《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

Visual C++

《电脑编程技巧与维护》杂志社　编著

Visual C++ Programming

编程技巧典型案例解析

——图形图像处理与数据库篇

Visual C++ Programming Visual C++ Programming

- 图库管理系统的设计
- 利用多线程实现屏幕动画
- 实现对火焰的计算机动态仿真
- 使用 MFC 开发具有图像处理功能的 ActiveX 控件
- 对数据库进行动态刷新和多表切换
- 基于 COM 的 ADO 数据库技术

超值CD，43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

Visual C++ 编程技巧典型案例解析

——图形图像处理与数据库篇

《电脑编程技巧与维护》杂志社　编著

中国电力出版社
www.infopower.com.cn

内 容 简 介

Visual C++是微软公司开发的基于Windows操作系统的编程工具，它采用一种巧妙的方法将Windows的编程复杂性封装起来，有着强大的类库支持和类改造能力以及高效率的运行速度，并为用户提供了更为友好的可视化开发环境。本书以图形图像处理编程和数据库编程为主线，贯穿了Visual C++的相关高级实现技术和理论，并且通过大量经典的实例，将知识点与开发实战紧密结合，使读者不仅可以全面掌握Visual C++的高级开发知识，而且可以了解更多Visual C++的应用技巧。

本书注重工程实践，实用性强，是软件开发人员不可多得的参考书，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——图形图像处理与数据库篇／《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 7-5083-3264-4

Ⅰ.V... Ⅱ.电... Ⅲ.①图像处理－C语言－程序设计－案例－分析②数据库系统－C语言－程序设计－案例－分析 Ⅳ.TP312

中国版本图书馆CIP数据核字（2005）第017788号

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——图形图像处理与数据库篇
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：北京铁成印刷厂
开本尺寸：185 × 260　　印　张：20.75
书　　号：ISBN 7-5083-3264-4
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，适合作为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。

全书分两章共55个实例。第1章　图形图像处理编程。该章精选了38个Visual C++在图形图像处理应用中的典型而实用的编程实例；第2章　数据库编程。该章精选了17个Visual C++在数据库应用中的典型而实用的编程实例。

本书的主要特色如下：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

第 2 章　数据库编程

第 1 章
图形图像处理编程

■实例 1
图像的数字化艺术——用 Visual C + +处理图像浅析

大家都知道 Photoshop 这款软件对于数字图像的处理功能非常强大，可是大家在使用它的过程中是否知道我们所使用的“淡化”、“噪声消除”、“模糊”等效果是如何实现的吗？本实例将带领大家用 Visual C + + 制作一个简单的图像处理软件，介绍数字图像的处理算法，从而体会数字与艺术相结合的魅力。

一、数字图像基本概念

人眼看到的人和自然界的图像都是连续的模拟图像，其形状和状态表现由图像的位置和颜色决定。色度学理论认为，任何颜色都可以由红(Red)、绿(Green)、蓝(Blue)三种基本颜色按不同的比例混合得到。而一幅图像中每一个点都可以表示成 p = (r, g, b) 这样的形式。其中 r、g、b 三种颜色取值的不同就组合产生出了不同颜色的点，所有这些点按照位置排列起来就构成了各式各样的图画，这种图像的存储方式又叫栅格结构，而我们要处理的正是这样的栅格结构的图像（也可以把这种结构的图像叫做“位图”）。彩色图像最常用的模式除了 RGB 模式还有 CMYK 模式，此外还有 HSL 和 YUV 等，我们主要以 RGB 模式为例进行介绍。

在正式编写代码前我们还要做一些准备工作，如打开图像、读取文件头结构、分配内存空间等等。这些功能都可以使用设备无关位图(DIB)类来实现。由于 Visual C + + 不带这类函数，所以我们只能自己编写相应的代码。下面直接给出一个设备无关位图类的实现，这个类的功能十分强大。

