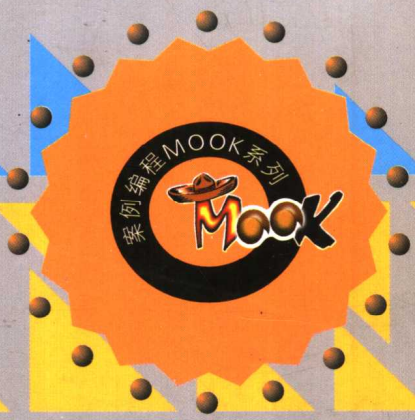
案例编程MOOK系列

Visual C/C++

编程精选集锦

关键技术精解分册



《电脑编程技巧与维护》杂志社 主编

科学出版社
www.sciencep.com

案例编程Book系列

Visual C/C++编程精选集锦

关键技术精解分册

《电脑编程技巧与维护》杂志社　主编

科学出版社

北　京

内 容 简 介

Visual C/C++ 作为功能强大的可视化应用程序开发工具，是计算机界公认的优秀应用开发工具。Microsoft 的基本类库 MFC 使得开发 Windows 应用程序变得很容易，适合作各种系统软件、应用软件、网络软件、游戏软件等开发平台。

根据 Visual C/C++ 的不同应用对象，将精选的 190 个实例共分数据库及图形图像分册、网络与通信分册、关键技术精解分册出版。本书为专家指点分册。全书本着实用第一的原则，紧紧围绕主题展开，循序渐进，由浅入深地介绍了使用 Visual C/C++ 进行应用程序开发思想方法与编程技巧。

本书的特色体现如下几点：第一，每一章都是通过一个个的实例来介绍 Visual C/C++ 应用编程方法和技巧，避免枯燥、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。第二，所选的每一个实例都是从事 Visual C/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。第三，每一个实例的程序源代码都是经过上机调试通过，给程序开发人员移植源代码带来了方便，从而加快应用编程的步伐。第四，对老版本经典实例进行点评，选取一些老版本开发环境的经典实例加以点评分析，使之能够起到触类旁通的作用。

本书适用于有一定 Visual C/C++ 应用基础的编程人员和应用开发人员，对初学 Visual C/C++ 编程的读者也有一定的参考价值。

图书在版编目（CIP）数据

Visual C/C++编程精选集锦（关键技术精解分册）/《电脑编程技巧与维护》杂志社主编. —北京：科学出版社，2003

（案例编程 MOOK 系列）

ISBN 7-03-011498-1

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（2003）第 036157 号

策划编辑：孟战龙/责任编辑：韩 洁

责任印制：吕春珉/封面设计：三函设计

科学出版社 出版

北京东黄城根北街16号

邮政编码:100717

http://www.sciencep.com

新蕾印刷厂 印刷

科学出版社发行 各地新华书店经销

*

2003 年 5 月第 一 版 开本：B5（720×1000

2003 年 5 月第一次印刷 印张：34 3/4

印数：1—5 000 字数：812 000

全套定价：150.00 元

（共三册 光盘另加 5.00 元）

（如有印装质量问题，我社负责调换〈路通〉）

前　言

Visual C/C++ 作为功能强大的可视化应用程序开发工具，是计算机界公认的优秀应用开发工具。Microsoft 的基本类库 MFC 使得开发 Windows 应用程序变得很容易，适合作各种系统软件、应用软件、网络软件、游戏软件等开发平台。

《Visual C/C++ 编程精选集锦》精选了《电脑编程技巧与维护》杂志近一二年发表的精彩编程文章，并根据读者要求，组织收入了更具价值的编程案例。《电脑编程技巧与维护》杂志是为从事电脑编程、系统应用和人员创办的专业性和实用性都很强的技术刊物，它从 1994 年创刊八年多以来，始终遵循着“实用第一，智慧密集”的办刊宗旨，紧紧跟踪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要点和技巧问题，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，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，然后溶入到自己的设计能力中去，这样无论是对于自身和整体都有莫大的提高，这正是编写这套书的初衷。

