http://www.pptph.com.cn

Visual C++

编程疑难详解

钱新贤　杨猛　程兆炜　张少东　等编著

人民邮电出版社

高手指路系列丛书

Visual C++编程疑难详解

钱新贤　杨　猛　程兆炜　张少东　等编著

人民邮电出版社

内容提要

Visual C++是 Microsoft 推出的一个功能强大的可视化应用程序开发工具，是计算机界公认的最优秀的专业化应用开发工具之一。本书内容主要介绍 Visual C++程序设计中经常碰到的一些疑难问题的解决方法以及相关技巧。

本书从实际应用程序设计的角度出发，以解决编程中常见的实际问题为书中内容的核心。全书内容分为 16 个专题，全面、深入、详实地介绍了 Visual C++编程的思路、方法、手段和技巧，对每个问题都进行了细致的描述和深入浅出的剖析。另外，书中还给出了许多程序源代码，并给出了相关的中文说明和注释，因此，本书具有较强的实用价值，可供广大编程人员在工作中参考。

高手指路系列丛书

Visual C++ 编程疑难详解

◆ 编　　著　钱新贤　杨　猛　程光炜　张少东　等

责任编辑　王晓明

◆ 人民邮电出版社出版发行　北京市崇文区夕照寺街 14 号

邮编　100061　　电子函件　315@pptph.com.cn

网址　http://www.pptph.com.cn

北京汉魂图文设计有限公司制作

北京朝阳展望印刷厂印刷

新华书店总店北京发行所经销

◆ 开本：787×1092　1/16

印张：29

字数：731 千字　　2000 年 7 月第 1 版

印数：1－6 000 册　　2000 年 7 月北京第 1 次印刷

ISBN 7-115-08619-2/TP·1698

定价：43.00 元

前言

Visual C++是 Microsoft 推出的一个功能强大的可视化应用程序开发工具，它在计算机领域中被公认为是最优秀的专业化应用开发工具之一，目前，全世界有超过 150 万的专业程序员在利用它进行编程工作。Visual C++作为一个集成开发工具，为编程工作者提供了程序框架代码自动生成和可视化的资源编辑功能，从而使编程工作变得更为简单。由于 Microsoft 为 Visual C++ 提供了强大的基本类库 MFC(Microsoft Fundation Classes)，因此确立了 Visual C++在开发语言平台上的领先地位，它真正把 Windows 应用程序开发带入了一个面向对象的时代。

Visual C++不仅仅是程序设计语言，而且也还是一个非常全面的应用程序开发环境，使用它可以开发出具有专业水平的 Windows 应用程序。要想充分利用 Visual C++的优势进行程序开发，必须首先理解 C++程序设计语言的规范，这样就可以充分了解 Microsoft 基本类库(MFC)的体系结构。MFC 体系结构包容了 Windows API 中的用户界面部分，并使程序员能够很容易地以面向对象的方式开发 Windows 应用程序。这种体系结构适用于所有版本的 Windows 系统并彼此兼容，因此，使用 MFC 所建立的源程序是完全可移植的。

使用过 Windows API 编制 Windows 应用程序的设计人员会有这样的体会：即使是开发一个简单的 Windows 应用程序也需要对 Windows 的编程原理有很深刻的认识，同时也要手工编写冗长的代码。因为程序的出错率几乎是随着代码长度的增加呈几何级数增长的，而且当程序容量逐渐膨胀的时候，调试程序会变得越来越困难，所以传统的 Windows 结构化程序设计需要程序开发人员有极大的耐心和丰富的编程经验。

Visual C++中引入了微软定义的基本类库(MFC)后，便使 Windows 程序设计彻底实现了模板化，从而大大降低了程序设计的复杂性。MFC 中包含了许多微软公司已经定义好的程序开发过程中最常用到的对象。我们知道，虽然我们要编写的程序在功能上是千差万别的，但从本质上来讲，都可以化归为用户界面的设计、对文件的操作、多媒体的使用、数据库的访问等等一些最主要的大类，这一点正是微软提供 MFC 类库最重要的原因。MFC 类库具有很好的扩展性，在进行程序设计的时候，如果类库中的某个对象能完成所需要的功能，那么我们只要简单地调用已有对象的方法就可以了。用户还可以利用面向对象技术中很重要的“继承”方法，从类库中的已有对象派生出自己所需要的对象。派生出来的对象除了具有类库中的对象的特性和功能之外，还可以由用户自己根据需要加上所需的特性和方法，成为一个更有专用特点、功能更为强大

