual C++ 编程技巧典型案例解析——网络与通信及计算机安全与维护篇》、《Visual Basic 编程技巧典型案例解析》、《Delphi 编程技巧典型案例解析》、《C#编程技巧典型案例解析》、《Java 编程技巧典型案例解析》、《PowerBuilder 管理信息系统编程技巧典型案例解析》9 册共 545 个典型案例。每册书的编程案例，均依不同的编程应用分成若干章，条目清晰可查，使用极为方便。

本丛书选编了《电脑编程技巧与维护》杂志近一两年发表的和一部分尚未发表而又极为实用、精彩的典型编程实例，特点是：其各册内容均来自编程高手的智慧，凝结了 500 余位编程高手与名家的心血，关键技术专家点评；其案例是从实际项目提炼出的开发范例，超过 800 个技术要点的经典解决方案。案例讲解部分先给出设计目标，然后介绍实现目标的基本思想和方法，最后详细给出其核心程序的源代码，对程序的关键部分进行讲解并给出程序的运行效果；其编程技巧新颖实用，构思巧妙，汇集了众多顶级程序员和业界知名专家的成功经验，告诉读者最好的创意和最实用的方法。全套书既讲究内容的深入性、专业性和权威性，同时兼顾轻松、通俗易懂、时效性强的特点，带给读者的是一份清

新、纯粹的体验感受。

本丛书是《电脑编程技巧与维护》杂志资源的二次开发，浓缩了当前主流编程语言 Visual C++、Visual Basic、Delphi、Java、C#、PowerBuilder 等程序设计的精华，其目的是力求为读者建造一个真正的知识整合，是编程思想、编程技术、技巧交流的平台，让读者从中学习到编程高手的诀窍，丰富读者的编程技巧，拓宽读者的编程思路，迅速提升读者的程序开发能力。该丛书可作为高等院校学生进行课程项目开发、毕业项目设计的参考教材，软件从业人员及编程爱好者的珍藏宝典，也可作为高等培训学校的案例教程。

实例导航学编程，自学成才成高手，思想、智慧、理念、经验、技巧无处不在……

《电脑编程技巧与维护》杂志社  
2005年1月

# 前 言

Visual C++ 作为功能强大的面向对象与可视化应用程序开发工具，是业界公认的优秀应用开发工具。Microsoft 的基本类库 MFC 将类之间的关系紧密地联系在一起，而 Visual C++ 支持 MFC 的程序开发，提高了 MFC Application Wizard 的功能，帮助程序员构建了一套基础程序，并从中开发应用程序，因此，可作为 Visual C++ 各种系统软件、应用软件、网络软件和游戏软件等的开发平台。

在《编程技巧典型案例集锦系列》丛书中，《Visual C++ 编程技巧典型案例》精选了《电脑编程技巧与维护》杂志近两年半共 30 期已发表的 238 个精彩编程实例。根据 Visual C++ 的不同应用对象，将其分为《Visual C++ 编程技巧典型案例解析——基础与应用篇（上）》、《Visual C++ 编程技巧典型案例解析——基础与应用篇（下）》、《Visual C++ 编程技巧典型案例解析——图形图像处理与数据库篇》、《Visual C++ 编程技巧典型案例解析——网络与通信及计算机安全与维护篇》四册出版。每一册书都始终遵循“实用第一，智慧密集”的宗旨，介绍了 Visual C++ 开发各类应用程序关键技术的解决案例，并且每一个案例都给出了开发过程、技术难点及其解决的方法和技巧，涉及到 Visual C++ 应用程序设计的新思路和方法。这些典型案例所涵盖的编程技巧是作者经验的总结；具有一定的代表性，很值得借鉴。该书本着实用的原则，紧紧围绕着一个主题展开，循序渐进、由浅入深地介绍了使用 Visual C++ 进行应用程序开发的思想方法与编程技巧。