根据 Visual C/C++ 的不同应用对象，将精选的 190 个实例共分为数据库及图形图像分册、网络与通信分册、关键技术精解分册出版。关键技术精解分册，介绍一些 Visual C/C++ 开发中关键技术问题解决的实例，这些实例是经验的总结，具有一定的代表性，在此我们与更多的程序员分享这些经验。

本书为关键技术精解分册。全书每一章都本着实用第一的原则，紧紧围绕一个主题展开，循序渐进，由浅入深地介绍了使用 Visual C/C++ 进行应用程序开发思想方法与编程技巧。

本书的特色体现如下几点：第一，每一章都是通过一个个的实例来介绍 Visual C/C++ 应用编程方法和技巧，避免枯燥、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设计目标，接着介绍实现该目标的基本思想和方法，然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。第二，所选的每一个实例都是从事 Visual C/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。第三，每一个实例的程序源代码都是经过上机调试通过，给程序开发人员移植源代码带来了方便，加快应用编程的步伐。第四，对老版本经典实例进行点评，选取一些老版本开发环境的经典实例加以点评分析，使之能够起到触类旁通的作用。

本书是《电脑编程技巧与维护》资源的二次开发，浓缩了 Visual C/C++ 程序设计的精华，其目的是提升 Visual C/C++ 程序开发能力，把应用 Visual C/C++ 进行编程的心得体会、经验与大家共享。本书定位于有 Visual C/C++ 应用基础上的编程人员和应用开发人员，对初学 Visual C/C++ 编程的新手也有一定的参考价值。本书内容全面、概念清晰、层次分明，例题典型而实用，但不足甚至疏漏之处在所难免，恳请广大读者批评指正。

《电脑编程技巧与维护》杂志社

目　　录

此为试读，需要完整PDF请访问：www.ertongbook.com

实例 1

用 Visual C++ 实现在 SDI 分隔器窗口中切换视

用户界面是系统与用户之间最直接的交互界面，随着人们对 Visual C++ 的不断深入的掌握，虽然各种风格的可视化界面多种多样，对于视切换的方法也很多，本文就 SDI 分隔器窗口中多种视的切换介绍了一种简单实用的方法。

工程实现

本工程 LotvChange 为一个 SDI 窗口，主框架由一个静态的一行二列的分隔器窗口组成。左列 CleftPaneView 派生于 CformView，右列 CrightPaneView 派生于 CframeView。要实现右边窗格中的视切换，CframeView 类为此提供了条件，因为 CframeView 类可以创建 Cview 的派生类，比如 CformView。

左边窗格是一个 CFormView 派生类，其中包含一个树形控件。右边窗格 OnCreateClient 函数创建所有不同的视，并设置活动视。同时右边窗格的视也可是分隔器窗口。

