

**MFC中四种常用绘图类的用法探究**

**软件工程Ⅱ大作业 参考题目4.15 在MFC中类CPen、CColorDialog、CFontDialog、CPrintDialog的用法**



作者：程嘉梁

学号：2016013225

手机号：15652667706

Email：chengjl16@163.com

日期：2017/05/25

目录

[MFC中四种常用绘图类的用法探究 0](#_Toc483508459)

[一、MFC中CPen类的介绍及用法 3](#_Toc483508460)

[1、CPen类的简介 3](#_Toc483508461)

[2、CPen类的成员简述 3](#_Toc483508462)

[3、CPen类的成员函数的用法详解 3](#_Toc483508463)

[3.1CPen::CPen函数 3](#_Toc483508464)

[3.2CPen::CreatePen 5](#_Toc483508465)

[3.3CPen::CreatePenIndirect 6](#_Toc483508466)

[3.4CPen::FromHandle 7](#_Toc483508467)

[3.5CPen::GetExtLogPen 7](#_Toc483508468)

[3.6CPen::GetLogPen 8](#_Toc483508469)

[3.7CPen :: operator HPEN 8](#_Toc483508470)

[4、例程：应用CPen类函数画一条红色虚线。 9](#_Toc483508471)

[二、MFC中CColorDialog类的介绍及用法 10](#_Toc483508472)

[1、CColorDialog类的简介 10](#_Toc483508473)

[2、CColorDialog类的成员简介 10](#_Toc483508474)

[3、CColorDialog类的成员的用法详解 10](#_Toc483508475)

[3.1CColorDialog::CColorDialog 10](#_Toc483508476)

[3.2CColorDialog::DoModal 11](#_Toc483508477)

[3.3CColorDialog::GetColor 12](#_Toc483508478)

[3.4CColorDialog::GetSavedCustomColors 12](#_Toc483508479)

[3.6CColorDialog::OnColorOK 15](#_Toc483508480)

[3.7CColorDialog :: SetCurrentColor 17](#_Toc483508481)

[4、例程：应用CColorDialog类，从(100,100)向(600,100)绘制一条宽度为10像素的直线。颜色由用户自行选择，初始值为红色，且最终不能为黑色。 18](#_Toc483508482)

[三、MFC中CFontDialog类的介绍及用法 19](#_Toc483508483)

[1、CFontDialog类的简介 19](#_Toc483508484)

[2、CFontDialog类的成员简介 19](#_Toc483508485)

[3、CFontDialog类的成员的用法详解3.1CFontDialog::CFontDialog 19](#_Toc483508486)

[3.2CFontDialog::DoModal 21](#_Toc483508487)

[3.3CFontDialog::GetCharFormat 21](#_Toc483508488)

[3.4CFontDialog::GetColor 21](#_Toc483508489)

[3.5CFontDialog::GetCurrentFont 22](#_Toc483508490)

[3.6CFontDialog::GetFaceName 22](#_Toc483508491)

[3.7CFontDialog::GetSize 23](#_Toc483508492)

[3.8CFontDialog::GetStyleName 23](#_Toc483508493)

[3.9CFontDialog::GetWeight 24](#_Toc483508494)

[3.10CFontDialog::IsBold 24](#_Toc483508495)

[3.11CFontDialog::IsItalic 25](#_Toc483508496)

[3.12CFontDialog::IsStrikeOut 25](#_Toc483508497)

[3.13CFontDialog::IsUnderLine 26](#_Toc483508498)

[3.14CFontDialog::m\_cf 26](#_Toc483508499)

[4、例程：应用CFontDialog类，让用户自行选择一种字体，输出“Good programmers have sailed!”。 27](#_Toc483508500)

[四、MFC中CPrintDialog类的介绍及用法 29](#_Toc483508501)

[1、CPrintDialog类的简介 29](#_Toc483508502)

[2、CPrintDialog类的成员简介 29](#_Toc483508503)

[3、CPrintDialog类的成员的用法详解 30](#_Toc483508504)

[3.1CPrintDialog::CPrintDialog 30](#_Toc483508505)

[3.2CPrintDialog::CreatePrinterDC 30](#_Toc483508506)

[3.3CPrintDialog::DoModal 31](#_Toc483508507)

[3.4CPrintDialog::GetCopies 32](#_Toc483508508)

[3.5CPrintDialog::GetDefaults 32](#_Toc483508509)

[3.6CPrintDialog::GetDeviceName 33](#_Toc483508510)

[3.7CPrintDialog::GetDevMode 34](#_Toc483508511)

[3.8CPrintDialog::GetDriverName 34](#_Toc483508512)

[3.9CPrintDialog::GetFromPage 34](#_Toc483508513)

[3.10CPrintDialog::GetPortName 34](#_Toc483508514)

[3.11CPrintDialog::GetPrinterDC 35](#_Toc483508515)

[3.12CPrintDialog::GetToPage 35](#_Toc483508516)

[3.13CPrintDialog::m\_pd 35](#_Toc483508517)

[3.14CPrintDialog::PrintAll 37](#_Toc483508518)

[3.15CPrintDialog::PrintCollate 37](#_Toc483508519)

[3.16CPrintDialog::PrintRange 38](#_Toc483508520)

[3.17CPrintDialog::PrintSelection 39](#_Toc483508521)

[五、资料来源 40](#_Toc483508522)

[1、CPen类参考资料 40](#_Toc483508523)

[2、百度百科-CPen 40](#_Toc483508524)

[3、CSDN-CPen类 40](#_Toc483508525)

[2、CColorDialog类参考资料 40](#_Toc483508526)

[3、CFontDialog类参考资料 40](#_Toc483508527)

[4、CPrintDialog类参考资料 40](#_Toc483508528)

# 一、MFC中CPen类的介绍及用法

## 1、CPen类的简介

CPen画笔是一种用来画线及绘制有形边框的工具，用户可以指定它的颜色及厚度，并且可以根据用途与个人喜好绘制不同形状的线。

CPen类，该类封装了Windows图形设备接口（GDI）画笔，主要通过构造函数来创建绘图对象。

CPen类的继承层次结构为CObject->CGdiObject->Cpen。

使用CPen类时需调用头文件afxwin.h。

## 2、CPen类的成员简述

|  |  |
| --- | --- |
| 名称 | 功能描述 |
| CPen::CPen | 构造一个CPen对象。 |
| CPen::CreatePen | 创建具有指定样式、宽度和画笔属性的画笔，并将其赋给CPen对象。 |
| CPen::CreatePenIndirect | 创建具有LOGPEN结构中给出的样式、宽度和颜色的画笔，并将其赋给CPen对象。 |
| CPen::FromHandle | CPen给定一个窗口时,返回一个指向对象的指针HPEN。 |
| CPen::GetExtLogPen | 获得EXTLOGPEN的底层结构。 |
| CPen::GetLogPen | 获得LOGPEN的底层结构。 |
| CPen::operator HPEN | 返回附加到CPen对象的Windows句柄。 |

## 3、CPen类的成员函数的用法详解

### 3.1 CPen::CPen函数

CPen类重载了三个构造函数。

CPen();

CPen(int nPenStyle,

int nWidth,

COLORREF crColor);

CPen(int nPenStyle,

int nWidth,

const LOGBRUSH\* pLogBrush,

int nStyleCount = 0,

const DWORD\* lpStyle = NULL);

函数功能：构造一个CPen类对象。

·第一个构造函数不含任何参数，它构造的是一个未初始化的CPen对象。

·第二个构造函数带有3个参数，分别对画笔的线形，线宽和颜色进行了初始化。

参数nPenStyle：指定画笔的风格（样式），也就是画笔的线形。

|  |  |
| --- | --- |
| 参数nPenStyle的值 | |
| 参数值 | 注释 |
| **PS\_SOLID** | 创建一支实心的笔。 |
| **PS\_DASH** | 创建虚线笔。仅当笔宽度为1或更小时才有效，以设备为单位。 |
| **PS\_DOT** | 创建一个虚线笔。仅当笔宽度为1或更小时才有效，以设备为单位。 |
| **PS\_DASHDOT** | 用交替的虚线和点创建笔。仅当笔宽度为1或更小时才有效，以设备为单位。 |
| **PS\_DASHDOTDOT** | 用交替的虚线和双点创建笔。仅当笔宽度为1或更小时才有效，以设备为单位。 |
| **PS\_NULL** | 创建一个空笔。 |
| **PS\_INSIDEFRAME** | 创建一个内框线画笔，该画笔可以在Windows GDI输出函数定义的矩形边界所生成的封闭状的边框内绘制直线。 |