全书共分两章。第 1 章 基础与应用编程实例。该章在对 Visual C++ 基础知识进行了深入探讨和分析的基础上，主要围绕 Visual C++ 基础应用，列举了 61 个典型而实用的编程实例；第 2 章 Visual C++ 编程技巧。在该章中列举了实用的编程技巧 40 则，每个技巧均给出了完整的源代码。

本书的主要特色为：①每一章都是通过一个个经典的实例来介绍 Visual C++ 应用编程方法和技巧，避免了枯燥、空洞的理论，并且每一个实例都具有很强的实用性和代表性。在实例的讲解上一般都是先给出设计目标，然后介绍实现该目标的基本思想和方法，最后详细给出其核心程序的源代码，并对程序的关键部分进行讲解，给出程序的运行效果。②所选的每一个实例都是从事 Visual C++ 应用编程人员的经验总结，具有很强的实用性，其中很多编程技巧可供借鉴。③每一个实例的程序源代码都经过上机调试通过，

给程序开发人员移植源代码带来了方便，加快编程应用的步伐。④对个别版本和开发环境稍微低一些的经典实例进行点评和分析，起到触类旁通的效果。

本书是《电脑编程技巧与维护》杂志的二次开发，浓缩了 Visual C++ 基础应用程序设计的精华，其目的是提升读者 Visual C++ 程序开发的能力，把应用 Visual C++ 进行编程的心得体会、经验与读者共享。该书定位于有 Visual C++ 应用基础的编程人员和应用开发人员，对初学 Visual C++ 编程的新手也有一定的参考价值。书中内容深入、概念清晰、层次分明，实例典型而实用，但不足及疏漏之处在所难免，恳请广大读者批评指正。

《电脑编程技巧与维护》杂志社

2005 年 1 月

# 目 录

丛书序

前 言

## 第 1 章 基础与应用编程实例

实例 1 C++ 中用指针调用类成员函数的问题及解决的方法 .....	3
实例 2 在 Visual C++ 中定制 DIB 类 .....	8
实例 3 在 Visual C++ 中实现自定义事件的编程 .....	12
实例 4 在 Visual C++ 6.0 中利用 ClassWizard 调用 COM 组件 .....	17
实例 5 用 Visual C++ 实现二维等值线的 COM 组件 .....	19
实例 6 在 Visual C++ 中使用 DataGrid 控件 .....	23
实例 7 对 MFC 树形控件 (CTreeCtrl) 进行扩展 .....	30
实例 8 在 Visual C++ 中利用子类化技术扩展通用控件的功能 .....	33
实例 9 对“Visual Studio 6.0 风格”程序中 Workspace 的 3 个改进 .....	36
实例 10 利用 Visual C++ 删除程序自己的方法 .....	42
实例 11 用 Visual C++ 实现 USB 接口智能卡读写器的编程 .....	44
实例 12 通用设备驱动程序的设计与实现 .....	47
实例 13 用 Visual C++ 6.0 实现屏幕截取 .....	55
实例 14 用 Visual C++ 实现全屏幕显示 .....	58
实例 15 全屏幕取词 For Windows 2000/XP .....	60
实例 16 Visual C++ 对多显示器系统的编程实现 .....	71
实例 17 Visual C++ 6.0 子用户界面线程及托盘的实现 .....	74
实例 18 Visual C++ 环境下多虚拟桌面程序的实现 .....	81
实例 19 在 Visual C++ 中实现对多画面窗口的控制 .....	88
实例 20 实现 Visual Studio 风格的窗口 .....	91
实例 21 用 Visual C++ 实现窗口的分割 .....	96
实例 22 在 Visual C++ 中窗口子类化技术的实现及其应用 .....	104
实例 23 用 Visual C++ 实现动态创建对话框 .....	107
实例 24 用 Visual C++ 制作个性对话框 .....	111
实例 25 用 Visual C++ 5.0/6.0 实现“更改图标”对话框 .....	119
实例 26 用 Visual C++ 设计实现可多次扩展的对话框 .....	125
实例 27 在对话框中使用用户界面对象的编程 .....	131