(1) 多种形式的构造函数，包括创建空 DIB、从 DDB 创建、从 DIB 句柄创建、从 DIB 数据块指针创建及从屏幕或窗口显示创建等。

(2) DIB 文件的读写操作。

(3) 从资源中装载 DIB 位图。

(4) DIB 的显示和缩放显示。

(5) 提供 DIB 的空间、颜色和格式特征等信息。

(6) DDB 与 DIB 的相互转换。

(7) DIB 格式转换。

(8) DIB 调色板操作。

(9) 能获取 DIB 位图数据的句柄。

(10) 能生成 DIB 数据的拷贝。

(11) 能直接在 DIB 上使用 GDI 操作（即在 DIB 上画图）。

二、程序清单

1. CDIB 类的头文件

```
//CDIB.h
#ifndef _CCDIB_H
#define _CCDIB_H
class CDib : public CObject
{
public:
    RGBQUAD * m_pRGB;
    BYTE * m_pData;
    UINT m_numberOfColors;
      BOOL m_valid;
    BITMAPFILEHEADER bitmapFileHeader;
    BITMAPINFOHEADER * m_pBitmapInfoHeader;
    BITMAPINFO * m_pBitmapInfo;
public:
    CDib();
    ~CDib();
     char m_fileName[256];
    char * GetFileName();
    BOOL IsValid();
    DWORD GetSize();
    UINT GetWidth();
    UINT GetHeight();
    UINT GetNumberOfColors();
    RGBQUAD * GetRGB();
    BYTE * GetData();
    BITMAPINFO * GetInfo();
      WORD PaletteSize(LPBYTE lpCDIB);
      WORD CDIBNumColors(LPBYTE lpCDIB);
    void SaveFile(const CString filename);
public:
    void LoadFile(const char * dibFileName);
      };
#endif
```

2. CDIB 类的实现文件

```
//CDIB.Cpp
#include "stdafx.h"
#include "cdib.h"
#include "windowsx.h"
#include "math.h"
#define WIDTHBYTES(bits)    (((bits) + 31) / 32 * 4)
CDib::CDib()
{
//  LoadFile();
}
CDib::~CDib()
{
```

```
    GlobalFreePtr(m_pBitmapInfo);
}
void CDib::LoadFile(const char * dibFileName)
{
    strcpy(m_fileName, dibFileName);
    CFile dibFile(m_fileName, CFile::modeRead);
    dibFile.Read((void *)&bitmapFileHeader, sizeof(BITMAPFILEHEADER));
    if (bitmapFileHeader.bfType == 0x4d42)
    {
        DWORD fileLength = dibFile.GetLength();
        DWORD size = fileLength -
                sizeof(BITMAPFILEHEADER);
        BYTE * pDib =
            (BYTE *)GlobalAllocPtr(GMEM_MOVEABLE, size);
        dibFile.Read((void *)pDib, size);
        dibFile.Close();
        m_pBitmapInfo = (BITMAPINFO *) pDib;
        m_pBitmapInfoHeader = (BITMAPINFOHEADER *) pDib;
        m_pRGB = (RGBQUAD *)(pDib +
                m_pBitmapInfoHeader->biSize);
        int m_numberOfColors = GetNumberOfColors();
        if (m_pBitmapInfoHeader->biClrUsed == 0)
            m_pBitmapInfoHeader->biClrUsed =
                    m_numberOfColors;
        DWORD colorTableSize = m_numberOfColors *
            sizeof(RGBQUAD);
        m_pData = pDib + m_pBitmapInfoHeader->biSize
            + colorTableSize;
     if (m_pRGB == (RGBQUAD *)m_pData) // No color table
                m_pRGB = NULL;
        m_pBitmapInfoHeader->biSizeImage = GetSize();
            m_valid = TRUE;
    }
    else
    {
        m_valid = FALSE;
        AfxMessageBox( "This isn't a bitmap file!" );
    }
}
BOOL CDib::IsValid()
{
    return m_valid;
}
char * CDib::GetFileName()
{
    return m_fileName;
}
UINT CDib::GetWidth()
{
    return (UINT) m_pBitmapInfoHeader->biWidth;
```

```
}
UINT CDib::GetHeight()
{
   return (UINT) m_pBitmapInfoHeader->biHeight;
}
DWORD CDib::GetSize()
{
   if (m_pBitmapInfoHeader->biSizeImage != 0)
      return m_pBitmapInfoHeader->biSizeImage;
    else
   {
      DWORD height = (DWORD) GetHeight();
      DWORD width = (DWORD) GetWidth();
      return height * width;
   }
}
UINT CDib::GetNumberOfColors()
{
    int numberOfColors;
   if ((m_pBitmapInfoHeader->biClrUsed == 0) &&
       (m_pBitmapInfoHeader->biBitCount < 9))
    {
        switch (m_pBitmapInfoHeader->biBitCount)
        {
           case 1: numberOfColors = 2; break;
           case 4: numberOfColors = 16; break;
           case 8: numberOfColors = 256;
        }
    }
   else
    numberOfColors = (int) m_pBitmapInfoHeader->biClrUsed;
   return numberOfColors;
}
BYTE * CDib::GetData()
{
   return m_pData;
}
RGBQUAD * CDib::GetRGB()
{
   return m_pRGB;
}
BITMAPINFO * CDib::GetInfo()
{
   return m_pBitmapInfo;
}
WORD CDib::PaletteSize(LPBYTE lpCDIB)
{
   return (CDIBNumColors(lpCDIB) * sizeof(RGBTRIPLE));
}
WORD CDib::CDIBNumColors(LPBYTE lpCDIB)
```

```
{
    WORD wBitCount;  // CDIB bit count
wBitCount = ((LPBITMAPCOREHEADER)lpCDIB) ->bcBitCount;
    switch (wBitCount)
    {
        case 1:
            return 2;
        case 4:
            return 16;
        case 8:
            return 256;
        default:
            return 0;
    }
}
void CDib: : SaveFile(const CString filename)
{
    BITMAPFILEHEADER    bmfHdr; // Header for Bitmap file
    LPBITMAPINFOHEADER  lpBI; // Pointer to CDIB info structure
    DWORD               dwCDIBSize;
        bmfHdr. bfType  = 0x4d42;  // "BM"
      lpBI  = (LPBITMAPINFOHEADER)m_pBitmapInfoHeader;     dwCDIBSize = * (LPDWORD)
      lpBI + PaletteSize((LPBYTE)lpBI);
      if ((lpBI ->biCompression  = = BI_RLE8) || (lpBI ->biCompression  = = BI_RLE4))
          dwCDIBSize  + = lpBI ->biSizeImage;
      else
      {
          DWORD dwBmBitsSize;  // Size of Bitmap Bits only
          dwBmBitsSize  = WIDTHBYTES((lpBI ->biWidth) * ((DWORD)lpBI ->biBitCount)) *
                lpBI ->biHeight;
          dwCDIBSize  + = dwBmBitsSize;
          lpBI ->biSizeImage  = dwBmBitsSize;
      }
    bmfHdr. bfSize  = dwCDIBSize  + sizeof(BITMAPFILEHEADER);
      bmfHdr. bfReserved1  = 0;
      bmfHdr. bfReserved2  = 0;
      bmfHdr. bfOffBits  = (DWORD)sizeof(BITMAPFILEHEADER)  + lpBI ->biSize  +
            PaletteSize((LPBYTE)lpBI);
  CFile dibFile(filename,  CFile: : modeWrite|CFile: : modeCreate);
      dibFile. Write(& bmfHdr,  sizeof(BITMAPFILEHEADER));
      dibFile. WriteHuge(lpBI,  dwCDIBSize);
      dibFile. Close();
```