参数nWidth：确定了画线的宽度。

参数crColor：包含了一个画笔所具有的RGB颜色。

·第三个构造函数带有5个参数，现一一介绍。

参数nPenStyle：功能同上。除了具有上一个构造函数中介绍的参数值外，还增加了如下的参数值。

|  |  |
| --- | --- |
| 新增的nPenStyle参数值 | |
| 参数值 | 注释 |
| **PS\_GEOMETRIC** | 创建一个几何画笔。 |
| **PS\_COSMETIC** | 创建一个装饰画笔。 |
| **PS\_ALTERNATE** | 创建一个设置其他像素的画笔。（此款式仅适用于装饰画笔）。 |
| **PS\_USERSTYLE** | 创建一个使用用户提供的样式数组的画笔。 |
| **PS\_ENDCAP\_ROUND** | 封顶是圆形的。 |
| **PS\_ENDCAP\_SQUARE** | 封顶是方形的。 |
| **PS\_ENDCAP\_FLAT** | 封顶是平的。 |
| **PS\_JOIN\_BEVEL** | 成尖角连接。 |
| **PS\_JOIN\_MITER** | 通过SetMiterLimit函数设置的当前极限值范围内斜接;否则，成尖角连接。 |
| **PS\_JOIN\_ROUND** | 成圆角连接。 |

参数nWidth：指定笔的宽度。

注：nWidth对于两个构造函数的区别：

对于第二个构造函数，如果此值为0，则不管映射模式如何，设备单位的宽度始终为1像素。而对于第三个构造函数，如果nPenStyle是PS\_GEOMETRIC，则以逻辑单位给出宽度。如果nPenStyle是PS\_COSMETIC，则宽度必须设置为1。

参数pLogBrush：指向一个LOGBRUSH结构。如果nPenStyle是PS\_COSMETIC，则结构的lbColor成员LOGBRUSH指定笔的颜色，并且结构的lbStyle成员LOGBRUSH必须设置为BS\_SOLID。如果nPenStyle是PS\_GEOMETRIC，则必须使用所有成员来指定笔的画笔属性。

参数nStyleCount：指定lpStyle数组的双字单位长度。如果nPenStyle不是 PS\_USERSTYLE，该值必须为零。

参数lpStyle：指向一组双字值。第一个值指定用户定义样式中第一个虚线的长度，第二个值指定第一个空格的长度，依此类推。如果不是PS\_USERSTYLE，则该指针必须为NULL。

注：几种构造函数的区别：

1、如果使用第一种不带参数的构造函数，则必须使用CreatePen，CreatePenIndirect或CreateStockObject函数进行初始化；如果使用后两种构造函数，则不需要进一步的初始化。

2、带有参数的构造函数可能会在遇到错误时抛出异常，而没有参数的构造函数将始终成功。

例：创建宽度为2的实心红画笔。

第一种方法：

//创建宽度为2的实心红色笔。

CPen myPen1(PS\_SOLID, 10, RGB(255, 0, 0));

第二种方法：

//创建宽度为2的实心红色笔。

LOGBRUSH logBrush;

logBrush.lbStyle = BS\_SOLID;

logBrush.lbColor = RGB(255, 0, 0);

CPen myPen2(PS\_SOLID|PS\_GEOMETRIC|PS\_ENDCAP\_ROUND,2,&logBrush);

### 3.2 CPen::CreatePen

BOOL CreatePen(

int nPenStyle,

int nWidth,

COLORREF crColor);

BOOL CreatePen(

int nPenStyle,

int nWidth,

const LOGBRUSH\* pLogBrush,

int nStyleCount = 0,

const DWORD\* lpStyle = NULL);

函数功能：创建具有指定样式，宽度和画笔属性的逻辑装饰或几何笔，并将其附加到CPen对象。该函数有两个重载函数

CreatePen函数的参数含义及用法与Cpen构造函数相同。

函数返回值：如果成功则为非零，如果失败则为零。

注：CreatePen函数注意事项：

1、宽度大于1像素的笔应始终具有**PS\_NULL**，**PS\_SOLID**或**PS\_INSIDEFRAME**样式。

2、如果笔具有**PS\_INSIDEFRAME**样式，并且颜色与逻辑颜色表中的颜色不匹配，笔将以抖动颜色绘制。该**PS\_SOLID**画笔样式不能用于创建一个带有抖动颜色的笔。如果笔宽小于或等于1，则样式**PS\_INSIDEFRAME**与**PS\_SOLID**相同。

3、第二个CreatePen函数初始化具有指定样式，宽度和画笔属性的逻辑装饰或几何笔。装饰笔的宽度总是为1; 几何笔的宽度总是以世界单位指定。应用程序创建逻辑笔后，可以通过调用[CDC :: SelectObject](https://msdn.microsoft.com/en-us/library/fxhhde73.aspx#cdc__selectobject)函数选择该画笔，从而用于绘制线条和曲线。

4、如果nPenStyle是**PS\_COSMETIC**和**PS\_USERSTYLE**，则lpStyle数组中的条目以样式单位指定破折号和空格的长度。样式单位由用于绘制线的设备定义。

5、如果nPenStyle是**PS\_GEOMETRIC**和**PS\_USERSTYLE**，则lpStyle数组中的条目以逻辑单位指定破折号和空格的长度。

6、如果nPenStyle是**PS\_ALTERNATE**，则忽略样式单位，并设置为其他像素。

7、当应用程序不再需要一个给定的笔时，它应该调用[CGdiObject :: DeleteObject](https://msdn.microsoft.com/en-us/library/e1af4h6d.aspx#cgdiobject__deleteobject)成员函数或销毁该CPen对象，以使该资源不再使用。在绘图设备中选择笔时，应用程序不应删除笔。

例：用该函数创建宽度为2的实心红画笔。

第一种方法：

//创建宽度为2的实心红色笔。

CPen myPen1;

myPen1.CreatePen(PS\_SOLID, 2, RGB(255, 0, 0));

第二种方法：

//创建宽度为2的实心红色笔。

CPen myPen2;

LOGBRUSH logBrush;

logBrush.lbStyle = BS\_SOLID;

logBrush.lbColor = RGB(0, 255, 0);

myPen2.CreatePen(PS\_DOT | PS\_GEOMETRIC | PS\_ENDCAP\_ROUND, 2, &logBrush);

### 3.3 CPen::CreatePenIndirect

BOOL CreatePenIndirect(LPLOGPEN lpLogPen);

函数功能：初始化具有指向的结构中给出的样式，宽度和颜色的笔lpLogPen。

参数lpLogPen：指向包含有关笔的信息的Windows LOGPEN结构。

函数返回值：如果成功则为非零，如果失败则为零。

注：CreatePenIndirect函数注意事项。

1、宽度大于1像素的笔应始终具有PS\_NULL，PS\_SOLID或PS\_INSIDEFRAME样式。

如果笔具有PS\_INSIDEFRAME样式，并且颜色与逻辑颜色表中的颜色不匹配，笔将以抖动颜色绘制。如果笔宽小于或等于1，则样式**PS\_INSIDEFRAME**与**PS\_SOLID**相同。

2、第二个成员变量lopnWidth虽然也是用于指定画笔的宽度，但其类型却是POINT结构，在该结构中的y成员变量不起任何作用，只采用x成员变量来表示画笔宽度。

3、在实际使用中，既可以使用CreatePenIndirect()函数来创建画笔对象，也可以使用CreatePen()函数来创建，其实这两个函数是可互换的。

例：用该函数创建宽度为2的实心红画笔。

//创建宽度为2的实心红色笔。

LOGPEN logpen;

CPen myPen;

logpen.lopnWidth.x = 2;//设置宽度为2

logpen.lopnColor = RGB(255, 0, 0);//设置颜色为红色

logpen.lopnStyle = PS\_SOLID;//设置风格为实线

myPen.CreatePenIndirect(&logpen);//使用新的设置创建画笔

### 3.4 CPen::FromHandle

static CPen\* PASCAL FromHandle(HPEN hPen);

函数功能：返回一个指向Windows GDI笔的对象句柄的指针。

参数hpen：HPEN指向Windows GDI笔的句柄。

函数返回值：CPen如果成功，则返回指向对象的指针; 否则为NULL。

注：FromHandle函数注意事项。

如果CPen对象未附加到句柄，则CPen会创建并附加临时对象。该临时CPen对象仅在下一次应用程序在其事件循环中有空闲时间时才有效，此时所有临时图形对象都将被删除。换句话说，临时对象仅在处理一个窗口消息期间有效。

例：将一个HPEN句柄转换为CPen\*对象。

//将一个HPEN句柄转换为CPen\*对象。

HPEN hPen;

CPen\* pPen = CPen::FromHandle(hPen);

### 3.5 CPen::GetExtLogPen

int GetExtLogPen(EXTLOGPEN\* pLogPen);

函数功能：获得EXTLOGPEN底层结构。

参数pLogPen：指向包含笔的信息的EXTLOGPEN结构。

函数返回值：如果成功则为非零，如果失败则为零。

注：EXTLOGPEN结构，限定出笔的风格、宽度和刷属性。

例：调用GetExtLogPen检索笔的属性，然后创建一个具有相同颜色的新的装饰笔。

//创建一个具有相同颜色的新的装饰笔。

CPen oldPen(PS\_SOLID, 1, RGB(255, 0, 0));