实例 28	用 Visual C++ 实现自定义工具栏 .....	135
实例 29	用 Visual C++ 编程实现显示和隐藏工具条 .....	140
实例 30	用 Visual C++ 6.0 实现汉字到区位码的批量转化 .....	146
实例 31	在 Visual C++ 6.0 中实现程序单一运行 .....	148
实例 32	ActiveMovie 控件在 Visual C++ 多媒体程序开发中的应用 .....	151
实例 33	借助内嵌资源实现 Visual C++ 对 Flash 动画的播放 .....	158
实例 34	用 Visual C++ 播放 CD 中第一首曲子的实现过程 .....	161
实例 35	用 Visual C++ 6.0 实现语音采集与播放的控件 .....	165
实例 36	利用控件聚合技术在 Visual C++ 中实现 MSFlexGrid 的编辑功能 .....	169
实例 37	用 Visual C++ 实现与 USB 驱动程序的通信 .....	173
实例 38	Visual C++ 与 Visual Basic 混合编程的几种方法 .....	177
实例 39	用 Visual C++ 实现内存使用率的动态图形显示 .....	181
实例 40	动态链接库输出函数的动态加载 .....	186
实例 41	基于 Visual C++ 和 L 系统的自然景物模拟 .....	191
实例 42	用 Visual C++ 开发基于 ToolHelp32 的进程监控程序 .....	197
实例 43	在多线程中实现 MFC 成员函数的调用 .....	201
实例 44	Pop - Up Menu 在非模态对话框中实现 UPDATE-COMMAND-UI 机制 .....	205
实例 45	用 Visual C++ 6.0 实现资源 ID 排序 .....	210
实例 46	在 Win32 下的多线程编程 .....	215
实例 47	用 Visual C++ 实现自画式菜单 .....	221
实例 48	编程实现从应用程序中直接关闭计算机的方法 .....	232
实例 49	在 Visual C++ 中使用托盘图标功能编写计算机定时关机程序 .....	235
实例 50	用 Visual C++ 操作 Office 文档功能的增强 .....	247
实例 51	用 Visual C++ 编程实现 Wizard 程序 .....	252
实例 52	编程实现显示文件信息 .....	257
实例 53	在 MFC 应用程序中浏览 PDF、Word 文档文件 .....	261
实例 54	在 Visual C++ 环境下控制线程的运行技术 .....	264
实例 55	在 Visual C++ 下对匿名管道的编程实现 .....	269
实例 56	Plug - In 应用软件的一种实现方法 .....	272
实例 57	在 Visual C++ 6.0 下利用互斥量同步线程来实现文件读取进度条 .....	277
实例 58	用 Visual C++ 实现通用的报表控件 .....	283
实例 59	用 Visual C++ 实现搜索功能 .....	289
实例 60	用 Visual C++ 6.0 开发 HTML 帮助文件 .....	293
实例 61	Visual C++ 与 MATLAB 的数字基带传输系统仿真的实现 .....	299

## 第 2 章 Visual C++ 编程技巧

实例 62	Visual C++ 编程小技巧 .....	305
-------	------------------------	-----

# 第 1 章

## 基础与应用编程实例





## ■实例 1

### C++ 中用指针调用类成员函数的问题及解决的方法

在编程工作中常会遇到在一个“类”中通过函数指针调用成员函数的要求，如当在一个类中使用了 C++ 标准库中的排序函数 `qsort` 时，因 `qsort` 参数需要一个“比较函数”指针，如果这个“类”使用某个成员函数作“比较函数”，就需要将这个成员函数的指针传给 `qsort` 供其调用。本实例所讨论的用指针调用“类”的成员函数包括以下 3 种情况：

(1) 将“类”的成员函数指针赋予同类型非成员函数指针，如：

例 1

```
#include <stdlib.h>
typedef void( * Function1)(); //定义一个函数指针类型
Function1 f1;
class Test1
{
public:
    //...被调用的成员函数
    void Memberfun1(){ printf( "%s \n", "Calling Test3: : Memberfun2 OK"); };
    void Memberfun2()
    {
        f1 = reinterpret_cast <Function1>(Memberfun1); //将成员函数指针赋予 f1. 编译出错
        f1();
    }
    //...
};
int main()
{
    Test1 t1;
    t1.Memberfun2();
    return 0;
}
```