的对象。当然，也可以在程序中创建全新的对象，并根据需要不断完善对象的功能。正是由于 MFC 编程方法充分利用了面向对象技术的优点，因此它使得用户编程时不必过多地在对象方法的实现细节上投入大量精力。同时类库中的各种对象的强大功能足以完成程序中的绝大部分所需功能。MFC 的应用使得程序员在编制应用程序时所需要编写的代码大为减少，并有力地保证了程序具有良好的可调试性。在 Visual C++程序设计中，对 MFC 类库中的对象进行派生是一种普遍做法，本书中将给出一些生动的例子来剖析这其中的奥妙。

但是，需要指出的是，MFC 类库编程并不是 Visual C++程序设计的全部内容。在很多场合，对 Windows API 的调用还是很必要的，或者说可以带来更高的效率。因此，本书也将把对 Windows API 的一些调用窍门融合到具体的范例中去。

本书的读者对象主要是具有一定编程基础的程序编制人员，因此，本书在编写中把着眼点放在应用程序的编制上，以解决编程中的常见问题为主要目的。书中将内容分为 16 个专题，全面、深入、详实地介绍 VC++编程的方法和技巧，对每个问题都作了细致的描述和深入浅出的剖析，因此，本书具有较强的实用性，可为广大编程人员在工作中提供有益的帮助。

本书主要由钱新贤、杨猛、程兆炜、张少东编写，另外，参与本书编写和资料整理的人员还有：贺晓天、黄敏、林云、林立、朱红尘、刘斌、刘芳、刘孟川、吴天远、钱雨、张毅、张泽宏、张宏林、张行天、韦略、高峰、肖楚寒、严飞、严清华、吕杰、范才全、范继德、陈新、陈文远、谢云、周海平、周新、黄志刚、黄宏威、傅明等。

由于作者水平有限，很难把 Visual C++编程技术全部掌握得十分精通，因此书中缺陷与偏颇之处在所难免，恳请广大读者批评指正。

作者

2000 年 4 月

目录

第一章

文档—视图—框架体系

热点透视

在文档/视图结构里，文档是一个应用程序数据基本元素的集合，它构成应用程序所使用的数据单元，并提供了管理和维护数据的手段。文档是一种数据源，数据源有很多种，最常见的是磁盘文件。文档对象负责管理来自所有数据源的数据。视图是数据的用户窗口，为用户提供了文档数据的可视化显示，它把文档的部分或全部内容在窗口中显示出来。视图给用户提供了一个与文档中的数据交互的界面，它把用户的输入转化为对文档中数据的操作。每个文档都会有一个或多个视图显示。

在本章中，我们将介绍如何在 SDI 中实现多视图及如何在打开文件对话框中选择多个文件等技巧和方法。

如何禁止在程序开始运行时新建一个文档

遇到难题

Visual C++的应用程序在开始运行后会自动创建一个空文档，如果不想在一开始运行时就创建文档，该如何处理呢？

钥匙在此

Visual C++是通过在初始化时传递一个命令行参数 cmdInfo 对应用程序的启动进行控制的，所以要想禁止空文档的创建，只要在初始化时传递合适的参数就可以了。

跟我来

在应用程序的初始化函数 InitInstance 中添加如下代码：

```
CCommandLineInfo cmdInfo;
    cmdInfo.m_nShellCommand =CCommandLineInfo::FileNew;
ParseCommandLine(cmdInfo);
```

这样，在应用程序开始运行时便不会自动创建空文档了。

如何在单文档应用中实现多视图

遇到难题

在一些情况下，应用程序有许多信息要显示，当在一个窗口内显示不下时，Visual C++提供的 MDI 应用程序框架可以解决这个问题，但如果应用程序只是对单个文档，用多种方式表现，则采用单文档更合适。那么，如何在单文档应用程序中实现多视图呢？