EXTLOGPEN extlogpen;

oldPen.GetExtLogPen(&extlogpen);

CPen newPen;

LOGBRUSH LogBrush = { extlogpen.elpBrushStyle, extlogpen.elpColor,

extlogpen.elpHatch };

newPen.CreatePen(PS\_COSMETIC, 1, &LogBrush,0,0);

### 3.6 CPen::GetLogPen

int GetLogPen(LOGPEN\* pLogPen);

函数功能：获得LOGPEN基础结构。

参数pLogPen：指向包含有关笔的信息的LOGPEN结构。

函数返回值：如果成功则为非零，如果失败则为零。

注：LOGPEN结构，定义笔的样式，颜色和图案。

例：调用ExtLogPen检索现有笔的属性，然后创建一个具有相同风格的新的画笔。

//创建一个具有相同风格的新的画笔。

CPen oldPen(PS\_SOLID, 2, RGB(255, 0, 0));

LOGPEN logpen;

oldPen.GetLogPen(&logpen);

CPen newPen(logpen.lopnStyle, logpen.lopnWidth.x, logpen.lopnColor);

### 3.7 CPen :: operator HPEN

operator HPEN() const;

函数功能：获取对象附加的Windows GDI句柄。

函数返回值：如果成功，则由对象表示的Windows GDI对象的句柄CPen; 否则**为NULL**。

注：这个操作符是一个铸造操作符，它支持直接使用一个HPEN对象。

例：获取创建画笔对象的句柄。

//创建宽度为2的实心红画笔。

CPen myPen(PS\_SOLID, 2, RGB(255, 0, 0));

//获取画笔对象的句柄

HPEN hPen = (HPEN)myPen;

## 4、例程：应用CPen类函数画一条红色虚线。

void CCPrintDialogClassView::OnDraw(CDC\* pDC)

{

CCPrintDialogClassDoc\* pDoc = GetDocument();

ASSERT\_VALID(pDoc);

if (!pDoc)

return;

// TODO: 在此处为本机数据添加绘制代码

CPen newPen;

newPen.CreatePen(PS\_DASH, 1, RGB(255, 0, 0));

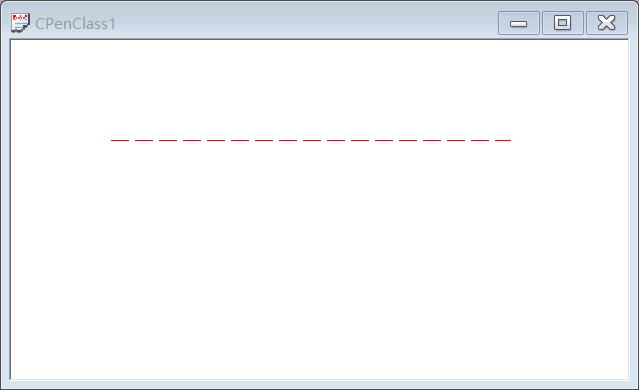
HGDIOBJ oldPen = pDC->SelectObject(newPen);

pDC->MoveTo(100, 100);

pDC->LineTo(500, 100);

pDC->SelectObject(oldPen);

}



**二、MFC中CColorDialog类的介绍及用法**

## 1、CColorDialog类的简介

CColorDialog 对象是用于显示系统定义颜色的列表的对话框。用户可以从列表中选择或创建特定颜色，然后在对话框退出时将其返回给应用程序。

CColorDialog类的继承层次结构为CObject -> CCmdTarget ->CWnd ->CDialog -> CCommonDialog->CColorDialog。

使用CColorDialog类时需调用头文件 afxdlgs.h。

## 2、CColorDialog类的成员简介

|  |  |  |
| --- | --- | --- |
| 类型 | 名称 | 描述 |
| 构造函数 | CColorDialog::CColorDialog | 构造一个CColorDialog对象。 |
| 公有成员函数 | CColorDialog::DoModal | 显示颜色对话框，允许用户进行选择。 |
| CColorDialog::GetColor | 返回包含所选颜色值的COLORREF结构。 |
| CColorDialog::GetSavedCustomColors | 检索用户创建的自定义颜色。 |
| CColorDialog::SetCurrentColor | 将当前颜色选择强制为指定的颜色。 |
| 保护型成员函数 | CColorDialog::OnColorOK | 覆盖以验证输入到对话框中的颜色。 |
| 公有数据结构 | CColorDialog::m\_cc | 用于自定义对话框设置的结构。 |

## 3、CColorDialog类的成员的用法详解

### 3.1 CColorDialog::CColorDialog

CColorDialog(

COLORREF clrInit = 0,

DWORD dwFlags = 0,

CWnd\* pParentWnd = NULL);

函数功能：构造一个CColorDialog对象。

参数clrInit：默认颜色选择。如果未指定值，则默认值为RGB（0,0,0）（黑色）。

参数dwFlags：一组自定义对话框的功能和外观的标志。

参数pParentWnd：指向对话框的父窗口或所有者窗口的指针。

例：创建一个所选颜色为红色的完全打开式对话框。

//创建一个所选颜色为红色的完全打开式对话框。

CColorDialog dlg(RGB(255, 0, 0), CC\_FULLOPEN);

### 3.2 CColorDialog::DoModal

virtual INT\_PTR DoModal();

函数功能：用于显示Windows常用颜色对话框，并允许用户选择颜色。

返回值：IDOK或IDCANCEL。如果返回IDCANCEL，则需调用Windows CommDlgExtendedError函数来确定是否发生错误。（IDOK和IDCANCEL是指示用户是否选择了“确定”或“取消”按钮的常量。）

注：如果要通过设置m\_cc结构的成员来初始化各种颜色对话框选项，则应该在构建对话框对象之后与调用该函数之前进行操作。调用后DoModal，可以调用其他成员函数，将用户输入的设置或信息检索到对话框中。

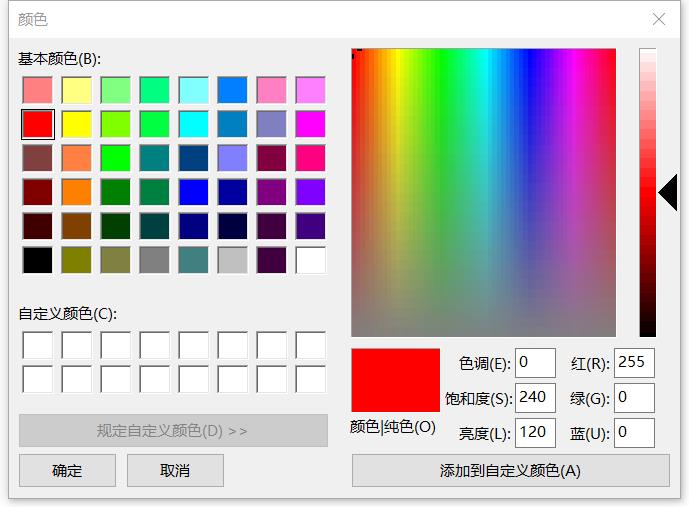
例：显示之前创建的所选颜色为红色的完全打开式对话框。

//创建一个所选颜色为红色的完全打开式对话框。

CColorDialog dlg(RGB(255, 0, 0), CC\_FULLOPEN);

//显示之前创建的所选颜色为红色的完全打开式对话框。

dlg.DoModal();



### 3.3 CColorDialog::GetColor

COLORREF GetColor() const;

函数功能：在调用DoModal函数之后调用该函数，来检索用户所选择的颜色。

返回值：一个包含颜色对话框中选择颜色的RGB信息的COLORREF值。

例：获取用户选择颜色的RGB值。

//创建一个所选颜色为红色的完全打开式对话框

CColorDialog dlg(RGB(255, 0, 0), CC\_FULLOPEN);

//显示之前创建的所选颜色为红色的完全打开式对话框。

if (dlg.DoModal() == IDOK)

{

COLORREF color = dlg.GetColor();//获取用户所选择颜色的RGB值

CString a;

a.Format(\_T("所选颜色的RGB值为red = %u, ")

\_T("green = %u, blue = %u\n"),

GetRValue(color), GetGValue(color), GetBValue(color));

pDC->TextOut(20, 20, a);//将获取的值输出

}

}

### 3.4 CColorDialog::GetSavedCustomColors

COLORREF GetColor() const;

函数功能：获取用户自定义的十六种颜色。

返回值：指向16个RGB颜色值数组的指针，用于存储用户创建的自定义颜色。

注：

1、GetSavedCustomColors函数提供检索用户自定义颜色的通道。DoModal函数返回IDOK后可以检索这些颜色。

2、返回数组中的16个RGB值中的每一个都初始化为RGB（255,255,255）（白色）。用户选择的自定义颜色仅在应用程序中的对话框调用的过程中进行保存。如果要在应用程序调用的过程中保存这些颜色，则必须以其他方式保存它们，例如在初始化（.INI）文件中。