右边窗格中所要切换的视(一个列表视、一个 FORM 视、一个由该列表视和该 FORM 视组成的一列两行的带分隔条的视、一个编辑框视)在添加完成后，要将其构造函数由 protected 类型更改为 public 类型，便于右边窗格视的调用，创建相应的视。

```
BOOL CRightPaneFrame:: OnCreateClient(LPCREATESTRUCT lpcs, CCreate_Context * pContext)
{
// TODO: Add your specialized code here and/or call the base class
m_pSplitterView = new CSplitterView;//在右窗口中创建带有分隔条的视
m_pSplitterView -> Create(NULL, NULL, 0L, CFrameWnd::rectDefault, this, VIEW_SPLITTER, pContext);
SetActiveView(m_pSplitterView);//并将该视设置为默认的当前活动视
    m_pSplitterView -> ShowWindow(SW_SHOW);
    m_pSplitterView -> SetDlgCtrlID(AFX_IDW_PANE_FIRST);//设置其 ID 号
    m_nCurrentViewID = VIEW_SPLITTER;
    m_pListCtrlView = new CListCtrlView;//在右窗口中创建列表视，并设置 ID((CView * )
m_pListCtrlView) -> Create(NULL, NULL, 0L, CFrameWnd::rectDefault, this, VIEW_LISTCTRL,
pContext);
    m_pListCtrlView -> ShowWindow(SW_HIDE);
    m_pListCtrlView -> SetDlgCtrlID(VIEW_LISTCTRL);
m_pEditCtrlView = new CEditCtrlView;//在右窗口中创建编辑控件视，并设置 ID((CView * )
```

```
m_pEditCtrlView) -> Create(NULL, NULL, 0L, CFrameWnd::rectDefault, this, VIEW_EDIT,
pContext);
    m_pEditCtrlView -> ShowWindow(SW_HIDE);
    m_pEditCtrlView -> SetDlgCtrlID(VIEW_EDIT);
    m_pAddfView = new AddfView;//在右窗口中创建 FORM 视,并设置 ID
    ((CView * ) m_pAddfView) -> Create(NULL, NULL, 0L, CFrameWnd::rectDefault, this,
VIEW_FORM, pContext);
    m_pAddfView -> ShowWindow(SW_HIDE);
    m_pAddfView -> SetDlgCtrlID(VIEW_FORM);
    RecalcLayout();//别忘了调用此函数
    return TRUE;//必须返回 TRUE
}
```

当选择了左边窗格树形结构中的某个选项时,将会触发相应的函数 void CLeftPaneView::OnSelchangedTree(),此函数在识别所选的选项后调用右边窗格的成员函数 SwithToView():

```
void CRightPaneFrame::SwitchToView(UINT nView)//添加的处理视图切换的函数
{
    CView * pOldActiveView = GetActiveView();
    CView * pNewActiveView = NULL;
    switch (nView)//判断所要显示的视
    {
    case VIEW_SPLITTER:
    pNewActiveView = (CView * ) m_pSplitterView;
        break;
    case VIEW_LISTCTRL:
    pNewActiveView = (CView * ) m_pListCtrlView;
    break;
    case VIEW_EDIT:
    pNewActiveView = (CView * ) m_pEditCtrlView;
    break;
    case VIEW_FORM:
    pNewActiveView = (CView * ) m_pAddfView;
            break;
    }
    if (pNewActiveView)
    {
        // 当视相同时不切换
    if (pOldActiveView = = pNewActiveView) return;
    SetActiveView(pNewActiveView); //实现切换视
```

```
    pNewActiveView->ShowWindow(SW_SHOW);
    pNewActiveView->SetDlgCtrlID(AFX_IDW_PANE_FIRST);
    pOldActiveView->ShowWindow(SW_HIDE);
    pOldActiveView->SetDlgCtrlID(m_nCurrentViewID);
    m_nCurrentViewID = nView;
        RecalcLayout();
    }
}
```

在该函数中，其主要功能是隐藏了所有的老的活动视，显示新的活动视，并设置当前窗口控件 ID，同时判断视是否真的改变，否则不切换视。（完整的程序参见工程 LotvChange）

在 SDI 分隔器中进行视切换，有一定的技巧性。因为 SDI 窗口只能在创建主框架时就创建所有的视，而且右边的窗格必须能创建 CView 的派生类，同时切换视时的处理顺序也很讲究。

（朱容波）

实例 2

Visual C++ 与 MATLAB 混合编程方法讨论

MATLAB 其强大的数据处理能力和丰富的工具箱，使得它的编程极为简单，可以极大地缩短应用程序开发周期，提高编程效率和缩短理论方案研制周期。对于纯理论方案来说，MATLAB语言是优势较多。但由于其执行效率低，对于对实时性或速度要求较高的场合来说，就不太适应了。其对底层硬件的控制能力很差，所以对于半实物仿真和偏工程化的产品来说，MATLAB 并不是一个很好的语言。对于发布软件公司来说，也希望发布的是一个可执行应用软件，而不是一个 MATLAB 源代码的产品。所以我们通过把 MATLAB 下的.m 文件转化为 Visual C 可调用动态链接库(DLL)，来达到脱离 MATLAB 环境，生成可执行文件并提高执行效率，以消除 MATLAB 不利因素。