(2) 在一个“类”内，有标准库函数，如 `qsort` 或其他全局函数，用函数指针调用类的成员函数，如：

例 2

```
#include <stdlib.h>
class Test2
{
private:
    int data[2];
```

## Visual C++ 编程技巧典型案例解析——基础与应用篇(下)

```

//...
public:
//...
int _cdecl Compare(const void * elem1, const void * elem2) //成员函数
{
    printf("%s \n", "Calling Test2:: Memberfun OK");
    return * ((int *)elem1) - * ((int *)elem2);
}
void Memberfun()
{
    data[0] = 2; data[1] = 5;
    qsort( data, 2, sizeof(int), Compare); //标准库函数调用成员函数.编译出错
}
//...
};
int main()
{
    Test2 t2;
    t2.Memberfun(); //调用成员函数.
    return 0;
}

```

(3) 同一个“类”内，一个成员函数调用另一个成员函数，如：

## 例 3

```

#include "stdlib.h"
class Test3
{
    public:
    //...
    void Memberfun1( void (* f2)() ) { f2(); } //成员函数 1 调用成员函数 2
    void Memberfun2() { printf("%s \n", "Calling Test3:: Memberfun2 OK"); } //成员函数 2
    void Memberfun3() { Memberfun1( Memberfun2); } //编译出错
    //...
};
int main()
{
    Test3 t3;
    t3.Memberfun3(); //调用成员函数
    return 0;
}

```

以上 3 种情况的代码语法上没有显著的错误，在一些较早的编译环境中，如 Visual C++ 4.0，通常可以编译通过或至多给出问题提醒（Warning）。后来的编译工具，如 Visual C++ 6.0 和其他一些常用的 C++ 编译软件，不能通过以上代码的编译，并指出错误如下（以第 3 种情况用 Visual C++ 6.0 编译为例）：

```

error C2664: 'Memberfun1': cannot convert parameter 1 from 'void (void)' to 'void
(_cdecl *) (void)'

```

None of the functions with this name in scope match the target type

即：**Memberfun1** 参数中所调用的函数类型不对。

按照以上提示，仅通过改变函数的类型无法消除错误，但是，如果单将这几个函数从类的定义中拿出来，不作任何改变就可以消除错误通过编译，仍以第 3 种情况为例，以下代码可通过编译：

```

#include <stdlib.h>
void Memberfun1( void (* f2)() ) { f2(); } //原成员函数 1 调用成员函数 2
void Memberfun2() { printf( "%s \n", "Calling Test3::Memberfun2 OK"); } //原成员函数 2
void Memberfun3() { Memberfun1( Memberfun2); }
int main()
{
    Memberfun3 ();
    return 0;
}

```

第 1、2 种情况和第 3 种情况完全相同。

由此可以得出结论，以上 3 种情况编译不能通过的原因表面上并不在于函数类型调用不对，而是与“类”有关。没通过编译的情况是用函数指针调用了“类”的成员函数，通过编译的是用函数指针调用了非成员函数，而函数的类型完全相同。那么，“类”的成员函数指针和非成员函数指针有什么不同吗？

在下面的程序中，用 `sizeof()` 函数可以查看各种“类”的成员函数指针和非成员函数指针的长度 (size) 并输出到屏幕上。

```

#include "stdafx.h"
#include <iostream>
#include <typeinfo.h>
class Test; //一个未定义类
class Test2 //一个空类
{
};
class Test3 //一个有定义类
{
public:
    //...
    void (* memberfun)();
    void Memberfun1( void (* f2)() ) { f2(); } //成员函数 1 调用成员函数 2
    void Memberfun2(); //成员函数 2
    //...
};
class Test4: virtual Test3, Test2 //一个有 virtual 继承的类(derivative class)
{
public:
    void Memberfun1( void (* f2)() ) { f2(); }
};
class Test5: Test3, Test2 //一个继承类(derivative class)
{
public:
    void Memberfun1( void (* f2)() ) { f2(); }
};
int main()
{
    std::cout << "一般函数指针长度 = " << sizeof(void(*)()) << '\n';
    std::cout << "-类的成员函数指针长度 - " << '\n' << '\n';
    std::cout << "Test3 类成员函数指针长度 = " << sizeof(void(Test3::*)()) << '\n' << '\n';
    std::cout << "Test5 类成员函数指针长度 = " << sizeof(void (Test5:: *)()) << '\n';
    std::cout << "Test4 类成员函数指针长度 = " << sizeof(void (Test4:: *)()) << '\n';
}