例：获取用户自定义的16种颜色的RGB值。

CColorDialog dlg;

dlg.DoModal();

if (dlg.DoModal() == IDOK)

{

//获取用户自定义的16种颜色的RGB值

COLORREF\* ccolor = dlg.GetSavedCustomColors();

CString a;

for (int i = 0; i < 16; i++)

{

a.Format(\_T("自定义的第%d种颜色的RGB值：red = %u,green = %u, blue= %u\n"),

i+1,

GetRValue(ccolor[i]),

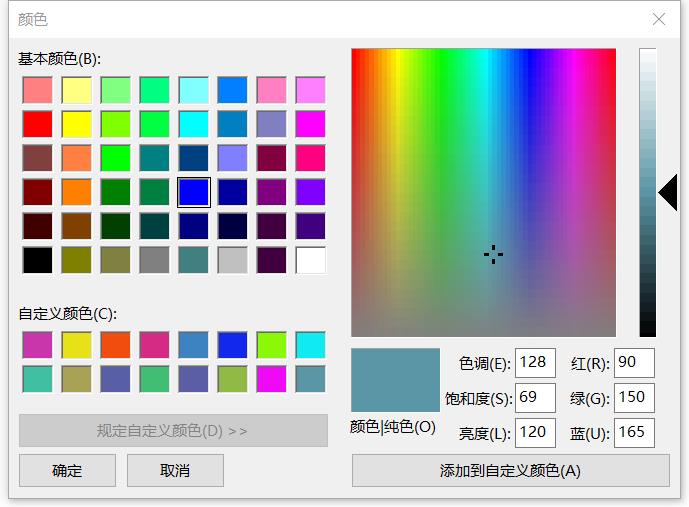
GetGValue(ccolor[i]),

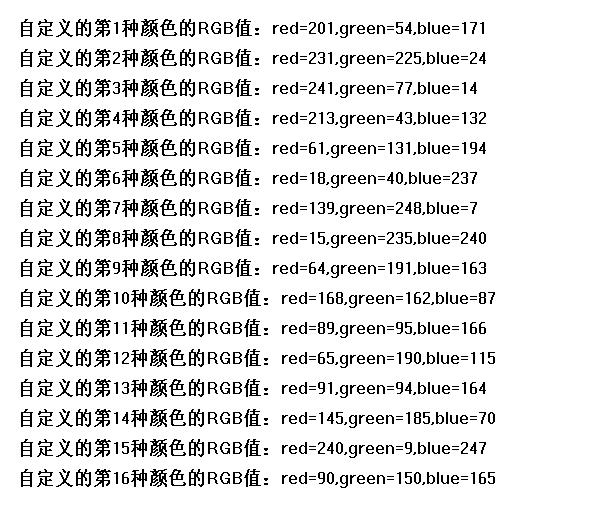
GetBValue(ccolor[i]));

pDC->TextOut(20, 20 + 30 \* i, a);//输出获取的值

}

}





**3.5** **CColorDialog::m\_cc**

CHOOSECOLOR m\_cc;

结构功能：一个CHOOSECOLOR类型的结构体，它的成员用于存储对话框的特征和值。

注：构建CColorDialog对象后，可以用m\_cc在调用[DoModal](https://msdn.microsoft.com/en-us/library/6w6cd538.aspx#ccolordialog__domodal)成员函数之前设置对话框的各个方面。

例：使用m\_cc成员变量自定义CColorDialog的设置。CColorDialog将被显示为完全打开，红色显示为所选颜色。

//使用m\_cc成员变量自定义CColorDialog的设置。

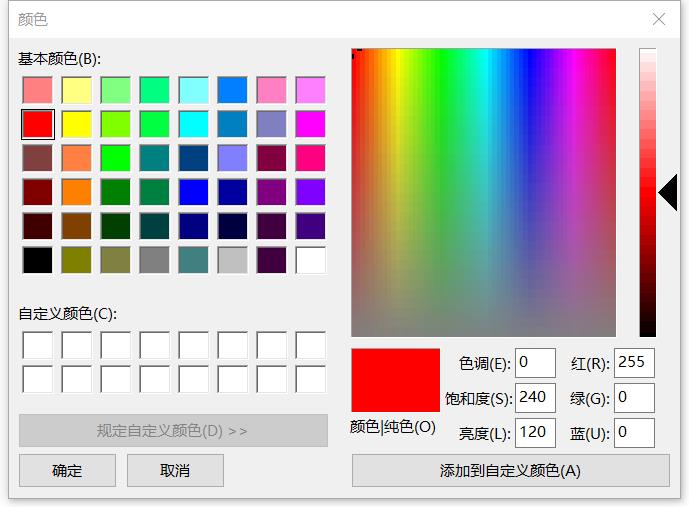
//CColorDialog将被显示为完全打开，红色显示为所选颜色。

CColorDialog dlg;

dlg.m\_cc.Flags |= CC\_FULLOPEN | CC\_RGBINIT;

dlg.m\_cc.rgbResult = RGB(255, 0, 0);

dlg.DoModal();



### 3.6CColorDialog::OnColorOK

virtual BOOL OnColorOK();

函数功能：覆盖并确认输入到对话框中的颜色。

注：

1、只有当用户在颜色对话框中提供自定义颜色的自定义验证时，才会重写此功能。

2、用户可以通过以下两种方法之一选择一种颜色：

a、单击调色板上的颜色。所选颜色的RGB值然后反映在相应的RGB编辑框中。

b、 在RGB编辑框中输入值。

3、该函数允许你根据具体应用，拒绝用户输入到常见颜色对话框中的颜色。

4、通常不需要使用此函数，因为框架提供颜色的默认验证，并在输入无效颜色时显示消息框。

5、您可以从OnColorOK内部调用SetCurrentColor来强制进行颜色选择。一旦OnColorOK被触发（即用户单击确定以接受颜色更改），您可以调用GetColor获取新颜色的RGB值。

例：用该函数验证输入的颜色，如果为黑色，则强制当前颜色选择为创建对话框时最初的颜色。

class cMyColorDlg :public CColorDialog

{

public:

cMyColorDlg(){}

~cMyColorDlg(){}

//OnColorOK函数覆盖输入道红色，绿色和蓝色编辑控件的颜色。

//如果颜色是黑色（RGB(0,0,0)），

//则强制当前颜色为创建对话框时最初选择的颜色。

//颜色对话框不会关闭。

//用户可以输入新的颜色。

BOOL OnColorOK()

{

COLORREF clrref = GetColor();

if (clrref == RGB(0, 0, 0))

{

AfxMessageBox(\_T("BLACK is not an acceptable color.")

\_T("Please enter a color again"));

//GetColor()返回最初选择的颜色。

SetCurrentColor(GetColor());

//不会关闭颜色对话框。

return TRUE;

}

//确定关闭颜色对话框。

return FALSE;

}

};

//用该函数验证输入的颜色，如果为黑色，

//则强制当前颜色选择为创建对话框时最初的颜色。

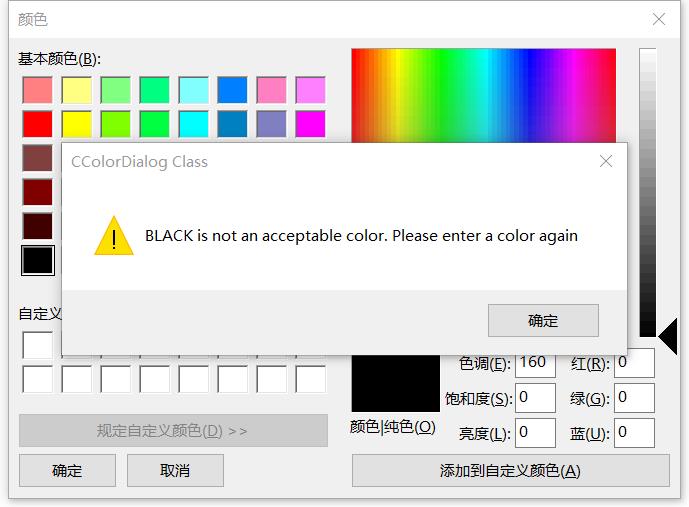
cMyColorDlg dlg;

dlg.m\_cc.Flags |= CC\_FULLOPEN | CC\_RGBINIT;

dlg.m\_cc.rgbResult = RGB(255, 0, 0);

dlg.DoModal();

dlg.OnColorOK();



### 3.7CColorDialog :: SetCurrentColor

void SetCurrentColor(COLORREF clr);

函数功能：在调用DoModal函数之后调用该函数，将当前颜色选择强制为指定的颜色值clr。

参数clr：RGB颜色值。

注：这个函数是从消息处理程序或者OnColorOk函数中调用。对话框将根据clr的参数值自动更新用户的颜色选择。

## 4、例程：应用CColorDialog类，从(100,100)向(600,100)绘制一条宽度为10像素的直线。颜色由用户自行选择，初始值为红色，且最终不能为黑色。

void CCPrintDialogClassView::OnDraw(CDC\* pDC)