一、MATLAB 与 C 语言混合编程方法简介

MATLAB 与 C 语言混合编程有三种方法：

1) 采用 MATLAB 与 C 的接口规范来编程。

2) 用 MATLAB 引擎来编程。

3) 用 MATLAB 下的.m 文件转化为 Visual C 可调用动态链接库(DLL)。

由于采用方法 1)和方法 2)都脱离不了 MATLAB 运行环境，这里不介绍，下文介绍方法 3)。

二、把 MATLAB 下的.m 文件转化为 Visual C 可调用的动态链接库

1. 用.m 文件创建一个 Visual C 可调用的 DLL 文件

(1) 编辑一个.m 文件

基于说明问题起见，用 MATLAB 的 edit 编了一个简单的 myfunc.m 文件，程序如下所示：

```
function y = myfunct(x,b)
if b = = 1
   x = mod(x,360) * pi/180;
   else
   x = mod(x,2 * pi);
end
x1 = linspace(0,2 * pi,100);
y1 = sin(x1);
```

```
y = interp1(x1,y1,x,'spline');
return;
```

(2) MATLAB 编译前的准备工作

对 MATLAB 编译环境进行设置。

1) 在 MATLAB 环境中运行 mex -setup,按屏幕提示要求选择编译器类型,位置等有关信息。

2) 在 MATLAB 环境中运行 mbuild -setup,设置方法与上面基本相同。

(3) 用 mcc 将 myfunc.m 转换为 matlab 可调用的 DLL

在 MATLAB 环境中运行 mcc -t -h -L C -W lib:ppp -T link:lib myfunc.m。

编译完成后,将生成如下一些文件 ppp.exp、ppp.lib、myfunc.c、ppp.c、ppp.exports、myfunc.h、ppp.dll、ppp.h。其中有用的文件有三个,分别是 ppp.h、ppp.lib、ppp.dll,它们需要被添加到 Visual C 程序中去。

(4) 用 MFC 编译一个 Visual C++6.0 可执行文件

1) 由于 dll 不能独立运行,所以要用 Visual C6.0 创建一个 EXE 可执行程序。在 Visual C6.0 中创建一个基于对话框的 MFC 工程,名为 mytest,具体过程参见一般的 Visual C 教程。在本例中 mytest 工程路径为 e:\mat2c\mytest。

2) 对 Visual C 编程环境进行设置。选择 Visual C 编译器主菜单下 Tools->options->directories,选择 Show Directories for 列表框,分别把 MATLAB 的包含文件路径(如 c:\matlab\extern\include),库文件(如 C:\matlab\extern\lib)路径添加到 Visual C 路径中去。

3) 把 ppp.h、ppp.lib、ppp.dll 三个复制到 e:\mat2c\mytest 目录下,以便 Visual C 调用 DLL 时能找到这三个文件。

4) 对对话框资源进行编辑,如图 1 所示。

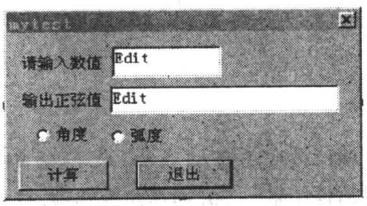


图 1

5) 引入头文件和库文件

在 mytestdlg.cpp 的头部加入一行:#include "ppp.h",在 project->settings->link 下的 object/libray modules 下加入 ppp.lib、libmx.lib、libmat.lib、libmatlb.lib、libmmfile.lib,目的是让 Visual C 能调用 ppp.dll,引入 libmx.lib、libmat.lib、libmatlb.lib 和 libmmfile.lib 的目的是让 Visual C 能调用 matlab 的数学库函数和一些功能性函数编译 Visual C 与 matlab 的代码接口和作者直接用 C 写的一些代码。