```

```
std::cout << "Test 类成员函数指针长度 = " << sizeof(void(Test::*)()) << '\n';
return 0;
}
```

输出结果为 (Visual C++ 6.0 编译, 运行于 Windows 98 操作系统, 其他操作系统可能有所不同) :

```
一般非成员函数指针长度 = 4
- 类的成员函数指针长度 -
Test3 类成员函数指针长度 = 4
Test5 类成员函数指针长度 = 8
Test4 类成员函数指针长度 = 12
Test 类成员函数指针长度 = 16
```

以上结果表明, 在 32 位 Windows 98 操作系统中, 一般函数指针的长度为 4 字节 (32 位), 而类的成员函数指针的长度随类的定义与否、类的继承种类和关系而变, 从无继承关系类 (Test3) 的 4 字节 (32 位) 到有虚继承关系类 (Virtual Inheritance) (Test4) 的 12 字节 (96 位), 仅有声明 (declaration) 没有定义的类 (Test) 因为与其有关的一些信息不明确成员函数指针最长为 16 字节 (128 位)。显然, 与一般函数指针不同, 指向“类”的成员函数的指针不仅包含成员函数地址的信息, 而且包含与类的属性有关的信息, 因此一般函数指针和类的成员函数指针是根本不同的两种类型, 当然也就不能用一般函数指针直接调用类的成员函数, 这就是为什么本文开始提到的 3 种情况编译出错的原因。尽管使用较早版本的编译软件编译仍然可以通过, 但这会给程序留下严重的隐患。

至于为什么同样是指向类的成员函数的指针, 其长度竟然不同, 从 32 位 ~ 128 位, 差别很大, 由于没有看到微软官方的资料, 只能推测 Visual C++ 6.0 在编译时对类的成员函数指针进行了优化, 以尽量缩短指针长度, 毕竟使用 128 位或 96 位指针在 32 位操作系统上对程序性能会有影响, 但是无论如何优化, 类的成员函数指针包含一定量的对象 (Objects) 信息是确定的。其他的操作系统和编译软件是否进行了类似的处理, 读者可以用以上程序自己验证。

那么, 如何用指针调用类的成员函数? 可以考虑以下方法:

(1) 将需要调用的成员函数设为 static 类型。如例 2, 将 class Test2 成员函数 Compare 定义前加上 static, 代码如下:

```
class Test2
{
//...
int static _cdecl Compare(const void * elem1, const void * elem2) //成员函数
//其他不变
}
```

改变后的代码编译顺利通过, 原因是 static 类型的成员函数与类是分开的, 其函数指针也不包含对象信息, 与一般函数指针一致。这种方法虽然简便, 但有两个缺点: ①被调用的函数成员定义内不能出现任何类的成员 (包括变量和函数); ②由于使用了 static 成员, 类在被继承时受到了限制。

(2) 使用一个函数参数含有对象信息的 static 类型的成员函数为中转, 间接地调用其他成员函数。如例 3, 将类 Test3 作如下修改, main () 函数不变, 则可顺利通过编译:

```
class Test3
{
public:
//...
void static _cdecl Helper(Test3 * test3)
```

```

    {
        test3 ->Memberfun2();
    }
void Memberfun1( void (*f2)(Test3*)) { f2(this); } //将对象信息传给 Helper 函数
void Memberfun2() { printf("%s \n", "Calling Test3::Memberfun2 OK"); } //成员函数 2
void Memberfun3() { Memberfun1( Helper); }
//...
};