{

CCPrintDialogClassDoc\* pDoc = GetDocument();

ASSERT\_VALID(pDoc);

if (!pDoc)

return;

// TODO: 在此处为本机数据添加绘制代码

//应用CColorDialog类，从(100,100)向(600,100)绘制一条宽度为10像素的直线。

//颜色由用户自行选择，初始值为红色，且最终不能为黑色。

CPen newPen;//构造CPen类对象newPen。

cMyColorDlg dlg;//构造cMyColorDlg对象dlg。

COLORREF color;//画笔选择颜色。

dlg.m\_cc.Flags |= CC\_FULLOPEN | CC\_RGBINIT;

dlg.m\_cc.rgbResult = RGB(255, 0, 0);

dlg.DoModal();

dlg.OnColorOK();

color = dlg.GetColor();//获取用户选择的颜色。

newPen.CreatePen(PS\_SOLID, 10, color);//创造相应颜色的画笔。

HGDIOBJ oldPen = pDC->SelectObject(newPen);//记录原有画笔。

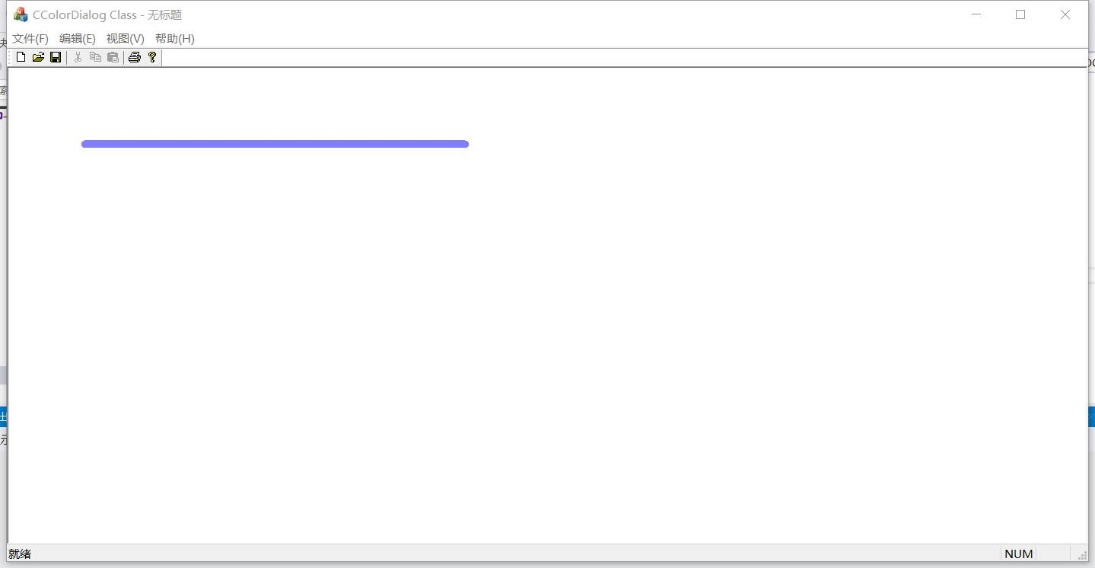
pDC-> SelectObject(newPen);

pDC->MoveTo(100, 100);

pDC->LineTo(600, 100);//从（100，100）向（600，100）画线。

pDC->SelectObject(oldPen);//恢复原有画笔。

}

****

# 三、MFC中CFontDialog类的介绍及用法

## 1、CFontDialog类的简介

CFontDialog对象是与当前系统中安装的字体列表对应的对话框。用户可以从列表中选择特定的字体，然后将该选择报告回应用程序。

CColorDialog类的继承层次结构为CObject -> CCmdTarget ->CWnd ->CDialog -> CCommonDialog->CFontDialog。

使用CFontDialog类时需调用头文件 afxdlgs.h。

## 2、CFontDialog类的成员简介

|  |  |  |
| --- | --- | --- |
| 类型 | 名称 | 描述 |
| 构造函数 | CFontDialog::CFontDialog | 构造一个CFontDialog对象。 |
| 公有成员函数 | CFontDialog::DoModal | 显示对话框，允许用户进行选择。 |
| CFontDialog::GetCharFormat | 检索所选字体的字符格式。 |
| CFontDialog::GetColor | 返回所选字体的颜色。 |
| CFontDialog::GetCurrentFont | 将当前所选字体的特征分配给LOGFONT结构。 |
| CFontDialog::GetFaceName | 返回所选字体的封面名称。 |
| CFontDialog::GetSize | 返回所选字体的大小。 |
| CFontDialog::GetStyleName | 返回所选字体的样式名称。 |
| CFontDialog::GetWeight | 返回所选字体的重量。 |
| CFontDialog::IsBold | 确定字体是否为粗体。 |
| CFontDialog::IsItalic | 确定字体是否是斜体。 |
| CFontDialog::IsStrikeOut | 确定字体是否显示与删除。 |
| CFontDialog::IsUnderline | 确定字体是否带下划线。 |
| 公有数据结构 | CFontDialog::m\_cf | 用于定制CFontDialog对象的结构。 |

## 3、CFontDialog类的成员的用法详解3.1CFontDialog::CFontDialog

CFontDialog(

LPLOGFONT lplfInitial = NULL,

DWORD dwFlags = CF\_EFFECTS | CF\_SCREENFONTS,

CDC\* pdcPrinter = NULL,

CWnd\* pParentWnd = NULL);

CFontDialog(

const CHARFORMAT& charformat,

DWORD dwFlags = CF\_SCREENFONTS,

CDC\* pdcPrinter = NULL,

CWnd\* pParentWnd = NULL);

函数功能：构造一个CFontDialog对象。

参数lplfInitial：指向[LOGFONT](https://msdn.microsoft.com/library/windows/desktop/dd145037)数据结构的指针，可以让您设置字体的某些特性。

参数charformat：指向CHARFORMAT数据结构的指针，允许您在丰富的编辑控件中设置某些字体的特征。

参数dwFlags：指定一个或多个选择字体标志。可以使用按位或运算符组合一个或多个预设值。（注：如果您修改m\_cf.Flags结构成员，请确保在更改中使用按位或运算符，以保持默认行为不变。）

参数pdcPrinter：指向打印机设备上下文的指针。如果提供，则该参数指向要在其上选择字体的打印机的打印机设备上下文。

参数pParentWnd：指向字体对话框的父窗口或所有者窗口的指针。

注：

1、构造函数会自动填充结构的成员CHOOSEFONT。如果想要一个不同于默认值的字体对话框，则应该只更改这些对话框。

2、此函数的第一个重载函数只有在没有丰富的编辑控件支持时才存在。

例：构造以12点“Times New Roman”的字体作为所选字体的对话框实例对象。

//构造以12点“Times New Roman”的字体作为所选字体的对话框实例对象。

LOGFONT lf;

memset(&lf, 0, sizeof(LOGFONT));//将lf的数据清零。

//假定存在一个被初始化的CWnd

CClientDC dc(this);

//将字体高度转换为显示器对应分辨率下12点的高度。

lf.lfHeight = -MulDiv(12, dc.GetDeviceCaps(LOGPIXELSX), 72);

//将lf对应字体的名字显示为“Times New Roman”

\_tcscpy\_s(lf.lfFaceName, LF\_FACESIZE, \_T("Times New Roman"));

//创建初始化字体为lf对应类型的字体对话框对象。

CFontDialog fdlg(&lf);

### 3.2CFontDialog::DoModal

virtual INT\_PTR DoModal();

函数功能：调用此功能显示Windows常用字体对话框，并允许用户选择字体。

返回值：IDOK或IDCANCEL。如果返回IDCANCEL，则需调用Windows CommDlgExtendedError函数来确定是否发生错误。（IDOK和IDCANCEL是指示用户是否选择了“确定”或“取消”按钮的常量。）

注：如果要通过设置m\_cf结构的成员来初始化各种字体对话框控件，那么需要在调用DoModal函数之前与在构建对话框对象之后执行此操作。调用DoModal之后，可以调用其他成员函数，将用户输入的设置或信息检索到对话框中。

例：显示之前所创建的对话框对象。

//显示所构造的以12点“Times New Roman”的字体作为所选字体的对话框实例对象。

LOGFONT lf;

memset(&lf, 0, sizeof(LOGFONT));//将lf的数据清零。

//假定存在一个被初始化的CWnd

CClientDC dc(this);

//将字体高度转换为显示器对应分辨率下12点的高度。

lf.lfHeight = -MulDiv(12, dc.GetDeviceCaps(LOGPIXELSX), 72);

//将lf对应字体的名字显示为“Times New Roman”

\_tcscpy\_s(lf.lfFaceName, LF\_FACESIZE, \_T("Times New Roman"));