钥匙在此

要达到在单文档应用程序中实现多个视图的目的，必须对 CView 的派生类做小小的改动。缺省情况下，派生类的构造函数是 protected 类型的，必须将它改为 public 类型。因为，我们将会动态生成派生视图类并切换活动的视图。也可以一次生成所有的视图，但这会造成系统资源的极大浪费。下面是对 CMainView 所做的修改：

```
class CMainView : public CFormView
{
protected:      // 缺省的类型为保护类型
public:   // 修改为公共类型
 CMainView();
  DECLARE_DYNCREATE(CMainView)
  ...
  ...
}
```

用来切换视图的代码位于 CFrame 的派生类中，因为在单文档应用程序中，CFrame 的派生类控制着所有的视图。而且，它允许用户通过消息处理进行视图的切换。进行视图切换的原理很简单，只是断开和旧视图的连接并把它删除，创建一个新视图，将文档和它相连，设置几个标志，然后显示新视图。下面是进行视图切换的代码：

```
void CMainFrame::SwitchToForm(int nForm)
{
 CView* pOldActiveView = GetActiveView();   // 保存旧视图
 CView* pNewActiveView = (CView*)GetDlgItem(nForm);   // 取得新视图
 if (pNewActiveView == NULL)
 {
  switch(nForm)
// 这些 ID 是对话框的标识符，但也可以用其他的标识
  {
   case IDD_MULTISCREEN_FORM1:
    pNewActiveView = (CView*)new CMainView;
    break;
   case IDD_MULTISCREEN_FORM2:
    pNewActiveView = (CView*)new CView2;
    break;
   case IDD_MULTISCREEN_FORM3:
    pNewActiveView = (CView*)new CView3;
    break;
   case IDD_MULTISCREEN_FORM4:
    pNewActiveView = (CView*)new CView4;
```

```
            break;
        }
        CCreateContext context;          // 将文档和视图相连
        context.m_pCurrentDoc = pOldActiveView->GetDocument();
        pNewActiveView->Create(NULL, NULL, 0L, CFrameWnd::rectDefault, this, nForm,
&context);
        pNewActiveView->OnInitialUpdate();
      }
    SetActiveView(pNewActiveView);      // 改变活动的视图
     pNewActiveView->ShowWindow(SW_SHOW);// 显示新的视图
     pOldActiveView->ShowWindow(SW_HIDE);  // 隐藏旧的视图

     ::SetWindowWord(pNewActiveView->m_hWnd,  GWL_ID,  AFX_IDW_PANE_FIRST);
RecalcLayout();          //   调整框架窗口
     delete  pOldActiveView;  //   删除旧视图
    }
```

下面，我们要做的就是添加几个菜单项和相应的消息处理函数，在这些消息处理函数中调用上面的函数。当然，我们还必须定义几个视类。

跟我来

下面是一个单文档对应四个视图的例子。

1. 首先，我们利用资源编辑器新建四个对话框，在对话框中添加需要的控件，各个对话框的 ID 分别为：IDD_MULTISCREEN_FORM1、IDD_MULTISCREEN_FORM2、IDD_MULTISCREEN_FORM3 和 IDD_MULTISCREEN_FORM4。

2. 为每个对话框创建一个 CFormView 类的派生类，分别是：CView1、CView2、CView3 和 CView4。

3. 在应用程序的初始化文件中，用 CView1 替换原先的视图。这样程序启动后的缺省显示的视图是 CView1。修改后的部分代码如下：

```
CSingleDocTemplate* pDocTemplate;
pDocTemplate = new CsingleDocTemplate(
    IDR_MAINFRAME,
        RUNTIME_CLASS(CDoc),
        RUNTIME_CLASS(CMainFrame),          // main SDI frame window
        RUNTIME_CLASS(CMainView));
    AddDocTemplate(pDocTemplate);
```

4. 利用资源编辑器，添加几个菜单项，并添加相应的消息处理函数。例如：

```
void CMainFrame::OnForm1()
{
```

```
    SwitchToForm(IDD_MULTISCREEN_FORM);
}
```

如何使用初始化文件来保存和恢复程序的运行状态信息

遇到难题

一个好的应用程序应该能够恢复上次运行时主窗口的状态，这样用户就不需要在每次进入应用程序时都重新设置窗口的大小和位置。在 Visual C++中有没有方便的方法可以在程序退出时保存当前运行窗口的信息，并在下次启动时恢复这些信息呢？