```

这种间接方式对成员函数没有任何限制，克服了第一种方法成员函数不能使用任何类的成员的缺点，但由于有 `static` 成员，类的继承仍受到制约。

(3) 使用一个全程函数 (global function) 为中转间接调用类的成员函数，仍以例 3 为例，将代码作如下修改 (Visual C++ 6.0 编译通过)：

```

class Test3;
void _cdecl Helper(Test3 * test3);
class Test3
{
public:
//...
void Memberfun1( void (*f2)(Test3*)) { f2(this); } //成员函数 1 调用成员函数 2
void Memberfun2(){ printf("%s \n", "Calling Test3::Memberfun2 OK"); } //成员函数 2
void Memberfun3() { Memberfun1( Helper); }
//...
};
void _cdecl Helper(Test3 * test3)
{
    test3 ->Memberfun2();
};

```

这个方法对成员函数没有任何要求，但是需要较多的代码。

除上述 3 种方法外还有其他方法，如可以在汇编层面上修改代码解决上述问题等，已不属本文范围。

结论：函数指针不能直接调用类的成员函数，需采取间接的方法。原因是成员函数指针与一般函数指针有根本的不同，成员函数指针除包含地址信息外，同时携带其所属对象信息。本文提供了 3 种办法用于间接调用成员函数，这 3 种办法各有优缺点，适用于不同的场合。

(刘书志)



## ■实例 2

# Visual C++ 中定制 DIB 类

BMP 格式是微软公司为其 Windows 操作系统设置的标准图像格式，在 Windows 的系统软件中包含了一系列支持 BMP 图像处理的 API 函数，但是在编写图像处理的程序时，如果直接调用这些 API 函数，会使程序的组织比较乱，使用也不是很方便。因此，把这些 API 函数用一个类来封装，在编写程序时直接把类添加到工程中，即可完成对 BMP 图像的读取、加载以及各种变换，并转换为 DDB 位图格式，使得可以使用 CDC::BitBlt() 来进行显示。

### 一、BMP 文件分析

BMP 文件是个人计算机上最常见、最简单的图像文件格式之一。BMP 文件可以描述多达 32 位彩色的图像，其文件结构由以下 4 部分组成：

(1) BITMAPFILEHEADER (BMP 文件头结构)，主要包括 BMP 文件格式标志和文件的大小。

(2) BITMAPINFOHEADER (BMP 文件信息头结构)，是对图像属性的描述结构，主要包含了图像的宽度、高度、位颜色数 (1 位色、4 位色、8 位色、24 位色)、重要颜色数等。

(3) RGBQUAD (BMP 文件调色板)，包含红色、绿色、蓝色 3 种基色的值。

(4) BITMAPDATA (BMP 文件数据)，以连续的方式存储，不过需要注意的是图像采用的是相反的存储顺序 (即文件读出的第一行是图像的最后一行)。

### 二、DIB 类的实现

根据 BMP 文件结构，在 DIB 类中定义了以下变量：

```
BITMAPFILEHEADER bmfHeader; // 图像文件头
LPBITMAPINFOHEADER bmlInfoHeader; // 图像信息头指针
BYTE * Handle; // DIB 位图数据指针
LPBITMAPINFO bmlInfo; // BITMAPINFO 数据结构指针，描述了位图大小和颜色数据
BYTE * lpDIBBits; // 图像数据指针，其指向的数组的每项表示图像点的颜色值
long m_Width; // 图像宽度
long m_Height; // 图像高度
```

DIB 的类函数包括：

```
int GetDIBColorsNum(BITMAPINFOHEADER * bmlInfoHeader); // 返回图像颜色数
void LoadDIB(LPTSTR lpFilename); // 装载 BMP 文件
HPALETTE CreateDIBPalette(); // 创建位图调色板
BOOL PaintDIB(HDC hDC, long width, long height); // 把 DIB 位图写入设备句柄
BOOL OnChangeToGray(); // 灰度化图像
```