6) 对 ppp.h 做一点改动

在#include "matlab.h"语句之后,加入一行 extern "C"{,在最后一行#endif 之前加入一

此为试读,需要完整PDF请访问: www.ertongbook.com

行|。

7) 对 Visual C 与 MATLAB 接口进行编程

对“计算”按钮消息处理函数编程如下：

```
void CMytestDlg::OnOnCalculate()
{
// TODO: Add your control notification handler code here
    UpdateData(TRUE);
    pppInitialize();
    static double a[1] = {0.0};
    static double b[1] = {0.0};
      a[0] = m_din;
      b[0] = m_select + 1;
       mxArray * A = mclGetUninitializedArray();
    mxArray * B = mclGetUninitializedArray();
    mxArray * C = mclGetUninitializedArray();
    mlfAssign(&A, mlfDoubleMatrix(1, 1, a, NULL));
    mlfAssign(&B, mlfDoubleMatrix(1, 1, b, NULL));
    mlfAssign(&C, mlfMyfunc(A, B));
    double * md = mxGetPr(C);
      m_dout = md[0];
  mxDestroyArray(A);
    mxDestroyArray(B);
    mxDestroyArray(C);
      pppTerminate();
      UpdateData(FALSE);
}
```

8) 做完以上工作后，DLL 就已被成功链入 Visual C，再经 Visual C 编译器编译链接即可生成可执行文件，运行程序，在对话框中输入 30，选择角度选项，按计算按钮即可得到结果，如图 2 所示。

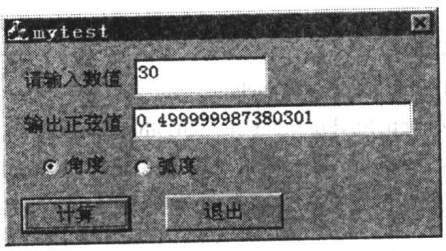


图 2

注意：libmx.lib、libmatlb.lib、libmmfile.lib、libmat.lib 文件并不是 MATLAB 自带的，MATLAB 只提供了 libmx.dll、libmatlb.dll、libmmfile.dll、libmat.dll，用户需要自己编译，把 Visual C 的 bin 目录下的 vcvars32.bat 拷贝到 C 盘根目录下，运行 msconfig 将 vcars32.bat 添加到 Auoexec.bat 中去。重新启动计算机，回到 MS-DOS 方式下，在 Visual C 的目录下 extern \ include 运行。

lib /def:libmat.def /machine:ix86 /out:libeng.lib

lib /def:libmatlb.def /machine:ix86 /out:libmatlab.lib

lib /def:libmmfile.def /machine:ix86 /out:libmmfile.lib

lib /def:libmx.def /machine:ix86 /out:libmx.lib

把生成的 libmx.lib、libmatlb.lib、libmmfile.lib、libmat.lib 文件复制到 c: \ matlab \ extern \ lib(也就是添加到 Visual C 的编译路径中去)。

本文中的文件路径可能跟读者计算机中的路径有所不同，请参照修改。

(钱云　夏祖明)

实例 3

在 MFC DLLs 中导出资源及其相关类的实现方法

在进行软件开发过程中，整个系统要分解成若干个小模块，各个模块的实现大多以动态库的形式存在。在动态库中完整地导出资源(对话框、字符串、工具条等)和相应的资源响应函数或类是必要的，尤其是在面向对象编程方法(OOP)中，类的完整导出将使动态库的使用更加方便。在一些参考文献中对于动态库的原理和 Win32 动态库介绍得比较详细，而对于 MFC DLLs 中的资源及资源类的导出介绍得很少。本文将结合一个调色板对话框类(CRisPalDlg)的导出示例对此加以说明。