钥匙在此

保存和恢复窗口的运行信息并不难，Visual C++提供了两对方便的函数可供使用。它们分别是：WriteProfileInt、LoadProfileInt 和 WriteProfileString、LoadProfileString。WriteProfileInt 和 WriteProfileString 用于向注册表或应用程序的.INI 文件中写入信息，LodProfileInt 和 LoadProfileString 则用于从注册表或.INI 文件中读取信息。

跟我来

1. 当应用程序关闭时，将框架窗口的状态：最小化、最大化或窗口的大小以及状态条的位置等信息保存在注册表中。

利用 Class Wizard，为 CMainFrame 添加 WM_CLOSE 消息的处理函数。

```
void CMainFrame::OnClose()
{
    // 添加自己的消息处理过程
    WINDOWPLACEMENT WndStatus;
    WndStatus.length = sizeof(WINDOWPLACEMENT);
    GetWindowPlacement(&WndStatus);
    AfxGetApp()->WriteProfileInt("WNDSTATUS","FLAG",WndStatus.flags);
    AfxGetApp()->WriteProfileInt("WNDSTATUS","SHOWCMD",WndStatus.showCmd);
```

```
    AfxGetApp()->WriteProfileInt("WNDSTATUS", "LEFT",
    WndStatus.rcNormalPosition.left);
    AfxGetApp()->WriteProfileInt("WNDSTATUS", "TOP",
    WndStatus.rcNormalPosition.top);
    AfxGetApp()->WriteProfileInt("WNDSTATUS", "RIGHT",
    WndStatus.rcNormalPosition.right);
    AfxGetApp()->WriteProfileInt("WNDSTATUS", "BOTTOM",
    WndStatus.rcNormalPosition.bottom);
    SaveBarState(AfxGetApp()->m_pszProfileName);
    CFrameWnd::OnClose();
```

2. 当应用程序重启时，读取注册表信息，并重新设置框架窗口的位置和大小等信息。

给 CMainFrame 添加一个布尔类型的成员变量 m_bFirst，并利用 Class Wizard，为 CMainFrame 添加 ActivateFrame 消息的处理函数。

```
void CMainFrame::ActivateFrame(int nCmdShow)
{
if(m_bFirst)//新添加的布尔变量，用于表示是否是初次激活窗口
{
        m_bFirst=FALSE;
        WINDOWPLACEMENT WndStatus;
        WndStatus.length = sizeof(WINDOWPLACEMENT);
        CRect rect;
        rect.left = AfxGetApp()->GetProfileInt("WNDSTATUS", "LEFT", 100);
        rect.top = AfxGetApp()->GetProfileInt("WNDSTATUS", "TOP", 100);
        rect.right = AfxGetApp()->GetProfileInt("WNDSTATUS", "RIGHT", 500);
        rect.bottom = AfxGetApp()->GetProfileInt("WNDSTATUS", "BOTTOM", 400);
        WndStatus.rcNormalPosition = rect;
        WndStatus.flags = AfxGetApp()->GetProfileInt("WNDSTATUS", "FLAG",0);
 nCmdShow=AfxGetApp()->GetProfileInt("WNDSTATUS",
"SHOWCMD",SW_SHOW);
        WndStatus.showCmd = nCmdShow;
        WndStatus.ptMinPosition = CPoint(0,0);
        WndStatus.ptMaxPosition = CPoint(-GetSystemMetrics(SM_CXBORDER),
GetSystemMetrics(SM_CYBORDER));
        LoadBarState(AfxGetApp()->m_pszProfileName);
 SetWindowPlacement(&WndStatus);
    }
    CFrameWnd::ActivateFrame(nCmdShow);
}
```

如何打开最近使用的文件

遇到难题

怎样才能让应用程序一启动就打开最近使用的文件？

钥匙在此

Visual C++的应用程序通过 App 的 InitInstance 函数进行初始化，生成新文档和打开文件等操作都可以在此函数中实现。要让应用程序一启动就打开最近使用的文档，只需要将下面代码添加到 App 的 InitInstance 函数中，放在 ParseCommandLine 和 ProcessShellCommand 之间。

```
//==================  下面是插入的代码==============================
// 如果在命令行上没有指定文件名，则打开最近使用的文件
if ( ! cmdInfo.m_strFileName.GetLength() )
{
CString strFileName;
    if (m_pRecentFileList->GetDisplayName(strFileName, 0, "", 0, true))
{
        cmdInfo.m_strFileName = strFileName;
cmdInfo.m_nShellCommand =CCommandLineInfo::FileOpen;
}
}
// ================    插入代码结束  ==============================
```