//创建初始化字体为lf对应类型的字体对话框对象。

CFontDialog fdlg(&lf);

### 3.3CFontDialog::GetCharFormat

void GetCharFormat(CHARFORMAT& cf) const;

函数功能：检索所选字体的字符格式。

参数cf：一个包含所选字体字符格式信息的CHARFORMAT结构。

### 3.4CFontDialog::GetColor

COLORREF GetColor() const;

函数功能：调用此函数来检索所选的字体颜色。

返回值：所选字体的颜色。

例：获取所选字体的颜色。

//获取所选字体的颜色。

CFontDialog dlg;

if (dlg.DoModal() == IDOK)

{

COLORREF color = dlg.GetColor();

CString s;

s.Format(\_T("所选字体的颜色的RGB值为（%d,%d,%d）"),

GetRValue(color), GetGValue(color), GetBValue(color));

pDC->TextOutW(100, 100, s);

}

### 3.5CFontDialog::GetCurrentFont

void GetCurrentFont(LPLOGFONT lplf);

函数功能：调用此函数将当前选定字体的特征分配给LOGFONT结构的成员。

参数lplf：一个指向LOGFONT的指针。

注：如果在调用[DoModal](https://msdn.microsoft.com/en-us/library/kck77523.aspx#cfontdialog__domodal)函数期间调用此函数，它将返回当前选择（用户在对话框中看到或已更改的内容）。如果在调用DoModal函数之后调用此函数（只有DoModal返回IDOK），则返回用户实际选择的内容。

例：获取当前选定字体的特征。

//获取所选字体的特征。

CFontDialog dlg;

if (dlg.DoModal() == IDOK)

{

LOGFONT lf;

dlg.GetCurrentFont(&lf);

CString s;

s.Format(\_T("所选字体的名称为："), lf.lfFaceName);

pDC->TextOutW(100, 100, s);

}

### 3.6CFontDialog::GetFaceName

CString GetFaceName() const;

函数功能：调用此函数检索所选字体的封面名称。

返回值：在CFontDialog对话框中选择的字体的封面名称。

例：获取所选字体的封面名称。

//获取所选字体的封面名称。

CFontDialog dlg;

if (dlg.DoModal() == IDOK)

{

CString facename = dlg.GetFaceName();

pDC->TextOutW(100, 75, \_T("所选字体名称为："));

pDC->TextOutW(100, 100, facename);

}

### 3.7CFontDialog::GetSize

int GetSize() const;

函数功能：调用此函数来检索所选字体的大小。

返回值：字体的大小，以十分之一点为单位。

例：获取所选字体的大小。

//获取所选字体的大小。

CFontDialog dlg;

if (dlg.DoModal() == IDOK)

{

int size = dlg.GetSize();

CString s;

s.Format(\_T("所选字体大小为：%d"), size);

pDC->TextOutW(100, 100, s);

}

### 3.8CFontDialog::GetStyleName

CString GetStyleName() const;

函数功能：调用此函数来检索所选字体的样式名称。

返回值：字体的样式名称。

例：获取所选字体的样式名称。

//获取所选字体的样式名称。

CFontDialog dlg;

dlg.m\_cf.Flags |= CF\_USESTYLE;

if (dlg.DoModal() == IDOK)

{

CString stylename = dlg.GetStyleName();

pDC->TextOutW(100, 75, \_T("所选字体的样式为："));

pDC->TextOutW(100, 100, stylename);

}

### 3.9CFontDialog::GetWeight

int GetWeight() const;

函数功能：调用此函数来检索所选字体的权重。

返回值：所选字体权重。

例：获取所选字体权重。

//获取所选字体的权重。

CFontDialog dlg;

if (dlg.DoModal() == IDOK)

{

int weight;

CString s;

s.Format(\_T("所选字体的权重为：%d"), weight);

pDC->TextOutW(100, 100, s);

}

### 3.10CFontDialog::IsBold

BOOL IsBold() const;

函数功能：调用此函数确定所选字体是否为粗体。

返回值：如果所选字体的Bold特性已启用，则为非零; 否则0。

例：判断所选字体是否为粗体。

//判断所选字体是否为粗体。

CFontDialog dlg;

if (dlg.DoModal() == IDOK)

{

BOOL bold = dlg.IsBold();

CString s;

if(bold==1) s.Format(\_T("所选字体是粗体"));

else s.Format(\_T("所选字体不是粗体"));

pDC->TextOutW(100, 100, s);

}

### 3.11CFontDialog::IsItalic

BOOL IsItalic() const;

函数功能：调用此函数确定所选字体是否为斜体。

返回值：如果所选字体的Italic特性已启用，则为非零; 否则0。

例：判断所选字体是否为斜体。

//判断所选字体是否为斜体。

CFontDialog dlg;

if (dlg.DoModal() == IDOK)

{

BOOL italic = dlg.IsItalic();

CString s;

if(italic ==1) s.Format(\_T("所选字体是斜体。"));

else s.Format(\_T("所选字体不是斜体。"));

pDC->TextOutW(100, 100, s);

}

### 3.12CFontDialog::IsStrikeOut

BOOL IsStrikeOut() const;

函数功能：调用此函数确定所选字体是否有删除符。

返回值：如果所选字体的StrikeOut特性已启用，则为非零; 否则0。

例：判断所选字体是否有删除符。

//判断所选字体是否有删除符。

CFontDialog dlg;

if (dlg.DoModal() == IDOK)

{

BOOL strikeout = dlg.IsStrikeOut();

CString s;

if(strikeout==1) s.Format(\_T("所选字体有删除符。"));

else s.Format(\_T("所选字体没有删除符。"));

pDC->TextOutW(100, 100, s);

}

### 3.13CFontDialog::IsUnderLine

BOOL IsUnderline() const;

函数功能：调用此函数确定所选字体是否有下划线。

返回值：如果所选字体的UnderLine特性已启用，则为非零; 否则0。

例：判断所选字体是否有下划线。

//判断所选字体是否有下划线。

CFontDialog dlg;

if (dlg.DoModal() == IDOK)

{

BOOL underline = dlg.IsUnderline();

CString s;

if(underline==1) s.Format(\_T("所选字体有下划线。"));

else s.Format(\_T("所选字体没有下划线。"));

pDC->TextOutW(100, 100, s);

}

### 3.14CFontDialog::m\_cf

CHOOSEFONT m\_cf;

结构功能：存储对话框对象的特征。

注：构建CFontDialog对象后，可以在调用DoModal成员函数之前用m\_cf修改对话框的各个方面。

例：用m\_cf构建一个字体对话框对象，使得字体初始颜色为红色。

//用m\_cf构建一个字体对话框对象，使得字体初始颜色为红色。

CFontDialog dlg;

dlg.m\_cf.Flags |= CF\_APPLY;

dlg.m\_cf.rgbColors = RGB(255, 0, 0);

dlg.DoModal();

## 4、例程：应用CFontDialog类，让用户自行选择一种字体，输出“Good programmers have sailed!”。

void CCPrintDialogClassView::OnDraw(CDC\* pDC)

{

CCPrintDialogClassDoc\* pDoc = GetDocument();

ASSERT\_VALID(pDoc);

if (!pDoc)

return;

// TODO: 在此处为本机数据添加绘制代码

//应用CFontDialog类，让用户自行选择一种字体，

//输出“Good programmers have sailed!”。

CFontDialog dlg;//构造CFontDialog类对象

CFont font;//构造CFont类对象

LOGFONT lf;//构造字体参数结构

dlg.DoModal();//打开对话框

memset(&lf, 0, sizeof(LOGFONT));//将lf结构归零

dlg.GetCurrentFont(&lf);//获取用户选择的字体参数

VERIFY(font.CreateFontIndirect(&lf));//创造该字体

CClientDC dc(this);

CFont\* def\_font = dc.SelectObject(&font);//选取该字体

dc.TextOut(5, 5, \_T("Good programmers have sailed!"));

dc.SelectObject(def\_font);

font.DeleteObject();//销毁该字体

}





**四、MFC中CPrintDialog类的介绍及用法**

## 1、CPrintDialog类的简介

CPrintDialog的功能是提供打印设置和页面设置的通用对话框，可以通过对话框来处理应用程序的打印过程的许多方面。

CPrintDialog类的继承层次结构为CObject -> CCmdTarget ->CWnd ->CDialog -> CCommonDialog->CPrintDialog。