三、几种算法

有了这个类以后，我们就可以非常简单地读入图像文件了，而图像文件被读入后存放在内存中一段连续的区域内，并且可以得到图像文件的起始地址。聪明的读者可能会想到，现在只要使用一些算法对这些连续的点进行相应的处理，就可以改变图像的样子了。比如，我们可以把每个点的颜色值减去 1，这样得到的就是比原图稍稍发暗的一个新图像了。

下面假设我们已经成功地读入了一张图片，并且获得了图像在内存中的指针。接下来我们对图

像进行处理。

1. 非零元素取一法

这是最简单的一个处理方法，它处理灰度图像（即图像中每个点只有一个灰度信息，范围在0～255之间），处理后的效果如图1－1所示。

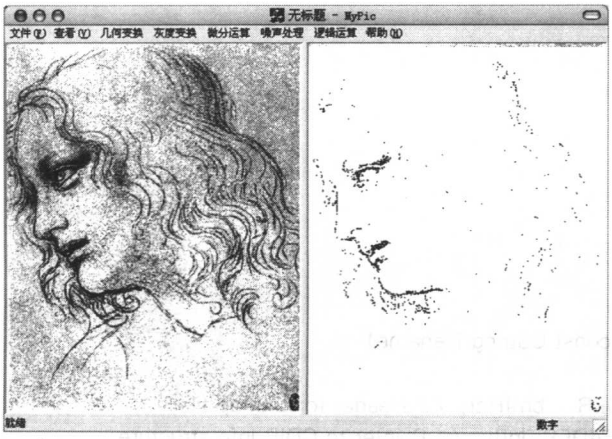

图1－1

代码清单：

```
/* = = = = = = = = = = = = = = = = = = = = = = = =
//函数名称：f0q1()
//函数类型：BOOL
//返回值：处理成功返回 TRUE;处理失败返回 FALSE
//功能：对图像用非 0 元素取 1 法进行二值化
= = = = = = = = = = = = = = = = = = = = = = = = = */
bool ChuLi::f0q1()
{
    BYTE *p_data;     //原图数据区指针
    int wide, height;     //原图长、宽
    if(m_valid)
    {
    p_data = this->GetData();     //取得原图的数据区指针
      wide = this->GetWidth();
      height = this->GetHeight();
        for(int j = 0; j < height; j++)
        for(int i = 0; i < wide; i++)     //所有像素依次循环
            {
                if(*p_data! = 0)     //若像素值不为 0
              *p_data = 255;     //将其置为 255
              p_data++;
            }
        return true;
    }
    return false;
}
```

2. 固定阈值法

对于灰度图像，把灰度小于固定阈值T的像素置为0，大于固定阈值T的像素置为255。阈值