跟我来

当最近使用的文件名太长时，上面的代码中会发生错误。AbbreviateName 会将目录名称截断成类似下面的方式：

C:\some path\...\file.txt

这种截断处理的方式，对于显示窗口的标题比较合适，但在此处却会导致文件打不开。所以，更好的办法是用下面的代码替换上面的代码：

```
    if (!cmdInfo.m_strFileName.GetLength())
    {
```

```
        if (m_pRecentFileList->m_nSize > 0 &&
            !m_pRecentFileList->m_arrNames[0].IsEmpty())
        {
            cmdInfo.m_strFileName = m_pRecentFileList->m_arrNames[0];
            cmdInfo.m_nShellCommand = CCommandLineInfo::FileOpen;
        }
    }
```

上面的方法针对的是单文档应用程序，要在多文档应用程序中打开最近使用的文档，则可以在 InitInstance 中添加如下的代码：

```
int nMaxFiles;
nMaxFiles = MAX_MRUFILES;
if(cmdInfo.m_strFileName.IsEmpty()
{
        for(int i=0;(i < m_pRecentFileList->GetSize() && i < n);i++)
{
        CString strFileName(m_pRecentFileList->m_arrNames[i]);
        if(strFileName.IsEmpty())
return;
        OpenDocumentFile(strFileName);
}
}
```

如何改变视图的背景色

遇到难题

Visual C++提供的 FormView 是非常方便的，但背景总是灰灰的，缺乏生气。怎样才能改变 FormView 的背景色，从而定义自己所喜欢的颜色？

钥匙在此

Visual C++的窗口一般都会接收一个 WM_ERASEBGRN 消息，窗口通过响应这个消息，

对背景区域进行刷新。但在 Class Wizard 中，我们却看不到 FormView 有这个消息，所以，必须在我们的视图类中手动添加对这个消息的响应。在下面的实例中，我们通过响应 CMyFormView 的 WM_ERASEBGRD 消息，将视图的背景色设置成了白色。如图 1－1 所示：

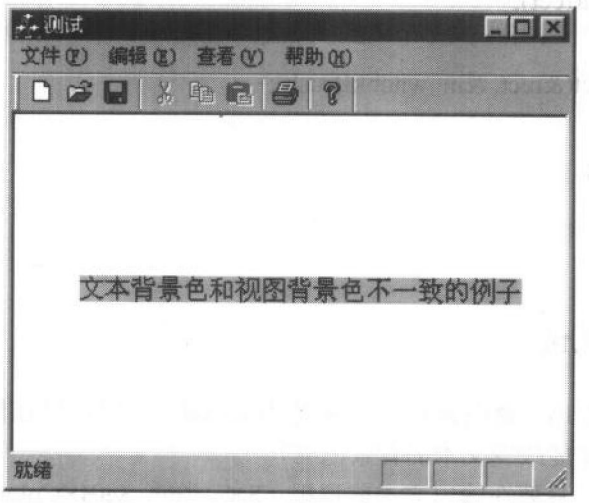

图 1－1　改变视图的背景色

跟我来

假设一个基于 FormView 的应用程序的视图类的名称为 CMyFormView，要改变视图的背景区域，请按下面的步骤操作：

1. 在 MyFormView.h 中添加下面的代码：

afx_msg BOOL OnEraseBkgnd(CDC* pDC);

请将该行代码添加在 //}}AFX_MSG 行的前面。

2. 在 MyFormView.h 中添加两个成员变量：

```
COLORREF m_crBackground;
CBrush m_wndbkBrush;
```

其中，m_crBackground 是颜色值，用来创建背景刷子；m_wndbkBrush 是填充背景区域的刷子。

3. 在 MyFormView.cpp 中，添加消息映射。将下面的代码添加到消息映射的 //}}AFX_MSG_MAP 行之前：

ON_WM_ERASEBKGND()

4. 在 CMyFormView 的构造函数中，添加下面的代码，对刷子进行初始化：

```
m_crBackground = RGB(0,0,255);
m_wndbkBrush.CreateSolidBrush(m_crBackground);
```

5. 在 MyFormView.cpp 中，添加下面的代码：

BOOL CMyFormView::OnEraseBkgnd(CDC* pDC)