使用CPrintDialog类时需调用头文件 afxdlgs.h。

## 2、CPrintDialog类的成员简介

|  |  |  |
| --- | --- | --- |
| 类型 | 名称 | 描述 |
| 构造函数 | CPrintDialog::CPrintDialog | 构造一个CPrintDialog对象。 |
| 公有成员函数 | CPrintDialog::CreatePrinterDC | 创建打印机设备描述表，而不显示打印对话框。 |
| CPrintDialog::DoModal | 显示对话框，允许用户进行选择。 |
| CPrintDialog::GetCopies | 检索所要求的份数。 |
| CPrintDialog::GetDefaults | 检索设备默认值，而不显示对话框。 |
| CPrintDialog::GetDeviceName | 检索当前选择的打印机设备的名称。 |
| CPrintDialog::GetDevMode | 检索DEVMODE结构。 |
| CPrintDialog::GetDriverName | 检索当前选择的打印机驱动程序的名称。 |
| CPrintDialog::GetFromPage | 检索打印范围的起始页。 |
| CPrintDialog::GetPortName | 检索当前选择的打印机端口的名称。 |
| CPrintDialog::GetPrinterDC | 检索打印机设备上下文的句柄。 |
| CPrintDialog::GetToPage | 检索打印范围的结束页。 |
| CPrintDialog::PrintAll | 确定是否打印文档的所有页面。 |
| CPrintDialog::PrintCollate | 确定是否请求已整理的副本。 |
| CPrintDialog::PrintRange | 确定是否仅打印指定范围的页面。 |
| CPrintDialog::PrintSelection | 确定是否只打印当前选择的项目。 |
| 公有数据结构 | CPrintDialog::m\_pd | 用于定制CPrintDialog对象的结构。 |

## 3、CPrintDialog类的成员的用法详解

### 3.1 CPrintDialog::CPrintDialog

CPrintDialog(

BOOL bPrintSetupOnly,

DWORD dwFlags = PD\_ALLPAGES | PD\_USEDEVMODECOPIES | PD\_NOPAGENUMS | PD\_HIDEPRINTTOFILE | PD\_NOSELECTION,

CWnd\* pParentWnd = NULL);

函数功能：构造一个Windows打印或打印设置对话框对象。

参数bPrintSetupOnly：指定是否显示标准Windows打印对话框或打印设置对话框。将此参数设置为TRUE以显示标准Windows打印设置对话框。将其设置为FALSE以显示Windows打印对话框。如果bPrintSetupOnly为FALSE，打印设置选项按钮仍显示在打印对话框中。

参数dwFlags：可以使用一个或多个标志来自定义对话框的设置，使用按位或运算符组合。PD\_ALLPAGES标志将默认打印范围设置为文档的所有页面。

参数pParentWnd：指向对话框的父窗口或所有者窗口的指针。

注：当你调用该函数时将bPrintSetupOnly设置为FALSE，将会自动调用PD\_RETURNDC的属性。在调用DoModal，GetDefaults或者GetPrinterDC之后，打印机的DC（设备描述表）将会返回到m\_pd.hDC。只有调用者调用DeleteDC才能将该DC释放。

例：创造打印对话框，选择最初选择的“Selection”按钮。“All”单选按钮被启用，但是“Pages”单选按钮被禁用。

//创造打印对话框，选择最初选择的“Selection”按钮。

//“All”单选按钮被启用，但是“Pages”单选按钮被禁用。

CPrintDialog dlg(FALSE, PD\_SELECTION | PD\_USEDEVMODECOPIES);

### 3.2CPrintDialog::CreatePrinterDC

HDC CreatePrinterDC();

函数功能：从DEVMODE和DEVNAMES结构创建一个打印机设备描述表。

返回值：指向新创建的设备描述表的句柄。

注：该DC被认为是当前打印机的设备描述表，并且之前获得的打印机DC必须被用户删除。可以不显示“打印”对话框调用此函数并使用最终的DC。

例：根据用户提供的信息创建打印机设备描述表。

//根据用户提供的信息创建打印机设备描述表。

CPrintDialog dlg(FALSE);

if (dlg.DoModal() == IDOK)

{

HDC hdc = dlg.CreatePrinterDC();

ASSERT(hdc);

}

### 3.3CPrintDialog::DoModal

virtual INT\_PTR DoModal();

函数功能：显示Windows常用打印对话框，并允许用户选择各种打印选项，如份数，页面范围以及是否应将文档整理。

返回值：**IDOK**或**IDCANCEL**。如果返回**IDCANCEL**，则需调用Windows的CommDlgExtendedError函数来确定是否发生错误。

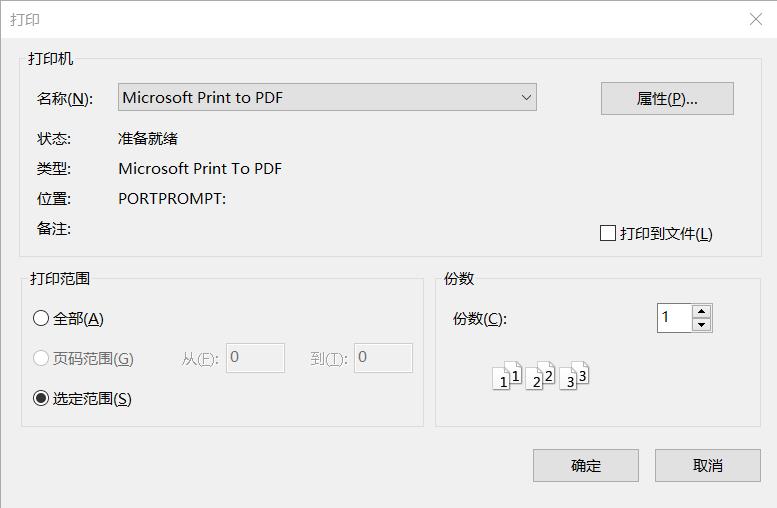
注：如果要通过设置m\_pd数据结构来初始化各种打印对话框选项，则应在调用DoModal函数之前与构建对话框对象之后执行此操作。调用DoModal函数之后，可以调用其他成员函数，将用户输入的设置或信息检索到对话框中。

例：显示3.1例中构建的打印对话框。

//创造打印对话框，选择最初选择的“Selection”按钮。

//“All”单选按钮被启用，但是“Pages”单选按钮被禁用。

CPrintDialog dlg(FALSE, PD\_SELECTION | PD\_USEDEVMODECOPIES);

 dlg.DoModal();

### 3.4CPrintDialog::GetCopies

int GetCopies() const;

函数功能：检索所要求的份数。

返回值：所要求的份数。

注：在调用DoModal函数之后调用此函数以检索所请求的副本数。

### 3.5CPrintDialog::GetDefaults

BOOL GetDefaults();

函数功能：检索默认打印机的设备默认值，而不显示对话框。

返回值：如果成功则为非0，否则为0。

注：

1、检索到的值存储在m\_pd数据结构中。

2、在某些情况下，调用此函数将会同时调用构造函数并将bPrintSetupOnly的值设置为FALSE。在这些情况中，打印机的设备描述表和hDevNames和hDevMode两个句柄将会自动分配。当结束使用该对话框时，调用者需要自行释放打印机的设备描述表（DC）。

例：获取默认打印机的设备描述表，并向用户返回打印机的分辨率（DPI）。

//获取默认打印机的设备描述表，并向用户返回打印机的分辨率（DPI）。

CPrintDialog dlg(FALSE);

if (!dlg.GetDefaults())

{

AfxMessageBox(\_T("你没有默认打印机！"));

}

else

{

//获取我们所获得的设备描述表

CDC dc;

dc.Attach(dlg.m\_pd.hDC);

//请求测量

int nHorz = dc.GetDeviceCaps(LOGPIXELSX);

int nVert = dc.GetDeviceCaps(LOGPIXELSY);

//大部分时候两个方向是相同的，但有时不同

CString str;

if (nHorz == nVert)

{

str.Format(\_T("您的打印机支持每英寸%d像素"), nHorz);

}

else

{

str.Format(\_T("您的打印机支持每英寸")

\_T("%d像素水平分辨率和%d像素垂直分辨率")

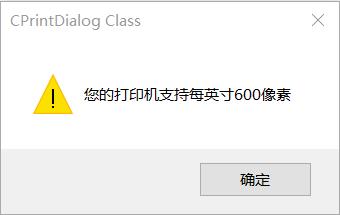
, nHorz, nVert);

}

//告知用户

AfxMessageBox(str);

}



### 3.6CPrintDialog::GetDeviceName

CString GetDeviceName() const;

函数功能：检索当前选择打印机设备的名称。

返回值：当前选择的打印机的名称。

注：在调用DoModal函数之后调用该函数检索当前选择的打印机名称，或者在调用GetDefaults函数之后检索默认打印机的默认设备名称。

例：显示用户的默认打印机名称及其连接的端口，以及打印机使用的假脱机程序名称。

//显示用户的默认打印机名称及其连接的端口，以及打印机使用的假脱机程序名称。

CPrintDialog dlg(FALSE);

if (!dlg.GetDefaults())

AfxMessageBox(\_T("你没有默认打印机！"));

else

{

CString strDescription;

strDescription.Format(\_T("你的默认打印机是 %s 在 %s 使用 %s."),

(LPCTSTR)dlg.GetDeviceName(),

(LPCTSTR)dlg.GetPortName(),

(LPCTSTR)dlg.GetDriverName());

AfxMessageBox(strDescription);