一、在扩展 MFC DLLs 中导出对话框类

从扩展 MFC DLLs 中导出一个不和任何资源相关的类是很容易的。首先生成扩展 MFC 用 DLLs 程序框架，再用 Class Wizard 生成一个要导出的类(当然也可以自己定义类)，加入要在该类中实现的功能函数，最后在类定义处插入 AFX_EXT_CLASS 宏，编译即可。剩下的工作便是如何在应用程序中调用的问题了。关于动态库的调用问题可以参见参考文献。

要在扩展 MFC DLLs 中导出对话框类，可在扩展 MFC 用 DLLs 程序框架基础上用 Class Wizard 插入从 CDialog 公有继承的 CRisPalDlg 类，再在 CRisPalDlg 定义处插入 AFX_EXT_CLASS 宏。代码如下：

```
/* RisPalDlg.h */
#if ! defined(AFX_RISPALDLG_H)
#define AFX_RISPALDLG_H
class AFX_EXT_CLASS CRisPalDlg : public CDialog
{
public:
    CRisPalDlg(CWnd* pParent = NULL);
    //{{AFX_DATA(CRisPalDlg)
    enum { IDD = IDD_DIALOG_PALETTE };
    //}}AFX_DATA
    //{{AFX_VIRTUAL(CRisPalDlg)
protected:
    virtual void DoDataExchange(CDataExchange* pDX);
    //}}AFX_VIRTUAL
```

```
// Implementation
protected:
    // Generated message map functions
    //{{AFX_MSG(CRisPalDlg)
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};
#endif
```

这样编译是通不过的,因为在编译时对话框资源没有包含在对话框类的文件中。在生成的palettedialog.cpp 中有这样一行代码 # include " \ add additional includes here",在此加入 resource.h 头文件,这样即可以顺利编译通过。另建工程 Test 测试动态库。在 Test 工程中加入头文件 RisPalDlg.h,并设置好连接项。编译工程会发现,应用程序不能通过编译,并发出如下编译错误信息:

```
error C2065: 'IDD_DIALOG_PALETTE' : undeclared identifier
error C2057: expected constant expression
```

出错的原因是,在应用程序 Test 中无法对资源 IDD_DIALOG_PALETTE 进行识别。在应用程序调用动态库时,其资源搜索顺序是,首先在 EXE 中搜索,然后在 DLLs 中搜索,最后在 MFC 动态库中搜索。为了使程序能正确地获取资源,对话框的源文件作如下改动:

1) 在含有 DllMain 函数的 PaletteDLL.cpp 文件中实现获取本模块句柄的函数。

```
HMODULE GetDllModule();
static AFX_EXTENSION_MODULE PaletteDLL = { NULL, NULL };
HMODULE GetDllModule()
{
    return PaletteDLL.hModule;
}
```

2) 对话框头文件 RisPalDlg.h 中,要把含有资源号的无名枚举注销,用同名 UINT 型变量 IDD 代替,并显示出声明对话框的析构函数和保存资源句柄用的句柄变量,源代码如下:

```
#if ! defined(AFX_RISPALDLG_H)
#define AFX_RISPALDLG_H
class AFX_EXT_CLASS CRisPalDlg : public CDialog
{
protected :
HINSTANCE m_hInst;//用于保存资源句柄
public:
    CRisPalDlg(CWnd * pParent = NULL);
    virtual ~CRisPalDlg();//析构函数
//enum { IDD = IDD_DIALOG_PALETTE };//注释掉无名枚举
    UINT IDD;//声明对话框资源 ID
```

```
protected:
virtual void DoDataExchange(CDataExchange * pDX);
protected:
      DECLARE_MESSAGE_MAP()
};
#endif
```

3) 在对话框实现文件 RisPalDlg.cpp 中，首先声明外部函数 GeDllModule，在构造函数中把 CRisPalDlg::IDD 直接用 IDD_DIALOG_PALETTE 代替，在其中对成员变量 IDD 进行赋值，并在构造函数中改变资源搜索顺序，使搜索顺序变为 DLLs->EXE->MFC DLLs；最后在析构函数中恢复正常的搜索顺序，源代码如下：

```
#include "stdafx.h"
#include "resource.h"
#include "RisPalDlg.h"
extern HMODULE GetDllModule();//动态库工程中的获取本模块句柄的函数
//////////////////////////////////
CRisPalDlg::CRisPalDlg(CWnd * pParent /* = NULL */)
: CDialog(/* CRisPalDlg::IDD */IDD_DIALOG_PALETTE, pParent)
{
IDD = IDD_DIALOG_PALETTE;//赋值 IDD 使 CRisPalDlg::IDD 仍然可用
//改变资源搜索顺序
m_hInst = AfxGetResourceHandle();//获取执文件的资源句柄
AfxSetResourceHandle(GetDllModule());///获 DLL 的资源句柄，改变资源搜索顺序
    }
CRisPalDlg::~CRisPalDlg()//析构函数
{
  AfxSetResourceHandle(m_hInst);//恢复原有资源搜索顺序
}
void CRisPalDlg::DoDataExchange(CDataExchange * pDX)
{
   CDialog::DoDataExchange(pDX);
}
BEGIN_MESSAGE_MAP(CRisPalDlg, CDialog)
END_MESSAGE_MAP()
```

上面的修改主要是避免在头文件中出现资源号，并在对话框类的构造函数和析构函数中实现对资源搜索顺序的变换。

经过上面三步修改，程序便能顺利通过编译，并且有效地防止了主程序资源与动态库中资源重名带来运行错误。

二、方法的改进

上面的方法是对单个的对话框导出而作的具体修改，在一个动态库中往往要导出多个对话框类。因此可以从 CDialog 类继承生成一抽象基类 CDllDialog，再让要导出的对话框类从此类继承。CDllDialog 类的实现方法与 CRisPalDlg 相似，在此不加详述。

此外，对上面的方法还可以这样改进：

1. 实现一简单的类 CDllStatus

```
class CDllStatus
{
    private:
        HINSTANCE m_hInst;
    public:
CDllStatus(HMODULE hModule)
        {
m_hInst = AfxGetResourceHandle();//获取执文件的资源句柄
    AfxSetResourceHandle(hModule);//改变资源搜索顺序
}
~CDllStatus()
{
    AfxSetResourceHandle(m_hInst);//改变资源搜索顺序
}
};
```

2. 设置导出类

如果导出类为模态对话框，则可以在导出类中重载 DoModal 函数，在 DoModal 函数中声明有 CDllStatus 变量即可，也可以在导出类中声明 CDllStatus 的成员变量。对于非模态对话框则只能在导出类中声明成员变量。

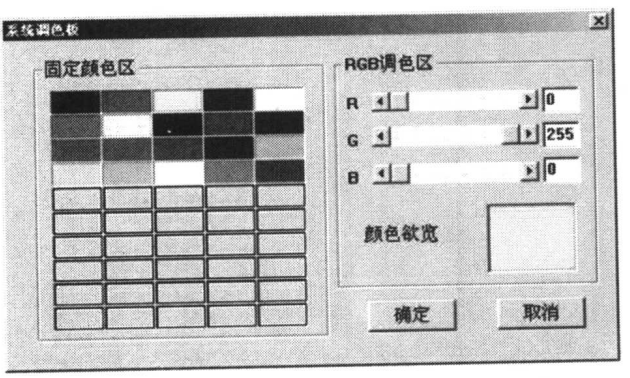


图 3　用动态库导出实现的调色板

此为试读，需要完整PDF请访问：www.ertongbook.com