}

### 3.7CPrintDialog::GetDevMode

LPDEVMODE GetDevMode() const;

函数功能：检索DEVMODE结构。

返回值：DEMOVE的数据结构，包含打印驱动程序的设备初始化和环境的信息。

注：在调用DoModal函数或DetDefaults函数之后调用该函数检索有关打印设备信息。必须使用Windows GlobalUnlock函数解锁此结构所占用的内存。

### 3.8CPrintDialog::GetDriverName

CString GetDriverName() const;

函数功能：检索当前选择的打印机驱动程序的名称。

返回值：一个定义驱动程序名称的CString。

注：在调用DoModal函数或GetDefaults函数之后调用此函数来检索系统定义的打印机设备驱动程序的名称。

### 3.9CPrintDialog::GetFromPage

int GetFromPage() const;

函数功能：检索打印范围的起始页。

返回值：要打印的页面范围内的起始页码。

注：调用DoModal函数之后调用此函数，在要打印的页面范围内检索起始页码。

### 3.10CPrintDialog::GetPortName

CString GetPortName() const;

函数功能：检索当前选择的打印机端口的名称。

返回值：当前选择的打印机端口的名称。

注：在调用DoModal函数或GetDefaults函数之后调用此函数来检索当前选择的打印机端口的名称。

### 3.11CPrintDialog::GetPrinterDC

HDC GetPrinterDC() const;

函数功能：检索打印机设备描述表句柄。

返回值：若成功则返回打印机设备描述表句柄，否则为NULL。

注：如果构造函数CPrintDialog的参数bPrintSetupOnly为FALSE（表示显示“打印”对话框），则GetPrinterDC返回打印机设备描述表的句柄。使用完成后，必须调用Windows的DeleteDC函数删除设备描述表。

例：该函数的使用模板。

//CPrintDialog::GetPrinterDC函数使用模板。

CPrintDialog dlg(FALSE);

if (dlg.DoModal() == IDOK)

{

// 获取打印机设备描述表句柄

HDC hdc = dlg.GetPrinterDC();

ASSERT(hdc);

//使用句柄进行某些操作

//清除句柄

CDC::FromHandle(hdc)->DeleteDC();

}

### 3.12CPrintDialog::GetToPage

int GetToPage() const;

函数功能：检索打印范围的结束页。

返回值：要打印的页面范围内的结束页码。

注：在调用DoModal函数之后调用该函数，从而在要打印的页面范围内检索结束页码。

### 3.13CPrintDialog::m\_pd

PRINTDLG& m\_pd;

结构功能：存储打印对话框对象的特征。

注：构建CPrintDialog对象后，可以在调用DoModal函数之前用m\_pd设置对话框的各个方面。如果m\_pd直接修改数据成员，将覆盖任何默认行为。

例：显示选择“页面”单选按钮为初始选择的打印对话框。“全部”和“页面”单选按钮均被启用。并且用m\_pd数据结构确定打印的页面范围，汇报从开始到结束的页码。

// 显示选择“页面”单选按钮为初始选择的打印对话框。

//“全部”和“页面”单选按钮均被启用。

//并且用m\_pd数据结构确定打印的页面范围，汇报从开始到结束的页码。

CPrintDialog dlg(FALSE, PD\_PAGENUMS | PD\_USEDEVMODECOPIES);

dlg.m\_pd.nMinPage = dlg.m\_pd.nFromPage = 1;//确定最小页数为1

dlg.m\_pd.nMaxPage = dlg.m\_pd.nToPage = 10;//确定最大页数为10

if (dlg.DoModal() == IDOK)

{

//确定打印的页面范围，从开始到结束的页码

int from\_page = -1, to\_page = -1;

if (dlg.PrintAll())//打印文档中所有页面

{

from\_page = dlg.m\_pd.nMinPage;

to\_page = dlg.m\_pd.nMaxPage;

}

else if (dlg.PrintRange()) //打印文档中某一范围的页面

{

from\_page = dlg.GetFromPage();

to\_page = dlg.GetToPage();

}

else if (dlg.PrintSelection())//仅打印当前选择的页面

{

from\_page = to\_page = -1;//-1表示未知

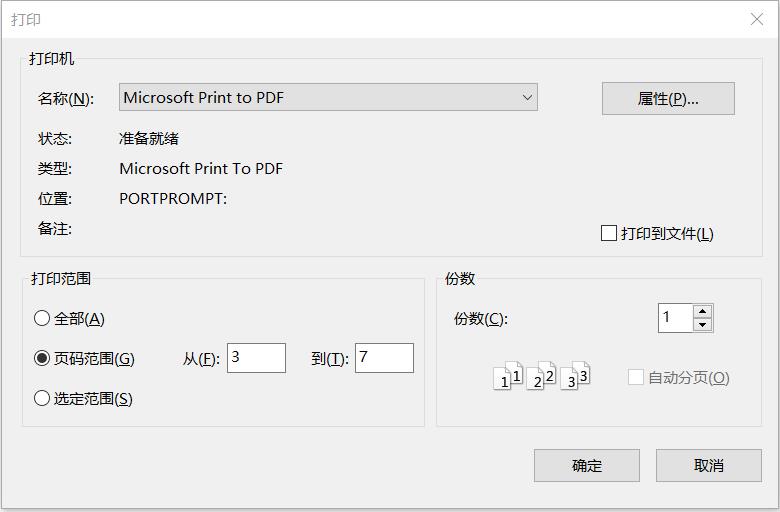
}

CString str;

str.Format(\_T("从%d打印到%d"), from\_page, to\_page);

AfxMessageBox(str);

}





### 3.14CPrintDialog::PrintAll

BOOL PrintAll() const;

函数功能：确定是否打印文档的所有页面。

返回值：若打印所有页面，则为非0，否则为0。

注：调用DoModal函数之后调用此函数确定是否打印文档中所有页面。

### 3.15CPrintDialog::PrintCollate

BOOL PrintCollate() const;

函数功能：确定是否打印副本。

返回值：如果用户在对话框中选择了逐份打印复选框则为非0，否则为0。

注：调用DoModal函数之后调用此函数确定打印机是否打印文档的所有副本。

例：显示打印对话框并选中逐份打印对话框，告知用户打印的份数。

//显示打印对话框并选中逐份打印对话框，告知用户打印的份数。

CPrintDialog dlg(FALSE, PD\_ALLPAGES | PD\_COLLATE | PD\_NOPAGENUMS |

PD\_HIDEPRINTTOFILE);

if (dlg.DoModal() == IDOK)

{

CString str;

//如果选中了逐份打印复选框，则GetCopies（）将返回

//打印的份数。否则，GetCopies（）总是返回1。

//然后，可以从打印设备的DEVMODE结构中找到打印的份数。

if (dlg.PrintCollate())

{

int num = dlg.GetCopies();

str.Format(\_T("打印的份数为%d\n"), num);

AfxMessageBox(str);

}

else

{

LPDEVMODE devmode = dlg.GetDevMode();

str.Format(\_T("Number of copies printed = %d\n"), devmode->dmCopies);

AfxMessageBox(str);

}

}



### 3.16CPrintDialog::PrintRange

BOOL PrintRange() const;

函数功能：确定是否只打印指定范围的页面。

返回值：如果用户只打印文档的一个页面范围则为非0，否则为0。

注：调用DoModal函数之后，调用此函数确定是否只打印文档的一个页面范围。

### 3.17CPrintDialog::PrintSelection

BOOL PrintSelection() const;

函数功能：确定是否只打印当前选择的项目。

返回值：如果用户只打印所选择项目则为非0，否则为0。

注：调用DoModal函数之后，调用此函数确定是否只打印当前选择的项目。

**五、资料来源**

## 1、CPen类参考资料

1、MSDN-CPen Class

<https://msdn.microsoft.com/en-us/library/czhcb94f.aspx#CPen::CPen>

2、百度百科-CPen

[http://baike.baidu.com/link?url=x1dv9AOCD79Tam1oWaVhr7lJgKnoKqjMY0N7iSDW5d prxx6Rr42E5amswICDmGBgqzqPgMoP-QKepbavLyNPsK](http://baike.baidu.com/link?url=x1dv9AOCD79Tam1oWaVhr7lJgKnoKqjMY0N7iSDW5d%09prxx6Rr42E5amswICDmGBgqzqPgMoP-QKepbavLyNPsK)

3、CSDN-CPen类

<http://blog.csdn.net/wuyuan2011woaini/article/details/7556123>

## 2、CColorDialog类参考资料

MSDN-CColorDialog Class

<https://msdn.microsoft.com/en-us/library/6w6cd538.aspx>

## 3、CFontDialog类参考资料

MSDN-CFontDialog Class

<https://msdn.microsoft.com/en-us/library/kck77523.aspx>

## 4、CPrintDialog类参考资料

MSDN-CPrintDialog Class

<https://msdn.microsoft.com/en-us/library/sk61115a.aspx>